

Development of Hyperplane-Based Models for Classification and Regression Problems

Ph.D. Thesis

By
Anuradha Kumari

under the supervision of
Prof. M. Tanveer



Department of Mathematics
INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025

Development of Hyperplane-based Models for Classification and Regression Problems

A THESIS

*submitted in partial fulfillment of the
requirements for the award of the degree
of*
DOCTOR OF PHILOSOPHY

By
Anuradha Kumari

under the supervision of
Prof. M. Tanveer



Department of Mathematics
INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2025



INDIAN INSTITUTE OF TECHNOLOGY INDORE

I hereby certify that the work which is being presented in the thesis entitled **Development of Hyperplane-based Models for Classification and Regression Problems** in the partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted in the **Department of Mathematics, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from **August 2021** to **May 2025** under the supervision of **Prof. M. Tanveer, Indian Institute of Technology Indore**.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Anuradha.
31/05/2025

Signature of the student with date

(Anuradha Kumari)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.



31/05/2025

Signature of thesis supervisor with date

(Prof. M. Tanveer)

Anuradha Kumari has successfully given her Ph.D. Oral Examination held on **May 30, 2025**.



31/05/2025

Signature of thesis supervisor with date

(Prof. M. Tanveer)

ACKNOWLEDGEMENTS

I would like to take this opportunity to sincerely thank everyone who has contributed to making this journey both enriching and memorable. Your support has made it not only a valuable learning experience but also a truly enjoyable one. First and foremost, I extend my sincere thanks to my supervisor, Prof. M. Tanveer, whose constant guidance and inspiration were invaluable throughout my research. His unwavering support, direction, and encouragement played a pivotal role in the completion of this work, helping me stay focused and motivated.

I am also deeply grateful to Prof. Ram Bilas Pachori and Dr. Mohd. Arshad, my research progress committee members, for their insightful feedback and constructive comments, which significantly enhanced the quality of my work. My sincere thanks to the Head of the Department of Mathematics for his continuous support and encouragement.

I would like to express my profound gratitude to the Director of the Indian Institute of Technology Indore for providing the research facilities and resources necessary for my work. I also acknowledge the financial support provided by the Council of Scientific and Industrial Research (CSIR), India, in the form of a fellowship during my Ph.D. program.

My heartfelt thanks go to Dr. Ashwani Kumar Malik and Dr. M.A. Ganaie for their continuous support and encouragement whenever needed. I am also thankful to my lab members, Mushir, Sajid, Ritik, Abdul, and Abdur, for their engaging discussions. I would also like to express my sincere appreciation to the departmental staff for their invaluable assistance and support. I am also grateful to the administrative and support staff of the institute for their assistance throughout my Ph.D. journey.

Lastly, I would like to express my deepest respect and love to my parents for their constant care and support. Special thanks go to my brothers, Bibhu and Ashish, my sister-in-law Sapna, my niece Anshi, and my friends, Anoop, Purna, Debdeep, Manasa, Rahul, Kulbhushan, Anup, Sheetal and Aniruddha, whose encouragement and belief in me made this thesis possible. Finally, I am thankful to everyone who, directly or indirectly, contributed to this work.

Dedicated
to
My Family and Friends

ABSTRACT

Support vector machines (SVMs), renowned for their robust mathematical and statistical underpinnings, offer optimal classification solutions through the construction of a single hyperplane that maximizes the margin between two classes. Despite their elegance, SVMs face a significant limitation: high computational complexity. This challenge led to the development of the twin support vector machine (TWSVM), a model that is approximately four times faster and employs two non-parallel hyperplanes to classify data more efficiently. Subsequently, least squares SVM (LSSVM) and least squares twin SVM (LSTSVM) introduced the use of least squares loss, further simplifying the computational burden by replacing quadratic programming problems with systems of linear equations. Despite these advancements, the sensitivity of hyperplane-based models to noise remains a challenge. To address this, researchers incorporated the pinball loss function into SVMs, enabling noise-insensitive learning. Expanding upon these advancements, we proposed the flexible pinball loss SVM (FP-SVM), which extends the versatility of the pinball loss function by broadening its parameter range. This enhancement enables FP-SVM to adapt more effectively to diverse data distributions, offering superior generalization and resilience against noise. Furthermore, we introduced the universum twin support vector machine with truncated pinball loss (Tpin-UTWSVM), which combines the sparsity-inducing properties of truncated pinball loss with the universum learning framework. This novel integration not only preserves the core advantages of pinball loss but also enhances classification accuracy by leveraging additional unlabeled universum data.

Traditional hyperplane-based models assume a uniform contribution from all training samples, an assumption that often leads to suboptimal solutions in the presence of noise and outliers. To address this limitation, intuitionistic fuzzy schemes (IFS) and dual-center-based intuitionistic fuzzy (DC-IF) weighting mechanisms are embedded within SVM and LSTSVM variants. These mechanisms underpin powerful models such as the intuitionistic fuzzy universum SVM (IFUSVM), dual-center-based intuitionistic fuzzy SVM (DC-IFSVM), and dual-center-based intuitionistic fuzzy least squares twin SVM (DC-IFLSTSVM), which demonstrate superior performance in handling noise, outliers, and class

imbalance. However, existing weighting techniques are constrained by high computational complexity and their inability to effectively handle data distributions where most samples are concentrated near class centers or boundaries. To address these shortcomings, we proposed novel fuzzy functions, class probability and generalized bell (CPGB) and class probability and bell-shaped (CPBS), which laid the foundation for two novel models: the class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM ID) and the bell-shaped fuzzy least squares twin support vector machine for imbalanced data (BS-FLSTSVM ID). Both models exhibit exceptional robustness to noise, outliers, and class imbalance, achieving state-of-the-art performance. As data scales rapidly in size, new challenges arise, necessitating models that balance computational efficiency with accuracy. To meet this need, we developed the proximity and distance scoring ensemble for universal representation (ProDiSE), a novel non-iterative pruning technique that intelligently selects representative data points. By significantly reducing computational overhead, ProDiSE enables large-scale data processing without compromising performance. Its integration with LSSVM results in ProDiSE-LSSVM, a scalable and efficient solution for massive datasets.

In one-class classification (OCC) tasks or anomaly detection, non-sparsity and the assumption of uniform contribution from each sample can hinder performance. Specifically, the least squares one-class SVM (LS-OCSVM) assumes a constant mean of the modeling error, which limits its robustness and ability to handle data distributions with varying error characteristics. To overcome this limitation, the robust least squares one-class SVM (RLS-1SVM) is proposed, which minimizes both the mean and variance of the modeling error, thus enhancing performance in the presence of non-Gaussian noise. Furthermore, the sparse RLS-1SVM (SRLS-1SVM) introduces sparsity to batch learning, marking the first sparse variant designed for this purpose. Finally, we leverage privileged information for regression by developing least square twin support vector regression with privileged information (LSTSVR+). This efficient model solves systems of linear equations, enhancing computational performance while maintaining predictive accuracy.

This thesis presents a comprehensive suite of hyperplane-based models addressing critical challenges in classification, regression, and OCC, including noise, outliers, class imbalance, and scalability, making significant contributions to the advancement of machine

learning techniques.

Keywords: Support vector machine, twin support vector machine, hyperplane-based models, intuitionistic fuzzy scheme, large-scale learning, one-class classification, privileged information.

List of Publications

Publications from the thesis

In Refereed Journals

1. **Anuradha Kumari**, Mushir Akhtar, M. Tanveer, Mohd. Arshad, Diagnosis of breast cancer using flexible pinball loss support vector machine, *Applied Soft Computing* (2024) 111454. doi:<https://doi.org/10.1016/j.asoc.2024.111454>.
2. **Anuradha Kumari**, M. Tanveer, Universum twin support vector machine with truncated pinball loss, *Engineering Applications of Artificial Intelligence* (2023) 106427. doi: <https://doi.org/10.1016/j.engappai.2023.106427>.
3. **Anuradha Kumari**, M. Tanveer, C.T. Lin, Class probability and generalized bell fuzzy twin SVM for imbalanced data, *IEEE Transactions on Fuzzy Systems* (2024). doi: 10.1109/TFUZZ.2024.3366936.
4. **Anuradha Kumari**, M. Tanveer, C.T. Lin, Bell-shaped fuzzy least square twin SVM with biomedical applications, *IEEE Transactions on Fuzzy Systems* (2024). doi: 10.1109/TFUZZ.2024.3421638.
5. **Anuradha Kumari**, M. Tanveer, LSTSVR+: Least square twin support vector regression with privileged information, *Engineering Applications of Artificial Intelligence* (2024), 108964. doi: <https://doi.org/10.1016/j.engappai.2024.108964>.
6. **Anuradha Kumari**, M. Tanveer, Enhancing robustness and sparsity: least squares one-class support vector machine, *Pattern Recognition* (2025) 111691. doi: <https://doi.org/10.1016/j.patcog.2025.111691>.

In Refereed Conferences

1. **Anuradha Kumari**, M.A. Ganaie, M. Tanveer, Intuitionistic fuzzy universum support vector machine, in: *International Conference on Neural Information Processing*,

Springer, 2022, pp. 236–247. doi: https://doi.org/10.1007/978-3-031-30105-6_20.

2. **Anuradha Kumari**, M.A. Ganaie, M. Tanveer, Dual center based intuitionistic fuzzy plane based classifiers, in: 2024 International Joint Conference on Neural Networks (IJCNN), 2024, pp. 1–8. doi:10.1109/IJCNN60899.2024.10650896

Under review

1. **Anuradha Kumari**, M. Tanveer, ProDiSE: Scalable sparse supervised learning with compact representation for large-scale data.

Publications other than thesis

In Refereed Journals

1. M. A. Ganaie, **Anuradha Kumari**, A. K. Malik, M. Tanveer, EEG signal classification using improved intuitionistic fuzzy twin support vector machines, Neural Computing and Applications (2024) vol. 36, pp. 163-179. doi:<https://doi.org/10.1016/j.engappai.2022.105151>. doi:10.1007/s00521-022-07655-x.
2. M. A. Ganaie, **Anuradha Kumari**, Anouck Girard, and Josephine Kasa-Vubu, M. Tanveer, Diagnosis of Alzheimer's disease via Intuitionistic fuzzy least squares twin SVM, Applied Soft Computing (2023) vol.149, 110899. doi:<https://doi.org/10.1016/j.asoc.2023.110899>.
3. **Anuradha Kumari**, Mushir Akhtar, Rupal Shah, M. Tanveer, Support matrix machine: A review, Neural Networks (2024) 106767. doi:<https://doi.org/10.1016/j.neunet.2024.106767>.

In Refereed Conferences

1. **Anuradha Kumari**, Mushir Akhtar, P.N. Suganthan, M. Tanveer, R^2VFL : A robust random vector functional link network with Huber-weighted framework, in: International Conference on Neural Information Processing, Springer, 2025.

Under review

1. M. Tanveer, M. Tabish, **Anuradha Kumari**, A.K. Malik, Weiping Ding, Support vector clustering uncovered: insights, challenges, and futures outlook.
2. **Anuradha Kumari**, M. Tanveer, P. N. Suganthan, Boosting RVFL networks: An outlier-resistant learning framework.
3. **Anuradha Kumari**, M. Tanveer, Fast and scalable restricted kernel machines: randomized Fourier and factorization approach.

4. Mushir Akhtar, **Anuradha Kumari**, Md. Sajid, Abdul Quadir, Mohd. Arshad, P.N. Suganthan, M. Tanveer, Towards robust and inversion-free randomized neural networks: the XG-RVFL framework

Under submission

1. **Anuradha Kumari**, A.K. Malik, M. Tanveer, P. N. Suganthan, Ensemble deep randomized neural network for anomaly detection.
2. **Anuradha Kumari**, M. Tanveer, Randomized neural network framework with a smooth, robust and adaptive loss function.
3. **Anuradha Kumari**, Ritik Mishra, M. Tanveer, Zephyr GAN: Redefining GAN with flexible gradient control.
4. Ritik Mishra, **Anuradha Kumari**, M. Tanveer, A kernelized geometric proximity framework for robust and imbalanced learning in restricted kernel machine.

Contents

List of Figures	ix
List of Tables	xix
List of Abbreviations	xx
1 Introduction	1
1.1 Background	2
1.2 Motivation	8
1.3 Objectives	9
1.4 Contribution of the thesis	10
1.5 Outline of the thesis	14
2 Literature Review	17
2.1 SVM and their variants	18
2.1.1 SVM: Formulation	18
2.1.2 LSSVM	19
2.1.3 Pin-SVM	20
2.1.4 Unified Pin-SVM	22
2.1.5 SVM with truncated pinball loss	23
2.1.6 Universum SVM	23
2.2 Twin SVM and its variants	25
2.2.1 TWSVM: Formulation	25
2.2.2 Least squares Twin SVM	26
2.2.3 Universum TWSVM	27

2.2.4	General TWSVM with pinball loss function (Pin-GTSVM)	28
2.3	Fuzzy theory	29
2.3.1	Intuitionistic fuzzy scheme (IFS)	29
2.3.2	SVM with IFN formulation:	30
2.3.3	Intuitionistic Fuzzy TWSVM (IFTWSVM)	31
2.3.4	Dual-center based intuitionistic fuzzy scheme	32
2.3.5	Affinity and class probability based fuzzy support vector machine for imbalanced datasets (ACFSVM)	34
2.3.6	Limitation of the existing weighting functions	35
2.4	Class imbalance learning	37
2.4.1	IFTWSVM for imbalanced data	38
2.4.2	Large scale fuzzy least squares TWSVM for CIL	39
2.5	Large scale training	40
2.5.1	Global-representation-based sparse LSSVM (GRS-LSSVM)	40
2.5.2	Limitation of globally representative point selection	42
2.6	OCC	42
2.6.1	LS-OCSVM	42
2.7	Regression	46
2.7.1	Support vector regression	46
2.7.2	SVR+	47
2.7.3	TSVR	48
2.8	Statistical test	49
2.8.1	Friedman's test	49
2.8.2	Wilcoxon signed rank test	51
2.8.3	Win-tie-loss sign test	51
3	Enhanced Pinball Loss Frameworks for Robust Classification and Generalization	53
3.1	Proposed flexible pinball loss SVM	54
3.1.1	Proposed flexible pinball loss	54

3.1.2	Formulation of proposed flexible pin-SVM	56
3.1.2.1	Convexity of proposed flexible pinball loss	59
3.1.2.2	Noise insensitivity	61
3.1.3	Experimental results	63
3.1.3.1	Experimental setup	63
3.1.3.2	Artificial dataset	64
3.1.3.3	Experimental outcomes and statistical comparison on UCI	
	and KEEL datasets	66
3.1.3.4	Parameter sensitivity analysis	73
3.1.3.5	Application on breakHis dataset	73
3.2	Proposed universum twin support vector machine with truncated pinball loss	75
3.2.1	Formulation of the proposed Tpin-UTWSVM	75
3.2.1.1	Linear case	75
3.2.1.2	Non-linear case	83
3.2.2	Properties of Tpin-UTWSVM	85
3.2.2.1	Noise insensitivity and sparsity	85
3.2.2.2	Complexity	89
3.2.3	Experiments	92
3.2.3.1	Experimental setup	92
3.2.3.2	Parameter selection	93
3.2.3.3	Experimental analysis on the binary datasets	94
3.2.3.4	Sensitivity of hyperparameters	97
3.2.3.5	Statistical analysis on the binary datasets	97
3.2.3.6	Confusion matrix of proposed Tpin-UTWSVM on UCI	
	datasets	100
3.2.4	Applications	102
3.2.4.1	EEG dataset and preprocessing	102
3.2.4.2	Experimental analysis over EEG data	115
3.2.4.3	AD dataset	116
3.2.4.4	Experimental analysis over AD data	116

3.3	Comparison between FP-SVM and Tpin-UTWSVM	117
3.4	Summary	118
4	Fuzzy Weight-Based Robust Classification: Addressing Noise, Outliers, and Class Imbalance	121
4.1	Proposed intuitionistic fuzzy universum SVM	123
4.1.1	Formulation of IFUSVM	123
4.1.2	Experimental results	125
4.1.2.1	Setup of the experiment	125
4.1.2.2	Results and discussion	126
4.2	Proposed dual center-based intuitionistic fuzzy models	129
4.2.1	Formulation of dual center based intuitionistic fuzzy classifiers	129
4.2.1.1	Linear case	130
4.2.1.2	Non-linear case	133
4.2.1.3	Optimization approach: Solving the proposed optimization problems	135
4.2.1.4	Time complexity	135
4.2.2	Numerical experiments	135
4.2.2.1	Experimental evaluation over KEEL datasets	137
4.2.2.2	Statistical analysis	138
4.2.3	Application	140
4.3	Proposed class probability and generalized bell fuzzy twin SVM for imbalanced data	143
4.3.1	Formulation of class probability and generalized bell fuzzy twin SVM for imbalanced data	143
4.3.1.1	Weight assignment to the training data points	144
4.3.1.2	Proposed CGFTSVM-ID: Linear case	147
4.3.1.3	Proposed CGFTSVM-ID: Non-linear case	149
4.3.1.4	Time complexity	151
4.3.2	Numerical experiments	152

4.3.2.1	Synthetic datasets	154
4.3.2.2	Real-world datasets	155
4.3.2.3	Sensitivity analysis of the proposed CGFTSVM-ID on	
	KEEL datasets	160
4.3.2.4	Statistical analysis over real-world KEEL datasets	166
4.3.2.5	Statistical analysis over real-world UCI datasets	167
4.3.2.6	Influence of parameters on the performance of the pro-	
	posed CGFTSVM-ID	168
4.3.2.7	Choice of steepness parameter r in the proposed CPGB	
	membership function	168
4.3.3	Application	169
4.3.3.1	Experimental discussion on AD dataset	169
4.4	Proposed bell-shaped fuzzy least square twin support vector machine for	
	imbalanced data	173
4.4.1	Formulation of proposed BSFLSTSVM-ID	174
4.4.1.1	Determination of weighting scheme	174
4.4.1.2	Linear case of proposed BSFLSTSVM-ID	177
4.4.1.3	Non-Linear case of proposed BSFLSTSVM-ID	179
4.4.1.4	Conjugate gradient method	180
4.4.2	Numerical experiments	181
4.4.2.1	Experimental setup	182
4.4.2.2	Experimental analysis on synthetic datasets	187
4.4.2.3	Experimental analysis on UCI and KEEL datasets	188
4.4.2.4	Parameter sensitivity	191
4.4.2.5	Statistical analysis	193
4.4.2.6	Experimental analysis on NDC datasets	196
4.4.3	Biomedical applications	197
4.4.3.1	Numerical experiments	198
4.5	Interpretability embedded in model design	200
4.6	Intuitive explanation of key mathematical notations	202

4.7 Summary	202
5 Breaking Complexity Barriers in Large-Scale Classification	205
5.1 Proposed work	206
5.1.1 Proximity and distance scoring ensemble for universal representation (ProDiSE)	206
5.2 Intuitive purpose of NNPR, PBS and DBS	210
5.3 RepScore	211
5.3.1 Proposed ProDiSE-LSSVM	213
5.4 Complexity	216
5.5 Numerical experiments	217
5.5.1 Experimental setup	217
5.5.2 Experimental results on category A datasets	221
5.5.3 Comparison of training time of the proposed ProDiSE-LSSVM and the baseline models	223
5.5.4 Sensitivity analysis of the proposed ProDiSE-LSSVM across various datasets	223
5.5.5 Impact of hyperparameters m and r on the performance of the proposed ProDiSE-LSSVM	224
5.5.6 Statistical analysis on category A datasets	226
5.5.7 Experimental results on category B datasets	226
5.5.8 Experiments over NDC dataset	228
5.6 Summary	229
6 Robust and sparse one-class classification framework	231
6.1 Proposed work	232
6.1.1 Proposed RLS-1SVM	233
6.1.2 Proposed SRLS-1SVM	237
6.1.3 Theoretical properties	239
6.1.4 Time complexity	245
6.2 Assessment of proposed models versus baseline approaches	246

6.2.1	Proposed models versus OCSVM	246
6.2.2	Proposed models versus LS-OCSVM	246
6.2.3	Proposed models versus ROCSVM-RHHQ	247
6.2.4	Proposed models versus MKCLS-OCSVM	247
6.3	Numerical experiments	248
6.3.1	Experimental setup	248
6.3.2	Experimental analysis	249
6.3.3	Win-tie-loss assessment	252
6.3.4	Impact of rank parameter r on proposed SRLS-1SVM	253
6.3.5	Performance of the models in the presence of outliers	253
6.4	Statistical analysis on real-world data	255
6.5	Application	257
6.5.1	Experimental setup	257
6.5.2	Experimental and statistical analysis	258
6.6	Summary	260
7	Leveraging Privileged Information in Regression Model	263
7.1	Proposed work	263
7.1.1	Proposed LSTSVR+: linear case	264
7.1.2	Proposed LSTSVR+: Non-Linear case	268
7.1.3	Complexity	271
7.1.4	Comparison between SVR+ and the proposed LSTVR-PI	271
7.1.5	Impact of PI in proposed LSTSVR+	273
7.1.6	Theoretical properties	275
7.2	Numerical experiments	279
7.2.1	Parameter selection	280
7.2.2	Artificially generated datasets	280
7.2.3	Real-world datasets	283
7.2.4	Statistical analysis	288
7.3	Application	290

7.3.1 Experimental analysis	293
7.4 Contribution	294
7.5 Summary	294
8 Conclusions and Future Works	297
8.1 Conclusions	299
8.2 Future research directions	305

List of Figures

1.1	Layout of the thesis	16
2.1	Representation of a training sample. Fig. (a) corresponds to a sample having majority of the data points lying on the boundary of class. Fig. (b) corresponds to a sample having majority of the data points lying around the center of the class.	36
2.2	Graphical illustration of membership function in equation (2.27).	36
3.1	Representation of the flexible pinball loss function with fixed $\tau_2 = 0.5$ and different values of τ_1 . The brown line depicts the case of $\tau_1 = -0.7 < -0.5$ which leads to non-convexity of the function as $\exists u_1$ and u_2 such that $L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) > \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2)$, $\beta \in [0, 1]$. . .	55
3.2	Flexible pinball loss for fixed $\tau_2 = 0.5$ and different τ_1 values between -0.5 and 1	55
3.3	Flexible pinball loss for fixed $\tau_2 = -0.5$ and different τ_1 values between 0.5 and 1	56
3.4	Bar graph representing the accuracy of models on an artificial dataset. . . .	65
3.5	Comparison of accuracies of baseline models and proposed FP-SVM. . . .	70
3.5	Comparison of accuracies of baseline models and proposed FP-SVM. . . .	71
3.6	Evaluation of the proposed FP-SVM's accuracy on UCI data while varying parameter σ , τ_2 and τ_2, τ_1	72
3.7	Block diagram to obtain the unknowns (w_1, b_1) of the first hyperplane in equation (3.17).	82
3.8	Bar graph illustrating	90

3.9 Bar graph representation to compare the performance of models over UCI datasets with different noise levels.	103
3.10 Bar graph representation to compare the training of models over UCI datasets with different noise levels.	104
3.11 Sensitivity evaluation of the proposed Tpin-UTWSVM classifier on UCI binary datasets using user-specified parameters and Gaussian kernel ($C_1 = C_2$).	108
3.12 Samples of EEG signal for different states.	112
3.13 Bar graph representation to compare the performance of models over EEG datasets with different noise levels.	113
3.14 MRI image of brain in three different planes	114
4.1 Sensitivity evaluation of the proposed IFUSVM classifier on UCI binary datasets using user-specified parameters and Gaussian kernel.	128
4.2 Box plot representation of results for different models over BreakHis dataset.	142
4.3 Graphical representation of gbell membership function of equation (4.29)	145
4.4 AUC representation of models over KEEL synthetic datasets.	155
4.5 Evaluation of AUC values of proposed CGFTSVM-ID while varying the parameters $\lambda_1, \lambda_2, \sigma$ and a	165
4.6 Representation of sensitivity metric of proposed CGFTSVM-ID and baseline models on UCI datasets.	170
4.7 (a) represents training time of ACFSVM and proposed CGFTSVM-ID. Fig (b)-(f) represents the effect of parameters on AUC values of the proposed CGFTSVM-ID over KEEL datasets cmc, monks1, monks3 and Yeast2vs8.	171
4.8 Representation of AUC for different models on ADNI datasets.	173
4.9 Illustration of bell-shaped function for different values of scale parameter λ while fixing width parameters a . Here, O corresponds to the class center.	175
4.10 Comparison of time taken for training of the proposed BSFLSTSVM-ID and baseline models over KEEL datasets with RBF kernel.	190

4.11 Evaluation of AUC values of proposed BSFLSTSVM-ID while varying the parameters C_1, C_2, σ and a .	191
4.12 Effect of variation in the parameters $C_1, C_2, \sigma, \lambda$ and a on the AUC values of proposed BSFLSTSVM-ID for synthetic dataset paw02a-800-7-60-BI.	192
4.13 Comparison of G-mean values of the proposed BSFLSTSVM-ID and base-line models on BreakHis datasets.	199
5.1 Illustration of radius of data points $t_i, t_j, t_k \in T$.	207
5.2 Representation of PBS of a training sample x_j based on equation (5.2). Pink data points represent the training samples, and black points represent samples of subset T . Sample x_j has radius $\mathcal{R}_{x_j}$. The radius of the nearest neighbour of x_j in T (denoted as $t_i \in T$) is given by $\mathcal{R}_{t_i}$. Fig. (a) illustrates $\text{PBS}(x_j) = M$ as $\mathcal{R}_{t_i} \leq \mathcal{R}_{x_j}$. Fig. (b) illustrates $\text{PBS}(x_j) = \text{NNPR}(t_i)$, where $\mathcal{R}_{t_i} > \mathcal{R}_{x_j}$.	208
5.3 Flow diagram of ProDiSE-LSSVM.	215
5.4 Demonstration of performance on category A datasets.	221
5.5 Comparison of training time (in seconds) for different datasets.	222
5.6 (a) Sensitivity analysis of the proposed ProDiSE-LSSVM with respect to different parameters on real-world datasets. (a) bank (b) musk_2 (c) ozone (d) titanic.	224
5.7 Fig. (a)-(c) represent the accuracy of datasets by varying the number of SVs (m). (a) BCW (b) banknote (c) blood. Figure (d) illustrates the impact of n on the performance of the model.	225
5.8 Demonstration of performance on category B datasets.	228
6.1 Sensitivity analysis of the proposed SRLS-1SVM by varying the rank parameter r across four benchmark datasets. The datasets include (a) breast_cancer (b) congressional_voting (c) haberman_survival (d) ilpd_indian_liver.	254

6.2	The testing performances of the four baseline models and proposed RLS-1SVM by varying ratios of outliers occupying the entire training set across four benchmark datasets. The datasets include (a) breast_cancer (b) congressional_voting (c) haberman_survival (d) ilpd_indian_liver.	255
6.3	F-measure representation of models over different cases of CIFAR-10 dataset.	258
7.1	Graphical view to represent the flow of algorithm for the non-linear proposed LSTSVR+.	270
7.2	A framework diagram illustrating the proposed LSTSVR+ applied to Function 1 with Type I noise using two samples.	282
7.3	Performance of baseline models and proposed LSTSVR+ across artificially generated datasets on various types of noise.	284
7.4	Comparison of RMSE values across different datasets for various models using bar graph.	285
7.5	Influence of parameters on the RMSE values of the proposed LSTSVR+ across different UCI datasets.	286

List of Tables

3.1 Performance comparison of the models on a 2-D artificial dataset using RBF kernel.	65
3.2 Classification performance of the proposed FP-SVM and baseline models on UCI and KEEL binary datasets with RBF kernel.	66
3.3 Classification performance of the proposed FP-SVM on UCI and KEEL binary datasets with different kernel functions.	67
3.4 Win-tie-loss of row method over column method on UCI and KEEL datasets. Here, in [a/b/c], a, b and c correspond to number of wins, ties and losses of row method over column method, respectively.	71
3.5 Nemenyi post hoc significant difference for the proposed FP-SVM with respect to the baseline models on UCI and KEEL datasets.	71
3.6 Wilcoxon-signed rank test with respect to the proposed FP-SVM on UCI and KEEL datasets.	72
3.7 Classification performance of proposed FP-SVM and baseline models on BreakHis dataset with RBF kernel.	74
3.8 Comparison of the models under study.	91
3.9 Performance of the proposed Tpin-UTWSVM model with baseline models on UCI binary datasets with Gaussian kernel.	95
3.9 Continued from previous page-Performance of the proposed Tpin-UTWSVM model with baseline models on UCI binary datasets with Gaussian kernel.	96
3.10 Training time and optimal hyperparameters of models on binary datasets with Gaussian kernel.	98
3.10 Continued from previous page-Training time and optimal hyperparameters	

of models on binary datasets with Gaussian kernel.	99
3.11 Pairwise win-tie-loss of the proposed Tpin-UTWSVM model with respect	
to baseline models over UCI datasets. With noise level r in the column,	
comparison of row method with the proposed Tpin-UTWSVM. Here, . . .	100
3.12 blood	101
3.13 breast-cancer	101
3.14 breast-cancer-wisc	101
3.15 breast-cancer-wisc-prog	101
3.16 conn-bench-sonar-mines	101
3.17 haberman-survival	101
3.18 heart-hungarian	101
3.19 heart-switzerland	101
3.20 hepatitis	101
3.21 ionosphere	101
3.22 molec-biol-promoter	101
3.23 monks-1	101
3.24 parkinsons	101
3.25 pima	101
3.26 planning	102
3.27 spect	102
3.28 echocardiogram	102
3.29 statlog-australian-credit	102
3.30 pittsburg-bridges-T-OR-D	102
3.31 Performance of proposed Tpin-UTWSVM model with baseline models on	
EEG datasets with Gaussian kernel.	105
3.31 Continued from previous page-Performance of proposed Tpin-UTWSVM	
model with baseline models on EEG datasets with Gaussian kernel.	106
3.31 Continued from previous page-Performance of proposed Tpin-UTWSVM	
model with baseline models on EEG datasets with Gaussian kernel.	107
3.31 Continued from previous page-Performance of proposed Tpin-UTWSVM	

model with baseline models on EEG datasets with Gaussian kernel.	108
3.32 Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.	109
3.32 Continued from previous page-Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.	110
3.32 Continued from previous page-Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.	111
3.32 Continued from previous page-Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.	112
3.33 Pairwise win-tie-loss of the proposed Tpin-UTWSVM with respect to baseline models over EEG datasets. With noise level r in the column, comparison of row method with the proposed Tpin-UTWSVM model. Here, [a,b,c] indicates that the proposed Tpin-UTWSVM technique wins 'a' number of times, ties 'b' number of times, and loses 'c' number of times in comparison to the row model.	114
3.34 Performance of proposed Tpin-UTWSVM model with baseline models on AD datasets with Gaussian kernel.	114
4.1 Classification performance of different models on binary datasets having Gaussian kernel	127
4.2 Win-tie-loss of the models in pair with proposed IFUSVM.	128
4.3 Abbreviation (abbre.) of datasets.	136
4.4 AUC with the time required for training of the proposed DC-IFSVM, DC-IFLSTSVM and baseline models over KEEL datasets with RBF kernel. . .	136
4.5 AUC with the time required for training of the proposed DC-IFSVM, DC-IFLSTSVM and baseline models over UCI datasets with RBF kernel.	139
4.6 Wilcoxon signed rank test of the proposed DC-IFLSTSVM with respect to other models. ✓ and ✗ denote null hypothesis (Hyp*) is rejected and not rejected, respectively.	140
4.7 AUC with the time required for training of the proposed DC-IFSVM, DC-	

IFLSTSVM and baseline models over BreakHis datasets with RBF kernel.	141
4.8 Details of the synthetic data used.	156
4.9 AUC with training time, sensitivity, specificity, G-mean and F-measure value of the proposed CGFTSVM-ID and baseline models on synthetic KEEL datasets.	157
4.10 Optimal parameters corresponding to AUC values of the proposed CGFTSVM-ID and baseline models on synthetic KEEL datasets.	158
4.11 Details of the real-world data used.	159
4.12 The AUC value with training time, sensitivity and specificity value of the proposed CGFTSVM-ID and baseline models on KEEL datasets.	160
4.13 The average AUC, average rank, average G-mean, average F-measure, average sensitivity, average specificity and average training time of the proposed CGFTSVM-ID and baseline models on KEEL and UCI datasets.	161
4.14 The performance evaluation of the proposed CGFTSVM-ID and baseline models in terms of win-tie-loss. The notation [win-tie-loss] indicates the count of wins, ties, and losses of the proposed CGFTSVM-ID over the respective column model.	161
4.15 AUC with training time, sensitivity, specificity, G-mean and F-measure of the proposed CGFTSVM-ID and baseline models on UCI datasets.	162
4.16 Optimal parameters corresponding to AUC values of the proposed CGFTSVM-ID and baseline models on UCI datasets.	163
4.17 The performance evaluation of the proposed CGFTSVM-ID and baseline models in terms of win-tie-loss. The notation [win-tie-loss] indicates the count of wins, ties, and losses of the proposed CGFTSVM-ID over the respective column model.	164
4.18 Nemenyi post hoc significant difference between the baseline column models and the proposed CGFTSVM-ID across UCI and KEEL datasets.	164
4.19 Wilcoxon-signed rank test of proposed CGFTSVM-ID with respect to baseline models on KEEL and UCI datasets.	164
4.20 The AUC value with training time, sensitivity and specificity of the models	

on ADNI datasets.	172
4.21 G-mean and F-measure of the proposed CGFTSVM-ID and baseline models on ADNI datasets.	172
4.22 Optimal parameters corresponding to AUC value of the proposed CGFTSVM-ID and baseline models on ADNI datasets.	172
4.23 AUC with training time, F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models over synthetic datasets with RBF kernel.	183
4.24 AUC with training time (in seconds), F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models over UCI datasets with RBF kernel.	184
4.25 AUC with training time, F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models over KEEL datasets with RBF kernel.	185
4.26 Effect of noise and outliers on datasets.	186
4.27 Wilcoxon signed rank test for the proposed BSFLSTSVM-ID in comparison to the baseline models.	187
4.28 Win-tie-loss sign test for the proposed BSFLSTSVM-ID with respect to the baseline models.	187
4.29 AUC with training time, F-measure and G-mean for the proposed BSFLSTSVM-ID in comparison to baseline models over NDC datasets having RBF kernel.	188
4.30 AUC with training time (in seconds), F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models on BreakHis and ADNI datasets with RBF kernel.	194
5.1 Accuracy, training time (in seconds) and testing time (in seconds) of the proposed ProDiSE-LSSVM and baseline models on category A datasets.	220
5.2 Optimal parameters corresponding to maximum accuracy for baseline and proposed ProDiSE-LSSVM across category A datasets.	221

5.3	Wilcoxon signed rank test of the baseline models with respect to the proposed ProDiSe-LSSVM.	227
5.4	Accuracy, training time (in seconds) and testing time (in seconds) of the proposed ProDiSe-LSSVM and baseline models on the category B datasets.	227
5.5	Training time and testing time (in seconds) of the proposed ProDiSe-LSSVM and baseline models on NDC datasets.	227
6.1	Details of the datasets.	250
6.2	Average G-mean with standard deviation (std), training time (Tr) and testing (Ts) time (in seconds) of the proposed RLS-1SVM, SRLS-1SVM, and the baseline models on UCI datasets (bold values represent the best G-mean values).	251
6.3	Win-tie-loss of the baseline column models with respect to the proposed RLS-1SVM and SRLS-1SVM across UCI datasets. The representation is in the form of (win,tie,loss).	252
6.4	Nemenyi post-hoc significant difference between the baseline column models and the proposed RLS-1SVM and SRLS-1SVM across UCI datasets.	256
6.5	Wilcoxon signed rank test of the column models with respect to the proposed RLS-1SVM and SRLS-1SVM across UCI datasets.	256
6.6	Average G-mean with standard deviation, training (Tr) time and testing (Ts) time (in seconds) of the proposed RLS-1SVM, SRLS-1SVM, and the baseline models on CIFAR-10 dataset (bold values represent the best G-mean values).	258
6.7	Win-tie-loss of the baseline column models with respect to the proposed RLS-1SVM and SRLS-1SVM over CIFAR-10 dataset.	259
6.8	Nemenyi post-hoc significant difference between the baseline column models and the proposed RLS-1SVM and SRLS-1SVM over CIFAR-10 dataset.	259
6.9	Wilcoxon signed rank test of the baseline column models with respect to the proposed RLS-1SVM and SRLS-1SVM over the CIFAR-10 dataset.	260
7.1	Different functions to generate artificial datasets.	281

7.2	RMSE values along with SSE, SSE/SST and training time (in seconds)	
	across the artificially generated synthetic datasets with different types of	
	noise for TSVR, TWSVR, LSTSVR, TSVQR, SVR+ and the proposed	
	LSTSVR+.	283
7.3	RMSE values along with SSE, SSE/SST and training time (in seconds)	
	across the real-world datasets for TSVR, TWSVR, LSTSVR, TSVQR,	
	SVR+ and the proposed LSTSVR+.	287
7.3	Continued from previous page-RMSE values along with SSE, SSE/SST and	
	training time (in seconds) across the real-world datasets for TSVR, TWSVR,	
	LSTSVR, TSVQR, SVR+ and the proposed LSTSVR+.	288
7.4	Nemenyi post hoc significant difference for the proposed LSTSVR+ with re-	
	spect to the baseline models on artificially generated datasets corresponding	
	to different metrics.	289
7.5	Nemenyi post hoc significant difference for the proposed LSTSVR+ with	
	respect to the baseline models on real-world datasets corresponding to dif-	
	ferent metrics.	290
7.6	RMSE values along with SSE, SSE/SST and training time (in seconds)	
	across the real-world time-series datasets for TSVR, TWSVR, LSTSVR,	
	TSVQR, SVR+ and the proposed LSTSVR+.	291
8.1	Comparison of different proposed models.	298

List of Abbreviations

IF	Intuitionistic fuzzy
IFN	Intuitionistic fuzzy number
DC-IF	Dual center based intuitionistic fuzzy
SRM	Structural risk minimization
ERM	Empirical risk minimization
CIL	Class imbalance learning
gbell	Generalized bell
SVM	Support vector machine
LSSVM	Least squares support vector machine
SVR	Support vector regression
TWSVM	Twin support vector machine
LSTSVM	Least squares twin support vector machine
QPP	Quadratic programming problem
ML	Machine learning
k-NN	k-nearest neighbour
AI	Artificial intelligence
SVs	Support vectors
OCC	one-class classification
OCSVM	one-class support vector machine
PI	Privileged information
SOR	Successive overrelaxation
K.K.T.	Karush Kuhn Tucker
ML	Machine learning
QPP	Quadratic programming problem

Chapter 1

Introduction

Artificial intelligence (AI) aims to develop machines capable of exhibiting human-like intelligence. As a rapidly advancing field, AI holds immense potential to drive significant advancements across various domains [1]. This growth has been largely propelled by developments in machine learning (ML) algorithms, which have brought transformative changes to society. Among the many contributions in the literature, numerous classification models have been proposed [2], leveraging principles from both computer science and mathematics to address classification challenges effectively.

In ML, several models such as support vector machines (SVMs) [3], artificial neural networks (ANNs) [4], and randomized neural networks (RdNNs) [5] have been designed to address classification tasks. Classification involves mapping an input variable (x) to a discrete (y) using a learning algorithm (g). The learning function is represented as $y = g(x, \theta)$, where θ denotes the algorithm's parameters, and $y \in Y$, where Y contains the possible class labels. For a discrete set Y , classification problems can be categorized as binary or multiclass, depending on the number of classes in Y . Conversely, if Y takes continuous values, the task corresponds to a regression problem.

The primary focus of this thesis is the development and enhancement of hyperplane-

based models, specifically SVMs, to address both classification and regression problems. This research explores the theoretical advancements and practical implementations of SVM-based models to improve their performance, scalability, and adaptability.

1.1 Background

Among the machine learning (ML) approaches, SVM [3] is a popular and efficient technique for classification and regression problems. It has a strong mathematical background and leads to a unique optimal solution. The objective is to find the optimal hyperplane by solving quadratic programming problem (QPP), with a maximal margin between the two classes. By considering the maximum margin approach, SVM imposes structural risk minimization (SRM) principle. It has wide applications in various domains, such as tea identification [6], detection of dendritic spines [7], classification of EEG signals [8], and so on. For regression tasks, SVM is commonly referred to as support vector regression (SVR) [9]. SVR has different applications such as forecasting holiday tourist flow [10], electric loads [11], predicting dam structural behaviour [12] and so forth. The distance between the optimal hyperplane and the nearest data points can be significantly affected by feature noise, making SVM-based classifiers sensitive to such noise and unstable under re-sampling [13]. Several initiatives have attempted to overcome the limitations of SVM, such as using class center vectors to build SVM [14], robust SVM with least median loss penalty [15], a weighted SVM for data classification [16], and so on. One of the remarkable algorithms to overcome the feature noise is pin-SVM introduced by Huang *et al.* [17]. Pin-SVM uses the pinball loss function for the classification problem, where the pinball loss is given by $L_{\text{pin}}(u) = \max\{u, -\tau u\}$, with $\tau \in [0, 1]$ and $u \in \mathbb{R}$. Pinball loss penalizes both correctly and incorrectly classified data points, and regulates the penalty for correctly classified samples by choosing the value of τ . Despite numerous advantages, SVM suffers from high time

complexity $O(l^3)$ [18], where l denotes total training samples.

To overcome the limitation of high complexity of SVM, multiple approaches have been proposed such as sequential minimal optimization (SMO) [19], SVMlight [20], and generalized eigenvalue proximal SVM (GEPSVM) [21]. Motivated by the idea of non-parallel hyperplanes in GEPSVM [21], twin support vector machine (TWSVM) [22] is introduced which is approximately four times faster than classical SVM. TWSVM solves two smaller-sized QPPs instead of one large-sized QPP resulting in better efficiency over SVM. Though, TWSVM is efficient it does not consider SRM. Twin bounded support vector machine (TBSVM) [23] incorporates the SRM principle in the optimization problem of TWSVM and solves two QPPs. Further, coordinate descent margin based TWSVM [24] is an efficient classification model to handle large datasets. Inference with universum (USVM) [25] introduced universum data having prior information to improve the generalization performance of SVM. The twin variants [26, 27, 28] incorporating universum data have improved performance with less complexity as compared to USVM.

To reduce computational complexity, least squares support vector machine (LSSVM) [29] solves a system of linear equations rather than a QPP. By using a least squares loss function instead of the hinge loss, LSSVM constructs parallel hyperplanes, offering a more computationally efficient alternative to traditional SVM. LSSVM has demonstrated several real-world applications like safety monitoring [30], text classification [31] and so on. Van Gestel et al. [32] underscores that the generalization capabilities of SVM and LSSVM are comparable. Following the least squares idea of LSSVM, Kumar and Gopal [33] proposed least squares twin support vector machine (LSTSVM). It constructs two non-parallel hyperplanes by solving two distinct linear systems of equations. However, LSSVM has a notable limitation: it lacks sparsity. Unlike SVM, which utilizes only the SVs, LSSVM treats all training points as SVs, leading to higher memory demands for large datasets. This

limitation poses challenges for LSSVM in efficiently managing large-scale problems.

Sparse LSSVM [34] provides significant advantages, including increased memory efficiency and faster computation, especially beneficial for large datasets. Researchers have achieved sparsity through pruning SVs or approximating classification functions. Additional research has refined pruning models, such as pruning error minimization in LSSVM [35, 36] and SMO-based pruning [37]. However, these models often struggle to achieve a substantial reduction in SVs. Moreover, these pruning techniques involve a considerable computational burden, as they require repeatedly solving a progressively smaller system of linear equations. As a result, while these models improve test speed, they are generally insufficient for efficient training on large-scale datasets.

Several sparse variants of LSSVM have been proposed by modifying the regularization term. Shao et al. [38] introduced the sparse primal and dual LSSVM (SPDLSSVM) model, which combines both sample and feature selection within a unified sparse primal and dual LSSVM framework. Another development, the jointly sparse LS-SVM (JS-LSSVM) [39], incorporates the $L_{2,1}$ -norm to achieve joint sparsity and feature selection. Tang and Li [40] extended the L_0 norm by employing feature selection and utilizing the alternating direction method of multipliers (ADMM) for training. Most pruning techniques rely heavily on iterative processes, which tend to incur substantial computational and memory costs. Another significant aspect of these techniques is the reliance on the relationship between SVs and the decision hyperplane to guide pruning decisions [41]. However, determining this hyperplane in advance can require considerable resources, and many approaches overlook the practical limitations on the number of SVs retained in sparse LSSVM, which is often constrained in real applications. To address these limitations, Ma et al. [42] introduced a noniterative pruning approach, GRS-LSSVM, incorporating a quantitative constraint on the number of preserved SVs. This model achieves global representation by evaluating both point density

and dispersion in feature space, allowing the model to maintain essential SVs without iterative refinement. This noniterative strategy enables an efficient path to achieving sparse LSSVM.

Learning from an imbalanced dataset and addressing the noise and outliers present in the training dataset are the two primary challenges related to classification problems. For the datasets having two classes, imbalanced datasets have one-class in abundance (majority class) as compared to the other class (minority class). There are several real-world domains such as face recognition [43], where imbalanced datasets can be seen. To ensure good performance of a model for imbalance learning with noise and outliers, various SVM variants have been proposed where the contribution of input data depends on membership functions, which lessen the influence of outliers and noise on classification technique. Fuzzy SVM (FSVM) [44] is proposed to handle the outliers and noise, where the assignment of fuzzy value depends on the distance from class center. The assignment of membership to the input data is the key point in FSVM variants. Though fuzzy concept overcomes the effect of noise/outliers, it still suffers from CIL issue. To handle the class imbalance problem in the presence of outliers and noise, Batuwita and Palade [45] proposed fuzzy SVM for imbalance data (FSVM-CIL) with four different membership functions. In FSVM-CIL, the hyperplane-based membership function assumes initially obtained hyperplane is an accurate prediction of the final hyperplane, which is not always the case. To overcome the limitations of the aforementioned model for CIL in the presence of the noise/outliers effectively, Tao et al. [46] introduced affinity and class probability-based FSVM (ACFSVM) designed for imbalanced datasets which uses support vector data description (SVDD) [47] and class probability to obtain the membership value of the data points. An alternative approach to tackle the challenge posed by outliers and noise in the data involves employing the large-scale pinball TSVM[48].

Similar to SVM, many twin variants have been proposed to overcome the impact of outliers and noise. Rezvani et al. [49] introduced intuitionistic fuzzy TSVM (IFTWSVM) by incorporating the intuitionistic fuzzy number (IFN) in TSVM model. It assigns membership and non-membership values to each sample which is used to calculate the score values of data points. Thus, IFTWSVM has diminished effect of noise/outliers for classification problems. However, IFTWSVM encounters computational challenges for large datasets due to the computation of matrix inverses, making it intractable in such scenarios. Improved IFTWSVM (IIFTWSVM) [50] overcomes the requirement of matrix inversion and has resistance against noise and outliers. To deal with the CIL problem along with noise/outliers, intuitionistic fuzzy TSVM for imbalanced data (IFTWSVM-ID) [51] is proposed, where a weighting strategy is given to the data points to tackle the class imbalance issue.

In pattern recognition and ML domains, one-class classification (OCC) [52] is considered a distinctive classification task. The challenge in OCC is to find samples among the data that deviate from a general pattern [53]. OCC is also known as an anomaly detection problem. In contrast to traditional binary and multi-class classification problems, OCC only requires data about one of the numerous classes that potentially make up the input space called the target class. Conversely, outliers can be isolated oddities or they could belong to any class, known as the non-target class. There are various real-world examples which constitute the OCC problem such as sea surface target detection [54], anomaly detection [55], and so on. Numerous models have been introduced to handle the OCC problem [56]. Among these models, researchers have focused more on SVM based techniques because of their kernel learning capacity, and generalization performance [57, 58]. It is equivalent to modeling a system, where only the target class is employed for training, and testing is carried out on datapoints belonging to the target or non-target class. Support vector data description (SVDD) [59] and one-class support vector machine (OCSVM) [60] are two

one-class classifiers that are frequently employed.

For regression, SVR [61] is a widely recognized hyperplane-based technique. Peng et al. [62] introduced twin support vector regression (TSVR) which akin to TWSVM, employs non-parallel hyperplanes and establishes ϵ -insensitive lower and upper bound functions by addressing two QPPs, with each QPP being smaller than that of SVR. Different variants of TSVR are introduced such as ϵ -TSVR [63], weighted TSVR [64], twin projection SVR [65], and so on. The aforementioned variants solve a pair of QPPs which has high time and memory complexity. To overcome the high computational burden twin least squares SVR (TLSSVR) [66] is introduced. TLSSVR replaces the inequality constraints of TSVR with equality constraints, leading to the solution of a system of linear equations to train the model. Further different least squares variants of TSVR are proposed, such as primal least squares TSVR (PLSTSVR) [67], and sparse LSTSVR (FS-LSTSVR) [68]. Recently, Huang et al. [69] provided an overview of TSVR, which comprehensively covers different variants of the twin regression approach.

Vapnik and Vashist [70] introduced a novel learning paradigm known as learning using privileged information (LUPI), to improve the generalization performance of learning models. In the conventional supervised learning paradigm, only the training data is available in the training stage. However, LUPI deviates from this norm by providing additional information for the training data during the training stage. This supplementary information is referred to as privileged information (PI). Importantly, this PI is not available during the test stage. The utilization of PI has demonstrated notable performance enhancements in various domains of ML, such as OCC [71], and multi-view learning [72]. However, within the realm of regression, the advantages conferred by PI have primarily been observed in the context of SVR, which inherently carries a high computational complexity.

1.2 Motivation

The variations in the pinball loss parameter τ , as explored by Huang et al. [73], have shown significant improvements in generalization performance. This suggests that expanding the range of pinball loss parameters could enhance the adaptability of classification models, allowing them to better handle diverse datasets. Moreover, integrating pinball loss into SVM models often compromises sparsity, a key factor for maintaining efficiency, particularly with large-scale datasets. To address this, the truncated pinball loss (Tpin-SVM) [74] is introduced, effectively balancing sparsity with noise insensitivity. These advancements point to two key motivations for further research: first, to refine the pinball loss function to extend its parameter range and improve model adaptability, and second, to develop a novel non-parallel hyperplane-based framework that preserves sparsity while expanding the applicability of pinball loss. These works aim to enhance model performance while preserving the theoretical advantages of pinball loss.

The presence of noisy and outlier-laden data during model training can lead to a decline in the model's ability to generalize effectively. Another significant concern affecting the training samples is class imbalance, where the number of samples in one of the classes (majority class) is greater than in the other class (minority class). In models utilizing fuzzy concepts, the design of the weighting function becomes crucial, as it influences the algorithm by determining the contribution of each sample, particularly in constructing the hyperplane. However, existing weighting functions designed to handle noise, outliers, and class imbalance learning (CIL) have notable limitations. For instance, the membership function often proves ineffective when the data distribution is such that the majority of samples lie at the class boundary or center. This limitation has motivated the development of novel weighting techniques to better address these challenges.

Further, LSSVM, despite its computational efficiency in solving linear systems of equations, suffers from a lack of sparsity, making it impractical for large-scale datasets. To address this limitation, several sparse LSSVM variants based on pruning technique, have been proposed, such as GRS-LSSVM [42]. However, these models exhibit certain limitations to scalability and effectiveness. This underscores the need for a scalable sparse LSSVM model to handle large-scale datasets efficiently.

In addition to binary classification, the OCC is highly sensitive to noise and outliers, which can degrade its performance. While various loss functions have been proposed to mitigate this issue, they often introduce additional complexity to the optimization problem. To address these challenges, a novel robust and sparse model for OCC is proposed, aimed at improving the performance of LS-OCSVM in the presence of noise and outliers.

The utilization of PI has demonstrated notable performance enhancements in various domains of ML, such as OCC [75], and multi-view learning [72]. However, within the realm of regression, the advantages conferred by PI have primarily been observed in the context of SVR, which inherently carries a high computational complexity. Motivated by the observed performance gains across diverse ML domains facilitated by PI, coupled with the desire for reduced computational complexity than SVR+ [70], PI can be employed in regression model.

1.3 Objectives

The objectives of the thesis are as follows:

- [1] To improve the adaptability of classification model by extending the pinball loss while retaining its core benefits, and to enhance sparsity in non-parallel hyperplane-based models using pinball loss while preserving its theoretical advantages.

- [2] To design robust SVM and TWSVM models by utilizing novel weighting schemes that effectively address challenges posed by noise, outliers, and class imbalance.
- [3] To develop a novel pruning technique that selects a representative subset of training data, ensuring diverse coverage of the dataset's distribution, for scalable and efficient large-scale data classification.
- [4] To develop a robust and sparse model for one-class classification by incorporating data distribution information and applying Nystrom approximation to LS-OCSVM.
- [5] To leverage privileged information for regression tasks using least squares twin SVR and develop an efficient and effective regression model.

1.4 Contribution of the thesis

In this Section, we present the detailed contribution of the thesis, which are as follows:

- [1] Utilizing pinball loss variants in two ways to address challenges in classification tasks. First, a novel flexible pinball loss function is introduced as an extension of the traditional pinball loss, allowing the parameters τ_1 and τ_2 to range from $[-1, 1]$ while preserving convexity. The integration of the flexible loss function into the SVM framework leads to the development of the flexible pinball loss SVM (FP-SVM) [76]. The proposed FP-SVM serves as a generalization of several prevalent SVM-type models. In particular, it reduces to C-SVM [77] for $\tau_1 = 1$ and $\tau_2 = 0$, pin-SVM [17] for $\tau_1 = 1$ and $\tau_2 \in [0, 1]$. Further, it reduces to unified pin-SVM [78] for $\tau_1 = 1$ and $\tau_2 \in [-1, 1]$. Second, incorporation of the truncated pinball loss with TWSVM along with universum data leads to the introduction of universum twin support vector machine with truncated pinball loss (Tpin-UTWSVM). The proposed Tpin-UTWSVM

considers quantile distance and punishes both correctly and incorrectly classified samples. Penalizing correctly classified samples lead to noise insensitivity and imposition of equal penalty to the correctly classified data points lead to sparsity of the model. Thus, the proposed model maintains a trade-off between noise insensitivity and sparsity of the model along with preserving the theoretical properties of pinball loss. The proposed Tpin-UTWSVM includes universum data points which give prior information about the distribution of data and helps in improving the generalization performance of the model. The objective function obtained for the proposed Tpin-UTWSVM is non-differentiable and non-convex. Thus, concave convex procedure (CCCP) is applied to solve the problem. To train the proposed Tpin-UTWSVM, efficient successive overrelaxation (SOR) method is used.

- [2] The uniform contribution of all training data points in constructing the hyperplane may result in a suboptimal hyperplane, particularly in the presence of noise or outliers. To address this issue, we incorporated the concept of IFS into SVM, along with universum data, leading to the development of the intuitionistic fuzzy universum support vector machine (IFUSVM). This approach effectively mitigates the influence of noise and outliers during model training. However, it does not address the significant challenge of CIL, which can adversely affect the learning process. This limitation is overcome by dual center based intuitionistic fuzzy plane based classifiers. which gives rise to dual center-based intuitionistic fuzzy SVM (DC-IFSVM) and its least-squares variant, DC-IFLSTSVM [79]. Furthermore, we propose two novel membership functions to address the adverse impacts of noise, outliers, and CIL. First, class probability and generalized bell (CPGB) and second, class probability and bell-shaped (CPBS) to solve imbalance learning problems having noise and outliers which assign weights to individual samples to decide their influence in the construction of optimal plane.

By integrating these functions with TWSVM and LSTSVM, two innovative models are introduced, which are class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM-ID) and bell-shaped fuzzy least square twin support vector machine for imbalanced data (BSFLSTSVM-ID), respectively, offering robust solutions to these challenges.

[3] To address the challenges of training ML models on large-scale datasets, this thesis proposes a novel pruning technique called ProDiSE (proximity and distance scoring ensemble for universal representation). ProDiSE identifies a subset of representative data points from the training set, ensuring that the selected points comprehensively and accurately capture the dataset's underlying distribution. This approach preserves data integrity while reducing the computational burden associated with large-scale training. ProDiSE is seamlessly integrated into the framework of LSSVM, resulting in a sparse and efficient variant named ProDiSE-LSSVM. This combined framework delivers a scalable and efficient solution for datasets containing millions of samples. ProDiSE-LSSVM mitigates the inefficiencies of traditional models, achieving faster training times and improved performance without compromising classification accuracy.

[4] Real-world data often contains non-Gaussian noise arising from measurement, operational, sampling, and modeling errors, which complicates anomaly detection. Traditional OCC methods, such as the least squares one-class support vector machine (LS-OCSVM) [80], lack the capability to handle these issues effectively. To overcome these limitations, we introduce two novel models that redefine the OCC paradigm: the robust least squares one-class support vector machine (RLS-1SVM) and the sparse robust least squares one-class support vector machine (SRLS-1SVM). The RLS-1SVM

leverages the distributional information of the data by minimizing both the mean and variance of modeling errors, ensuring reliable performance in the presence of random noise. Taking a step further, the SRLS-1SVM pioneers the introduction of sparsity into the LS-OCSVM framework. By applying the representer theorem and pivoted Cholesky decomposition [81], the SRLS-1SVM achieves significant computational efficiency, transforming the LS-OCSVM into a scalable model capable of handling large-scale datasets with ease. This marks the first adaptation of sparse least squares modeling to OCC, addressing the memory and scalability concerns prevalent in existing approaches. Together, the RLS-1SVM and SRLS-1SVM provide a unified solution for modern OCC challenges, advancing the learning for anomaly detection.

- [5] The use of PI has demonstrated significant improvements in the performance of ML models across various domains. In regression tasks, the integration of PI has primarily been explored within the framework of SVR, a method known for its high computational complexity. To address the computational challenges of SVR+ [70], we propose least square twin support vector regression with privileged information (LSTSVR+), which introduces PI into the least squares variant of TSVM (LSTSVR). The incorporation of PI in LSTSVR enriches the training process by leveraging additional information, thereby enhancing the model’s generalization capabilities. This integration represents a pioneering application of PI within the context of least squares plane-based regression models. To ensure robust performance, LSTSVR+ employs regularization terms in the optimization problems for both the regression function and the correcting function. These regularization terms align with the SRM principle, effectively preventing overfitting and improving model stability. Additionally, the proposed LSTSVR+ achieves computational efficiency by solving a system of linear equations, significantly reducing the computational burden compared to SVR+. This

combination of reduced complexity, enhanced generalization, and efficient training establishes LSTSVR+ as a robust and innovative contribution to the field of regression modeling.

1.5 Outline of the thesis

This thesis comprises eight chapters, each addressing a specific aspect of the research. Figure 1.1 demonstrates the layout of the thesis. A brief overview of the thesis is as follows:

- Chapter 2: This chapter reviews the existing literature related to the work presented in this thesis. It provides a comprehensive overview of the methodologies and models developed in the fields of classification, regression, and anomaly detection, laying the foundation for the subsequent contributions.
- Chapter 3: In this chapter, we propose two novel models. The first is the flexible pinball loss SVM (FP-SVM) [76], which enhances generalization performance by introducing a flexible pinball loss function. The second is the universum TWSVM with truncated pinball loss (Tpin-UTWSVM) [82], which incorporates universum data and a truncated pinball loss function to ensure an optimal balance between sparsity and noise insensitivity preserving theoretical properties.
- Chapter 4: This chapter introduces four innovative models to address challenges such as noise, outliers, and imbalanced data. The models include the intuitionistic fuzzy universum SVM (IFUSVM) [83], dual-center-based intuitionistic fuzzy plane-based classifiers [79], class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM-ID) [84], and bell-shaped fuzzy least squares twin SVM tailored for biomedical applications (BSFLSTSVM-ID) [85]. Each model leverages advanced

fuzzy/weighting techniques to enhance robustness and performance in the presence of noise/outliers and CIL.

- Chapter 5: Here, we present a novel pruning technique, termed ProDiSE (proximity and distance scoring ensemble for universal representation). This technique is integrated with LSSVM to develop ProDiSE-LSSVM, a sparse and efficient model designed to handle large-scale datasets effectively. The chapter discusses the advantages of ProDiSE-LSSVM in terms of scalability and computational efficiency.
- Chapter 6: This chapter proposes two robust models for anomaly detection. The first is the robust least squares one-class SVM (RLS-1SVM), which minimizes both the mean and variance of the modeling error. The second is the sparse and robust least squares one-class SVM, which incorporates sparsity to improve scalability for large-scale anomaly detection problems.
- Chapter 7: In this chapter, we introduce a novel regression model called least squares twin support vector regression with privileged information (LSTSVR+) [86]. This model integrates PI into the least squares twin regression framework, enhancing generalization performance and computational efficiency.

Each chapter systematically addresses specific challenges in ML, offering innovative solutions that contribute to advancements in the fields of classification, regression, and anomaly detection.

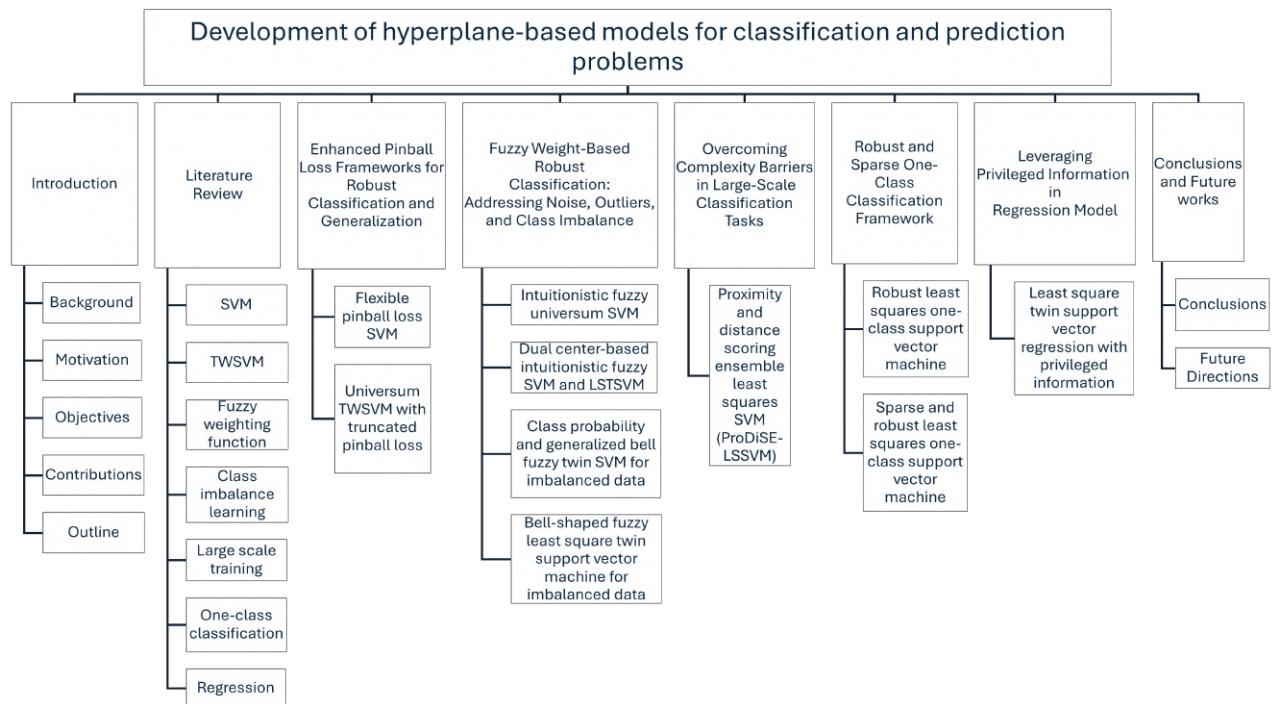


Figure 1.1: Layout of the thesis

Chapter 2

Literature Review

This chapter reviews the literature relevant to the work presented in this thesis. Section 2.1 discusses SVM and their various extensions, including models incorporating different loss functions and universum data. Section 2.2 focuses on Twin SVM (TWSVM). Section 2.3 examines membership functions proposed in the literature to address noise and outliers, highlighting their limitations and formulations within SVM and TWSVM frameworks. Section 2.4 explores models designed for imbalanced learning. Section 2.5 reviews approaches for large-scale training. Section 2.6 discusses literature on one-class classification models. Section 2.7 examines hyperplane-based regression models. Finally, Section 2.8 presents statistical tests used to evaluate the performance of the models.

Let T be the training set defined by $\{x_j, y_j\}_{j=1}^l$, where $x_j \in \mathbb{R}^n$, $y_j \in \{-1, 1\}$. Assume $\psi(\cdot)$ is the kernel map from input space to higher dimensional space and $K(x_j, x_k) = \psi(x_j)^t \psi(x_k)$. The decision hyperplane for data point $x \in \mathbb{R}^n$ is represented by $f(x) = w^t x + b = 0$, where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The set T has l_1 data points of the positive label and l_2 data points of the negative label. Assume M_1 and M_2 represent matrices of positive and negative class data points with dimensions $l_1 \times n$ and $l_2 \times n$, respectively. Let U be the set of universum data points given by $U = \{x_1^*, x_2^*, \dots, x_u^*\}$, where $x_i^* \in \mathbb{R}^n$.

2.1 SVM and their variants

Classification is a core ML task that enables models to accurately assign labels to data, playing a crucial role in a wide range of applications [87, 88]. Techniques such as decision trees, neural networks, Bayes methods, and others contribute to this objective. Among these, SVM [77, 89] is an efficient ML algorithm for classification and regression problems with a strong theoretical foundation and good generalization performance. SVM is built on the SRM concept and has its origin in statistical learning theory (SLT) [90]. SVM has several properties including kernel method, sparsity, and unique global solution. SVM approach has been effectively applied to a wide range of pattern categorization tasks, for example, face recognition [91], recognition of handwritten characters and digits [92], text categorization [93], diagnosis of cancer disease [94], and so forth.

Classical SVM [77] uses hinge loss function $L_{hinge} = \max\{0, u\}$, where $u \in \mathbb{R}$, to construct a hyperplane with maximum margin between the data points of distinct classes and giving no penalty to the correctly classified data points. Thus, only a few data points called ‘support vectors’ contribute to the formation of the separating hyperplane which leads to a sparse model.

2.1.1 SVM: Formulation

In SVM [77], a decision hyperplane is constructed to separate two classes by maximizing the margin between them while minimizing the penalty for misclassified samples. The penalty for misclassified data points is determined by using hinge loss, which is defined as $L_{hinge}(u_j) = \max\{0, u_j\}$, where $u_j = 1 - y_j f(x_j)$, $\forall j = 1, 2, \dots, l$. The optimization

problem for SVM model is formulated as:

$$\begin{aligned}
 \min_{w, b, \zeta} \quad & \frac{1}{2} \|w\|_2^2 + C_1 \sum_{j=1}^l \zeta_j \\
 \text{subject to} \quad & y_j (w^t \psi(x_j) + b) \geq 1 - \zeta_j, \\
 & \zeta_j \geq 0, \forall j = 1, 2, \dots, l,
 \end{aligned} \tag{2.1}$$

where $C_1 > 0$ signifies the regularization parameter. Using the Lagrangian method, the Wolfe dual for the primal problem (2.1) is obtained as:

$$\begin{aligned}
 \min_{\gamma} \quad & \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l \gamma_j \gamma_k y_j y_k K(x_k, x_j) - \sum_{j=1}^l \gamma_j \\
 \text{subject to} \quad & \sum_{j=1}^l \gamma_j y_j = 0, \\
 & 0 \leq \gamma_j \leq C_1, \forall j = 1, 2, \dots, l,
 \end{aligned} \tag{2.2}$$

where γ_j ($j = 1, 2, \dots, l$) are the Lagrangian multipliers. Using obtained γ_j , one can find the parameters of the plane w and b . The label for a test data point x is assigned using the function:

$$f(x) = \text{sign}(w^t \psi(x) + b). \tag{2.3}$$

2.1.2 LSSVM

The LSSVM is a plane-based classification technique that employs a least squares loss function. Unlike the traditional SVM, which solves a QPP, LSSVM solves a system of linear equations. This shift to linear equations enhances its computational efficiency, making it less complex than SVM in terms of optimization. The optimization problem for LSSVM is

formulated as follows:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 + \frac{C_1}{2} \sum_{j=1}^l \zeta_j^2 \\ \text{subject to} \quad & y_j(w^t \psi(x_j) + b) = 1 - \zeta_j, \quad \forall j = 1, 2, \dots, l. \end{aligned} \quad (2.4)$$

2.1.3 Pin-SVM

The distance between the optimal hyperplane and nearest points is easily influenced by feature noise, thus, the classifier trained using SVM is sensitive to feature noise and unstable under re-sampling [13]. Some previous initiatives have attempted to overcome noise sensitivity limitations such as using class center vectors to build SVM [14], robust SVM with least median loss penalty [15], a weighted SVM for data classification [16], and so on. One remarkable algorithm to overcome the feature noise is pin-SVM introduced by Huang et al. [17]. Pin-SVM uses the pinball loss function for the classification problem. It penalizes both correctly and incorrectly classified data points, and regulates the penalty for correctly classified samples by choosing the value of τ . The pinball loss is based on quantile distance [95] and seeks to maximize the quantile distance between two classes rather than the shortest distance, resulting in less sensitive to noise. The pinball loss function is defined as:

$$L_{\text{pin}}(u_j) = \begin{cases} u_j, & u_j \geq 0, \\ -\tau u_j, & u_j < 0, \end{cases} \quad (2.5)$$

where $u_j = 1 - y_j f(x_j)$, $\forall j = 1, 2, \dots, l$ and $\tau \in [0, 1]$ is the loss parameter. The pin-SVM achieves noise insensitivity, however it lacks sparsity which is an essential attribute of SVM. To obtain sparseness, Huang et al. [17] introduced ϵ -insensitive pin-SVM model.

The formulation of pin-SVM is expressed as:

$$\begin{aligned}
 & \min_{w,b,\zeta} \frac{1}{2} \|w\|_2^2 + C_1 \sum_{j=1}^l \zeta_j \\
 & \text{subject to } \zeta_j \geq 1 - y_j (w^t \psi(x_j) + b), \\
 & \zeta_j \geq -\tau (1 - y_j (w^t \psi(x_j) + b)), \forall j = 1, 2, \dots, l,
 \end{aligned} \tag{2.6}$$

where $C_1 > 0$ signifies the regularization parameter. The Wolfe dual for the primal problem (2.6) is obtained using the Lagrangian method and it can be written as follows:

$$\begin{aligned}
 & \min_{\gamma} \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l \gamma_j \gamma_k y_j y_k K(x_k, x_j) - \sum_{j=1}^l \gamma_j \\
 & \text{subject to } \sum_{j=1}^l \gamma_j y_j = 0, \\
 & \gamma_j + \left(1 + \frac{1}{\tau}\right) \eta_j = C_1, \\
 & \gamma_j + \eta_j \geq 0, \eta_j \geq 0, \forall j = 1, 2, \dots, l,
 \end{aligned} \tag{2.7}$$

where γ_j, η_j : $j = 1, 2, \dots, l$ are the Lagrangian multipliers. Solving (2.7) gives the Lagrangian multipliers which leads to unknowns w and b . The label for a test data point is assigned using the equation (2.3).

The penalization in the pin-SVM model leads it to minimize data point scatter along the separating hyperplane. However, it reduces sparsity, a crucial aspect of SVM. The elimination of sparsity is often harmful to SVM, since it increases computing time [96]. To address this shortcoming, pinball loss with ϵ -insensitive zone was introduced [17]. The insensitive zone modification makes pin-SVM sparse, but certain theoretical aspects of pinball loss are lost, i.e., insensitive zone pinball version is not a Bayes classifier. Researchers have attempted to overcome this problem by proposing SVM with truncated pinball loss [74, 96],

however, these initiatives render the problem non-convex, which makes it difficult to optimize. Furthermore, several pin-SVM variants have been introduced, in particular, solution path for pin-SVM with positive and negative parameter values (pin-SVM $_{\tau \in [-1,1]}$) [73], sequential minimal optimization (SMO) for pin-SVM [97], generalized pinball loss SVMs (GP-SVM) [98], improvement over pinball loss SVM [78] and so forth.

2.1.4 Unified Pin-SVM

It can be observed that the model in [73] needs to solve two separate QPPs for positive and negative τ values. Anand et al. [78] introduced unified pin-SVM which can be used to solve the pin-SVM without concerning about sign of τ . Unified pin-SVM re-examines the pinball loss for negative values and develops the appropriate optimization problem. It outperforms the pin-SVM model in terms of accuracy. The unified pin-SVM formulation is given as follows:

$$\begin{aligned} \min_{w,b,\zeta} \quad & \frac{1}{2} \|w\|_2^2 + C_1 \sum_{j=1}^l \zeta_j \\ \text{subject to} \quad & \zeta_j \geq 1 - y_j (w^t \psi(x_j) + b), \\ & \zeta_j \geq -\tau (1 - y_j (w^t \psi(x_j) + b)), \end{aligned} \quad (2.8)$$

where $C_1 > 0$ signifies the regularization parameter. The Wolfe dual of the primal problem (2.8) is written using the Lagrangian function as follows:

$$\begin{aligned} \min_{\gamma,\eta} \quad & \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l (\gamma_k - \tau \eta_k) (\gamma_j - \tau \eta_j) y_j y_k K(x_k, x_j) - \sum_{j=1}^l (\gamma_j - \tau \eta_j) \\ \text{subject to} \quad & \sum_{j=1}^l (\gamma_j - \tau \eta_j) y_j = 0, \\ & C_1 - \gamma_j - \eta_j = 0, \quad \gamma_j \geq 0, \eta_j \geq 0, \quad \forall j = 1, 2, \dots, l. \end{aligned} \quad (2.9)$$

The Lagrangian multipliers γ_j, η_j : $j = 1, 2, \dots, l$, are obtained by solving (2.9) which give w and b . The determination of w and b helps assign the label to the test data points using equation (2.3).

2.1.5 SVM with truncated pinball loss

Truncated pinball loss is a modified version of pinball loss which preserves the advantages of pinball loss along with leading to sparsity. It is defined by using an auxiliary function $H_\tau(u) = \max \{\tau u, 0\}$ ($u \in \mathbb{R}$, $\tau \in [0, 1]$ is a constant) as:

$$\mathbb{P}_{\tau,s}(u) = H_{1+\tau}(u) - H_\tau(u+s) + \tau s = \begin{cases} \tau s, & \text{if } u \leq -s, \\ -\tau u, & \text{if } -s < u < 0, \\ u, & \text{if } u \geq 0. \end{cases} \quad (2.10)$$

Tpin-SVM [74] incorporates truncated pinball loss in classical SVM which retains the properties of pinball loss along with leading to a sparse model. The goal of SVM is to maximize the margin between the bounding hyperplanes. The mathematical formulation of Tpin-SVM is presented as:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C_1 \sum_{j=1}^m \mathbb{P}_{\tau,s}(1 - y_j f(x_j)), \quad (2.11)$$

where decision hyperplane is given as: $f(x) = x^t w + b = 0$ with $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$; C_1 is a positive regularization parameter. QPP (2.11) can be solved by using CCCP method.

2.1.6 Universum SVM

Universum SVM [25] is more effective than classical SVM for classification problems. Universum data gives prior knowledge about the distribution of training data. The QPP of universum SVM is given by:

$$\begin{aligned}
 & \min_{w,b,\eta,\xi} \frac{1}{2} \|w\|^2 + C_1 \sum_{k=1}^l \zeta_k + C_u \sum_{j=1}^{2u} \eta_j \\
 & \text{subject to } y_k(w^t \psi(x_k) + b) \geq 1 - \zeta_k, \\
 & \quad y_j(w^t \psi(x_j^*) + b) \geq -\epsilon - \eta_j, \\
 & \quad \zeta_k \geq 0, \eta_j \geq 0 \forall j = 1, 2, \dots, 2u \text{ and } k = 1, 2, 3, \dots, l.
 \end{aligned} \tag{2.12}$$

Here, C_1, C_u are positive regularization parameters. ζ_k 's and η_j 's are the slack variables for $k = 1, 2, \dots, l$ and $j = 1, 2, \dots, 2u$. The number of universum data points is given by u . Also, $x_{m+i} = x_i^*$ with $y_{m+i} = 1$ and $x_{m+u+i} = x_i^*$ with $y_{m+u+i} = -1$ for $i = 1, 2, \dots, u$.

The Wolfe dual of above optimization problem using K.K.T. condition is obtained as:

$$\begin{aligned}
 & \min_{\gamma_k} \frac{1}{2} \sum_{k=1}^{l+2u} \sum_{j=1}^{l+2u} \gamma_k y_k \gamma_j y_j \psi(x_k)^t \psi(x_j) - \sum_{k=1}^{l+2u} \lambda_k \gamma_k, \\
 & \text{subject to } 0 \leq \gamma_k \leq C_1 \text{ for } k = 1, 2, \dots, l, \\
 & \quad 0 \leq \gamma_k \leq C_u \text{ for } k = l+1, l+2, \dots, l+2u, \\
 & \quad \lambda_k = 1 \text{ for } k = 1, 2, \dots, l, \\
 & \quad \lambda_k = -\epsilon \text{ for } k = l+1, l+2, \dots, l+2u, \\
 & \quad \sum_{k=1}^{l+2u} \gamma_k y_k = 0,
 \end{aligned} \tag{2.13}$$

where γ_k 's are non negative Lagrange multipliers. Applying K.K.T. condition to QPP (2.13), we get,

$$w = \sum_{k=1}^{l+2u} \gamma_k y_k \psi(x_k). \tag{2.14}$$

For test data point, the label is determined by equation (2.3).

2.2 Twin SVM and its variants

Despite its robust mathematical formulation, SVM entails solving a convex QPP, which introduces a significant computational burden. To overcome the computational cost, Mangasarian and Wild [21] proposed generalized eigenvalue proximal SVM. Further, Jayadeva et al. [22] proposed twin support vector machine (TWSVM), which constructs two non-parallel hyperplanes and transforms a large QPP of SVM into two smaller QPPs. TWSVM exhibits four times increase in efficiency as compared to the SVM [22] and solely focuses on empirical risk minimization, unlike SVM which considers SRM. Shao et al. [23] introduced twin bounded support vector machine (TBSVM), emphasizing the essence of statistical learning theory. Further, several variants of TWSVM are introduced which are discussed in a comprehensive survey by Tanveer et al. [99].

2.2.1 TWSVM: Formulation

TWSVM [22] constructs two non parallel hyperplanes such that each hyperplane is proximal to the corresponding class and atleast one unit away from other class. The two non parallel hyperplanes are given by:

$$K(x, M^t)w_1 + b_1 = 0 \text{ and } K(x, M^t)w_2 + b_2 = 0, \quad (2.15)$$

where $w_1, w_2 \in \mathbb{R}^n$ and $b_1, b_2 \in \mathbb{R}$. The formulation of TWSVM can be expressed as:

$$\begin{aligned} \min_{w_1, b_1, \zeta_2} \quad & \frac{1}{2} \|K(M_1, M^t)w_1 + e_1 b_1\|^2 + C_1 e_2^t \zeta_2 \\ \text{subject to} \quad & -(K(M_2, M^t)w_1 + e_2 b_1) + \zeta_2 \geq e_2, \quad \zeta_2 \geq 0 \end{aligned} \quad (2.16)$$

$$(2.17)$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_1} \quad & \frac{1}{2} \|K(M_2, M^t)w_2 + e_2 b_2\|^2 + C_2 e_1^t \zeta_1 \\ \text{subject to} \quad & (K(M_1, M^t)w_2 + e_1 b_2) + \zeta_1 \geq e_1, \quad \zeta_1 \geq 0. \end{aligned} \quad (2.18)$$

Here, e_1, e_2 are the vectors of ones having appropriate dimensions. ζ_1, ζ_2 represent slack variables and C_1, C_2 are the positive regularization parameters. After the determination of the planes (2.15), an arbitrary test data (say x) is assigned to a class depending on

$$f(x) = \underset{k=1,2}{\operatorname{argmin}} |x^t w_k + b_k|. \quad (2.19)$$

2.2.2 Least squares Twin SVM

The computational burden of solving QPPs is high, thus TWSVM incur high computational costs. To overcome the issue, Suykens and Vandewalle [29] proposed the least squares SVM (LSSVM) which solves a system of linear equations. Extending the paradigm, Kumar and Gopal [33] introduced least squares TWSVM (LSTSVM) which efficiently tackles two systems of linear equations of smaller size compared to LSSVM. Various extensions of LSTSVM have been introduced in literature such as multi-task LSTSVM [100] and so on.

The replacement of classical hinge loss in TWSVM with least squares loss leads to the formation of LSTSVM. LSTSVM [33] solves primal formulation by solving a system of linear equations. It leads to equality constraint with 2-norm of slack variables unlike TWSVM [22]. The non-negativity constraint on the slack variables is redundant, hence it is not considered in the formulation. The optimization problems for LSTSVM are as follows:

$$\begin{aligned} \min_{w_1, b_1, \zeta_2} \quad & \frac{1}{2} \|(K(M_1, M^t)w_1 + e_1 b_1)\|^2 + C_1 \zeta_2^t \zeta_2 \\ \text{subject to} \quad & -(K(M_2, M^t)w_1 + e_2 b_1) + \zeta_2 = e_2 \end{aligned} \quad (2.20)$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_1} \quad & \frac{1}{2} \| (K(M_2, M^t)w_2 + e_2b_2) \|^2 + C_2\zeta_1^t\zeta_1 \\ \text{subject to} \quad & (K(M_1, M^t)w_2 + e_1b_2) + \zeta_1 = e_1. \end{aligned} \quad (2.21)$$

2.2.3 Universum TWSVM

Universum TWSVM (UTSVM) [26] introduced the concept of universum data in TWSVM [22] which has prior information about the distribution of the data leading to better classification performance, however the improvement in the performance comes at the expense of increased complexity.

Let $U \in \mathbb{R}^{u \times n}$ denotes the universum data set. The non parallel hyperplanes are given by equations in (2.15) and the loss function considered for universum data is ϵ insensitive loss ($\lambda[t]$) which is a combination of two hinge losses ($\lambda_{-\epsilon}[t]$) and ($\lambda_{-\epsilon}[-t]$), i.e.,

$$\lambda[t] = \lambda_{-\epsilon}[t] + \lambda_{-\epsilon}[-t], \text{ where } \lambda_{-\epsilon}[t] = \max \{0, t - \epsilon\}. \quad (2.22)$$

The formulation for UTSVM is given by:

$$\begin{aligned} \min_{w_1, b_1, \zeta_2, \eta_2} \quad & \frac{1}{2} \| K(M_1, M^t)w_1 + e_1b_1 \|^2 + C_1e_2^t\zeta_2 + C_ue_u^t\eta_2 \\ \text{subject to} \quad & - (K(M_2, M^t)w_1 + e_2b_1) + \zeta_2 \geq e_2, \quad \zeta_2 \geq 0, \\ & (K(U, M^t)w_1 + e_ub_1) + \eta_2 \geq (-1 + \epsilon)e_u, \quad \eta_2 \geq 0 \end{aligned} \quad (2.23)$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_1, \eta_1} \quad & \frac{1}{2} \| K(M_2, M^t)w_2 + e_2b_2 \|^2 + C_2e_1^t\zeta_1 + C_ue_u^t\eta_1 \\ \text{subject to} \quad & (K(M_1, M^t)w_2 + e_1b_2) + \zeta_1 \geq e_1, \quad \zeta_1 \geq 0, \end{aligned} \quad (2.24)$$

$$-(K(U, M^t)w_2 + e_u b_2) + \eta_1 \geq (-1 + \epsilon)e_u, \quad \eta_1 \geq 0,$$

where e_u represents the vector of ones having appropriate dimension. η_1, η_2 are slack variables; C_u is the positive penalty parameter; ϵ signifies the tolerance value for the universum data.

After determining the unknowns of the planes in equation (2.15), for an arbitrary test data point, the class is determined by equation (2.19).

2.2.4 General TWSVM with pinball loss function (Pin-GTSVM)

The basic idea behind Pin-GTSVM [101] is incorporating pinball loss in the classical TWSVM. It overcomes the issues of noise sensitivity and instability to resampling. The proximal non-parallel planes are given by (2.15). The QPPs obtained are as follows:

$$\begin{aligned} \min_{w_1, b_1, \zeta_2} \quad & \frac{1}{2} \|K(M_1, M^t)w_1 + e_1 b_1\|^2 + C_1 e_2^t \zeta_2 \\ \text{subject to} \quad & -(K(M_2, M^t)w_1 + e_2 b_1) + \zeta_2 \geq e_2, \\ & -(K(M_2, M^t)w_1 + e_2 b_1) - \frac{\zeta_2}{\tau_2} \leq e_2 \end{aligned} \quad (2.25)$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_1} \quad & \frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_2 e_1^t \zeta_1 \\ \text{subject to} \quad & (K(M_1, M^t)w_2 + e_1 b_2) + \zeta_1 \geq e_1, \\ & (K(M_1, M^t)w_2 + e_1 b_2) - \frac{\zeta_1}{\tau_1} \leq e_1. \end{aligned} \quad (2.26)$$

Here $\tau_1, \tau_2 \in [0, 1]$ are the parameters of the pinball loss. Similar to TWSVM and UTSVM, for a test data point class is determined from equation (2.19).

2.3 Fuzzy theory

To tackle uncertainty, Zadeh [102] introduced fuzzy set theory, which extends the concept of classical (crisp) sets [103, 104]. In fuzzy set theory, the degree to which an element belongs to a set is determined by a membership function. Consider X as a non-empty set. A fuzzy set within X is defined as follows:

$$B = \{(x, \mu_B(x)) : x \in B\}.$$

Here, $\mu_B(x)$ represents the degree of membership of an element x in the fuzzy set B . The membership function assigns a value to each element, ranging between 0 and 1, and is expressed as $\mu_B : B \rightarrow [0, 1]$. Tools and methodologies derived from fuzzy set theory (FST) have been effectively integrated with ML models across various fields, including engineering [105, 106], medical sciences [107], and data mining [108], among others.

2.3.1 Intuitionistic fuzzy scheme (IFS)

The intuitionistic fuzzy [49] approach involves assigning an intuitionistic fuzzy number (IFN) to each training sample in the dataset based on their membership and non-membership functions, respectively. To assess the presence of noise or outliers in the dataset, a score function is established utilizing the membership and non-membership functions.

- **Membership function:** It is defined in terms of the distance between the class center and the training data points in the high-dimensional feature space. Mathematically,

$$\mu_k = \begin{cases} 1 - \frac{\|\psi(x_k) - C_+\|}{R_+ + \Delta}, & \text{if } y_k = 1, \\ 1 - \frac{\|\psi(x_k) - C_-\|}{R_- + \Delta}, & \text{if } y_k = -1, \end{cases} \quad (2.27)$$

where Δ is a small positive number. $C_+ = \frac{1}{l_1} \sum_{y_k=1} \psi(x_k)$ and $C_- = \frac{1}{l_2} \sum_{y_k=-1} \psi(x_k)$ are the centers of positive and negative classes, respectively. And, $R_+ = \max_{y_k=1} \|\psi(x_i) - C_+\|$ and $R_- = \max_{y_k=-1} \|\psi(x_i) - C_-\|$ denote the radius of positive and negative classes, respectively.

- **Non-membership function:** It depends on the proportion between dissimilar points to the total number of points in a neighbourhood around each sample. It is defined as:

$$\nu_k = (1 - \mu_k)\zeta(x_k), \quad (2.28)$$

where

$$\zeta(x_k) = \frac{|x_j : \|\psi(x_k) - \psi(x_j)\| \leq \beta, y_k \neq y_j|}{|x_j : \|\psi(x_k) - \psi(x_j)\| \leq \beta|}.$$

Here, β corresponds to user-defined positive parameter.

- **Score function:** The pair (μ_k, ν_k) for each training sample (x_k) is called IFN and is used to calculate score value of data points by the function given as:

$$s_k = \begin{cases} \mu_k, & \text{if } \nu_k = 0, \\ 0, & \text{if } \mu_k \leq \nu_k, \\ \frac{1-\nu_k}{2-\mu_k-\nu_k}, & \text{otherwise.} \end{cases} \quad (2.29)$$

2.3.2 SVM with IFN formulation:

The optimization problem for SVM with IFN [109] assigned to data points is given by:

$$\begin{aligned} \min_{w, \zeta} \quad & \frac{1}{2} \|w\|^2 + C_1 \sum_{k=1}^l s_k \zeta_k \\ \text{subject to} \quad & y_k(w^t \psi(x_k) + b) \geq 1 - \zeta_k, \quad \zeta_k \geq 0, \quad \forall k = 1, 2, \dots, l, \end{aligned} \quad (2.30)$$

where s_k denotes the score value of k^{th} training data calculated using equation (2.29). Using K.K.T. condition, the dual of the optimization problem is obtained as:

$$\begin{aligned} \min_{\gamma_k} \quad & \frac{1}{2} \sum_{k=1}^l \sum_{j=1}^l \gamma_k y_k y_j \gamma_j \psi(x_k)^t \psi(x_j) - \sum_{k=1}^l \gamma_k \\ \text{subject to} \quad & \sum_{k=1}^l \gamma_k y_k = 0, \\ & 0 \leq \gamma_k \leq s_k C_1, \quad \forall \quad k = 1, 2, \dots, l, \end{aligned} \quad (2.31)$$

where γ_k 's are non negative Lagrange multipliers for $k = 1, 2, \dots, l$. Solving the QPP (2.31), we get γ_k . Obtain w by the relation $w = \sum_{k=1}^l \gamma_k y_k \psi(x_k)$ and determine b with the help of support vectors. For an arbitrary test data sample, the decision function is given by equation (2.3).

2.3.3 Intuitionistic Fuzzy TWSVM (IFTWSVM)

By incorporating the concept of IFN in the classical TWSVM, Rezvani et al. [49] proposed IFTWSVM which is effective against noise and outliers. The kernel-generated surfaces for the non-linear case are given by:

$$K(x, M^t)w_1 + b_1 = 0, \text{ and } K(x, M^t)w_2 + b_2 = 0, \quad (2.32)$$

where $K(\cdot, \cdot)$ represents the kernel function; w_1, w_2 are weight vectors; b_1, b_2 are bias terms and $M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$. The nonlinear optimization problems corresponding to IFTWSVM are given as follows:

$$\begin{aligned} \min_{w_1, b_1, \zeta_2} \quad & \frac{1}{2} \|K(M_1, M^t)w_1 + e_1 b_1\|^2 + \frac{1}{2} C_1 \|w_1\|^2 + C_2 s_2^t \zeta_2 \\ \text{s.t.} \quad & -(K(M_2, M^t)w_1 + e_2 b_1) + \zeta_2 \geq e_2, \zeta_2 \geq 0 \end{aligned} \quad (2.33)$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_1} & \frac{1}{2} \|K(M_2, M^t)w_2 + e_2b_2\|^2 + \frac{1}{2}C_3\|w_2\|^2 + C_4s_1^t\zeta_1 \\ \text{s.t. } & (K(M_1, M^t)w_2 + e_1b_2) + \zeta_1 \geq e_1, \zeta_1 \geq 0, \end{aligned} \quad (2.34)$$

s_1, s_2 are score vectors obtained from Eq. (2.29) for positive and negative class data points, respectively.

After determining the hyperplanes, a test sample is labeled depending on function:

$$f(x) = \underset{j=1,2}{\operatorname{argmin}} \left(\frac{|w_j^t K(x, M^t) + b_j|}{\sqrt{w_j^t K(x, M^t) w_j}} \right). \quad (2.35)$$

Subsequent models are proposed using IF to mitigate the impact of noise and outliers such as IF weighted least squares twin SVMs [110], IFTWSVM with the insensitive pinball loss [111], and IF LSTSVM [112]. These aforementioned models include matrix inversion which is overcome by improved IFTWSVM (IIFTWSVM) [50]. To handle imbalance learning, IFTWSVM for imbalanced data (IFTWSVM-ID) [51] is proposed. Recently, Rezvani and Wu [113] proposed IF TWSVM based on relative density information for multi-class classification using one versus one strategy.

2.3.4 Dual-center based intuitionistic fuzzy scheme

Though IF scheme [109] works effectively against the adverse impact of impure data points present in the training data, it has certain limitations. First, the non-membership function of IF scheme is dependent on a threshold value, which is difficult to choose. Additionally, determining the non-membership value is time-consuming as it entails Euclidean distance calculations among data points. To address the aforementioned limitations, Zhang et al. [114] proposed dual center-based intuitionistic fuzzy scheme. Here, we discuss the

DC-IF scheme as follows:

- [1] **Membership function:** Similar to IF scheme, it is calculated by the distance of the data points to the class center. This calculation can be expressed as follows:

$$\mu(x_j) = \begin{cases} 1 - \frac{P_j(C_+)}{R_+ + \Delta}, & \text{if } y_j = 1, \\ 1 - \frac{N_j(C_-)}{R_- + \Delta}, & \text{if } y_j = -1, \end{cases}$$

where Δ is a small positive number and

$$P_j(C_{\pm}) = \|\psi(x_j) - C_{\pm}\|^2 \text{ for } y_j = 1,$$

$$N_j(C_{\pm}) = \|\psi(x_j) - C_{\pm}\|^2 \text{ for } y_j = -1,$$

$R_+ = \max_{y_i=1} P_i(C_+)$ and $R_- = \max_{y_i=-1} N_i(C_-)$ denote the radius of positive and negative class.

- [2] **Non-membership function:** The computation of the non-membership value involves considering the distance of data points from the center of the other class. Mathematically,

$$\nu_j = \begin{cases} \frac{1/P_j(C_-)}{1/P_j(C_+) + 1/P_j(C_-)}, & \text{if } y_j = 1, \\ \frac{1/N_j(C_+)}{1/N_j(C_+) + 1/N_j(C_-)}, & \text{if } y_j = -1. \end{cases} \quad (2.36)$$

Score function: For the training data points, the score function can be obtained by combining weighted sum of membership and non-membership functions which is expressed as:

$$s(x_j) = (\lambda_1 \mu(x_j)^q + (1 - \lambda_1)(1 - \nu(x_j)^q))^{\frac{1}{q}}, \quad (2.37)$$

where q is a positive parameter and $\lambda_1 \in [0, 1]$.

2.3.5 Affinity and class probability based fuzzy support vector machine for imbalanced datasets (ACFSVM)

To overcome the limitation of SVM in the presence of noise/outliers and CIL issue, Tao et al. [46] proposed ACFSVM, where they assign membership values to the data points. Using the concept of support vector data description (SVDD) [47], the affinity of the majority class is calculated which identifies border samples (outliers) existing among the majority samples effectively. The k -nearest neighbourhood (KNN) is employed to calculate the probability of data points in the majority class. Here, R denotes the center and radius of training data points and d_k^{svdd} denotes the distance of each training data point (x_k) from the class center. The function to calculate affinity is written as :

$$\mu_{aff}(x_k) = \begin{cases} 0.8 \left(\frac{d_k^{svdd} - \min(d_k^{svdd})}{\max(d_k^{svdd}) - \min(d_k^{svdd})} \right), & \text{if } d_k^{svdd} < \rho \times R, \\ 0.2 \left(\exp(\beta(1 - \frac{d_k^{svdd}}{\rho \times R})) \right), & \text{if } d_k^{svdd} \geq \rho \times R, \end{cases} \quad (2.38)$$

where $\min(d_k^{svdd})$ and $\max(d_k^{svdd})$ denote the minimum and maximum distance of the data points from the class center, $\rho \in (0, 1]$ controls the size of outliers and the border samples and β is a positive weight decay parameter.

Using the concept of KNN, a probability value ($p(x_k)$) is assigned to each data point which is calculated as the ratio between the same class data points among the k selected nearest neighbourhood data points in the kernel space and k . Hence, ACFSVM calculates the final membership values as follows:

$$\mu(x_k) = \begin{cases} 1, & \text{if } x_k = x_k^{min}, \\ \mu_{aff}(x_k) \times p(x_k), & \text{if } x_k = x_k^{maj}. \end{cases} \quad (2.39)$$

2.3.6 Limitation of the existing weighting functions

Different existing weighting functions have certain limitations which are analysed as follows:

[1] Though the score function defined using the concept of IFN in equation (2.29) has the resistance against noise and outliers, it has certain limitations such as:

- Assume data having distribution characterised by a significant concentration of training data points along the boundary of the class as depicted in Figure 2.1a. Then, the majority of data points are assigned a membership that is close to zero according to equation (2.27), as visually depicted in Figure 2.2a. Consequently, their impact on the classification process is minimized.
- The membership function of IFN decreases linearly with distance, i.e., the membership value decreases as the distance of data points and corresponding class center increases. Though the points lying near its class center belong to the dense region of its class distribution (*e.g.*, data point A in Figure 2.1b), these are given a lower membership value compared to the class centre (C). Consequently, their significance in the classification process is reduced.

[2] The membership function discussed in ACFSVM in equation (2.39) is an effective way to deal with noise, outliers and imbalanced data simultaneously, however, it has the following limitation:

- The computation of affinity in equation (2.38) involves solving QPP associated with SVDD technique, which has high complexity. Furthermore, the determination of class probability entails selecting the k nearest neighbors for each training data point, a factor that significantly contributes to the overall increase in time

complexity.

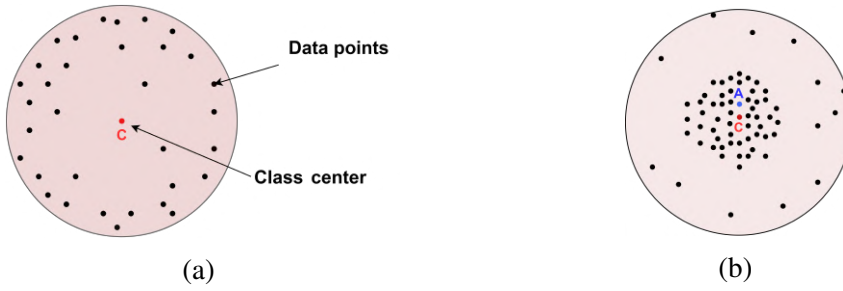


Figure 2.1: Representation of a training sample. Fig. (a) corresponds to a sample having majority of the data points lying on the boundary of class. Fig. (b) corresponds to a sample having majority of the data points lying around the center of the class.

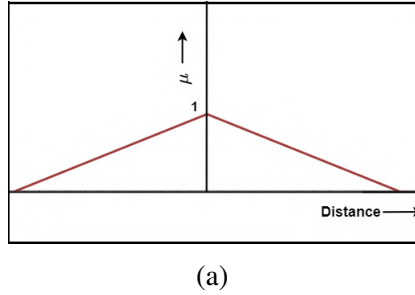


Figure 2.2: Graphical illustration of membership function in equation (2.27).

[3] The membership function of LS-FLSTSVM-CIL [115] discussed in equation (2.40) is suitable for outliers/noise and imbalanced data, it still has a limitation as follows:

- In a data distribution with a dense region surrounding the class center, class samples receive decreasing membership values as their distance from the center increases, with the center itself assigned a value of one. Consequently, despite their location in the dense region, the contribution of a substantial amount of samples from this vicinity is reduced.

2.4 Class imbalance learning

Learning from an imbalanced dataset and addressing the noise and outliers present in the training dataset are the two primary challenges related to classification problems. For the datasets having two classes, imbalanced datasets have one-class in abundance (majority class) as compared to the other class (minority class). There are several real-world domains such as face recognition [43], where imbalanced dataset can be seen. In general, there are two techniques to deal with CIL problems: data-level approach and algorithm-level approach. Data-level approaches involve preprocessing techniques to get balanced data such as undersampling (neighbourhood cleaning rule [116]) and oversampling (synthetic minority oversampling technique [117]). However, data-level approaches may discard valuable information (undersampling) or introduce noise (oversampling) [118]. The algorithmic level approach modifies the algorithm. The TWSVM algorithm performs well on a balanced dataset, however, when dealing with imbalanced datasets, TWSVM considers all samples equal and overlooks the distinction between minority and majority classes, which might lead to the obtained classification boundary being biased in favour of the majority class [49]. Furthermore, TWSVM provides equal attention to all the training samples including potential outliers or class noises that can significantly skew the decision hyperplane [22].

To ensure good performance of a model for imbalance learning with noise and outliers, various SVM variants have been proposed where the contribution of input data depends on membership functions, which lessen the influence of outliers and noise on classification technique. Fuzzy SVM (FSVM) [44] is proposed to handle the outliers and noise, where the assignment of fuzzy value depends on the distance from class center. The assignment of membership to the input data is the key point in FSVM variants. Though fuzzy concept overcomes the effect of noise/outliers, it still suffers from CIL issue. To handle the

CIL in the presence of outliers and noise, Batuwita and Palade [45] proposed fuzzy SVM for imbalance data (FSVM-CIL) with four different membership functions. In FSVM-CIL, the hyperplane-based membership function assumes initially obtained hyperplane is an accurate prediction of the final hyperplane which is not the case always. To overcome the limitations of the aforementioned model for CIL in the presence of the noise/outliers effectively, Tao et al. [46] introduced affinity and class probability-based FSVM (ACFSVM) designed for imbalanced datasets which uses support vector data description (SVDD) [47] and class probability to obtain the membership value of the data points.

2.4.1 IFTWSVM for imbalanced data

Though IFTWSVM [49] works effectively for balanced data, it may not work effectively for imbalanced data. To overcome this, Rezvani and Wang [51] proposed IFTWSVM-ID by assigning different weights to the majority and minority samples through a weighting technique. The optimization problems associated with IFTWSVM-ID are given as follows:

$$\begin{aligned} \min_{w_1, b_1, \zeta_1} & \frac{1}{2} \|K(M_1, M^t)w_1 + e_1 b_1\|^2 + \frac{C_1 d_1}{2} \|w_1\|^2 + C_2 d_1 s_2^t \zeta_1 \\ \text{subject to} & -(K(M_2, M^t)w_1 + e_2 b_1) + \zeta_1 \geq e_2, \quad \zeta_1 \geq 0 \end{aligned}$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_2} & \frac{1}{2} \|K(M_2, M^t)w_2 + e_2 b_2\|^2 + \frac{C_3 d_2}{2} \|w_2\|^2 + C_4 d_2 s_1^t \zeta_2 \\ \text{subject to} & (K(M_1, M^t)w_2 + e_1 b_2) + \zeta_2 \geq e_1, \quad \zeta_2 \geq 0, \end{aligned}$$

d_1 and d_2 denote the weights to deal with CIL.

2.4.2 Large scale fuzzy least squares TWSVM for CIL

Ganaie et al. [115] introduced large-scale fuzzy LSTSVM (LS-FLSTSVM-CIL) which incorporated fuzzy membership function with LSTSVM and solved the obtained optimization problem using sequential minimal optimization [119]. The membership value assigned to the minority class samples is 1. The majority of class samples are assigned membership values using the following function:

$$\text{mem} = \left(\frac{1}{1 + ir} \right) + \left(\frac{ir}{1 + ir} \right) \left(\frac{\exp(c_0(m_1 - m_2)/m - m_2/r_2) - \exp(2c_0)}{\exp(c_0) - \exp(-2c_0)} \right), \quad (2.40)$$

where ir denotes the imbalance ratio of the majority samples to the minority samples. m_1 and m_2 correspond to the distance between x_i and the centroid of the minority and majority class, respectively. m corresponds to the distance between the centroids of the binary classes. r_2 signifies the maximum distance of the data points of the majority class from its center. c_0 is the scale of the exponential function. The optimization problems of LS-FLSTSVM-CIL are expressed as follows:

$$\begin{aligned} \min_{w_1, b_1, \zeta_1} \quad & \frac{C_1}{2} (\|u_1\|^2 + b_1^2) + \frac{1}{2} \|\xi_1\|^2 + \frac{C_2}{2} \|H_2 \zeta_1\|^2 \\ \text{subject to} \quad & \psi(M_1)w_1 + e_1 b_1 = \xi_1, \\ & -(\psi(M_2)u_1 + e_2 b_1) + \zeta_1 = e_2, \end{aligned}$$

and

$$\begin{aligned} \min_{w_2, b_2, \zeta_2} \quad & \frac{C_3}{2} (\|w_2\|^2 + b_2^2) + \frac{1}{2} \|\xi_2\|^2 + \frac{C_4}{2} \|H_1 \zeta_2\|^2 \\ \text{subject to} \quad & \psi(M_2)w_2 + e_2 b_2 = \xi_2, \\ & (\psi(M_1)u_2 + e_1 b_2) + \zeta_2 = e_1. \end{aligned}$$

H_1 and H_2 denote the diagonal matrices having fuzzy membership values of positive and negative samples, respectively.

2.5 Large scale training

In the rapidly evolving field of ML, the ability to process and train on large-scale datasets has become crucial as industries generate vast amounts of data. Traditional supervised learning models often face computational and memory limitations, leading to increased training times. With big data driving advancements in areas like AI [120], and healthcare [121], accurate and scalable classification models are essential. Existing methods often trade off scalability for accuracy or impose high memory demands, highlighting the pressing need for innovative models that combine efficiency, precision, and adaptability to diverse data structures.

2.5.1 Global-representation-based sparse LSSVM (GRS-LSSVM)

To address the issues of non-sparsity of the LSSVM [29], Ma et al. [42] presented GRS-LSSVM. Instead of training the model with the entire training data, GRS-LSSVM chooses to directly select the SVs from input data and forms $S \subset X$. The selection of SVs heavily relies on the properties of points within the feature space, which is a noniterative process. The decision hyperplane of GRS-LSSVM is given by:

$$grs(x) = u^t \psi(x) + b,$$

where $u = \sum_{j=1}^s \beta_j \psi(x_j)$, $s < l$, s denotes the number of reserved support vectors (SVs), and $S \subset X$ with $|S| = s$ be the set of reserved SVs. The optimization problem of GRS-

LSSVM is written as:

$$\min_{u,b} \frac{1}{2} \|u\|^2 + \frac{C}{2} \sum_{j=1}^s (grs(x_j) - y_j)^2$$

subject to $|S| \leq s$ and $S \subset X$.

GRS-LSSVM constructs subset S by selecting points that are well-suited for representing other points based on their point density and dispersion.

Point density: The point density (ρ_j) of a sample x_j with respect to a kernel function K is determined by the number of points in the feature space that fall within the θ neighbourhood of x_j . Mathematically, $\rho_j = \sum_{i=1}^l \Delta(D_{jk})$, where

$$\Delta(d) = \begin{cases} 1, & \text{if } d \leq \theta, \\ -1, & \text{if } d > \theta. \end{cases}$$

Point dispersion: Point dispersion (ζ_j) for a sample x_j is defined in the feature space as follows:

$$\zeta_j = \min_{\rho_j < \rho_k} D_{jk}.$$

Here, ζ_j represents the minimum distance among the distances from x_j to other points whose ρ values are greater than that of x_j .

Due to the variations in their units, comparing the values of ρ and ζ directly across different locations is not feasible. Therefore, ρ and ζ are normalized using min-max normalization. Further, a global indicator τ is taken as $\tau_j = \min\{\rho_j, \zeta_j\}$. A data point becomes more representative as the value of τ increases, thus, GRS-LSSVM chooses the data points with higher values of τ to carry out the operation.

2.5.2 Limitation of globally representative point selection

Despite increasing the sparsity of the LSSVM model by selecting data points with higher τ values, GRS-LSSVM contains inherent limitations, as detailed below:

- Although GRS-LSSVM is designed to avoid iterative optimization, its reliance on calculating pairwise distances between all data points (D_{jk}) for evaluating their dispersions may become computationally expensive, especially for large-scale datasets. The need for pairwise distance calculations increases the overall time complexity, making the approach less scalable for large datasets.
- GRS-LSSVM relies on the parameter θ to define the neighbourhood of a data point and determine its point density (ρ_j). The choice of θ is crucial, as it influences the density calculation and, consequently, the selection of representative SVs. An inappropriate selection of θ can lead to overfitting (by selecting too many SVs) or underfitting (by selecting too few), resulting in a suboptimal model.

2.6 OCC

In this section, we provide a concise overview of the related work. Suppose $\mathcal{T} = \{x_j\}_{j=1}^l$ be a set of l training samples, which belong to a particular target class. We define ψ as the kernel map from input space to higher dimensional space such that $(\psi(x_i)^t \psi(x_j)) = K(x_i, x_j)$ and $\psi(x_j) \in \mathbb{R}^d$.

2.6.1 LS-OCSVM

LS-OCSVM [80] attempts to find an optimal hyperplane in the feature space, reflecting the degree of similarity between a sample and the training set based on the distance between the image of the sample and the hyperplane. The optimization problem for LS-OCSVM is

expressed as:

$$\begin{aligned} \min_{w, \rho} \frac{1}{2} \|w\|^2 - \rho + \frac{C}{2} \sum_{j=1}^l \zeta_j^2 \\ \text{subject to } w^t \psi(x_j) = \rho - \zeta_j, \quad j = 1, 2, \dots, l, \end{aligned} \quad (2.41)$$

where C denotes the trade-off parameter; w and ρ correspond to weight vector and bias, respectively. ζ_j represents the modeling error. By employing the Lagrange method to solve (2.41), we derive the Lagrangian, Γ , which is given as:

$$\Gamma = \frac{1}{2} \|w\|^2 - \rho + \frac{C}{2} \sum_{j=1}^l \zeta_j^2 - \sum_{j=1}^l \beta_j (w^t \psi(x_j) - \rho + \zeta_j),$$

where β_j denotes the Lagrange multipliers. By applying the Karush-Kuhn-Tucker (K.K.T.) conditions, we obtain the following equations by setting the partial derivatives of Γ with respect to each variable to zero:

$$\frac{\partial \Gamma}{\partial w} = 0 \implies w = \sum_{j=1}^l \beta_j \psi(x_j), \quad (2.42)$$

$$\frac{\partial \Gamma}{\partial \rho} = 0 \implies \sum_{j=1}^l \beta_j = 1, \quad (2.43)$$

$$\frac{\partial \Gamma}{\partial \zeta_j} = 0 \implies \zeta_j = \frac{\beta_j}{C}, \quad (2.44)$$

$$\frac{\partial \Gamma}{\partial \beta_j} = 0 \implies w^t \psi(x_j) + \zeta_j - \rho = 0. \quad (2.45)$$

Solving the equations (2.42)-(2.45), we obtain a linear system of equations as follows:

$$\begin{bmatrix} 0 & e^t \\ e & K + I/C \end{bmatrix} \begin{bmatrix} -\rho \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

where e signifies a column vector of ones and I denotes the identity matrix, both having the appropriate dimensions; β is a vector of Lagrange multipliers and K denotes the kernel

matrix. The decision function for determining the label of a test data point is expressed as follows:

$$f(x) = \begin{cases} 1, & \text{if } p(x) \leq \varrho, \\ -1, & \text{if } p(x) > \varrho. \end{cases} \quad (2.46)$$

Here, $p(x) = \frac{|g(x)|}{\|\beta\|}$, $g(x) = w^t \psi(x) - \rho = \sum_{j=1}^l K(x_j, x) - \rho$, denotes the equation of hyperplane. ϱ corresponds to the threshold value which is obtained as $\max_{j=1,2,\dots,l} p(x_j)$.

From equations (2.43) and (2.44), we get $\sum_{j=1}^l \zeta_j = \frac{1}{C}$, that implies LS-OCSVM assumes mean of the modeling error as a constant.

Let y be a non-linear system with uncontrolled variation and random disturbance given as follows:

$$y = h(x) + \epsilon, \quad (2.47)$$

where x denotes the input. h corresponds to the non-linear function and ϵ denotes noise which may be Gaussian or non-Gaussian. The model generated by LS-OCSVM has the equation

$$\hat{y} = g(x) + \zeta, \quad (2.48)$$

which approximates the nonlinear system $h(x)$. Here, $\hat{y}$ is the predicted label and ζ is the sum of noise and the approximation error $\delta (= h(x) - g(x))$, that is,

$$\zeta = \epsilon + \delta. \quad (2.49)$$

Considering the expectation of a real system and obtained model, we get

$$\mathbb{E}(y) = \mathbb{E}[h(x)] + \mathbb{E}(\epsilon) \text{ and } \mathbb{E}(\hat{y}) = \mathbb{E}(w^t \psi(x) - \rho) + \mathbb{E}(\zeta),$$

respectively, where $\mathbb{E}$ denotes expectation operation. From aforementioned equation, we observe that if the mean of the noise present in the real system is constant, i.e., $\mathbb{E}(\epsilon) = \frac{1}{C}$, then LS-OCSVM approximates the real system $h(x)$ as $\mathbb{E}(\zeta) = \frac{1}{C}$. However, if the mean of the modeling error present in the real system is not equal to constant $\frac{1}{C}$, then the LS-OCSVM is not effective. Thus, LS-OCSVM has two limitations as stated:

- It assigns equal weights to all the samples, and is not effective in the presence of outliers.
- It is biased for the systems which have a mean of modeling error as constant, i.e., $\frac{1}{C}$.

Moreover, for another type of system, such as having non-Gaussian noise, the performance of LS-OCSVM deteriorates. Therefore, it is imperative to develop an effective LS-OCSVM for modeling unknown nonlinear systems with random disturbance.

Considering the variances of the nonlinear system (2.47) and the modeled system using LS-OCSVM (2.48), we get,

$$\mathbb{V}(y) = \mathbb{V}(h(x) + \epsilon) \text{ and } \mathbb{V}(\hat{y}) = \mathbb{V}(g(x) + \zeta), \quad (2.50)$$

respectively, where $\mathbb{V}$ denotes variance operation. The equations in (2.50) can be expressed by taking into account the statistical independence between the noise and the model [122], as follows:

$$\mathbb{V}(y) = \mathbb{V}(\epsilon) \text{ and } \mathbb{V}(\hat{y}) = \mathbb{V}(\zeta). \quad (2.51)$$

Further, following equation (2.49), we get $\mathbb{V}(\hat{y}) = \mathbb{V}(\zeta) = \mathbb{V}(\epsilon + \delta) = \mathbb{V}(\epsilon) + \mathbb{V}(\delta) \geq \mathbb{V}(y) = \mathbb{V}(\epsilon)$.

Thus, $\mathbb{V}(\zeta) \geq \mathbb{V}(\epsilon)$ or variance of the modeled system is greater than the variance of the real system. The equality holds if the approximation error δ is zero. By minimizing the variance in modeling error ζ , the model becomes more resilient to disturbances and enhances its performance in representing the real system.

Exclusively reducing $\mathbb{V}(\zeta)$ may introduce bias, leading to an inadequate model fit with the real system. To overcome this limitation, we additionally minimize $\mathbb{E}^2(\zeta)$. Consequently, minimizing the squared modeling error ($\mathbb{E}^2(\zeta)$) can contribute to achieving a well-fitting and accurate representation of the system.

2.7 Regression

In this section, we provide a concise overview of existing models, namely SVR, SVR+, and TSVR. We mainly discuss the non-linear case of the models. Let $S = \{(x_1, x_{*1}, y_1), (x_2, x_{*2}, y_2), \dots, (x_m, x_{*m}, y_m)\}$ denote the training samples, where $x_i \in \mathbb{R}^n$, and $x_{*i} \in \mathbb{R}^k$ denote the training features and PI, respectively. $y_i \in \mathbb{R}$ corresponds to the target value. Assume M and M^* denote matrices containing training input data and their PI, respectively. Y denotes the vector containing the target values of input data. Assume ψ be the nonlinear map from input space ($\mathbb{R}^n$) to higher dimensional space, and $K(x_i, x_j) = \psi(x_i)^t \psi(x_j)$ corresponds to the kernel matrix.

2.7.1 Support vector regression

The classical SVR [9] employs an ϵ -insensitive loss function aiming to identify a regressor function that permits a maximum deviation of ϵ from the true target values of the training dataset. In the non-linear case, the objective is to find the flattest function within

the feature space, rather than in the input space. The equation of the regressor function for the non-linear case is given as follows:

$$r(x) = u^t \psi(x) + b, \quad (2.52)$$

where u and b are the parameters of the regression function. The optimization problem for non-linear SVR is expressed as:

$$\begin{aligned} \min_{u, \zeta_1, \zeta_2} & \frac{1}{2} \|u\|^2 + C(e^t \zeta_1 + e^t \zeta_2) \\ \text{such that } & Y - (\psi(M)u + be) \leq \epsilon e + \zeta_1, \quad \zeta_1 \geq 0, \\ & (\psi(M)u + be) - Y \leq \epsilon e + \zeta_2, \quad \zeta_2 \geq 0. \end{aligned} \quad (2.53)$$

Here, $C > 0$ is the regularization parameter between the penalty term and flatness of regressor. ζ_1 and ζ_2 are the vectors of slack variables. e denotes a vector of ones having a suitable dimension. By employing the equation (2.52) with the unknowns obtained from QPP (2.53), the target value of the unseen samples can be determined.

2.7.2 SVR+

Incorporating PI, Vapnik and Vashist [70] introduced SVM+ regression (SVR+), where the PI is integrated into SVR by employing a correction function for slack variables. This model is trained utilizing both the original features and supplementary information derived from the training data samples. The two correcting functions of the SVR+ are given by:

$$r_1^*(x_*) = u_1^{*t} \psi(x_*) + b_1^* \text{ and } r_2^*(x_*) = u_2^{*t} \psi(x_*) + b_2^*.$$

Here, $r_1^*(x)$ and $r_2^*(x)$ are the correcting functions for the approximation of slack variables ζ_1 and ζ_2 , respectively. The regressor function is given by equation (2.52). The optimization

problem of SVR+ is expressed as:

$$\begin{aligned}
 & \min_{u, u_1^*, u_2^*, b, b_1^*, b_2^*} \frac{1}{2} [\|u\|^2 + C_1(\|u_1^*\|^2 + \|u_2^*\|^2)] + C_2[(u_1^{*t} \psi(M^*) + b_1^* e) + (u_2^{*t} \psi(M^*) + b_2^* e)] \\
 & \text{such that } Y - (\psi(M)u + be) \leq \epsilon e + (\psi(M^*)u_1^* + b_1^* e), \\
 & (\psi(M)u + be) - Y \leq \epsilon e + (\psi(M^*)u_2^* + b_2^* e), \\
 & (\psi(M^*)u_1^* + b_1^* e) \geq 0, \\
 & (\psi(M^*)u_2^* + b_2^* e) \geq 0.
 \end{aligned} \tag{2.54}$$

Here, e signifies a column vector of ones with appropriate dimensions. C_1, C_2 corresponds to the positive regularization parameters. One can obtain the unknowns $u, u_1^*, u_2^*, b, b_1^*, b_2^*$ using the dual problem, explained in [70].

2.7.3 TSVR

TSVR [62] constructs non-parallel hyperplanes akin to TWSVM [22], but they constitute distinct geometric and structural properties. Firstly, while TWSVM aims to find hyperplanes close to each class, TSVR identifies ϵ -insensitive down and upbound regressor functions. Secondly, in TWSVM, the objective function of each QPP relies solely on the data points from one-class, whereas TSVR incorporates all data points from both classes into its QPPs. These distinctions highlight the unique characteristics and objectives of each approach. The ϵ_1 -insensitive down bound regressor and ϵ_2 -insensitive up bound regressor for non-linear cases are given by:

$$r_1(x) = K(x^t, M^t)u_1 + b_1 \text{ and } r_2(x) = K(x^t, M^t)u_2 + b_2,$$

respectively. The optimization problem for TSVR is written as:

$$\begin{aligned} \min_{u_1, b_1, \zeta_1} \quad & \frac{1}{2} \|Y - e\epsilon_1 - (K(M, M^t)u_1 + eb_1)\|^2 + C_1 e^t \zeta_1 \\ \text{such that} \quad & Y - (K(M, M^t)u_1 + eb_1) \geq e\epsilon_1 - \zeta_1, \quad \zeta_1 \geq 0, \end{aligned}$$

and

$$\begin{aligned} \min_{u_2, b_2, \zeta_2} \quad & \frac{1}{2} \|Y + e\epsilon_2 - (K(M, M^t)u_2 + eb_2)\|^2 + C_2 e^t \zeta_2 \\ \text{such that} \quad & (K(M, M^t)u_2 + eb_2) - Y \geq e\epsilon_2 - \zeta_2, \quad \zeta_2 \geq 0, \end{aligned}$$

where ζ_1 and ζ_2 are the slack vectors; C_1, C_2 are positive regularization parameters. Thus, by solving two smaller-sized QPPs, TSVR demonstrates greater computational efficiency compared to SVR.

For an unseen data point, the target is given by the function

$$r(x) = \frac{1}{2}(r_1(x) + r_2(x)) = \frac{1}{2}(u_1 + u_2)^t K(M, x) + \frac{1}{2}(b_1 + b_2). \quad (2.55)$$

2.8 Statistical test

In this section, we present the statistical tests conducted to assess the performance of the models. The following tests are performed:

2.8.1 Friedman's test

Friedman test has been proven to be a robust approach to checking the statistical significance among classifiers and has been practiced by numerous researchers [88, 123]. This method ranks the algorithms for each dataset separately, the best-performing algorithm on each dataset achieves the lower rank. Then, the average of the rank across all the datasets is

taken as the rank of the classifier. Let r_i^j be the rank of the j^{th} algorithm on the i^{th} dataset among md algorithms and N datasets. The Friedman test compares the average rank of algorithms, $R_j = \frac{1}{N} \sum_i r_i^j$. Friedman's test has a null hypothesis assumption of equivalence among the models. The statistic of Friedman is calculated as

$$\chi_F^2 = \frac{12N}{md(md+1)} \left[\sum_j R_j^2 - \frac{md(md+1)^2}{4} \right], \quad (2.56)$$

where md and N correspond to the count of total models and total datasets, respectively. Further, F_F statistic is calculated to overcome the conservative nature of the Friedman statistic. The following equation is utilized to calculate the F_F statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(md-1) - \chi_F^2}. \quad (2.57)$$

The degree of freedom corresponding to χ_F^2 and F_F statistics are $(md-1)$ and $(md-1, (md-1)(N-1))$, respectively. Further, the Nemenyi post hoc test is utilized which is a non-parametric statistical test used for multiple comparisons following a significant result from the Friedman test. The Nemenyi test helps identify which groups or treatments differ significantly from one another by calculating the critical difference (cd) between each pair of models using

$$cd = q_{0.05} \sqrt{\frac{md(md+1)}{6N}}. \quad (2.58)$$

If the calculated cd is greater than the difference in average ranks of the concerned models, then the models are significantly different.

2.8.2 Wilcoxon signed rank test

The Wilcoxon test examines the assumption of the null hypothesis, i.e., equivalence of two models. This test assigns ranks to the dissimilarity in performance metrics of two models based on their magnitudes. The differences are ranked according to their absolute values; average ranks are assigned in case of ties. Let R^+ be the sum of positive ranks, and R^- be the sum of negative ranks. For a pair of models, if the obtained p -value is less than 0.05, the null hypothesis is rejected.

2.8.3 Win-tie-loss sign test

The win-Tie-Loss test is a well-known approach for comparing classifiers' overall performance, where we tally the wins of the proposed algorithm on different datasets. In Win-Tie-Loss test, a pair of classifiers are assumed to be equivalent if null hypothesis is true, i.e., each classifier wins around $\frac{N}{2}$ out of N datasets. At 5% level of significance, a classifier is significantly better if its number of wins is greater than or equal to

$$N/2 + 1.96\sqrt{N}/2. \quad (2.59)$$

The number of ties should be shared equally between the classifiers if the number of ties is even, otherwise, we ignore one and distribute the remaining equally between the models.

Chapter 3

Enhanced Pinball Loss Frameworks for Robust Classification and Generalization

This chapter explores advanced frameworks utilizing pinball loss variants to address challenges in classification tasks. First, a novel flexible pinball loss function is introduced as an extension of the traditional pinball loss, allowing the parameters τ_1 and τ_2 to range from $[-1, 1]$ while preserving convexity. This novel loss function improves performance by applying distinct loss values to correctly and incorrectly classified samples. The integration of this flexible pinball loss function into the SVM framework leads to the development of the flexible pinball loss SVM (FP-SVM) [76]. Designed to improve resilience against noise and misclassification errors, FP-SVM leverages the flexibility of τ_1 and τ_2 to optimize performance without sacrificing convexity.

Building on the concept of pinball loss, the chapter further introduces the sparse version of Pin-GTSVM, known as the universum twin support vector machine with truncated pinball loss (Tpin-UTWSVM) [82]. This model enhances the sparsity and efficiency of Pin-GTSVM by incorporating a truncated version of the pinball loss. Additionally, it integrates universum data to provide prior knowledge of data distributions, thereby enhancing generalization. The truncated pinball loss ensures an optimal balance between robustness

and sparsity, addressing noise insensitivity while maintaining computational scalability. Together, these models exemplify the versatility of pinball loss variants in developing robust, and generalizable classification frameworks.

3.1 Proposed flexible pinball loss SVM

This section highlights the formulation of flexible pinball loss, its incorporation with SVM and analyze the results of numerical experiments. Pin-SVM [17] appears to possess most of the desirable properties of a classification model, including noise insensitivity and stability to re-sampling. However, the loss parameter τ has choice only to take value between 0 and 1, which affects the accuracy of the model. Taking inspiration from pin-SVM $_{\tau \in [-1, 1]}$ [73], we attempt to control the penalty for incorrectly and correctly classified data points using two parameters, τ_1 and τ_2 respectively, and allowing them to take negative values for enhancing the model's accuracy. The parameters τ_2 and τ_1 take values from $[-1, 1]$ and $[-\tau_2, 1]$, respectively. If the parameter τ_1 takes values in the range of $[-1, 1]$, the optimization problem may no longer be convex and would not lead to a global solution. Geometrically, it can be observed from Figure 3.1 that if we restrict the τ_1 values between $-\tau_2$ and 1, the proposed problem remains convex.

3.1.1 Proposed flexible pinball loss

Given the preceding discussion, we propose a novel flexible pinball loss support vector machine (FP-SVM) for classification problems. We first define the flexible pinball loss function (see Figures (3.2), (3.3)):

$$L_{\tau_1, \tau_2}(u) = \begin{cases} \tau_1 u, & u \geq 0, \\ -\tau_2 u, & u < 0, \end{cases} \quad (3.1)$$

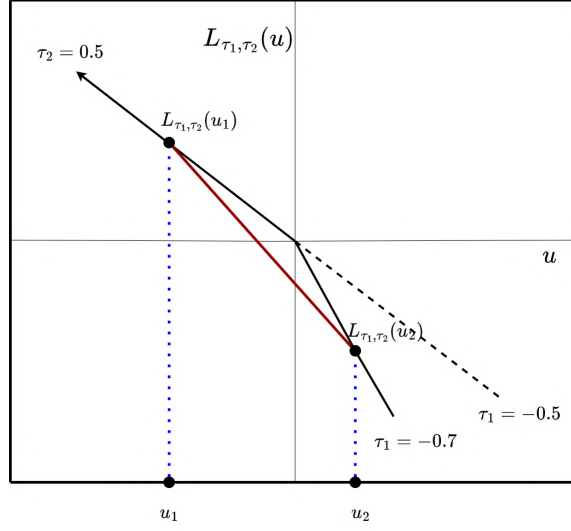


Figure 3.1: Representation of the flexible pinball loss function with fixed $\tau_2 = 0.5$ and different values of τ_1 . The brown line depicts the case of $\tau_1 = -0.7 < -0.5$ which leads to non-convexity of the function as $\exists u_1$ and u_2 such that $L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) > \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2)$, $\beta \in [0, 1]$.

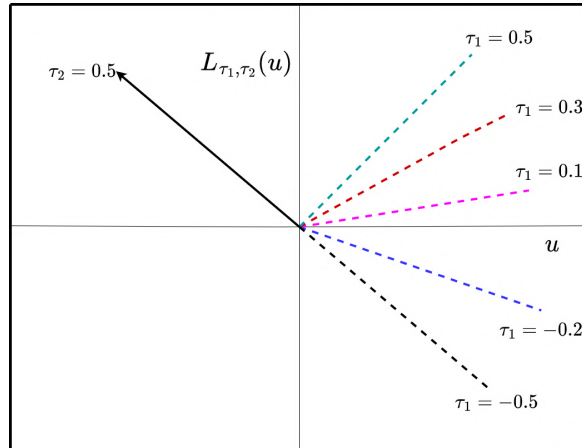


Figure 3.2: Flexible pinball loss for fixed $\tau_2 = 0.5$ and different τ_1 values between -0.5 and 1 .

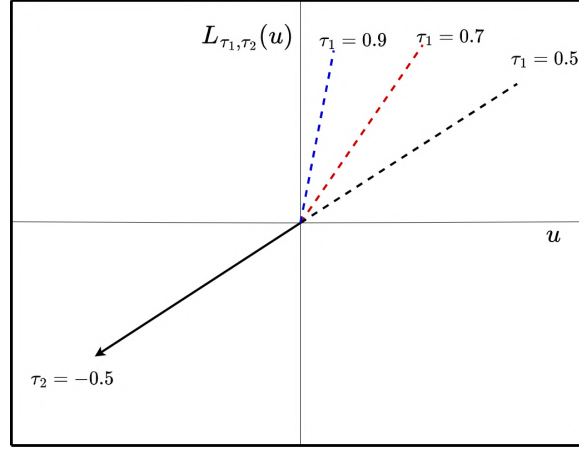


Figure 3.3: Flexible pinball loss for fixed $\tau_2 = -0.5$ and different τ_1 values between 0.5 and 1.

where $\tau_2 \in [-1, 1]$, $\tau_1 \in [-\tau_2, 1]$ and $u = 1 - y_j f(x_j)$, $\forall j = 1, 2, \dots, l$. The flexible pinball loss function can be equivalently stated as:

$$L_{\tau_1, \tau_2}(u) = \max\{\tau_1 u, -\tau_2 u\}, \quad (3.2)$$

for the same range of τ_2 and τ_1 as above. Figures (3.2) and (3.3) show the graph of $L_{\tau_1, \tau_2}(u)$ with respect to u for $\tau_2 = 0.5$ and $\tau_2 = -0.5$, respectively, with different τ_1 values.

3.1.2 Formulation of proposed flexible pin-SVM

By seamlessly incorporating the proposed flexible pinball loss to SVM, we proposed flexible pinball loss SVM (FP-SVM) [76]. The optimization problem corresponding to non-linear case of FP-SVM is defined as follows:

$$\begin{aligned} \min_{w, b, \zeta} \quad & \frac{1}{2} \|w\|_2^2 + C_1 \sum_{j=1}^l \zeta_j \\ \text{subject to} \quad & \zeta_j \geq \tau_1 (1 - y_j (w^t \psi(x_j) + b)), \\ & \zeta_j \geq -\tau_2 (1 - y_j (w^t \psi(x_j) + b)), \end{aligned} \quad (3.3)$$

where $C_1 > 0$ signifies the regularization parameter and ψ represents the kernel map from the input space to higher dimensional space. Assume $K(x_j, x_k) = \psi(x_j)^t \psi(x_k)$. To formulate the Wolfe dual of primal problem (3.3), we obtain the Lagrangian function as follows:

$$L(w, b, \zeta_j, \gamma_j, \eta_j) = \frac{1}{2} \|w\|^2 + C_1 \sum_{j=1}^l \zeta_j - \sum_{j=1}^l \gamma_j (\zeta_j - \tau_1 (1 - y_j (w^t \psi(x_j) + b))) - \sum_{j=1}^l \eta_j (\zeta_j + \tau_2 (1 - y_j (w^t \psi(x_j) + b))), \quad (3.4)$$

where $\gamma_j, \eta_j: j = 1, 2, \dots, l$ are the non-negative Lagrangian multipliers. The Karush-Kuhn-Tucker (K.K.T.) conditions of optimality for (3.4) are given by:

$$\begin{aligned} \frac{\partial L}{\partial w} &= w - \sum_{j=1}^l \tau_1 \gamma_j y_j \psi(x_j) + \sum_{j=1}^l \tau_2 \eta_j y_j \psi(x_j) = 0 \\ \implies w - \sum_{j=1}^l (\tau_1 \gamma_j - \tau_2 \eta_j) y_j \psi(x_j) &= 0, \end{aligned} \quad (3.5)$$

$$\begin{aligned} \frac{\partial L}{\partial b} &= - \sum_{j=1}^l \tau_1 \gamma_j y_j + \sum_{j=1}^l \tau_2 \eta_j y_j = 0 \\ \implies \sum_{j=1}^l (\tau_1 \gamma_j - \tau_2 \eta_j) y_j &= 0, \end{aligned} \quad (3.6)$$

$$\frac{\partial L}{\partial \zeta_j} = C_1 - \gamma_j - \eta_j = 0, \quad \forall j = 1, 2, \dots, l. \quad (3.7)$$

Using the above conditions, the Wolfe dual of problem (3.3) is obtained as:

$$\begin{aligned} \min_{(\gamma, \eta)} \quad & \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l (\tau_1 \gamma_k - \tau_2 \eta_k) (\tau_1 \gamma_j - \tau_2 \eta_j) y_j y_k K(x_j, x_k) - \sum_{j=1}^l (\tau_1 \gamma_j - \tau_2 \eta_j) \\ \text{subject to} \quad & \sum_{j=1}^l (\tau_1 \gamma_j - \tau_2 \eta_j) y_j = 0, \\ & C_1 - \gamma_j - \eta_j = 0, \\ & \gamma_j \geq 0, \eta_j \geq 0, \quad \forall j = 1, 2, \dots, l. \end{aligned} \quad (3.8)$$

If we replace the variable $\gamma := |\tau_1|\gamma$ and $\eta := |\tau_2|\eta$ in dual problem (3.8) and define a function $s_\tau = \begin{cases} 1, & \text{if } \tau \geq 0, \\ -1, & \text{if } \tau < 0, \end{cases}$ then the dual problem can be expressed as:

$$\begin{aligned} \min_{(\gamma, \eta)} \quad & \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l (s_{\tau_1} \gamma_k - s_{\tau_2} \eta_k) (s_{\tau_1} \gamma_j - s_{\tau_2} \eta_j) y_j y_k K(x_j, x_k) - \sum_{j=1}^l (s_{\tau_1} \gamma_j - s_{\tau_2} \eta_j) \\ \text{subject to} \quad & \sum_{j=1}^l (s_{\tau_1} \gamma_j - s_{\tau_2} \eta_j) y_j = 0, \\ & C_1 - \frac{\gamma_j}{|\tau_1|} - \frac{\eta_j}{|\tau_2|} = 0, \\ & \gamma_j \geq 0, \eta_j \geq 0, \forall j = 1, 2, \dots, l, \end{aligned} \quad (3.9)$$

where $\gamma_j, \eta_j : j = 1, 2, \dots, l$ are Lagrangian multipliers.

After finding the solution of the dual (3.9), we obtain the value of w as follows:

$$w = \sum_{j=1}^l (s_{\tau_1} \gamma_j - s_{\tau_2} \eta_j) y_j \psi(x_j). \quad (3.10)$$

Now, we consider SVs and find the value of b according to the complementary slackness conditions as follows:

$$\begin{aligned} b &= y_j - w^t \psi(x_j) \\ &= y_j - \sum_{k=1}^l (s_{\tau_1} \gamma_k - s_{\tau_2} \eta_k) y_k K(x_k, x_j). \end{aligned} \quad (3.11)$$

We find the final value of b by taking the average of all possible values of b 's obtained. Then, for any test point $x \in \mathbb{R}^n$, the decision function is given by:

$$\begin{aligned} f(x) &= \text{sign}(w^t \psi(x) + b) \\ &= \text{sign} \left(\sum_{k=1}^l (s_{\tau_1} \gamma_k - s_{\tau_2} \eta_k) y_k K(x_k, x) + b \right). \end{aligned} \quad (3.12)$$

The proposed FP-SVM possesses the properties of related state-of-the-art models like insensitivity to noise, and stability under re-sampling, and also enhances the generalization performance. In the proposed FP-SVM, we expand the parameters τ_1 and τ_2 to negative values and develop an algorithm that travels the maximum possible solution path so that the optimization problem remains convex and gets a global solution to the problem.

3.1.2.1 Convexity of proposed flexible pinball loss

The convexity of a loss function in SVM is an important trait and viewed as highly preferable because of their computational advantages, including unique optima, ease of use, capability to be effectively enhanced by convex optimization tools, and so forth. In this subsection, we prove that the proposed flexible pinball loss is convex.

Theorem 1. *The flexible pinball loss $L_{\tau_1, \tau_2}(u)$ in equation (3.1) is convex $\forall \tau_2 \in [-1, 1]$ and $\tau_1 \in [-\tau_2, 1]$. In other words, $\forall u_1, u_2 \in \mathbb{R}$ and $\beta \in [0, 1]$, the following holds:*

$$L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) \leq \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2). \quad (3.13)$$

Proof. Without loss of generality, let us assume $u_1 \leq u_2$. Now, two possible cases arises: case (I) $u_1 \geq 0$ and case (II) $u_1 < 0$.

Case (I): $u_1 \geq 0$

$$\implies u_2 \geq 0,$$

$$\implies (\beta u_1 + (1 - \beta)u_2) \geq 0,$$

$$\implies L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) = \tau_1(\beta u_1 + (1 - \beta)u_2).$$

$$\text{Also, } \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2) = \beta \tau_1 u_1 + (1 - \beta)\tau_1 u_2.$$

Thus, when $u_1 \geq 0$, inequality (3.13) holds.

Case (II): $u_1 < 0$

Again, we have 2 sub-cases: $u_2 < 0$ and $u_2 \geq 0$.

Subcase (a): $u_1 < 0$ and $u_2 < 0$.

$$\implies (\beta u_1 + (1 - \beta)u_2) < 0,$$

$$\implies L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) = -\tau_2(\beta u_1 + (1 - \beta)u_2),$$

$$\text{Also, } \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2) = -\tau_2\beta u_1 - \tau_2(1 - \beta)u_2.$$

Hence, when $u_1 < 0$ and $u_2 < 0$, inequality (3.13) holds.

Subcase (b): $u_1 < 0$ and $u_2 > 0$.

Two possibility arises here, $(\beta u_1 + (1 - \beta)u_2) < 0$ or ≥ 0 .

First, let us take $(\beta u_1 + (1 - \beta)u_2) < 0$.

$$\implies L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) = -\tau_2(\beta u_1 + (1 - \beta)u_2),$$

$$\text{And, } \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2) = -\tau_2\beta u_1 + \tau_1(1 - \beta)u_2.$$

$$\text{Also, } -\tau_2(1 - \beta)u_2 \leq \tau_1(1 - \beta)u_2 \text{ as } \tau_1 \in [-\tau_2, 1].$$

Hence, when $u_1 < 0$, $u_2 > 0$ and $(\beta u_1 + (1 - \beta)u_2) < 0$, inequality (3.13) holds, i.e.,

$$L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) \leq \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2).$$

Second, consider the last possibility: $u_1 < 0$, $u_2 > 0$ and $(\beta u_1 + (1 - \beta)u_2) \geq 0$.

$$\implies L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) = \tau_1(\beta u_1 + (1 - \beta)u_2),$$

$$\text{and } \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2) = -\tau_2\beta u_1 + \tau_1(1 - \beta)u_2.$$

$$\text{Also, } -\tau_1\beta u_1 \leq \tau_2\beta u_1 \text{ as } \tau_1 \in [-\tau_2, 1] \text{ and } u_1 < 0.$$

Thus, for $u_1 < 0$, $u_2 > 0$ and $(\beta u_1 + (1 - \beta)u_2) \geq 0$,

$$L_{\tau_1, \tau_2}(\beta u_1 + (1 - \beta)u_2) \leq \beta L_{\tau_1, \tau_2}(u_1) + (1 - \beta)L_{\tau_1, \tau_2}(u_2).$$

Therefore, the inequality (3.13) holds for all the possible cases, hence the proposed flexible pinball loss function $L_{\tau_1, \tau_2}(u)$ is convex $\forall \tau_2 \in [-1, 1]$ and $\tau_1 \in [-\tau_2, 1]$.

3.1.2.2 Noise insensitivity

Huang et al. [17] explained that the pinball loss is insensitive to noise near the optimal boundary due to the penalty on correctly classified samples. Similar to pinball loss, the proposed flexible pinball loss also gives the penalty on correctly classified samples to make it insensitive towards noise. In this subsection, we prove theoretically that the proposed flexible pinball loss is insensitive to noise near the optimal boundary. For simplicity, we explain for the case of linear classifier. Define a generalized sign function $\text{sign}_{\tau_1, \tau_2}(u)$ as follows:

$$\text{sign}_{\tau_1, \tau_2}(u) = \begin{cases} \tau_1, & u > 0, \\ [-\tau_2, \tau_1], & u = 0, \\ -\tau_2, & u < 0. \end{cases} \quad (3.14)$$

$\text{sign}_{\tau_1, \tau_2}(u)$ is the subgradient of the flexible pinball loss (equation (3.1)). Now, the optimality condition for equation (3.3) can be expressed as:

$$\frac{w}{C_1} - \sum_{j=1}^l \text{sign}_{\tau_1, \tau_2}(1 - y_j(w^T x_j + b)) y_j x_j = \mathbf{0}, \quad (3.15)$$

where $\mathbf{0}$ denotes the vector of which all the components equal to zero. For given w and b , we partitioned the index set into three sets, S^+ , S^- , and S^0 as follows:

$$\begin{aligned} S^+ &= \{j: 1 - y_j(w^T x_j + b) > 0\}, \\ S^- &= \{j: 1 - y_j(w^T x_j + b) < 0\}, \\ S^0 &= \{j: 1 - y_j(w^T x_j + b) = 0\}, \end{aligned}$$

where S^+ , S^- , S^0 represent the sets that contain the misclassified samples, correctly classified samples, and samples that lie on the decision boundary, respectively. Using these notations, the optimality condition, equation (3.15), can be written as follows:

$$\frac{w}{C_1} - \tau_1 \sum_{j \in S^+} y_j x_j + \tau_2 \sum_{j \in S^-} y_j x_j - \chi_j \sum_{j \in S^0} y_j x_j = 0, \quad (3.16)$$

where $\chi_j \in [-\tau_2, \tau_1]$.

The equation (3.16) expresses that τ_1 and τ_2 control the number of samples in S^+ and S^- . When both τ_1 and τ_2 equal to 1, the sets S^+ and S^- contain many points, and hence the result is insensitive to feature noise. The reason is that as the value of τ_2 increases, the number of samples in S^- increases, and consequently, the model is less sensitive towards noise. When τ_2 is small, there are fewer samples in S^+ , and hence the result is sensitive towards noise. More specifically, as the value of τ_2 decreases, the number of samples in S^+ also decreases, and hence the result is less sensitive to noise. In SVM, $\tau_2 = 0$ which leads to small number of samples in S^+ . Therefore, the noise around the optimal boundary will remarkably influence the performance of SVM. Hence, similar to pinball loss, the classifier obtained by the proposed flexible pinball loss is also less sensitive to the classifier obtained using hinge loss.

This serves as a qualitative tool to explain the association between τ_1 and τ_2 with the noise sensitivity of the proposed FP-SVM model. Its primary purpose is to highlight the noise insensitivity inherent in our approach, akin to pin-SVM [17]. Furthermore, it is crucial to note that the proposed FP-SVM stands as a generalization of pin-SVM. Specifically, when $\tau_1 = 1$ and $\tau_2 \in [0, 1]$, FP-SVM seamlessly reduces to pin-SVM, emphasizing the versatility and adaptability of our proposed model.

Further, to address the need for practical implications, we conducted comprehensive experiments on various datasets, including an artificial dataset, UCI and KEEL datasets, and the BreakHis dataset.

3.1.3 Experimental results

This section contains the experimental setup used to evaluate the models and examine the results of the proposed FP-SVM and the baseline models, including SVM [77], pin-SVM [17], unified pin-SVM [78], QN-SSVM [124], and intuitionistic fuzzy random vector functional link neural network (IFRVFL) [125].

3.1.3.1 Experimental setup

To analyze the proposed FP-SVM performance, we used publicly available UCI [126] and KEEL [127] benchmark datasets. The experiments are carried out on a system equipped with MATLAB R2023a, Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, 2904 Mhz, 8 Core(s), 16 Logical processor(s), and 16 GB of RAM. We initially divided the datasets into a 70:30 ratio for training and testing purposes. Within the 70% data designated for training, we employed a five-fold cross-validation approach with grid search, exploring various parameter ranges. In each iteration of the five-fold cross-validation, we divided the 70% data into five subsets, using four for training and one for validation. This process is repeated,

ensuring each subset serves as the validation set once. Taking the average of the accuracies with different validation subsets, we obtain the validation accuracy. Consequently, we obtained a set of optimal parameters associated with the highest validation accuracy. These optimal parameters are then applied to the testing data, and the resulting testing accuracy is obtained. We used quadprog function of MATLAB to solve the dual QPP (3.9) of the proposed FP-SVM. We used RBF kernel, i.e., $K(x_j, x_k) = \exp(-\|x_j - x_k\|^2 / \sigma^2)$ to execute the case of non linearity. We also conducted experiments with polynomial ($K(x_j, x_k) = (x_j^t x_k + 1)^p$) and linear kernels ($K(x_j, x_k) = x_j^t x_k$) to check the performance of the proposed FP-SVM with different kernels. For the proposed FP-SVM and each baseline model, we set the parameters $C_1 = [10^{-5}, 10^{-4}, \dots, 10^4, 10^5]$, and $\sigma = [10^{-5}, 10^{-4}, \dots, 10^4, 10^5]$. For FP-SVM, the range for τ_2 and τ_1 are $[-1, -0.9, -0.8, \dots, 0.8, 0.9, 1]$, and $[-\tau_2, -\tau_2 + 0.1, -\tau_2 + 0.2, \dots, 1]$, respectively. For unified pin-SVM and pin-SVM models, the range for τ are $[-1, -0.9, -0.8, \dots, 0.8, 0.9, 1]$, and $[0.2, 0.4, 0.5, 0.8, 1]$ [17], respectively. For QN-SSVM, the range for different parameters τ_2 , τ_1 , δ_2 , and δ_1 are taken same as in [124]. For IFRVFL, the number of hidden nodes N_h is chosen from 3 to 203 with step size 20 [128]. For the polynomial kernel, we tuned p in the range $[1, 2, 3, 4, 5]$.

We used the accuracy metric to evaluate the performance of the proposed FP-SVM, along with the training time of the models. The accuracy is calculated by the ratio between the correctly classified samples and the total number of samples, i.e.,

$$\text{Accuracy} = \frac{\text{Number of correctly classified samples}}{\text{Total number of samples}}.$$

3.1.3.2 Artificial dataset

To compare the proposed FP-SVM with baseline models, we performed the experiments on 2-D artificial dataset [17], in which instances come from two Gaussian distributions with

Table 3.1: Performance comparison of the models on a 2-D artificial dataset using RBF kernel.

Noise level →	0	0.05	0.1
Model ↓	Accuracy	Accuracy	Accuracy
proposed FP-SVM	99.97	97.33	94.70
QN-SSVM [124]	96.4	94.65	92.24
unified pin-SVM [78]	93.367	97.33	93.54
pin-SVM [17]	93.37	96.01	93.52
SVM [90]	92.87	94.69	92.20

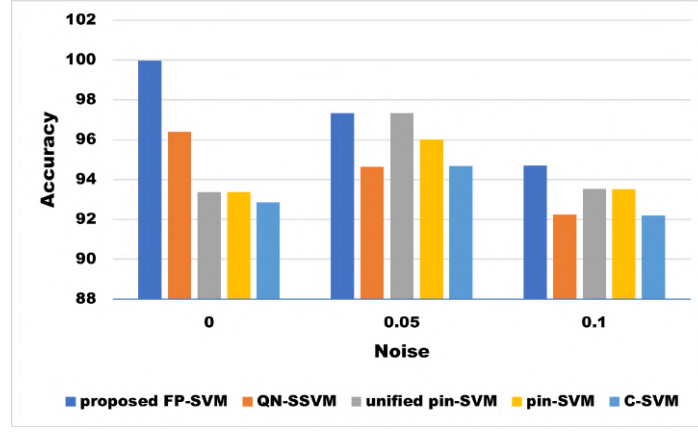


Figure 3.4: Bar graph representing the accuracy of models on an artificial dataset.

equal probability, $N(\mu_+, \Sigma_+)$ and $N(\mu_-, \Sigma_-)$. Here, $\mu_+ = [0.5, -3]^t$, $\mu_- = [-0.5, 3]^t$ and $\Sigma_+ = \Sigma_- = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}$. Now, we consider noisy points in the training dataset which follows Gaussian distribution with mean $[0, 0]^t$ and covariance matrix $\begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$. For the comparison of the proposed FP-SVM with baseline models, we generate 50 artificial data points for each class and analyzed three cases with the ratio of noise data being 0, 0.05, and 1. We repeat the sampling and training process 100 times. Table 3.1 represents the average accuracy of the proposed FP-SVM with the baseline models SVM, pin-SVM, unified pin-SVM, and QN-SSVM over the artificially generated 2-D dataset. Figure 3.4 represents the bar graph corresponding to the obtained accuracy on the artificial dataset.

3.1.3.3 Experimental outcomes and statistical comparison on UCI and KEEL datasets

Table 3.2: Classification performance of the proposed FP-SVM and baseline models on UCI and KEEL binary datasets with RBF kernel.

Dataset	SVM [90] (Accuracy, Time(s)) (C_1, σ)	pin-SVM [17] (Accuracy, Time(s)) (C_1, τ_1, σ)	unified pin-SVM [78] (Accuracy, Time(s)) (C_1, τ_1, σ)	QN-SSVM [124] (Accuracy, Time(s)) ($C_1, \tau_1, \tau_2, \delta_1, \delta_2, \nu, \sigma$)	IFRVFL [125] (Accuracy, Time(s)) (C_1, N_k, σ)	proposed FP-SVM (Accuracy, Time(s)) ($C_1, \tau_2, \tau_1, \sigma$)
breast-cancer-wisc (699 × 9)	(95.22, 0.1116) (1, 1)	(96.17, 0.9445) (1, 0.4, 1)	(95.69, 0.6358) (0.00001, -0.8, 1)	(93.3, 0.0264) (0.0001, 0.6, 0.75, 0.1, 0.5, 0.1, 1)	(95.69, 0.0277) (0, 3, 0.00001)	(95.69, 0.0877) (0.0001, 0.1, 1)
breast-cancer-wisc-prog (198 × 33)	(74.58, 0.0124) (0.00001, 0.00001)	(74.58, 0.0377) (0.00001, 0.2, 0.00001)	(55.93, 0.0345) (0.00001, -0.7, 0.1)	(74.58, 0.0623) (0.01, 0.75, 0.75, 0.25, 0.5, 0.5, 10)	(72.88, 0.0029) (1000, 23, 100000)	(76.27, 0.009) (0.00001, 0.9, 0.01)
congressional-voting (435 × 16)	(60, 0.0142) (10, 0.1)	(60, 0.2046) (1, 0.2, 0.1)	(60, 0.0175) (1, 0, 0.1)	(60.77, 0.0065) (0.00001, 0.75, 0.1, 0.05, 1.5, 0.9, 0.1)	(62.31, 0.006) (0.01, 23, 10)	(60.77, 0.0195) (1, 0.1, 0.0.1)
cylinder-bands (512 × 35)	(60.78, 0.0141) (1, 0.1)	(60.78, 0.1854) (1, 0.2, 0.1)	(57.52, 0.1232) (1, -0.8, 1)	(41.18, 0.214) (10000, 0.75, 0.4, 0.45, 1, 0.9, 10)	(69.93, 0.0095) (0.1, 3, 100000)	(69.28, 0.0179) (100000, 0.2, 2, 1)
dermatology (366 × 35)	(83.49, 0.0416) (10, 1)	(83.49, 0.1237) (10, 0.2, 1)	(97.25, 0.0934) (0.00001, -0.9, 1)	(83.49, 0.0068) (0.001, 0.5, 0.75, 0.45, 1.75, 0.1, 1)	(97.25, 0.0072) (0.01, 203, 100000)	(98.17, 0.0365) (10, 0, 0.9, 1)
fertility (100 × 9)	(90, 0.003) (0.00001, 0.00001)	(90, 0.0111) (0.00001, 0.2, 0.00001)	(93.33, 0.0133) (1, -0.2, 1)	(90, 0.0005) (0.0001, 0.5, 0.75, 0.05, 0.5, 0.1, 1)	(66.67, 0.0011) (1000, 3, 1)	(90, 0.0168) (10, 0.8, 0.1, 1)
haberman-survival (306 × 3)	(69.23, 0.0066) (1, 10)	(70.33, 0.1628) (1, 0.2, 10)	(69.23, 0.2143) (0.00001, -0.6, 1)	(60.44, 0.0831) (0.001, 0.7, 0.1, 0.45, 1.75, 0.5, 1)	(70.33, 0.0027) (10, 103, 10)	(74.73, 0.0108) (0.001, 0.9, 0.1)
hepatitis (155 × 19)	(71.74, 0.0021) (0.00001, 0.00001)	(71.74, 0.022) (0.00001, 0.2, 0.00001)	(71.74, 0.0139) (0.00001, -0.2, 0.1)	(71.74, 0.0326) (1, 0.5, 0.1, 0.01, 0.5, 0.01, 10)	(84.78, 0.0019) (100000, 3, 0.00001)	(71.74, 0.0025) (100000, 0.9, 0.00001)
mammographic (961 × 5)	(81.25, 0.0594) (1, 1)	(80.56, 2.712) (1, 0.2, 1)	(81.25, 0.0744) (1, 0, 1)	(78.13, 0.0282) (0.00001, 1, 0.1, 0.5, 1, 0.01, 1)	(79.86, 0.0189) (0.1, 163, 100000)	(81.25, 0.0616) (10, 0.1, 0.1)
monks-1 (556 × 6)	(62.05, 0.0803) (1, 0.1)	(62.05, 0.5514) (1, 0.2, 0.1)	(79.52, 0.7073) (10, -1, 1)	(54.22, 0.4037) (100, 0.5, 0.75, 0.25, 1.5, 0.9, 10)	(78.92, 0.0055) (0.01, 143, 100000)	(76.51, 0.0098) (0.1, 0, 0.8, 1)
planning (182 × 12)	(75.93, 0.0024) (0.00001, 0.00001)	(75.93, 0.024) (0.00001, 0.2, 0.00001)	(75.93, 0.0535) (100000, -0.4, 0.1)	(75.93, 0.0009) (0.00001, 1, 0.75, 0.1, 1.5, 0.01, 1)	(64.81, 0.0015) (10000, 3, 100)	(75.93, 0.2767) (0.0001, 0.7, 0.7, 0.1)
statlog-australian-credit (690 × 14)	(66.67, 0.029) (1, 10)	(66.67, 0.918) (1, 0.2, 10)	(66.67, 0.3044) (0.00001, -1, 0.00001)	(66.67, 0.5407) (0.1, 0.6, 0.75, 0.45, 1, 0.1, 1)	(65.7, 0.0105) (0, 3, 100000)	(66.67, 0.4351) (0.1, 0.6, 0.3, 1)
statlog-heart (270 × 13)	(81.48, 0.0058) (1, 0.1)	(81.48, 0.0467) (1, 0.2, 0.1)	(80.25, 0.0402) (1, -0.7, 0.1)	(51.85, 0.0693) (0.01, 0.5, 0.1, 0.45, 1, 0.9, 10)	(77.78, 0.0039) (10000, 103, 0.0001)	(81.48, 0.0639) (0.00001, 0.7, 0.7, 0.1)
trains (10 × 29)	(33.33, 0.0011) (0.00001, 0.00001)	(33.33, 0.0023) (0.00001, 0.2, 0.00001)	(33.33, 0.0033) (0.1, -0.9, 10)	(66.67, 0.0075) (0.00001, 0.5, 0.1, 0.01, 0.5, 0.1, 0.00001)	(66.67, 0.0039) (0, 23, 100000)	(66.67, 0.0039) (0.00001, 0.2, 0.9, 0.00001)
vertebral-column-2classes (310 × 6)	(74.19, 0.0061) (100, 1)	(68.82, 0.122) (100000, 1, 10)	(66.67, 0.6544) (0.0001, -0.5, 1)	(65.59, 0.0877) (10, 0.5, 0.4, 0.1, 1.5, 0.9, 1)	(33.33, 0.0044) (0, 63, 1)	(66.67, 0.0034) (0.01, 0, 0.2, 1)
crossplane130 (130 × 2)	(97.44, 0.0755) (0.1, 1)	(97.44, 0.0943) (0.1, 0.2, 1)	(97.44, 0.3473) (0.01, -1, 10)	(97.44, 0.0126) (0.0001, 1, 0.4, 0.01, 1.5, 0.5, 0.1)	(94.87, 0.0012) (1, 163, 10)	(97.44, 0.0692) (0.00001, 0.3, 2.78E - 17, 0.0001)
ecoli-0-1-4-6_vs_5 (280 × 6)	(95.24, 0.0146) (1, 0.01)	(95.24, 0.0618) (1, 0.2, 0.01)	(94.05, 0.0598) (0.00001, -0.5, 0.01)	(92.86, 0.0055) (0.00001, 0.5, 0.1, 0.01, 0.5, 0.01, 0.00001)	(100, 0.0031) (100000, 123, 0.00001)	(94.05, 0.0422) (0.00001, 0.8, 0.8, 0.01)
ecoli-0-1-4-7_vs_5-6 (332 × 6)	(94.95, 0.0117) (1, 0.01)	(94.95, 0.0799) (1, 0.2, 0.01)	(96.97, 0.2126) (0.00001, -0.9, 1)	(92.93, 0.0038) (0.00001, 0.5, 0.1, 0.01, 0.5, 0.01, 0.00001)	(97.98, 0.0026) (100000, 63, 0.00001)	(96.97, 0.0311) (1, 1, -0.9, 0.1)
ecoli-0-1_vs_2-3-5 (244 × 7)	(90.41, 0.0094) (10, 0.1)	(90.41, 0.0358) (10, 0.2, 0.1)	(87.67, 0.0502) (0.1, -0.4, 0.1)	(87.67, 0.0028) (0.0001, 1, 0.75, 0.45, 1, 0.5, 10)	(95.89, 0.0034) (0.1, 143, 100000)	(95.89, 0.9466) (0.00001, 0.1, 0.1, 0.1)
ecoli-0-1_vs_5 (240 × 6)	(94.44, 0.0405) (1, 0.01)	(94.44, 0.0733) (1, 0.2, 0.01)	(95.83, 0.072) (0.00001, -0.9, 0.1)	(87.5, 0.007) (0.00001, 0.5, 0.75, 0.25, 1, 0.5, 10)	(98.61, 0.0025) (0.01, 203, 1000)	(95.83, 0.0355) (0.00001, 0.1, 0.1, 0.1)
ecoli-0-2-6-7_vs_3-5 (224 × 7)	(85.07, 0.0045) (10, 0.1)	(85.07, 0.0271) (10, 0.2, 0.1)	(82.09, 0.0327) (0.001, -0.1, 0.1)	(82.09, 0.0015) (0.00001, 0.5, 0.4, 0.25, 1.5, 0.5, 10)	(91.04, 0.0023) (1, 23, 1000)	(94.03, 0.7198) (1, 0.5, 0.5, 0.1)
ecoli-0-3-4-6_vs_5 (205 × 7)	(95.08, 0.0028) (10, 0.01)	(95.08, 0.0221) (10, 0.2, 0.01)	(95.08, 0.0377) (0.00001, -0.6, 0.01)	(86.89, 0.0523) (1, 0.7, 0.75, 0.6, 1.5, 0.1, 100)	(93.44, 0.0018) (1000, 83, 10000)	(95.08, 0.0047) (0.00001, 0.4, 0.4, 0.1)
ecoli-0-3-4-7_vs_5-6 (257 × 7)	(96.1, 0.0029) (1, 0.01)	(96.1, 0.0472) (1, 0.2, 0.01)	(98.7, 0.0245) (0.1, -0.7, 0.1)	(93.51, 0.0022) (0.0001, 0.7, 0.75, 0.05, 1, 0.01, 10)	(97.4, 0.0031) (10000, 183, 0.00001)	(97.4, 0.0036) (1, 0, 1, 0.1)
ecoli-0-6-7_vs_3-5 (222 × 7)	(96.97, 0.0028) (10, 0.01)	(96.97, 0.032) (10, 0.2, 0.01)	(96.97, 0.0204) (1, -0.6, 0.1)	(93.94, 0.0023) (0.00001, 1, 0.1, 0.25, 1.75, 0.5, 10)	(96.97, 0.0022) (100000, 103, 0.00001)	(96.97, 0.0023) (0.01, 0, 0.5, 0.01)
ecoli0137vs26 (311 × 7)	(92.47, 0.0065) (1, 10)	(91.4, 0.1598) (100000, 0.8, 1)	(93.55, 0.0182) (100, 0, 1)	(92.47, 0.002) (0.00001, 0.75, 0.75, 0.25, 1, 0.1, 0.1)	(95.7, 0.0056) (100, 203, 1)	(96.77, 0.0077) (10000, 0.6, 10)
ecoli3 (336 × 7)	(90, 0.0043) (0.00001, 0.00001)	(90, 0.0946) (0.00001, 0.2, 0.00001)	(90, 0.2871) (0.00001, -1, 0.00001)	(90, 0.0926) (1, 0.6, 0.75, 0.25, 1, 0.5, 1)	(94, 0.0059) (10, 183, 1)	(90, 0.0777) (10, 0.7, 0.4, 10)
ecoli4 (336 × 7)	(97, 0.007) (10, 10)	(98, 0.1866) (100, 0.2, 1)	(98, 0.0068) (1, 0, 10)	(75, 0.1004) (1, 0.7, 0.4, 0.05, 0.5, 0.9, 0.1)	(99, 0.006) (1000, 183, 1)	(99, 0.009) (10, 0, 0, 10)
glass5 (214 × 9)	(92.19, 0.003) (0.00001, 0.00001)	(92.19, 0.0305) (0.00001, 0.2, 0.00001)	(92.19, 0.2177) (0.00001, -0.4, 1)	(92.19, 0.001) (0.00001, 0.5, 0.1, 0.01, 0.5, 0.01, 0.00001)	(92.19, 0.0034) (10, 123, 1)	(92.19, 0.0084) (0.00001, -1, 1, 0.00001)
haber (306 × 3)	(63.74, 0.0114) (10000, 1)	(70.33, 0.0611) (1, 0.2, 0.01)	(69.23, 0.8993) (0.00001, -0.5, 0.1)	(58.24, 0.0022) (0.00001, 0.6, 0.75, 0.25, 1.5, 0.9, 1)	(71.43, 0.0035) (10000, 143, 10)	(73.63, 0.0807) (1, 0.3, 0.9, 1)
new-thyroid1 (215 × 5)	(93.75, 0.0036) (10, 0.1)	(93.75, 0.0247) (10, 0.2, 0.1)	(92.19, 0.0461) (0.00001, -0.9, 1)	(82.81, 0.0011) (0.001, 0.6, 0.1, 0.1, 1.5, 0.01, 1)	(96.88, 0.0028) (0.1, 183, 100)	(95.31, 0.413) (0.00001, 0.1, 0.1, 1)
Average accuracy	81.83	81.91	82.14	78	83.41	85.46
Average rank	3.68	3.65	3.63	5.12	2.78	2.13

In this subsection, we evaluate the classification performance of the proposed FP-SVM against the baseline models on real-world benchmark UCI [126] and KEEL [127] datasets. The experimental results of SVM, pin-SVM, unified pin-SVM, QN-SSVM, IFRVFL, and the proposed FP-SVM having RBF kernel are presented in Table 3.2. The average accuracy

Table 3.3: Classification performance of the proposed FP-SVM on UCI and KEEL binary datasets with different kernel functions.

Dataset	proposed FP-SVM (RBF) (Accuracy, Time(s)) ($C_1, \tau_2, \tau_1, \sigma$)	proposed FP-SVM (Polynomial) (Accuracy, Time(s)) ($C_1, \sigma_1, \tau_2, \tau_1$)	proposed FP-SVM (Linear) (Accuracy, Time(s)) (C_1, τ_2, τ_1)
breast-cancer-wisc	(95.69, 0.0877) (0.0001, 0, 1, 1)	(95.69, 0.0433) (1, 1, 0.1, 0)	(95.69, 0.0224) (1, 0.1, 0)
breast-cancer-wisc-prog	(76.27, 0.009) (0.00001, 0, 0.9, 0.01)	(74.58, 0.0034) (0.001, 4, 0, 0.1)	(76.27, 0.0523) (10, 0.4, 0.2)
congressional-voting	(60.77, 0.0195) (1, 0.1, 0, 0.1)	(60, 0.1598) (1000, 2, 0.5, -0.4)	(60.77, 0.0495) (0.01, 0.9, -0.8)
cylinder-bands	(69.28, 0.0179) (100000, 0, 0.2, 1)	(76.47, 5.7357) (0.00001, 4, 0.3, 0.3)	(64.05, 0.0945) (0.01, 0.7, -0.4)
dermatology	(98.17, 0.0365) (10, 0, 0.9, 1)	(97.25, 0.007) (0.0001, 4, 0.2, 0)	(70.64, 0.0538) (0.01, 0.9, -0.1)
fertility	(90, 0.0168) (10, 0.8, 0.1, 1)	(90, 0.0238) (0.001, 4, 0.6, 1)	(90, 0.0036) (0.00001, -1, 1)
haberman-survival	(74.73, 0.0108) (0.001, 0, 0.9, 0.1)	(73.63, 0.0034) (0.0001, 5, 0, 0)	(71.43, 1.3959) (0.0001, 0.9, 0.9)
hepatitis	(71.74, 0.0025) (100000, 0, 0.9, 0.00001)	(80.43, 0.0121) (0.00001, 4, 1, 0.1)	(71.74, 0.2539) (0.00001, 0.8, 0.6)
mammographic	(81.25, 0.0616) (10, 0.1, 0, 1)	(81.25, 0.1717) (10, 5, 0, 0.7)	(80.21, 0.0474) (10, 0.1, 0)
monks-1	(76.51, 0.0098) (0.1, 0, 0.8, 1)	(76.51, 0.0151) (0.0001, 4, 0.2, 0)	(69.88, 0.0067) (0.0001, 0, 0.1)
planning	(75.93, 0.2767) (0.0001, 0.7, 0.7, 0.1)	(75.93, 0.1797) (0.01, 5, 0.6, 0.5)	(75.93, 0.0164) (0.001, 0.9, 1)
statlog-australian-credit	(66.67, 0.4351) (0.1, 0.6, 0.3, 1)	(65.7, 8.4555) (10000, 5, 0.2, 0.2)	(66.67, 0.1917) (0.1, 0.8, -0.4)
statlog-heart	(81.48, 0.0639) (0.00001, 0.7, 0.7, 0.1)	(83.95, 0.035) (0.01, 1, 0.7, -0.1)	(76.54, 0.0179) (0.01, 0.6, -0.5)
trains	(66.67, 0.0039) (0.00001, 0.2, 0.9, 0.00001)	(66.67, 0.0006) (0, 2, 0.1, 0.6)	(66.67, 0.0005) (0.01, 0.1, 1)
vertebral-column-2classes	(66.67, 0.0034) (0.01, 0, 0.2, 1)	(66.67, 0.0037) (0, 1, 0, 0.2)	(69.89, 0.1881) (0.00001, 0.4, 0.2)
crossplane130	(97.44, 0.0692) (0.00001, 0.3, 2.78E - 17, 0.0001)	(69.23, 0.0046) (0.00001, 5, 0.8, -0.3)	(97.44, 0.0027) (0.1, 0.3, -0.3)
ecoli-0-1-4-6_vs_5	(94.05, 0.0422) (0.00001, 0.8, 0.8, 0.01)	(92.86, 0.0083) (0.00001, 4, 0.1, 0)	(92.86, 0.01) (0.00001, -0.1, 0.1)
ecoli-0-1-4-7_vs_5-6	(96.97, 0.0311) (1, 1, -0.9, 0.1)	(94.95, 0.004) (0.0001, 2, 0.1, 0)	(91.92, 0.972) (0.00001, 0.4, 0.4)
ecoli-0-1_vs_2-3-5	(95.89, 0.9466) (0.00001, 0.1, 0.1, 0.1)	(90.41, 0.003) (10, 3, 0.1, 0)	(87.67, 0.0083) (0.00001, -0.1, 0.1)
ecoli-0-1_vs_5	(95.83, 0.0355) (0.00001, 0, 0.1, 0.1)	(97.22, 0.0028) (0.00001, 3, 0.2, 0)	(97.22, 0.0029) (0.0001, 0.2, 0)
ecoli-0-2-6-7_vs_3-5	(94.03, 0.7198) (1, 0.5, 0.5, 0.1)	(94.03, 0.0035) (0.00001, 4, 0.2, 0)	(80.6, 0.0149) (0.0001, -0.3, 1)
ecoli-0-3-4-6_vs_5	(95.08, 0.0047) (0.00001, 0, 0.4, 0.1)	(93.44, 0.0021) (0.00001, 3, 0.1, 0)	(85.25, 0.4952) (0.00001, 0.3, 0.3)
ecoli-0-3-4-7_vs_5-6	(98.7, 0.0036) (1, 0, 1, 0.1)	(6.49, 0.0041) (0.001, 4, 0, 0.1)	(93.51, 0.0087) (0.00001, -0.1, 0.1)
ecoli-0-6-7_vs_3-5	(96.97, 0.0023) (0.01, 0, 0.5, 0.01)	(93.94, 0.0061) (10, 4, 0, 0.2)	(90.91, 0.0116) (0.00001, -0.4, 0.7)
ecoli0137vs26	(96.77, 0.0077) (10000, 0, 0.6, 10)	(87.1, 1.1045) (0.001, 1, 0.5, 0.5)	(91.4, 1.4176) (0.00001, 0.4, 0.4)
ecoli3	(90, 0.0777) (10, 0.7, 0.4, 10)	(90, 0.023) (0.00001, 1, -1, 1)	(90, 0.0181) (0.00001, -1, 1)
ecoli4	(99, 0.009) (10, 0, 0, 10)	(93, 1.2017) (0.00001, 4, 0.3, 0.3)	(97, 1.6133) (0.00001, 0.4, 0.4)
glass5	(92.19, 0.0084) (0.00001, -1, 1, 0.00001)	(92.19, 0.0175) (0.00001, 2, -0.8, 0.9)	(93.75, 0.006) (0.0001, -0.5, 0.5)
haber	(73.63, 0.0807) (1, 0.3, 0.9, 1)	(69.23, 0.0029) (0.00001, 3, 0, 0.8)	(74.73, 1.1434) (0.00001, 0.8, 0.8)
new-thyroid1	(95.31, 0.413) (0.00001, 0.1, 0.1, 1)	(84.38, 0.7896) (1, 4, 0.8, -0.8)	(92.19, 0.028) (0.0001, 1, 0.1)
Average accuracy	85.46	80.44	82.09

of SVM, pin-SVM, unified pin-SVM, QN-SSVM, IFRVFL and the proposed FP-SVM are 81.83%, 81.91%, 82.14%, 78%, 83.41% and 85.46%, respectively. Thus, the proposed FP-SVM outperforms the baseline models. Figure 3.5 presents the comparison of the accuracies of baseline models with the proposed FP-SVM and signifies that the proposed FP-SVM is more efficient than the baseline models.

We also conducted experiments on the proposed FP-SVM with different kernel functions, which include polynomial and linear kernels. The result of their generalization performance is presented in Table 3.3 which depicts that the RBF kernel has the best average accuracy of 85.46%, followed by linear and polynomial kernel with 82.09% and 80.44%, respectively.

In addition to assess the effectiveness of the proposed FP-SVM, we carried out the win-tie-loss sign test, whose results are presented in Table 3.4. The value of $\frac{N}{2} + 1.96\sqrt{N}$ at 5% level of significance, is 20.37 for $N = 30$. Out of 30 real-world datasets, the proposed FP-SVM wins on 17 datasets with respect to both SVM and pin-SVM. In comparison to unified pin-SVM and QN-SSVM, the proposed FP-SVM wins on 13 and 21 datasets, respectively. In comparison with IFRVFL, the proposed FP-SVM has 15 wins. As a result, the proposed FP-SVM has the maximum wins and is a better classifier than the baseline models.

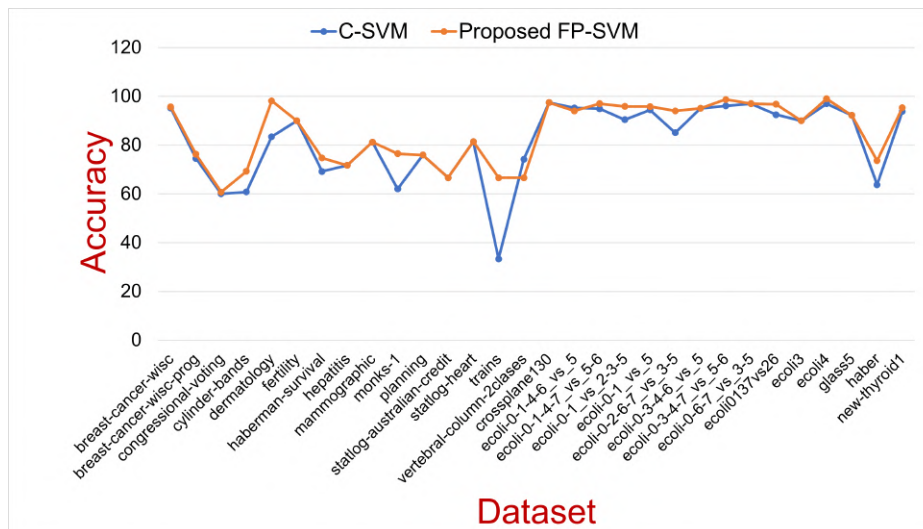
However, contrary to the common assumption, considering only important wins and losses make the test less trustworthy rather than more reliable. As good performance on a single dataset might make up for bad performance on others, average accuracy may be influenced. Therefore, to further assess the model's performance, we use ranking scheme. The model with the highest rank is the poorest performer, while the model with the lowest rank is the best performer. The average rank of SVM, pin-SVM, unified pin-SVM, QN-SSVM, IFRVFL and the proposed FP-SVMs are 3.68, 3.65, 3.63, 5.12, 2.78, and 2.13, respectively (see Table 3.2). Hence, the proposed FP-SVM surpasses the baseline models.

For a more comprehensive evaluation of the model's effectiveness, we used the Friedman test [123], which uses average rank. In our case, we have $N = 30$ datasets and $md = 6$ models thus, χ_F^2 and F_F have 5 and (5, 145) degrees of freedom, respectively. From the equations (2.56) and (2.57), we have $\chi_F^2 = 43.04$ and $F_F = 11.67$. When $\alpha = 0.05$, the value of $F_{(5,145)} = 2.28$. As $F_F > 11.67$ implies that the null hypothesis is rejected resulting in non-equivalence of the algorithm.

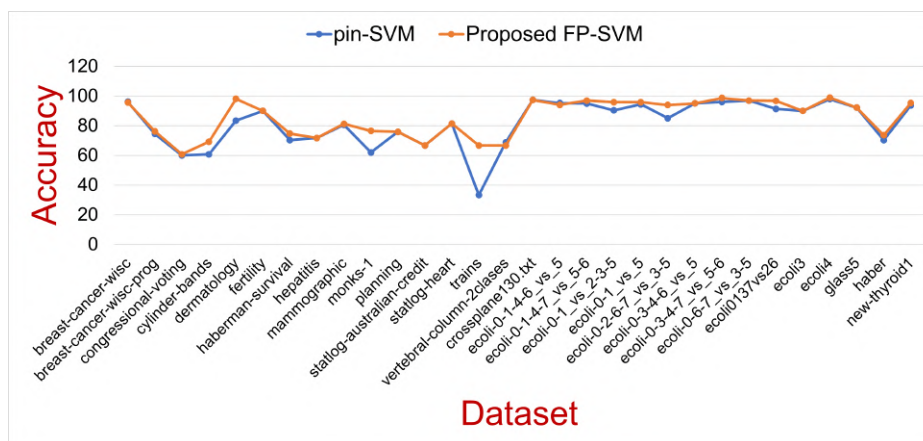
Now, we proceed with the Nemenyi post hoc test [129], which is employed to compare the classifiers in pairs. Here, we have $md = 6$, so $q_{0.05} = 2.85$ [123] and thus from equation (2.58), critical difference (cd) = 1.3767. Table 3.5 illustrates that the proposed FP-SVM differs significantly from the baseline models SVM, pin-SVM, unified pin-SVM, and QN-SSVM, as the difference in the average rank is greater than cd. Though the rank difference of the proposed FP-SVM with IFRVFL is less than cd, the superiority of the proposed FP-SVM is depicted by average rank, average accuracy and other statistical tests.

Furthermore, we use the Wilcoxon signed-ranks test [123] which ranks the performance differences between the two classifiers for each data set. We reject the null hypothesis assumption that the models are equivalent, if test's p -value is not greater than 0.05. Table 3.6 shows that the p -values of baseline algorithms are not greater than 0.05 which leads to the rejection of the null hypothesis and thus, the proposed FP-SVM outperforms the baseline models, i.e., SVM, pin-SVM, unified pin-SVM, QN-SSVM, and IFRVFL.

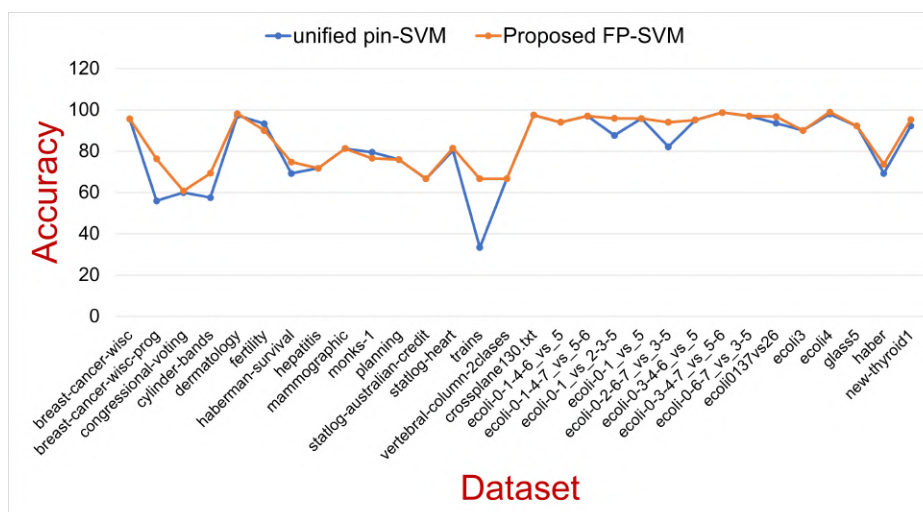
The numerical experiments and statistical comparisons of the proposed FP-SVM and the baseline models show that the proposed FP-SVM has superior classification accuracy compared to baseline models.



(a) Comparison between accuracies of SVM and proposed FP-SVM

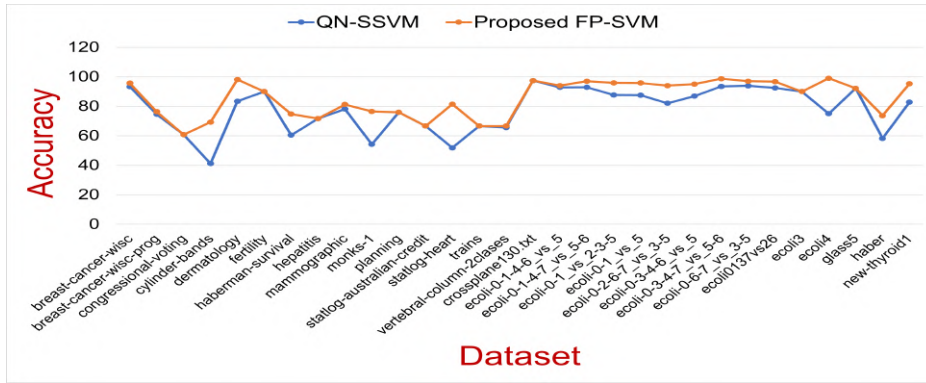


(b) Comparison between accuracies of pin-SVM and proposed FP-SVM

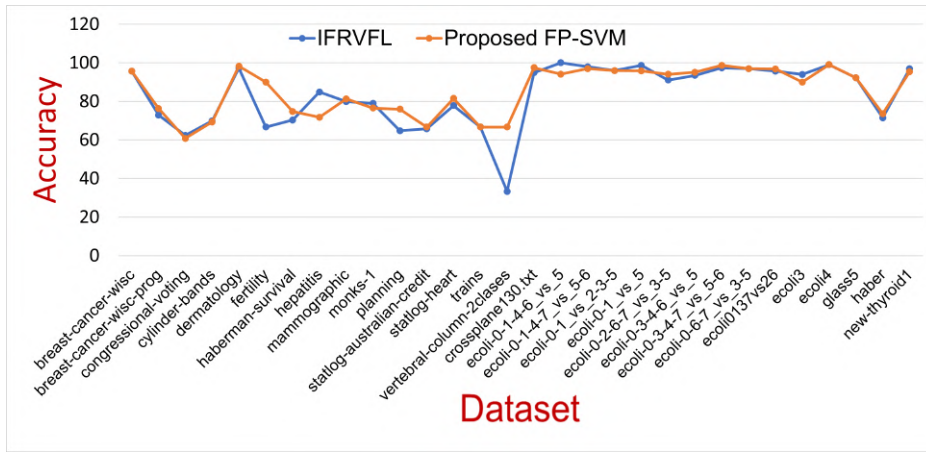


(c) Comparison between accuracies of unified pin-SVM and proposed FP-SVM

Figure 3.5: Comparison of accuracies of baseline models and proposed FP-SVM.



(d) Comparison between accuracies of QN-SSVM and proposed FP-SVM



(e) Comparison between accuracies of IFRVFL and proposed FP-SVM

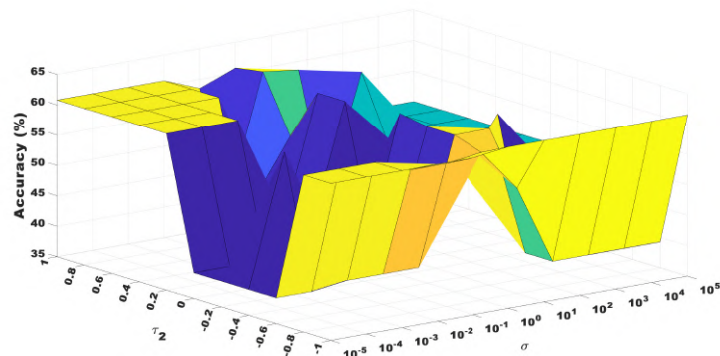
Figure 3.5: Comparison of accuracies of baseline models and proposed FP-SVM.

Table 3.4: Win-tie-loss of row method over column method on UCI and KEEL datasets. Here, in [a/b/c], a, b and c correspond to number of wins, ties and losses of row method over column method, respectively.

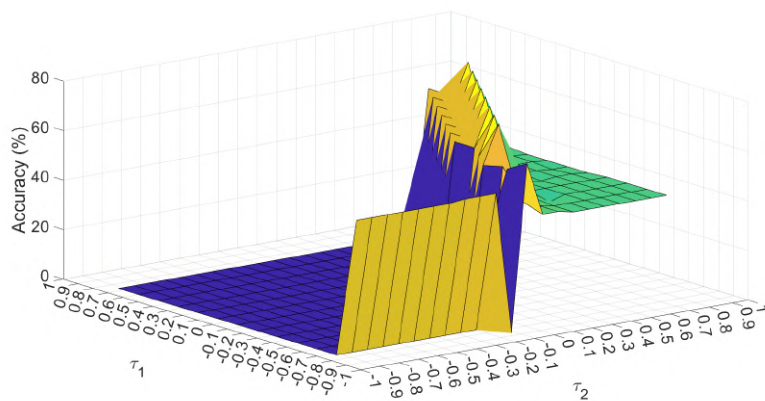
Model	SVM [77]	pin-SVM [17]	unified pin-SVM [78]	QN-SSVM [124]	IFRVFL [125]
Proposed FP-SVM	[17/11/2]	[17/10/3]	[13/15/2]	[21/9/0]	[15/6/9]
IFRVFL [125]	[9/2/19]	[10/3/17]	[10/4/16]	[6/2/22]	
QN-SSVM [124]	[2/10/18]	[3/9/18]	[3/8/19]		
unified pin-SVM [78]	[10/12/8]	[8/11/11]			
pin-SVM [17]	[4/23/3]				

Table 3.5: Nemenyi post hoc significant difference for the proposed FP-SVM with respect to the baseline models on UCI and KEEL datasets.

Significance	SVM [77]	pin-SVM [17]	unified pin-SVM [78]	QN-SSVM [124]	IFRVFL [125]
Proposed FP-SVM	Yes	Yes	Yes	Yes	No



(a) congressional-voting



(b) Cylinder-bands

Figure 3.6: Evaluation of the proposed FP-SVM's accuracy on UCI data while varying parameter σ , τ_2 and τ_1 .

Table 3.6: Wilcoxon-signed rank test with respect to the proposed FP-SVM on UCI and KEEL datasets.

Model	R_+	R_-	p -value	Hypothesis
SVM [77]	16	174	0.00148	Rejected
pin-SVM [17]	16	194	0.0009	Rejected
unified pin-SVM [78]	13	107	0.00758	Rejected
QN-SSVM [124]	0	231	< 0.00001	Rejected
IFRVFL [125]	108	192	0.2301	Not rejected

3.1.3.4 Parameter sensitivity analysis

To examine the influence of the loss hyperparameters (τ_2 , τ_1) and kernel parameter σ on the performance of the proposed FP-SVM, we used surface plot command of Matlab to create a three-dimensional surface plot. While plotting a surface diagram against two hyperparameter, all the other hyperparameters are set to their optimal values. In Figure 3.6a, X and Y axis represents the loss hyperparameter τ_2 and kernel parameter σ , respectively and Z axis represents the model's accuracy for congressional-voting dataset. Further, Figure 3.6b shows the variation of τ_2 and τ_1 for cylinder-bands dataset. For ease of representation and graph plotting, we chose the values of τ_2 and τ_1 from $\{-1, -0.9, \dots, 0.9, 1\}$. Thus, it is clear from the Figure 3.6 that the accuracy of the proposed FP-SVM depends on the choice of parameters.

3.1.3.5 Application on breakHis dataset

To demonstrate the efficacy of the proposed FP-SVM on real-world applications, we implement it on breast cancer dataset which contains histopathology scans named BreakHis, available in [130]. We used 1240 scans from the BreakHis dataset with 400 times magnification. The scans mainly belong to two classes: malignant and benign. The subclasses of the malignant class are ductal carcinoma (dc), mucinous carcinoma (mc), lobular carcinoma (lc), papillary carcinoma (pc) with 208, 169, 137, and 138 scans, respectively. While the benign class has also four subclasses: adenosis (a), phyllodes tumour (pt), fibroadenoma (f), and tubular adenoma (ta), with 106, 115, 237, and 130 scans, respectively. For feature extraction from the histopathological scans, we follow the same process as opted in [8, 131]. We use 16 combinations of benign and malignant classes to distinguish the four subclasses of benign and malignant cancer, as shown in Table 3.7.

Table 3.7: Classification performance of proposed FP-SVM and baseline models on BreakHis dataset with RBF kernel.

Dataset	SVM [90] (Accuracy,Time(s)) (C_1, σ)	pin-SVM [17] (Accuracy,Time(s)) (C_1, τ_1, σ)	unified pin-SVM [78] (Accuracy,Time(s)) (C_1, τ_1, σ)	QN-SSVM [124] (Accuracy,Time(s)) ($C_1, \tau_1, \tau_2, \delta_1, \delta_2, \nu, \sigma$)	IFRVFL [125] (Accuracy,Time(s)) (C_1, N_k, σ)	Proposed FP-SVM (Accuracy,Time(s)) ($C_1, \tau_2, \tau_1, \sigma$)
a_vs_dc (314 × 769)	(85.11, 0.0795) (10, 0.01)	(85.11, 0.1286) (10, 0.2, 0.01)	(81.91, 0.1534) (100, -0.5, 0.1)	(81.91, 0.0151) (0.00001, 0.75, 0.75, 0.25, 1.75, 0.1, 10)	(70.21, 0.0384) (0.01, 23, 10000)	(85.11, 0.0646) (10, 0, 0.9, 0.1)
a_vs_lc (243 × 769)	(58.33, 0.0143) (1, 0.1)	(58.33, 0.0415) (1, 0.2, 0.1)	(58.33, 0.0558) (0.00001, -0.9, 1)	(40.28, 0.0728) (1000, 0.5, 0.75, 0.6, 1.5, 0.9, 10)	(51.39, 0.0064) (100000, 103, 0.00001)	(40.28, 0.0103) (0.0001, 0, 0.8, 0.1)
a_vs_mc (275 × 769)	(60.98, 0.0117) (0.00001, 0.00001)	(60.98, 0.0684) (0.00001, 0.2, 0.00001)	(60.98, 0.0259) (1000, -0.2, 0.001)	(60.98, 0.0795) (100000, 1, 0.1, 0.45, 1.75, 0.01, 10)	(52.44, 0.0242) (0.01, 163, 100000)	(60.98, 0.0145) (1, 0, 0.4, 0.01)
a_vs_pc (244 × 769)	(54.79, 0.0099) (0.1, 0.1)	(57.53, 0.0646) (1000, 0.8, 100)	(54.79, 0.0087) (0.1, 0, 0.1)	(43.84, 0.0601) (10, 1, 0.4, 0.5, 1.5, 0.01, 10)	(54.79, 0.0191) (0.01, 203, 100000)	(54.79, 0.0231) (0.1, 0.8, 0.5, 0.1)
f_vs_dc (445 × 769)	(83.46, 0.0462) (1, 0.1)	(83.46, 0.1656) (1, 0.2, 0.1)	(83.46, 0.076) (1, -0.8, 0.1)	(49.62, 0.1744) (0.01, 0.7, 0.4, 0.6, 1.75, 0.01, 10)	(58.65, 0.0166) (1, 163, 10)	(83.46, 0.0389) (0.00001, 0, 0.2, 0.1)
f_vs_lc (374 × 769)	(62.5, 0.0073) (0.00001, 0.00001)	(62.5, 0.1068) (0.00001, 0.2, 0.00001)	(64.29, 0.0657) (0.1, -0.7, 0.1)	(37.5, 0.125) (0.1, 0.5, 0.4, 0.25, 1, 0.01, 10)	(66.96, 0.0104) (100000, 3, 0.001)	(69.64, 0.0487) (1, 0.6, -0.1, 0.1)
f_vs_mc (406 × 769)	(57.02, 0.0067) (0.1, 0.1)	(57.02, 0.1313) (0.00001, 0.2, 0.00001)	(59.5, 0.0656) (1, -0.8, 0.1)	(57.02, 0.1322) (0.001, 1, 0.4, 0.25, 1.5, 0.1, 10)	(57.85, 0.014) (0.01, 123, 10)	(59.5, 0.7187) (0.001, 0.6, 0.6, 0.1)
f_vs_pc (375 × 769)	(67.86, 0.0066) (1, 0.1)	(67.86, 0.0865) (1, 0.2, 0.1)	(67.86, 0.0662) (0.00001, -0.7, 0.1)	(67.86, 0.1237) (1000, 0.5, 0.4, 0.01, 1, 0.1, 10)	(67.86, 0.0125) (0.001, 63, 10)	(62.5, 0.0334) (0.1, 0.5, 1, 0.1)
pt_vs_dc (323 × 769)	(81.25, 0.0054) (10, 0.01)	(81.25, 0.0472) (10, 0.2, 0.01)	(67.71, 0.0385) (10000, -1, 0.01)	(84.38, 0.0033) (0.0001, 0.6, 0.4, 0.25, 0.5, 0.01, 10)	(69.79, 0.0593) (0.0001, 123, 100000)	(83.33, 0.7596) (1, 1, 1, 0.1)
pt_vs_lc (252 × 769)	(53.33, 0.005) (1000, 100)	(53.33, 0.0872) (1000, 0.2, 100)	(46.67, 0.014) (0.00001, -0.7, 1)	(53.33, 0.0939) (1, 0.5, 0.4, 0.25, 1.5, 0.5, 10)	(46.67, 0.0173) (0, 83, 1000)	(54.67, 0.0073) (100, 0, 1, 1)
pt_vs_mc (284 × 769)	(54.12, 0.0049) (0.00001, 0.00001)	(54.12, 0.0589) (0.00001, 0.2, 0.00001)	(54.12, 0.0156) (0.1, -0.6, 0.1)	(56.47, 0.0964) (10, 0.75, 0.4, 0.5, 1, 0.01, 10)	(43.53, 0.0237) (0.001, 63, 100)	(63.53, 0.0246) (0.1, 0.9, 0.2, 0.1)
pt_vs_pc (253 × 769)	(45.33, 0.0041) (0.00001, 0.00001)	(60, 0.0531) (0.1, 0.2, 0.1)	(45.33, 0.0276) (1, -0.8, 1)	(42.67, 0.0769) (10, 1, 0.4, 0.6, 1.75, 0.01, 10)	(54.67, 0.0086) (10, 183, 10)	(65.33, 0.0147) (0.1, 0.5, 0.3, 0.1)
ta_vs_dc (338 × 769)	(78.22, 0.0058) (1, 0.1)	(78.22, 0.075) (1, 0.2, 0.1)	(77.23, 0.8584) (1, -1, 0.1)	(73.27, 0.0043) (0.0001, 0.6, 0.4, 0.05, 1.75, 0.5, 10)	(58.42, 0.0124) (1, 143, 10)	(75.25, 0.0285) (1, 0.9, 0.2, 0.1)
ta_vs_lc (267 × 769)	(55, 0.0042) (1, 0.1)	(55, 0.0365) (1, 0.2, 0.1)	(55, 0.0351) (0.001, -0.9, 0.1)	(62.5, 0.0776) (0.01, 0.7, 0.1, 0.45, 1, 0.1, 10)	(56.25, 0.0083) (10, 183, 0.00001)	(68.75, 0.0192) (10, 0.5, 0.1, 0.1)
ta_vs_mc (299 × 769)	(62.92, 0.0048) (1, 0.1)	(62.92, 0.0659) (0.1, 0.2, 0.1)	(64.04, 0.2353) (1, -1, 0.1)	(64.04, 0.006) (0.0001, 0.6, 0.1, 0.45, 0.5, 0.5, 10)	(49.44, 0.0079) (10000, 43, 0.00001)	(66.29, 0.9503) (10, 0.1, 0.1, 0.1)
ta_vs_pc (268 × 769)	(62.5, 0.0044) (1, 0.1)	(62.5, 0.0492) (1, 0.2, 0.1)	(67.5, 0.8178) (1, -1, 0.1)	(60, 0.0026) (0.0001, 0.6, 0.4, 0.5, 0.5, 0.1, 10)	(45, 0.0057) (0, 3, 0.001)	(63.75, 0.0211) (1, 1, 0.3, 0.1)
Average Accuracy	63.92	65.01	63.05	58.48	56.5	67.1
Average Rank	3.47	3.22	3.47	4.13	4.5	2.22

Table 3.7 demonstrates the efficiency of the proposed FP-SVM in terms of accuracy. We analyze that the average accuracy of proposed FP-SVM is 67.1% which is superior to the baseline models SVM [77], pin-SVM [17], unified pin-SVM [78], QN-SSVM [124], and IFRVFL having 63.92%, 65.01%, 63.05%, 58.48%, and 56.5% accuracy, respectively. The average ranks of the baseline models in the aforementioned order are 3.47, 3.22, 3.47 4.13, and 4.5 respectively, whereas the rank of the proposed FP-SVM is 2.22. Therefore, the proposed FP-SVM obtains the best generalization performance compared to the baseline models, which shows the superiority of the proposed FP-SVM for breast cancer diagnosis.

3.2 Proposed universum twin support vector machine with truncated pinball loss

In this section, we present a novel classification model, the universum twin support vector machine with truncated pinball loss (Tpin-UTWSVM) [82], which extends the UTWSVM framework [26] by incorporating a truncated pinball loss function. Further, we also analyze the performance of the proposed Tpin-UTWSVM with respect to the baseline models.

3.2.1 Formulation of the proposed Tpin-UTWSVM

Pin-GTSVM [101] leads to a model which is noise insensitive and stable to resampling, however the model lacks sparsity as compared to TWSVM [22]. The truncated pinball loss function acts as a flexible framework of trade-off between sparsity and feature noise insensitivity. Also, the presence of universum data (denoted by $U = \{x_1^*, x_2^*, \dots, x_u^*\}$) improves the generalization performance of the model.

3.2.1.1 Linear case

The two non parallel hyperplanes are:

$$f_j(x) = 0 \text{ with } f_j(x) = x^t w_j + b_j, \quad (3.17)$$

where $j = 1, 2$ denote positive and negative hyperplanes, respectively. Assume M_1 and M_2 be the matrices of positive and negative class samples with dimension $l_1 \times n$ and $l_2 \times n$, respectively. The QPPs for the linear case are as follows:

$$\min_{w_1, b_1} \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_u \sum_{l=1}^u \lambda[f_1(x_l^*)] + C_1 \sum_{x_i \in M_2} \mathbb{P}_{\tau_2, s_2}(1 - y_i f_1(x_i)) \quad (3.18)$$

and

$$\min_{w_2, b_2} \frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_u \sum_{l=1}^u \lambda[f_2(x_l^*)] + C_2 \sum_{x_i \in M_1} \mathbb{P}_{\tau_1, s_1}(1 - y_i f_2(x_i)), \quad (3.19)$$

where C_1, C_2, C_u are the positive regularization parameters. $\tau_1, \tau_2 \in [0, 1]$ and s_1, s_2 are positive parameters; $\mathbb{P}_{\tau_1, s_1}$ and $\mathbb{P}_{\tau_2, s_2}$ signify the truncated pinball loss as shown in equation (2.10). Also, $\lambda[f_j(x_l^*)]$, ($j = 1, 2$) is the ϵ insensitive loss shown in equation (2.22), explicitly written as:

$$\lambda[w_j^t x_l^* + b_j] = \begin{cases} -\epsilon - (w_j^t x_l^* + b_j), & \text{if } w_j^t x_l^* + b_j \leq -\epsilon, \\ 0, & \text{if } -\epsilon < w_j^t x_l^* + b_j \leq \epsilon, \\ -\epsilon + w_j^t x_l^* + b_j, & \text{if } w_j^t x_l^* + b_j > \epsilon. \end{cases} \quad (3.20)$$

for $l = 1, 2, \dots, u$. The first term in problems (3.18) and (3.19) minimize the distance of data points from the respective class. The second and third terms impose a penalty on the data points. Using the equation (2.10) of the truncated pinball loss, the QPPs (3.18) and (3.19) can be written as:

$$\begin{aligned} \min_{w_1, b_1} \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_u \sum_{l=1}^u \lambda[f_1(x_l^*)] + C_1 \sum_{x_i \in M_2} [H_{1+\tau_2}(1 - y_i f_1(x_i))] \\ \underbrace{\hspace{15em}}_{J_{convex}} \\ + \underbrace{\left(-C_1 \sum_{x_i \in M_2} [H_{\tau_2}(1 - y_i f_1(x_i) + s_2)] \right)}_{J_{concave}} \end{aligned} \quad (3.21)$$

and

$$\min_{w_2, b_2} \underbrace{\frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_u \sum_{l=1}^u \lambda[f_2(x_l^*)] + C_2 \sum_{x_i \in M_1} [H_{1+\tau_1}(1 - y_i f_2(x_i))]}_{J_{convex}} + \underbrace{\left(-C_2 \sum_{x_i \in M_1} [H_{\tau_1}(1 - y_i f_2(x_i) + s_1)] \right)}_{J_{concave}}. \quad (3.22)$$

The above two optimization problems are non-convex and can be solved by concave-convex procedure (CCCP). It decomposes as the sum of convex (J_{convex}) and concave ($J_{concave}$) part. Convex part is represented by first three terms and concave part is represented by fourth term of equations (3.21) and (3.22). Now onwards in this section, we assume

$$v_1 = \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} \text{ and } v_2 = \begin{bmatrix} w_2 \\ b_2 \end{bmatrix}. \quad (3.23)$$

3.2.1.1.1 Primal problems

CCCP [132] is a popular method which is widely used to solve a non-differentiable optimization problem which has the difference of convex functions. It iteratively solves a sequence of convex subproblems. The CCCP algorithm used here converges to the stationary point [132] and has a linear convergence rate.

By applying CCCP method, the optimization problem (3.21) and (3.22) takes the form:

$$\min_{v_j} J_{convex} + \langle J'_{concave}(v_j^r), v_j \rangle \quad (3.24)$$

for $j = 1, 2$, respectively. Here, superscript r denotes the iterations which we will drop in further analysis. Representing the derivative $J'_{concave}(v_j)$ by ϱ_j , where $\varrho_j = (\varrho_{j1}, \varrho_{j2}, \dots, \varrho_{j\tilde{m}})^t$ having $\tilde{m}$ as l_1, l_2 for j being 1, 2, respectively. We have $\forall i =$

$1, 2, \dots, l_2,$

$$\varrho_{2i} = \begin{cases} C_1 \tau_2, & \text{if } 1 - y_i f_1(x_i) + s_2 > 0, \\ 0, & \text{if } 1 - y_i f_1(x_i) + s_2 \leq 0. \end{cases} \quad (3.25)$$

Applying CCCP method to QPP (3.21), we get:

$$\begin{aligned} \min_{w_1, b_1} & \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_u \sum_{l=1}^u \lambda[f_1(x_l^*)] \\ & + C_1 \sum_{x_i \in M_2} [H_{1+\tau_2}(1 - y_i f_1(x_i))] - \varrho_2^t(e_2 + M_2 w_1 + e_2 b_1), \end{aligned} \quad (3.26)$$

where each component of ϱ_2 is defined in equation (3.25). Similarly,

$$\varrho_{1i} = \begin{cases} C_2 \tau_1, & \text{if } 1 - y_i f_2(x_i) + s_1 > 0, \\ 0, & \text{if } 1 - y_i f_2(x_i) + s_1 \leq 0, \end{cases} \quad (3.27)$$

for $i = 1, 2, \dots, l_1$. Applying CCCP to QPP (3.22), we get

$$\begin{aligned} \min_{w_2, b_2} & \frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_u \sum_{l=1}^u \lambda[f_2(x_l^*)] \\ & + C_2 \sum_{x_i \in M_1} [H_{1+\tau_1}(1 - y_i f_2(x_i))] - \varrho_1^t(e_1 - (M_1 w_2 + e_1 b_2)) \end{aligned} \quad (3.28)$$

with each component of ϱ_1 as defined in equation (3.27). Introducing slack variables ζ_2, η_2 in problem (3.26) and ζ_1, η_1 in problem (3.28), the primal problems are:

$$\begin{aligned} \min_{w_1, b_1} & \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_u e_u^t \eta_2 + C_1 e_2^t \zeta_2 - \varrho_2^t(e_2 + M_2 w_1 + e_2 b_1) \\ \text{subject to} & - (M_2 w_1 + e_2 b_1) + \frac{\zeta_2}{1 + \tau_2} \geq e_2, \quad \zeta_2 \geq 0, \\ & (U w_1 + e_u b_1) + \eta_2 \geq (-1 + \epsilon) e_u, \quad \eta_2 \geq 0 \end{aligned} \quad (3.29)$$

and

$$\begin{aligned}
 \min_{w_2, b_2} \quad & \frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_u e_u^t \eta_1 + C_2 e_1^t \zeta_1 - \varrho_1^t (e_1 - (M_1 w_2 + e_1 b_2)) \\
 \text{subject to} \quad & (M_1 w_2 + e_1 b_2) + \frac{\zeta_1}{1 + \tau_1} \geq e_1, \quad \zeta_1 \geq 0 \\
 & - (U w_2 + e_u b_2) + \eta_1 \geq (-1 + \epsilon) e_u, \quad \eta_1 \geq 0.
 \end{aligned} \tag{3.30}$$

Thus, QPPs (3.29) and (3.30) are the primal problems of the proposed Tpin-UTWSVM model.

3.2.1.1.2 Dual problems

We will demonstrate the detailed calculation to obtain the dual of QPP (3.29) and similarly the dual of QPP (3.30) can be obtained. The Lagrangian obtained for the problem (3.29) is:

$$\begin{aligned}
 L(w_1, b_1, \zeta_2, \eta_2, \beta_1, \gamma_1, \delta_1, \mu_1) = & \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_u e_u^t \eta_2 + C_1 e_2^t \zeta_2 \\
 & - \varrho_2^t (e_2 + M_2 w_1 + e_2 b_1) + \beta_1^t \left(e_2 - \frac{\zeta_2}{1 + \tau_2} + (M_2 w_1 + e_2 b_1) \right) \\
 & + \gamma_1^t ((-1 + \epsilon) e_u - (U w_1 + e_u b_1) - \eta_2) - \delta_1^t \eta_2 - \mu_1^t \zeta_2,
 \end{aligned} \tag{3.31}$$

where $\beta_1, \gamma_1, \delta_1, \mu_1$ are vectors of non negative Lagrange multipliers. Use the necessary and sufficient optimality condition, i.e., K.K.T. conditions. Differentiating the Lagrangian (3.31), we get the following K.K.T. conditions:

$$\frac{\partial L}{\partial w_1} = M_1^t (M_1 w_1 + e_1 b_1) + M_2^t (\beta_1 - \varrho_2) - U^t \gamma_1 = 0, \tag{3.32}$$

$$\frac{\partial L}{\partial b_1} = e_1^t (M_1 w_1 + e_1 b_1) + e_2^t (\beta_1 - \varrho_2) - e_u^t \gamma_1 = 0, \tag{3.33}$$

$$\frac{\partial L}{\partial \zeta_2} = C_1 e_2 - \frac{\beta_1}{1 + \tau_2} - \mu_1 = 0, \tag{3.34}$$

$$\frac{\partial L}{\partial \eta_2} = C_u e_u - \gamma_1 - \delta_1 = 0, \tag{3.35}$$

$$\beta_1^t \left(e_2 - \frac{\zeta_2}{1 + \tau_2} + (M_2 w_1 + e_2 b_1) \right) = 0, \quad (3.36)$$

$$\gamma_1^t ((-1 + \epsilon) e_u - (U w_1 + e_u b_1) - \eta_2) = 0, \quad (3.37)$$

$$\delta_1^t \eta_2 = 0, \quad (3.38)$$

$$\mu_1^t \zeta_2 = 0, \quad (3.39)$$

$$\beta_1, \gamma_1, \delta_1, \mu_1 \geq 0. \quad (3.40)$$

On solving equations (3.32) and (3.33), we get

$$v_1 = -(E^t E)^{-1} [F^t (\beta_1 - \varrho_2) - O^t \gamma_1], \quad (3.41)$$

where

$$E = \begin{bmatrix} M_1 & e_1 \end{bmatrix}, F = \begin{bmatrix} M_2 & e_2 \end{bmatrix} \text{ and } O = \begin{bmatrix} U & e_u \end{bmatrix}. \quad (3.42)$$

From (3.34) and (3.40), we obtain

$$0 \leq \beta_1 \leq C_1 e_2 (1 + \tau_2). \quad (3.43)$$

From (3.35) and (3.40), we get $0 \leq \gamma_1 \leq C_u e_u$. From Lagrangian (3.31) and equations (3.32)-(3.35), Wolfe dual obtained for primal problem (3.29) is:

$$\begin{aligned} \max_{\beta_1, \gamma_1} \quad & -\frac{1}{2} ((\beta_1 - \varrho_2)^t F - \gamma_1^t O) (E^t E)^{-1} (F^t (\beta_1 - \varrho_2) - O^t \gamma_1) + e_2^t (\beta_1 - \varrho_2) + (\epsilon - 1) e_u^t \gamma_1 \\ \text{subject to} \quad & 0 \leq \beta_1 \leq C_1 e_2 (1 + \tau_2), \\ & 0 \leq \gamma_1 \leq C_u e_u. \end{aligned} \quad (3.44)$$

Setting $\alpha_1 = \beta_1 - \varrho_2$, the dual problem changes to

$$\begin{aligned} \max_{\alpha_1, \gamma_1} \quad & -\frac{1}{2}(\alpha_1^t F - \gamma_1^t O)(E^t E)^{-1}(F^t \alpha_1 - O^t \gamma_1) + e_2^t \alpha_1 + (\epsilon - 1)e_u^t \gamma_1 \\ \text{subject to} \quad & -\varrho_2 \leq \alpha_1 \leq C_1 e_2(1 + \tau_2) - \varrho_2, \\ & 0 \leq \gamma_1 \leq C_u e_u. \end{aligned} \quad (3.45)$$

Similarly, writing the Lagrangian for problem (3.30) and solving using K.K.T. conditions, we obtain:

$$v_2 = (F^t F)^{-1}[E^t(\beta_2 - \varrho_1) - O^t \gamma_2]. \quad (3.46)$$

The Wolfe dual problem of (3.30) can be derived as

$$\begin{aligned} \max_{\alpha_2, \gamma_2} \quad & -\frac{1}{2}(\alpha_2^t E - \gamma_2^t O)(F^t F)^{-1}(E^t \alpha_2 - O^t \gamma_2) + e_1^t \alpha_2 + (\epsilon - 1)e_u^t \gamma_2 \\ \text{subject to} \quad & -\varrho_1 \leq \alpha_2 \leq C_2 e_1(1 + \tau_1) - \varrho_1, \\ & 0 \leq \gamma_2 \leq C_u e_u, \end{aligned} \quad (3.47)$$

where $\alpha_2 = \beta_2 - \varrho_1$ and β_2, γ_2 are Lagrange multipliers. After determining α_k, γ_k , calculate w_k, b_k for $k = 1, 2$. For a test point say x , the label can be determined by $f(x)$, where

$$f(x) = \underset{j=1,2}{\operatorname{argmin}} |x^t w_k + b_k|. \quad (3.48)$$

The block diagram in Figure 3.7 represents the method to obtain the optimal plane (in Figure 3.7 m_1 is same as l_1). Algorithm 3.1 consists of framework to get the optimal hyperplane.

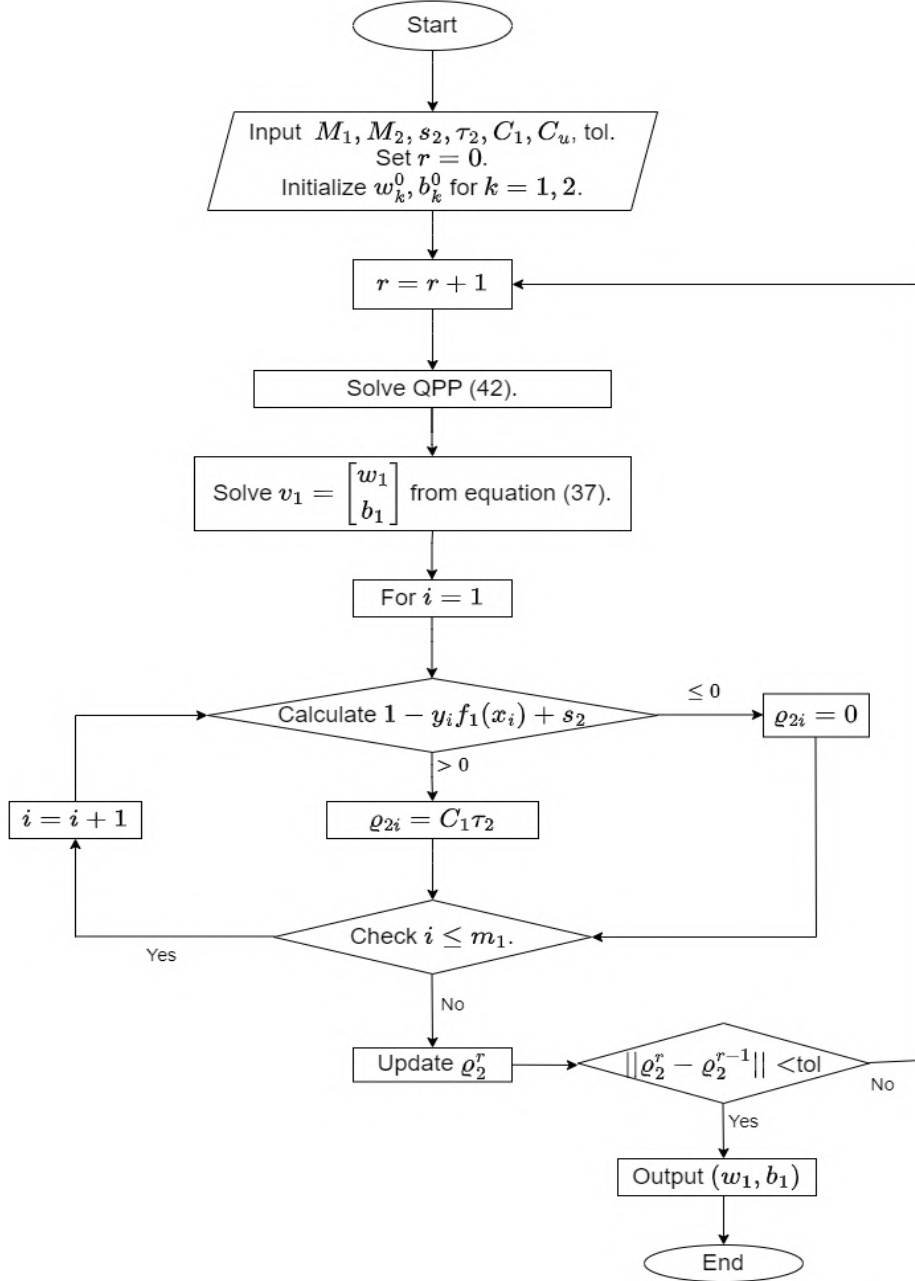


Figure 3.7: Block diagram to obtain the unknowns (w_1, b_1) of the first hyperplane in equation (3.17).

Algorithm 3.1 CCCP for proposed Tpin-UTWSVM

Input: Training data $\{(x_1, y_1), \dots, (x_m, y_m)\}$, testing data set and tolerance value 'tol'.

Output: The labels of testing data.

- Choose appropriate $\tau_1, \tau_2 \in [0, 1]$, $s_1, s_2 \geq 0$ and the positive regularization parameters C_1, C_2, C_u .
 - Obtain the optimal solution (w_1, b_1) and (w_2, b_2) .
 - [1] For iteration variable $r = 0$, initialize $w_1^0, b_1^0, w_2^0, b_2^0$ and calculate ϱ_2^0, ϱ_1^0 using equations (3.25) and (3.27).
 - [2] Solve dual problem (3.45) and (3.47) in r^{th} iteration to get α_j^r, γ_j^r , thus (w_j^r, b_j^r) using equations (3.41) and (3.46) for $j = 1, 2$, respectively. Compute $f_j(x_i)$ for $i = 1, 2, \dots, l$.
 - [3] Update ϱ_2 and ϱ_1 by (3.25) and (3.27), respectively.
 - [4] If $\|\varrho_i^r - \varrho_i^{r-1}\| < \text{tol}$, stop the iteration and obtain optimal (w_i, b_i) , $i = 1, 2$, else set $r = r + 1$ and repeat step 2.
 - For a test point x , determine the label using equation (3.48).
-

3.2.1.2 Non-linear case

For the non-linear case, we map the data points to some higher dimensional space. Let the map be θ , so x is mapped to $\theta(x)$. Also, introduce the kernel function $K(x_i, x_j) = (\theta(x_i), \theta(x_j))$. The positive and negative non parallel kernel generated surfaces are given by

$$K(x^t, M^t)u_1 + b_1 = 0 \text{ and } K(x^t, M^t)u_2 + b_2 = 0, \quad (3.49)$$

respectively, where $M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$ and $u_1, u_2 \in \mathbb{R}^m$; $b_1, b_2 \in \mathbb{R}$. Following a similar algorithm as that of the linear case, we get the primal problem:

$$\min_{u_1, b_1} \frac{1}{2} \|K(M_1, M^t)u_1 + e_1 b_1\|^2 + C_u \sum_{l=1}^u \lambda[f_1(\theta(x_l^*))] + C_1 \sum_{x_i \in M_2} \mathbb{P}_{\tau_2, s_2}(1 - y_i f_1(x_i)) \quad (3.50)$$

and

$$\min_{u_2, b_2} \frac{1}{2} \|K(M_2, M^t)u_2 + e_2 b_2\|^2 + C_u \sum_{l=1}^u \lambda[f_2(\theta(x_l^*))] + C_2 \sum_{x_i \in M_1} \mathbb{P}_{\tau_1, s_1}(1 - y_i f_2(x_i)), \quad (3.51)$$

where C_1, C_2, C_u are the positive regularization parameters. M_1, M_2 denote the positive and negative class data points respectively, $\tau_1, \tau_2 \in [0, 1]$ and s_1, s_2 are positive parameters. $\mathbb{P}_{\tau_1, s_1}$ and $\mathbb{P}_{\tau_2, s_2}$ signify the truncated pinball loss as shown in equation (2.10). Also, $\lambda[f_j(\theta(x_l^*))]$ ($j = 1, 2$) is the ϵ insensitive loss shown in equation (2.22). Introducing the slack variables and applying the CCCP, the primal forms obtained are given by:

$$\begin{aligned} \min_{u_1, b_1} \quad & \frac{1}{2} \|K(M_1, M^t)u_1 + e_1 b_1\|^2 + C_u e_u^t \eta_2 + C_1 e_2^t \zeta_2 \\ & - \varrho_2^t (e_2 + K(M_2, M^t)u_1 + e_2 b_1) \\ \text{subject to} \quad & (K(U, M^t)u_1 + e_u b_1) + \eta_2 \geq (-1 + \epsilon)e_u, \quad \eta_2 \geq 0, \\ & - (K(M_2, M^t)u_1 + e_2 b_1) + \frac{\zeta_2}{1 + \tau_2} \geq e_2, \quad \zeta_2 \geq 0 \end{aligned} \quad (3.52)$$

and

$$\begin{aligned} \min_{u_2, b_2} \quad & \frac{1}{2} \|K(M_2, M^t)u_2 + e_2 b_2\|^2 + C_u e_u^t \eta_1 + C_2 e_1^t \zeta_1 \\ & - \varrho_1^t (e_1 - (K(M_1, M^t)u_2 + e_1 b_2)) \\ \text{subject to} \quad & - (K(U, M^t)u_2 + e_u b_2) + \eta_1 \geq (-1 + \epsilon)e_u, \quad \eta_1 \geq 0, \\ & (K(M_1, M^t)u_2 + e_1 b_2) + \frac{\zeta_1}{1 + \tau_1} \geq e_1, \quad \zeta_1 \geq 0, \end{aligned} \quad (3.53)$$

where ζ_1, ζ_2 and η_1, η_2 are the slack variables. Also, ϱ_2, ϱ_1 are defined similar to equations (3.25) and (3.27), respectively. Proceeding in the similar way as done for the linear case,

the dual obtained for QPP (3.52) is given by:

$$\begin{aligned} \max_{\alpha_1, \gamma_1} \quad & -\frac{1}{2}(\alpha_1^t F - \gamma_1^t O)(E^t E)^{-1}(F^t \alpha_1 - O^t \gamma_1) + e_2^t \alpha_1 + (\epsilon - 1)e_u^t \gamma_1 \\ \text{subject to} \quad & -\varrho_2 \leq \alpha_1 \leq C_1 e_2(1 + \tau_2) - \varrho_2, \\ & 0 \leq \gamma_1 \leq C_u e_u. \end{aligned} \quad (3.54)$$

Similarly, the dual problem of QPP (3.53), can be derived as

$$\begin{aligned} \max_{\alpha_2, \gamma_2} \quad & -\frac{1}{2}(\alpha_2^t E - \gamma_2^t O)(F^t F)^{-1}(E^t \alpha_2 - O^t \gamma_2) + e_1^t \alpha_2 + (\epsilon - 1)e_u^t \gamma_2 \\ \text{subject to} \quad & -\varrho_1 \leq \alpha_2 \leq C_2 e_1(1 + \tau_1) - \varrho_1, \\ & 0 \leq \gamma_2 \leq C_u e_u, \end{aligned} \quad (3.55)$$

where $E = [K(M_1, M^t) \ e_1]$, $F = [K(M_2, M^t) \ e_2]$ and $O = [K(U, M^t) \ e_u]$ and $\alpha_1 = \beta_1 - \varrho_2$, $\alpha_2 = \beta_2 - \varrho_1$. $\beta_1, \beta_2, \gamma_1$, and γ_2 are the vectors of non-negative Lagrange multipliers. After getting the unknowns, label of a test point (say) x can be determined by:

$$f(x) = \underset{k=1,2}{\operatorname{argmin}} (K(x^t, M^t)u_k + b_k). \quad (3.56)$$

3.2.2 Properties of Tpin-UTWSVM

In this section, we analyze Tpin-UTWSVM with respect to noise insensitivity and sparsity. Also, we will show the bounds on the number of negative and positive boundary errors.

3.2.2.1 Noise insensitivity and sparsity

The main advantage of considering truncated pinball loss is insensitivity to noise around the decision boundary and sparsity of the model. Subgradient of truncated pinball loss

function (2.10) is defined as:

$$\partial \mathbb{P}_{\tau,s}(x, y, f(x)) = \begin{cases} 0, & \text{if } 1 - yf(x) < -s, \\ [-\tau, 0], & \text{if } 1 - yf(x) = -s, \\ -\tau, & \text{if } -s < 1 - yf(x) < 0, \\ [-\tau, 1], & \text{if } 1 - yf(x) = 0, \\ 1, & \text{if } 1 - yf(x) > 0. \end{cases} \quad (3.57)$$

For convenience, we illustrate the property of noise insensitivity and sparsity for linear positive classifier, others can be analyzed similarly. Using the K.K.T. condition of optimality in problem (3.18), we get:

$$0 \in M_1^t(M_1 w_1 + e_1 b_1) + C_u \sum_{l=1}^u \partial \lambda[f_1(x_l^*)] + C_1 \sum_{x_i \in M_2} \partial \mathbb{P}_{\tau_2, s_2}(1 - y_i f_1(x_i)). \quad (3.58)$$

Here, 0 denotes a vector having each component zero. For a fixed value of w_1 and b_1 , we split the whole index set ($i = 1, 2, \dots, l_2$) into the following five sets:

$$\begin{aligned} S_0^{w_1, b_1} &= \{i \mid 1 - y_i(w_1^t x_i + b_1) < -s_2\}, \\ S_1^{w_1, b_1} &= \{i \mid 1 - y_i(w_1^t x_i + b_1) = -s_2\}, \\ S_2^{w_1, b_1} &= \{i \mid -s_2 < 1 - y_i(w_1^t x_i + b_1) < 0\}, \\ S_3^{w_1, b_1} &= \{i \mid 1 - y_i(w_1^t x_i + b_1) = 0\}, \\ S_4^{w_1, b_1} &= \{i \mid 1 - y_i(w_1^t x_i + b_1) > 0\}. \end{aligned} \quad (3.59)$$

The data points in $S_0^{w_1, b_1}$ have no SVs so it does not contribute to w_1 . $S_0^{w_1, b_1}$ is related to the sparsity of the proposed Tpin-UTWSVM. The number of negative class data points in $S_0^{w_1, b_1}$ is determined by the parameter s_2 . If s_2 is small, $S_0^{w_1, b_1}$ has more number of points and the model become sparser. The truncated pinball loss approaches hinge loss as $s_2 \rightarrow 0$.

Contrarily, larger s_2 means less negative samples $S_0^{w_1, b_1}$ has and the model is insensitive to noise, i.e., as $s_2 \rightarrow \infty$, the sparsity of the model will be lost entirely.

For the ϵ insensitive loss function represented by equation (3.20), the subgradient can be written as:

$$\partial\lambda[w_1^t x_l^* + b_1] = \begin{cases} -1, & \text{if } w_1^t x_l^* + b_1 < -\epsilon, \\ [-1, 0], & \text{if } w_1^t x_l^* + b_1 = -\epsilon, \\ 0, & \text{if } -\epsilon < w_1^t x_l^* + b_1 < \epsilon, \\ [0, 1], & \text{if } w_1^t x_l^* + b_1 = \epsilon, \\ 1, & \text{if } w_1^t x_l^* + b_1 > \epsilon. \end{cases} \quad (3.60)$$

Again splitting the index set of the universum data, we have the following five sets:

$$\begin{aligned} P_0 &= \{j \in U \mid w_1^t x_l^* + b_1 < -\epsilon\}, \\ P_1 &= \{j \in U \mid w_1^t x_l^* + b_1 = -\epsilon\}, \\ P_2 &= \{j \in U \mid -\epsilon < w_1^t x_l^* + b_1 < \epsilon\}, \\ P_3 &= \{j \in U \mid w_1^t x_l^* + b_1 = \epsilon\}, \\ P_4 &= \{j \in U \mid w_1^t x_l^* + b_1 > \epsilon\}. \end{aligned} \quad (3.61)$$

From equation (3.58), and using the index sets $S_k^{w_1, b_1}$ and P_k , where $k = 0, 1, 2, 3, 4$, we obtain

$$\begin{aligned} & \frac{M_1}{C_1} (M_1 w_1 + e_1 b_1) + \frac{C_u}{C_1} \left(\sum_{j \in P_0} x_j^* + \sum_{j \in P_1} a_j x_j^* + \sum_{j \in P_3} b_j x_j^* + \sum_{j \in P_4} x_j^* \right) \\ & + \left(\sum_{i \in S_1^{w_1, b_1}} \psi_i x_i + \sum_{i \in S_2^{w_1, b_1}} -\tau_1 x_i + \sum_{i \in S_3^{w_1, b_1}} \xi_i x_i + \sum_{i \in S_4^{w_1, b_1}} x_i \right) = 0 \end{aligned} \quad (3.62)$$

where $a_j \in [-1, 0]$, $b_j \in [0, 1]$ and $\psi_i \in [-\tau, 0]$, $\xi_i \in [-\tau_1, 1]$. In comparison to the index set

$S_0^{w_1, b_1}$, $S_2^{w_1, b_1}$ and $S_4^{w_1, b_1}$, the number of samples in $S_1^{w_1, b_1}$ and $S_3^{w_1, b_1}$ is very less, so we will not consider them in the analysis. Fixing the value of s_2 , number of samples in $S_0^{w_1, b_1}$, $S_2^{w_1, b_1}$ and $S_4^{w_1, b_1}$ depend on τ_2 . If the number of samples in $S_0^{w_1, b_1}$ is getting effected, it also effects the sparsity of the model. The number of samples in $S_4^{w_1, b_1}$ is small when τ_2 is small, which results in a model sensitive to feature noise around the decision hyperplane. Conversely, if τ_2 is large, $S_k^{w_1, b_1}$ ($k \in \{0, 2, 4\}$) will have large number of samples, resulting in less sensitive model. Changing τ_2 changes number of samples in $S_0^{w_1, b_1}$, hence also effecting the sparsity of model. Thus, both s_2 , τ_2 determine the noise sensitivity and sparsity of the model. The bar graph shown in Figure 3.8a shows the effect of changing τ on SVs, hence sparsity of binary UCI datasets with a fixed s . Similarly, the bar graph in Figure 3.8b shows the consequence of changing s with fixed τ on the SVs, thus sparsity of the binary UCI datasets. Table 3.8 shows the comparison among the proposed Tpin-UTWSVM model and the baseline models.

The following proposition which is also followed by Pin-GTSVM [101] gives the upper bound on the negative and positive boundary errors.

Proposition 1. Assume r_1 represents the number of negative boundary errors. Then, there exists an upper bound given by $\frac{e_2^t \beta_1}{C_1(1+\tau_2)}$ on number of negative boundary errors.

Proof: For negative data points having error, i.e., $\zeta_2 \neq 0$. We have from equation (3.39), $\mu_1 = 0$ which implies $\beta_1 = (1 + \tau_2)C_1 e_2$.

$$\beta_{1i} = (1 + \tau_2)C_1 \quad \forall i = 1, 2, \dots, m_2.$$

As per assumption, r_1 points has $\zeta_2 \neq 0$.

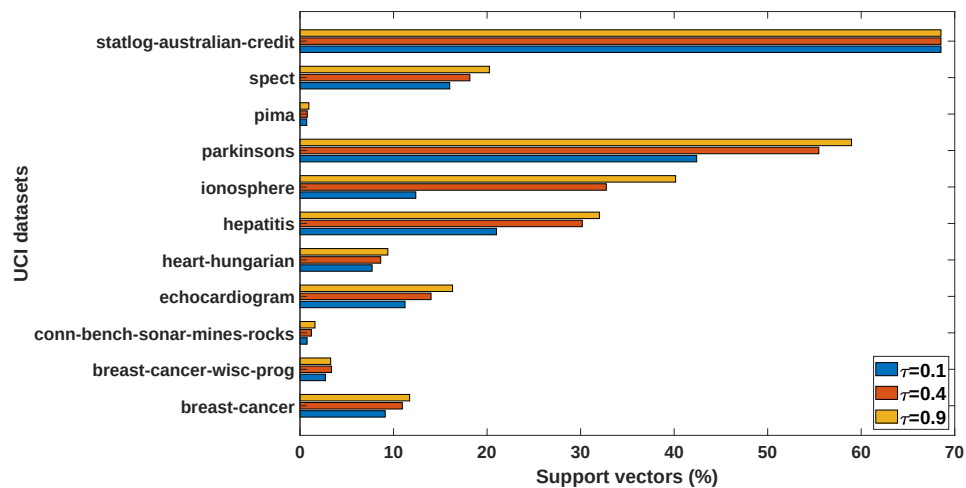
$$\begin{aligned} \implies r_1 \beta_{1i} &\leq e_2^t \beta_1 \\ \implies r_1(1 + \tau_2) C_1 &\leq e_2^t \beta_1 \\ \implies r_1 &\leq \frac{e_2^t \beta_1}{C_1(1 + \tau_2)}. \end{aligned}$$

Proposition 2. Assume r_2 represents the number of positive boundary errors. Then, there exists an upper bound given by $\frac{e_1^t \beta_2}{C_2(1 + \tau_1)}$ on number of positive boundary errors.

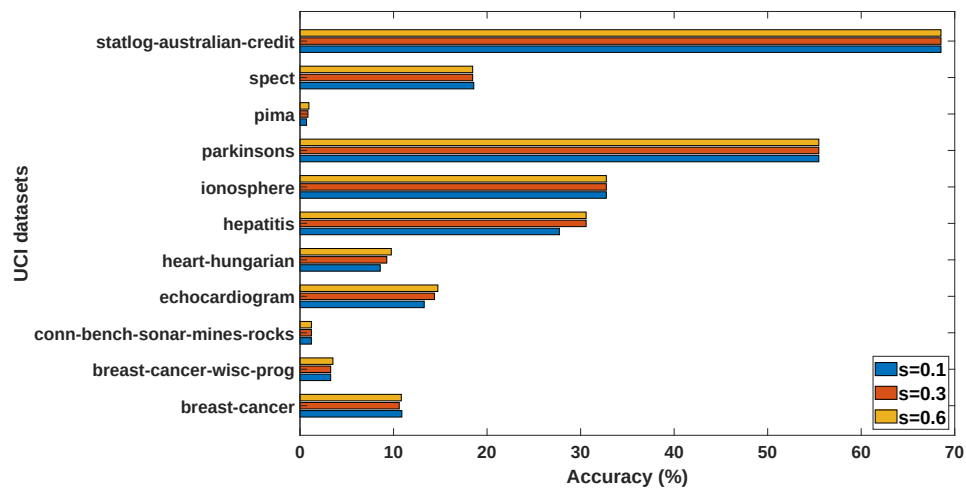
Proof: Proceed similarly as above.

3.2.2.2 Complexity

The computational complexity of SVM is $O(l^3)$ [77], l being the number of samples. In TWSVM, we divide the samples into two parts, so its computational complexity is approximately $O(2 * (m/2)^3)$. The computational complexity of Pin-GTSVM and TWSVM are similar [101]. Due to the non-convexity of the proposed model, the complexity may increase. However, CCCP acts as a speedup technique and compensates for the complexity. The computational cost of taking inverse of a matrix is $O(n^3)$ [133], where n is the order of the matrix. Also, in the proposed Tpin-UTWSVM, we considered universum data leading to complexity $O((l + 2u)^3/4)$ [134], where u denotes the number of universum data. Thus, universum data is responsible for the increased computational time as compared to the baseline models. Hence, the approximate complexity of the proposed Tpin-UTWSVM is $O((l + 2u)^3/4)$. To make the training of the proposed Tpin-UTWSVM efficient, we used successive overrelaxation (SOR) technique [135]. It has the advantage of processing datasets without storing them in memory [23]. We have also applied SOR technique to the TWSVM, UTWSVM and Pin-GTSVM, thus boosting their performance.



(a) Bar graph representing the effect of changing τ with fixed $s = 0.2$ on the SVs of noise free data ($r = 0$).



(b) Bar graph representing the effect of changing s with fixed $\tau = 0.4$ on the SVs of noise free data ($r = 0$).

Figure 3.8: Bar graph illustrating the effect of loss parameters τ and s on the sparsity of the proposed Tpin-UTWSVM model over UCI datasets.

Table 3.8: Comparison of the models under study.

pin SVM [17]	TWSVM [22]	UTSVM [26]	Pin-GTSVM [101]	Proposed Tpin-UTWSVM
Uses pinball loss for imposing penalty to both correctly and wrongly classified data points.	Uses hinge loss for imposing penalty to misclassified data points.	Uses hinge loss as penalty for misclassified data points and ϵ insensitive loss for universum data.	Uses pinball loss for imposing penalty on both correctly and incorrectly classified data points.	Uses truncated pinball loss for imposing penalty on both correctly and incorrectly classified data points.
Insensitive to noise.	Sensitive to noise.	Sensitive to noise.	Insensitive to noise.	Insensitive to noise.
The model is not sparse.	The model is sparse.	The model is sparse.	The model is not sparse.	The model is sparse.
Robust to outliers.	Effected by outliers.	Effected by outliers.	Robust to outliers.	Robust to outliers.
Leads to convex optimization problems.	Leads to convex optimization problems.	Leads to convex optimization problems.	Leads to convex optimization problems.	Leads to non-convex optimization problems.
It has very high complexity of $O(l^3)$.	It is efficient than pin-SVM having complexity $\approx O((l/2)^3)$ [22].	The complexity of the model is $O((l + 2u)^3)$ [134]	Its complexity is similar to TWSVM [101].	It has complexity similar to UTSVM, however, the sparsity of the model leads to reduced complexity as compared to UTSVM.
The pinball loss function has theoretical properties of minimization leading to Bayes's classifier and bounded misclassification error by Zhang's inequality.	Hinge loss is not a Bayesian classifier and does not have a bounded misclassification error.	Hinge loss is not a Bayesian classifier and does not have a bounded misclassification error.	The loss function of the model has the theoretical properties of being a Bayesian classifier and having a bounded misclassification error.	The truncated pinball loss function has the theoretical properties of pinball loss [74].

3.2.3 Experiments

In this section, we performed numerical experiments to compare the proposed Tpin-UTWSVM model with the baseline models, i.e., pin-SVM [17], TWSVM [22], UTSVM [26], Pin-GTSVM [101] and carried out statistical tests to demonstrate the superiority among the compared models. We also evaluate the aforementioned models for the classification of EEG signals and detection of Alzheimer's disease.

3.2.3.1 Experimental setup

All the experiments have been performed on a system with MATLAB R2022a version and Windows 10 operating system, with Intel(R) Xenon(R) W-2223 CPU @ 3.6 GHz processor and 64 GB RAM. To obtain the optimal parameters, we used grid search with ten fold cross validation (CV) which considers each possible combination of the parameter values and split the training datasets into 10 parts. Taking one part as the validation data and remaining as the training data, we obtain classification accuracy and training time. Repeating it 9 more times with each different part as the validation data, we get 10 different classification accuracies with training times. Averaging the 10 accuracies and training times, we get the accuracy and time with particular set of parameter values. The combination of parameter values corresponding to highest classification accuracy are called optimal hyperparameters. We have splitted training and testing data randomly in the ratio 70 : 30. Using the optimal hyperparameters, obtain the accuracy on testing data and refer it as testing accuracy. In further article, we will term testing accuracy as accuracy. For the non-linear case, we have considered Gaussian kernel given by

$$K(x_p, x_q) = \exp\left(-\frac{\|x_p - x_q\|^2}{2\sigma^2}\right),$$

where σ signifies the kernel parameter. For the comparison of the proposed Tpin-UTWSVM model, we have evaluated the models on 19 benchmark binary datasets from the UCI repository [136], Alzheimer's disease data publicly available on Alzheimer's Disease Neuroimaging Initiative (ADNI), and EEG data publicly available in the article [137]. The training and testing data are corrupted by the same zero mean Gaussian noise. The ratio of variance of noise to that of each feature (denoted by r) is at $r = 0, 0.05$ and 0.1 [17]. We have employed z-score normalization to the datasets.

3.2.3.2 Parameter selection

For the baseline models and the proposed Tpin-UTWSVM model, we set the penalty parameters $C_1 = C_2$, to ease the computational complexity. The parameters C_k , ($k = 1, 2$ and u) of the proposed Tpin-UTWSVM model and baseline models, i.e., pin-SVM [17], TWSVM [22], UTSVM [26], Pin-GTSVM [101] have been chosen from the range: $[10^{-5}, \dots, 10^5]$, σ is chosen from $[2^{-5}, \dots, 2^5]$ [138]. The truncated pinball loss parameter for the proposed Tpin-UTWSVM are chosen as $s_1 = s_2 = 0.25$, $\tau_1 = \tau_2 = 0.5$ following the article [139]. For Pin-GTSVM, τ is considered as 0.5. The parameter ϵ in the proposed Tpin-UTWSVM model and UTSVM is chosen from the range $[0.1, 0.3, 0.5, 0.6]$ [27]. τ for pin-SVM is taken from the range $[0.1, 0.2, 0.5, 1]$ from [17]. To initialize the variables for the CCCP algorithm, we have followed initialization method in [140]. Following the article [101], for evaluation of the models, we have calculated the metric accuracy. To evaluate the compared models in a better way, two additional metrics precision and recall are being considered. These metrics are defined as follows:

$$\text{Accuracy} = \frac{tp + tn}{tp + fp + tn + fn}, \text{ Recall} = \frac{tp}{tp + fn}, \text{ Precision} = \frac{tp}{tp + fp},$$

where tp , tn and fp , fn signify the correctly classified positive, negative samples and misclassified positive, negative samples, respectively.

3.2.3.3 Experimental analysis on the binary datasets

The classification performance of the proposed Tpin-UTWSVM along with models pin-SVM [17], TWSVM [22], UTSVM [26] and Pin-GTSVM [101] on the binary datasets is shown in Table 3.9, where sd denotes the standard deviation of the model on each dataset. Accuracy with standard deviation, recall and precision for the three noise levels $r = 0, 0.05, 0.1$ have been shown in Table 3.9 for each model on the randomly chosen 19 UCI datasets. For $r = 0$, the average accuracies of pin-SVM, TWSVM, UTSVM, Pin-GTSVM and the proposed Tpin-UTWSVM are 79.76, 79.88, 79.79, 76.91 and 80.93, respectively. For the same order of models, the average accuracies are 79.15, 80.18, 77.63, 77.6 and 80.74 for $r = 0.05$ and 78.9, 80.33, 77.75, 77.23 and 80.68 for $r = 0.1$. Thus, though the level of noise increases, the performance of the proposed Tpin-UTWSVM remains the best. There may be a possibility that higher accuracy of the model over one dataset compensates for lower accuracy over other datasets, hence, influencing the average accuracy. In order to overcome such biasness, we assigned ranks to each dataset for every model. The model with the best performance on a dataset is assigned a rank of 1, with higher ranks assigned to less effective models. For models having the same performance on a dataset, consider same ranks and take the average of the considered ranks and assign it to the models having same performance. The lowest average rank signifies the superior model. Among the models into consideration, for all the three values of r , the proposed Tpin-UTWSVM has the best average rank, i.e., 2.55, 2.32 and 2.18 for $r = 0, 0.05$ and 0.1 , respectively. The training time and optimal hyperparameters for the models under comparison over the 19 UCI datasets are shown in Table 3.10. The Table 3.10 also includes size

Table 3.9: Performance of the proposed Tpin-UTWSVM model with baseline models on UCI binary datasets with Gaussian kernel.

Dataset	r	pin-SVM [17] (Accuracy $\pm$ sd) (Recall, Precision)	TWSVM [22] (Accuracy $\pm$ sd) (Recall, Precision)	UTSVM [26] (Accuracy $\pm$ sd) (Recall, Precision)	Pin-GTSVM [101] (Accuracy $\pm$ sd) (Recall, Precision)	Tpin-UTWSVM (Accuracy $\pm$ sd) (Recall, Precision)
blood	0	(79.91 $\pm$ 5.88)	(76.34 $\pm$ 6.28)	(78.57 $\pm$ 5.12)	(63.84 $\pm$ 7.71)	(76.79 $\pm$ 6.48)
		(0.377, 0.7667)	(0.2951, 0.6429)	(0.3279, 0.7407)	(0.3934, 0.3529)	(0.2951, 0.6667)
	0.05	(76.34 $\pm$ 5.26)	(77.68 $\pm$ 5.91)	(73.66 $\pm$ 5.8)	(72.32 $\pm$ 6.45)	(78.13 $\pm$ 5.09)
		(0.3443, 0.6176)	(0.3115, 0.7037)	(0.0328, 1)	(0, 0)	(0.2951, 0.75)
	0.1	(72.77 $\pm$ 6.84)	(77.23 $\pm$ 5.18)	(74.11 $\pm$ 6.04)	(72.32 $\pm$ 6.63)	(77.23 $\pm$ 6.26)
		(0.3607, 0.5)	(0.3279, 0.6667)	(0.1148, 0.6364)	(0, 0)	(0.3279, 0.6667)
breast-cancer	0	(74.42 $\pm$ 8.23)	(74.42 $\pm$ 11.35)	(80.23 $\pm$ 10.87)	(72.09 $\pm$ 10.27)	(76.74 $\pm$ 10.12)
		(0.4348, 0.5263)	(0.5652, 0.52)	(0.3043, 0.875)	(0, 0)	(0.4348, 0.5882)
	0.05	(74.42 $\pm$ 7.17)	(74.42 $\pm$ 11.56)	(76.74 $\pm$ 10.87)	(70.93 $\pm$ 8.43)	(75.58 $\pm$ 11.17)
		(0.4348, 0.5263)	(0.5652, 0.52)	(0.1739, 0.8)	(0.3913, 0.45)	(0.4783, 0.55)
	0.1	(73.26 $\pm$ 9.44)	(74.42 $\pm$ 11.83)	(79.07 $\pm$ 8.88)	(69.77 $\pm$ 11.11)	(75.58 $\pm$ 11.89)
		(0.2174, 0.5)	(0.5652, 0.52)	(0.3478, 0.7273)	(0.5652, 0.4483)	(0.5217, 0.5455)
breast-cancer-wisc	0	(95.71 $\pm$ 2.57)	(96.67 $\pm$ 2.44)	(95.24 $\pm$ 2.31)	(95.71 $\pm$ 1.73)	(96.67 $\pm$ 2.44)
		(0.961, 0.925)	(0.961, 0.9487)	(0.9351, 0.9351)	(0.9221, 0.9595)	(0.961, 0.9487)
	0.05	(95.71 $\pm$ 2.52)	(93.81 $\pm$ 2.2)	(94.76 $\pm$ 3.05)	(94.76 $\pm$ 1.73)	(94.29 $\pm$ 1.38)
		(0.961, 0.925)	(0.8961, 0.9324)	(0.8961, 0.9583)	(0.8961, 0.9583)	(0.8831, 0.9577)
	0.1	(96.19 $\pm$ 2.03)	(94.76 $\pm$ 1.99)	(96.19 $\pm$ 2.51)	(95.24 $\pm$ 1.73)	(94.29 $\pm$ 1.44)
		(0.974, 0.9259)	(0.8961, 0.9583)	(0.974, 0.9259)	(0.9221, 0.9467)	(0.8831, 0.9577)
breast-cancer-wisc-prog	0	(74.58 $\pm$ 5.96)	(74.58 $\pm$ 4.87)	(77.97 $\pm$ 10.06)	(74.58 $\pm$ 5.91)	(72.88 $\pm$ 12.58)
		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0.3077, 0.3636)
	0.05	(74.58 $\pm$ 5.96)	(74.58 $\pm$ 4.92)	(69.49 $\pm$ 8.96)	(74.58 $\pm$ 4.92)	(72.88 $\pm$ 11.22)
		(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0.3077, 0.3636)
	0.1	(74.58 $\pm$ 5.96)	(76.27 $\pm$ 9.61)	(72.88 $\pm$ 9.48)	(74.58 $\pm$ 4.92)	(76.27 $\pm$ 5.96)
		(0, 0)	(0.0769, 0.3333)	(0, 0)	(0, 0)	(0, 0)
conn-bench-sonar-mines-rocks	0	(82.26 $\pm$ 11.99)	(79.03 $\pm$ 5.61)	(82.26 $\pm$ 9.48)	(85.48 $\pm$ 9.75)	(82.26 $\pm$ 7.28)
		(0.8788, 0.8056)	(0.6364, 0.9545)	(0.8485, 0.8235)	(0.7273, 1)	(0.7576, 0.8929)
	0.05	(80.65 $\pm$ 11.99)	(85.48 $\pm$ 10.21)	(80.65 $\pm$ 6.31)	(85.48 $\pm$ 10.21)	(85.48 $\pm$ 7.93)
		(0.8485, 0.8)	(0.7273, 1)	(0.8485, 0.8)	(0.7273, 1)	(0.8182, 0.9)
	0.1	(82.26 $\pm$ 10.64)	(85.48 $\pm$ 9.62)	(79.03 $\pm$ 7.42)	(85.48 $\pm$ 9.53)	(85.48 $\pm$ 6.36)
		(0.8485, 0.8235)	(0.7273, 1)	(0.8182, 0.7941)	(0.7273, 1)	(0.7879, 0.9286)
echocardiogram	0	(84.62 $\pm$ 11.16)	(87.18 $\pm$ 10.33)	(89.74 $\pm$ 9.28)	(74.36 $\pm$ 15.57)	(89.74 $\pm$ 10.76)
		(0.4545, 1)	(0.5455, 1)	(0.6364, 1)	(0.6364, 0.5385)	(0.6364, 1)
	0.05	(84.62 $\pm$ 11.16)	(89.74 $\pm$ 10.32)	(87.18 $\pm$ 10.76)	(71.79 $\pm$ 14.06)	(89.74 $\pm$ 10.76)
		(0.4545, 1)	(0.6364, 1)	(0.5455, 1)	(0.6364, 0.5)	(0.6364, 1)
	0.1	(82.05 $\pm$ 12.68)	(89.74 $\pm$ 9.4)	(87.18 $\pm$ 10.76)	(71.79 $\pm$ 16.52)	(89.74 $\pm$ 10.33)
		(0.3636, 1)	(0.6364, 1)	(0.5455, 1)	(0.6364, 0.5)	(0.6364, 1)
haberman-survival	0	(69.57 $\pm$ 11.78)	(69.57 $\pm$ 12.39)	(73.91 $\pm$ 11.34)	(67.39 $\pm$ 13.48)	(69.57 $\pm$ 11.59)
		(0.0741, 0.4)	(0.0741, 0.4)	(0.2222, 0.6667)	(0, 0)	(0.037, 0.3333)
	0.05	(69.57 $\pm$ 12.29)	(69.57 $\pm$ 12.39)	(71.74 $\pm$ 12.61)	(70.65 $\pm$ 12.43)	(70.65 $\pm$ 12.25)
		(0.0741, 0.4)	(0.0741, 0.4)	(0.1111, 0.6)	(0, -)	(0.037, 0.5)
	0.1	(69.57 $\pm$ 12.1)	(70.65 $\pm$ 12.23)	(73.91 $\pm$ 12.11)	(70.65 $\pm$ 12.8)	(71.74 $\pm$ 12.01)
		(0.0741, 0.4)	(0.1111, 0.5)	(0.2222, 0.6667)	(0, -)	(0.037, 1)
heart-hungarian	0	(86.36 $\pm$ 8.03)	(85.23 $\pm$ 7.77)	(78.41 $\pm$ 11.66)	(84.09 $\pm$ 11.9)	(86.36 $\pm$ 6.59)
		(0.6774, 0.913)	(0.7419, 0.8214)	(0.7419, 0.6765)	(0.7742, 0.7742)	(0.8065, 0.8065)
	0.05	(86.36 $\pm$ 7.48)	(78.41 $\pm$ 9.7)	(81.82 $\pm$ 5.25)	(79.55 $\pm$ 12.03)	(84.09 $\pm$ 10.08)
		(0.6774, 0.913)	(0.7097, 0.6875)	(0.8387, 0.7027)	(0.7419, 0.697)	(0.7419, 0.7931)
	0.1	(86.36 $\pm$ 7.48)	(80.68 $\pm$ 8.2)	(79.55 $\pm$ 12.41)	(78.41 $\pm$ 11.04)	(84.09 $\pm$ 8.87)
		(0.6774, 0.913)	(0.6774, 0.75)	(0.7097, 0.7097)	(0.7419, 0.6765)	(0.7742, 0.7742)
heart-switzerland	0	(54.05 $\pm$ 20.39)	(56.76 $\pm$ 14.34)	(54.05 $\pm$ 14.73)	(54.05 $\pm$ 25.06)	(56.76 $\pm$ 14.34)
		(0, 0)	(0.2353, 0.5714)	(0.3529, 0.5)	(0.4118, 0.5)	(0.2353, 0.5714)
	0.05	(51.35 $\pm$ 16.81)	(56.76 $\pm$ 16.97)	(51.35 $\pm$ 10.74)	(54.05 $\pm$ 18.35)	(56.76 $\pm$ 16.31)
		(0, 0)	(0.2353, 0.5714)	(0.2941, 0.4545)	(0, 0)	(0.2353, 0.5714)
	0.1	(48.65 $\pm$ 14.38)	(56.76 $\pm$ 16.97)	(48.65 $\pm$ 16.48)	(54.05 $\pm$ 18.35)	(56.76 $\pm$ 16.97)
		(0.0588, 0.25)	(0.2353, 0.5714)	(0.1765, 0.375)	(0, 0)	(0.2353, 0.5714)
hepatitis	0	(86.96 $\pm$ 9.6)	(91.3 $\pm$ 12.7)	(91.3 $\pm$ 9.6)	(86.96 $\pm$ 8.72)	(89.13 $\pm$ 11.16)
		(0.875, 0.9722)	(0.925, 0.9737)	(0.95, 0.95)	(0.9, 0.9474)	(0.925, 0.9487)
	0.05	(86.96 $\pm$ 11.35)	(91.3 $\pm$ 10.88)	(91.3 $\pm$ 9.6)	(89.13 $\pm$ 9.74)	(89.13 $\pm$ 10.88)
		(0.875, 0.9722)	(0.925, 0.9737)	(1, 0.9091)	(0.925, 0.9487)	(0.925, 0.9487)
	0.1	(86.96 $\pm$ 11.35)	(91.3 $\pm$ 12.7)	(91.3 $\pm$ 9.6)	(91.3 $\pm$ 12.23)	(91.3 $\pm$ 12.38)
		(0.875, 0.9722)	(0.925, 0.9737)	(0.95, 0.95)	(0.95, 0.95)	(0.925, 0.9737)

Table 3.9: Continued from previous page-Performance of the proposed Tpin-UTWSVM model with baseline models on UCI binary datasets with Gaussian kernel.

Dataset	r	pin-SVM [17] (Accuracy $\pm$ sd) (Recall,Precision)	TWSVM [22] (Accuracy $\pm$ sd) (Recall,Precision)	UTSVM [26] (Accuracy $\pm$ sd) (Recall,Precision)	Pin-GTSVM [101] (Accuracy $\pm$ sd) (Recall,Precision)	Tpin-UTWSVM (Accuracy $\pm$ sd) (Recall,Precision)
ionosphere	0	(90.48 $\pm$ 6.64) (0.8592, 1)	(97.14 $\pm$ 3.43) (0.9859, 0.9722)	(97.14 $\pm$ 7.43) (0.9859, 0.9722)	(96.19 $\pm$ 3.14) (0.9718, 0.9718)	(96.19 $\pm$ 3.47) (0.9577, 0.9855)
	0.05	(89.52 $\pm$ 7.61) (0.8451, 1)	(97.14 $\pm$ 2.86) (0.9859, 0.9722)	(93.33 $\pm$ 4.36) (0.9155, 0.9848)	(95.24 $\pm$ 5.56) (0.9577, 0.9714)	(95.24 $\pm$ 3.95) (0.9437, 0.9853)
	0.1	(91.43 $\pm$ 8.55) (0.8732, 1)	(97.14 $\pm$ 3.91) (0.9859, 0.9722)	(94.29 $\pm$ 4.13) (0.9296, 0.9851)	(94.29 $\pm$ 6.31) (0.9296, 0.9851)	(95.24 $\pm$ 5.49) (0.9577, 0.9714)
molec-biol-promoter	0	(75 $\pm$ 9) (0.7692, 0.6667)	(71.88 $\pm$ 9.93) (0.7692, 0.625)	(65.63 $\pm$ 12.21) (0.7692, 0.5556)	(71.88 $\pm$ 9.93) (0.7692, 0.625)	(68.75 $\pm$ 6.58) (0.6923, 0.6)
	0.05	(75 $\pm$ 9) (0.7692, 0.6667)	(71.88 $\pm$ 9.93) (0.7692, 0.625)	(65.63 $\pm$ 15.5) (0.6923, 0.5625)	(71.88 $\pm$ 9.93) (0.7692, 0.625)	(71.88 $\pm$ 8.89) (0.7692, 0.625)
	0.1	(75 $\pm$ 10.61) (0.7692, 0.6667)	(71.88 $\pm$ 9.93) (0.7692, 0.625)	(65.63 $\pm$ 13.31) (0.8462, 0.55)	(71.88 $\pm$ 9.93) (0.7692, 0.625)	(71.88 $\pm$ 10.61) (0.7692, 0.625)
monks-1	0	(88.62 $\pm$ 4.39) (0.8621, 0.9146)	(98.8 $\pm$ 2.96) (0.977, 1)	(92.81 $\pm$ 4.01) (0.8851, 0.9747)	(91.62 $\pm$ 4.51) (0.954, 0.8925)	(100 $\pm$ 1.81) (1, 1)
	0.05	(86.83 $\pm$ 4.72) (0.8736, 0.8736)	(95.81 $\pm$ 3.02) (0.9195, 1)	(87.43 $\pm$ 4.33) (0.7701, 0.9853)	(94.01 $\pm$ 3.58) (0.9425, 0.9425)	(95.21 $\pm$ 3.59) (0.9425, 0.9647)
	0.1	(85.63 $\pm$ 4.19) (0.8621, 0.8621)	(93.41 $\pm$ 2.96) (0.931, 0.9419)	(84.43 $\pm$ 5.81) (0.8046, 0.8861)	(89.22 $\pm$ 2.48) (0.8736, 0.9157)	(90.42 $\pm$ 4.52) (0.908, 0.908)
parkinsons	0	(89.66 $\pm$ 8.18) (0.9778, 0.898)	(93.1 $\pm$ 4.32) (0.9778, 0.9362)	(91.38 $\pm$ 6.34) (0.9556, 0.9348)	(93.1 $\pm$ 6.29) (0.9778, 0.9362)	(96.55 $\pm$ 5.74) (1, 0.9574)
	0.05	(86.21 $\pm$ 8.37) (0.9778, 0.8627)	(93.1 $\pm$ 5.09) (0.9778, 0.9362)	(93.1 $\pm$ 7.75) (0.9778, 0.9362)	(94.83 $\pm$ 5.46) (0.9778, 0.9565)	(94.83 $\pm$ 5.28) (0.9778, 0.9565)
	0.1	(87.93 $\pm$ 7.96) (1, 0.8654)	(94.83 $\pm$ 5.09) (1, 0.9375)	(89.66 $\pm$ 9.62) (0.9778, 0.898)	(93.1 $\pm$ 5.83) (0.9778, 0.9362)	(94.83 $\pm$ 4.54) (0.9778, 0.9565)
pima	0	(74.35 $\pm$ 4.38) (0.4839, 0.8036)	(76.52 $\pm$ 4.15) (0.5054, 0.8545)	(71.3 $\pm$ 7.89) (0.4946, 0.7077)	(65.22 $\pm$ 5.85) (0.5484, 0.573)	(76.09 $\pm$ 4.65) (0.4839, 0.8654)
	0.05	(74.35 $\pm$ 3.75) (0.4839, 0.8036)	(76.52 $\pm$ 4.28) (0.5054, 0.8545)	(71.3 $\pm$ 6.65) (0.4839, 0.7143)	(67.39 $\pm$ 5.63) (0.5591, 0.6047)	(74.78 $\pm$ 5.22) (0.4194, 0.907)
	0.1	(73.91 $\pm$ 4.17) (0.4946, 0.7797)	(75.65 $\pm$ 3.65) (0.4946, 0.8364)	(73.48 $\pm$ 4.3) (0.5054, 0.7581)	(67.39 $\pm$ 6) (0.5484, 0.6071)	(75.65 $\pm$ 4.12) (0.4731, 0.8627)
pittsburg-bridges-T-OR-D	0	(90.32 $\pm$ 17.82) (0.6667, 0.5)	(87.1 $\pm$ 11.27) (0, 0)	(87.1 $\pm$ 12.05) (0, 0)	(83.87 $\pm$ 12.51) (0, 0)	(87.1 $\pm$ 9.99) (0, 0)
	0.05	(93.55 $\pm$ 17.1) (0.6667, 0.6667)	(87.1 $\pm$ 11.27) (0, 0)	(80.65 $\pm$ 11.76) (0, 0)	(87.1 $\pm$ 12.51) (0, 0)	(83.87 $\pm$ 9.99) (0, 0)
	0.1	(93.55 $\pm$ 14.75) (0.6667, 0.6667)	(87.1 $\pm$ 11.27) (0, 0)	(87.1 $\pm$ 11.76) (0, 0)	(87.1 $\pm$ 9.52) (0, 0)	(87.1 $\pm$ 10.1) (0, 0)
planning	0	(78.18 $\pm$ 15.75) (0.0769, 1)	(78.18 $\pm$ 15.75) (0.0769, 1)	(78.18 $\pm$ 15.75) (0.0769, 1)	(78.18 $\pm$ 15.75) (0.0769, 1)	(78.18 $\pm$ 15.75) (0.0769, 1)
	0.05	(78.18 $\pm$ 15.75) (0.0769, 1)	(78.18 $\pm$ 15.75) (0.0769, 1)	(76.36 $\pm$ 15.84) (0, 0)	(76.36 $\pm$ 15.75) (0, -)	(78.18 $\pm$ 15.75) (0.0769, 1)
	0.1	(78.18 $\pm$ 15.75) (0.0769, 1)	(78.18 $\pm$ 15.75) (0.0769, 1)	(74.55 $\pm$ 15.84) (0, 0)	(76.36 $\pm$ 15.75) (0, -)	(78.18 $\pm$ 15.75) (0.0769, 1)
spect	0	(74.68 $\pm$ 13.93) (0.5625, 0.75)	(58.23 $\pm$ 16.48) (0.4375, 0.4828)	(64.56 $\pm$ 15.49) (0.2188, 0.7)	(56.96 $\pm$ 14.89) (0.5938, 0.475)	(72.15 $\pm$ 13.85) (0.625, 0.6667)
	0.05	(73.42 $\pm$ 16.26) (0.5938, 0.7037)	(65.82 $\pm$ 15.18) (0.4688, 0.6)	(63.29 $\pm$ 14.82) (0.2813, 0.6)	(58.23 $\pm$ 9.89) (0.2813, 0.4737)	(77.22 $\pm$ 14.67) (0.625, 0.7692)
	0.1	(74.68 $\pm$ 16.06) (0.625, 0.7143)	(64.56 $\pm$ 16.61) (0.4375, 0.5833)	(62.03 $\pm$ 12.31) (0.5, 0.5333)	(58.23 $\pm$ 12.68) (0.2813, 0.4737)	(70.89 $\pm$ 15.2) (0.5938, 0.6552)
statlog-australian-credit	0	(65.7 $\pm$ 6.36) (0.9927, 0.6602)	(65.7 $\pm$ 5.72) (0.9927, 0.6602)	(66.18 $\pm$ 5.92) (1, 0.6618)	(65.7 $\pm$ 5.72) (0.9927, 0.6602)	(65.7 $\pm$ 5.72) (0.9927, 0.6602)
	0.05	(66.18 $\pm$ 5.98) (1, 0.6618)	(66.18 $\pm$ 5.98) (1, 0.6618)	(65.22 $\pm$ 5.84) (0.9854, 0.6585)	(66.18 $\pm$ 5.98) (1, 0.6618)	(66.18 $\pm$ 5.72) (1, 0.6618)
	0.1	(66.18 $\pm$ 7.16) (0.9562, 0.6718)	(66.18 $\pm$ 5.98) (1, 0.6618)	(64.25 $\pm$ 5.94) (0.9635, 0.6567)	(66.18 $\pm$ 5.98) (1, 0.6618)	(66.18 $\pm$ 5.98) (1, 0.6618)
average accuracy	0	79.76	79.88	79.79	76.91	80.93
	0.05	79.15	80.18	77.63	77.6	80.74
	0.1	78.9	80.33	77.75	77.23	80.68
average rank	0	3.08	2.74	2.76	3.87	2.55
	0.05	3.13	2.5	3.63	3.42	2.32
	0.1	3.21	2.26	3.66	3.68	2.18

$(f \times s)$ of each UCI dataset, where f and s denote the number of features and number of samples of the datasets, respectively. Figure 3.9 represents the comparison of accuracies using bar graph of different models under study with different noise levels ($r = 0, 0.05, 0.1$) over 19 UCI datasets. Figure 3.10 depicts the comparison of training time of TWSVM, UTSVM, Pin-GTSVM and proposed Tpin-UTWSVM for all the noise levels and datasets under consideration. Comparison with pin-SVM is shown in the Table 3.10 only, for the ease of representation it is not included in the Figure 3.10.

3.2.3.4 Sensitivity of hyperparameters

The hyperparameters C_1 , C_u and σ effect the generalization performance of the proposed Tpin-UTWSVM model which is depicted in the Figure 3.11. It clearly represents the evaluation of sensitivity of the proposed Tpin-UTWSVM model with user specified hyperparameters having Gaussian kernel on UCI data. Figure 3.11a and 3.11b show the effect of C_1 and C_u on the accuracy of blood, and heart-hungarian data, respectively. Figure 3.11a shows the accuracy decreases with increased value of C_u for blood data and for the heart-hungarian data (Figure 3.11b) middle range of values of C_u have good performance. Figure 3.11c and 3.11d depict the effect of varying the pair (σ, C_u) and (C_1, σ) on the performance of breast-cancer and breast-cancer-wisc data, respectively. Figure 3.11c represents increase in accuracy of the model at the higher and lower range values of C_u . Figure 3.11d shows increase in accuracy of the proposed Tpin-UTWSVM model for breast-cancer-wisc data at the middle range values of C_1 which remains nearly constant at higher values of C_1 .

3.2.3.5 Statistical analysis on the binary datasets

We conduct statistical test to show the difference among pin-SVM [17], TWSVM [22], UTSVM [26], Pin-GTSVM [101] and the proposed Tpin-UTWSVM models. The statistical

Table 3.10: Training time and optimal hyperparameters of models on binary datasets with Gaussian kernel.

Dataset size ($f \times s$)	r	pin-SVM [17] (Time(s), τ) (C_1, σ)	TWSVM [22] Time(s) ($C_1 = C_2, \sigma$)	UTSVM [26] (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Pin-GTSVM [101] Time(s) ($C_1 = C_2, \sigma$)	Tpin-UTWSVM (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
blood (748 $\times$ 4)	0	(5.9297, 0.2) (10, 1)	0.0426 (0.1, 4)	(0.0989, 1) (0.01, 16, 0.1)	0.0742 (10^{-5} , 0.5)	(0.1736, 10^{-5}) (0.1, 4, 0.1)
	0.05	(5.59, 0.5) (100, 1)	0.0457 (0.1, 2)	(0.0479, 0.01) (0.1, 16, 0.1)	0.0874 (10^{-5} , 0.03125)	(0.3374, 1) (0.1, 2, 0.5)
	0.1	(5.2235, 1) (1000, 1)	0.0474 (0.1, 4)	(0.0318, 0.001) (0.01, 0.5, 0.6)	0.1307 (10^{-5} , 0.03125)	(0.2518, 0.1) (0.1, 4, 0.6)
	0	(0.3723, 0.1) (1, 0.25)	0.021 (1, 8)	(0.005, 10^{-5}) (10^{-5} , 2, 0.1)	0.0052 (10^{-5} , 0.03125)	(0.0674, 10^{-5}) (1, 8, 0.1)
	0.05	(0.3888, 0.1) (1, 0.25)	0.0097 (1, 8)	(0.0043, 0.001) (0.01, 4, 0.1)	0.0068 (0.01, 32)	(0.0663, 10^{-5}) (1, 8, 0.1)
	0.1	(0.2869, 0.1) (1, 0.0625)	0.0104 (1, 8)	(0.0055, 0.001) (0.1, 2, 0.5)	0.0072 (0.01, 32)	(0.1203, 0.1) (1, 8, 0.6)
breast-cancer (286 $\times$ 9)	0	(6.3268, 0.1) (0.01, 1)	0.0223 (10^{-5} , 8)	(0.0345, 10^{-5}) (10^{-5} , 4, 0.1)	0.0697 (0.1, 16)	(0.0619, 10^{-5}) (10^{-5} , 8, 0.1)
	0.05	(5.673, 0.1) (0.01, 1)	0.026 (10^{-4} , 4)	(0.0336, 0.001) (0.001, 8, 0.1)	0.0672 (0.1, 16)	(0.2601, 10^{-5}) (10, 16, 0.1)
	0.1	(2.3398, 0.1) (10, 1)	0.0243 (0.01, 8)	(0.0436, 0.01) (0.1, 4, 0.5)	0.0648 (10^{-5} , 32)	(0.2994, 0.1) (10, 16, 0.1)
	0	(0.1086, 0.1) (1, 0.5)	0.0023 (0.001, 2)	(0.0023, 10^{-5}) (10^{-5} , 4, 0.1)	0.0016 (10^{-5} , 2)	(0.0394, 10) (1, 32, 0.3)
	0.05	(0.1479, 0.1) (1, 0.5)	0.0024 (10^{-4} , 2)	(0.0021, 10^{-4}) (10^{-5} , 4, 0.1)	0.0021 (10^{-5} , 2)	(0.0369, 10) (1, 32, 0.3)
	0.1	(0.1237, 0.1) (1, 0.5)	0.0026 (10^{-4} , 4)	(0.0023, 10^{-4}) (10^{-5} , 4, 0.1)	0.002 (10^{-5} , 2)	(0.0063, 10^{-5}) (10^{-4} , 2, 0.5)
breast-cancer-wisc (699 $\times$ 9)	0	(0.1233, 0.1) (1, 0.25)	0.0031 (10^{-5} , 8)	(0.003, 0.01) (0.001, 4, 0.1)	0.0025 (10^{-5} , 4)	(0.0084, 10^{-4}) (10^{-4} , 8, 0.5)
	0.05	(0.1226, 0.1) (10, 0.25)	0.0031 (10^{-4} , 4)	(0.0039, 0.01) (0.001, 4, 0.1)	0.0027 (10^{-5} , 4)	(0.0068, 10^{-4}) (10^{-4} , 8, 0.6)
	0.1	(0.1338, 0.1) (1, 0.25)	0.0029 (10^{-4} , 4)	(0.004, 0.01) (10^{-4} , 4, 0.1)	0.0027 (10^{-4} , 4)	(0.0083, 10^{-5}) (10^{-4} , 8, 0.5)
	0	(0.0507, 0.1) (1, 0.5)	0.0016 (0.001, 4)	(0.0015, 0.01) (0.1, 16, 0.1)	0.0037 (0.1, 8)	(0.0078, 10^{-5}) (1, 32, 0.1)
	0.05	(0.0458, 0.1) (1, 0.5)	0.0023 (0.1, 8)	(0.0042, 0.01) (1, 8, 0.1)	0.0041 (10^{-5} , 8)	(0.0173, 10^{-5}) (1, 32, 0.1)
	0.1	(0.0305, 0.1) (10, 0.25)	0.0017 (0.1, 8)	(0.0039, 0.01) (1, 8, 0.1)	0.0036 (10^{-5} , 8)	(0.0138, 0.01) (1, 8, 0.1)
echocardiogram (131 $\times$ 10)	0	(0.5338, 0.1) (10, 0.125)	0.0041 (10^{-5} , 32)	(0.0205, 1) (0.01, 16, 0.1)	0.0092 (10^{-5} , 0.03125)	(0.0425, 10) (1, 16, 0.5)
	0.05	(0.3771, 0.1) (100, 0.125)	0.0054 (10^{-5} , 32)	(0.014, 1) (0.1, 16, 0.5)	0.009 (10^{-5} , 0.03125)	(0.044, 10) (1, 16, 0.5)
	0.1	(0.338, 0.1) (10, 0.125)	0.0042 (10^{-5} , 32)	(0.0105, 1) (0.1, 16, 0.1)	0.011 (10^{-5} , 0.03125)	(0.036, 100) (1, 16, 0.5)
	0	(0.3926, 0.1) (1, 0.125)	0.005 (10^{-5} , 2)	(0.0321, 100) (0.01, 32, 0.3)	0.016 (100, 16)	(0.032, 10^{-5}) (0.1, 8, 0.1)
	0.05	(0.2904, 0.1) (10, 0.125)	0.0049 (0.001, 2)	(0.0154, 1) (10^{-5} , 32, 0.1)	0.0149 (10, 8)	(0.0476, 10^{-5}) (10, 16, 0.1)
	0.1	(0.3048, 0.1) (1, 0.25)	0.0046 (10^{-4} , 2)	(0.03, 10) (10^{-5} , 32, 0.1)	0.0151 (10, 8)	(0.0412, 10^{-5}) (10, 16, 0.1)
haberman-survival (306 $\times$ 3)	0	(0.0615, 0.1) (10, 2)	0.0016 (0.1, 32)	(0.0095, 100) (10, 8, 0.3)	0.0027 (100, 32)	(0.0056, 10^{-5}) (0.1, 32, 0.1)
	0.05	(0.0431, 0.1) (1000, 2)	0.0014 (0.1, 32)	(0.0107, 100) (10, 8, 0.5)	0.0012 (10^{-5} , 0.03125)	(0.0047, 1) (0.1, 32, 0.3)
	0.1	(0.058, 0.1) (1, 1)	0.0015 (0.1, 32)	(0.002, 0.01) (0.01, 16, 0.5)	0.001 (10^{-5} , 0.03125)	(0.0108, 10^{-5}) (0.1, 32, 0.1)
	0	(0.0603, 0.1) (1, 0.25)	0.0034 (1, 32)	(0.0017, 0.01) (10^{-4} , 2, 0.1)	0.0068 (10, 16)	(0.0288, 0.1) (0.1, 16, 0.6)
	0.05	(0.0522, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0041, 0.1) (1, 32, 0.1)	0.0067 (10, 16)	(0.0095, 0.1) (0.1, 16, 0.5)
	0.1	(0.0634, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0019, 0.01) (10^{-4} , 2, 0.1)	0.0048 (0.01, 16)	(0.0306, 1) (1, 32, 0.5)
heart-hungarian (294 $\times$ 12)	0	(0.0615, 0.1) (10, 2)	0.0016 (0.1, 32)	(0.0095, 100) (10, 8, 0.3)	0.0027 (100, 32)	(0.0056, 10^{-5}) (0.1, 32, 0.1)
	0.05	(0.0431, 0.1) (1000, 2)	0.0014 (0.1, 32)	(0.0107, 100) (10, 8, 0.5)	0.0012 (10^{-5} , 0.03125)	(0.0047, 1) (0.1, 32, 0.3)
	0.1	(0.058, 0.1) (1, 1)	0.0015 (0.1, 32)	(0.002, 0.01) (0.01, 16, 0.5)	0.001 (10^{-5} , 0.03125)	(0.0108, 10^{-5}) (0.1, 32, 0.1)
	0	(0.0603, 0.1) (1, 0.25)	0.0034 (1, 32)	(0.0017, 0.01) (10^{-4} , 2, 0.1)	0.0068 (10, 16)	(0.0288, 0.1) (0.1, 16, 0.6)
	0.05	(0.0522, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0041, 0.1) (1, 32, 0.1)	0.0067 (10, 16)	(0.0095, 0.1) (0.1, 16, 0.5)
	0.1	(0.0634, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0019, 0.01) (10^{-4} , 2, 0.1)	0.0048 (0.01, 16)	(0.0306, 1) (1, 32, 0.5)
heart-switzerland (123 $\times$ 12)	0	(0.0615, 0.1) (10, 2)	0.0016 (0.1, 32)	(0.0095, 100) (10, 8, 0.3)	0.0027 (100, 32)	(0.0056, 10^{-5}) (0.1, 32, 0.1)
	0.05	(0.0431, 0.1) (1000, 2)	0.0014 (0.1, 32)	(0.0107, 100) (10, 8, 0.5)	0.0012 (10^{-5} , 0.03125)	(0.0047, 1) (0.1, 32, 0.3)
	0.1	(0.058, 0.1) (1, 1)	0.0015 (0.1, 32)	(0.002, 0.01) (0.01, 16, 0.5)	0.001 (10^{-5} , 0.03125)	(0.0108, 10^{-5}) (0.1, 32, 0.1)
	0	(0.0603, 0.1) (1, 0.25)	0.0034 (1, 32)	(0.0017, 0.01) (10^{-4} , 2, 0.1)	0.0068 (10, 16)	(0.0288, 0.1) (0.1, 16, 0.6)
	0.05	(0.0522, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0041, 0.1) (1, 32, 0.1)	0.0067 (10, 16)	(0.0095, 0.1) (0.1, 16, 0.5)
	0.1	(0.0634, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0019, 0.01) (10^{-4} , 2, 0.1)	0.0048 (0.01, 16)	(0.0306, 1) (1, 32, 0.5)
hepatitis (155 $\times$ 19)	0	(0.0603, 0.1) (1, 0.25)	0.0034 (1, 32)	(0.0017, 0.01) (10^{-4} , 2, 0.1)	0.0068 (10, 16)	(0.0288, 0.1) (0.1, 16, 0.6)
	0.05	(0.0522, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0041, 0.1) (1, 32, 0.1)	0.0067 (10, 16)	(0.0095, 0.1) (0.1, 16, 0.5)
	0.1	(0.0634, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0019, 0.01) (10^{-4} , 2, 0.1)	0.0048 (0.01, 16)	(0.0306, 1) (1, 32, 0.5)
	0	(0.0603, 0.1) (1, 0.25)	0.0034 (1, 32)	(0.0017, 0.01) (10^{-4} , 2, 0.1)	0.0068 (10, 16)	(0.0288, 0.1) (0.1, 16, 0.6)
	0.05	(0.0522, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0041, 0.1) (1, 32, 0.1)	0.0067 (10, 16)	(0.0095, 0.1) (0.1, 16, 0.5)
	0.1	(0.0634, 0.1) (1, 0.25)	0.0035 (1, 32)	(0.0019, 0.01) (10^{-4} , 2, 0.1)	0.0048 (0.01, 16)	(0.0306, 1) (1, 32, 0.5)

Table 3.10: Continued from previous page-Training time and optimal hyperparameters of models on binary datasets with Gaussian kernel.

Dataset size ($f \times s$)	r	pin-SVM [17] (Time(s), τ) (C_1, σ)	TWSVM [22] Time(s) ($C_1 = C_2, \sigma$)	UTSVM [26] (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Pin-GTSVM [101] Time(s) ($C_1 = C_2, \sigma$)	Tpin-UTWSVM (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
ionosphere (351 $\times$ 33)	0	(0.6903, 0.1)	0.0056	(0.0075, 0.1)	0.0099	(0.0158, 10^{-4})
		(10, 1)	(10^{-5} , 2)	(10^{-5} , 32, 0.1)	(10^{-5} , 2)	(10^{-5} , 2, 0.3)
	0.05	(1.0821, 0.1)	0.008	(0.0069, 10^{-5})	0.0052	(0.0135, 10^{-4})
		(1, 1)	(10^{-5} , 2)	(10^{-4} , 2, 0.1)	(10^{-5} , 2)	(10^{-5} , 2, 0.3)
	0.1	(0.4487, 0.2)	0.007	(0.006, 10^{-5})	0.0036	(0.0097, 10^{-5})
		(1, 1)	(10^{-5} , 2)	(10^{-4} , 2, 0.1)	(10^{-5} , 2)	(0.001, 4, 0.1)
molec-biol-promoter (106 $\times$ 57)	0	(0.0315, 0.1)	0.0009	(0.0013, 0.01)	0.0008	(0.0044, 10^{-4})
		(1, 0.125)	(10^{-4} , 4)	(0.001, 32, 0.1)	(10^{-5} , 4)	(0.001, 8, 0.5)
	0.05	(0.0326, 0.1)	0.0011	(0.0017, 1)	0.0009	(0.0037, 10^{-4})
		(1, 0.125)	(10^{-4} , 4)	(10^{-5} , 32, 0.1)	(10^{-5} , 4)	(0.001, 8, 0.6)
	0.1	(0.024, 0.1)	0.0015	(0.0012, 0.01)	0.0009	(0.006, 10^{-4})
		(10, 0.25)	(10^{-4} , 4)	(0.001, 32, 0.1)	(10^{-5} , 4)	(10^{-4} , 4, 0.6)
monks-1 (124 $\times$ 6)	0	(1.7698, 0.2)	0.0106	(0.0138, 0.001)	0.0138	(0.0312, 0.001)
		(10, 1)	(10^{-4} , 2)	(0.001, 2, 0.1)	(10^{-5} , 2)	(10^{-4} , 2, 0.5)
	0.05	(2.1398, 0.1)	0.0139	(0.0133, 0.001)	0.0155	(0.048, 10^{-4})
		(10, 1)	(0.01, 2)	(0.1, 2, 0.1)	(10^{-5} , 2)	(0.001, 2, 0.5)
	0.1	(2.3103, 0.1)	0.0156	(0.0149, 0.01)	0.0146	(0.0422, 0.001)
		(10, 1)	(0.1, 2)	(0.001, 2, 0.3)	(10^{-5} , 2)	(0.001, 2, 0.1)
parkinsons (195 $\times$ 22)	0	(0.1928, 0.1)	0.0026	(0.0021, 0.001)	0.0019	(0.0088, 10^{-5})
		(10, 1)	(0.001, 2)	(10^{-5} , 2, 0.1)	(10^{-5} , 2)	(10^{-4} , 2, 0.1)
	0.05	(0.1051, 0.1)	0.0025	(0.0027, 0.001)	0.0022	(0.0063, 0.001)
		(10, 1)	(0.001, 2)	(10^{-5} , 2, 0.1)	(0.001, 2)	(0.01, 2, 0.1)
	0.1	(0.2886, 0.1)	0.0027	(0.0025, 0.001)	0.0022	(0.0159, 10^{-4})
		(1, 1)	(0.001, 2)	(10^{-5} , 2, 0.1)	(0.001, 2)	(0.001, 2, 0.1)
pima (768 $\times$ 8)	0	(3.7004, 0.1)	0.0718	(0.2115, 10)	0.091	(0.5041, 10)
		(1, 0.5)	(1, 32)	(10^{-4} , 32, 0.1)	(0.1, 4)	(1, 32, 0.1)
	0.05	(3.8614, 0.1)	0.067	(0.2134, 10)	0.0798	(0.4701, 0.01)
		(10, 0.5)	(1, 32)	(10^{-5} , 32, 0.1)	(0.01, 4)	(1, 32, 0.3)
	0.1	(2.5047, 0.1)	0.0696	(0.1113, 1)	0.0867	(0.1279, 1)
		(10, 0.5)	(1, 32)	(0.01, 32, 0.3)	(0.01, 4)	(0.1, 32, 0.6)
pittsburg-bridges-T-OR-D (102 $\times$ 4)	0	(0.026, 0.5)	0.0013	(0.0008, 0.001)	0.0016	(0.012, 10)
		(1, 0.5)	(0.01, 4)	(0.01, 2, 0.1)	(10^{-5} , 4)	(1, 16, 0.3)
	0.05	(0.0361, 0.1)	0.0016	(0.0045, 1)	0.0015	(0.0254, 10)
		(1, 0.25)	(0.01, 2)	(1, 32, 0.1)	(10^{-5} , 4)	(1, 16, 0.3)
	0.1	(0.0275, 0.1)	0.0012	(0.0036, 1)	0.0011	(0.012, 10)
		(1, 0.25)	(0.001, 2)	(1, 32, 0.3)	(10^{-5} , 4)	(1, 16, 0.6)
planning (182 $\times$ 12)	0	(0.1833, 0.1)	0.0022	(0.0022, 10^{-4})	0.0019	(0.004, 10^{-5})
		(1, 2)	(10^{-5} , 0.03125)	(10^{-5} , 0.03125, 0.1)	(10^{-5} , 0.03125)	(10^{-5} , 0.03125, 0.1)
	0.05	(0.179, 0.1)	0.0022	(0.0024, 10^{-5})	0.005	(0.0068, 10^{-5})
		(1, 2)	(10^{-5} , 0.25)	(0.01, 16, 0.1)	(10^{-5} , 0.125)	(10^{-5} , 0.25, 0.1)
	0.1	(0.1245, 0.1)	0.0017	(0.0025, 10^{-5})	0.0018	(0.0079, 10^{-5})
		(1, 2)	(10^{-5} , 0.5)	(10^{-4} , 2, 0.1)	(10^{-5} , 0.25)	(10^{-5} , 0.5, 0.1)
spect (80 $\times$ 22)	0	(0.4291, 0.1)	0.0043	(0.007, 0.1)	0.0133	(0.0819, 0.1)
		(1, 0.125)	(0.01, 4)	(1, 32, 0.6)	(10, 8)	(1, 8, 0.5)
	0.05	(0.2821, 0.1)	0.0038	(0.0049, 0.001)	0.0045	(0.0618, 0.1)
		(10, 0.25)	(10^{-5} , 32)	(0.01, 4, 0.6)	(10^{-5} , 1)	(1, 16, 0.3)
	0.1	(0.2327, 0.1)	0.0048	(0.0136, 1)	0.0031	(0.0788, 1)
		(1, 0.25)	(0.1, 32)	(0.1, 32, 0.1)	(10^{-5} , 1)	(1, 8, 0.1)
statlog-australian-credit (690 $\times$ 14)	0	(4.0553, 0.2)	0.1338	(0.0264, 10^{-5})	0.1413	(0.2216, 10^{-5})
		(1, 8)	(10^{-4} , 0.125)	(0.001, 16, 0.1)	(10^{-5} , 0.125)	(10^{-4} , 0.125, 0.1)
	0.05	(1.4346, 0.1)	0.0206	(0.1634, 0.01)	0.0274	(0.2537, 10^{-4})
		(10^{-5} , 0.03125)	(10^{-5} , 0.03125)	(10, 8, 0.3)	(10^{-5} , 0.03125)	(10^{-4} , 0.125, 0.6)
	0.1	(3.526, 0.1)	0.0261	(0.1577, 1)	0.0218	(0.0437, 10^{-5})
		(0.1, 0.5)	(10^{-5} , 0.03125)	(100, 16, 0.6)	(10^{-5} , 0.03125)	(10^{-5} , 0.03125, 0.1)
average time	0	1.3178	0.0179	0.0253	0.0246	0.028
	0.05	1.1518	0.0119	0.0291	0.0181	0.0328
	0.1	0.9679	0.0123	0.0236	0.0199	0.0177

Table 3.11: Pairwise win-tie-loss of the proposed Tpin-UTWSVM model with respect to baseline models over UCI datasets. With noise level r in the column, comparison of row method with the proposed Tpin-UTWSVM. Here,

[a,b,c] indicates that the proposed Tpin-UTWSVM technique wins ‘a’ number of times, ties ‘b’ number of times, and loses ‘c’ number of times in comparison to the row model.

Model	Tpin-UTWSVM			
	$r = 0$	$r = 0.05$	$r = 0.1$	Total
pin-SVM [17]	[9, 5, 5]	[12, 2, 5]	[12, 2, 5]	[33, 9, 15]
TWSVM [22]	[8, 6, 5]	[7, 6, 6]	[4, 12, 3]	[19, 24, 14]
UTSVM [26]	[8, 4, 7]	[15, 0, 4]	[14, 2, 3]	[37, 6, 14]
Pin-GTSVM [101]	[13, 3, 3]	[9, 7, 3]	[13, 5, 1]	[35, 15, 7]

test conducted is Friedman test [123].

In the Friedman test, we have the null hypothesis assumption that each model has the same average rank. Friedman statistic has χ^2 distribution and it is calculated as: Here, we have $N = 19$ and $md = 5$. On simple calculation, we get $\chi_F^2 = 8.2916$ and $F_F = 2.2043$. F_F has degree of freedom $(4, 72)$. The critical value $F_{(4,72)} = 2.4989$ with 5% level of significance. As the obtained F_F is greater than $F_{(4,72)}$, the null hypothesis fails, and the considered models are statistically different.

Also, Table 3.11 gives the detail about number of wins, ties and loses of the proposed Tpin-UTWSVM in comparison to baseline models at different noise levels ($r = 0, 0.05$ and 0.1). The table clearly shows the proposed model has more number of wins for all the three cases.

3.2.3.6 Confusion matrix of proposed Tpin-UTWSVM on UCI datasets

The confusion matrices of the proposed Tpin-UTWSVM with the optimal hyperparameters over 19 UCI datasets having $r = 0$ are shown here from Table 3.12 to Table 3.30. It is

defined as

confusion matrix =

		Actual value	
		positive label	negative label
Predicted value	positive label	tp	fp
	negative label	fn	tn

where tp , tn and fp , fn signify the correctly classified positive, negative samples and misclassified positive, negative samples, respectively.

18	9
43	154

Table 3.12: blood

10	7
13	56

Table 3.13: breast-cancer

74	4
3	129

Table 3.14: breast-cancer-wisc

4	7
9	39

Table 3.15: breast-cancer-wisc-prog

25	3
8	26

Table 3.16: conn-bench-sonar-mines

1	2
26	63

Table 3.17: haberman-survival

25	6
6	51

Table 3.18: heart-hungarian

4	3
13	17

Table 3.19: heart-switzerland

37	2
3	4

Table 3.20: hepatitis

68	1
3	33

Table 3.21: ionosphere

9	6
4	13

Table 3.22: molec-biol-promoter

87	0
0	80

Table 3.23: monks-1

45	2
0	11

Table 3.24: parkinsons

45	7
48	130

Table 3.25: pima

1	0
12	42

Table 3.26: planning

20	10
12	37

Table 3.27: spect

7	0
4	28

Table 3.28: echocardiogram

136	70
1	0

Table 3.29: statlog-australian-credit

0	1
3	27

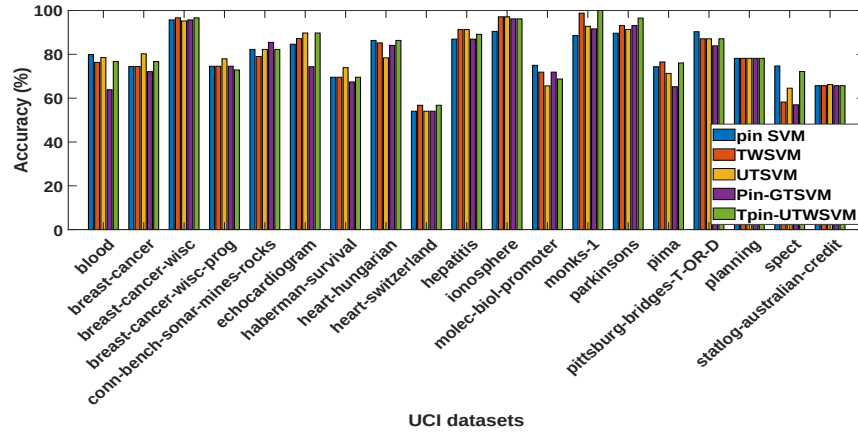
Table 3.30: pittsburg-bridges-T-OR-D

3.2.4 Applications

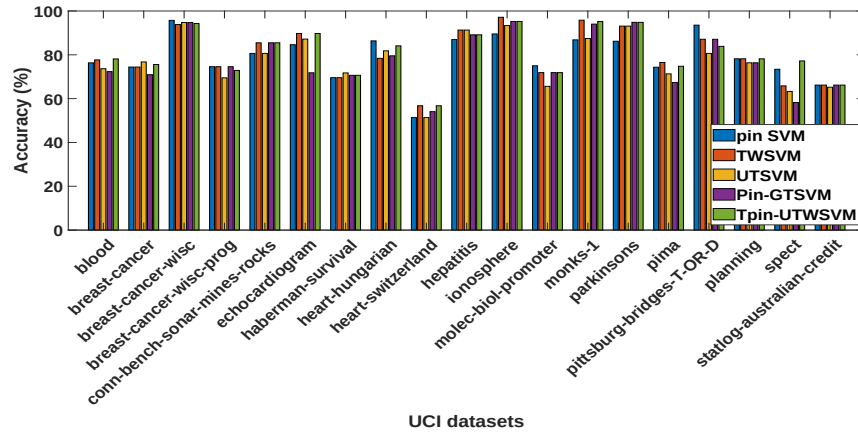
In this section, we demonstrate the applications of the proposed Tpin-UTWSVM model on EEG signal classification and Alzheimer disease detection.

3.2.4.1 EEG dataset and preprocessing

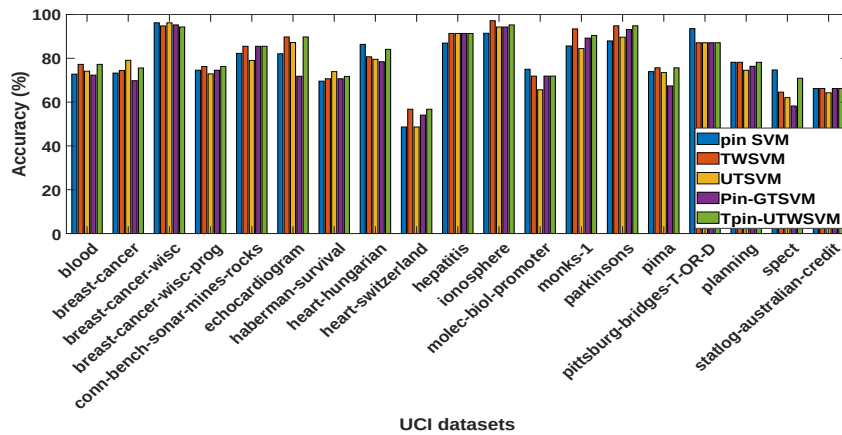
The proposed Tpin-UTWSVM model is used for the classification of the EEG signals. The EEG dataset to carry out the numerical experiments is taken from the publicly available data [137]. The dataset is having 5 collections namely S, N, F, Z, O, where the signals in each of the collection were sampled at 173.61 Hz for 23.6 seconds having hundred single channel signals. The five collections represent the five states: S signifies ictal (seizure) state, N and F denote interictal state (seizure-free), Z and O correspond to healthy (control) subjects with open and close eyes, respectively. The processing of all the EEG signals was done by 128 channel amplifier system. The collections S, F and N has intra-cranial recording mode. The rankfeatures() function and its various criterion's of MATLAB is used to assess the significance of each features. Different criterions are T-Test [141], Entropy [142], Bhattacharyya [141], ROC [141] and Wilcoxon [143]. Figure 3.12 depicts samples of EEG signals for different states.



(a) Bar graph to compare the accuracies of models on UCI datasets with $r = 0$.

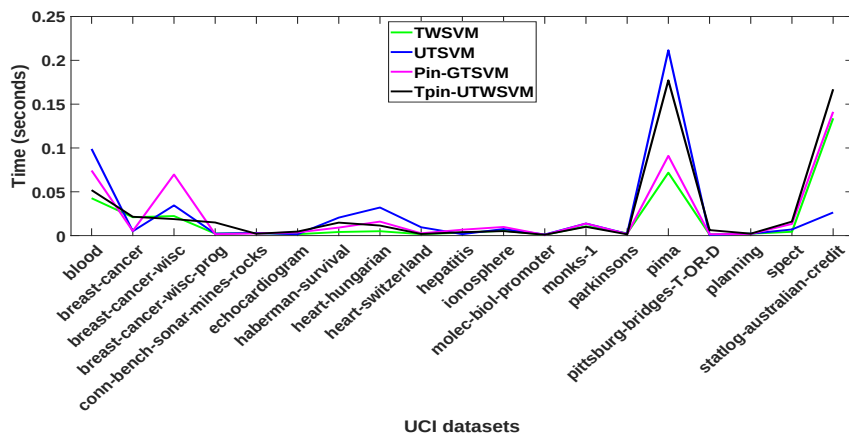


(b) Bar graph to compare the accuracies of models on UCI datasets with $r = 0.05$.

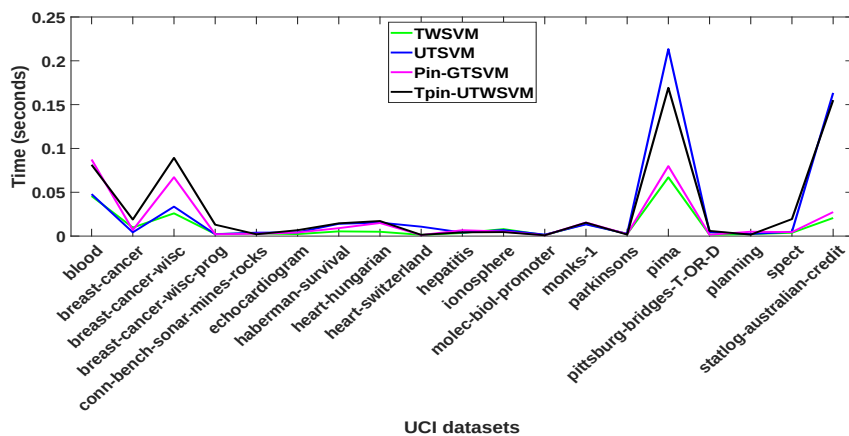


(c) Bar graph to compare the accuracies of models on UCI datasets with $r = 0.1$.

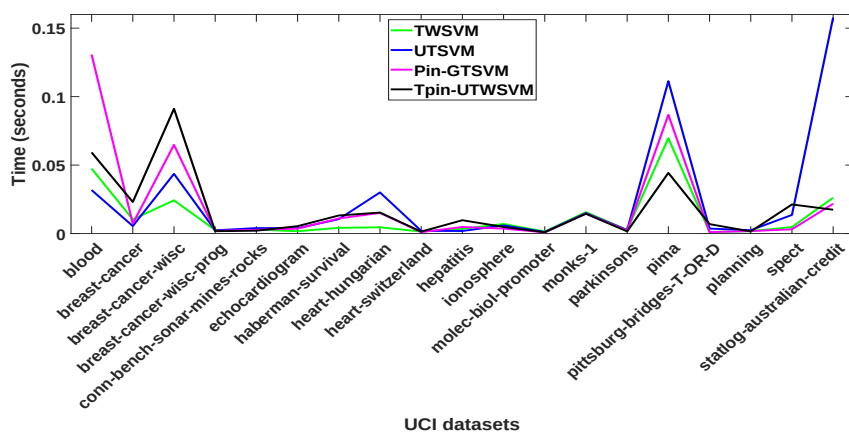
Figure 3.9: Bar graph representation to compare the performance of models over UCI datasets with different noise levels.



(a) Comparison of training time of models on UCI datasets with $r = 0$.



(b) Comparison of training time of models on UCI datasets with $r = 0.05$.



(c) Comparison of training time of models on UCI datasets with $r = 0.1$.

Figure 3.10: Bar graph representation to compare the training of models over UCI datasets with different noise levels.

Table 3.31: Performance of proposed Tpin-UTWSVM model with baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17] (Accuracy $\pm$ sd) (Recall,Precision)	TWSVM [22] (Accuracy $\pm$ sd) (Recall,Precision)	UTWSVM [26] (Accuracy $\pm$ sd) (Recall,Precision)	Pin-GTSVM [101] (Accuracy $\pm$ sd) (Recall,Precision)	Tpin-UTWSVM (Accuracy $\pm$ sd) (Recall,Precision)
F_vs_S	Bhattacharyya	0	(80 $\pm$ 11.761) (0.5862, 1)	(98.33 $\pm$ 4.8211) (0.9655, 1)	(98.33 $\pm$ 4.8211) (0.9655, 1)	(95 $\pm$ 3.0117) (0.9655, 0.9333)	(96.67 $\pm$ 2.2588) (0.9655, 0.9655)
		0.05	(80 $\pm$ 11.6885) (0.5862, 1)	(95 $\pm$ 4.9943) (1, 0.9063)	(98.33 $\pm$ 4.8211) (0.9655, 1)	(96.67 $\pm$ 3.4503) (0.9655, 0.9655)	(96.67 $\pm$ 2.2588) (0.9655, 0.9655)
		0.1	(78.33 $\pm$ 12.7997) (0.5517, 1)	(95 $\pm$ 4.9943) (1, 0.9063)	(95 $\pm$ 3.0117) (1, 0.9063)	(96.67 $\pm$ 3.4503) (0.9655, 0.9655)	(98.33 $\pm$ 3.0117) (1, 0.9667)
	Entropy	0	(80 $\pm$ 11.761) (0.5862, 1)	(98.33 $\pm$ 4.8211) (0.9655, 1)	(98.33 $\pm$ 4.8211) (0.9655, 1)	(95 $\pm$ 3.0117) (0.9655, 0.9333)	(96.67 $\pm$ 2.2588) (0.9655, 0.9655)
		0.05	(80 $\pm$ 11.6885) (0.5862, 1)	(95 $\pm$ 4.9943) (1, 0.9063)	(98.33 $\pm$ 4.8211) (0.9655, 1)	(96.67 $\pm$ 3.4503) (0.9655, 0.9655)	(98.33 $\pm$ 2.2588) (0.9655, 1)
		0.1	(78.33 $\pm$ 12.7997) (0.5517, 1)	(95 $\pm$ 4.9943) (1, 0.9063)	(95 $\pm$ 3.0117) (1, 0.9063)	(96.67 $\pm$ 3.4503) (0.9655, 0.9655)	(98.33 $\pm$ 3.0117) (1, 0.9667)
F_vs_S	ROC	0	(78.33 $\pm$ 11.8809) (0.5517, 1)	(90 $\pm$ 5.8805) (0.9655, 0.8485)	(91.67 $\pm$ 7.1031) (0.8276, 1)	(91.67 $\pm$ 5.8805) (1, 0.8529)	(88.33 $\pm$ 5.8805) (0.9655, 0.8235)
		0.05	(75 $\pm$ 12.9538) (0.4828, 1)	(91.67 $\pm$ 4.9943) (0.9655, 0.875)	(88.33 $\pm$ 7.1031) (0.7586, 1)	(93.33 $\pm$ 6.5638) (1, 0.8788)	(91.67 $\pm$ 4.9943) (1, 0.8529)
		0.1	(76.67 $\pm$ 19.4744) (0.5172, 1)	(91.67 $\pm$ 4.8211) (1, 0.8529)	(86.67 $\pm$ 5.6344) (0.931, 0.8182)	(91.67 $\pm$ 7.3771) (0.8966, 0.9286)	(91.67 $\pm$ 4.8211) (1, 0.8529)
	T-Test	0	(80 $\pm$ 9.9887) (0.5862, 1)	(91.67 $\pm$ 4.9943) (1, 0.8529)	(91.67 $\pm$ 5.8805) (1, 0.8529)	(93.33 $\pm$ 7.3771) (0.931, 0.931)	(91.67 $\pm$ 4.9943) (1, 0.8529)
		0.05	(80 $\pm$ 11.3938) (0.5862, 1)	(90 $\pm$ 4.9943) (0.931, 0.871)	(90 $\pm$ 4.9943) (1, 0.8286)	(91.67 $\pm$ 5.6344) (1, 0.8529)	(93.33 $\pm$ 4.8211) (0.931, 0.931)
		0.1	(80 $\pm$ 20.7703) (0.5862, 1)	(91.67 $\pm$ 5.6344) (0.931, 0.9)	(90 $\pm$ 3.7646) (1, 0.8286)	(91.67 $\pm$ 5.6344) (1, 0.8529)	(93.33 $\pm$ 5.8805) (0.931, 0.931)
F_vs_S	Wilcoxon	0	(76.67 $\pm$ 11.8809) (0.5172, 1)	(90 $\pm$ 7.3771) (0.9655, 0.8485)	(90 $\pm$ 4.0546) (0.9655, 0.8485)	(88.33 $\pm$ 6.5638) (1, 0.8056)	(91.67 $\pm$ 5.8805) (1, 0.8529)
		0.05	(76.67 $\pm$ 11.8809) (0.5172, 1)	(91.67 $\pm$ 5.6344) (0.931, 0.9)	(88.33 $\pm$ 4.8211) (0.9655, 0.8235)	(91.67 $\pm$ 7.5668) (0.8966, 0.9286)	(91.67 $\pm$ 5.8805) (1, 0.8529)
		0.1	(75 $\pm$ 16.2185) (0.4828, 1)	(90 $\pm$ 5.6344) (1, 0.8286)	(91.67 $\pm$ 4.5175) (1, 0.8529)	(90 $\pm$ 5.8805) (1, 0.8286)	(93.33 $\pm$ 5.8805) (1, 0.8788)
	ROC	0	(65 $\pm$ 10.1296) (0.2759, 1)	(95 $\pm$ 2.2588) (0.9655, 0.9333)	(96.67 $\pm$ 3.0117) (0.931, 1)	(96.67 $\pm$ 0) (0.931, 1)	(98.33 $\pm$ 0) (0.9655, 1)
		0.05	(66.67 $\pm$ 13.9036) (0.3103, 1)	(96.67 $\pm$ 2.2588) (0.9655, 0.9655)	(96.67 $\pm$ 3.0117) (0.931, 1)	(98.33 $\pm$ 0) (0.9655, 1)	(95 $\pm$ 0) (0.9655, 0.9333)
		0.1	(61.67 $\pm$ 15.5036) (0.2069, 1)	(96.67 $\pm$ 0) (0.9655, 0.9655)	(95 $\pm$ 3.6886) (0.8966, 1)	(98.33 $\pm$ 0) (0.9655, 1)	(98.33 $\pm$ 0) (0.9655, 1)
N_vs_S	T-Test	0	(63.33 $\pm$ 10.2132) (0.2414, 1)	(96.67 $\pm$ 0) (0.9655, 0.9655)	(96.67 $\pm$ 3.4503) (0.931, 1)	(98.33 $\pm$ 0) (0.9655, 1)	(95 $\pm$ 0) (0.9655, 0.9333)
		0.05	(63.33 $\pm$ 12.0468) (0.2414, 1)	(96.67 $\pm$ 0) (0.9655, 0.9655)	(93.33 $\pm$ 3.0117) (1, 0.8788)	(98.33 $\pm$ 0) (0.9655, 1)	(98.33 $\pm$ 0) (0.9655, 1)
		0.1	(63.33 $\pm$ 17.2352) (0.2414, 1)	(98.33 $\pm$ 0) (0.9655, 1)	(93.33 $\pm$ 0) (0.9655, 0.9032)	(98.33 $\pm$ 0) (0.9655, 1)	(98.33 $\pm$ 0) (0.9655, 1)
	Wilcoxon	0	(66.67 $\pm$ 10.1296) (0.3103, 1)	(95 $\pm$ 3.0117) (0.9655, 0.9333)	(96.67 $\pm$ 3.0117) (0.931, 1)	(98.33 $\pm$ 0) (0.9655, 1)	(98.33 $\pm$ 0) (0.9655, 1)
		0.05	(63.33 $\pm$ 11.0657) (0.2414, 1)	(96.67 $\pm$ 0) (0.9655, 0.9655)	(96.67 $\pm$ 3.0117) (0.931, 1)	(95 $\pm$ 0) (0.8966, 1)	(96.67 $\pm$ 0) (0.9655, 0.9655)
		0.1	(61.67 $\pm$ 15.8114) (0.2069, 1)	(96.67 $\pm$ 0) (0.9655, 0.9655)	(95 $\pm$ 2.2588) (0.9655, 0.9333)	(95 $\pm$ 2.2588) (0.8966, 1)	(96.67 $\pm$ 0) (0.931, 1)
O_vs_F	Bhattacharyya	0	(60 $\pm$ 20.3289) (0.6897, 0.5714)	(61.67 $\pm$ 11.193) (0.8966, 0.5652)	(68.33 $\pm$ 11.2687) (0.6207, 0.6923)	(78.33 $\pm$ 9.6715) (0.8621, 0.7353)	(76.67 $\pm$ 11.3938) (0.931, 0.6923)
		0.05	(60 $\pm$ 20.1469) (0.6897, 0.5714)	(66.67 $\pm$ 10.6745) (0.6207, 0.6667)	(68.33 $\pm$ 6.0234) (0.6207, 0.6923)	(75 $\pm$ 11.2938) (0.8276, 0.7059)	(63.33 $\pm$ 10.6745) (1, 0.5686)
		0.1	(66.67 $\pm$ 19.7633) (0.8276, 0.6154)	(75 $\pm$ 8.5516) (0.7586, 0.7333)	(61.67 $\pm$ 13.5735) (0.5172, 0.625)	(76.67 $\pm$ 11.761) (0.8276, 0.7273)	(60 $\pm$ 12.6213) (1, 0.5472)
	Entropy	0	(60 $\pm$ 20.3289) (0.6897, 0.5714)	(61.67 $\pm$ 11.193) (0.8966, 0.5652)	(68.33 $\pm$ 11.2687) (0.6207, 0.6923)	(78.33 $\pm$ 9.6715) (0.8621, 0.7353)	(76.67 $\pm$ 11.3938) (0.931, 0.6923)
		0.05	(60 $\pm$ 20.3289) (0.6897, 0.5714)	(61.67 $\pm$ 11.193) (0.8966, 0.5652)	(68.33 $\pm$ 8.7805) (0.6207, 0.6923)	(78.33 $\pm$ 11.2938) (0.8621, 0.7353)	(63.33 $\pm$ 10.6745) (1, 0.5686)
		0.1	(66.67 $\pm$ 19.7633) (0.8276, 0.6154)	(71.67 $\pm$ 6.5638) (0.7586, 0.6875)	(63.33 $\pm$ 11.0657) (0.6897, 0.6061)	(78.33 $\pm$ 11.2938) (0.8621, 0.7353)	(78.33 $\pm$ 11.6885) (0.8621, 0.7353)

Table 3.31: Continued from previous page-Performance of proposed Tpin-UTWSVM model with baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17] (Accuracy $\pm$ sd) (Recall,Precision)	TWSVM [22] (Accuracy $\pm$ sd) (Recall,Precision)	UTWSVM [26] (Accuracy $\pm$ sd) (Recall,Precision)	Pin-GTSVM [101] (Accuracy $\pm$ sd) (Recall,Precision)	Tpin-UTWSVM (Accuracy $\pm$ sd) (Recall,Precision)
O_vs_F	ROC	0	(75 $\pm$ 15.8651) (0.8966, 0.6842)	(73.33 $\pm$ 12.5988) (0.9655, 0.6512)	(75 $\pm$ 10.5409) (0.8966, 0.6842)	(83.33 $\pm$ 12.1638) (0.8276, 0.8276)	(73.33 $\pm$ 7.5668) (0.6897, 0.7407)
		0.05	(75 $\pm$ 15.8651) (0.8966, 0.6842)	(81.67 $\pm$ 8.5516) (0.7931, 0.8214)	(76.67 $\pm$ 9.788) (0.7931, 0.7419)	(83.33 $\pm$ 11.2938) (0.8276, 0.8276)	(86.67 $\pm$ 11.193) (0.931, 0.8182)
		0.1	(73.33 $\pm$ 15.8651) (0.8966, 0.6667)	(83.33 $\pm$ 7.5292) (0.7931, 0.8519)	(75 $\pm$ 9.788) (0.7931, 0.7188)	(83.33 $\pm$ 10.2131) (0.8276, 0.8276)	(86.67 $\pm$ 11.193) (0.931, 0.8182)
	T-Test	0	(73.33 $\pm$ 7.8608) (0.6552, 0.76)	(73.33 $\pm$ 7.8608) (0.6552, 0.76)	(78.33 $\pm$ 13.8013) (0.8966, 0.7222)	(66.67 $\pm$ 12.5085) (0.9655, 0.5957)	(75 $\pm$ 11.2939) (0.931, 0.675)
		0.05	(73.33 $\pm$ 7.8608) (0.6552, 0.76)	(73.33 $\pm$ 6.2543) (0.6552, 0.76)	(78.33 $\pm$ 12.6213) (0.8966, 0.7222)	(66.67 $\pm$ 11.8809) (0.9655, 0.5957)	(76.67 $\pm$ 11.2939) (0.9655, 0.6829)
		0.1	(73.33 $\pm$ 7.3771) (0.6552, 0.76)	(66.67 $\pm$ 10.5409) (0.9655, 0.5957)	(80 $\pm$ 12.5085) (0.931, 0.7297)	(83.33 $\pm$ 9.1906) (0.7931, 0.8519)	(85 $\pm$ 9.0664) (0.931, 0.7941)
O_vs_N	Bhattacharyya	0	(76.67 $\pm$ 22.08) (0.5172, 1)	(80 $\pm$ 13.0627) (0.8276, 0.7742)	(80 $\pm$ 13.8013) (0.7586, 0.8148)	(80 $\pm$ 11.6885) (0.8276, 0.7742)	(81.67 $\pm$ 12.2336) (0.8621, 0.7813)
		0.05	(76.67 $\pm$ 22.08) (0.5172, 1)	(80 $\pm$ 13.3843) (0.8276, 0.7742)	(78.33 $\pm$ 13.8013) (0.7586, 0.7857)	(80 $\pm$ 12.1638) (0.8276, 0.7742)	(81.67 $\pm$ 12.2336) (0.8621, 0.7813)
		0.1	(76.67 $\pm$ 21.5604) (0.5172, 1)	(80 $\pm$ 13.4897) (0.8276, 0.7742)	(76.67 $\pm$ 13.8013) (0.7586, 0.7857)	(81.67 $\pm$ 12.0468) (0.8621, 0.7813)	(85 $\pm$ 11.5666) (0.931, 0.7941)
	Entropy	0	(76.67 $\pm$ 21.9384) (0.5172, 1)	(80 $\pm$ 13.1277) (0.8276, 0.7742)	(78.33 $\pm$ 13.8013) (0.7586, 0.7857)	(80 $\pm$ 11.6885) (0.8276, 0.7742)	(81.67 $\pm$ 12.2336) (0.8621, 0.7813)
		0.05	(76.67 $\pm$ 21.5604) (0.5172, 1)	(80 $\pm$ 14.7542) (0.8276, 0.7742)	(78.33 $\pm$ 13.8013) (0.7586, 0.7857)	(80 $\pm$ 11.6885) (0.8276, 0.7742)	(81.67 $\pm$ 10.6745) (0.8621, 0.7813)
		0.1	(76.67 $\pm$ 21.9384) (0.5172, 1)	(80 $\pm$ 12.1405) (0.8276, 0.7742)	(78.33 $\pm$ 12.1639) (0.7586, 0.7857)	(80 $\pm$ 12.0468) (0.8276, 0.7742)	(80 $\pm$ 11.5666) (0.8621, 0.7576)
O_vs_N	ROC	0	(63.33 $\pm$ 19.6483) (0.8276, 0.5854)	(81.67 $\pm$ 10.6479) (0.8966, 0.7647)	(61.67 $\pm$ 16.166) (0.5517, 0.6154)	(80 $\pm$ 12.0468) (0.931, 0.7297)	(80 $\pm$ 12.1405) (0.931, 0.7297)
		0.05	(63.33 $\pm$ 19.6483) (0.8276, 0.5854)	(78.33 $\pm$ 11.5666) (0.8966, 0.7222)	(61.67 $\pm$ 14.7542) (0.5862, 0.6071)	(76.67 $\pm$ 10.6745) (0.8621, 0.7143)	(80 $\pm$ 10.8849) (0.931, 0.7297)
		0.1	(66.67 $\pm$ 19.1071) (0.3448, 0.9091)	(78.33 $\pm$ 11.3938) (0.8966, 0.7222)	(63.33 $\pm$ 14.7542) (0.6207, 0.6207)	(78.33 $\pm$ 10.6479) (0.8966, 0.7222)	(76.67 $\pm$ 10.8849) (0.8621, 0.7143)
	T-Test	0	(71.67 $\pm$ 19.7059) (0.4828, 0.875)	(76.67 $\pm$ 12.5085) (0.8276, 0.7273)	(75 $\pm$ 15.8829) (0.7586, 0.7333)	(76.67 $\pm$ 8.9087) (0.8276, 0.7273)	(75 $\pm$ 7.8608) (0.8621, 0.6944)
		0.05	(66.67 $\pm$ 17.561) (0.3448, 0.9091)	(76.67 $\pm$ 11.0657) (0.8276, 0.7273)	(75 $\pm$ 16.2185) (0.7586, 0.7333)	(76.67 $\pm$ 8.9087) (0.8276, 0.7273)	(73.33 $\pm$ 7.8608) (0.8276, 0.6857)
		0.1	(66.67 $\pm$ 21.0953) (0.5172, 0.7143)	(76.67 $\pm$ 10.1015) (0.8276, 0.7273)	(75 $\pm$ 18.2108) (0.7586, 0.7333)	(76.67 $\pm$ 9.1906) (0.8276, 0.7273)	(80 $\pm$ 9.788) (0.8966, 0.7429)
O_vs_N	Wilcoxon	0	(66.67 $\pm$ 20.3289) (0.4483, 0.7647)	(80 $\pm$ 10.2132) (0.7931, 0.7931)	(71.67 $\pm$ 7.5292) (0.8276, 0.6667)	(80 $\pm$ 6.9007) (0.931, 0.7297)	(83.33 $\pm$ 7.1031) (0.8276, 0.8276)
		0.05	(68.33 $\pm$ 20.3289) (0.4828, 0.7778)	(83.33 $\pm$ 7.6783) (0.7931, 0.8519)	(76.67 $\pm$ 7.8608) (0.8276, 0.7273)	(80 $\pm$ 9.0351) (0.8276, 0.8)	(83.33 $\pm$ 8.1092) (0.7931, 0.8519)
		0.1	(66.67 $\pm$ 20.1469) (0.4828, 0.7368)	(81.67 $\pm$ 9.0351) (0.8276, 0.8)	(76.67 $\pm$ 6.7344) (0.8276, 0.7273)	(81.67 $\pm$ 10.8849) (0.6897, 0.7692)	(83.33 $\pm$ 5.8805) (0.7931, 0.8519)
	Bhattacharyya	0	(86.67 $\pm$ 15.1336) (0.7241, 1)	(95 $\pm$ 3.0117) (1, 0.9063)	(100 $\pm$ 2.2588) (1, 1)	(96.67 $\pm$ 0) (1, 0.9355)	(95 $\pm$ 3.0117) (1, 0.9063)
		0.05	(88.33 $\pm$ 16.2185) (0.7586, 1)	(95 $\pm$ 3.0117) (1, 0.9063)	(100 $\pm$ 3.0117) (1, 1)	(96.67 $\pm$ 0) (1, 0.9355)	(96.67 $\pm$ 2.2588) (1, 0.9355)
		0.1	(51.67 $\pm$ 19.343) (1, 0.5)	(95 $\pm$ 3.0117) (1, 0.9063)	(95 $\pm$ 2.2588) (1, 0.9063)	(96.67 $\pm$ 2.2588) (1, 0.9355)	(96.67 $\pm$ 2.2588) (1, 0.9355)
O_vs_S	Entropy	0	(86.67 $\pm$ 15.1336) (0.7241, 1)	(95 $\pm$ 3.0117) (1, 0.9063)	(100 $\pm$ 2.2588) (1, 1)	(96.67 $\pm$ 2.2588) (1, 0.9355)	(95 $\pm$ 3.0117) (1, 0.9063)
		0.05	(86.67 $\pm$ 16.2185) (0.7241, 1)	(95 $\pm$ 3.0117) (1, 0.9063)	(100 $\pm$ 2.2588) (1, 1)	(96.67 $\pm$ 2.2588) (1, 0.9355)	(95 $\pm$ 2.2588) (1, 0.9063)
		0.1	(50 $\pm$ 14.85) (1, 0.4915)	(95 $\pm$ 3.0117) (1, 0.9063)	(95 $\pm$ 3.0117) (1, 0.9063)	(96.67 $\pm$ 2.2588) (1, 0.9355)	(96.67 $\pm$ 2.2588) (1, 0.9355)
	T-Test	0	(51.67 $\pm$ 19.343) (1, 0.5)	(93.33 $\pm$ 2.2588) (0.9655, 0.9032)	(93.33 $\pm$ 4.9943) (0.9655, 0.9032)	(93.33 $\pm$ 3.0117) (0.931, 0.931)	(95 $\pm$ 3.0117) (0.9655, 0.9333)
		0.05	(51.67 $\pm$ 19.343) (1, 0.5)	(93.33 $\pm$ 3.0117) (0.9655, 0.9032)	(93.33 $\pm$ 4.8211) (0.9655, 0.9032)	(95 $\pm$ 3.0117) (0.931, 0.9643)	(93.33 $\pm$ 3.0117) (0.931, 0.931)
		0.1	(51.67 $\pm$ 14.0658) (1, 0.5)	(93.33 $\pm$ 3.4503) (0.9655, 0.9032)	(95 $\pm$ 3.4503) (0.9655, 0.9333)	(95 $\pm$ 3.0117) (0.931, 0.9643)	(93.33 $\pm$ 3.0117) (0.931, 0.931)

CHAPTER 3. ENHANCED PINBALL LOSS FRAMEWORKS FOR ROBUST CLASSIFICATION AND GENERALIZATION

Table 3.31: Continued from previous page-Performance of proposed Tpin-UTWSVM model with baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17] (Accuracy $\pm$ sd) (Recall,Precision)	TWSVM [22] (Accuracy $\pm$ sd) (Recall,Precision)	UTSVM [26] (Accuracy $\pm$ sd) (Recall,Precision)	Pin-GTSVM [101] (Accuracy $\pm$ sd) (Recall,Precision)	Tpin-UTWSVM (Accuracy $\pm$ sd) (Recall,Precision)
O_vs_S	Wilcoxon	0	(51.67 $\pm$ 14.6772) (1, 0.5)	(96.67 $\pm$ 2.2588) (1, 0.9355)	(93.33 $\pm$ 3.0117) (1, 0.8788)	(95 $\pm$ 3.4503) (0.931, 0.9643)	(96.67 $\pm$ 3.0117) (0.9655, 0.9655)
		0.05	(51.67 $\pm$ 14.6772) (1, 0.5)	(91.67 $\pm$ 2.2588) (0.9655, 0.875)	(93.33 $\pm$ 3.0117) (1, 0.8788)	(93.33 $\pm$ 4.9943) (1, 0.8788)	(96.67 $\pm$ 3.0117) (0.9655, 0.9655)
		0.1	(51.67 $\pm$ 14.6772) (1, 0.5)	(91.67 $\pm$ 2.2588) (0.9655, 0.875)	(93.33 $\pm$ 2.2588) (1, 0.8788)	(93.33 $\pm$ 4.9943) (1, 0.8788)	(96.67 $\pm$ 3.0117) (0.9655, 0.9655)
Z_vs_F	Bhattacharyya	0	(55 $\pm$ 12.2567) (0.7586, 0.5238)	(68.33 $\pm$ 15.2269) (0.7241, 0.6563)	(78.33 $\pm$ 13.4897) (0.7241, 0.8077)	(73.33 $\pm$ 12.2336) (0.7931, 0.697)	(70 $\pm$ 12.1639) (0.7931, 0.6571)
		0.05	(55 $\pm$ 12.2336) (0.7931, 0.5227)	(65 $\pm$ 13.2352) (0.6897, 0.625)	(78.33 $\pm$ 12.2336) (0.7241, 0.8077)	(63.33 $\pm$ 19.1071) (0.6207, 0.6207)	(66.67 $\pm$ 13.041) (0.6207, 0.6667)
		0.1	(53.33 $\pm$ 10.6745) (0.7931, 0.5111)	(65 $\pm$ 9.404) (0.6552, 0.6333)	(70 $\pm$ 13.5526) (0.6207, 0.72)	(60 $\pm$ 13.3843) (0.4828, 0.6087)	(66.67 $\pm$ 11.6885) (0.6207, 0.6667)
Z_vs_F	Entropy	0	(55 $\pm$ 12.2567) (0.7586, 0.5238)	(68.33 $\pm$ 15.2269) (0.7241, 0.6563)	(78.33 $\pm$ 13.4897) (0.7241, 0.8077)	(73.33 $\pm$ 12.2336) (0.7931, 0.697)	(70 $\pm$ 12.1639) (0.7931, 0.6571)
		0.05	(58.33 $\pm$ 12.2336) (0.7931, 0.5476)	(63.33 $\pm$ 13.2352) (0.6552, 0.6129)	(76.67 $\pm$ 13.0627) (0.6897, 0.8)	(61.67 $\pm$ 19.3576) (0.5862, 0.6071)	(75 $\pm$ 11.8809) (0.7931, 0.7188)
		0.1	(55 $\pm$ 10.6745) (0.7931, 0.5227)	(60 $\pm$ 13.4897) (0.5862, 0.5862)	(73.33 $\pm$ 11.761) (0.6897, 0.7407)	(60 $\pm$ 11.761) (0.4483, 0.619)	(58.33 $\pm$ 12.7108) (0.4483, 0.5909)
Z_vs_F	T-Test	0	(63.33 $\pm$ 13.1277) (0.7241, 0.6)	(76.67 $\pm$ 8.2822) (0.7241, 0.7778)	(75 $\pm$ 8.2479) (0.6897, 0.7692)	(75 $\pm$ 6.9007) (0.6897, 0.7692)	(80 $\pm$ 7.8608) (0.8276, 0.7742)
		0.05	(63.33 $\pm$ 14.3846) (0.7241, 0.6)	(80 $\pm$ 9.6421) (0.7931, 0.7931)	(71.67 $\pm$ 8.1092) (0.6552, 0.7308)	(75 $\pm$ 6.7763) (0.6897, 0.7692)	(76.67 $\pm$ 6.7763) (0.8276, 0.7273)
		0.1	(65 $\pm$ 13.9849) (0.7586, 0.6111)	(81.67 $\pm$ 9.6421) (0.8276, 0.8)	(71.67 $\pm$ 10.1015) (0.6552, 0.7308)	(75 $\pm$ 9.788) (0.6897, 0.7692)	(78.33 $\pm$ 7.6783) (0.7241, 0.8077)
Z_vs_F	Wilcoxon	0	(80 $\pm$ 12.9538) (0.8966, 0.7429)	(81.67 $\pm$ 6.9007) (0.8276, 0.8)	(76.67 $\pm$ 7.5292) (0.6552, 0.8261)	(86.67 $\pm$ 15.9542) (0.8276, 0.8889)	(76.67 $\pm$ 9.9887) (0.7586, 0.7586)
		0.05	(80 $\pm$ 12.9538) (0.8966, 0.7429)	(80 $\pm$ 6.9007) (0.7931, 0.7931)	(76.67 $\pm$ 5.8321) (0.6552, 0.8261)	(86.67 $\pm$ 16.078) (0.8276, 0.8889)	(81.67 $\pm$ 5.6344) (0.8276, 0.8)
		0.1	(80 $\pm$ 12.9538) (0.8966, 0.7429)	(80 $\pm$ 5.8805) (0.7931, 0.7931)	(76.67 $\pm$ 7.1031) (0.7241, 0.7778)	(78.33 $\pm$ 5.8805) (0.7931, 0.7667)	(83.33 $\pm$ 6.2543) (0.8276, 0.8276)
Z_vs_N	Bhattacharyya	0	(75 $\pm$ 8.3842) (1, 0.6591)	(86.67 $\pm$ 14.3055) (0.7931, 0.92)	(75 $\pm$ 13.9647) (0.7241, 0.75)	(80 $\pm$ 20.2031) (0.6552, 0.9048)	(85 $\pm$ 15.1336) (0.7931, 0.8846)
		0.05	(75 $\pm$ 8.4179) (1, 0.6591)	(86.67 $\pm$ 14.6965) (0.7931, 0.92)	(75 $\pm$ 13.1277) (0.7241, 0.75)	(80 $\pm$ 21.4153) (0.6207, 0.9474)	(85 $\pm$ 15.0773) (0.7931, 0.8846)
		0.1	(75 $\pm$ 8.4179) (1, 0.6591)	(91.67 $\pm$ 14.6965) (0.8966, 0.9286)	(73.33 $\pm$ 13.9647) (0.8276, 0.6857)	(83.33 $\pm$ 14.3055) (0.7241, 0.913)	(81.67 $\pm$ 16.7684) (0.7241, 0.875)
Z_vs_N	Entropy	0	(75 $\pm$ 9.0351) (1, 0.6591)	(90 $\pm$ 15.1336) (0.8966, 0.8966)	(75 $\pm$ 14.3649) (0.7586, 0.7333)	(80 $\pm$ 20.3289) (0.6552, 0.9048)	(83.33 $\pm$ 14.6965) (0.7586, 0.88)
		0.05	(75 $\pm$ 8.4179) (1, 0.6591)	(90 $\pm$ 14.3846) (0.8966, 0.8966)	(75 $\pm$ 14.3846) (0.7586, 0.7333)	(85 $\pm$ 14.5997) (0.7931, 0.8846)	(85 $\pm$ 14.6965) (0.7931, 0.8846)
		0.1	(75 $\pm$ 9.6421) (1, 0.6591)	(88.33 $\pm$ 15.5036) (0.8621, 0.8929)	(75 $\pm$ 12.6885) (0.8276, 0.7059)	(85 $\pm$ 15.5948) (0.7931, 0.8846)	(86.67 $\pm$ 14.3649) (0.8276, 0.8889)
Z_vs_N	ROC	0	(68.33 $\pm$ 14.7542) (0.931, 0.6136)	(78.33 $\pm$ 9.788) (0.7586, 0.7857)	(76.67 $\pm$ 8.7805) (0.8276, 0.7273)	(81.67 $\pm$ 10.2131) (0.8621, 0.7813)	(78.33 $\pm$ 9.0351) (0.7586, 0.7857)
		0.05	(68.33 $\pm$ 14.7542) (0.931, 0.6136)	(78.33 $\pm$ 9.788) (0.7586, 0.7857)	(76.67 $\pm$ 9.0351) (0.8276, 0.7273)	(80 $\pm$ 9.0664) (0.8621, 0.7576)	(78.33 $\pm$ 9.0351) (0.7586, 0.7857)
		0.1	(68.33 $\pm$ 14.7542) (0.931, 0.6136)	(78.33 $\pm$ 9.404) (0.7586, 0.7857)	(78.33 $\pm$ 7.5668) (0.8276, 0.75)	(78.33 $\pm$ 9.0664) (0.8276, 0.75)	(78.33 $\pm$ 9.0351) (0.7586, 0.7857)
Z_vs_N	T-Test	0	(70 $\pm$ 13.8013) (0.931, 0.6279)	(76.67 $\pm$ 14.2857) (0.8276, 0.7273)	(68.33 $\pm$ 10.351) (0.7241, 0.6563)	(63.33 $\pm$ 10.1295) (0.7241, 0.6)	(78.33 $\pm$ 12.5085) (0.7931, 0.7667)
		0.05	(70 $\pm$ 13.6568) (0.931, 0.6279)	(80 $\pm$ 13.3843) (0.7931, 0.7931)	(68.33 $\pm$ 10.351) (0.7241, 0.6563)	(63.33 $\pm$ 9.9887) (0.7241, 0.6)	(80 $\pm$ 12.3489) (0.8276, 0.7742)
		0.1	(70 $\pm$ 14.983) (0.931, 0.6279)	(78.33 $\pm$ 12.6213) (0.8276, 0.75)	(68.33 $\pm$ 11.2687) (0.7241, 0.6563)	(63.33 $\pm$ 9.5535) (0.7241, 0.6)	(83.33 $\pm$ 14.0658) (0.8276, 0.8276)
Z_vs_N	Wilcoxon	0	(73.33 $\pm$ 12.1639) (1, 0.6444)	(86.67 $\pm$ 8.1092) (0.8966, 0.8387)	(75 $\pm$ 8.7805) (0.7931, 0.7188)	(81.67 $\pm$ 5.6344) (0.8276, 0.8)	(86.67 $\pm$ 8.1092) (0.8966, 0.8387)
		0.05	(70 $\pm$ 12.1639) (1, 0.617)	(81.67 $\pm$ 11.8809) (0.7586, 0.8462)	(76.67 $\pm$ 9.9887) (0.7241, 0.7778)	(85 $\pm$ 9.9887) (0.8621, 0.8333)	(83.33 $\pm$ 9.788) (0.7931, 0.8519)
		0.1	(70 $\pm$ 12.1639) (1, 0.617)	(80 $\pm$ 12.1405) (0.7241, 0.84)	(78.33 $\pm$ 9.404) (0.7241, 0.8077)	(83.33 $\pm$ 10.7539) (0.8276, 0.8276)	(81.67 $\pm$ 9.788) (0.7586, 0.8462)

Table 3.31: Continued from previous page-Performance of proposed Tpin-UTWSVM model with baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17] (Accuracy $\pm$ sd) (Recall,Precision)	TWSVM [22] (Accuracy $\pm$ sd) (Recall,Precision)	UTSVM [26] (Accuracy $\pm$ sd) (Recall,Precision)	Pin-GTSVM [101] (Accuracy $\pm$ sd) (Recall,Precision)	Tpin-UTWSVM (Accuracy $\pm$ sd) (Recall,Precision)
Z_vs_S	Bhattacharyya	0	(95 $\pm$ 8.9087) (0.8966, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
		0.05	(95 $\pm$ 8.7805) (0.8966, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
		0.1	(88.33 $\pm$ 7.5668) (0.7586, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
Z_vs_S	Entropy	0	(95 $\pm$ 8.9087) (0.8966, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
		0.05	(93.33 $\pm$ 11.1677) (0.8621, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
		0.1	(93.33 $\pm$ 8.1092) (0.8621, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
Z_vs_S	T-Test	0	(78.33 $\pm$ 12.7997) (0.5517, 1)	(96.67 $\pm$ 0) (1, 0.9355)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(98.33 $\pm$ 0) (1, 0.9667)
		0.05	(80 $\pm$ 14.5997) (0.5862, 1)	(96.67 $\pm$ 0) (1, 0.9355)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
		0.1	(91.67 $\pm$ 25.5218) (0.8276, 1)	(96.67 $\pm$ 0) (1, 0.9355)	(95 $\pm$ 0) (1, 0.9063)	(100 $\pm$ 0) (1, 1)	(100 $\pm$ 0) (1, 1)
average accuracy		0	71.92	85.25	84.29	85.96	86.31
		0.05	71.72	85.2	84.04	85.45	86.21
		0.1	69.65	85.56	82.73	85.66	86.82
average rank		0	4.7	2.7	2.82	2.41	2.38
		0.05	4.79	2.7	2.98	2.41	2.12
		0.1	4.65	2.62	3.58	2.29	1.86

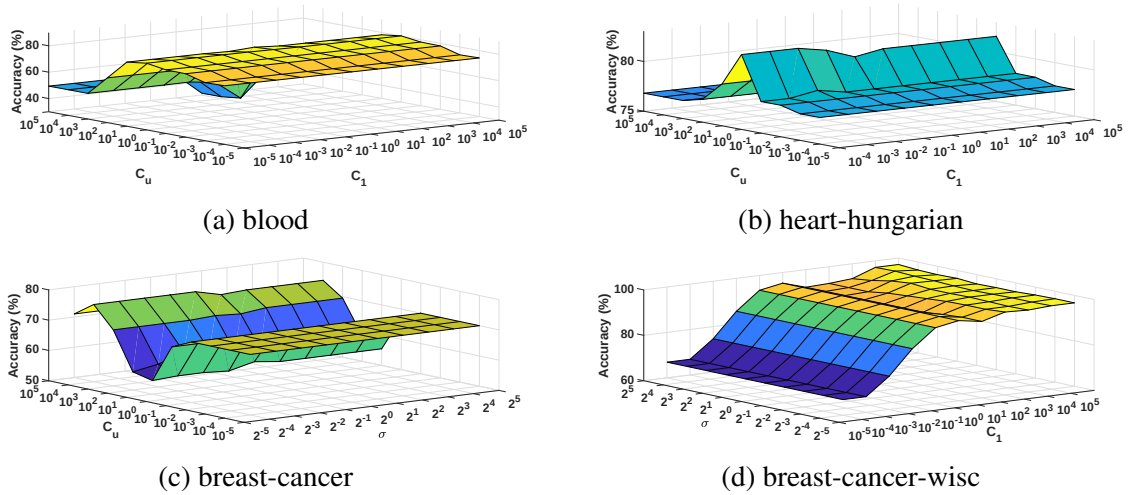


Figure 3.11: Sensitivity evaluation of the proposed Tpin-UTWSVM classifier on UCI binary datasets using user-specified parameters and Gaussian kernel ($C_1 = C_2$).

CHAPTER 3. ENHANCED PINBALL LOSS FRAMEWORKS FOR ROBUST CLASSIFICATION AND GENERALIZATION

Table 3.32: Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17]	TWSVM [22]	UTSVM [26]	Pin-GTSVM [101]	Tpin-UTWSVM
size ($f \times s$)			(Time(s), τ) (C_1, σ)	Time(s) ($C_1 = C_2, \sigma$)	(Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Time(s) ($C_1 = C_2, \sigma$)	(Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
F_vs_S	Bhattacharyya	0	(0.4238, 0.1) (1, 1)	0.0023 (0.00001, 2)	(0.0024, 0.00001) (0.00001, 2, 0.1)	0.0033 (0.0001, 4)	(0.0108, 0.00001) (0.0001, 4, 0.6)
		0.05	(0.3733, 0.1) (1, 1)	0.0022 (0.0001, 4)	(0.0029, 0.00001) (0.00001, 2, 0.1)	0.0037 (0.0001, 4)	(0.0326, 0.00001) (0.0001, 4, 0.6)
		0.1	(0.4949, 0.1) (1, 1)	0.0059 (0.0001, 4)	(0.0092, 1) (0.00001, 32, 0.1)	0.0036 (0.00001, 4)	(0.0112, 0.00001) (0.0001, 4, 0.3)
		0.1	(0.4273, 0.1) (1, 1)	0.0129 (0.00001, 2)	(0.0061, 0.00001) (0.00001, 2, 0.1)	0.0038 (0.0001, 4)	(0.0035, 0.00001) (0.0001, 4, 0.6)
F_vs_S	Entropy	0	(0.4098, 0.1) (1, 1)	0.0044 (0.0001, 4)	(0.0036, 0.00001) (0.00001, 2, 0.1)	0.0029 (0.0001, 4)	(0.0034, 0.00001) (0.0001, 4, 0.6)
		0.05	(0.3089, 0.1) (1, 1)	0.008 (0.0001, 4)	(0.0098, 10) (0.00001, 32, 0.1)	0.0029 (0.00001, 4)	(0.0035, 0.00001) (0.0001, 4, 0.3)
		0.1	(0.4179, 0.1) (1, 1)	0.0046 (0.0001, 16)	(0.0031, 0.0001) (0.00001, 2, 0.1)	0.0041 (0.0001, 8)	(0.0035, 0.00001) (0.0001, 16, 0.1)
		0.05	(0.3715, 0.1) (1, 1)	0.0032 (0.0001, 16)	(0.0034, 0.0001) (0.00001, 2, 0.1)	0.0033 (0.0001, 8)	(0.0034, 0.00001) (0.0001, 16, 0.1)
F_vs_S	ROC	0	(0.2846, 0.1) (1, 1)	0.0031 (0.0001, 16)	(0.0031, 0.00001) (0.00001, 8, 0.1)	0.0031 (0.00001, 4)	(0.0036, 0.00001) (0.0001, 16, 0.1)
		0.05	(0.173, 0.1) (1, 1)	0.005 (0.00001, 16)	(0.0082, 1) (0.001, 16, 0.1)	0.004 (0.00001, 4)	(0.0036, 0.00001) (0.00001, 16, 0.1)
		0.1	(0.5232, 0.1) (1, 1)	0.007 (0.0001, 4)	(0.0034, 0.1) (0.001, 16, 0.1)	0.0029 (0.00001, 8)	(0.0365, 0.00001) (0.00001, 4, 0.1)
		0.1	(0.309, 0.1) (1, 1)	0.0022 (0.0001, 4)	(0.0047, 1) (0.001, 16, 0.5)	0.0034 (0.00001, 8)	(0.0201, 0.0001) (0.0001, 4, 0.5)
F_vs_S	T-Test	0	(0.4574, 0.1) (1, 1)	0.0023 (0.0001, 16)	(0.0043, 1) (0.00001, 16, 0.1)	0.0027 (0.00001, 16)	(0.0038, 0.0001) (0.001, 8, 0.1)
		0.05	(0.8652, 0.1) (1, 1)	0.0044 (0.00001, 16)	(0.0035, 1) (0.00001, 16, 0.1)	0.003 (0.00001, 4)	(0.011, 0.00001) (0.001, 8, 0.1)
		0.1	(0.6062, 0.1) (1, 1)	0.0061 (0.001, 8)	(0.012, 0.1) (0.00001, 16, 0.1)	0.0036 (0.00001, 8)	(0.0139, 0.00001) (0.001, 8, 0.1)
		0.1	(0.3573, 0.1) (10, 1)	0.0043 (0.0001, 4)	(0.003, 0.0001) (0.00001, 2, 0.1)	0.0022 (0.00001, 4)	(0.0036, 0.00001) (0.0001, 4, 0.1)
N_vs_S	ROC	0	(0.3466, 0.1) (10, 1)	0.0041 (0.0001, 4)	(0.0029, 0.0001) (0.00001, 2, 0.1)	0.0017 (0.00001, 4)	(0.0128, 0.00001) (0.00001, 4, 0.1)
		0.05	(0.2449, 0.1) (1, 1)	0.0162 (0.00001, 4)	(0.0025, 0.0001) (0.00001, 2, 0.1)	0.0023 (0.00001, 4)	(0.0035, 0.00001) (0.00001, 4, 0.1)
		0.1	(0.4479, 0.1) (10, 1)	0.0056 (0.0001, 4)	(0.0167, 0.0001) (0.00001, 2, 0.1)	0.0058 (0.00001, 4)	(0.0061, 0.00001) (0.00001, 4, 0.3)
		0.05	(0.1901, 0.1) (1, 1)	0.0053 (0.00001, 4)	(0.011, 0.1) (0.0001, 16, 0.1)	0.0019 (0.00001, 4)	(0.0099, 0.00001) (0.00001, 4, 0.1)
N_vs_S	T-Test	0	(2.0203, 0.1) (1, 1)	0.006 (0.00001, 4)	(0.0045, 0.00001) (0.0001, 4, 0.1)	0.0108 (0.00001, 4)	(0.0108, 0.00001) (0.00001, 4, 0.1)
		0.05	(0.0704, 0.1) (10, 1)	0.0045 (0.0001, 4)	(0.0051, 0.0001) (0.00001, 2, 0.1)	0.0021 (0.00001, 4)	(0.0035, 0.00001) (0.0001, 4, 0.1)
		0.1	(0.8785, 0.1) (1, 1)	0.004 (0.0001, 4)	(0.0034, 0.0001) (0.00001, 2, 0.1)	0.0013 (0.00001, 4)	(0.0038, 0.00001) (0.00001, 4, 0.1)
		0.1	(0.3614, 0.1) (10, 1)	0.0045 (0.0001, 4)	(0.0029, 0.00001) (0.0001, 4, 0.1)	0.0039 (0.00001, 4)	(0.0038, 0.00001) (0.00001, 4, 0.1)
O_vs_F	Bhattacharyya	0	(0.196, 0.1) (0.1, 0.5)	0.008 (0.0001, 2)	(0.0233, 1) (1, 16, 0.1)	0.0034 (0.001, 4)	(0.0039, 0.00001) (0.001, 4, 0.6)
		0.05	(0.3557, 0.1) (0.1, 0.5)	0.0032 (0.001, 8)	(0.0228, 1) (1, 16, 0.1)	0.0068 (0.001, 4)	(0.0037, 0.0001) (0.0001, 2, 0.6)
		0.1	(0.384, 0.1) (0.1, 0.5)	0.0044 (0.0001, 4)	(0.0318, 1) (10, 16, 0.1)	0.0021 (0.0001, 4)	(0.0039, 0.001) (0.0001, 2, 0.6)
		0.1	(0.5333, 0.1) (0.1, 0.5)	0.0046 (0.0001, 2)	(0.0264, 1) (1, 16, 0.1)	0.0015 (0.001, 4)	(0.0038, 0.00001) (0.001, 4, 0.6)
O_vs_F	Entropy	0	(0.4898, 0.1) (0.1, 0.5)	0.003 (0.0001, 2)	(0.0286, 1) (1, 16, 0.1)	0.0365 (0.001, 4)	(0.0035, 0.0001) (0.0001, 2, 0.6)
		0.05	(0.1146, 0.1) (0.1, 0.5)	0.0039 (0.0001, 2)	(0.0106, 0.1) (0.01, 8, 0.1)	0.0063 (0.0001, 4)	(0.0153, 0.0001) (0.001, 4, 0.1)
		0.1	(0.1092, 0.1) (0.1, 0.5)	0.0092 (0.0001, 2)	(0.0106, 0.1) (0.01, 8, 0.1)	0.0063 (0.0001, 4)	(0.0153, 0.0001) (0.001, 4, 0.1)
		0.1	(0.1092, 0.1) (0.1, 0.5)	0.0092 (0.0001, 2)	(0.0106, 0.1) (0.01, 8, 0.1)	0.0063 (0.0001, 4)	(0.0153, 0.0001) (0.001, 4, 0.1)

Table 3.32: Continued from previous page-Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17]	TWSVM [22]	UTWSVM [26]	Pin-GTSVM [101]	Tpin-UTWSVM
size ($f \times s$)			(Time(s), τ) (C_1, σ)	Time(s) ($C_1 = C_2, \sigma$)	(Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Time(s) ($C_1 = C_2, \sigma$)	(Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
O_vs_F	ROC	0	(0.1597, 0.1)	0.0043	(0.0072, 0.0001)	0.0025	(0.0043, 0.0001)
			(1, 0.03125)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.0001, 8)	(0.001, 16, 0.6)
		0.05	(0.0941, 0.1)	0.0042	(0.0021, 0.001)	0.0015	(0.0037, 0.00001)
			(1, 0.03125)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.0001, 8)	(0.0001, 8, 0.5)
		0.1	(0.1474, 0.1)	0.0077	(0.0039, 0.001)	0.0016	(0.0033, 0.00001)
			(1, 0.03125)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.0001, 8)	(0.0001, 8, 0.3)
O_vs_F	T-Test	0	(0.0727, 0.1)	0.0243	(0.0023, 0.01)	0.0015	(0.0034, 0.0001)
			(1, 0.0625)	(0.001, 16)	(0.00001, 8, 0.1)	(0.00001, 4)	(0.001, 8, 0.5)
		0.05	(0.074, 0.1)	0.0087	(0.0033, 0.01)	0.0071	(0.0034, 0.0001)
			(1, 0.0625)	(0.001, 16)	(0.00001, 8, 0.1)	(0.00001, 4)	(0.001, 8, 0.5)
		0.1	(0.0542, 0.1)	0.0036	(0.0017, 0.01)	0.0031	(0.0035, 0.00001)
			(1, 0.0625)	(0.0001, 4)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.001, 8, 0.5)
O_vs_N	Bhattacharyya	0	(0.1024, 0.1)	0.005	(0.0708, 0.0001)	0.0038	(0.0036, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.0001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.6)
		0.05	(0.0584, 0.1)	0.0048	(0.0042, 0.0001)	0.0029	(0.0101, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.0001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.6)
		0.1	(0.1596, 0.1)	0.0025	(0.0035, 0.0001)	0.0028	(0.0034, 0.0001)
			(1, 0.25)	(0.0001, 8)	(0.0001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.5)
O_vs_N	Entropy	0	(0.1593, 0.1)	0.0046	(0.0039, 0.0001)	0.0019	(0.0035, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.6)
		0.05	(0.0971, 0.1)	0.0057	(0.0037, 0.0001)	0.0015	(0.0112, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.0001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.6)
		0.1	(0.0714, 0.1)	0.0041	(0.0042, 0.0001)	0.0033	(0.0035, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.6)
O_vs_N	ROC	0	(0.1802, 0.1)	0.0061	(0.0067, 0.001)	0.0012	(0.0139, 0.00001)
			(0.1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.001, 8)	(0.0001, 8, 0.1)
		0.05	(0.2474, 0.1)	0.0047	(0.0039, 0.001)	0.0013	(0.0036, 0.00001)
			(0.1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.0001, 8)	(0.0001, 8, 0.1)
		0.1	(0.1067, 0.1)	0.0039	(0.0031, 0.001)	0.0013	(0.0035, 0.00001)
			(0.01, 0.0625)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.001, 8, 0.1)
O_vs_N	T-Test	0	(0.1693, 0.1)	0.0097	(0.0083, 0.0001)	0.0014	(0.0034, 0.001)
			(0.01, 0.125)	(0.001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.0001, 8, 0.5)
		0.05	(0.1438, 0.1)	0.0052	(0.0031, 0.0001)	0.0053	(0.0036, 0.001)
			(0.01, 0.0625)	(0.001, 8)	(0.0001, 8, 0.1)	(0.00001, 8)	(0.0001, 8, 0.5)
		0.1	(0.0616, 0.1)	0.0044	(0.0029, 0.0001)	0.0012	(0.0035, 0.00001)
			(1, 0.25)	(0.001, 8)	(0.0001, 8, 0.1)	(0.00001, 8)	(0.00001, 8, 0.6)
O_vs_N	Wilcoxon	0	(0.4515, 0.1)	0.007	(0.0024, 0.001)	0.0027	(0.0031, 0.001)
			(1, 0.25)	(0.001, 16)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.0001, 16, 0.5)
		0.05	(0.276, 0.1)	0.0037	(0.0173, 0.0001)	0.0048	(0.0034, 0.001)
			(1, 0.25)	(0.001, 32)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.001, 32, 0.5)
		0.1	(0.3436, 0.1)	0.0109	(0.0116, 0.0001)	0.0015	(0.0034, 0.00001)
			(1, 0.25)	(0.001, 16)	(0.0001, 8, 0.1)	(0.00001, 16)	(0.0001, 16, 0.1)
O_vs_S	Bhattacharyya	0	(0.2371, 0.1)	0.0033	(0.004, 0.0001)	0.0013	(0.0041, 0.00001)
			(1, 1)	(0.0001, 4)	(0.00001, 2, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
		0.05	(0.1745, 0.1)	0.0047	(0.0031, 0.0001)	0.0223	(0.0103, 0.00001)
			(1, 1)	(0.00001, 4)	(0.00001, 2, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
		0.1	(0.0994, 0.1)	0.0048	(0.0066, 0.00001)	0.0032	(0.0035, 0.00001)
			(1, 0.5)	(0.00001, 4)	(0.00001, 8, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
O_vs_S	Entropy	0	(0.2877, 0.1)	0.0036	(0.0036, 0.0001)	0.0016	(0.0087, 0.00001)
			(1, 1)	(0.0001, 4)	(0.00001, 2, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
		0.05	(0.3541, 0.1)	0.0043	(0.0019, 0.0001)	0.0041	(0.0042, 0.00001)
			(1, 1)	(0.00001, 4)	(0.00001, 2, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
		0.1	(0.2072, 0.1)	0.0067	(0.0024, 0.00001)	0.0022	(0.0034, 0.00001)
			(1, 0.25)	(0.00001, 4)	(0.00001, 8, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
O_vs_S	T-Test	0	(0.1848, 0.1)	0.0057	(0.0018, 0.001)	0.0034	(0.0194, 0.00001)
			(1, 0.5)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
		0.05	(0.2979, 0.1)	0.004	(0.0053, 0.001)	0.0023	(0.0052, 0.00001)
			(1, 0.5)	(0.00001, 4)	(0.00001, 4, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)
		0.1	(0.2239, 0.1)	0.0158	(0.0018, 0.001)	0.004	(0.011, 0.00001)
			(1, 0.25)	(0.00001, 4)	(0.00001, 4, 0.1)	(0.00001, 4)	(0.00001, 4, 0.1)

Table 3.32: Continued from previous page-Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.

Dataset size ($f \times s$)	Features	r	pin-SVM [17] (Time(s), τ) (C_1, σ)	TWSVM [22] Time(s) ($C_1 = C_2, \sigma$)	UTSVM [26] (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Pin-GTSVM [101] Time(s) ($C_1 = C_2, \sigma$)	Tpin-UTWSVM (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
O_vs_S	Wilcoxon	0	(0.0999, 0.1)	0.0052	(0.002, 0.001)	0.0024	(0.0034, 0.00001)
			(1, 0.25)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.00001, 4)	(0.00001, 4, 0.3)
		0.05	(0.1028, 0.1)	0.0172	(0.0022, 0.001)	0.0107	(0.0034, 0.00001)
			(1, 0.25)	(0.00001, 4)	(0.00001, 4, 0.1)	(0.00001, 8)	(0.00001, 4, 0.3)
		0.1	(0.0773, 0.1)	0.0071	(0.0018, 0.001)	0.0031	(0.0036, 0.00001)
			(1, 0.25)	(0.00001, 4)	(0.00001, 4, 0.1)	(0.00001, 8)	(0.00001, 4, 0.5)
Z_vs_F	Bhattacharyya	0	(0.3403, 0.1)	0.0063	(0.0026, 0.00001)	0.003	(0.0035, 0.01)
			(1, 0.5)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.0001, 2)	(0.001, 8, 0.1)
		0.05	(0.1516, 0.1)	0.0044	(0.0023, 0.0001)	0.0017	(0.0043, 0.00001)
			(1, 0.5)	(0.0001, 4)	(0.0001, 4, 0.1)	(0.0001, 4)	(0.0001, 4, 0.1)
		0.1	(0.1541, 0.1)	0.0048	(0.0023, 0.00001)	0.0043	(0.004, 0.00001)
			(1, 0.5)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.0001, 4)	(0.0001, 4, 0.6)
Z_vs_F	Entropy	0	(0.2304, 0.1)	0.0157	(0.0145, 0.00001)	0.0013	(0.0033, 0.01)
			(1, 0.5)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.0001, 2)	(0.001, 8, 0.1)
		0.05	(0.1035, 0.1)	0.019	(0.0027, 0.0001)	0.0032	(0.0034, 0.001)
			(1, 0.5)	(0.0001, 4)	(0.00001, 4, 0.1)	(0.001, 4)	(0.0001, 2, 0.3)
		0.1	(0.0817, 0.1)	0.0081	(0.0025, 0.00001)	0.0017	(0.0041, 0.00001)
			(1, 0.5)	(0.001, 4)	(0.00001, 4, 0.1)	(0.001, 4)	(0.001, 4, 0.1)
Z_vs_F	T-Test	0	(0.138, 0.1)	0.004	(0.0019, 0.0001)	0.0052	(0.0045, 0.1)
			(1, 0.25)	(0.01, 32)	(0.0001, 8, 0.1)	(0.00001, 32)	(0.01, 32, 0.6)
		0.05	(0.2069, 0.1)	0.0091	(0.0022, 0.0001)	0.004	(0.0042, 0.1)
			(10, 0.25)	(0.001, 16)	(0.0001, 8, 0.1)	(0.00001, 32)	(0.01, 32, 0.6)
		0.1	(0.0728, 0.1)	0.0026	(0.0023, 0.0001)	0.0023	(0.0036, 0.001)
			(1, 0.25)	(0.001, 16)	(0.001, 8, 0.1)	(0.00001, 32)	(0.0001, 16, 0.3)
Z_vs_F	Wilcoxon	0	(0.19, 0.1)	0.0048	(0.0022, 0.0001)	0.0024	(0.0039, 0.0001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 4)	(0.00001, 8, 0.6)
		0.05	(0.1707, 0.1)	0.004	(0.0018, 0.0001)	0.0013	(0.0034, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 4)	(0.0001, 8, 0.1)
		0.1	(0.0775, 0.1)	0.0043	(0.0019, 0.0001)	0.0095	(0.0034, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.0001, 8, 0.1)
Z_vs_N	Bhattacharyya	0	(0.0596, 0.1)	0.0052	(0.0024, 0.0001)	0.0038	(0.0036, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.0001, 4)	(0.0001, 8, 0.6)
		0.05	(0.1309, 0.1)	0.0036	(0.0019, 0.0001)	0.0043	(0.0031, 0.00001)
			(1, 0.25)	(0.0001, 8)	(0.001, 8, 0.1)	(0.00001, 4)	(0.0001, 8, 0.6)
		0.1	(0.1867, 0.1)	0.0052	(0.0053, 0.01)	0.0021	(0.0127, 0.001)
			(1, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.0001, 8, 0.1)
Z_vs_N	Entropy	0	(0.071, 0.1)	0.0049	(0.0018, 0.0001)	0.0012	(0.0035, 0.0001)
			(1, 0.25)	(0.0001, 8)	(0.0001, 8, 0.1)	(0.00001, 4)	(0.0001, 8, 0.3)
		0.05	(0.1027, 0.1)	0.0078	(0.0021, 0.0001)	0.007	(0.0037, 0.001)
			(1, 0.25)	(0.0001, 8)	(0.001, 8, 0.1)	(0.00001, 8)	(0.0001, 8, 0.3)
		0.1	(0.1255, 0.1)	0.0043	(0.002, 0.01)	0.0014	(0.0036, 0.00001)
			(10, 0.25)	(0.0001, 8)	(0.00001, 8, 0.1)	(0.00001, 8)	(0.0001, 8, 0.6)
Z_vs_N	ROC	0	(0.4702, 0.1)	0.0069	(0.002, 0.0001)	0.0069	(0.0054, 0.00001)
			(1, 0.125)	(0.00001, 8)	(0.00001, 8, 0.1)	(0.00001, 16)	(0.00001, 8, 0.6)
		0.05	(0.1132, 0.1)	0.0041	(0.002, 0.0001)	0.0031	(0.0035, 0.00001)
			(1, 0.125)	(0.00001, 8)	(0.00001, 8, 0.1)	(0.00001, 16)	(0.00001, 8, 0.6)
		0.1	(0.0882, 0.1)	0.004	(0.0021, 0.0001)	0.0014	(0.0037, 0.00001)
			(1, 0.125)	(0.00001, 8)	(0.00001, 8, 0.1)	(0.00001, 16)	(0.00001, 8, 0.6)

Table 3.32: Continued from previous page-Optimal parameters of proposed Tpin-UTWSVM model and baseline models on EEG datasets with Gaussian kernel.

Dataset	Features	r	pin-SVM [17] (Time(s), τ) (C_1, σ)	TWSVM [22] Time(s) ($C_1 = C_2, \sigma$)	UTWSVM [26] (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Pin-GTSVM [101] Time(s) ($C_1 = C_2, \sigma$)	Tpin-UTWSVM (Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
Z_vs_N	Wilcoxon	0	(0.2177, 0.1) (1, 0.125)	0.0068 (0.00001, 16)	(0.0019, 0.0001) (0.00001, 8, 0.1)	0.0023 (0.001, 32)	(0.0034, 0.00001) (0.00001, 16, 0.1)
		0.05	(0.1137, 0.1) (1, 0.125)	0.0072 (0.00001, 8)	(0.0019, 0.001) (0.00001, 8, 0.1)	0.0012 (0.00001, 16)	(0.0033, 0.00001) (0.00001, 8, 0.6)
		0.1	(0.1421, 0.1) (1, 0.125)	0.004 (0.00001, 8)	(0.002, 0.001) (0.00001, 8, 0.1)	0.0045 (0.00001, 16)	(0.0034, 0.0001) (0.00001, 8, 0.6)
Z_vs_S	Bhattacharyya	0	(0.1579, 0.1) (0.1, 1)	0.0046 (0.00001, 2)	(0.002, 0.00001) (0.00001, 2, 0.1)	0.0023 (0.00001, 2)	(0.0042, 0.00001) (0.00001, 2, 0.1)
		0.05	(0.1351, 0.1) (0.1, 1)	0.0047 (0.00001, 2)	(0.0024, 0.00001) (0.00001, 2, 0.1)	0.0041 (0.00001, 2)	(0.0099, 0.00001) (0.00001, 2, 0.1)
		0.1	(0.3219, 0.1) (1, 1)	0.0068 (0.00001, 2)	(0.0052, 0.00001) (0.00001, 2, 0.1)	0.0015 (0.00001, 2)	(0.0037, 0.00001) (0.00001, 2, 0.1)
Z_vs_S	Entropy	0	(0.2352, 0.1) (0.1, 1)	0.0057 (0.00001, 2)	(0.0045, 0.00001) (0.00001, 2, 0.1)	0.0031 (0.00001, 2)	(0.0142, 0.00001) (0.00001, 2, 0.1)
		0.05	(0.1345, 0.1) (0.1, 1)	0.0026 (0.00001, 2)	(0.0032, 0.00001) (0.00001, 2, 0.1)	0.004 (0.00001, 2)	(0.0034, 0.00001) (0.00001, 2, 0.1)
		0.1	(0.0905, 0.1) (1, 1)	0.007 (0.00001, 2)	(0.0052, 0.00001) (0.00001, 2, 0.1)	0.0053 (0.00001, 2)	(0.0034, 0.00001) (0.00001, 2, 0.1)
Z_vs_S	T-Test	0	(0.0986, 0.1) (1, 1)	0.0042 (0.00001, 4)	(0.0017, 0.0001) (0.00001, 2, 0.1)	0.0037 (0.00001, 4)	(0.0035, 0.00001) (0.00001, 4, 0.1)
		0.05	(0.2026, 0.1) (10, 1)	0.0043 (0.00001, 4)	(0.0022, 0.0001) (0.00001, 2, 0.1)	0.0013 (0.00001, 4)	(0.0034, 0.00001) (0.00001, 4, 0.1)
		0.1	(0.1042, 0.1) (1, 1)	0.0039 (0.00001, 4)	(0.0092, 0.00001) (0.00001, 4, 0.1)	0.0044 (0.00001, 4)	(0.0036, 0.00001) (0.00001, 4, 0.1)
average time		0	0.2457	0.0064	0.0076	0.0028	0.0054
		0.05	0.2538	0.0055	0.0049	0.0051	0.0072
		0.1	0.2505	0.0058	0.0054	0.0033	0.0057

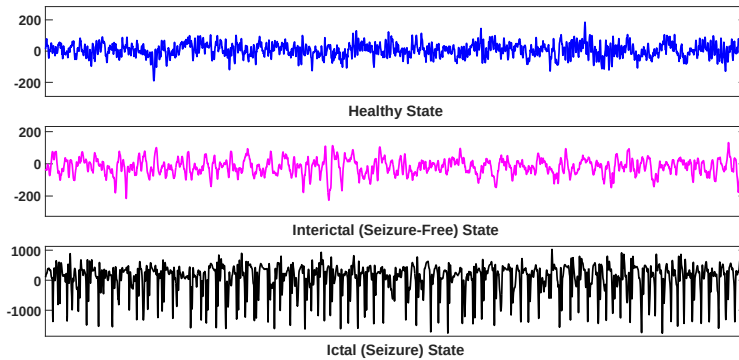
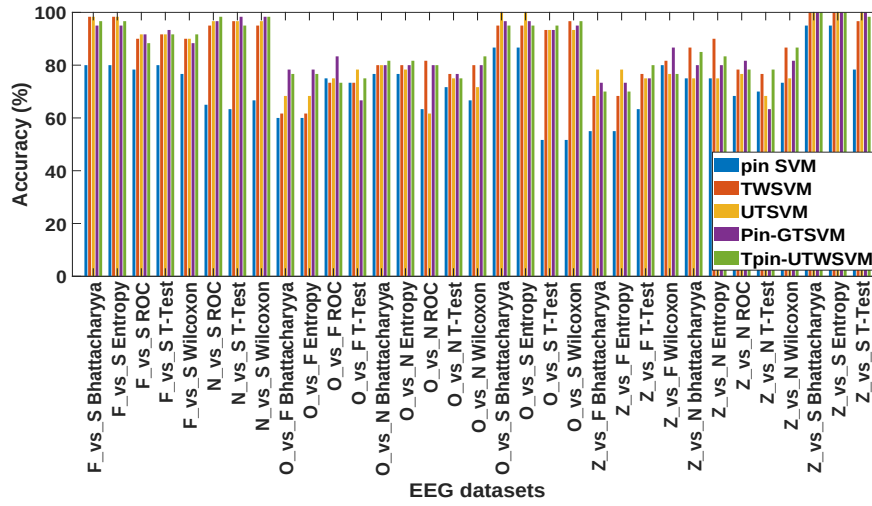
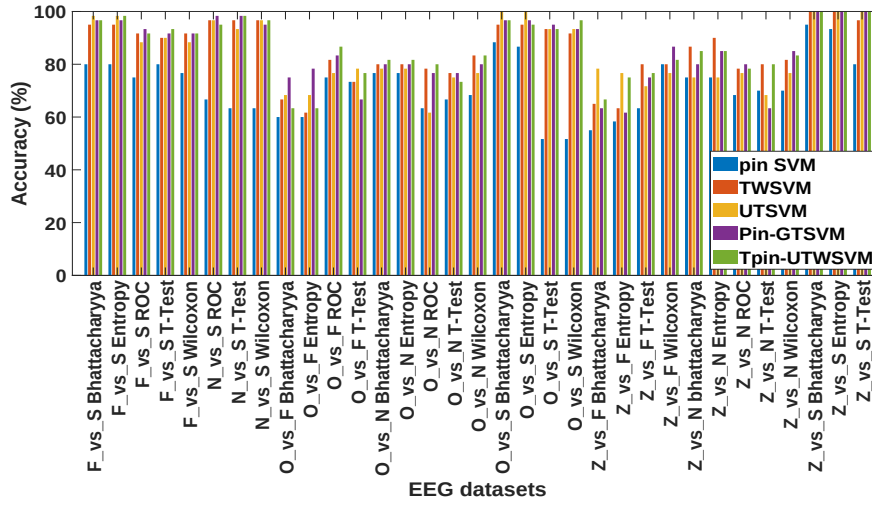


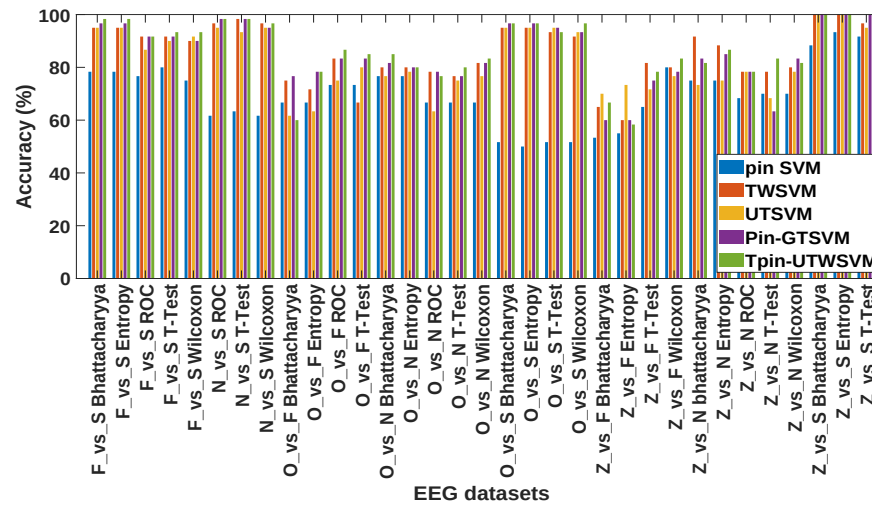
Figure 3.12: Samples of EEG signal for different states.



(a) Bar graph to compare the accuracies of models on EEG datasets with $r = 0$.



(b) Bar graph to compare the accuracies of models on EEG datasets with $r = 0.05$.



(c) Bar graph to compare the accuracies of models on EEG datasets with $r = 0.1$.

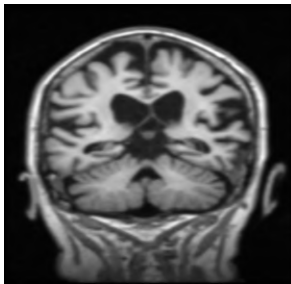
Figure 3.13: Bar graph representation to compare the performance of models over EEG datasets with different noise levels.

Table 3.33: Pairwise win-tie-loss of the proposed Tpin-UTWSVM with respect to baseline models over EEG datasets. With noise level r in the column, comparison of row method with the proposed Tpin-UTWSVM model. Here, [a,b,c] indicates that the proposed Tpin-UTWSVM technique wins ‘a’ number of times, ties ‘b’ number of times, and loses ‘c’ number of times in comparison to the row model.

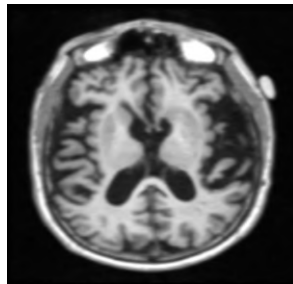
Model	Tpin-UTWSVM			
	$r = 0$	$r = 0.05$	$r = 0.1$	Total
pin-SVM [17]	[31, 0, 2]	[33, 0, 0]	[32, 0, 1]	[96, 0, 3]
TWSVM [22]	[15, 9, 9]	[17, 10, 6]	[19, 8, 6]	[51, 27, 21]
UTSVM [26]	[17, 5, 11]	[17, 6, 10]	[26, 3, 4]	[60, 14, 25]
Pin-GTSVM [101]	[15, 3, 15]	[18, 4, 11]	[19, 4, 10]	[52, 11, 36]

Table 3.34: Performance of proposed Tpin-UTWSVM model with baseline models on AD datasets with Gaussian kernel.

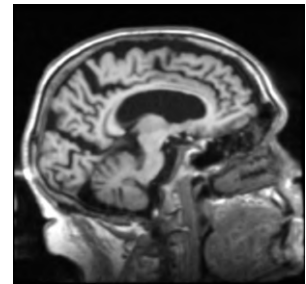
Dataset	r	pin-SVM [17]	TWSVM [22]	UTSVM [26]	Pin-GTSVM [101]	Tpin-UTWSVM
		(Accuracy $\pm$ sd) (Recall,Precision)	(Accuracy $\pm$ sd) (Recall,Precision)	(Accuracy $\pm$ sd) (Recall,Precision)	(Accuracy $\pm$ sd) (Recall,Precision)	(Accuracy $\pm$ sd) (Recall,Precision)
size ($f \times s$) (415 $\times$ 91)		(Time(s), τ) (C_1, σ)	Time(s) ($C_1 = C_2, \sigma$)	(Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)	Time(s) ($C_1 = C_2, \sigma$)	(Time(s), C_u) ($C_1 = C_2, \sigma, \epsilon$)
CN_vs_AD	0	82.26 $\pm$ 6.63 (0.6863, 0.8537) (0.5428, 0.1) (1, 0.125)	(86.29 $\pm$ 5.72) (0.9412, 0.7742) 0.0114 (0.1, 4)	(76.61 $\pm$ 8.98) (0.549, 0.8235) (0.0122, 0.00001) (0.001, 8, 0.1)	(89.52 $\pm$ 6.62) (0.8824, 0.8654) 0.0043 (0.00001, 8)	(90.32 $\pm$ 6.64) (0.8824, 0.8824) (0.008, 0.0001) (0.00001, 8, 0.5)
CN_vs_AD	0.05	82.26 $\pm$ 6 (0.6863, 0.8537) (0.5373, 0.1) (1, 0.125)	(88.71 $\pm$ 5.5) (0.9216, 0.8246) 0.0237 (0.00001, 0.5)	(86.29 $\pm$ 14.41) (0.7647, 0.8864) (0.008, 0.01) (0.00001, 4, 0.1)	(89.52 $\pm$ 6.58) (0.8824, 0.8654) 0.0039 (0.00001, 8)	(93.55 $\pm$ 8.24) (0.902, 0.9388) (0.0065, 0.0001) (0.0001, 8, 0.3)
CN_vs_AD	0.1	83.06 $\pm$ 6.42 (0.6863, 0.875) (0.7724, 0.1) (1, 0.125)	(87.1 $\pm$ 9.75) (0.902, 0.807) 0.008 (0.001, 1)	(87.1 $\pm$ 12.85) (0.7647, 0.907) (0.0078, 0.01) (0.00001, 4, 0.1)	(90.32 $\pm$ 7.34) (0.8627, 0.898) 0.0045 (0.01, 16)	(90.32 $\pm$ 6.45) (0.8824, 0.8824) (0.0072, 0.00001) (0.00001, 8, 0.3)



(a) coronal plane



(b) axial plane



(c) sagittal plane

Figure 3.14: MRI image of brain in three different planes

3.2.4.2 Experimental analysis over EEG data

We compared the proposed Tpin-UTWSVM model with the baseline models, pin-SVM [17], TWSVM [22], UTSVM [26] and Pin-GTSVM [101] on EEG dataset. Each of the dataset considered has 200 samples and 200 features. Table 3.31 depicts the accuracy, standard deviation (sd), recall and precision of each model with different noise levels ($r = 0, 0.05, 0.1$) on different EEG datasets. The average accuracies of pin-SVM, TWSVM, UTSVM, Pin-GTSVM, and the proposed Tpin-UTWSVM for noise-free data ($r = 0$) are 71.92%, 85.25%, 84.29%, 85.96% and 86.31%, respectively, which clearly depicts the proposed Tpin-UTWSVM model has highest accuracy for $r = 0$ case. For $r = 0.05$ and $r = 0.1$ case, the accuracies for the aforementioned order of models are 71.72%, 85.2%, 84.04%, 85.45%, 86.21% and 69.65%, 85.56%, 82.73%, 85.66%, 86.82%, respectively. Hence, the proposed Tpin-UTWSVM has the best accuracies for different r values. Table 3.31 also shows the average ranks of all the models under comparison. The proposed Tpin-UTWSVM has the lowest rank 2.38, 2.12, 1.86 for $r = 0, 0.05, 0.1$, respectively. In the Table 3.31, the best average values are depicted in bold. Table 3.32 consist of training time and different optimal hyperparameters for the models under study over the 33 EEG datasets and different noise levels. Figure 3.13 pictorially represents the comparison of different models on EEG datasets for three noise levels.

We statistically analyse the results of the models under comparison. We employed Friedman test for $r = 0$ case and calculate $\chi_F^2 = 50.2447$ and $F_F = 19.6537$ from equations (2.56) and (2.57), respectively, for $L = 33$ and $d = 5$. The degree of freedom of F_F is $(4, 128)$ and have critical value $F_{4,128} = 2.4424$ at 5% level of significance. As $F_{(4,128)} < F_F$, thus the models are statistically different. Table 3.33 shows the number of wins, ties and losses of baseline models with respect to the proposed Tpin-UTWSVM over the EEG datasets for different noise levels.

3.2.4.3 AD dataset

AD data is publicly available on ADNI repository which was launched by Michael W. Weiner as Principal Investigator in 2003 and aims to analyse different approaches for neuroimaging such as magnetic resonance imaging (MRI) to diagnose AD from mild cognitive impairment (MCI) stage. The reader can visit www.adni-info.org for further details. The MRI images of brain from ADNI data is shown in Figure 3.14 from 3 different planes. From ADNI database 150 T1-weighted structural MRI images are downloaded. The three categories control normal (CN), mild cognitive impairment (MCI) and AD consist of 50 subjects each in the year range 60 – 90 having 75.83 and 6.07 as the mean and standard deviation, respectively. For extracting the features, we opted the methods mentioned in [144]. The classification performance of the compared models over AD data is obtained in terms of classifying CN versus AD subjects, CN versus MCI subjects and MCI versus AD subjects.

3.2.4.4 Experimental analysis over AD data

The performance of the proposed Tpin-UTWSVM and baseline models pin-SVM [17], TWSVM [22], UTSVM [26], Pin-GTSVM [101] have been evaluated on CN versus AD (CN_vs_AD), CN versus MCI (CN_vs_MCI) and MCI versus AD (MCI_vs_AD) cases for $r = 0, 0.05, 0.1$. The proposed Tpin-UTWSVM model performs better on CN_vs_AD case as compare to the baseline models. Table 3.34 represents the results with optimal hyperparameters of the proposed Tpin-UTWSVM with respect to the baseline models for CN_vs_AD case. It can be seen from Table 3.34 that for noise ratio (r) = 0, 0.05 and 0.1, the proposed Tpin-UTWSVM model has the best performance for the CN_vs_AD case which are 90.32%, 93.55% and 90.32%, respectively.

3.3 Comparison between FP-SVM and Tpin-UTWSVM

This section presents a detailed theoretical comparison between FP-SVM and Tpin-UTWSVM. Both models are grounded in the framework of hyperplane-based classification and aim to improve robustness against noise while enhancing generalization performance. However, they differ in their underlying principles, optimization formulations, and theoretical properties.

- [1] While FP-SVM employs a flexible pinball loss that generalizes the traditional pinball loss [17] by introducing two tunable parameters τ_1 and τ_2 , enabling more flexible asymmetric control over misclassified and correctly classified samples, Tpin-UTWSVM adopts a truncated pinball loss [96] that introduces sparsity by fixing the loss beyond a predefined margin, thereby restricting the influence of samples far from the decision boundary.
- [2] The loss function in FP-SVM ensures convexity under the condition that $\tau_2 \in [-1, 1]$ and $\tau_1 \in [-\tau_2, 1]$, thus guaranteeing the existence of a unique global solution. In contrast, the Tpin-UTWSVM model utilizes a fixed loss due to truncation, which is not convex throughout, though still amenable to efficient optimization techniques.
- [3] While FP-SVM achieves noise insensitivity by penalizing even correctly classified samples through asymmetric loss slopes, Tpin-UTWSVM enhances robustness not only to noise but also to outliers, owing to its truncation mechanism and selective focus on margin-critical instances.
- [4] FP-SVM is formulated as a single hyperplane model based on the traditional SVM framework and solves a single quadratic programming problem (QPP). On the other hand, Tpin-UTWSVM is a twin hyperplane model that solves two smaller subprob-

lems, thereby improving computational efficiency. Moreover, it incorporates universum data, which provides prior information about data distribution and supports better generalization.

- [5] Whereas FP-SVM does not explicitly enforce sparsity, the careful tuning of the τ parameters can indirectly influence the number of support vectors. In contrast, Tpin-UTWSVM promotes sparsity directly by assigning equal penalties to correctly classified samples, effectively reducing the number of active constraints in the optimization process.

3.4 Summary

This chapter introduces two novel classification frameworks designed to enhance noise insensitivity, sparsity, and generalization performance in challenging classification tasks. The first contribution is the flexible pinball loss SVM (FP-SVM) [76], which extends the traditional pinball loss by allowing the parameters $[\tau_1, \tau_2]$ to take values in the range $[-1, 1]$, overcoming the restrictive $[0, 1]$ interval of earlier models. This flexibility enables τ_1 and τ_2 to control penalties for misclassified and correctly classified data points, respectively, improving the model's accuracy. To ensure the convexity of the optimization problem, τ_1 is further constrained to lie between τ_2 and 1. The proposed FP-SVM demonstrates its efficacy in diverse applications, particularly in biomedical domains, through extensive experiments on UCI and KEEL datasets as well as the BreakHis dataset for breast cancer diagnosis. Statistical analysis confirm its superior generalization performance compared to baseline models such as pin-SVM.

The second contribution is the universum twin support vector machine with truncated pinball loss (Tpin-UTWSVM) [82], designed to combine the robustness of pinball loss with

sparsity. The proposed Tpin-UTWSVM leverages universum data to integrate prior knowledge about data distributions, enhancing generalization performance. By incorporating the truncated pinball loss, Tpin-UTWSVM balances noise insensitivity, stability to resampling, and sparsity, while preserving the theoretical properties of the pinball loss. The optimization problem, which is non-convex and non-differentiable, is efficiently solved using the concave-convex procedure (CCCP) [132] and the successive overrelaxation (SOR) technique to minimize computational and memory overhead. Experiments on UCI datasets confirm the superior performance of Tpin-UTWSVM, particularly under noisy conditions, and illustrate the impact of the loss parameters τ and s on sparsity. Further validation on biomedical datasets for Alzheimer's disease detection shows that Tpin-UTWSVM achieves the highest classification accuracy compared to baseline models, demonstrating its potential for real-world medical applications. Together, these advancements underscore the versatility of pinball loss-based frameworks in addressing diverse classification challenges with robustness, efficiency, and adaptability.

In real-world applications, datasets are often plagued by noise, outliers, and class imbalance, which can significantly degrade the performance of classification models. While classical SVMs and TWSVMs are effective, they remain highly sensitive to such data irregularities. The next chapter introduces hyperplane-based classifiers that utilize fuzzy theory and advanced weighting techniques to address these challenges. These methods effectively mitigate the impact of noise and outliers while improving generalization in imbalanced datasets through novel fuzzy membership/weighting functions.

Chapter 4

Fuzzy Weight-Based Robust Classification: Addressing Noise, Outliers, and Class Imbalance

In this chapter, we address the challenges posed by noise, outliers, and class imbalance in classification tasks by developing robust plane-based learning models that leverage fuzzy theory and advanced weighting techniques. We begin with the intuitionistic fuzzy universum support vector machine (IFUSVM) [83], which assigns intuitionistic fuzzy numbers (IFNs) [49] to training samples. By incorporating universum data to encode prior knowledge of the data distribution, IFUSVM reduces the impact of noise and outliers, improving generalization performance. However, IFUSVM's tendency to favor majority-class samples limits its performance on imbalanced datasets, often leading to the misclassification of minority-class samples.

To overcome these limitations, we propose the dual center-based intuitionistic fuzzy support vector machine (DC-IFSVM) [79] and its least-squares variant, dual center-based intuitionistic fuzzy least square twin support vector machine (DC-IFLSTSVM) [79]. These models employ a dual-center weighting mechanism [114], addressing class imbalance by

incorporating class-specific terms into the weighting function. The optimization in DC-IFLSTSVM is solved using the conjugate gradient method, making it scalable to large datasets.

Recognizing the critical role of membership functions in fuzzy-weighting methods, we further propose two novel functions: first, the class probability and generalized bell (CPGB) function and second, class probability and bell-shaped (CPBS) function. Based on these functions, we develop two advanced models: class probability and generalized bell fuzzy twin support vector machine for imbalanced data (CGFTSVM-ID) [84], and bell-shaped fuzzy least square twin support vector machine for imbalanced data (BSFLSTSVM-ID) [85]. CGFTSVM-ID integrates CPGB functions to combine generalized bell metrics, class probabilities, and imbalance ratios. This model mitigates the impact of outliers by assigning lower membership values to noisy samples while prioritizing minority-class samples to improve training. The optimization problem in CGFTSVM-ID is efficiently solved using a successive over-relaxation technique.

Similarly, BSFLSTSVM-ID employs the CPBS function, combining class probabilities, bell-shaped distance metrics, and imbalance ratios. By assigning lower weights to noisy samples and higher weights to minority-class samples, this model achieves enhanced robustness against noise and class imbalance. Its optimization process, involving linear system solutions, is efficiently handled using the conjugate gradient method.

Together, these proposed models demonstrate significant advancements in handling imbalanced datasets with noise and outliers, making them highly effective for real-world applications.

4.1 Proposed intuitionistic fuzzy universum SVM

Building on prior knowledge about data distribution from the work of Weston et al. [25] and aiming to mitigate the influence of noise and outliers as highlighted by Ha et al. [109], this section introduces the intuitionistic fuzzy universum support vector machine (IFUSVM). The proposed IFUSVM incorporates IFNs into the training process while leveraging universum data to enhance classification. By assigning IFNs to the training samples, the model quantifies both the degrees of membership and nonmembership for each data point. Data points with a higher degree of nonmembership are effectively identified as noise, thereby reducing their impact on the classification results. Furthermore, the inclusion of universum data provides valuable prior knowledge about the underlying data distribution, enabling the proposed model to achieve more robust and accurate classification in the presence of noise and outliers.

4.1.1 Formulation of IFUSVM

The proposed IFUSVM contains universum data whose presence leads to prior knowledge about data distribution. The assignment of membership and nonmembership values determine the contribution of each data point for the construction of the hyperplanes, thereby reducing the consequences of noise and outliers. The score values are considered for the training data points only and it differentiates support vectors from noise and outliers. The optimization problem for the proposed IFUSVM is:

$$\begin{aligned}
 & \min_{w,b,\eta,\xi} \quad \frac{1}{2} \|w\|^2 + C_1 \sum_{k=1}^l s_k \eta_k + C_u \sum_{j=1}^{2u} \xi_j \\
 & \text{subject to } y_k(w^t \psi(x_k) + b) \geq 1 - \eta_k, \quad \eta_k \geq 0, \quad \forall k = 1, 2, \dots, l, \\
 & \quad y_j(w^t \psi(x_j^*) + b) \geq -\epsilon - \xi_j, \quad \xi_j \geq 0, \quad \forall j = 1, 2, \dots, 2u,
 \end{aligned} \tag{4.1}$$

where C_1, C_u are the positive regularization parameter. s_k is the score value for the k^{th} data point. η_k and ξ_j are slack variables which are the measure of hinge loss and ϵ -insensitive loss, respectively. The Lagrangian obtained for QPP (4.1) is given by:

$$\begin{aligned} L(w, b, \eta_k, \xi_j, \gamma_k, \beta_j, \alpha_j, \mu_k) = & \frac{1}{2} \|w\|^2 + C_1 \sum_{k=1}^l s_k \eta_k + C_u \sum_{j=1}^{2u} \xi_j + \sum_{k=1}^l \gamma_k (1 - \eta_k - y_k (w^t \psi(x_k) \\ & + b)) + \sum_{j=1}^{2u} \beta_j (-\epsilon - \xi_j - y_j (w^t \psi(x_j^*) + b)) - \sum_{k=1}^l \mu_k \eta_k - \sum_{j=1}^{2u} \alpha_j \eta_j. \end{aligned} \quad (4.2)$$

Here, $\gamma_k, \beta_j, \mu_k, \alpha_j$ are non negative Lagrange multipliers having k in the range 1 to m and j in the range 1 to $2u$. Also we will follow, $x_{m+i} = x_i^*$ with $y_{m+i} = 1$ and $x_{m+u+i} = x_i^*$ with $y_{m+u+i} = -1$ for $i = 1, 2, \dots, u$. The Wolfe dual obtained for QPP (4.1) using the necessary sufficient condition of optimality is the following:

$$\min_{\gamma_k} \frac{1}{2} \sum_{k=1}^{l+2u} \sum_{j=1}^{l+2u} \gamma_k y_k \gamma_j y_j \psi(x_k)^t \psi(x_j) - \sum_{k=1}^{l+2u} \lambda_k \gamma_k$$

subject to $0 \leq \gamma_k \leq C_1 s_k$ for $k = 1, 2, \dots, m$,

$$0 \leq \gamma_k \leq C_u \text{ for } k = l+1, l+2, \dots, l+2u,$$

$$\lambda_k = 1 \text{ for } k = 1, 2, \dots, l,$$

$$\lambda_k = -\epsilon \text{ for } k = l+1, l+2, \dots, l+2u \text{ and } \sum_{k=1}^{l+2u} \gamma_k y_k = 0.$$

Using K.K.T. condition, we obtain the relation

$$w = \sum_{k=1}^{l+2u} \gamma_k y_k \psi(x_k). \quad (4.3)$$

With the help of support vectors, determine b . A test data, say x is assigned to the class depending on the following:

$$f(x) = \text{sign}(w^t \psi(x) + b). \quad (4.4)$$

4.1.2 Experimental results

In this section, we discuss the numerical experiments that are performed to compare the proposed IFUSVM with the baseline models such as classical SVM [90], USVM [25] and KIFSVM [109].

4.1.2.1 Setup of the experiment

All the experimental execution are performed on a PC having MATLAB 2019a, intel(R) core processor and 8 GB RAM. We employed ten-fold cross validation (cv) to compare the performance of the proposed IFUSVM with baseline methods. In ten-fold cv, each dataset is randomly partitioned into ten subsets where one subset is for testing (testing data) and the remaining subsets are for training (training data). We have represented the non-linear case through our experiment by using the Gaussian kernel $\exp(-\|x_1 - x_2\|^2/2\sigma^2)$, where σ denotes the kernel parameter. The range to choose different hyperparameters σ , C_1 , C_u is $[10^{-5}, 10^{-4}, \dots, 10^4, 10^5]$ and range for ϵ is $[0.1, 0.2, 0.3, 0.5, 0.6, 0.7]$. To obtain the optimal parameters, we have used the grid search method [145]. The metrics used for the evaluation of the models are rank and accuracy. Universum data is generated by randomly averaging the data points of both the classes [138]. The binary class datasets for performing the experiments are obtained from UCI repository [146].

4.1.2.2 Results and discussion

For the evaluation of the proposed IFUSVM in comparison to the baseline models, the results are given in Table 4.1 which shows the performance of each model on the testing data of the 17 UCI datasets. From Table 4.1, it is evident that the proposed IFUSVM model has the highest average accuracy. We assigned rank to each model across every dataset depending on their performance. Taking average of the assigned ranks across every dataset, we get average ranks of each model which are 1.41, 1.76, 3.47 and 3.35 for proposed IFUSVM, USVM, KIFSVM and SVM, respectively. The best performing model is the one with smallest rank and vice versa. Thus, the proposed IFUSVM model has best performance. For the analysis of the models, we conducted statistical tests such as Friedman test and Nemenyi post hoc test [123]. Friedman test has the assumption under null hypothesis that all the models have equal average rank. Here, we have $N = 17$ is the number of datasets and $md = 4$ is the number of models. R_i signifies i^{th} classifier's average rank. After calculation from equations (2.56) and (2.57), we get $\chi_F^2 = 34.1608$ and $F_F = 32.4584$. Degree of freedom of F_F statistic is $(3, 48)$. The critical value of $F_{(3,48)} = 2.798$ at 5% level of significance. As $F_F > 2.798$, we reject the null hypothesis. Thus, statistically the models are different. For pairwise comparison of the models, we use the Nemenyi post hoc test. In this test, pairs of models are compared. Models are considered statistically different, if the difference in average ranks of the compared models is greater than the critical difference (cd), where cd is determined by (2.58): having $q_\alpha = 2.569$ for 4 classifiers. On calculating, we get $cd = 1.1376$. The average rank differences of the proposed IFUSVM model with respect to classical SVM, USVM and KIFSVM are 1.94, 0.35 and 2.06, respectively. The average rank differences of the proposed IFUSVM with respect to SVM and KIFSVM are greater than cd. Hence, the proposed method is better than SVM and KIFSVM. With USVM, though the rank difference is less than cd, we can see from the Table 4.1 that the overall rank and

Table 4.1: Classification performance of different models on binary datasets having Gaussian kernel

Dataset	SVM [90] (ACC,Time(s)) (C_1, σ)	USVM [25] (ACC,Time(s)) ($C_1, C_u, \sigma, \epsilon$)	KIFSVM [109] (ACC,Time(s)) (C_1, σ)	Proposed IFUSVM (ACC,Time(s)) ($C_1, C_u, \sigma, \epsilon$)
breast-cancer	(71.98, 0.0164) ($10^4, 0.1$)	(73.05, 0.0781) (1, 10, 0.1, 0.2)	(71.97, 0.0231) ($10^4, 0.1$)	(75.1, 0.0672) ($10^3, 10^2, 0.1, 0.1$)
breast-cancer-wisc-prog	(76.29, 0.0084) ($10^{-5}, 10^{-5}$)	(76.79, 0.0139) (0.1, 10^{-5} , 0.1, 0.3)	(76.29, 0.0107) ($10^{-5}, 10^{-5}$)	(76.79, 0.0145) ($10^3, 10^{-5}, 0.1, 0.3$)
congressional-voting	(62.24, 0.0249) ($10^5, 0.1$)	(62.49, 0.0607) (0.1, 10^{-5} , 1, 0.1)	(61.56, 0.0332) ($10^5, 0.1$)	(62.73, 0.1184) (10, 1, 0.1, 0.6)
echocardiogram	(74.78, 0.0073) ($10^4, 0.1$)	(79.34, 0.0228) (0.1, $10^5, 0.1, 0.5$)	(74.78, 0.0055) ($10^4, 0.1$)	(79.34, 0.0232) ($10^3, 10^5, 0.1, 0.1$)
fertility	(88, 0.0068) ($10^3, 1$)	(89, 0.0039) (10, 0.1, 1, 0.1)	(90, 0.0061) ($10^3, 1$)	(91, 0.0057) ($10^3, 1, 1, 0.1$)
heart-hungarian	(82.63, 0.0129) ($10^4, 0.1$)	(84.33, 0.0798) (0.1, 1, 0.1, 0.5)	(82.63, 0.0133) ($10^4, 0.1$)	(84.33, 0.0841) ($10^3, 10^2, 0.1, 0.3$)
hepatitis	(83.96, 0.0062) ($10^5, 0.1$)	(85.25, 0.0077) (1, 0.01, 0.1, 0.2)	(83.96, 0.0088) ($10^5, 0.1$)	(85.25, 0.0114) ($10^3, 10^3, 0.1, 0.1$)
molec-biol-promoter	(76.55, 0.0073) ($10^4, 0.1$)	(82.27, 0.0067) (0.1, $10^3, 0.1, 0.6$)	(76.55, 0.0059) ($10^4, 0.1$)	(82.27, 0.0055) ($10^{-5}, 10, 0.1, 0.5$)
parkinsons	(91.39, 0.0059) ($10^5, 1$)	(92.87, 0.0125) (10, $10^{-5}, 1, 0.1$)	(91.39, 0.0107) ($10^5, 1$)	(92.87, 0.0123) ($10^5, 10^{-5}, 1, 0.1$)
pittsburg-bridges-T-OR-D	(87.18, 0.0052) ($10^5, 0.1$)	(88.09, 0.0102) (10, 0.1, 0.1, 0.1)	(87.18, 0.0069) ($10^5, 0.1$)	(88.09, 0.009) ($10^5, 1, 0.1, 0.5$)
spect	(69.17, 0.013) ($10^4, 0.1$)	(70.66, 0.0928) (0.1, 0.01, 0.1, 0.3)	(68.4, 0.0151) ($10^4, 0.1$)	(72.89, 0.0604) ($10^3, 1, 0.1, 0.1$)
trains	(60, 0.0051) ($10^4, 0.1$)	($10^2, 0.006$) (0.01, 10, $10^{-4}, 0.5$)	(60, 0.0049) ($10^4, 0.1$)	($10^2, 0.0049$) ($10^{-5}, 10^{-3}, 10^{-4}, 0.2$)
vertebral-column-2clases	(83.87, 0.0121) ($10^5, 1$)	(84.84, 0.022) (1, 1, 1, 0.1)	(83.23, 0.0156) ($10^5, 1$)	(85.81, 0.0252) (10, 10, 1, 0.6)
heart-switzerland	(60.77, 0.005) ($10^{-5}, 10^{-5}$)	(64.1, 0.0071) ($10^4, 10^4, 10^5, 0.6$)	(60.77, 0.0065) ($10^{-5}, 10^{-5}$)	(64.1, 0.0184) ($10^{-4}, 10^{-3}, 0.1, 0.5$)
ilpd-indian-liver	(72.04, 0.0491) (1, 10^2)	(72.04, 0.1695) (1, $10^{-5}, 10^2, 0.1$)	(72.04, 0.0648) (1, 10^2)	(72.04, 0.1279) (1, $10^{-5}, 10^2, 0.1$)
ionosphere	(87.16, 0.0164) ($10^5, 1$)	(88.87, 0.068) (1, $10^2, 1, 0.2$)	(87.16, 0.0198) ($10^5, 1$)	(88.87, 0.0456) ($10^4, 10, 1, 0.5$)
pima	(75, 0.0835) ($10^4, 0.1$)	(75.52, 0.3994) (0.1, $10^5, 0.1, 0.5$)	(75, 0.0962) ($10^4, 0.1$)	(75.52, 0.9315) ($10^3, 10^5, 0.1, 0.5$)
Average Accuracy	76.65	80.56	76.64	81
Average Rank	3.35	1.76	3.47	1.41

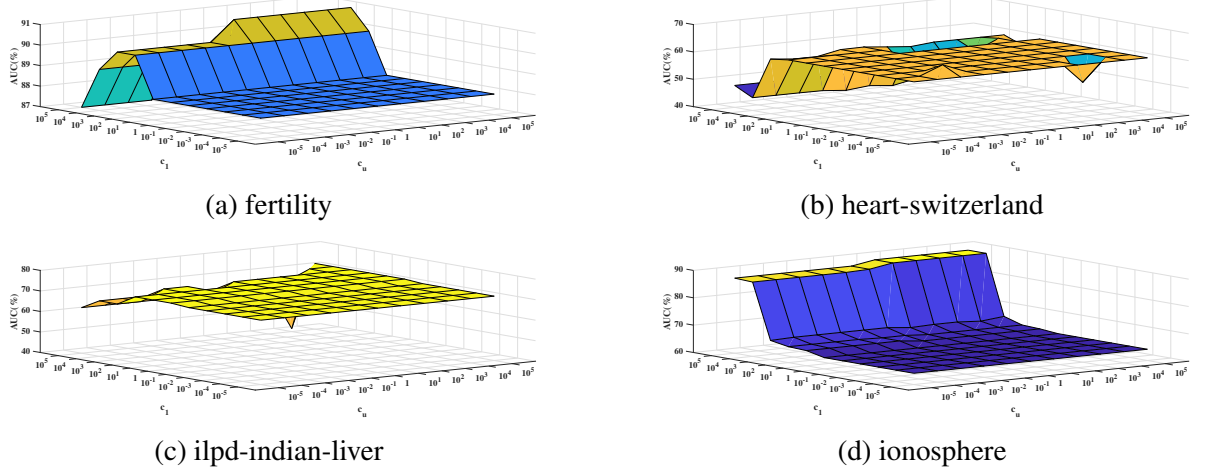


Figure 4.1: Sensitivity evaluation of the proposed IFUSVM classifier on UCI binary datasets using user-specified parameters and Gaussian kernel.

Table 4.2: Win-tie-loss of the models in pair with proposed IFUSVM.

	SVM [90]	KIFSVM [109]	USVM [25]
KIFSVM	[1, 12, 4]		
USVM	[16,1,0]	[15,1,1]	
IFUSVM	[16,1,0]	[16,1,0]	[5,12,0]

Here, $[p, q, r]$ means p times wins, q times ties and r times losses of the row method over the column method

accuracy are better for the proposed model. Further, we employed the win-tie-loss sign test [123]. Table 4.2 represents the win-tie-loss of baseline models in pair with the proposed IFUSVM. For this case, we get, if number of wins of a model in a pair is greater than 12.54 following from (2.59), then the null hypothesis fails. Thus, the proposed IFUSVM model performs comparatively better than SVM and KIFSVM. Figure 4.1 demonstrates sensitivity curve of the proposed IFUSVM classifier on UCI binary datasets using user-specified parameters and Gaussian kernel.

IFUSVM introduces IFNs to reduce the influence of noisy points and integrates univsum data to generalize decision boundaries. While highly robust, it does not directly address class imbalance and involves higher computational complexity.

4.2 Proposed dual center-based intuitionistic fuzzy models

Building on the enhanced generalization capabilities of DC-IF over IF and the effectiveness of class-specific weighting for imbalanced data [147], in this section, we propose the dual center-based intuitionistic fuzzy support vector machine (DC-IFSVM) and its least squares variant, DC-IFLSTSVM, designed for large-scale datasets. These models effectively address the challenges posed by noise, outliers, and class imbalance by incorporating class-specific terms in the weighting mechanism. The DC-IFLSTSVM has enhanced computational efficiency by solving the optimization problem using conjugate gradient method.

4.2.1 Formulation of dual center based intuitionistic fuzzy classifiers

The IF scheme in incorporation with TWSVM [51] and LSTSVM [110], encounters limitations as discussed in 2.3.4. Moreover, the DC-IFTBLDM [114] exhibits high complexity and does not tackle the problem of CIL. To overcome the influence of noise, outliers in the training data and the issue of CIL, we propose dual center-based intuitionistic fuzzy support vector machine (DC-IFSVM). However, DC-IFSVM solves a QPP which has high computational burden. To address the complexity issue along with the negative impact of impure data points, we propose dual center-based intuitionistic fuzzy least square twin support vector machine for large-scale problems (DC-IFLSTSVM). The DC-IFLSTSVM uses the DC-IF membership function for assigning weights to the data points, thus diminishing the effect of unwanted points. Further, the proposed DC-IFLSTSVM addresses the CIL problem by using the class-specific weighting scheme [147]. Thus, the weights assigned to

the training data points can be expressed as:

$$\kappa_+(x_j) = s(x_j) \left(\frac{l_2}{l} \right), \text{ if } y_j = 1, \quad (4.5)$$

$$\kappa_-(x_j) = s(x_j) \left(\frac{l_1}{l} \right), \text{ if } y_j = -1, \quad (4.6)$$

where $s(x_j)$ denote the DC-IF score value for j^{th} data point obtained by equation (2.37).

Further, we use

$$\kappa_j = \begin{cases} \kappa_+(x_j), & \text{if } y_j = 1, \\ \kappa_-(x_j), & \text{if } y_j = -1. \end{cases} \quad (4.7)$$

4.2.1.1 Linear case

For linearly separable data points, linear cases of the proposed models are employed.

4.2.1.1.1 DC-IFSVM

The equation of the hyperplane for linear DC-IFSVM is given by $\mathcal{G}(x) := w^t x + b = 0$,

where $w \in \mathbb{R}^n$. The optimization problem is given by:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|w\|^2 + C \sum_{j=1}^l \kappa_j \rho_j \\ \text{subject to} \quad & y_j(w^t x_j + b) + \rho_j \geq 1, \rho_j \geq 0, \end{aligned} \quad (4.8)$$

where C is the positive regularization parameter and κ_j corresponds to the weight assigned using equation (4.7).

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l y_j y_k x_j^t x_k \alpha_j \alpha_k - \sum_{j=1}^l \alpha_j \\ \text{subject to} \quad & \sum_{j=1}^l \alpha_j y_j = 0, \quad 0 \leq \alpha_j \leq \kappa_j \rho_j, \quad j = 1, 2, \dots, l, \end{aligned} \quad (4.9)$$

where α_j is positive Lagrange multiplier. After the decision surface is learned, the label of test data point is given as:

$$\mathcal{F}_{svm}(x) = \text{sign} \left(\sum_{j=1}^l y_j \alpha_j x_j^t x + b \right). \quad (4.10)$$

4.2.1.1.2 DC-IFLSTSVM

The equations of the non-parallel hyperplanes $\mathcal{G}_1(x)$ and $\mathcal{G}_2(x)$ is given by $w_1^t x + b_1 = 0$ and $w_2^t x + b_2 = 0$, respectively, where w_1, w_2 are the vectors in n -dimensional space, b_1, b_2 are scalars. The optimization problem of the linear case of DC-IFLSTSVM can be written as:

$$\begin{aligned} \min_{w_1, b_1, \rho_1} & \frac{C_1}{2} (\|w_1\|^2 + b_1^2) + \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_2 \|S_2 \rho_1\|^2 \\ \text{subject to} & -(M_2 w_1 + e_2 b_1) + \rho_1 = e_2 \end{aligned} \quad (4.11)$$

and

$$\begin{aligned} \min_{w_2, b_2, \rho_2} & \frac{C_3}{2} (\|w_2\|^2 + b_2^2) + \frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_4 \|S_1 \rho_2\|^2 \\ \text{subject to} & (M_1 w_2 + e_1 b_2) + \rho_2 = e_1, \end{aligned} \quad (4.12)$$

where C_k ($k = 1, 2, 3, 4$) denotes positive regularization constant. S_1 and S_2 represent diagonal matrices of positive and negative class data weights, respectively, obtained from equations (4.5) and (4.6). In the optimization problems (4.11) and (4.12), the first term corresponds to the regularization term, which enforces the SRM principle in the proposed DC-IFLSTSVM. The second term aims to minimize the squared distance of data points from their corresponding hyperplanes. The third term corresponds to the squared sum of weighted slack variables.

We illustrate the solving procedure for optimization problem (4.11), same approach can

be extrapolated for optimization problem (4.12). By substituting the constraint of problem (4.11) into the objective function, we obtain the Lagrangian as follows:

$$\mathcal{L} = \frac{C_1}{2}(\|w_1\|^2 + b_1^2) + \frac{1}{2}\|M_1 w_1 + e_1 b_1\|^2 + C_2\|S_2(e_2 + (M_2 w_1 + e_2 b_1))\|^2. \quad (4.13)$$

Partially differentiating equation (4.13) with respect to w_1 and b_1 , we obtain,

$$C_1 w_1 + M_1^t(M_1 w_1 + e_1 b_1) + C_2(S_2 M_2)^t(S_2(M_2 w_1 + e_2 b_1 + e_2)) = 0, \quad (4.14)$$

$$C_1 b_1 + e_1^t(M_1 w_1 + e_1 b_1) + C_2(S_2 e_2)^t(S_2(M_2 w_1 + e_2 b_1 + e_2)) = 0. \quad (4.15)$$

Assuming $M = \begin{bmatrix} M_1 & e_1 \end{bmatrix}$, $N = \begin{bmatrix} S_2 M_2 & S_2 e_2 \end{bmatrix}$ and $v_1 = \begin{bmatrix} w_1 \\ b_1 \end{bmatrix}$ and rewriting the equations (4.14) and (4.15) together, we obtain

$$-(M^t M + C_1 I + C_2 N^t N)^{-1} v_1 = C_2 N^t S_2 e_2 \quad (4.16)$$

$$\implies v_1 = -(M^t M + C_1 I + C_2 N^t N)^{-1} C_2 N^t S_2 e_2, \quad (4.17)$$

For problem (4.12), assuming $v_2 = \begin{bmatrix} w_2 \\ b_2 \end{bmatrix}$, $N = \begin{bmatrix} M_2 & e_2 \end{bmatrix}$, $M = \begin{bmatrix} S_1 M_1 & S_1 e_1 \end{bmatrix}$, we obtain

$$(N^t N + C_3 I + C_4 M^t M)^{-1} v_2 = C_4 M^t S_1 e_1 \quad (4.18)$$

$$\implies v_2 = (N^t N + C_3 I + C_4 M^t M)^{-1} C_4 M^t S_1 e_1. \quad (4.19)$$

Solving equations (4.17) and (4.19) results in the hyperplane parameters, i.e., w_1 , b_1 and w_2 , b_2 , respectively. The label for a test data point is given by the function expressed as:

$$\operatorname{argmin}_{k=1,2} \mathcal{F}_k(x), \text{ where } \mathcal{F}_k(x) = \frac{|u_k^t x + b_k|}{\|u_k\|}. \quad (4.20)$$

4.2.1.2 Non-linear case

To classify the linearly inseparable data points, the kernel trick is employed which transforms the data points to kernel space. Let ψ be the function which maps the data points, then $K(x_1, x_2) = \psi(x_1)^t \psi(x_2)$.

4.2.1.2.1 Non-linear DC-IFSVM

The equation of non-linear surfaces of DC-IFSVM generated by the kernel is given as $\mathcal{G}(x) := K(x, D^t)z + b = 0$, where z corresponds to the weight vector and $b \in \mathbb{R}$. The optimization problem for DC-IFSVM to obtain the hyperplane $\mathcal{F}(x)$ is expressed as:

$$\begin{aligned} \min_{z, b} \quad & \frac{1}{2} \|z\|^2 + C \sum_{j=1}^l \kappa_j \rho_j \\ \text{subject to} \quad & y_j(z^t \psi(x_j) + b) + \rho_j \geq 1, \quad \rho_j \geq 0, \quad j = 1, 2, \dots, l. \end{aligned} \quad (4.21)$$

The dual for the problem (4.21) using K.K.T. is obtained as:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{j=1}^l \sum_{k=1}^l y_j y_k K(x_j, x_k) \alpha_j \alpha_k - \sum_{j=1}^l \alpha_j \\ \text{subject to} \quad & \sum_{j=1}^l \alpha_j y_j = 0, \quad 0 \leq \alpha_j \leq \kappa_j \rho_j, \quad j = 1, 2, \dots, l, \end{aligned} \quad (4.22)$$

where α_j is positive Lagrange multiplier. For a test data point, the decision is given as

$$\mathcal{F}_{svm}(x) = \text{sign} \left(\sum_{j=1}^l y_j \alpha_j K(x_j, x) + b \right). \quad (4.23)$$

4.2.1.2.2 Non-linear DC-IFLSTSVM

The equation of non-linear surfaces of DC-IFLSTSVM generated by the kernel are given as $\mathcal{G}(x) := K(x, D^t)z_1 + b_1 = 0$ and $\mathcal{G}(x) := K(x, D^t)z_2 + b_2 = 0$, where z_1 and z_2 are the

weight vectors, and $D = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$. The optimization problems are as follows:

$$\begin{aligned} \min_{z_1, b_1, \xi_1} & \frac{C_1}{2} (\|z_1\|^2 + b_1^2) + \frac{1}{2} \|K(M_1, D^t)z_1 + e_1 b_1\|^2 + C_2 \|S_2 \rho_1\|^2 \\ \text{subject to} & -(K(M_2, D^t)z_1 + e_2 b_1) + \rho_1 = e_2 \end{aligned} \quad (4.24)$$

and

$$\begin{aligned} \min_{z_2, b_2, \rho_2} & \frac{C_3}{2} (\|z_2\|^2 + b_2^2) + \frac{1}{2} \|K(M_2, D^t)z_2 + e_2 b_2\|^2 + C_4 \|S_1 \rho_2\|^2 \\ \text{subject to} & (K(M_1, D^t)z_2 + e_1 b_2) + \rho_2 = e_1. \end{aligned} \quad (4.25)$$

Here, the terms have similar meanings as explained for the optimization problem (4.11) of the linear case. Solving in a similar way as that of linear case, and assuming $M = \begin{bmatrix} K(M_1, D^t) & e_1 \end{bmatrix}$, $N = \begin{bmatrix} S_2 K(M_2, D^t) & S_2 e_2 \end{bmatrix}$, we obtain

$$w_1 = -(M^t M + C_1 I + C_2 N^t N)^{-1} C_2 N^t S_2 e_2 \quad (4.26)$$

and by assuming $N = \begin{bmatrix} K(M_2, D^t) & e_2 \end{bmatrix}$, $M = \begin{bmatrix} S_1 K(M_1, D^t) & S_1 e_1 \end{bmatrix}$, we get

$$w_2 = (N^t N + C_3 I + C_4 M^t M)^{-1} C_4 M^t S_1 e_1, \quad (4.27)$$

where $w_1 = \begin{bmatrix} z_1 \\ b_1 \end{bmatrix}$, $w_2 = \begin{bmatrix} z_2 \\ b_2 \end{bmatrix}$. The decision of a test data point of its belongingness to a class is given by the function

$$\operatorname{argmin}_{k=1,2} \mathcal{F}_k(x), \text{ where } \mathcal{F}_k(x) = \frac{|K(x, D^t)z_k + b_k|}{\sqrt{z_k^t K(D, D^t) z_k}}. \quad (4.28)$$

4.2.1.3 Optimization approach: Solving the proposed optimization problems

The optimization problems (4.9) and (4.22) for the linear and non-linear cases of DC-IFSVM are QPPs. Solving QPPs has high computational burden. To handle the computational complexity, the proposed DC-IFLSTSVM solves system of linear equations which involves the matrix inversion. The complexity of matrix inversion is $O(l^3)$ [133], for a matrix of order $l \times l$, which is also very high. To handle the complexity of matrix inversion, the conjugate gradient (CG) method is employed, which solves the system of linear equations of the form $At = b$, where A denote the matrix, t and b are the vectors.

Theorem 2. *For any initial point $t_0 \in \mathbb{R}^m$, the sequence $\{t_k\}$ CG method generates, converges to the solution t^* of the system of equations $At = b$. [148]*

4.2.1.4 Time complexity

The proposed DC-IFSVM and DC-IFLSTSVM impose weights on the training data points through equations (4.5) and (4.6), which has a time complexity of $O(l)$ [114]. In addition to the weight assignment, the proposed models (DC-IFSVM and DC-IFLSTSVM) solve optimization problems. DC-IFSVM solves a QPP which has the complexity of $O(l^3)$ [149]. Thus, the overall complexity of the proposed DC-IFSVM is $O(l^3) + O(l)$. Assuming $l_1 \approx l_2 \approx l/2$, the time complexity of CG method to solve the optimization problem of the proposed DC-IFLSTSVM is $O(r\sqrt{\kappa})$ [150], r is the non-zero entries of matrix and κ corresponds to the condition number of the matrix. Thus, the overall complexity of the proposed DC-IFLSTSVM is $O(r\sqrt{\kappa}) + O(l)$.

4.2.2 Numerical experiments

In this section, we discuss the numerical experiments performed to demonstrate the performance of the proposed DC-IFSVM and DC-IFLSTSVM over UCI and KEEL datasets.

Table 4.3: Abbreviation (abbre.) of datasets.

KEEL Dataset	abbre.*	UCI Dataset	abbre.*
aus	k1	blood	d1
cmc	k2	breast_cancer_wisc	d2
ecoli-0-6-7_vs_5	k3	breast_cancer_wisc_prog	d3
heart-stat	k4	congressional_voting	d4
iono	k5	credit_approval	d5
monk1	k6	echocardiogram	d6
monk2	k7	haberman_survival	d7
monk3	k8	heart_hungarian	d8
pima	k9	hepatitis	d9
shuttle-c0-vs-c4	k10	horse_colic	d10
transfusion	k11	ionosphere	d11
yeast-0-2-5-6_vs_3-7-8-9	k12	monks_1	d12
yeast-0-2-5-7-9_vs_3-6-8	k13	monks_3	d13
yeast-2_vs_4	k14	musk_1	d14
yeast1vs7	k15	pima	d15
yeast2vs8	k16	pittsburg_bridges.T.OR.D	d16
yeast5	k17	spect	d17
bupa-or-liver-disorders	k18	statlog_australian_credit	d18
ecoli-0-1-4-7_vs_5-6	k19	titanic	d19
ecoli-0-1_vs_5	k20	BCW	d20
ecoli-0-2-6-7_vs_3-5	k21	banknote_processed	d21
ecoli-0-4-6_vs_5	k22	ringnorm	d22
ecoli-0-6-7_vs_3-5	k23	twonorm	d23
ecoli0137vs26	k24	magic	d24
ecoli01vs5	k25	adult	d25
ecoli3	k26		
glass4	k27		
haber	k28		
new-thyroid1	k29		

Table 4.4: AUC with the time required for training of the proposed DC-IFSVM, DC-IFLSTSV and baseline models over KEEL datasets with RBF kernel.

Dataset (ir)	(Samples,Dim)	TWSVM [22] (AUC,time(sec))	KIFSVM [109] (AUC,time(sec))	RFLSTSV-CIL [151] (AUC,time(sec))	IIFTWSVM [50] (AUC,time(sec))	IFTWSVM-ID [51] (AUC,time(sec))	DC-IFTBLDM [114] (AUC,time(sec))	DC-IFSVM (AUC,time(sec))	DC-IFLSTSV (AUC,time(sec))
k1 (1.25)	(690, 14)	(0.8687, 0.0229)	(0.8754, 0.1293)	(0.8763, 0.0276)	(0.8584, 0.0605)	(0.8745, 0.043)	(0.8921, 0.2541)	(0.8754, 0.083)	(0.8933, 0.0113)
k2 (1.34)	(1473, 9)	(0.6536, 0.113)	(0.6287, 0.5414)	(0.697, 0.1567)	(0.6248, 0.2579)	(0.6557, 0.1102)	(0.6651, 0.6724)	(0.6353, 0.6492)	(0.7073, 0.2788)
k3 (10)	(222, 7)	(0.7273, 0.0021)	(0.8636, 0.0145)	(0.6818, 0.0035)	(0.7727, 0.0072)	(0.8182, 0.0058)	(0.8364, 0.033)	(0.8818, 0.0092)	(0.8091, 0.0021)
k4 (1.25)	(270, 13)	(0.8702, 0.0023)	(0.8382, 0.0172)	(0.8158, 0.004)	(0.8416, 0.0219)	(0.8165, 0.0051)	(0.8484, 0.0252)	(0.8382, 0.0068)	(0.8668, 0.0004)
k5 (1.79)	(351, 33)	(0.9312, 0.0036)	(0.9062, 0.0565)	(0.9079, 0.0057)	(0.8543, 0.0236)	(0.9312, 0.008)	(0.8543, 0.1263)	(0.9138, 0.0238)	(0.9155, 0.0046)
k6 (1.04)	(556, 6)	(0.4675, 0.0084)	(0.4623, 0.0226)	(0.5, 0.0143)	(0.4654, 0.034)	(0.4599, 0.027)	(0.5, 0.1002)	(0.4467, 0.0117)	(0.5027, 0.0026)
k7 (1.92)	(601, 7)	(0.9343, 0.0247)	(0.93, 0.1726)	(0.9254, 0.0193)	(0.7336, 0.0394)	(0.9032, 0.0168)	(0.9194, 0.0766)	(0.93, 0.1003)	(0.8893, 0.0526)
k8 (1.01)	(556, 6)	(0.4726, 0.0219)	(0.5, 0.0296)	(0.5, 0.0126)	(0.4454, 0.0556)	(0.4375, 0.0271)	(0.5, 0.124)	(0.5, 0.0111)	(0.5115, 0.0019)
k9 (1.87)	(768, 8)	(0.7365, 0.0388)	(0.707, 0.2075)	(0.7273, 0.0253)	(0.5146, 0.0629)	(0.7428, 0.0262)	(0.7242, 0.1442)	(0.7434, 0.1349)	(0.7402, 0.0783)
k10 (13.87)	(1829, 9)	(1, 0.1307)	(0.9833, 0.9789)	(0.9833, 0.2794)	(0.9833, 0.4632)	(1, 0.2049)	(0.9833, 1.9569)	(0.9833, 0.6125)	(1, 0.0493)
k11 (3.2)	(748, 4)	(0.7062, 0.032)	(0.7001, 0.1102)	(0.6755, 0.0352)	(0.6128, 0.11)	(0.655, 0.0289)	(0.6643, 0.0616)	(0.6652, 0.1477)	(0.7072, 0.013)
k12 (9.14)	(1004, 8)	(0.7408, 0.0325)	(0.7793, 0.4865)	(0.7651, 0.0616)	(0.6712, 0.1029)	(0.772, 0.0797)	(0.7244, 0.398)	(0.7633, 0.2834)	(0.7898, 0.0117)
k13 (9.14)	(1004, 8)	(0.8574, 0.0584)	(0.8946, 0.4157)	(0.889, 0.0628)	(0.7815, 0.1778)	(0.9223, 0.0815)	(0.9223, 0.2131)	(0.8946, 0.2437)	(0.9075, 0.1136)
k14 (9.08)	(2968, 8)	(0.8346, 0.0073)	(0.8195, 0.0834)	(0.8271, 0.0093)	(0.6867, 0.0338)	(0.8709, 0.0259)	(0.782, 0.087)	(0.8195, 0.033)	(0.8221, 0.003)
k15 (14.3)	(459, 9)	(0.627, 0.016)	(0.8036, 0.1225)	(0.8115, 0.0091)	(0.5714, 0.0319)	(0.75, 0.02)	(0.5337, 0.203)	(0.8036, 0.0647)	(0.6429, 0.0008)
k16 (23.15)	(483, 8)	(0.625, 0.009)	(0.625, 0.045)	(0.6862, 0.0135)	(0.711, 0.0257)	(0.6108, 0.0182)	(0.6179, 0.1676)	(0.625, 0.0272)	(0.7358, 0.0017)
k17 (32.73)	(1484, 8)	(0.9919, 0.0972)	(0.9584, 0.151)	(0.97, 0.1578)	(0.8326, 0.2651)	(0.9445, 0.1434)	(0.9804, 1.2086)	(0.9596, 0.385)	(0.9746, 0.0568)
k18 (1.38)	(345, 6)	(0.6478, 0.0248)	(0.6411, 0.1635)	(0.7296, 0.022)	(0.5981, 0.0582)	(0.7174, 0.076)	(0.6911, 0.2803)	(0.6441, 0.0618)	(0.7248, 0.0076)
k19 (12.28)	(332, 6)	(0.928, 0.0065)	(0.917, 0.0615)	(0.9225, 0.0098)	(0.8004, 0.0205)	(0.928, 0.0109)	(0.9115, 0.2305)	(0.9115, 0.0305)	(0.9335, 0.0038)
k20 (11)	(240, 6)	(0.9929, 0.0019)	(0.9714, 0.0246)	(0.9786, 0.0036)	(0.9714, 0.0144)	(0.95, 0.013)	(0.9786, 0.0267)	(0.9714, 0.0163)	(0.9857, 0.0012)
k21 (9.18)	(224, 7)	(0.8125, 0.0037)	(0.8665, 0.0395)	(0.8665, 0.0035)	(0.7415, 0.0127)	(0.8125, 0.0069)	(0.9831, 0.0244)	(0.8665, 0.0168)	(0.9492, 0.0005)
k22 (9.15)	(203, 6)	(0.8894, 0.0015)	(1, 0.0257)	(0.9909, 0.0032)	(0.8333, 0.0084)	(0.8333, 0.0059)	(0.9909, 0.0237)	(1, 0.0122)	(1, 0.0015)
k23 (9.09)	(222, 7)	(0.6729, 0.0014)	(0.819, 0.0208)	(0.7914, 0.0033)	(0.7459, 0.0109)	(0.8823, 0.0046)	(0.8458, 0.0318)	(0.819, 0.0137)	(0.8734, 0.0006)
k24 (4.76)	(311, 7)	(0.862, 0.0066)	(0.8998, 0.0385)	(0.862, 0.0043)	(0.8933, 0.022)	(0.931, 0.0129)	(0.8308, 0.5082)	(0.8933, 0.0245)	(0.9298, 0.0006)
k25 (11)	(240, 7)	(0.9922, 0.0024)	(0.9141, 0.0223)	(0.9375, 0.0033)	(0.9688, 0.0085)	(0.875, 0.0041)	(0.9922, 0.052)	(0.9141, 0.01)	(0.9766, 0.0009)
k26 (8.6)	(336, 7)	(0.8883, 0.0105)	(0.9368, 0.0297)	(0.8998, 0.0056)	(0.8756, 0.0208)	(0.8998, 0.0113)	(0.8998, 0.0759)	(0.931, 0.0331)	(0.954, 0.0009)
k27 (15.46)	(214, 9)	(0.6421, 0.0024)	(0.6421, 0.0178)	(0.8005, 0.0032)	(0.6011, 0.0098)	(0.8087, 0.0034)	(0.6175, 0.0164)	(0.6421, 0.0122)	(0.776, 0.0005)
k28 (2.78)	(306, 3)	(0.6516, 0.0048)	(0.5248, 0.0445)	(0.6547, 0.0045)	(0.5, 0.0121)	(0.7162, 0.0044)	(0.6778, 0.0495)	(0.6333, 0.0198)	(0.6593, 0.001)
k29 (5.14)	(215, 5)	(1, 0.0024)	(1, 0.0267)	(1, 0.0031)	(1, 0.0139)	(1, 0.0041)	(0.9904, 0.0313)	(1, 0.0166)	(1, 0.0008)
Avg AUC		0.79	0.81	0.82	0.74	0.81	0.81	0.81	0.83
Avg rank		4.33	4.64	4.19	6.81	4.4	4.62	4.57	2.45

The baseline models considered are KIFSVM [109], TWSVM [22], RFLSTSVM-CIL [151], IIFTWSVM [50], IFTWSVM-ID [51], and DC-IFTBLDM [114]. We conduct the experiments in MATLAB 2022b on a desktop PC with 11th Gen Intel(R) Core(TM), 2.50GHz processor and 16 GB RAM. The datasets are split into a 70:30 ratio for training and testing, respectively. Within the 70% training data, 10-fold cross-validation is applied alongside grid search to determine optimal parameters. Using the optimal parameters, the testing performance is calculated over 30% data. The ranges chosen for different parameters are as follows: $C, C_1, C_2 \in \{10^{-5}, \dots, 10^5\}$; $\sigma \in \{2^{-5}, \dots, 2^5\}$, where σ denotes RBF kernel parameter ($\exp\|x_i - x_j\|/2\sigma^2$). The metrics reported to evaluate the performance is $AUC = \frac{1+tp/(tp+fn)-fp/(fp+tn)}{2}$ with training time in seconds (sec). Here, tp , fp denote true positive and false positive; tn , fn denote true negative and false negative. The number of maximum iterations for CG method is 100. Also, we considered $\lambda = 2$ in the weight function (2.37) [114]. Table 4.3 provides the abbreviation of the dataset names used in experiments.

4.2.2.1 Experimental evaluation over KEEL datasets

To assess the performance of the proposed DC-IFSVM, DC-IFLSTSVM and baseline models, we conducted experiments over 29 KEEL datasets. Table 4.4 presents data details, containing number of samples, features, and imbalance ratio, as well as the models' performance and training time (in seconds). The average AUC of the proposed DC-IFLSTSVM is 0.83, surpassing the baseline models and proposed DC-IFSVM. The average AUC of the other models in decreasing order are as follows: RFLSTSVM-CIL with 0.82, proposed DC-IFSVM, IFTWSVM-ID, DC-IFTBLDM and KIFSVM have same average AUC value of 0.81, TWSVM has average AUC of 0.79 followed by IIFTWSVM with average AUC of 0.74. However, the average AUC values may not reliably reflect a model's performance,

as a smaller value over one dataset can be compensated by a greater AUC over another. Therefore, we considered ranks of the models across each dataset, where a lower rank indicates better performance. The average rank of the proposed DC-IFLSTSVM is 2.45, which is the least among the models. The average ranks of the other models are as: 4.19 for RFLSTSVM-CIL, 4.33 for TWSVM, 4.4 for IFTWSVM-ID, 4.57 for proposed DC-IFSVM, 4.62 for DC-IFTBLDM, 4.64 for KIFSVM and 6.81 for DC-IFTBLDM.

Further, we conduct experiments on UCI datasets with the results on each dataset are presented in Table 4.5. Moreover, the average AUC and average rank shown in Table 4.5, demonstrate the proposed DC-IFLSTSVM has average AUC and average rank of 0.81 and 3.12, respectively. The average AUC of the proposed DC-IFSVM is 0.79, which is less than IFTWSVM-ID having 0.8. DC-IFTBLDM and RFLSTSVM-CIL have an average AUC of 0.79, followed by TWSVM, KIFSVM and IIFTWSVM with average AUCs 0.78, 0.76 and 0.73, respectively.

Thus, the least rank and greatest AUC of the proposed DC-IFLSTSVM signifies its superior performance over baseline and proposed DC-IFSVM. The proposed DC-IFSVM incorporates weights for CIL, while DC-IFTBLDM does not employ a CIL strategy. Interestingly, despite the absence of a CIL strategy in DC-IFTBLDM, the comparable performance of DC-IFSVM with DC-IFTBLDM empirically validates the theoretical analysis presented in [152].

4.2.2.2 Statistical analysis

In this section, we employ statistical tests to evaluate the performance of the models. We employ Friedman test and Wilcoxon signed rank test [123].

According to Friedman test, we assume the null hypothesis that all models are equivalent. The Friedman statistic is calculated using equation (2.56). To overcome the

Table 4.5: AUC with the time required for training of the proposed DC-IFSVM, DC-IFLSTSVM and baseline models over UCI datasets with RBF kernel.

Dataset	(Samples,Dim)	TWSVM [22] (AUC,time(sec))	KIFSVM [109] (AUC,time(sec))	RFLSTSVM-CIL [151] (AUC,time(sec))	IIFTWSVM [50] (AUC,time(sec))	IFTWSVM-ID [51] (AUC,time(sec))	DC-IFTBLDM [114] (AUC,time(sec))	DC-IFSVM (AUC,time(sec))	DC-IFLSTSVM (AUC,time(sec))
d1 (3.2)	(748, 4)	(0.7062, 0.0362)	(0.7001, 0.1727)	(0.6755, 0.0362)	(0.6128, 0.1791)	(0.655, 0.0455)	(0.6643, 0.6029)	(0.6652, 0.1681)	(0.7072 , 0.0139)
d2 (1.9)	(699, 9)	(0.9655, 0.0178)	(0.9662, 0.3709)	(0.9785, 0.0351)	(0.9607, 0.0631)	(0.9525, 0.0322)	(0.9812 , 0.2847)	(0.9699, 0.1264)	(0.9812 , 0.0073)
d3 (3.21)	(198, 33)	(0.5995 , 0.0031)	(0.5301, 0.0139)	(0.5452, 0.0046)	(0.5059, 0.0087)	(0.5008, 0.0047)	(0.49, 0.0244)	(0.5192, 0.0042)	(0.4356, 0.0007)
d4 (1.59)	(435, 16)	(0.5953, 0.0099)	(0.5, 0.0439)	(0.618, 0.0104)	(0.5678, 0.0186)	(0.6356 , 0.0092)	(0.6248, 0.1229)	(0.6137, 0.0404)	(0.6178, 0.004)
d5 (1.25)	(690, 15)	(0.8675 , 0.0151)	(0.8581, 0.1752)	(0.8533, 0.0251)	(0.8509, 0.0504)	(0.8585, 0.0233)	(0.8623, 0.1294)	(0.8581, 0.0721)	(0.8578, 0.0626)
d6 (2.05)	(131, 10)	(0.7727, 0.0011)	(0.7273, 0.0587)	(0.8003, 0.0022)	(0.6364, 0.0057)	(0.8182 , 0.0021)	(0.8182 , 0.0166)	(0.7273, 0.0054)	(0.8182 , 0.0005)
d7 (2.78)	(306, 3)	(0.6516, 0.0047)	(0.5248, 0.0392)	(0.6547, 0.0046)	(0.5, 0.0109)	(0.7162 , 0.0048)	(0.6778, 0.0355)	(0.6333, 0.0196)	(0.6593, 0.0007)
d8 (1.77)	(294, 12)	(0.8271, 0.0024)	(0.8212, 0.0148)	(0.8113, 0.0049)	(0.8212, 0.0182)	(0.7216, 0.006)	(0.858, 0.0264)	(0.8212, 0.0077)	(0.899 , 0.0007)
d9 (3.84)	(155, 19)	(0.6875, 0.0015)	(0.8542 , 0.0094)	(0.8417, 0.0028)	(0.8417, 0.0078)	(0.8417, 0.0054)	(0.8042, 0.031)	(0.8542 , 0.0054)	(0.8042, 0.0003)
d10 (1.71)	(368, 25)	(0.7492, 0.004)	(0.7888, 0.014)	(0.8032, 0.007)	(0.6479, 0.029)	(0.8169, 0.0099)	(0.8032, 0.3061)	(0.7751, 0.0096)	(0.8301 , 0.0014)
d11 (1.79)	(351, 33)	(0.9783 , 0.0039)	(0.9014, 0.0491)	(0.9283, 0.0081)	(0.8631, 0.0195)	(0.929, 0.0094)	(0.8401, 0.1657)	(0.9085, 0.0221)	(0.9642, 0.005)
d12 (1)	(556, 6)	(0.9765, 0.0112)	(0.8118, 0.2703)	(0.953, 0.0166)	(0.7871, 0.0579)	(0.965, 0.0172)	(0.9828 , 0.0776)	(0.878, 0.055)	(0.9688, 0.0465)
d13 (1.08)	(554, 6)	(0.9239 , 0.0101)	(0.8657, 0.1623)	(0.9022, 0.0151)	(0.7197, 0.0637)	(0.9063, 0.021)	(0.9158, 1.273)	(0.9119, 0.066)	(0.9185, 0.0461)
d14 (1.3)	(476, 166)	(0.9577 , 0.0066)	(0.821, 0.0124)	(0.9437, 0.0149)	(0.7062, 0.0264)	(0.93, 0.0123)	(0.9098, 0.0807)	(0.821, 0.0076)	(0.9226, 0.0076)
d15 (1.87)	(768, 8)	(0.7365, 0.0386)	(0.707, 0.2337)	(0.7273, 0.0323)	(0.5146, 0.0565)	(0.7428, 0.0285)	(0.7242, 0.2831)	(0.7434 , 0.147)	(0.7402, 0.0339)
d16 (6.29)	(102, 7)	(0.3571, 0.0016)	(0.744 , 0.0106)	(0.4643, 0.0013)	(0.6667, 0.0023)	(0.619, 0.0031)	(0.5, 0.0196)	(0.7083, 0.0051)	(0.6488, 0.0001)
d17 (1.41)	(265, 22)	(0.6223, 0.002)	(0.7118, 0.0183)	(0.6536, 0.0034)	(0.6137, 0.019)	(0.6579, 0.0051)	(0.6416, 0.0245)	(0.7224 , 0.0068)	(0.7055, 0.0008)
d18 (2.11)	(690, 14)	(0.4531, 0.0298)	(0.5247 , 0.0333)	(0.5035, 0.0264)	(0.496, 0.0459)	(0.5015, 0.0241)	(0.5188, 0.0977)	(0.5209, 0.014)	(0.5098, 0.0241)
d19 (2.1)	(2201, 3)	(0.7119, 0.1959)	(0.7117, 1.831)	(0.7298 , 0.4686)	(0.7, 2.7331)	(0.7119, 0.3628)	(0.7119, 4.0651)	(0.7119, 0.5962)	(0.7119, 0.0409)
d20 (1.86)	(683, 9)	(0.9598, 0.3757)	(0.9524, 0.465)	(0.9598, 0.1829)	(0.9778 , 0.173)	(0.9778 , 0.086)	(0.9778 , 0.253)	(0.9667, 0.1696)	(0.9741, 0.0123)
d21 (1.25)	(1372, 4)	(1 , 0.0963)	(0.9977, 1.0136)	(0.9977, 0.4604)	(1 , 0.7158)	(0.9977, 0.2399)	(0.9977, 0.4304)	(1 , 0.546)	(1 , 0.2276)
d22 (1.02)	(7400, 20)	(0.9825, 21.6567)	(0.8818, 271.3213)	(0.9846, 21.01)	(0.8451, 31.7247)	(0.9835, 13.4319)	(0.9751, 37.163)	(0.8668, 121.1155)	(0.9864 , 2.7862)
d23 (1)	(7400, 20)	(0.9744, 8.8872)	(0.9748, 2.6268)	(0.9263, 27.3189)	(0.974, 14.605)	(0.9748, 18.1506)	(0.9757, 43.9777)	(0.9748, 0.864)	(0.9761 , 7.0746)
d24 (1.84)	(19020, 10)	(0.8104, 18.9714)	(0.6098, 25.1728)	(0.8066, 42.2576)	(0.7646, 92.9534)	(0.7876, 57.2607)	(0.804, 123.213)	(0.8364 , 40.3638)	(0.7966, 38.5192)
d25 (8.03)	(48842, 14)	(0.6955, 120.2381)	(0.5, 565.7869)	(0.678, 657.9286)	(0.7767, 890.8303)	(0.7487, 318.7236)	*	(0.7705, 525.311)	(0.7816 , 491.0077)
Avg AUC		0.78	0.76	0.79	0.73	0.8	0.79	0.79	0.81
Avg rank		4.28	5.38	4.28	6.5	4.14	4.04	4.1	3.12

* Experiment stopped as the training extended for more than 1 day.

Table 4.6: Wilcoxon signed rank test of the proposed DC-IFLSTSVM with respect to other models. ✓ and ✗ denote null hypothesis (Hyp*) is rejected and not rejected, respectively.

Model	KEEL				UCI			
	r_+	r_-	p -value	Hyp*	r_+	r_-	p -value	Hyp*
TWSVM [22]	321	57	0.0015	✓	201.5	74.5	0.0536	✗
KIFSVM [109]	315	63	0.0024	✓	264	61	0.0063	✓
RFLSTSVM-CIL [151]	334	72	0.0029	✓	244.5	80.5	0.0271	✓
IIFTWSVM [50]	406	0	<0.00001	✓	269	31	0.0007	✓
IFTWSVM-ID [51]	266	112	0.0643	✗	183	93	0.17068	✗
DC-IFTBLDM [114]	351	84	0.0039	✓	139	71	0.2041	✗
DC-IFSVM	310	68	0.0036	✓	192.5	83.5	0.0969	✗

conservative nature of Friedman statistic, F_F statistic is calculated using equation (2.57). For the experiments over KEEL datasets, we have $N = 29$ and $md = 8$, thus $\chi_f^2 = 47.3788$ and $F_F = 8.5246$. The d.o.f. of F_F is (7, 196), hence $F_{(7,196)} = 2.0565$ which is less than F_F , leading to the inference that significant difference exists between the models. For UCI datasets $N = 25$ results in $\chi_f^2 = 24.32$, $F_F = 3.8736$ with d.o.f. (7, 168) which is greater than $F_{(7,168)} = 2.0644$. Thus, we infer the models are significantly different. Further, we performed Wilcoxon-signed rank test to support our analysis. Let r_+ and r_- signify the sum of positive and negative ranks, respectively. For a pair of model, if obtained p -value is less than 0.05, null hypothesis is rejected. Table 4.6 denotes the Wilcoxon-signed rank test over KEEL and UCI datasets. Hence, the overall statistical analysis depicts the superiority of the proposed DC-IFLSTSVM over the proposed DC-IFSVM and baseline models.

4.2.3 Application

As an application, we executed the experiment of the proposed DC-IFLSTSVM, DC-IFSVM and baseline models over BreakHis dataset [130] which comprises microscopic biopsy images. The dataset that we used contains 1240 images with $400\times$ magnification, categorized into benign and malignant tumors, further divided into four types each.

Benign types consist of adenosis (A), phyllodes tumor (PT), fibroadenoma (FB), and tubular adenoma (TA), while malignant types include papillary carcinoma (PC), mucinous carcinoma (MC), ductal carcinoma (DC), and lobular carcinoma (LC). The malignant cancer

Table 4.7: AUC with the time required for training of the proposed DC-IFSVM, DC-IFLSTSVM and baseline models over BreakHis datasets with RBF kernel.

Dataset	(sam,dim)	TWSVM [22] (AUC,time(sec))	KIFSVM [109] (AUC,time(sec))	RFLSTSVM-CIL [151] (AUC,time(sec))	IIFTWSVM [50] (AUC,time(sec))	IIFTWSVM-ID [51] (AUC,time(sec))	DC-IFTBLDM [114] (AUC,time(sec))	DC-IFSVM (AUC,time(sec))	DC-IFLSTSVM (AUC,time(sec))
A vs DC	(314, 768)	(0.7985, 0.017)	(0.8, 0.2393)	(0.8049, 0.0331)	(0.8005, 0.068)	(0.8868 , 0.0612)	(0.7917, 0.2055)	(0.8333, 0.0457)	(0.8593, 0.0084)
A vs LC	(243, 768)	(0.5015, 0.0081)	(0.5143, 0.0487)	(0.4992, 0.02)	(0.5, 0.0197)	(0.5, 0.0238)	(0.5395 , 0.0724)	(0.5143, 0.0185)	(0.5263, 0.0038)
A vs MC	(275, 768)	(0.5952, 0.0077)	(0.5569, 0.0497)	(0.5847, 0.0161)	(0.5628, 0.0179)	(0.6131 , 0.0132)	(0.5767, 0.0424)	(0.6038, 0.0107)	(0.586, 0.0023)
A vs PC	(244, 768)	(0.571, 0.0052)	(0.5945, 0.0713)	(0.5795, 0.0129)	(0.5223, 0.0195)	(0.6321, 0.0103)	(0.6736, 0.0268)	(0.6071, 0.009)	(0.677 , 0.0016)
F vs DC	(445, 768)	(0.8107, 0.0133)	(0.5434, 0.0683)	(0.7775, 0.0412)	(0.8208, 0.0692)	(0.8348 , 0.0343)	(0.6331, 0.0939)	(0.5434, 0.0265)	(0.8046, 0.0058)
F vs LC	(374, 768)	(0.6319, 0.0071)	(0.5847, 0.0175)	(0.6042, 0.0244)	(0.6583, 0.0417)	(0.5792, 0.0168)	(0.5431, 0.0652)	(0.7097 , 0.0087)	(0.6597, 0.0024)
F vs MC	(406, 768)	(0.5201, 0.0069)	(0.5341, 0.0199)	(0.5508, 0.0285)	(0.5508, 0.0286)	(0.569, 0.0155)	(0.5148, 0.0615)	(0.5657, 0.0091)	(0.5984 , 0.0016)
F vs PC	(375, 768)	(0.5139, 0.0071)	(0.5007, 0.0255)	(0.6038, 0.0248)	(0.6637 , 0.0498)	(0.6462, 0.0222)	(0.4773, 0.2999)	(0.4942, 0.0078)	(0.5877, 0.0027)
PT vs DC	(323, 768)	(0.8176, 0.0062)	(0.8354, 0.0147)	(0.8392, 0.0216)	(0.857, 0.0313)	(0.8702 , 0.0153)	(0.8211, 0.1831)	(0.8033, 0.0074)	(0.8702 , 0.0016)
PT vs LC	(252, 768)	(0.5161, 0.0042)	(0.4928, 0.029)	(0.519, 0.0132)	(0.5025, 0.0145)	(0.4814, 0.0093)	(0.5305 , 0.0236)	(0.4928, 0.0061)	(0.4978, 0.0013)
PT vs MC	(284, 768)	(0.631, 0.0052)	(0.6458 , 0.0149)	(0.5867, 0.0146)	(0.6407, 0.0156)	(0.6407, 0.0108)	(0.6265, 0.1887)	(0.604, 0.0065)	(0.6349, 0.0013)
PT vs PC	(253, 768)	(0.5462, 0.0045)	(0.6464 , 0.011)	(0.6022, 0.0121)	(0.5567, 0.0168)	(0.5357, 0.0095)	(0.3971, 0.0284)	(0.619, 0.0063)	(0.6464 , 0.0012)
TA vs DC	(338, 768)	(0.6455, 0.0067)	(0.6905, 0.028)	(0.6524, 0.0198)	(0.707 , 0.0277)	(0.6979, 0.0124)	(0.6754, 0.1485)	(0.6905, 0.0068)	(0.6905, 0.0012)
TA vs LC	(267, 768)	(0.7621, 0.0046)	(0.5814, 0.0136)	(0.7913, 0.0131)	(0.6125, 0.015)	(0.7467, 0.0142)	(0.7605, 0.0634)	(0.5814, 0.0069)	(0.7913 , 0.0016)
TA vs MC	(299, 768)	(0.6824, 0.0048)	(0.5811, 0.0141)	(0.6926 , 0.015)	(0.5607, 0.0156)	(0.6824, 0.0103)	(0.5677, 0.2274)	(0.5811, 0.0071)	(0.6538, 0.002)
TA vs PC	(268, 768)	(0.5683, 0.0048)	(0.546, 0.0137)	(0.5905, 0.0137)	(0.6127, 0.0152)	(0.5968, 0.0092)	(0.5984, 0.0545)	(0.546, 0.0069)	(0.6667 , 0.0011)
Avg AUC		0.63	0.6	0.64	0.63	0.65	0.61	0.61	0.67
Avg rank		5.09	5.41	4.56	4.28	3.5	5.5	5	2.66
<i>p</i> -value	-	0.0071	0.0015	0.0076	0.0385	0.3077	0.0032	0.0054	-

subtypes are differentiated through 16 pairings of benign and malignant data. Table 4.7 represent the performance of the models over BreakHis database.

The proposed DC-IFLSTSVM achieved an AUC of 0.67, leading the comparison with IFTWSVM-ID having an AUC of 0.65, followed by RFLSTSVM-CIL with 0.64, IIFTWSVM and TWSVM with 0.63, DC-IFTBLDM and proposed DC-IFSVM with an identical AUC of 0.61, than KIFSVM with an AUC of 0.60. The proposed DC-IFLSTSVM, with the lowest average rank of 2.66, demonstrates superior performance compared to the proposed DC-IFSVM and the baseline models. Consequently, the proposed DC-IFLSTSVM outperforms other models in classifying cancerous tumors from non-cancerous tumors. Figure 4.2 represents the box plot illustrating the performance of different models on the BreakHis dataset.

To strengthen the analysis over BreakHis dataset, we employ Friedman and Wilcoxon statistical tests. In this case, we have $N = 16$ and $md = 8$, which results in $\chi_f^2 = 18.3035$ and $F_F = 2.9302$ with d.o.f. 7 and (7, 105), respectively. $F_{(7,105)} = 2.098 < F_F$. Thus, the null hypothesis is rejected. Further, the last row of Table 4.7 represents the p -values of the Wilcoxon test, inferring that the models except IFTWSVM-ID are statistically different from the proposed DC-IFLSTSVM.

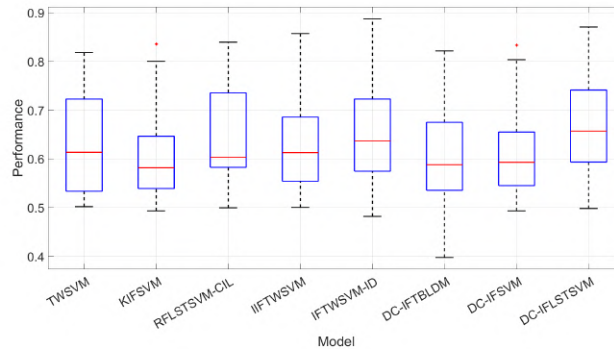


Figure 4.2: Box plot representation of results for different models over BreakHis dataset.

DC-IFSVM and DC-IFLSTSVM employ dual-center-based fuzzy scoring to emphasize

central samples and suppress outliers. Class imbalance is addressed via class-specific scaling of weights. DC-IFLSTSVM further improves scalability by solving a system of linear equations instead of quadratic programming.

4.3 Proposed class probability and generalized bell fuzzy twin SVM for imbalanced data

This section introduces a novel class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM-ID). Unlike IFUSVM, which has limitations in handling imbalance learning, CGFTSVM-ID introduces a new class probability and generalized bell (CPGB) membership function that assigns weights to individual samples based on their significance in constructing non-parallel hyperplanes. The generalized bell (gbell) function dynamically adjusts the membership values of majority class data points based on their location, effectively reducing the influence of outliers. Additionally, the class probability of majority samples is calculated to further mitigate the impact of noise. By integrating gbell values, class probabilities, and the imbalance ratio of the dataset, the CPGB membership function ensures robust handling of imbalanced data. Furthermore, CGFTSVM-ID employs a regularization term to adhere to the SRM principle, thereby avoiding overfitting, and its optimization problem is solved using the SOR technique, enabling efficient processing of large datasets without requiring them to reside entirely in memory.

4.3.1 Formulation of class probability and generalized bell fuzzy twin SVM for imbalanced data

To address the limitations of the well-known weighting assignment techniques discussed in Section 2.3.6 of Chapter 2, we propose class probability and generalized bell (CPGB)

membership function. By amalgamation of the CPGB membership function with TSVM, we present a novel framework called class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM-ID).

4.3.1.1 Weight assignment to the training data points

Here, we discuss the strategy of assigning weights to the training data points.

Determination of membership value: The generalized bell (gbell) function is a well-known fuzzy set membership function, frequently employed in fuzzy logic and fuzzy systems. It enables more adaptable and versatile fuzzy modelling and is an extension of the bell-shaped membership function [153]. In the proposed CGFTSVM-ID, it assigns membership value to the data points depending on their location in higher dimensional space. The assigned membership value is used to identify the outliers and border samples present in the majority class. Formally, the gbell function [153] is expressed as follows:

$$\mu_{gbell}(x_k) = \begin{cases} \frac{1}{1 + \left(\frac{\|\theta(x_k) - C_+\|}{a} \right)^{2r}}, & \text{if } y_k = 1, \\ \frac{1}{1 + \left(\frac{\|\theta(x_k) - C_-\|}{a} \right)^{2r}}, & \text{if } y_k = -1. \end{cases} \quad (4.29)$$

Here, C_+ and C_- correspond to the center of the positive class and negative class, respectively, defined as $C_+ = \frac{1}{l_1} \sum_{y_k=1} \theta(x_k)$ and $C_- = \frac{1}{l_2} \sum_{y_k=-1} \theta(x_k)$. a and r correspond to membership function positive parameters responsible for the width and steepness of the function (4.29), respectively. The gbell function is illustrated in Figure 4.3.

Determination of class probability: The impact of noise during model training is mitigated by the class probability value assigned to the training data points. This value, computed for each sample, is determined by the ratio of the number of samples belonging to the same class within a predefined neighborhood to the total number of data points in that

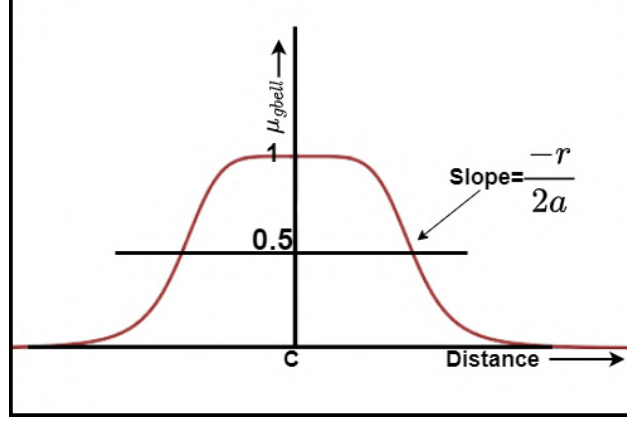


Figure 4.3: Graphical representation of gbell membership function of equation (4.29)

neighborhood.

$$p(x_k) = \frac{|\{x_j : \|\theta(x_j) - \theta(x_k)\| \leq \delta, y_j = y_k\}|}{|\{x_j : \|\theta(x_j) - \theta(x_k)\| \leq \delta\}|}, \quad (4.30)$$

where δ is a fixed positive real number. The greater value of $p(x_k)$ implies the larger belongingness of x_k to its own class. Thus, less probability value of noise leads to their reduced effect in the construction of hyperplanes.

$$(\|\theta(x) - \theta(\tilde{x})\|)^2 = K(x, x) - 2K(x, \tilde{x}) + K(\tilde{x}, \tilde{x}).$$

Proposed class probability and generalized bell (CPGB) membership function:

Combining the calculated gbell function values of each sample using equation (4.29) with the probability value calculated using equation (4.30) and the imbalance ratio (ir) of classes, we define CPGB membership function as follows:

$$\mu_g(x_k) = \begin{cases} 1, & \text{if } x_k = x_k^{min}, \\ \mu_{gbell}(x_k) \times p(x_k) \times \frac{1}{ir}, & \text{if } x_k = x_k^{maj}. \end{cases} \quad (4.31)$$

Theorem 3. For all the positive (negative) data points and the gbell function parameters C_+ (C_-), $a > 0$ and $r > 0$, the proposed CPGB membership function lies between 0 and 1.

In other words, $0 \leq \mu_g(x_k) \leq 1$, $k = 1, 2, \dots, l$.

Proof. For the minority (negative) class sample, the proof holds. For majority class samples,

$$\begin{aligned}
 & \|\theta(x_k) - C_+\| \geq 0 \quad \forall x_k \in \text{majority (positive) class} \\
 \Rightarrow & \left(\frac{\|\theta(x_k) - C_+\|}{a} \right)^{2r} \geq 0 \quad \forall C_+, a > 0 \text{ and } r > 0. \\
 \Rightarrow & 1 + \left(\frac{\|\theta(x_k) - C_+\|}{a} \right)^{2r} \geq 1 \Rightarrow \frac{1}{1 + \left(\frac{\|\theta(x_k) - C_+\|}{a} \right)^{2r}} \leq 1. \\
 \text{Also, } & \frac{1}{1 + \left(\frac{\|\theta(x_k) - C_+\|}{a} \right)^{2r}} > 0.
 \end{aligned}$$

Hence, $\forall x_k \in \text{majority class}$, $0 < \mu_{gbell}(x_k) \leq 1$. Also, $0 \leq p(x_k) \leq 1$ and $0 < \frac{1}{ir} \leq 1$.

Thus, the value of the proposed CPGB membership function lies between 0 and 1. $\square$

The proposed CPGB function effectively overcomes the limitations discussed in the previous section [2.3.6](#) in the following way.

- [1] Unlike IFN, where membership value decreases linearly from the center, the proposed CPGB function assigns membership value depending on the parameter ‘ a ’ of equation [\(4.29\)](#) that decides the width up to which the training data points should be assigned high membership value such that the classification performance improves. Thus, for the data distribution having majority samples at the boundary of the class, membership value varies with that of ‘ a ’ leading to an optimal classifier.
- [2] It can be clearly seen from Figure [4.3](#) that upto the distance ‘ a ’ from class center, assigned membership value is high depending on the steepness parameter ‘ r ’ of the gbell function. Thus, membership value of all the training points lying in the dense region of the class distribution varies depending on ‘ r ’ and may take an equivalent value to that of class center. Hence, leading to an improved contribution of such

points in the classification process.

- [3] The proposed CPGB function does not require the solution of the another QPP of SVDD. Also, we are considering the class probability of training data points in some fixed radii (δ) instead of KNN. Hence, the calculation of the proposed weighting technique has better efficiency with respect to time than the membership function of ACFSVM.

4.3.1.2 Proposed CGFTSVM-ID: Linear case

By integrating the above proposed CPGB membership function (4.31) with the TSVM model, we propose a novel class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM-ID). The equations of non-parallel hyperplanes for positive and negative classes are given as:

$$w_1^t x + b_1 = 0 \text{ and } w_2^t x + b_2 = 0, \quad (4.32)$$

respectively. The optimization problems for the proposed CGFTSVM-ID are expressed as:

$$\begin{aligned} \min_{w_1, b_1, \chi_2} & \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + \frac{1}{2} \lambda_1 (\|w_1\|^2 + b_1^2) + \lambda_2 \mu_{g_2}^t \chi_2 \\ \text{subject to} & -(M_2 w_1 + e_2 b_1) + \chi_2 \geq e_2, \chi_2 \geq 0, \end{aligned} \quad (4.33)$$

and

$$\begin{aligned} \min_{w_2, b_2, \chi_1} & \frac{1}{2} \|(M_2 w_2 + e_2 b_2)\|^2 + \frac{1}{2} \lambda_3 (\|w_2\|^2 + b_2^2) + \lambda_4 \mu_{g_1}^t \chi_1 \\ \text{subject to} & (M_1 w_2 + e_1 b_2) + \chi_1 \geq e_1, \chi_1 \geq 0, \end{aligned} \quad (4.34)$$

where $\lambda_1, \lambda_2, \lambda_3$ and λ_4 are the positive hyperparameters, and e_1, e_2 are the column vectors.

$\mu_{g_1} \in \mathbb{R}^{l_1}, \mu_{g_2} \in \mathbb{R}^{l_2}$ are the membership values of positive and negative classes; χ_1 and χ_2

are the slack variables. The first term in QPP (4.33) minimizes the distance of the positive class data points from the positive class hyperplane, and the second term corresponds to the regularization term which considers SRM principle. The third term minimizes the total penalty imposed on negative class data points which are at a distance less than 1 from the positive class hyperplane. A similar explanation follows for the QPP (4.34). The Lagrangian corresponding to QPP (4.33) having α_1, β_1 as positive Lagrange multipliers, is written as:

$$\begin{aligned} \mathcal{L}(w_1, b_1, \chi_2, \alpha_1, \beta_1) &= \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + \frac{1}{2} \lambda_1 (\|w_1\|^2 + b_1^2) \\ &+ \lambda_2 \mu_{g2}^t \chi_2 + \alpha_1^t (e_2 - \chi_2 + (M_2 w_1 + e_2 b_1)) - \beta_1^t \chi_2. \end{aligned} \quad (4.35)$$

From the K.K.T. conditions, differentiate the Lagrangian (4.35) we obtain:

$$\frac{\partial \mathcal{L}}{\partial w_1} = M_1^t (M_1 w_1 + e_1 b_1) + \lambda_1 w_1 + M_2^t \alpha_1 = 0, \quad (4.36)$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = e_1^t (M_1 w_1 + e_1 b_1) + \lambda_1 b_1 + e_2^t \alpha_1 = 0, \quad (4.37)$$

$$\frac{\partial \mathcal{L}}{\partial \chi_2} = \lambda_2 \mu_{g2} - \alpha_1 - \beta_1 = 0. \quad (4.38)$$

Combining equations (4.36) and (4.37), we get

$$\left(\begin{bmatrix} M_1^t \\ e_1^t \end{bmatrix} \begin{bmatrix} M_1 & e_1 \end{bmatrix} + \lambda_1 I \right) \begin{bmatrix} w_1 \\ b_1 \end{bmatrix} + \begin{bmatrix} M_2^t \\ e_2^t \end{bmatrix} \alpha_1 = 0. \quad (4.39)$$

Assuming $P = \begin{bmatrix} M_1 & e_1 \end{bmatrix}$, $Q = \begin{bmatrix} M_2 & e_2 \end{bmatrix}$, the equation (4.39) reduces to

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(P^t P + \lambda_1 I)^{-1} Q^t \alpha_1. \quad (4.40)$$

Similarly, for the QPP (4.34) having α_2, β_2 as the positive Lagrange multipliers, we get the relation:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Q^t Q + \lambda_3 I)^{-1} P^t \alpha_2. \quad (4.41)$$

The Wolfe dual for the primal problems (4.33) and (4.34) can be expressed as:

$$\begin{aligned} \min_{\alpha_1} \quad & \frac{1}{2} \alpha_1^t Q (P^t P + \lambda_1 I)^{-1} Q^t \alpha_1 - e_2^t \alpha_1 \\ \text{subject to} \quad & 0 \leq \alpha_1 \leq \lambda_2 \mu_{g_2}, \end{aligned} \quad (4.42)$$

and

$$\begin{aligned} \min_{\alpha_2} \quad & \frac{1}{2} \alpha_2^t P (Q^t Q + \lambda_3 I)^{-1} P^t \alpha_2 - e_1^t \alpha_2 \\ \text{subject to} \quad & 0 \leq \alpha_2 \leq \lambda_4 \mu_{g_1}, \end{aligned} \quad (4.43)$$

respectively. A new data point x is allocated to a class based on the function

$$f(x) = \operatorname{argmin} \left\{ \frac{|w_1^t x + b_1|}{\|w_1\|}, \frac{|w_2^t x + b_2|}{\|w_2\|} \right\}. \quad (4.44)$$

4.3.1.3 Proposed CGFTSVM-ID: Non-linear case

For the non-linear case, the kernel-generated non-parallel surfaces are given by equations:

$$K(x, M^t) w_1 + b_1 = 0, \quad K(x, M^t) w_2 + b_2 = 0, \quad (4.45)$$

where $K(.,.)$ represents the kernel function and $M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$. The nonlinear optimization problems corresponding to the proposed CGFTSVM-ID are defined as:

$$\begin{aligned} \min_{w_1, b_1, \chi_2} & \frac{1}{2} \|K(M_1, M^t)w_1 + e_1 b_1\|^2 + \frac{1}{2} \lambda_1 (\|w_1\|^2 + b_1^2) + \lambda_2 \mu_{g_2}^t \chi_2 \\ \text{subject to} & - (K(M_2, M^t)w_1 + e_2 b_1) + \chi_2 \geq e_2, \chi_2 \geq 0 \end{aligned} \quad (4.46)$$

and

$$\begin{aligned} \min_{w_2, b_2, \chi_1} & \frac{1}{2} \|K(M_2, M^t)w_2 + e_2 b_2\|^2 + \frac{1}{2} \lambda_3 (\|w_2\|^2 + b_2^2) + \lambda_4 \mu_{g_1}^t \chi_1 \\ \text{subject to} & (K(M_1, M^t)w_2 + e_1 b_2) + \chi_1 \geq e_1, \chi_1 \geq 0, \end{aligned} \quad (4.47)$$

where the terms have the same notion as discussed in the linear case. Solving as the linear case by using K.K.T. conditions, we obtain the dual for primal problems (4.46) and (4.47) as:

$$\begin{aligned} \min_{\alpha_1} & \frac{1}{2} \alpha_1^t Q (P^t P + \lambda_1 I)^{-1} Q^t \alpha_1 - e_2^t \alpha_1 \\ \text{subject to} & 0 \leq \alpha_1 \leq \lambda_2 \mu_{g_2} \end{aligned} \quad (4.48)$$

and

$$\begin{aligned} \min_{\alpha_2} & \frac{1}{2} \alpha_2^t P (Q^t Q + \lambda_3 I)^{-1} P^t \alpha_2 - e_1^t \alpha_2 \\ \text{subject to} & 0 \leq \alpha_2 \leq \lambda_4 \mu_{g_1}, \end{aligned} \quad (4.49)$$

respectively, where $P = \begin{bmatrix} K(M_1, M^t) & e_1 \end{bmatrix}$, $Q = \begin{bmatrix} K(M_2, M^t) & e_2 \end{bmatrix}$.

A new data point (x) is labeled as positive or negative depending on the function:

$$f(x) = \operatorname{argmin} \left\{ \frac{|K(x, M^t)w_1 + b_1|}{\sqrt{w_1^t K(M, M^t)w_1}}, \frac{|K(x, M^t)w_2 + b_2|}{\sqrt{w_2^t K(M, M^t)w_2}} \right\}. \quad (4.50)$$

The Algorithm 4.1 depicts the steps for proposed CGFTSVM-ID.

Algorithm 4.1 Proposed CGFTSVM-ID

Input: Training data point $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, testing data, hyperparameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, membership function parameters a and r .

Output: Labels of testing data set.

- [1] Determine the gbell membership value of training data points using $\mu_{gbell}(x_k) = \begin{cases} \frac{1}{1 + \left(\frac{\|\theta(x_k) - C_+\|}{a} \right)^{2r}}, & \text{if } y_k = 1, \\ \frac{1}{1 + \left(\frac{\|\theta(x_k) - C_-\|}{a} \right)^{2r}}, & \text{if } y_k = -1. \end{cases}$
 - [2] Obtain class probability of training data using $p(x_k) = \frac{|\{x_j: \|\theta(x_j) - \theta(x_k)\| \leq \delta, y_j = y_k\}|}{|\{x_j: \|\theta(x_j) - \theta(x_k)\| \leq \delta\}|}$.
 - [3] Calculate the CPGB membership value of the training data points following $\mu_g(x_k) = \begin{cases} 1, & \text{if } x_k = x_k^{min}, \\ \mu_{gbell}(x_k) \times p(x_k) \times \frac{1}{ir}, & \text{if } x_k = x_k^{maj}. \end{cases}$
 - [4] Integrating the assigned membership value, obtain α_1, α_2 by solving the optimization problems (4.42), (4.43) for the linear case and (4.48), (4.49) for the non-linear case.
 - [5] Obtain $\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(P^t P + \lambda_1 I)^{-1} Q^t \alpha_1$ and $\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (Q^t Q + \lambda_3 I)^{-1} P^t \alpha_2$.
 - [6] Determine labels of test data point using $f(x) = \operatorname{argmin} \left\{ \frac{|w_1^t x + b_1|}{\|w_1\|}, \frac{|w_2^t x + b_2|}{\|w_2\|} \right\}$ for linear case and $f(x) = \operatorname{argmin} \left\{ \frac{|K(x, M^t)w_1 + b_1|}{\sqrt{w_1^t K(M, M^t)w_1}}, \frac{|K(x, M^t)w_2 + b_2|}{\sqrt{w_2^t K(M, M^t)w_2}} \right\}$ for the non-linear case.
-

4.3.1.4 Time complexity

Consider a binary class classification scenario with a total of n samples, where l_1 samples represent the positive class and l_2 samples represent the negative class. Assuming the positive class as the majority class, the imbalance ratio (ir) = $\frac{l_1}{l_2}$. The time complexity of the

proposed CGFTSVM-ID is determined by two primary components. Firstly, it involves calculating the CPGB membership value for the data points, which comprises the calculation of the generalized bell function value (equation 4.29) and the class probability (equation 4.30) for the majority class samples. The complexity of calculating gbell value would be $O(l_1)$, as it entails a fixed number of arithmetic operations. Similarly, the complexity for calculating the class probability is equal to $O(l_1)$ [49]. Secondly, the computational complexity of SVM is $O(n^3)$ which implies complexity of TSVM is $O(l_1^3) + O(l_2^3)$ [22], where $n = l_1 + l_2$. Therefore, solving the obtained QPPs involves a computational complexity of $(ir^3 + 1)O(l_2^3)$. Consequently, the overall time complexity of the proposed CGFTSVM-ID is $(ir^3 + 1)O(l_2^3) + O(l_1) + O(l_1)$.

4.3.2 Numerical experiments

In this section, we delve into the numerical experiments and present the corresponding results for the proposed CGFTSVM-ID along with state-of-the-art baseline models which are TSVM [22], IFTWSVM [49], IFTWSVM-ID [51], intuitionistic fuzzy twin support vector machines with the insensitive pinball loss (pin-IFTWSVM) [111], IIFTWSVM [50], ACFSVM [46] and ACFTSVM [154].

Experimental setup: All the experiments are carried out in MATLAB 2022b on a desktop PC with 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz processor and 16 GB RAM. 70% of each dataset is used for training and the remaining dataset is used for testing. For the selection of the optimal parameters, we used grid search with 10-fold cross-validation. For brevity and increased computational efficiency, we assumed $\lambda_1 = \lambda_3$ and $\lambda_2 = \lambda_4$. All the datasets are normalized using Z-score normalization technique. To implement the non-linear case, we used the Gaussian kernel ($\exp(\|x_i - x_j\|^2 / 2\sigma^2)$). The optimization problems of the proposed CGFTSVM-ID are solved using SOR technique. The relaxation

factor (ω) in the SOR method is typically constrained to the interval $(0, 2)$ [135]. A larger ω accelerates convergence, but it can introduce instability. On the contrary, a smaller ω results in slower convergence. To strike a balance and achieve optimal convergence with stable results, we empirically selected ω as 0.5. For better comparison, we solved the optimization problems of the compared twin variants using SOR except ACFSVM which is solved using MATLAB quadprog toolbox. Different parameters of the optimization problems vary in the following way: λ_1 and λ_2 in the twin variant models (and λ in ACFSVM) are chosen from range $\{10^{-5}, \dots, 10^5\}$; the Gaussian kernel parameter σ is chosen from the range $\{2^{-5}, \dots, 2^5\}$. The value of ϵ and τ for pin-IFTWSVM are taken same as in [111]. The parameters of ACFSVM except for λ and σ are taken the same as in [46]. The gbell membership function parameter ' a ' takes the range $[1/2, 5/8, 3/4, 7/8, 1] \times (\text{radius of positive/negative class})$. We consider the proposed CPGB membership function steepness parameter ' r ' same as the value of ' a '. We considered δ as the maximum radius encompassing positive and negative classes, ensuring a broader neighborhood coverage. The class probability, influenced by this neighborhood information, extends across a wide area around each data point. Moreover, choosing δ as the maximum radii of the positive and negative classes dynamically adapts δ according to the spread or variance of the data. This can help the model to adjust to the scale of the dataset, potentially improving generalization performance. The performance indicators for the experiments are area under ROC (AUC), sensitivity (Sens.) specificity (spec.), G-mean and F-measure which are defined as follows:

$$\text{Sens.} = \frac{\text{tp}}{\text{tp} + \text{fn}}, \text{Spec.} = \frac{\text{tn}}{\text{tn} + \text{fp}}, \text{AUC} = \frac{1 + \text{tpr} - \text{fpr}}{2},$$

$$\text{G-mean} = \sqrt{\text{Sens.} \times \text{Spec.}}, \text{F-measure} = \frac{2 \times \text{Sens.} \times \text{Prec.}}{\text{Sens.} + \text{Prec.}},$$

where tp, fp, tn and fn denote the true positive, false positive, true negative and false negative samples, respectively. tpr, fpr and Prec. denote the true positive rate, false positive rate and precision calculated as:

$$\text{tpr} = \frac{\text{tp}}{\text{tp} + \text{fn}}, \quad \text{fpr} = \frac{\text{fp}}{\text{fp} + \text{tn}} \quad \text{and} \quad \text{Prec.} = \frac{\text{tp}}{\text{tp} + \text{fp}},$$

respectively. To carry out the experiments, we have considered the synthetic KEEL datasets and several real-world KEEL and UCI [126] datasets. For the application of the proposed CGFTSVM-ID, we employed it on the publicly available Alzheimer's disease neuroimaging initiative (ADNI) dataset for the classification of Alzheimer's disease (AD).

4.3.2.1 Synthetic datasets

To analyse the performance of the proposed CGFTSVM-ID with the baseline models, we employed experiments on the synthetic datasets publicly available on the imbalance KEEL dataset repository. The details of the used synthetic datasets are given in Table 4.8. The performance of the models on the binary class synthetic datasets is depicted in Table 4.9. The proposed CGFTSVM-ID exhibits the highest AUC values for datasets 04clover5z-600-5-0-BI, 04clover5z-600-5-30-BI, 04clover5z-600-5-60-BI, 04clover5z-800-7-0-BI, and 04clover5z-800-7-50-BI. While baseline models outperform the proposed CGFTSVM-ID in the remaining datasets, the average AUC of the proposed CGFTSVM-ID is 0.8289 with average rank of 1.7, which is superior to those of the baseline models. This clearly establishes the superiority of CGFTSVM-ID over the baseline models. Table 4.10 comprises the optimal parameters of the models corresponding to the maximum AUC across each synthetic dataset. For better visualisation, we have compared the AUC values using bar graph which is depicted in Figure 4.4.

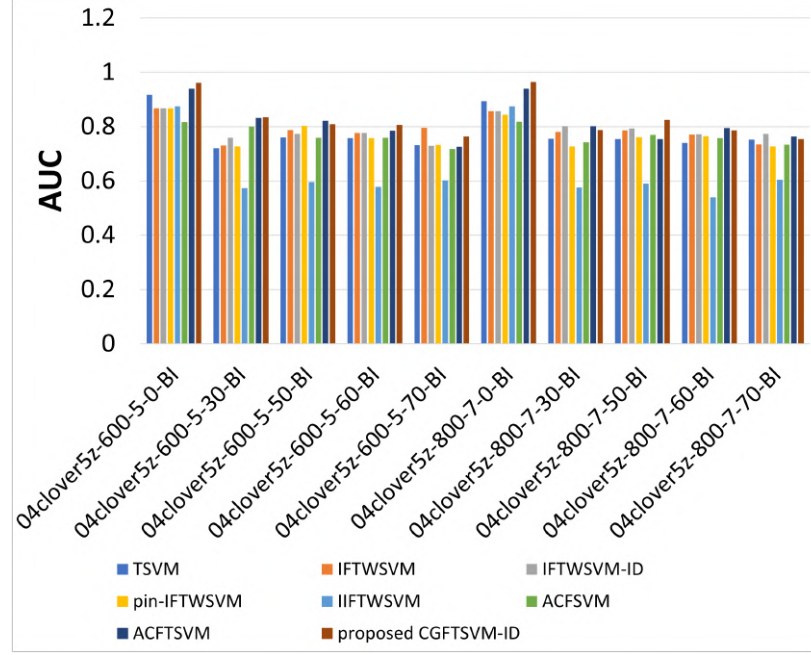


Figure 4.4: AUC representation of models over KEEL synthetic datasets.

4.3.2.2 Real-world datasets

To further examine the performance of the proposed CGFTSVM-ID in comparison to the state-of-the-art methods TSVM [22], IFTWSVM [49], IFTWSVM-ID [51], pin-IFTWSVM [111], IIFTWSVM [50], ACFSVM [46] and ACFTSVM [154], we performed experiments on 51 real-world KEEL and UCI datasets. The details of the datasets are mentioned in Table 4.11. First, we will discuss the results of the KEEL datasets, followed by the results over UCI datasets.

The detailed performance of 33 real-world KEEL datasets (with an imbalance ratio in the range of 1 to 85.88) in terms of AUC, training time (in seconds), sensitivity and specificity are depicted in Table 4.12. For the datasets namely crossplane150, ecoli0267vs35, ecoli0347vs56, ecoli067vs35, aus, checkerboard, monk2, monk3, sonar, vowel, wpbc, yeast2vs8, abalone-17vs78910, abalone-20vs8910, kr-vs-k-zero_vs_eight, shuttle-2vs5, winequality-red-8vs67, winequality-red-8vs6, winequality-white-39vs5 and

Table 4.8: Details of the synthetic data used.

Dataset	abbreviation	Disturbance ratio	number of samples	number of features	Imbalance ratio
04clover5z-600-5-0-BI	c5z-600-5-0	0	600	2	5
04clover5z-600-5-30-BI	c5z-600-5-30	30	600	2	5
04clover5z-600-5-50-BI	c5z-600-5-50	50	600	2	5
04clover5z-600-5-60-BI	c5z-600-5-60	60	600	2	5
04clover5z-600-5-70-BI	c5z-600-5-70	70	600	2	5
04clover5z-800-7-0-BI	c5z-800-7-0	0	800	2	7
04clover5z-800-7-30-BI	c5z-800-7-30	30	800	2	7
04clover5z-800-7-50-BI	c5z-800-7-50	50	800	2	7
04clover5z-800-7-60-BI	c5z-800-7-60	60	800	2	7
04clover5z-800-7-70-BI	c5z-800-7-70	70	800	2	7

winequality-white-3vs7, the proposed CGFTSVM-ID has the highest AUC value. Other datasets have greater AUC for baseline models, nevertheless, the average AUC of the proposed CGFTSVM-ID is 0.8344 which is greater than that of baseline models. ACFTSVM has the second greater value which is 0.7914. The average AUC of the remaining models are as follows: 0.7769, 0.7754, 0.7728, 0.7694, 0.7641 and 0.7022 for IFTWSVM, TSVM, pin-IFTWSVM, ACFSVM, IFTWSVM-ID and IIFTWSVM, respectively. Hence, CGFTSVM-ID has better classification performance over KEEL datasets than baseline models. However, the average AUC value is not reliable to depict the superiority of a model, as the lower AUC value of a dataset may be compensated by a high AUC value of another dataset. To overcome this, we assigned ranks to all the models over each dataset and calculated the average ranks of the models. The assignment of rank is done via: the best-performing model is assigned the lowest rank and the worst-performing model is assigned the highest rank over each dataset. The proposed CGFTSVM-ID has the lowest average rank of 2.09, followed by IFTWSVM with average rank 4.2. The ranks of the other baseline models are following: 4.24, 4.44, 4.56, 4.82, 4.88, 6.77 for the models ACFTSVM, pin-IFTWSVM, TSVM, IFTWSVM-ID, ACFSVM, and IIFTWSVM, respectively. The average metric values across the KEEL datasets of the proposed CGFTSVM-ID and baseline models are presented in

Table 4.9: AUC with training time, sensitivity, specificity, G-mean and F-measure value of the proposed CGFTSVM-ID and baseline models on synthetic KEEL datasets.

Dataset (ir) (samples, features)	T SVM [22] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	IFTWSVM [49] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	IFTWSVM-ID [51] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	pin-IFTWSVM [111] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	IIFTWSVM [50] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	ACFSVM [46] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	ACFTSVM [154] (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)	proposed CGFTSVM-ID (AUC, time(seconds)) (Sens., Spec.) (G-mean, F-measure)
04clover5z-600-5-0-BI	(0.9175, 0.0388) (0.8421, 0.993) (0.9014, 0.9144)	(0.8675, 0.05) (0.7632, 0.9718) (0.8169, 0.8612)	(0.8675, 0.0489) (0.7368, 0.9789) (0.8116, 0.8493)	(0.8675, 0.0902) (0.7632, 0.9718) (0.8169, 0.8612)	(0.8753, 0.0614) (0.8421, 0.9085) (0.7711, 0.8746)	(0.8163, 0.2733) (0.8158, 0.8169) (0.6526, 0.8163)	(0.9394, 0.1435) (0.9211, 0.9577) (0.8861, 0.9392)	(0.9394, 0.1435) (1, 0.9225) (0.8736, 0.9605)
04clover5z-600-5-30-BI	(0.72, 0.0137) (0.5526, 0.8873) (0.56, 0.7003)	(0.7304, 0.0304) (0.6579, 0.8028) (0.5495, 0.7268)	(0.7583, 0.0282) (0.6579, 0.8169) (0.5618, 0.7331)	(0.7269, 0.0298) (0.7297, 0.7241) (0.45, 0.7269)	(0.5739, 0.0432) (1, 0.1479) (0.3858, 0.3846)	(0.8002, 0.1553) (0.9737, 0.6268) (0.5781, 0.7812)	(0.8321, 0.0997) (0.8684, 0.7958) (0.66, 0.8313)	(0.8321, 0.0997) (0.8947, 0.7746) (0.6538, 0.8325)
04clover5z-600-5-50-BI	(0.76, 0.0209) (0.8158, 0.7042) (0.5586, 0.758)	(0.7871, 0.0219) (0.9474, 0.6268) (0.5669, 0.7706)	(0.773, 0.0221) (0.9474, 0.5986) (0.5496, 0.7531)	(0.8021, 0.0283) (0.9211, 0.6831) (0.5932, 0.7932)	(0.5953, 0.0431) (0.8947, 0.2958) (0.3953, 0.5144)	(0.7589, 0.1839) (0.9474, 0.5704) (0.5333, 0.7351)	(0.8208, 0.1351) (0.7895, 0.8521) (0.6742, 0.8202)	(0.8091, 0.0311) (0.9211, 0.6972) (0.6034, 0.8013)
04clover5z-600-5-60-BI	(0.7582, 0.0205) (0.8684, 0.6479) (0.5455, 0.7501)	(0.7767, 0.0249) (0.8421, 0.7113) (0.5766, 0.7739)	(0.7767, 0.0274) (0.7895, 0.7324) (0.566, 0.7604)	(0.7574, 0.0267) (0.7895, 0.7254) (0.5607, 0.7567)	(0.578, 0.0919) (0.1842, 0.9718) (0.2857, 0.4231)	(0.7589, 0.1828) (0.9474, 0.5704) (0.5333, 0.7351)	(0.7847, 0.113) (0.8158, 0.7535) (0.5962, 0.784)	(0.8056, 0.0297) (0.9211, 0.6901) (0.5983, 0.7973)
04clover5z-600-5-70-BI	(0.7317, 0.0272) (0.9211, 0.5423) (0.5072, 0.7067)	(0.796, 0.0328) (0.8947, 0.6972) (0.5913, 0.7898)	(0.7294, 0.0307) (0.6579, 0.7746) (0.5263, 0.7139)	(0.732, 0.0288) (0.7105, 0.7535) (0.54, 0.7317)	(0.6021, 0.0427) (1, 0.2042) (0.4021, 0.4519)	(0.7172, 0.1526) (0.6316, 0.8028) (0.5333, 0.7121)	(0.7259, 0.0976) (0.6842, 0.7676) (0.5361, 0.7247)	(0.7628, 0.0181) (0.7368, 0.7887) (0.5833, 0.7623)
04clover5z-800-7-0-BI	(0.8931, 0.0373) (0.8108, 0.9754) (0.8333, 0.8893)	(0.8575, 0.0368) (0.7297, 0.9852) (0.806, 0.8479)	(0.8575, 0.036) (0.6216, 0.9852) (0.7302, 0.7826)	(0.844, 0.0344) (0.7027, 0.9852) (0.7879, 0.8321)	(0.8746, 0.0565) (0.8378, 0.9113) (0.7209, 0.8738)	(0.8179, 0.2844) (0.8378, 0.798) (0.5688, 0.8177)	(0.9398, 0.222) (0.9189, 0.9606) (0.8608, 0.9395)	(0.9643, 0.0413) (0.973, 0.9557) (0.878, 0.9643)
04clover5z-800-7-30-BI	(0.7554, 0.0233) (0.5946, 0.9163) (0.5789, 0.7381)	(0.781, 0.0458) (0.8378, 0.7241) (0.5, 0.7789)	(0.8018, 0.0378) (0.9189, 0.6847) (0.5037, 0.7932)	(0.7269, 0.0339) (0.7297, 0.7241) (0.45, 0.7269)	(0.5764, 0.0713) (1, 0.1527) (0.3008, 0.3908)	(0.7429, 0.2914) (0.7568, 0.7291) (0.4667, 0.7428)	(0.802, 0.1853) (0.8108, 0.7931) (0.5505, 0.8019)	(0.7872, 0.0432) (0.8108, 0.7635) (0.5217, 0.7868)
04clover5z-800-7-50-BI	(0.754, 0.0349) (0.7838, 0.7241) (0.4754, 0.7534)	(0.7858, 0.034) (0.9459, 0.6059) (0.473, 0.7693)	(0.7932, 0.0318) (0.9459, 0.7389) (0.4605, 0.7571)	(0.7614, 0.0402) (0.7838, 0.7389) (0.4874, 0.761)	(0.5897, 0.122) (0.1892, 0.9901) (0.3043, 0.4328)	(0.7697, 0.3106) (0.973, 0.5665) (0.4472, 0.7424)	(0.754, 0.1624) (0.7838, 0.7241) (0.4754, 0.7534)	(0.8241, 0.0542) (0.8649, 0.7833) (0.5664, 0.823)
04clover5z-800-7-60-BI	(0.7404, 0.0305) (0.8108, 0.67) (0.4478, 0.737)	(0.7712, 0.0312) (0.7838, 0.7586) (0.5043, 0.7711)	(0.7724, 0.0521) (0.8108, 0.7389) (0.5, 0.774)	(0.7638, 0.0322) (0.7838, 0.7438) (0.4915, 0.7636)	(0.5394, 0.058) (1, 0.0788) (0.2835, 0.2807)	(0.7574, 0.2836) (0.973, 0.5419) (0.4337, 0.7261)	(0.4472, 0.7424) (0.8108, 0.7783) (0.5357, 0.7944)	(0.7858, 0.0552) (0.8919, 0.6798) (0.4889, 0.7787)
04clover5z-800-7-70-BI	(0.7527, 0.0282) (0.8108, 0.6946) (0.4651, 0.7504)	(0.7343, 0.03) (0.7297, 0.7389) (0.4615, 0.7343)	(0.7735, 0.0304) (0.8919, 0.6552) (0.4714, 0.7644)	(0.7269, 0.0305) (0.7297, 0.7241) (0.45, 0.7269)	(0.6047, 0.0601) (0.973, 0.2365) (0.3158, 0.4796)	(0.7341, 0.311) (0.8919, 0.5764) (0.4231, 0.717)	(0.7637, 0.156) (0.8378, 0.6897) (0.4733, 0.7601)	(0.754, 0.0531) (0.7297, 0.7783) (0.4954, 0.7536)
Average AUC	0.7783	0.7888	0.7903	0.7709	0.6409	0.7674	0.8157	0.8289
Average Rank	5.15	4.1	3.7	5.6	7.2	6	2.55	1.7

Table 4.10: Optimal parameters corresponding to AUC values of the proposed CGFTSVM-ID and baseline models on synthetic KEEL datasets.

Dataset	TSVM [22] (λ_1, σ)	IFTWSVM [49] ($\lambda_1, \lambda_2, \sigma$)	IFTWSVM-ID [51] ($\lambda_1, \lambda_2, \sigma$)	pin-IFTWSVM [111] ($\lambda_1, \lambda_2, \sigma, \tau, \epsilon$)	IIFTWSVM [50] ($\lambda_1, \lambda_2, \sigma$)	ACFSVM [46] (λ_1, σ)	ACFTSVM [154] (λ_1, σ)	proposed CGFTSVM-ID ($(\lambda_1, \lambda_2, \sigma, a)$)
04clover5z-600-5-0-BI	(10, 0.5)	(0.1, 1000, 0.125)	(0.1, 1000, 0.125)	(0.001, 0.1, 0.125, 0.5, 0.5)	(0.00001, 0.00001, 0.125)	(1, 1)	(1, 0.5)	(0.01, 100, 0.5, 0.5)
04clover5z-600-5-30-BI	(1, 0.125)	(1, 1, 1)	(10, 0.1, 1)	(0.0001, 0.01, 8, 0.5, 1)	(0.00001, 1, 1)	(10, 1)	(1, 1)	(1, 10, 0.5, 0.625)
04clover5z-600-5-50-BI	(100, 2)	(0.0001, 0.00001, 16)	(0.00001, 0.00001, 16)	(0.00001, 0.1, 8, 0.2, 0.2)	(1, 1, 4)	(10, 1)	(0.1, 0.5)	(10, 10, 0.5, 1)
04clover5z-600-5-60-BI	(100, 2)	(0.00001, 0.1, 32)	(0.00001, 0.1, 32)	(0.0001, 0.1, 8, 0.05, 0.01)	(10, 100, 1)	(10, 1)	(1, 1)	(10, 100, 0.5, 0.5)
04clover5z-600-5-70-BI	(100, 32)	(0.0001, 0.1, 8)	(0.001, 0.00001, 2)	(0.00001, 0.1, 8, 0.5, 0.01)	(0.00001, 1, 1)	(1, 1)	(1, 4)	(0.1, 0.00001, 0.5, 0.5)
04clover5z-800-7-0-BI	(10, 0.5)	(0.1, 10000, 0.125)	(0.1, 10000, 0.125)	(0.0001, 0.01, 0.125, 0.05, 1)	(0.00001, 0.00001, 0.125)	(1, 1)	(1, 0.5)	(0.01, 100, 0.5, 0.875)
04clover5z-800-7-30-BI	(1, 0.125)	(0.0001, 0.01, 2)	(0.00001, 0.1, 16)	(0.0001, 0.01, 8, 0.5, 1)	(0.00001, 1, 1)	(1, 1)	(1, 1)	(100, 10, 0.25, 0.625)
04clover5z-800-7-50-BI	(100, 2)	(0.0001, 0.1, 8)	(0.001, 0.00001, 8)	(0.0001, 0.01, 8, 0.5, 1)	(10, 10, 1)	(10, 1)	(100, 2)	(10, 100, 0.5, 0.75)
04clover5z-800-7-60-BI	(100, 2)	(0.1, 0.00001, 4)	(0.0001, 0.1, 16)	(0.00001, 0.01, 16, 0.5, 1)	(0.00001, 1, 1)	(10, 1)	(1, 1)	(10000, 10000, 0.25, 0.875)
04clover5z-800-7-70-BI	(100, 2)	(0.01, 0.00001, 4)	(0.00001, 0.00001, 16)	(0.0001, 0.01, 8, 0.5, 0.5)	(0.00001, 1, 1)	(10, 1)	(10, 2)	(10, 10, 0.5, 1)

Table 4.11: Details of the real-world data used.

UCI Datasets	abbreviation	number of samples	number of features	imbalance ratio
breast_cancer	bc	286	9	2.36
breast_cancer_wisc	bcw	699	9	1.9
congressional_voting	cv	435	16	1.59
credit_approval	ca	690	15	1.25
cylinder_bands	cb	512	35	1.56
echocardiogram	echo	131	10	2.05
ilpd_indian_liver	iil	583	9	2.05
ionosphere	iono	351	33	2.49
mammographic	mammo	961	5	1.16
molec_biol_promoter	mbp	106	57	1
monks_2	monks2	601	6	1.92
musk_1	musk1	476	166	1.3
oocytes_merluccius_nucleus_4d	omn_4d	1473	9	1.34
oocytes_trisopterus_nucleus_2f	otn_2f	912	25	1.37
spect	spect	265	22	1.41
statlog_german_credit	sgc	1000	24	2.33
tic_tac_toe	tic_tac_toe	958	9	1.89
vertebral_column_2clases	vc_2clases	310	6	2.1
KEEL Datasets				
bupa-or-liver-disorders	bupa	345	6	1.38
crossplane150	crossplane150	150	2	1
ecoli-0-1-4-7_vs_2-3-5-6	ecoli0147vs2356	336	7	10.59
ecoli-0-2-3-4_vs_5	ecoli0234vs5	202	7	9.1
ecoli-0-2-6-7_vs_3-5	ecoli0267vs35	224	7	9.18
ecoli-0-3-4-7_vs_5-6	ecoli0347vs56	257	7	9.28
ecoli-0-6-7_vs_3-5	ecoli067vs35	222	7	9.09
ecoli-0-1_vs_5	ecoli01vs5	240	6	11
glass5	glass5	214	9	22.78
aus	aus	690	14	1.25
brwiscosin	brwis	683	9	1.86
checkerboard_Data	checkerboard	690	14	1.25
iono	iono	351	33	1.79
monk1	monk1	556	6	1.04
monk2	monk2	601	7	1.92
monk3	monk3	556	6	1.01
segment0	segment0	2308	19	6.02
sonar	sonar	208	60	1.14
vowel	vowel	988	10	9.98
wdbc	wdbc	194	33	3.22
yeast2vs8	yeast2vs8	483	8	23.15
abalone-17_vs_7-8-9-10	abalone-17vs78910	2338	7	39.31
abalone-19_vs_10-11-12-13	abalone-19vs10111213	1622	7	49.69
abalone-20_vs_8-9-10	abalone-20vs8910	1916	7	72.69
abalone-21_vs_8	kr-vs-k-zero_vs_eight	581	3	40.5
kr-vs-k-zero_vs_eight	poker-89vs5	1460	10	32.47
poker-8-9_vs_5	poker-89vs6	2075	10	53.07
poker-8-9_vs_6	shuttle-2vs5	1485	9	80.22
winequality-red-8_vs_6-7	winequality-red-8vs67	855	11	82
winequality-red-8_vs_6	winequality-red-8vs6	656	11	58.4
winequality-white-3-9_vs_5	winequality-white-39vs5	1482	11	85.88
winequality-white-3_vs_7	winequality-white-3vs7	900	11	66.67

Table 4.13. A comparison of count of wins, ties, and losses for the proposed CGFTSVM-ID with respect to the baseline models over KEEL datasets is shown in the first row of Table 4.14 which is always greater for the proposed CGFTSVM-ID. Thus, the average rank and win, tie, loss statistics show the superiority of the proposed CGFTSVM-ID over the baseline models.

4.3.2.3 Sensitivity analysis of the proposed CGFTSVM-ID on KEEL datasets

The generalization performance of the proposed CGFTSVM-ID model is sensitive to the variation in any pair of hyperparameters λ_1 , λ_2 , σ and a . Figure 4.5 represents the change in AUC with changes in the parameters. Figure 4.5a demonstrates the variation in AUC with λ_1 , λ_2 for bupa dataset.

Table 4.12: The AUC value with training time, sensitivity and specificity value of the proposed CGFTSVM-ID and baseline models on KEEL datasets.

Dataset (samples,features) (ir)	T SVM [22] (AUC,time(seconds)) (Sens.,Spec.)	IFTWSVM [49] (AUC,time(seconds)) (Sens.,Spec.)	IFTWSVM-ID [51] (AUC,time(seconds)) (Sens.,Spec.)	pin-IFTWSVM [111] (AUC,time(seconds)) (Sens.,Spec.)	IFTWSVM [50] (AUC,time(seconds)) (Sens.,Spec.)	ACFSVM [46] (AUC,time(seconds)) (Sens.,Spec.)	ACFTSVM [154] (AUC,time(seconds)) (Sens.,Spec.)	proposed CGFTSVM-ID (AUC,time(seconds)) (Sens.,Spec.)
bupa (345, 6) (1.38)	(0.6478, 0.0248)	(0.6804, 0.049)	(0.7174, 0.076)	(0.6989, 0.0525)	(0.5981, 0.0582)	(0.6367, 0.1523)	(0.6478, 0.0942)	(0.7111, 0.022)
crossplane150 (150, 2) (1)	(0.9783, 0.0034)	(1.00046)	(1.00057)	(1.00121)	(0.9348, 0.0147)	(0.8696, 0.0308)	(1.00181)	(1.00066)
ecoli0147vs2356 (336, 7) (10.59)	(1, 0.9565)	(1, 1)	(1, 1)	(1, 1)	(1, 0.8696)	(1, 0.7391)	(1, 1)	(1, 1)
ecoli0234vs5 (202, 7) (9.1)	(0.6728, 0.0142)	(0.9056, 0.0161)	(0.9228, 0.0182)	(0.8998, 0.016)	(0.8157, 0.0264)	(0.8699, 0.0765)	(0.8699, 0.0358)	(0.8998, 0.0089)
ecoli0267vs35 (224, 7) (9.18)	(0.3571, 0.9885)	(0.8571, 0.954)	(0.8571, 0.9885)	(0.8571, 0.9425)	(0.6429, 0.9885)	(0.7857, 0.954)	(0.7857, 0.954)	(0.8571, 0.9425)
ecoli0347vs56 (257, 7) (9.28)	(0.8061, 0.0017)	(0.7409, 0.0036)	(0.7409, 0.0039)	(0.7227, 0.0119)	(0.8333, 0.0103)	(0.8152, 0.031)	(0.8242, 0.0281)	(0.8242, 0.005)
ecoli067vs35 (222, 7) (9.09)	(0.6667, 0.9455)	(0.5, 0.9818)	(0.5, 0.9818)	(0.5, 0.9455)	(0.6667, 1)	(0.6667, 0.9636)	(0.6667, 0.9818)	(0.6667, 0.9818)
ecoli01vs5 (240, 6) (11)	(0.8125, 0.0037)	(0.9375, 0.0079)	(0.8125, 0.0069)	(0.8125, 0.0051)	(0.7415, 0.0127)	(0.8665, 0.0326)	(0.8496, 0.0206)	(0.9375, 0.0083)
glass5 (214, 9) (22.78)	(0.625, 1)	(0.875, 1)	(0.625, 1)	(0.625, 1)	(0.5, 0.9831)	(0.75, 0.9831)	(0.75, 0.9492)	(0.875, 1)
aus (690, 14) (1.25)	(1, 0.0046)	(0.75, 0.0068)	(1, 0.0081)	(0.9932, 0.0109)	(0.75, 0.0144)	(1, 0.0203)	(0.9932, 0.021)	(1, 0.0042)
brwiconsin (683, 9) (1.86)	(1, 1)	(0.5, 1)	(1, 1)	(1, 0.9863)	(0.5, 1)	(1, 1)	(1, 0.9863)	(1, 1)
checkerboard (690, 14) (1.25)	(0.6729, 0.0014)	(0.8644, 0.0058)	(0.8823, 0.0046)	(0.7183, 0.0131)	(0.7459, 0.0109)	(0.819, 0.0307)	(0.6729, 0.0203)	(0.8912, 0.0069)
cmc (1473, 9) (1.34)	(0.3636, 0.9821)	(0.8182, 0.9107)	(0.8182, 0.9464)	(0.4545, 0.9821)	(0.5455, 0.9464)	(0.7273, 0.9107)	(0.3636, 0.9821)	(0.8182, 0.9643)
iono (351, 33) (1.79)	(0.9929, 0.0024)	(0.9786, 0.0045)	(0.95, 0.0041)	(0.9786, 0.0071)	(0.9714, 0.0085)	(0.9714, 0.0208)	(0.9714, 0.019)	(0.9786, 0.0042)
monk1 (556, 6) (1.04)	(1, 0.9857)	(1, 0.9571)	(1, 0.9)	(1, 0.9571)	(1, 0.9429)	(1, 0.9429)	(1, 0.9429)	(1, 0.9571)
monk2 (601, 7) (1.92)	(0.85, 0.0019)	(0.625, 0.0038)	(0.625, 0.004)	(0.625, 0.0049)	(0.5, 0.0053)	(0.7083, 0.0273)	(0.85, 0.0178)	(0.7333, 0.0067)
monk3 (556, 6) (1.01)	(0.75, 0.95)	(0.25, 1)	(0.25, 1)	(0.25, 1)	(0, 1)	(0.5, 0.9167)	(0.75, 0.95)	(0.5, 0.9667)
segment0 (2308, 19) (6.02)	(0.8687, 0.0229)	(0.8784, 0.0414)	(0.8745, 0.043)	(0.8863, 0.078)	(0.8584, 0.0605)	(0.8548, 0.1809)	(0.8809, 0.2114)	(0.8939, 0.0366)
sonar (208, 60) (1.14)	(0.8557, 0.8818)	(0.866, 0.8909)	(0.8763, 0.8727)	(0.8454, 0.9273)	(0.8351, 0.8818)	(0.7732, 0.9364)	(0.9072, 0.8545)	(0.8969, 0.8909)
vowel (988, 10) (9.98)	(0.9598, 0.0159)	(0.9778, 0.0263)	(0.9778, 0.0322)	(0.9601, 0.0398)	(0.9778, 0.0767)	(0.963, 0.2592)	(0.9558, 0.1465)	(0.9741, 0.0341)
wdbc (194, 33) (3.22)	(0.9481, 0.9714)	(0.9556, 1)	(0.9556, 1)	(0.963, 0.9571)	(0.9556, 1)	(0.9259, 1)	(0.9259, 0.9857)	(0.9481, 1)
yeast2vs8 (483, 8) (23.15)	(0.8687, 0.0193)	(0.8784, 0.029)	(0.8745, 0.041)	(0.8863, 0.0335)	(0.8584, 0.066)	(0.8548, 0.1436)	(0.8809, 0.1313)	(0.8939, 0.0196)
abalone-17vs78910 (2338, 7) (39.31)	(0.8557, 0.8818)	(0.866, 0.8909)	(0.8763, 0.8727)	(0.8454, 0.9273)	(0.8351, 0.8818)	(0.7732, 0.9364)	(0.9072, 0.8545)	(0.8969, 0.8909)
abalone-19vs10111213 (1622, 7) (49.69)	(0.6536, 0.113)	(0.6483, 0.1106)	(0.6557, 0.1102)	(0.6907, 0.1395)	(0.6248, 0.2579)	(0.6419, 0.4752)	(0.6913, 0.7641)	(0.6812, 0.1024)
abalone-20vs8910 (1916, 7) (72.69)	(0.7888, 0.5183)	(0.7888, 0.5079)	(0.8088, 0.5026)	(0.7689, 0.6126)	(0.7052, 0.5445)	(0.6135, 0.6702)	(0.749, 0.6335)	(0.7131, 0.6492)
kr-vs-k-zero_vs_eight (1460, 3) (40.5)	(0.9312, 0.0036)	(0.9161, 0.0099)	(0.9312, 0.008)	(0.8951, 0.0181)	(0.8543, 0.0236)	(0.9114, 0.0593)	(0.9056, 0.0442)	(0.9289, 0.0123)
poker-89vs5 (2075, 10) (32.47)	(0.9394, 0.9231)	(0.9394, 0.9231)	(0.9394, 0.9231)	(0.9697, 0.8205)	(0.9394, 0.7692)	(0.9394, 0.8718)	(0.9394, 0.8718)	(0.9091, 0.9487)
poker-89vs6 (1485, 10) (53.07)	(0.4675, 0.0084)	(0.4534, 0.0166)	(0.4599, 0.027)	(0.4688, 0.0291)	(0.4654, 0.034)	(0.5, 0.0312)	(0.5, 0.0712)	(0.4753, 0.024)
shuttle-2vs5 (3316, 9) (80.22)	(0.5395, 0.3956)	(0.4342, 0.4725)	(0.4474, 0.4725)	(0.4211, 0.5165)	(0.4474, 0.4835)	(1, 0)	(1, 0)	(0.5, 0.4505)
winequality-red-8vs67 (855, 11) (82)	(0.9343, 0.0247)	(0.9032, 0.0174)	(0.9032, 0.0168)	(0.9109, 0.0172)	(0.7336, 0.0394)	(0.93, 0.2117)	(0.9343, 0.1013)	(0.9516, 0.018)
winequality-red-8vs6 (656, 11) (58.4)	(0.9194, 0.9492)	(0.8065, 1)	(0.8065, 1)	(0.8387, 0.9831)	(0.6452, 0.822)	(0.9194, 0.9407)	(0.9194, 0.9492)	(0.9032, 1)
winequality-white-39vs5 (1482, 11) (85.88)	(0.4726, 0.0219)	(0.4433, 0.0267)	(0.4375, 0.0271)	(0.4767, 0.0286)	(0.4454, 0.0556)	(0.4959, 0.0315)	(0.4726, 0.0813)	(0.5129, 0.0235)
winequality-white-3vs7 (900, 11) (66.67)	(0.4051, 0.5402)	(0.2658, 0.6207)	(0.2658, 0.6092)	(0.4937, 0.4598)	(0.0633, 0.8276)	(0.7848, 0.2069)	(0.4051, 0.5402)	(0.8969, 0.8909)
	(0.983, 0.254)	(0.99, 0.9605)	(0.9814, 0.3263)	(0.983, 0.371)	(0.9483, 1.2872)	(0.983, 0.4312)	(0.9814, 0.2668)	(0.983, 0.2825)
	(0.9677, 0.9983)	(1, 0.98)	(0.9677, 0.995)	(0.9677, 0.9983)	(0.9032, 0.9933)	(0.9677, 0.9983)	(0.9677, 0.995)	(0.9677, 0.9983)
	(0.8906, 0.0015)	(0.9063, 0.0056)	(0.8906, 0.0039)	(0.9063, 0.0045)	(0.6292, 0.0088)	(0.7365, 0.01)	(0.8906, 0.016)	(0.9063, 0.0035)
	(0.7813, 1)	(0.8125, 1)	(0.7813, 1)	(0.8125, 1)	(0.625, 0.6333)	(0.9063, 0.5667)	(0.7813, 1)	(0.8125, 1)
	(1, 0.036)	(1, 0.0465)	(1, 0.0495)	(0.9945, 0.0614)	(0.9596, 0.1602)	(0.9792, 0.5026)	(0.9982, 0.2501)	(1, 0.0325)
	(1, 1)	(1, 1)	(1, 1)	(1, 0.989)	(1, 0.9191)	(0.9583, 1)	(1, 0.9963)	(1, 1)
	(0.6979, 0.0053)	(0.6979, 0.0052)	(0.6548, 0.0035)	(0.6979, 0.0059)	(0.5818, 0.0108)	(0.7054, 0.0097)	(0.686, 0.0144)	(0.7098, 0.0034)
	(0.5625, 0.8333)	(0.5625, 0.8333)	(0.5, 0.8095)	(0.5625, 0.8333)	(0.1875, 0.9762)	(0.625, 0.7857)	(0.5625, 0.8095)	(0.5625, 0.8571)
	(0.625, 0.009)	(0.7429, 0.0121)	(0.6108, 0.0182)	(0.6179, 0.0086)	(0.711, 0.0257)	(0.625, 0.028)	(0.625, 0.065)	(0.75, 0.0088)
	(0.25, 1)	(0.5, 0.9858)	(0.25, 0.9716)	(0.25, 0.9858)	(0.5, 0.922)	(0.25, 1)	(0.25, 1)	(0.5, 1)
	(0.8732, 1.0705)	(0.8056, 0.4753)	(0.738, 1.3645)	(0.8708, 0.6744)	(0.6733, 1.5582)	(0.8688, 0.9422)	(0.8732, 3.759)	(0.9035, 0.8151)
	(0.8824, 0.864)	(1, 0.6111)	(0.6471, 0.8289)	(0.8235, 0.9181)	(0.7647, 0.5819)	(0.8824, 0.8553)	(0.8824, 0.864)	(1, 0.807)
	(0.5672, 0.2561)	(0.6931, 0.1427)	(0.6089, 0.1591)	(0.6243, 0.2004)	(0.5074, 0.3197)	(0.6083, 1.9325)	(0.5942, 1.058)	(0.6438, 0.5263)
	(0.1875, 0.9469)	(0.5625, 0.8238)	(0.375, 0.8429)	(0.4375, 0.811)	(1, 0.0149)	(0.4375, 0.7792)	(0.25, 0.9384)	(0.5, 0.7877)
	(0.7003, 0.6691)	(0.7508, 0.2707)	(0.7854, 0.6667)	(0.7677, 0.286)	(0.7007, 0.5471)	(0.7579, 2.8265)	(0.7393, 1.5345)	(0.796, 0.1741)
	(0.4167, 0.984)	(0.5833, 0.9183)	(0.6667, 0.9041)	(0.5833, 0.952)	(1, 0.4014)	(0.5833, 0.9325)	(0.5, 0.9787)	(0.6667, 0.9254)
	(0.8338, 0.1008)	(0.5622, 0.1103)	(0.5681, 0.3963)	(0.5787, 0.3136)	(0.6638, 0.5525)	(0.7929, 0.8691)	(0.7861, 0.6879)	(0.8641, 0.28)
	(0.8, 0.8676)	(0.2, 0.9244)	(0.5333, 0.6028)	(0.5333, 0.6241)	(0.5333, 0.7943)	(0.9333, 0.6525)	(0.6667, 0.9054)	(1, 0.7281)
	(0.5672, 0.2561)	(0.6931, 0.1427)	(0.6089, 0.1591)	(0.6243, 0.2004)	(0.5074, 0.3197)	(0.6083, 1.9325)	(0.5942, 1.058)	(0.6438, 0.5263)
	(0.1875, 0.9469)	(0.5625, 0.8238)	(0.375, 0.8429)	(0.4375, 0.811)	(1, 0.0149)	(0.4375, 0.7792)	(0.25, 0.9384)	(0.5, 0.7877)
	(0.7875, 0.075)	(0.7846, 0.1227)	(0.5, 0.1444)	(0.7143, 0.1341)	(0.5714, 0.2205)	(0.764, 0.7146)	(0.9235, 1.222)	(0.8571, 0.0857)
	(0.7143, 0.8607)	(0.5714, 0.9977)	(0, 1)	(0.4286, 1)	(0.1429, 1)	(0.5714, 0.9566)	(1, 0.847)	(0.7143, 1)
	(1, 0.6343)	(1, 0.862)	(1, 0.8346)	(1, 0.039)	(0.9934, 12.4044)	(1, 12.9723)	(1, 0.9161)	(1, 0.7203)
	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 0.9867)	(1, 1)	(1, 1)	(1, 1)
	(0.4444, 0.0299)	(0.6944, 0.0895)	(0.6984, 0.0834)	(0.7044, 0.0791)	(0.504, 0.0708)	(0.5139, 0.3582)	(0.4702, 0.2158)	(0.8075, 0.056)
	(0.75, 0.6389)	(0.75, 0.6389)	(0.75, 0.6468)	(0.75, 0.6587)	(1, 0.0079)	(0, 0.9405)	(0, 0.9405)	(1, 0.6151)
	(0.5706, 0.0229)	(0.4903, 0.0219)	(0.5635, 0.0218)	(0.4722, 0.0324)	(0.5, 0.0437)	(0.5835, 0.1773)	(0.5887, 0.1292)	(0.8782, 0.0403)
	(0.656, 11) (58.4)	(0.25, 0.7306)	(0.25, 0.6269)	(0.25, 0.6943)	(0, 1)	(0.25, 0.9171)	(0.25, 0.9275)	(1, 0.7565)
	(0.6463, 0.2181)	(0.6939, 0.1781)	(0.636, 0.2921)	(0.7358, 0.1454)	(0.6005, 0.2664)	(0.617, 0.9501)	(0.6761, 0.709)	(0.7439, 0.1694)
	(0.375, 0.9176)	(0.375, 0.8879)	(0.375, 0.897)	(0.625, 0.8467)	(0.375, 0.8261)	(0.25, 0.984)	(0.375, 0.9771)	(0.625, 0.8627)
	(0.8125, 0.028)	(0.642, 0.0825)	(0.6307, 0.0374)	(0.7708, 0.0491)	(0.5871, 0.1292)	(0.6572, 0.3035)	(0.8125, 0.2652)	(0.8883, 0.0484)
	(0.6667, 0.9583)	(0.5, 0.7841)	(0.5, 0.7614)	(0.8333, 0.7083)	(0.6667, 0.5076)	(1, 0.3144)	(0.6667, 0.9583)	(0.8333, 0.9432)

Table 4.13: The average AUC, average rank, average G-mean, average F-measure, average sensitivity, average specificity and average training time of the proposed CGFTSVM-ID and baseline models on KEEL and UCI datasets.

Dataset	metric	TSVM [22]	IFTWSVM [49]	IFTWSVM-ID [51]	pin-IFTWSVM [111]	IIFTWSVM [50]	ACFSVM [46]	ACFTSVM [154]	proposed CGFTSVM-ID
KEEL	Average AUC	0.7754	0.7769	0.7641	0.7728	0.7022	0.7694	0.7914	0.8344
	Average Rank	4.56	4.2	4.82	4.44	6.77	4.88	4.24	2.09
	Average G-mean	0.7347	0.7593	0.732	0.7601	0.613	0.7318	0.7406	0.831
	Average F-measure	0.5859	0.5763	0.5483	0.5732	0.4583	0.5686	0.5773	0.6357
	Average Sens.	0.6605	0.6869	0.6627	0.6893	0.6479	0.7215	0.6731	0.7999
	Average Spec.	0.8904	0.8724	0.8671	0.8675	0.7672	0.8245	0.9111	0.8766
	Average time (seconds)	0.1198	0.1484	0.1514	0.1749	0.5667	1.0902	0.7462	0.1247
UCI	Average AUC	0.7892	0.7889	0.7849	0.7764	0.7186	0.762	0.7998	0.8098
	Average Rank	4.06	4.11	4.67	4.72	6.83	6.03	3.56	2.03
	Average G-mean	0.7805	0.7783	0.7766	0.7675	0.6931	0.7471	0.7921	0.8048
	Average F-measure	0.7558	0.7573	0.7537	0.7422	0.6693	0.733	0.762	0.7768
	Average Sens.	0.7873	0.8084	0.7918	0.7734	0.6886	0.7803	0.814	0.8153
	Average Spec.	0.791	0.7693	0.7781	0.7794	0.7487	0.7436	0.7857	0.8043
	Average time (seconds)	0.02	0.0266	0.0252	0.1352	0.032	0.1752	0.1255	0.0227

Table 4.14: The performance evaluation of the proposed CGFTSVM-ID and baseline models in terms of win-tie-loss. The notation [win-tie-loss] indicates the count of wins, ties, and losses of the proposed CGFTSVM-ID over the respective column model.

Dataset	TSVM [22]	IFTWSVM [49]	IFTWSVM-ID [51]	pin-IFTWSVM [111]	IIFTWSVM [50]	ACFSVM [46]	ACFTSVM [154]
KEEL	[26 4 3]	[22 6 5]	[25 4 4]	[26 6 1]	[31 0 2]	[29 3 1]	[26 3 4]

Table 4.15: AUC with training time, sensitivity, specificity, G-mean and F-measure of the proposed CGFTSVM-ID and baseline models on UCI datasets.

Dataset (ir) (samples,features)	TSVM [22] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	IFTWSVM [49] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	IFTWSVM-ID [51] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	pin-IFTWSVM [111] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	IIFTWSVM [50] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	ACFSVM [46] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	ACFTSVM [154] (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)	proposed CGFTSVM-ID (AUC,time(seconds)) (Sens.,Spec.) (G-mean,F-measure)
bc	(0.6874, 0.008) (0.5652, 0.8095) (0.5417, 0.6764)	(0.7464, 0.0237) (0.8261, 0.6667) (0.6032, 0.7421)	(0.6794, 0.0198) (0.5652, 0.7937) (0.5306, 0.6698)	(0.6439, 0.0677) (0.4783, 0.8095) (0.4783, 0.6222)	(0.7012, 0.0284) (0.6087, 0.7937) (0.56, 0.695)	(0.638, 0.0555) (0.4348, 0.8413) (0.4651, 0.6048)	(0.6418, 0.0614) (0.5217, 0.7619) (0.48, 0.6305)	(0.7305, 0.016) (0.8261, 0.6349) (0.5846, 0.7242)
bcw	(0.961, 0.9699) (0.9548, 0.9655) (0.5953, 0.0099)	(0.9812, 0.0396) (1, 0.9624) (0.9686, 0.981)	(0.9525, 0.0322) (0.9351, 0.9699) (0.9412, 0.9523)	(0.9655, 0.0808) (0.961, 0.9699) (0.9548, 0.9655)	(0.9607, 0.0332) (0.974, 0.9474) (0.9434, 0.9606)	(0.9433, 0.2339) (0.9091, 0.9774) (0.9333, 0.9426)	(0.9487, 0.1481) (0.9351, 0.9624) (0.9351, 0.9486)	(0.972, 0.0474) (0.974, 0.9699) (0.9615, 0.972)
cv	(0.8393, 0.3514) (0.6225, 0.543) (0.8675, 0.0151)	(0.9107, 0.3108) (0.6456, 0.532) (0.8488, 0.0261)	(0.8929, 0.3784) (0.6579, 0.5812) (0.8585, 0.0233)	(0.8036, 0.4595) (0.6383, 0.6076) (0.8585, 0.1065)	(0.5678, 0.0162) (0.9464, 0.1892) (0.6272, 0.4231)	(0.5804, 0.0809) (0.6607, 0.5) (0.5692, 0.5748)	(0.6243, 0.0626) (0.9107, 0.3378) (0.6538, 0.5547)	(0.6404, 0.0142) (0.8214, 0.4595) (0.6479, 0.6143)
ca	(0.8288, 0.9063) (0.8679, 0.8667) (0.6658, 0.014)	(0.8018, 0.8958) (0.8476, 0.8475) (0.7086, 0.0152)	(0.8108, 0.9063) (0.8571, 0.8572) (0.6925, 0.0128)	(0.8108, 0.9063) (0.8571, 0.8572) (0.7007, 0.0552)	(0.7748, 0.9271) (0.8431, 0.8475) (0.5757, 0.0261)	(0.8919, 0.8021) (0.8646, 0.8458) (0.6394, 0.0327)	(0.8018, 0.9271) (0.8599, 0.8622) (0.7461, 0.0816)	(0.8198, 0.9063) (0.8626, 0.862) (0.7142, 0.0234)
cb	(0.8478, 0.4839) (0.7723, 0.6405) (0.7727, 0.0011)	(0.8043, 0.6129) (0.7789, 0.7021) (0.7825, 0.002)	(0.8043, 0.5806) (0.7708, 0.6834) (0.8182, 0.0021)	(0.837, 0.5645) (0.7857, 0.6874) (0.7549, 0.0103)	(0.587, 0.5645) (0.6243, 0.5756) (0.6364, 0.0029)	(0.9239, 0.3548) (0.7834, 0.5726) (0.7273, 0.0121)	(0.7826, 0.7097) (0.7912, 0.7453) (0.737, 0.0115)	(0.8478, 0.5806) (0.7959, 0.7016) (0.8182, 0.0029)
echo	(0.5455, 1) (0.7059, 0.7385) (0.6838, 0.0248)	(0.6364, 0.9286) (0.7, 0.7687) (0.6851, 0.0161)	(0.6364, 1) (0.7778, 0.7977) (0.6699, 0.0191)	(0.5455, 0.9643) (0.6667, 0.7252) (0.6536, 0.0768)	(0.2727, 1) (0.4286, 0.5222) (0.6571, 0.0213)	(0.4545, 1) (0.625, 0.6742) (0.6343, 0.2428)	(0.5455, 0.9286) (0.6316, 0.7117) (0.6536, 0.0897)	(0.6364, 1) (0.7778, 0.7977) (0.6894, 0.0148)
ii1	(0.766, 0.6016) (0.5373, 0.6788) (0.9783, 0.0039)	(0.8936, 0.4766) (0.5385, 0.6526) (0.936, 0.0105)	(0.8085, 0.5313) (0.5241, 0.6554) (0.929, 0.0094)	(0.7447, 0.5625) (0.5072, 0.6472) (0.9571, 0.0269)	(0.8298, 0.4844) (0.5132, 0.634) (0.8631, 0.0094)	(0.8936, 0.375) (0.497, 0.5789) (0.8944, 0.0472)	(0.7447, 0.5625) (0.5072, 0.6472) (0.9501, 0.007)	(0.8085, 0.5703) (0.5429, 0.679) (0.9501, 0.007)
iono	(0.9859, 0.9706) (0.9859, 0.9782) (0.809, 0.0341)	(0.9014, 0.9706) (0.9412, 0.9354) (0.809, 0.0446)	(0.8873, 0.9706) (0.9333, 0.928) (0.809, 0.0501)	(0.9437, 0.9706) (0.964, 0.957) (0.816, 0.9838)	(0.8732, 0.8529) (0.8986, 0.863) (0.7917, 0.0607)	(0.7887, 1) (0.8819, 0.8881) (0.809, 0.6384)	(0.9296, 0.9706) (0.9565, 0.9499) (0.8403, 0.2566)	(0.9296, 0.9706) (0.9565, 0.9499) (0.816, 0.0533)
mammo	(0.7083, 0.9097) (0.7876, 0.8027) (0.7267, 0.001)	(0.7431, 0.875) (0.7955, 0.8063) (0.7267, 0.0016)	(0.7292, 0.8889) (0.7925, 0.8051) (0.7004, 0.0025)	(0.8264, 0.8056) (0.8179, 0.8159) (0.7267, 0.0143)	(0.8125, 0.7708) (0.7959, 0.7914) (0.753, 0.0027)	(0.8125, 0.8056) (0.8097, 0.809) (0.753, 0.0083)	(0.8264, 0.8542) (0.838, 0.8402) (0.7267, 0.0073)	(0.7083, 0.9236) (0.7938, 0.8088) (0.753, 0.002)
mbp	(0.7692, 0.6842) (0.6897, 0.7255) (0.7985, 0.0189)	(0.7692, 0.6842) (0.6897, 0.7255) (0.7348, 0.0182)	(0.7692, 0.6316) (0.6667, 0.697) (0.7191, 0.0186)	(0.7692, 0.6842) (0.6897, 0.7255) (0.6568, 0.0801)	(0.7143, 0.7529) (0.7143, 0.7529) (0.6232, 0.0174)	(0.7692, 0.7368) (0.7143, 0.7529) (0.8253, 0.1896)	(0.7692, 0.6842) (0.6897, 0.7255) (0.7981, 0.0924)	(0.7692, 0.7368) (0.7143, 0.7529) (0.7947, 0.0236)
monks2	(0.7581, 0.839) (0.7344, 0.7975) (0.9577, 0.0066)	(0.5968, 0.8729) (0.6491, 0.7217) (0.93, 0.0107)	(0.5484, 0.8898) (0.6239, 0.6986) (0.93, 0.0123)	(0.5, 0.8136) (0.5391, 0.6378) (0.8955, 0.0469)	(0.3226, 0.9237) (0.4396, 0.5459) (0.7062, 0.0106)	(0.871, 0.7797) (0.7606, 0.8241) (0.821, 0.0185)	(0.7742, 0.822) (0.7328, 0.7978) (0.9577, 0.0604)	(0.7419, 0.8475) (0.7302, 0.7929) (0.9577, 0.0093)
musk1	(0.9429, 0.9726) (0.9565, 0.9576) (0.8132, 0.0645)	(0.9286, 0.9315) (0.9286, 0.93) (0.805, 0.0785)	(0.9286, 0.9315) (0.9286, 0.93) (0.7915, 0.0765)	(0.9143, 0.8767) (0.8951, 0.8953) (0.8125, 0.2315)	(0.7, 0.7123) (0.7, 0.7061) (0.6905, 0.1026)	(0.9571, 0.6849) (0.8375, 0.8097) (0.7122, 0.4857)	(0.9429, 0.9726) (0.9565, 0.9576) (0.8432, 0.3242)	(0.9429, 0.9726) (0.9565, 0.9576) (0.8189, 0.0467)
omn_4d	(0.8641, 0.7624) (0.8725, 0.8116) (0.8369, 0.0342)	(0.9466, 0.6634) (0.8966, 0.7924) (0.7843, 0.0546)	(0.8107, 0.7723) (0.8434, 0.7912) (0.7859, 0.0565)	(0.9417, 0.6832) (0.8981, 0.8021) (0.7693, 0.1993)	(0.7573, 0.6238) (0.78, 0.6873) (0.6507, 0.0629)	(0.8204, 0.604) (0.8145, 0.7039) (0.7536, 0.5118)	(0.835, 0.8515) (0.8753, 0.8432) (0.8119, 0.2639)	(0.7864, 0.8515) (0.846, 0.8183) (0.8199, 0.038)
otn_2f	(0.908, 0.7658) (0.8783, 0.8338) (0.6223, 0.002)	(0.9018, 0.6667) (0.8473, 0.7754) (0.6323, 0.007)	(0.9141, 0.6577) (0.8514, 0.7754) (0.6579, 0.0051)	(0.908, 0.6306) (0.8409, 0.7567) (0.6948, 0.0185)	(0.6258, 0.6757) (0.6777, 0.6502) (0.6137, 0.0051)	(0.8405, 0.6667) (0.8131, 0.7486) (0.6905, 0.0133)	(0.8221, 0.8018) (0.8401, 0.8119) (0.7354, 0.0224)	(0.865, 0.7748) (0.8571, 0.8187) (0.7104, 0.0029)
spect	(0.5, 0.7447) (0.5333, 0.6102) (0.6752, 0.0676)	(0.5625, 0.7021) (0.5625, 0.6284) (0.6779, 0.0588)	(0.6563, 0.6596) (0.6087, 0.6579) (0.6991, 0.0544)	(0.6875, 0.7021) (0.6471, 0.6948) (0.6607, 0.1781)	(0.3125, 0.9149) (0.4348, 0.5347) (0.5557, 0.0753)	(0.5938, 0.7872) (0.623, 0.6837) (0.6802, 0.091)	(0.875, 0.5957) (0.7089, 0.722) (0.677, 0.2849)	(0.7188, 0.7021) (0.6667, 0.7104) (0.7017, 0.0418)
sgc	(0.6316, 0.7188) (0.5134, 0.6738) (0.9895, 0.0275)	(0.5789, 0.7768) (0.5176, 0.6706) (0.9947, 0.0567)	(0.6974, 0.7009) (0.5408, 0.6991) (0.9789, 0.0445)	(0.6974, 0.7009) (0.4935, 0.6409) (0.9947, 0.2024)	(0.5, 0.8214) (0.3805, 0.5541) (0.9421, 0.0587)	(0.5132, 0.5982) (0.5212, 0.6705) (0.9684, 0.3783)	(0.6711, 0.683) (0.5152, 0.677) (0.9895, 0.2741)	(0.6579, 0.7455) (0.5464, 0.7003) (1, 0.0387)
tic_tac_toe	(1, 0.9789) (0.9948, 0.9894) (0.7596, 0.0094)	(1, 0.9895) (0.9974, 0.9947) (0.8058, 0.0064)	(1, 0.9579) (0.9897, 0.9787) (0.8209, 0.0055)	(1, 0.9895) (0.9974, 0.9947) (0.7827, 0.0215)	(1, 0.8842) (0.9722, 0.9403) (0.7956, 0.0091)	(1, 0.9368) (0.9846, 0.9679) (0.7978, 0.0535)	(1, 0.9789) (0.9948, 0.9894) (0.8514, 0.0291)	(1, 1) (1, 1) (0.8261, 0.0074)
vc_2classes	(0.75, 0.7692) (0.6563, 0.7596)	(0.75, 0.8615) (0.7241, 0.8038)	(0.8571, 0.7846) (0.7273, 0.8201)	(0.75, 0.8154) (0.6885, 0.782)	(0.7143, 0.8769) (0.7143, 0.7914)	(0.8571, 0.7385) (0.6957, 0.7956)	(0.9643, 0.7385) (0.75, 0.8439)	(0.8214, 0.8308) (0.7419, 0.8261)

Table 4.16: Optimal parameters corresponding to AUC values of the proposed CGFTSVM-ID and baseline models on UCI datasets.

Dataset	TSVM [22] (λ_1, σ)	IFTWSVM [49] $(\lambda_1, \lambda_2, \sigma)$	IFTWSVM-ID [51] $(\lambda_1, \lambda_2, \sigma)$	pin-IFTWSVM [111] $(\lambda_1, \lambda_2, \sigma, \tau, \epsilon)$	IIFTWSVM [50] $(\lambda_1, \lambda_2, \sigma)$	ACFSVM [46] (λ_1, σ)	ACFTSVM [154] (λ_1, σ)	proposed CGFTSVM-ID $((\lambda_1, \lambda_2, \sigma, a))$
bc	(1, 8)	(1000, 10, 32)	(1000, 0.00001, 2)	(0.1, 0.1, 8, 0, 0.01)	(10, 10, 8)	(10, 0.25)	(0.01, 2)	(1, 0.00001, 32, 1)
bcw	(0.00001, 8)	(0.01, 0.00001, 32)	(10, 0.00001, 2)	(0.001, 0.01, 32, 0.2, 1)	(10, 10, 32)	(0.1, 0.5)	(10, 16)	(0.1, 1, 16, 0.5)
cv	(100, 32)	(0.001, 0.01, 8)	(0.0001, 0.01, 8)	(0.0001, 0.1, 8, 0, 0.5)	(0.00001, 10, 0.03125)	(1, 0.5)	(0.001, 16)	(0.00001, 1, 8, 0.5)
ca	(0.01, 16)	(1, 0.1, 8)	(0.1, 0.00001, 16)	(1, 0.1, 8, 0, 0.01)	(0.00001, 0.00001, 2)	(10, 0.25)	(1, 32)	(0.0001, 0.01, 16, 0.5)
cb	(1, 16)	(0.01, 1, 4)	(0.01, 1, 4)	(0.00001, 0.1, 16, 0.2, 1)	(10, 100, 4)	(10, 0.5)	(1, 16)	(0.0001, 1, 16, 1)
echo	(0.001, 4)	(0.00001, 0.01, 8)	(0.00001, 0.001, 4)	(0.001, 0.01, 8, 0, 1)	(10, 100, 16)	(10, 0.5)	(0.01, 4)	(0.0001, 10, 8, 0.5)
iil	(10, 32)	(0.0001, 0.01, 16)	(0.1, 0.1, 8)	(0.1, 0.1, 8, 0.05, 0.01)	(1, 100, 8)	(1000, 0.5)	(1, 4)	(0.00001, 0.00001, 4, 0.625)
iono	(0.001, 2)	(1, 1000, 4)	(1, 10000, 2)	(0.0001, 0.1, 2, 0.5, 0.01)	(10, 1, 4)	(10, 1)	(0.00001, 2)	(0.1, 1, 2, 0.5)
mammo	(0.1, 4)	(0.0001, 0.00001, 4)	(0.0001, 0.00001, 4)	(0.1, 1, 4, 0.5, 0.01)	(1, 10, 2)	(10, 1)	(10, 16)	(0.0001, 1, 4, 0.5)
mbp	(0.0001, 4)	(0.01, 0.00001, 32)	(0.01, 0.00001, 32)	(0.0001, 1, 32, 0.5, 0.01)	(100, 10, 8)	(10, 0.125)	(0.001, 4)	(1, 0.00001, 8, 0.5)
monks2	(10, 2)	(0.01, 1000, 1)	(0.01, 100, 1)	(0.00001, 0.1, 1, 0.2, 0.01)	(100, 10, 1)	(1000, 1)	(1, 2)	(0.001, 10, 2, 0.5)
muskl	(0.001, 8)	(0.00001, 0.001, 8)	(0.00001, 0.001, 8)	(0.001, 0.1, 8, 0, 0.01)	(1, 1, 8)	(10, 0.25)	(0.01, 8)	(0.01, 1, 8, 1)
omn_4d	(100, 8)	(0.00001, 1, 16)	(0.0001, 1, 8)	(0.00001, 1, 16, 0, 0.01)	(0.00001, 0.00001, 1)	(10, 1)	(1, 8)	(0.00001, 0.1, 8, 0.875)
otn_2f	(0.1, 4)	(0.00001, 1, 8)	(0.00001, 1, 8)	(0.00001, 1, 16, 0.05, 0.1)	(0.00001, 0.00001, 1)	(10, 1)	(10, 8)	(0.01, 1, 4, 0.5)
spect	(0.00001, 16)	(0.01, 1, 32)	(10, 0.00001, 8)	(1, 1, 4, 0.2, 0.5)	(10, 100, 4)	(100, 0.25)	(0.1, 32)	(0.1, 0.00001, 16, 0.5)
sgc	(10, 32)	(0.1, 0.00001, 32)	(0.00001, 0.01, 16)	(0.1, 0.1, 32, 0.5, 0.01)	(1, 100, 4)	(10, 0.25)	(0.00001, 16)	(0.01, 0.1, 8, 1)
tic_tac_toe	(0.001, 2)	(1, 10000, 1)	(0.01, 1, 2)	(0.01, 1, 1, 0, 0.01)	(0.00001, 0.00001, 1)	(1, 1)	(0.001, 2)	(1, 0.1, 1, 0.5)
vc_2classes	(100, 32)	(0.001, 0.1, 8)	(0.0001, 0.1, 16)	(0.01, 1, 4, 0.5, 0.1)	(1, 10, 1)	(1, 1)	(1, 8)	(0.1, 10, 4, 0.5)

Table 4.17: The performance evaluation of the proposed CGFTSVM-ID and baseline models in terms of win-tie-loss. The notation [win-tie-loss] indicates the count of wins, ties, and losses of the proposed CGFTSVM-ID over the respective column model.

Dataset	TSVM [22]	IFTWSVM [49]	IFTWSVM-ID [51]	pin-IFTWSVM [111]	IIFTWSVM [50]	ACFSVM [46]	ACFTSVM [154]
UCI	[13 1 3]	[16 0 2]	[17 1 0]	[16 1 1]	[17 1 0]	[16 1 1]	[9 2 7]

Table 4.18: Nemenyi post hoc significant difference between the baseline column models and the proposed CGFTSVM-ID across UCI and KEEL datasets.

Dataset	Significance	TSVM [22]	IFTWSVM [49]	IFTWSVM-ID [51]	pin-IFTWSVM [111]	IIFTWSVM [50]	ACFSVM [46]	ACFTSVM [154]
KEEL	proposed CGFTSVM-ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes
UCI	proposed CGFTSVM-ID	No	No	Yes	Yes	Yes	Yes	No

Table 4.19: Wilcoxon-signed rank test of proposed CGFTSVM-ID with respect to baseline models on KEEL and UCI datasets.

Model	R^+	R^-	p-value	Hypothesis
KEEL dataset				
TSVM [22]	406.5	28.5	< 0.00001	Rejected
IFTWSVM [49]	341	37	0.00026	Rejected
IFTWSVM-ID [51]	415	20	< 0.00001	Rejected
pin-IFTWSVM [111]	371	7	< 0.00001	Rejected
IIFTWSVM [50]	556	5	< 0.00001	Rejected
ACFSVM [46]	456	9	< 0.00001	Rejected
ACFTSVM [154]	400	65	0.00056	Rejected
UCI dataset				
TSVM [22]	131	22	0.00988	Rejected
IFTWSVM [49]	157	14	0.00188	Rejected
IFTWSVM-ID [51]	153	0	0.0003	Rejected
pin-IFTWSVM [111]	148	5	0.00072	Rejected
IIFTWSVM [50]	153	0	0.0003	Rejected
ACFSVM [46]	146	7	0.001	Rejected
ACFTSVM [154]	84	52	0.40654	Not rejected

Figure 4.5b shows the sensitivity of AUC with σ and a for *ecoli-0-2-3-4_vs_5* dataset. Figures 4.5c and 4.5d illustrates the sensitivity curve with λ_1 and a for datasets *ecoli-0-3-4-7_vs_5-6* and *ecoli-0-6-7_vs_3-5*, respectively. Hence, the performance of the proposed CGFTSVM-ID model gets affected by pairwise variation in the parameters.

The performances on 18 UCI datasets are shown in Table 4.15 and Table 4.13 depicts the average AUC and rank over the 18 datasets. The average AUC values of the baseline models over UCI datasets are 0.7998, 0.7892, 0.7889, 0.7849, 0.7764, 0.762, 0.7804, for ACFTSVM, TSVM, IFTWSVM, IFTWSVM-ID, pin-IFTWSVM, IIFTWSVM

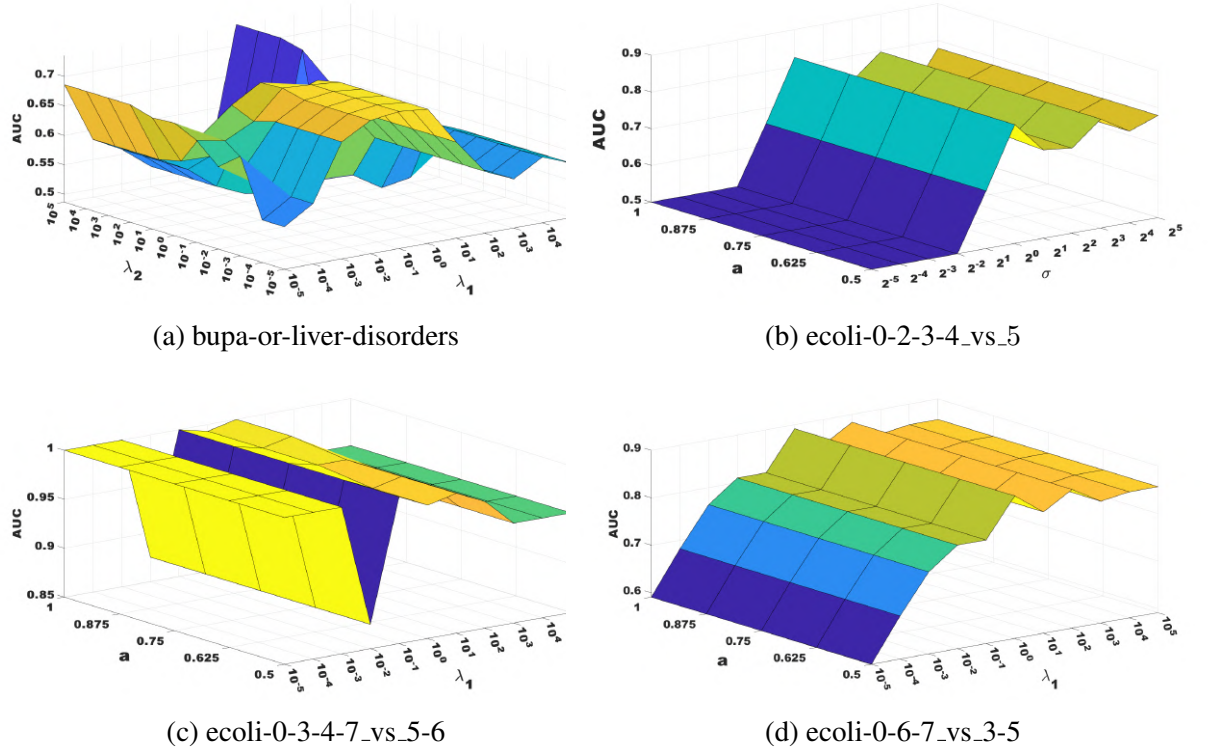


Figure 4.5: Evaluation of AUC values of proposed CGFTSVM-ID while varying the parameters λ_1 , λ_2 , σ and a .

and ACFSVM, respectively, whereas the proposed CGFTSVM-ID has average AUC value 0.8098. Hence, CGFTSVM-ID has better classification performance over UCI datasets than baseline models. The mean rank of the proposed CGFTSVM-ID is 2.03, which is the least, followed by ACFSVM with value 3.56. The average rank of TSVM, IFTWSVM, IFTWSVM-ID, pin-IFTWSVM, IIFTWSVM and ACFSVM are 4.06, 4.11, 4.67, 4.72, 6.83, and 6.03, respectively. Table 4.16 consists of the optimal parameters corresponding to maximum AUC values over UCI datasets. Figure 4.6 depicts the pictorial comparison of the sensitivity metric of the baseline models with proposed CGFTSVM-ID on UCI datasets. Through the last row of Table 4.17, we demonstrate the count of wins, ties and losses of the proposed CGFTSVM-ID with respect to the baseline models over UCI datasets.

In order to experimentally verify the limitation of membership of ACFSVM [46] dis-

cussed in subsection 2.3.6 with respect to the proposed CPGB function, we carried out numerical experiments on the UCI datasets with their optimal parameters for the ACFSVM and proposed CGFTSVM-ID. The training time (in seconds) obtained for each dataset is demonstrated in Figure 4.7a which clearly depicts the proposed model is more efficient in terms of training time than ACFSVM.

4.3.2.4 Statistical analysis over real-world KEEL datasets

In this subsection, we statistically analyse the performance of the models over KEEL datasets by conducting the Friedman, Nemenyi-post-hoc and Wilcoxon signed rank test [123].

The null hypothesis of the Friedman test hypothesize that the models are equivalent and possess identical average ranks. Friedman statistic is calculated as shown in equation 2.56 and has the chi-squared (χ_F^2) distribution. To refrain from the undesirably conservative nature of Friedman statistic, F_F statistic is calculated as shown in equation (2.57) which follows F -distribution. In our case, we have ($md =$) 8 models executed over ($N =$) 33 real-world KEEL datasets, thus $\chi_F^2 = 62.5493$ and $F_F = 11.8823$ with degree of freedom 7 and (7, 224), respectively. At 5% level of significance, $F_{(7,224)} = 2.0506$ which is less than calculated F_F , hence null hypothesis is rejected and models under comparison are not equivalent.

To further strengthen the statistical analysis, we conduct the Nemenyi-post-hoc test which includes pairwise comparison of the models. According to this test, if the difference in mean ranks between the two models exceeds the critical difference (cd), they are deemed to be significantly different. The cd is calculated as shown in equation (2.58), where $q_\alpha = 3.031$ for 8 classifiers [123]. The calculated value of cd with $\alpha = 0.05$ is 1.8278. Table 4.18 first row demonstrates the significant difference of the proposed CGFTSVM-ID with

respect to the baseline models. It is clear from Table 4.18 that all the baseline models are significantly different from the proposed CGFTSVM-ID.

Further, we also conduct the Wilcoxon-signed rank test, a pairwise test with the null hypothesis assumption of the equivalence of the two models. It ranks the differences in the performances of the two models depending on their magnitude. Let R^+ and R^- denote the sum of positive and negative ranks, respectively. Table 4.19 first part depicts the Wilcoxon signed rank tests of the performance of the proposed and baseline algorithms over KEEL datasets. If the p -value of an algorithm is less than 0.05, then the null hypothesis assumption is rejected. Thus, Table 4.19 first part shows the baseline models are not equivalent to the proposed CGFTSVM-ID.

4.3.2.5 Statistical analysis over real-world UCI datasets

To further demonstrate the superiority of the proposed CGFTSVM-ID model, we perform statistical analysis on the results over UCI datasets. On simple calculations for $N = 18$ and $md = 8$, we get Friedman statistics $\chi_F^2 = 45.8019$ and $F_F = 9.7089$ with degree of freedom 7 and (7, 119), respectively. At 5% level of significance $F_{(7,119)} = 2.1713$ which is less than calculated F_F , thus, the null hypothesis assumption is rejected leading to non-equivalence of the models. Now, we employ the pairwise Nemenyi-post-hoc test and Wilcoxon test. The calculated cd value for $\alpha = 0.05$ is 2.0874. Table 4.18 second row shows all the models except TSVM, IFTWSVM and ACFTSVM are significantly different from the proposed CGFTSVM-ID. However, other statistical analysis and numerical experiments demonstrate the superiority of the proposed CGFTSVM-ID with respect to TSVM, IFTWSVM and ACFTSVM. Table 4.19 second part contains the results of the Wilcoxon test over UCI datasets which clearly demonstrate the proposed CGFTSVM-ID and baseline models except ACFTSVM are not equivalent. The non-equivalence of ACFTSVM with the

proposed CGFTSVM-ID is shown by other tests.

Hence, through the numerical experiments and statistical analysis of real-world datasets, we conclude that the proposed CGFTSVM-ID model is superior to the baseline models TSVM, IFTWSVM, IFTWSVM-ID, pin-IFTWSVM, IIFTWSVM, ACFSVM and ACFTSVM.

4.3.2.6 Influence of parameters on the performance of the proposed CGFTSVM-ID

The non-linear case of the proposed CGFTSVM-ID has four parameters that have to be pre-specified: the hyperparameters λ_1 , λ_2 , the membership function parameter a and the Gaussian kernel parameter σ . To demonstrate the effect of each parameter on the generalization performance of the proposed CGFTSVM-ID, we carried out experiments on four randomly selected KEEL datasets cmc, monks1, monks3 and yeast2vs8. While carrying out the investigations, we implemented grid search on the respective parameters whose effect is to be determined and the other parameters are fixed at their optimal values. The values of parameters λ_1 , λ_2 and σ are same as given in experimental setup and $a \in [0.5, 0.55, \dots, 0.95, 1]$. Figures 4.7b-4.7f represent the effect of parameters on AUC values, which clearly demonstrate that the parameters significantly effect the AUC values of the proposed CGFTSVM-ID. Thus, the generalization performance of the proposed model is sensitive to the choice of optimal parameters.

4.3.2.7 Choice of steepness parameter r in the proposed CPGB membership function

To determine the optimal steepness parameter (r) of the proposed CGFTSVM-ID, we conducted numerical experiments on several real-world KEEL datasets, including bupa, crossplane150, ecoli0147vs2356, ecoli0234vs5, ecoli0267vs35, ecoli0347vs56, ecoli067vs35, ecoli01vs5, and glass5. We systematically tuned the parameter within the

range of $[a, a/2, a/4, a/8]$. For most datasets, the optimal value for the steepness parameter (r) was found to be ' a ', except bupa, where the optimal value was determined as ' $a/4$ '. Importantly, the variation in the steepness parameter of bupa from ' a ' to ' $a/4$ ' did not lead to any significant change in the testing AUC value. Consequently, for the sake of consistency, we conducted all numerical experiments using a fixed value of $r = a$.

4.3.3 Application

In this section, we examine the application of the proposed CGFTSVM-ID in real world by conducting numerical experiments on ADNI datasets (www.adni-info.org) which is publicly available. It was launched in 2003 by its Principal Investigator Michael W. Weiner to analyze different neuroimaging techniques for the diagnosis of AD from mild cognitive impairment (MCI). From the ADNI database, we downloaded and processed 150 T1-weighted images using the technique outlined in [144]. Subsequently, we employed our proposed CGFTSVM-ID and baseline models for the classification tasks involving AD versus mild cognitive impairment (MCI) subjects, AD subjects versus control normal (CN) subjects, and MCI versus CN subjects.

4.3.3.1 Experimental discussion on AD dataset

The numerical experiments are employed on the proposed CGFTSVM-ID and baseline models over ADNI datasets with their performance shown in Table 4.20. For the CN versus AD case, ACFTSVM is the best-performing model with AUC 0.9128, pin-IFTWSVM with AUC 0.901 is the second best model, followed by the proposed CGFTSVM-ID with AUC 0.8962. For the CN versus MCI classification, the proposed model lies at the fourth number in performance having AUC 0.6776 with ACFTSVM, IFTWSVM-ID and ACFSVM being at the first, second and third number having AUC 0.7071, 0.7006 and 0.6939,

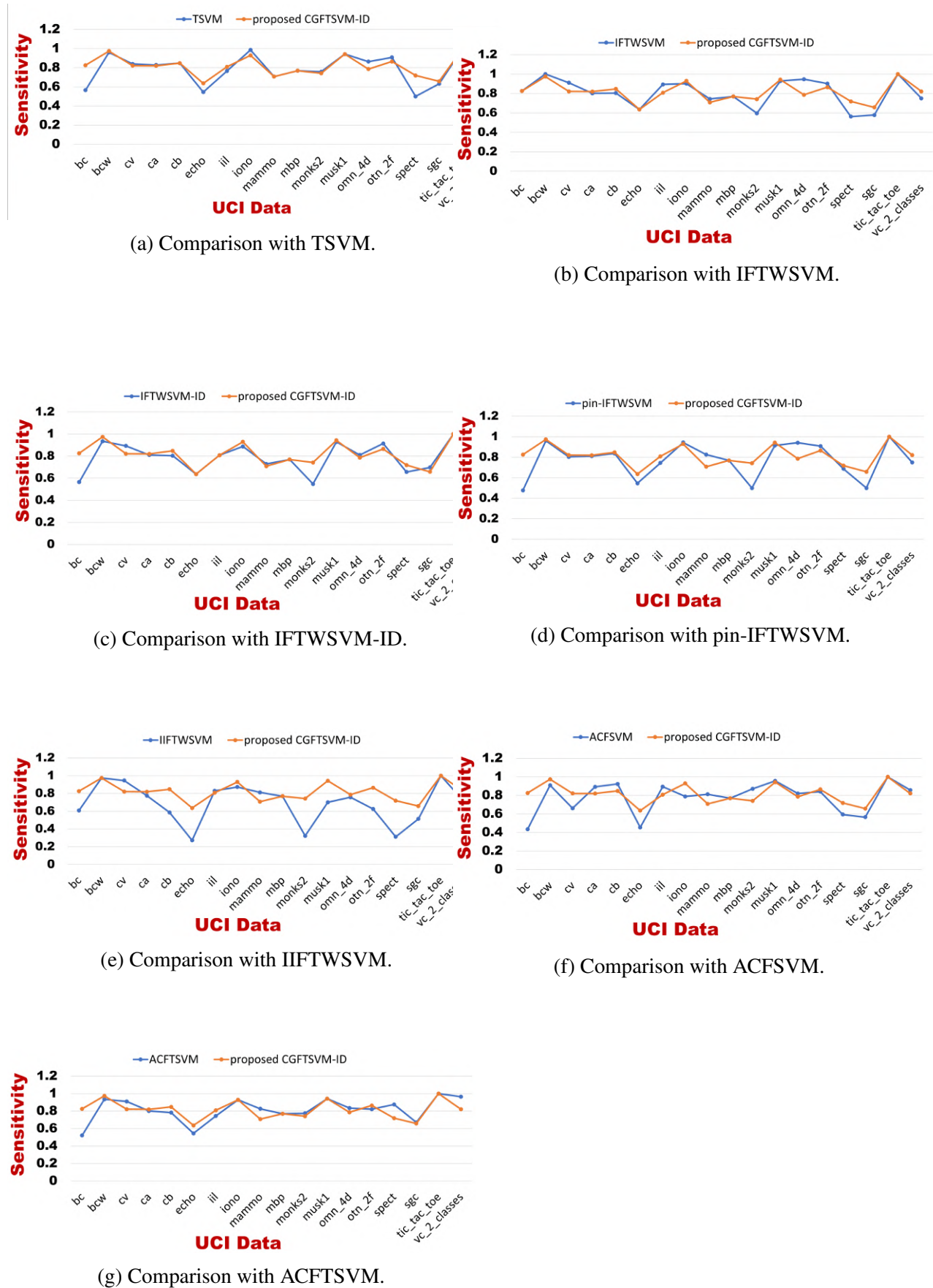
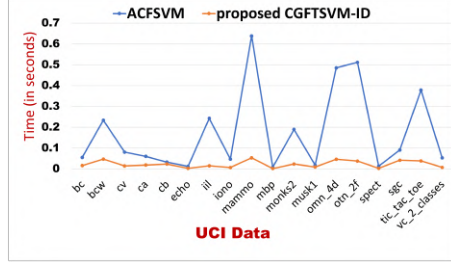
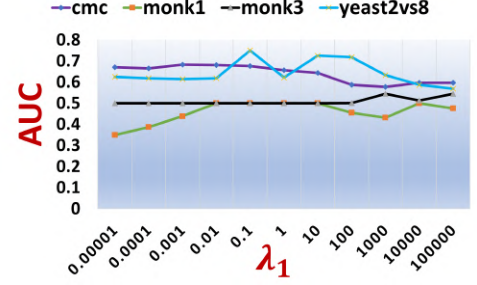


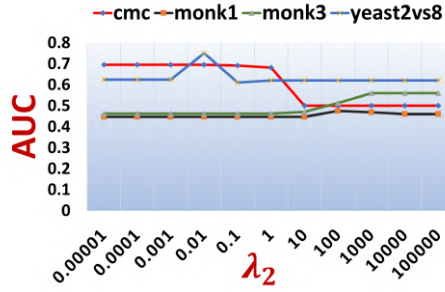
Figure 4.6: Representation of sensitivity metric of proposed CGFTSVM-ID and baseline models on UCI datasets.



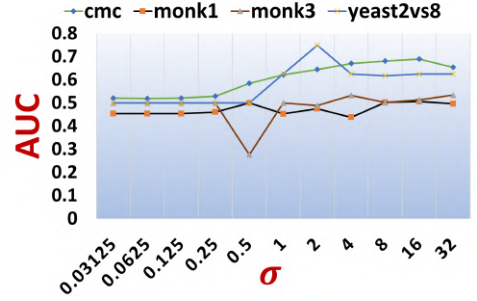
(a)



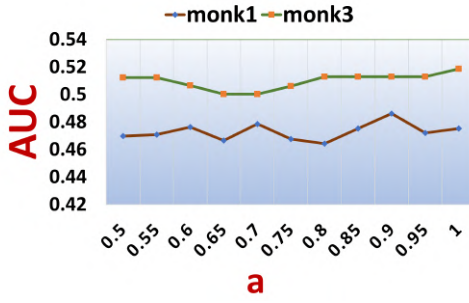
(b)



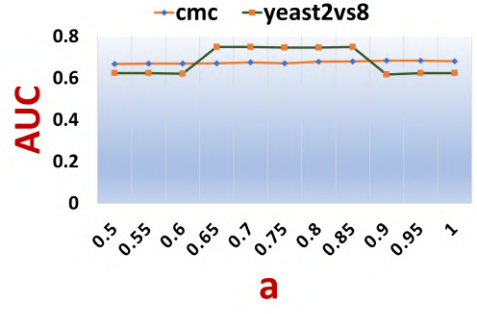
(c)



(d)



(e)



(f)

Figure 4.7: (a) represents training time of ACFSVM and proposed CGFTSVM-ID. Fig (b)-(f) represents the effect of parameters on AUC values of the proposed CGFTSVM-ID over KEEL datasets cmc, monks1, monks3 and Yeast2vs8.

Table 4.20: The AUC value with training time, sensitivity and specificity of the models on ADNI datasets.

Dataset (ir) (samples,features)	TSVM [22] (AUC,time(sceonds)) (Sens.,Spec.)	IFTWSVM [49] (AUC,time(sceonds)) (Sens.,Spec.)	IFTWSVM-ID [51] (AUC,time(sceonds)) (Sens.,Spec.)	pin-IFTWSVM [111] (AUC,time(sceonds)) (Sens.,Spec.)	IIFTWSVM [50] (AUC,time(sceonds)) (Sens.,Spec.)	ACFSVM [46] (AUC,time(sceonds)) (Sens.,Spec.)	ACFTSVM [] (AUC,time(sceonds)) (Sens.,Spec.)	proposed CGFTSVM-ID (AUC,time(sceonds)) (Sens.,Spec.)
CN_vs_AD (1.22) (415, 91)	(0.8736, 0.0206) (0.8431, 0.9041)	(0.8834, 0.0444) (0.8627, 0.9041)	(0.8834, 0.0615) (0.8627, 0.9041)	(0.901, 0.0412) (0.8431, 0.9589)	(0.8033, 0.067) (0.9216, 0.6849)	(0.802, 0.053) (0.6863, 0.9178)	(0.9128 , 0.0797) (0.9216, 0.90411)	(0.8962, 0.0244) (0.902, 0.8904)
CN_vs_MCI (1.75) (626, 91)	(0.6219, 0.0158) (0.8504, 0.3934)	(0.673, 0.029) (0.7559, 0.5902)	(0.7006, 0.0359) (0.811, 0.5902)	(0.6621, 0.0371) (0.4882, 0.8361)	(0.6461, 0.0988) (0.4724, 0.8197)	(0.6939, 0.0423) (0.5354, 0.8525)	(0.7071 , 0.1037) (0.6929, 0.721311)	(0.6776, 0.029) (0.7323, 0.623)
MCI_vs_AD (2.13) (585, 91)	(0.6587, 0.0293) (0.5538, 0.7636)	(0.6615, 0.0209) (0.5231, 0.8)	(0.615, 0.0264) (0.3846, 0.8455)	(0.6462, 0.0271) (0.4923, 0.8)	(0.6455, 0.0441) (0.4, 0.8909)	(0.6182, 0.0416) (0.4, 0.8364)	(0.6178, 0.0849) (0.5538, 0.681818)	(0.6801 , 0.0229) (0.5692, 0.7909)
Average AUC	0.7181	0.7393	0.733	0.7364	0.6983	0.7047	0.7459	0.7513
Average Rank	5.67	3.83	4.83	4	6.33	5.67	3	2.67

Table 4.21: G-mean and F-measure of the proposed CGFTSVM-ID and baseline models on ADNI datasets.

Dataset	TSVM [22] (G-mean,F-measure)	IFTWSVM [49] (G-mean,F-measure)	IFTWSVM-ID [51] (G-mean,F-measure)	pin-IFTWSVM [111] (G-mean,F-measure)	IIFTWSVM [50] (G-mean,F-measure)	ACFSVM [46] (G-mean,F-measure)	ACFTSVM [154] (G-mean,F-measure)	proposed CGFTSVM-ID (G-mean,F-measure)
CN_vs_AD	(0.8515, 0.8731)	(0.8627, 0.8832)	(0.8627, 0.8832)	(0.8866, 0.8992)	(0.7769, 0.7945)	(0.7609, 0.7936)	(0.8952, 0.9128)	(0.8762, 0.8962)
CN_vs_MCI	(0.7941, 0.5784)	(0.7742, 0.6679)	(0.8078, 0.6918)	(0.6231, 0.6389)	(0.6061, 0.6223)	(0.6667, 0.6756)	(0.7586, 0.707)	(0.7654, 0.6754)
MCI_vs_AD	(0.5669, 0.6503)	(0.562, 0.6469)	(0.4673, 0.5702)	(0.5378, 0.6276)	(0.5049, 0.597)	(0.4771, 0.5784)	(0.5294, 0.6145)	(0.592, 0.671)

Table 4.22: Optimal parameters corresponding to AUC value of the proposed CGFTSVM-ID and baseline models on ADNI datasets.

Dataset	TSVM [22] (λ_1, σ)	IFTWSVM [49] ($\lambda_1, \lambda_2, \sigma$)	IFTWSVM-ID [51] ($\lambda_1, \lambda_2, \sigma$)	pin-IFTWSVM [111] ($\lambda_1, \lambda_2, \sigma, \tau, \epsilon$)	IIFTWSVM [50] ($\lambda_1, \lambda_2, \sigma$)	ACFSVM [46] (λ_1, σ)	ACFTSVM [154] (λ_1, σ)	proposed CGFTSVM-ID ($(\lambda_1, \lambda_2, \sigma, a)$)
CN_vs_AD	(1, 32)	(0.01, 0.00001, 32)	(0.01, 0.00001, 32)	(0.1, 10, 8, 0, 0.1)	(10, 100, 8)	(10, 0.125)	(0.0001, 8)	(0.1, 0.1, 16, 0.5)
CN_vs_MCI	(0.1, 32)	(0.1, 0.00001, 8)	(1, 1, 8)	(10, 1, 8, 0.5, 0.2)	(10, 1, 8)	(10, 0.25)	(0.0001, 8)	(0.1, 1, 16, 0.625)
MCI_vs_AD	(10, 32)	(10, 0.00001, 8)	(100, 10, 8)	(1, 0.1, 8, 0.05, 0.01)	(1, 1, 8)	(10, 0.0625)	(1, 32)	(10, 1, 8, 0.625)

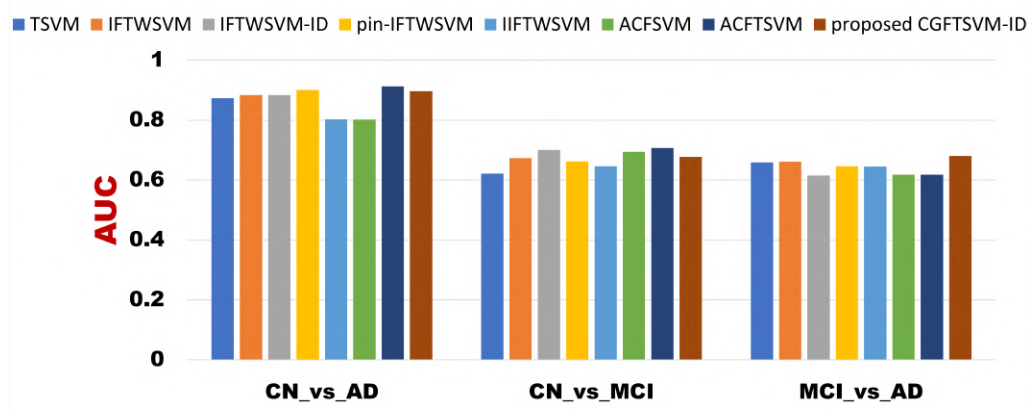


Figure 4.8: Representation of AUC for different models on ADNI datasets.

respectively. The third and most important classification of MCI versus AD case, the proposed CGFTSVM-ID is the best performing with AUC value 0.6801 followed by baseline models in the order IFTWSVM, TSVM, pin-IFTWSVM, IIFTWSVM, ACFSVM, ACFTSVM and IFTWSVM-ID with AUC values 0.6615, 0.6587, 0.6462, 0.6455, 0.6182, 0.6178 and 0.615, respectively. The average AUC of the proposed CGFTSVM-ID is 0.7513 and average rank for the proposed CGFTSVM-ID is 2.67, which are the best among the models. Table 4.21 and 4.22 consist of metric G-mean, F-measure and optimal parameters corresponding to maximum AUC over ADNI dataset. For ease of visualisation, the AUC values of the models are shown in Figure 4.8.

Thus, to infer CGFTSVM-ID incorporates the CPGB-based fuzzy function, blending geometric distance, class probability, and imbalance ratio to produce adaptive weights. This allows effective handling of noisy and imbalanced data.

4.4 Proposed bell-shaped fuzzy least square twin support vector machine for imbalanced data

The novel CPGB discussed in the previous section relies on the fixed structure of the generalized bell function, which may not fully capture the nuanced positioning of data

points in the feature space. Further, the optimization problem of CGFTSVM-ID solves QPPs for training of the model which incurs high complexity. To address these limitations, in this section, we propose another novel membership function class probability and bell-shaped (CPBS), and seamlessly incorporate it with LSTSVM leading to a novel classification model bell-shaped fuzzy least square twin support vector machine for imbalanced data (BSFLSTSVM-ID).

4.4.1 Formulation of proposed BSFLSTSVM-ID

The novel CPBS function overcomes the limitations of existing weighting techniques along with handling noise and outliers effectively. Utilizing the novel CPBS function, BSFLSTSVM-ID is proposed. We discuss the linear and non-linear cases of the proposed BSFLSTSVM-ID in detail.

4.4.1.1 Determination of weighting scheme

Here, we discuss the proposed weighting technique assigned to the data points.

Bell-shaped function: Taking motivation from [155], we design a flexible bell-shaped function to assign weights to the samples of the majority class based on their positions in higher dimensional space. The function is defined as:

$$\mu_g(x_i) = \begin{cases} \frac{1}{1 + \left(\frac{\lambda \|\psi(x_i) - C_+\|}{a} \right)^{2a}}, & \text{if } y_i = 1, \\ \frac{1}{1 + \left(\frac{\lambda \|\psi(x_i) - C_-\|}{a} \right)^{2a}}, & \text{if } y_i = -1. \end{cases} \quad (4.51)$$

Here, C_+ and C_- denote the centers of positive and negative classes, respectively, which are calculated using the mean of the data points in the higher dimensional space. The parameter a in the membership function represents positive values determining the width of the function. λ corresponds to the positive scale parameter which scales the distance between class

center and data points. The scale parameter provides flexibility to the curve as shown in Figure 4.9 (for equation 4.51) by varying λ and fixing a . Additionally, ψ denotes a function from input space to a space of higher-dimension. The membership value assigned using the function (4.51) is utilized to identify the samples on the border and detect outliers within the majority class.

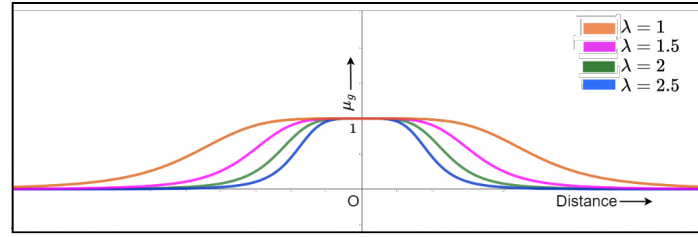


Figure 4.9: Illustration of bell-shaped function for different values of scale parameter λ while fixing width parameters a . Here, O corresponds to the class center.

Class probability: For each data point, the class probability is the ratio between the number of the same class samples to the total samples, in some fixed neighbourhood. The larger value of class probability (P) corresponds to the greater belongingness of the data point to its own class. Hence, less value is assigned to noisy samples effectively. The class probability is calculated as follows:

$$P(x_i) = \frac{|\{x_k : \|\psi(x_k) - \psi(x_i)\| \leq \Delta, y_k = y_i\}|}{|x_k : \|\psi(x_k) - \psi(x_i)\| \leq \Delta|}, \quad (4.52)$$

where Δ is the user-given positive parameter.

Proposed CPBS membership function: The CPBS membership function is obtained from the amalgamation of the aforementioned bell-shaped and class probability functions (equations (4.51) and (4.52)), along with the dataset's imbalance ratio (ir)—namely, the ratio between count of majority to that of minority samples. Finally, the proposed CPBS

function is defined as follows:

$$\mu(x_i) = \begin{cases} 1, & \text{if } x_i \in \text{minority class,} \\ \mu_g(x_i) \times P(x_i) \times \frac{1}{ir}, & \text{if } x_i \in \text{majority class.} \end{cases} \quad (4.53)$$

Theorem 4. *The membership value determined by the CPBS, as expressed in equation (4.53), lies in the interval $[0, 1]$ for parameters $\lambda, a > 0$, and the class centers o_1 and o_2 , corresponding to the positive and negative classes, respectively.*

Proof. For the data points of the minority class, the theorem holds.

For majority class data points, we have

$$\begin{aligned} & \|\psi(x_i) - C_+\| \geq 0 \\ \Rightarrow & \left(\frac{\lambda \|\psi(x_i) - C_+\|}{a} \right)^{2a} \geq 0 \quad \forall o_1, \lambda > 0, a > 0 \\ \Rightarrow & 1 + \left(\frac{\lambda \|\psi(x_i) - C_+\|}{a} \right)^{2a} \geq 1 \\ \Rightarrow & \frac{1}{1 + \left(\frac{\lambda \|\psi(x_i) - C_+\|}{a} \right)^{2a}} \leq 1. \end{aligned}$$

Further, we have $\frac{1}{1 + \left(\frac{\lambda \|\psi(x_i) - C_+\|}{a} \right)^{2a}} \geq 0$.

Thus, $0 < \mu_g(x_i) \leq 1$ and $0 \leq P(x_i) \leq 1$. Furthermore, the term $\frac{1}{ir} \leq 1$ since ir represents the ratio of the count of majority samples to that of minority samples. Therefore, combining all, we get $0 \leq \mu(x_i) \leq 1, \forall x_i \in \text{majority class}$. Hence, the theorem holds. $\square$

The proposed CPBS membership function overcomes the limitations of fuzzy functions mentioned in subsection 2.3.6 in the following way:

- The CPBS membership value does not decrease linearly on moving away from the center as in the case with IFN assignment, rather it depends on the parameter ‘ a ’ of the bell-shaped function (4.51). Hence, by selecting an optimal parameter ‘ a ’, the

classification performance is enhanced, particularly for datasets where a significant portion of samples lies at the boundary of the class or the datasets having dense data distribution near the center.

Further, the CPBS incorporates a bell-shaped function with adjustable scale parameters, allowing for finer control over the weight assignment based on data distributions in contrast to CPGB. This increased adaptability ensures better identification of borderline samples and outliers within the majority class while preserving the emphasis on minority-class samples.

4.4.1.2 Linear case of proposed BSFLSTSVM-ID

For classifying linearly separable data points, the proposed BSFLSTSVM-ID employs the linear case approach. Employing the CPBS membership value obtained by equation (4.53) with the LSTSVM model leads to the introduction of the proposed BSFLSTSVM-ID which is noise/outliers tolerant and impactful against class imbalance training data. The equations of the non-parallel hyperplanes are assumed as:

$$f_1(x) : w_1^t x + b_1 = 0 \quad \text{and} \quad f_2(x) : w_2^t x + b_2 = 0.$$

The optimization problems can be formulated as:

$$\begin{aligned} \min_{w_1, b_1, \xi_1} \quad & \frac{C_1}{2} (\|w_1\|^2 + b_1^2) + \frac{1}{2} \|M_1 w_1 + e_1 b_1\|^2 + C_2 \|\mu_2 \xi_1\|^2 \\ \text{subject to} \quad & -(M_2 w_1 + e_2 b_1) + \xi_1 = e_2 \end{aligned} \quad (4.54)$$

and

$$\begin{aligned} \min_{w_2, b_2, \xi_2} \quad & \frac{C_3}{2} (\|w_2\|^2 + b_2^2) + \frac{1}{2} \|M_2 w_2 + e_2 b_2\|^2 + C_4 \|\mu_1 \xi_2\|^2 \\ \text{subject to} \quad & (M_1 w_2 + e_1 b_2) + \xi_2 = e_1. \end{aligned} \quad (4.55)$$

Here, C_1 , C_2 , C_3 and C_4 are the positive hyperparameters; μ_1 and μ_2 are the diagonal matrices of appropriate dimensions having weights for the positive and negative classes, respectively, calculated using equation (4.53). The optimization problems (4.54) and (4.55) draw inspiration from the LSTSVM principle proposed by [33]. This principle involves minimizing both, the distance between data points and their respective hyperplanes, along with the weighted penalties on data points from the opposite class which deviate beyond a margin of one. The first term of the problem (4.54) signifies the regularization term which implements the SRM principle; the second term minimizes the square of the sum of distance between data points and its corresponding hyperplane; the third term denotes square of the 2-norm of weighted slack variables. A similar explanation follows for problem (4.55).

To acquire the unknowns of the hyperplane $f_1(x)$, we substitute the slack variable from the constraint into the objective function and then equate the derivatives with respect to w_1 and b_1 to zero. Thus, we get:

$$M_1^t(M_1w_1 + e_1b_1) + C_1w_1 + C_2(\mu_2M_2)^t(\mu_2(M_2w_1 + e_2b_1 + e_2)) = 0, \quad (4.56)$$

$$e_1^t(M_1w_1 + e_1b_1) + C_1b_1 + C_2(\mu_2e_2)^t(\mu_2(M_2w_1 + e_2b_1 + e_2)) = 0. \quad (4.57)$$

Combining the equations (4.56) and (4.57), we obtain:

$$-(A^tA + C_1I + C_2B^tB)z_1 = C_2B^t\mu_2e_2 \quad (4.58)$$

$$\implies z_1 = -(A^tA + C_1I + C_2B^tB)^{-1}C_2B^t\mu_2e_2, \quad (4.59)$$

where $A = \begin{bmatrix} M_1 & e_1 \end{bmatrix}$, $B = \begin{bmatrix} \mu_2M_2 & \mu_2e_2 \end{bmatrix}$ and $z_1 = \begin{bmatrix} w_1 \\ b_1 \end{bmatrix}$. Assuming $z_2 = \begin{bmatrix} w_2 \\ b_2 \end{bmatrix}$ and solving in a similar manner as for $f_1(x)$, we obtain the following relation for the second

hyperplane $f_2(x)$:

$$(E^t E + C_3 I + C_4 F^t F) z_2 = C_4 F^t \mu_1 e_1 \quad (4.60)$$

$$\implies z_2 = (E^t E + C_3 I + C_4 F^t F)^{-1} C_4 F^t \mu_1 e_1, \quad (4.61)$$

with $E = \begin{bmatrix} M_2 & e_2 \end{bmatrix}$, $F = \begin{bmatrix} \mu_1 M_1 & \mu_1 e_1 \end{bmatrix}$. After training the model, the decision function for a test data point (x) is expressed as:

$$f(x) = \underset{j=1,2}{\operatorname{argmin}} \left(\frac{|u_j^t x + b_j|}{\|u_j\|} \right). \quad (4.62)$$

4.4.1.3 Non-Linear case of proposed BSFLSTSVM-ID

When the training data is not linearly separable, it is projected into a higher-dimensional space using a mapping function ψ . The non-linear surfaces generated by the kernel are described by the following equations: $K(x, M^t)v_1 + e_1 b_1 = 0$, $K(x, M^t)v_2 + e_2 b_2 = 0$, where $b_1, b_2 \in \mathbb{R}$ are the bias terms; v_1, v_2 are the weight vectors and $M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$. The optimization problems of non-linear BSFLSTSVM-ID are given by:

$$\begin{aligned} \min_{v_1, b_1, \xi_1} & \frac{C_1}{2} (\|v_1\|^2 + b_1^2) + \frac{1}{2} \|K(M_1, M^t)v_1 + e_1 b_1\|^2 + C_2 \|\mu_2 \xi_1\|^2 \\ \text{subject to} & - (K(M_2, M^t)v_1 + e_2 b_1) + \xi_1 = e_2 \end{aligned} \quad (4.63)$$

and

$$\begin{aligned} \min_{v_2, b_2, \xi_2} & \frac{C_3}{2} (\|v_2\|^2 + b_2^2) + \frac{1}{2} \|K(M_2, M^t)v_2 + e_2 b_2\|^2 + C_4 \|\mu_1 \xi_2\|^2 \\ \text{subject to} & (K(M_1, M^t)v_2 + e_1 b_2) + \xi_2 = e_1, \end{aligned} \quad (4.64)$$

where the notations have the same meaning as discussed in the linear case. We obtain the unknown parameters v_1, b_1 and v_2, b_2 by solving the following linear system of equations:

$$-(A^t A + C_1 I + C_2 B^t B)z_1 = C_2 B^t \mu_2 e_2 \quad (4.65)$$

$$\text{and } (E^t E + C_3 I + C_4 F^t F)z_2 = C_4 F^t \mu_1 e_1, \quad (4.66)$$

respectively which are equivalent to

$$z_1 = -(A^t A + C_1 I + C_2 B^t B)^{-1} C_2 B^t \mu_2 e_2 \quad (4.67)$$

$$\text{and } z_2 = (E^t E + C_3 I + C_4 F^t F)^{-1} C_4 F^t \mu_1 e_1, \quad (4.68)$$

where $A = \begin{bmatrix} K(M_1, M^t) & e_1 \end{bmatrix}$, $B = \begin{bmatrix} \mu_2 K(M_2, M^t) & \mu_2 e_2 \end{bmatrix}$, $F = \begin{bmatrix} \mu_1 K(M_1, M^t) & \mu_1 e_1 \end{bmatrix}$, $E = \begin{bmatrix} K(M_2, M^t) & e_2 \end{bmatrix}$, $z_1 = \begin{bmatrix} v_1 \\ b_1 \end{bmatrix}$, $z_2 = \begin{bmatrix} v_2 \\ b_2 \end{bmatrix}$. The decision function for a test data point (x) is expressed as

$$f(x) = \underset{j=1,2}{\operatorname{argmin}} \left\{ \frac{|K(x, M^t)v_j + b_j|}{\sqrt{v_j^t K(M, M^t)v_j}} \right\}. \quad (4.69)$$

4.4.1.4 Conjugate gradient method

As evident from the equations (4.59), (4.61), (4.67) and (4.68) that the proposed BSFLSTSVM-ID necessitates matrix inversion, which incurs a high complexity of $O(l^3)$, where l is the order of the matrix. To mitigate the high computational complexity of inversion, we employ the conjugate gradient (CG) method [150] and find the solution of the system of linear equations of the proposed optimization problems (4.58), (4.60), (4.65), and (4.66), which are in the form $Az = b$. CG is one of the most popular and widely used iterative methods for solving large systems of linear equations. It is particularly efficient for solving symmetric and positive definite linear systems, making it a popular choice in

various scientific and engineering applications. The main steps of CG method are shown in Algorithm 4.2. The convergence of the CG method is guaranteed through the following theorem.

Theorem 5. *For any starting point $z_0 \in \mathbb{R}^n$, the sequence $\{z_k\}$ generated by the CG algorithm converges to the unique solution of the system of linear equations $Az = b$. [148]*

Algorithm 4.2 CG algorithm for proposed BSFLSTSVM-ID

Initialization: Starting point: z_0 , residual $r_0 = b - Ax_0$, $g_0 = r_0$, max iteration N and tolerance value ‘tol’.

Output: Optimal solution, z^* of system of linear equation $Az = b$.

for $j = 1, 2, \dots, N$

$$\alpha = \frac{r_0^t r_0}{g_0^t A g_0}$$

$$z_{j+1} = z_j + \alpha g_0$$

$$r_{j+1} = r_j - \alpha A g_0$$

if $\|r_{j+1}\| < \text{tol}$

break

else

$$\beta = \frac{\|r_{j+1}\|}{\|r_j\|}$$

$$g_{j+1} = r_{j+1} + \beta * g_j$$

end if

4.4.2 Numerical experiments

In this section, we thoroughly discuss the numerical experiments conducted to compare the performance of the proposed BSFLSTSVM-ID with the baseline models, i.e., TWSVM [22], LSTSVM [33], IFTWSVM [49], IFTWSVM-ID [156], pin-IFTWSVM [111], RFLSTSVM-CIL [151], IIFTWSVM [50], LS-FLSTSVM-CIL [115] and ACFTSVM [154]. The experiments are carried out on 12 synthetic KEEL datasets, 17 UCI [126] and 42 KEEL [127] datasets. To show its scalability to large-scale datasets, we have conducted experiments on large-scale NDC [157] datasets.

4.4.2.1 Experimental setup

All experiments, excluding NDC data, are conducted on a desktop PC equipped with an 11th Gen Intel(R), 16 GB RAM Core(TM) i7-11700 @ 2.50GHz processor having MATLAB 2022b. The training and testing datasets are divided in the 70 : 30 ratio, respectively. The experiments for non-linear cases are conducted with RBF kernel ($\exp(\frac{\|x_i - x_j\|^2}{2\sigma^2})$). The parameters involved in the proposed BSFLSTSVM-ID are $C_1, C_2, C_3, C_4, \sigma, a$ and λ with $C_1 = C_3$ and $C_2 = C_4$ to enhance the tuning time. The optimal parameters are chosen via grid search with 10-fold cross-validation from the following ranges: $C_1 \in \{10^{-5}, 10^{-4}, \dots, 10^4, 10^5\}$, $C_2 \in \{10^{-5}, 10^{-4}, \dots, 10^4, 10^5\}$, $\sigma \in \{2^{-5}, 2^{-4}, \dots, 2^4, 2^5\}$, $a \in [1/2, 5/8, 3/4, 7/8, 1] \times (\text{radius of positive/negative class})$ and $\lambda \in [1, 1.5, 2, 2.5]$. Initially, the training dataset is partitioned into 10 segments, with each segment serving as the validation data once while the remaining 9 parts constitute the training dataset. This process results in 10 accuracies. The optimal parameters are determined based on the maximum of the mean accuracies obtained across different parameter sets. Subsequently, utilizing the optimal parameters, we assess the test accuracy of the dataset. The baseline models' respective parameters c_0, C_1 ($C_3 = C_1$), C_2 ($C_4 = C_2$) and σ take the aforementioned range of values. Ranges for parameters τ and ϵ of pin-IFTWSVM are same as in [111]; c_0 for LS-FLSTSVM-CIL is taken from $\{0.5, 1, 1.5, 2, 2.5\}$. Assume tp, tn and fp, fn correspond to the correctly classified positive, negative and incorrectly classified positive and negative samples, respectively. Metrics considered along with training time to check the performance of the models are as follows:

$$\text{AUC} = \frac{1 + \frac{tp}{tp+fn} - \frac{fp}{fp+tn}}{2},$$

$$\text{F-measure } (F) = \frac{2 \times S \times P}{S + P}, \quad \text{G-mean } (G) = \sqrt{S \times P},$$

Table 4.23: AUC with training time, F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models over synthetic datasets with RBF kernel.

Dataset (<i>ir</i>) (samples,features)	TWSVM [22] (AUC,time(seconds)) (G-mean,F-measure)	LSTSVM [33] (AUC,time(seconds)) (G-mean,F-measure)	IFTWSVM [49] (AUC,time(seconds)) (G-mean,F-measure)	IFTWSVM-ID [51] (AUC,time(seconds)) (G-mean,F-measure)	pin-IFTWSVM [111] (AUC,time(seconds)) (G-mean,F-measure)	RFLSTSVM [151] (AUC,time(seconds)) (G-mean,F-measure)	IIFTWSVM [50] (AUC,time(seconds)) (G-mean,F-measure)	LS-FLSTSVM-CIL [115] (AUC,time(seconds)) (G-mean,F-measure)	ACFTSVM [154] (AUC,time(seconds)) (G-mean,F-measure)	proposed BSFLSTSVM-ID (AUC,time(seconds)) (G-mean,F-measure)
03subcl5-600-5-0-BI (5) (600,2)	(0.8428, 0.0342) (0.7436, 0.8391)	(0.9042, 0.3294) (0.7865, 0.904)	(0.8315, 0.1273) (0.7761, 0.8184)	(0.8592, 0.1811) (0.6849, 0.782)	(0.7674, 0.0597) (0.6479, 0.7501)	(0.8708, 0.089) (0.7416, 0.8708)	(0.735, 0.2477) (0.6061, 0.7047)	(0.806, 0.0978) (0.6835, 0.8003)	(0.8708, 0.2887) (0.7416, 0.8708)	(0.8525, 0.0916) (0.75, 0.8502)
03subcl5-600-5-30-BI (5) (600,2)	(0.7809, 0.042) (0.5645, 0.7683)	(0.6866, 0.0175) (0.5067, 0.6608)	(0.7013, 0.0439) (0.5, 0.7011)	(0.7109, 0.0336) (0.5143, 0.7144)	(0.7741, 0.0397) (0.5794, 0.773)	(0.8073, 0.0304) (0.5873, 0.7899)	(0.5665, 0.0641) (0.2807, 0.4407)	(0.7266, 0.0974) (0.5225, 0.7257)	(0.7943, 0.1447) (0.6038, 0.7929)	(0.8117, 0.0407) (0.6, 0.8003)
03subcl5-600-5-70-BI (5) (600,2)	(0.7107, 0.0274) (0.496, 0.7029)	(0.7967, 0.019) (0.5736, 0.7768)	(0.7591, 0.0403) (0.5517, 0.7545)	(0.7511, 0.0358) (0.518, 0.7167)	(0.7259, 0.0419) (0.5361, 0.7247)	(0.795, 0.0392) (0.5833, 0.785)	(0.6577, 0.1052) (0.4403, 0.6027)	(0.7048, 0.0715) (0.5049, 0.7045)	(0.7107, 0.1496) (0.496, 0.7029)	(0.8152, 0.0377) (0.605, 0.8045)
03subcl5-800-7-0-BI (7) (800,2)	(0.8426, 0.0411) (0.6744, 0.8406)	(0.9004, 0.0198) (0.7805, 0.8997)	(0.8427, 0.0435) (0.7397, 0.8351)	(0.8549, 0.1209) (0.6526, 0.8523)	(0.8512, 0.0712) (0.6742, 0.8503)	(0.8684, 0.0447) (0.6737, 0.8684)	(0.7592, 0.0902) (0.6176, 0.7346)	(0.8021, 0.1082) (0.6265, 0.7959)	(0.8291, 0.2773) (0.6588, 0.826)	(0.8414, 0.0691) (0.6452, 0.8408)
04clover5z-600-5-50-BI (5) (600,2)	(0.76, 0.0274) (0.5586, 0.758)	(0.8523, 0.0105) (0.6869, 0.8512)	(0.7871, 0.0274) (0.5669, 0.7706)	(0.773, 0.0205) (0.5496, 0.7531)	(0.8021, 0.0273) (0.5932, 0.7932)	(0.8145, 0.0242) (0.6286, 0.8127)	(0.5953, 0.0486) (0.3953, 0.5144)	(0.7748, 0.0658) (0.562, 0.7655)	(0.8208, 0.189) (0.6742, 0.8202)	(0.826, 0.0442) (0.6598, 0.8258)
04clover5z-800-7-0-BI (7) (800,2)	(0.8931, 0.0592) (0.8333, 0.8893)	(0.9459, 0.0213) (0.8434, 0.9459)	(0.8575, 0.0642) (0.806, 0.8479)	(0.8575, 0.0467) (0.7302, 0.7826)	(0.844, 0.0476) (0.7879, 0.8321)	(0.9533, 0.0429) (0.875, 0.9532)	(0.8746, 0.1055) (0.7209, 0.8738)	(0.8968, 0.1125) (0.8052, 0.8948)	(0.9398, 0.3465) (0.8608, 0.9395)	(0.9581, 0.1179) (0.8132, 0.9572)
04clover5z-800-7-30-BI (7) (800,2)	(0.7554, 0.0332) (0.5789, 0.7381)	(0.8118, 0.0202) (0.5714, 0.8118)	(0.781, 0.0483) (0.5, 0.7789)	(0.8018, 0.0493) (0.5037, 0.7932)	(0.7269, 0.0486) (0.45, 0.7269)	(0.8131, 0.0433) (0.6353, 0.8089)	(0.5764, 0.1302) (0.3008, 0.3908)	(0.7197, 0.1054) (0.4742, 0.713)	(0.802, 0.2729) (0.5505, 0.8019)	(0.8216, 0.0488) (0.5614, 0.8205)
04clover5z-800-7-60-BI (7) (800,2)	(0.7404, 0.0416) (0.4478, 0.737)	(0.7821, 0.022) (0.4755, 0.7701)	(0.7712, 0.0385) (0.5043, 0.7711)	(0.7724, 0.0461) (0.5, 0.774)	(0.7638, 0.0441) (0.4915, 0.7636)	(0.8019, 0.0509) (0.5246, 0.7994)	(0.5394, 0.1155) (0.2835, 0.2807)	(0.7036, 0.1143) (0.4364, 0.7015)	(0.7946, 0.2508) (0.5357, 0.7944)	(0.8141, 0.0503) (0.5231, 0.8074)
paw02a-600-5-60-BI (5) (600,2)	(0.8619, 0.0316) (0.6931, 0.8599)	(0.8495, 0.011) (0.6491, 0.8404)	(0.805, 0.0279) (0.6667, 0.8021)	(0.854, 0.0247) (0.6739, 0.8268)	(0.7776, 0.0225) (0.5849, 0.7767)	(0.8425, 0.0223) (0.6379, 0.8322)	(0.7198, 0.0658) (0.5319, 0.7171)	(0.8345, 0.0718) (0.6179, 0.8179)	(0.8234, 0.1181) (0.6667, 0.8234)	(0.8427, 0.0114) (0.6804, 0.8423)
paw02a-800-7-0-BI (7) (800,2)	(0.968, 0.0272) (0.9459, 0.9678)	(0.9582, 0.0196) (0.8974, 0.9581)	(0.968, 0.0454) (0.9459, 0.9678)	(0.9582, 0.0421) (0.9167, 0.9397)	(0.9778, 0.0494) (0.8916, 0.9776)	(0.93, 0.0417) (0.9143, 0.9277)	(0.9286, 0.2322) (0.7778, 0.9285)	(0.9778, 0.1045) (0.8916, 0.9776)	(0.9656, 0.3106) (0.9333, 0.9654)	(0.9766, 0.1157) (0.9351, 0.9766)
paw02a-800-7-50-BI (7) (800,2)	(0.8498, 0.0668) (0.5738, 0.8444)	(0.8215, 0.0217) (0.5354, 0.8157)	(0.8203, 0.0527) (0.5455, 0.8172)	(0.7783, 0.038) (0.5034, 0.8002)	(0.8326, 0.0451) (0.569, 0.8305)	(0.8091, 0.041) (0.4966, 0.7924)	(0.6492, 0.1511) (0.3509, 0.6288)	(0.8289, 0.1095) (0.5255, 0.8162)	(0.829, 0.2351) (0.5766, 0.8282)	(0.8301, 0.017) (0.5385, 0.822)
paw02a-800-7-60-BI (7) (800,2)	(0.8486, 0.0634) (0.5862, 0.8457)	(0.8239, 0.0217) (0.518, 0.8103)	(0.7909, 0.042) (0.5421, 0.7909)	(0.7857, 0.0395) (0.5565, 0.8179)	(0.7909, 0.0616) (0.5421, 0.7909)	(0.8141, 0.039) (0.5231, 0.8074)	(0.7368, 0.1524) (0.4655, 0.7368)	(0.7835, 0.1066) (0.5041, 0.7816)	(0.8486, 0.2358) (0.5862, 0.8457)	(0.8154, 0.0334) (0.5366, 0.8118)
Average AUC	0.8212	0.8444	0.8096	0.8131	0.8029	0.8433	0.6949	0.7966	0.8357	0.8505
Average Rank	5.13	3.63	6.38	6	6.08	3.88	9.75	6.96	4.46	2.75

Table 4.24: AUC with training time (in seconds), F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models over UCI datasets with RBF kernel.

Dataset (<i>ir</i>) (samples,features)	TWSVM [22] (AUC,train time) (G, F)	LSTSVM [33] (AUC,train time) (G, F)	IFTWSVM [49] (AUC,train time) (G, F)	IFTWSVM-ID [51] (AUC,train time) (G, F)	pin-IFTWSVM [111] (AUC,train time) (G, F)	RFLSTSVM-CIL [151] (AUC,train time) (G, F)	IIFTWSVM [50] (AUC,train time) (G, F)	LS-FLSTSVM-CIL [115] (AUC,train time) (G, F)	ACFTSVM [154] (AUC,train time) (G, F)	proposed BSFLSTSVM-ID (AUC,train time) (G, F)
blood (3.2) (748,4)	(0.7062, 0.0362) (0.5693, 0.703)	(0.6562, 0.0176) (0.514, 0.6488)	(0.6601, 0.0309) (0.5106, 0.6564)	(0.655, 0.0455) (0.5036, 0.6499)	(0.6611, 0.1209) (0.5079, 0.6468)	(0.6755, 0.0362) (0.5294, 0.67)	(0.6128, 0.2045) (0.4043, 0.5336)	(0.6879, 0.0989) (0.5455, 0.6861)	(0.651, 0.2743) (0.5032, 0.6509)	(0.7164, 0.02) (0.5846, 0.7103)
breast_cancer_wisc (1.9) (699,9)	(0.9655, 0.0178) (0.9548, 0.9655)	(0.9586, 0.0145) (0.9333, 0.9578)	(0.9812, 0.0396) (0.9686, 0.981)	(0.9525, 0.0322) (0.9412, 0.9523)	(0.9655, 0.0808) (0.9548, 0.9655)	(0.9785, 0.0351) (0.9682, 0.9784)	(0.9607, 0.0666) (0.9434, 0.9606)	(0.9645, 0.0809) (0.9494, 0.9644)	(0.9487, 0.1744) (0.9351, 0.9486)	(0.9812, 0.0111) (0.9686, 0.981)
echocardiogram (2.05) (131,10)	(0.7727, 0.0011) (0.7059, 0.7385)	(0.8182, 0.0007) (0.7778, 0.7977)	(0.7825, 0.002) (0.7, 0.7687)	(0.8182, 0.0021) (0.7778, 0.7977)	(0.7549, 0.0103) (0.6667, 0.7252)	(0.8003, 0.0022) (0.7368, 0.7833)	(0.6364, 0.0127) (0.4286, 0.5222)	(0.7727, 0.018) (0.7059, 0.7385)	(0.737, 0.0163) (0.6316, 0.7117)	(0.8636, 0.0015) (0.8421, 0.8528)
haberman_survival (2.78) (306,3)	(0.6516, 0.0047) (0.5091, 0.6378)	(0.6778, 0.0021) (0.5455, 0.6667)	(0.7162, 0.0049) (0.6, 0.698)	(0.7162, 0.0048) (0.6, 0.698)	(0.69, 0.022) (0.56, 0.6684)	(0.6547, 0.0046) (0.5172, 0.6472)	(0.5, 0.0165) (0, 0)	(0.5182, 0.0263) (0.4466, 0.3966)	(0.5459, 0.0234) (0.4583, 0.475)	(0.7085, 0.0021) (0.5882, 0.6918)
horse_colic (1.71) (368,25)	(0.7492, 0.004) (0.6667, 0.7382)	(0.8101, 0.0033) (0.75, 0.8061)	(0.8232, 0.01) (0.7595, 0.8231)	(0.8169, 0.0099) (0.7606, 0.8123)	(0.8236, 0.0252) (0.7671, 0.8209)	(0.8032, 0.007) (0.7397, 0.7999)	(0.6479, 0.0742) (0.5, 0.6008)	(0.7821, 0.0332) (0.7059, 0.7816)	(0.8162, 0.0753) (0.747, 0.8159)	(0.8369, 0.0035) (0.7792, 0.8365)
ionosphere (1.79) (351,33)	(0.9783, 0.0039) (0.9859, 0.9782)	(0.906, 0.0062) (0.9362, 0.9057)	(0.936, 0.0105) (0.9412, 0.9354)	(0.929, 0.0094) (0.9333, 0.928)	(0.9571, 0.0269) (0.964, 0.957)	(0.9283, 0.0081) (0.942, 0.9282)	(0.8631, 0.0364) (0.8986, 0.863)	(0.5441, 0.0267) (0.8208, 0.297)	(0.9501, 0.0621) (0.9565, 0.9499)	(0.9571, 0.0091) (0.964, 0.957)
mammographic (1.16) (961,5)	(0.809, 0.0102) (0.7876, 0.8027)	(0.8333, 0.0074) (0.8209, 0.8304)	(0.809, 0.0249) (0.7955, 0.8063)	(0.809, 0.0275) (0.7925, 0.8051)	(0.816, 0.1194) (0.8179, 0.8159)	(0.8125, 0.0264) (0.7923, 0.8067)	(0.7917, 0.1342) (0.7959, 0.7914)	(0.8021, 0.0576) (0.8014, 0.8021)	(0.8403, 0.379) (0.838, 0.8402)	(0.8194, 0.0581) (0.806, 0.8165)
molec_biol_promoter (1) (106,57)	(0.7267, 0.001) (0.6897, 0.7255)	(0.7267, 0.0007) (0.6897, 0.7255)	(0.7267, 0.0016) (0.6897, 0.7255)	(0.7004, 0.0025) (0.6667, 0.697)	(0.7267, 0.0143) (0.6897, 0.7255)	(0.7267, 0.0017) (0.6897, 0.7255)	(0.753, 0.0085) (0.7143, 0.7529)	(0.7267, 0.0104) (0.6897, 0.7255)	(0.7267, 0.0117) (0.6897, 0.7255)	(0.753, 0.0009) (0.7143, 0.7529)
monks_1 (1) (556,6)	(0.9765, 0.0099) (0.9767, 0.9764)	(0.8958, 0.0039) (0.9071, 0.8939)	(0.965, 0.0088) (0.9647, 0.9648)	(0.965, 0.0092) (0.9647, 0.9648)	(0.8168, 0.0337) (0.8098, 0.8147)	(0.953, 0.0104) (0.9529, 0.9528)	(0.7871, 0.0233) (0.7778, 0.7845)	(0.9098, 0.0354) (0.9143, 0.9097)	(1, 0.0897) (1, 1)	(0.9438, 0.0037) (0.9508, 0.9421)
monks_2 (1.92) (601,6)	(0.7985, 0.0189) (0.7344, 0.7975)	(0.7896, 0.0103) (0.7218, 0.7895)	(0.7348, 0.0182) (0.6491, 0.7217)	(0.7191, 0.0186) (0.6239, 0.6986)	(0.6568, 0.0801) (0.5391, 0.6378)	(0.715, 0.0187) (0.6331, 0.715)	(0.6232, 0.2116) (0.4396, 0.5459)	(0.6495, 0.052) (0.5185, 0.6186)	(0.7981, 0.1144) (0.7328, 0.7978)	(0.7771, 0.0606) (0.703, 0.7607)
monks_3 (1.08) (554,6)	(0.9239, 0.0101) (0.9176, 0.9208)	(0.9239, 0.0076) (0.9176, 0.9208)	(0.9063, 0.0204) (0.8994, 0.9027)	(0.9063, 0.021) (0.8994, 0.9027)	(0.9076, 0.0688) (0.8982, 0.9029)	(0.9022, 0.0151) (0.8916, 0.8969)	(0.7197, 0.1602) (0.7668, 0.7148)	(0.9076, 0.0476) (0.8982, 0.9029)	(0.9348, 0.1063) (0.9302, 0.9325)	(0.913, 0.0538) (0.9048, 0.9089)
parkinsons (3.06) (195,22)	(0.8846, 0.0017) (0.9677, 0.8771)	(0.912, 0.0012) (0.967, 0.9096)	(0.8462, 0.004) (0.9574, 0.8321)	(0.8735, 0.0038) (0.9565, 0.8673)	(0.9009, 0.0147) (0.9556, 0.8992)	(0.9504, 0.0031) (0.9778, 0.95)	(0.8453, 0.016) (0.8941, 0.8453)	(0.9231, 0.0141) (0.9783, 0.9199)	(0.8846, 0.0196) (0.9677, 0.8771)	(0.912, 0.0036) (0.967, 0.9096)
pima (1.87) (768,8)	(0.7365, 0.0386) (0.6848, 0.7341)	(0.7122, 0.0154) (0.6335, 0.6931)	(0.7428, 0.0329) (0.7053, 0.7416)	(0.7428, 0.0285) (0.7053, 0.7416)	(0.73, 0.0925) (0.6948, 0.727)	(0.7273, 0.0323) (0.6772, 0.7262)	(0.5146, 0.1099) (0.5831, 0.1709)	(0.7407, 0.0758) (0.707, 0.7368)	(0.7528, 0.185) (0.7033, 0.7501)	(0.7528, 0.0208) (0.7033, 0.7501)
spect (1.41) (265,22)	(0.6223, 0.002) (0.5333, 0.6102)	(0.6749, 0.0017) (0.6, 0.6654)	(0.6323, 0.007) (0.5625, 0.6284)	(0.6579, 0.0051) (0.6087, 0.6579)	(0.6948, 0.0185) (0.6471, 0.6948)	(0.6536, 0.0034) (0.5806, 0.6472)	(0.6137, 0.0562) (0.4348, 0.5347)	(0.6656, 0.0182) (0.549, 0.6253)	(0.7354, 0.0293) (0.7089, 0.722)	(0.6998, 0.0018) (0.6571, 0.6995)
statlog_german_credit (2.33) (1000,24)	(0.6752, 0.0676) (0.5134, 0.6738)	(0.7115, 0.0379) (0.5695, 0.6964)	(0.6779, 0.0588) (0.5176, 0.6706)	(0.6991, 0.0544) (0.5408, 0.6991)	(0.6607, 0.1781) (0.4935, 0.6409)	(0.7125, 0.0608) (0.5579, 0.7124)	(0.5557, 0.2963) (0.3805, 0.5541)	(0.6376, 0.1061) (0.4688, 0.6359)	(0.677, 0.3245) (0.5152, 0.677)	(0.7042, 0.1229) (0.5517, 0.7004)
titanic (2.1) (2201,3)	(0.7119, 0.1959) (0.6027, 0.6866)	(0.7119, 0.251) (0.6027, 0.6866)	(0.7117, 0.267) (0.6042, 0.6937)	(0.7119, 0.3628) (0.6027, 0.6866)	(0.7117, 0.4291) (0.6042, 0.6937)	(0.7298, 0.4686) (0.6318, 0.7267)	(0.7, 0.50148) (0.5833, 0.6708)	(0.7117, 0.4992) (0.6042, 0.6937)	(0.7117, 2.6185) (0.6042, 0.6937)	(0.7119, 0.0623) (0.6027, 0.6866)
vertebral_column_2clases (2.1) (310,6)	(0.7596, 0.0094) (0.6563, 0.7596)	(0.8236, 0.0017) (0.7458, 0.8228)	(0.8058, 0.0064) (0.7241, 0.8038)	(0.8209, 0.0055) (0.7273, 0.8201)	(0.7827, 0.0215) (0.6885, 0.782)	(0.8363, 0.0047) (0.75, 0.836)	(0.7956, 0.0234) (0.7143, 0.7914)	(0.8261, 0.0237) (0.7419, 0.8261)	(0.8514, 0.0402) (0.75, 0.8439)	(0.8415, 0.0024) (0.7667, 0.8412)
Average AUC	0.7911	0.7966	0.7916	0.7937	0.7798	0.7976	0.7012	0.7512	0.7977	0.8172
Average Rank	5.32	5	5.35	5.65	5.76	4.94	9.15	6.56	4.68	2.59

Table 4.25: AUC with training time, F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models over KEEL datasets with RBF kernel.

Dataset (<i>nr</i>) (samples,features)	TWSVM [22] (AUC,time(seconds)) (G-mean,F-measure)	LSTSVM [33] (AUC,time(seconds)) (G-mean,F-measure)	IFTWSVM [49] (AUC,time(seconds)) (G-mean,F-measure)	IFTWSVM-ID [51] (AUC,time(seconds)) (G-mean,F-measure)	pin-IFTWSVM [111] (AUC,time(seconds)) (G-mean,F-measure)	RFLSTSVM [151] (AUC,time(seconds)) (G-mean,F-measure)	IIFTWSVM [50] (AUC,time(seconds)) (G-mean,F-measure)	LS-FLSTSVM-CIL [115] (AUC,time(seconds)) (G-mean,F-measure)	ACFTSVM [154] (AUC,time(seconds)) (G-mean,F-measure)	proposed BSFLSTSVM-ID (AUC,time(seconds)) (G-mean,F-measure)
crossplane130 (1)	(1, 0.0044)	(1, 0.0016)	(1, 0.0071)	(1, 0.008)	(1, 0.0192)	(1, 0.0073)	(0.9773, 0.0108)	(0.9773, 0.0218)	(1, 0.0666)	(1, 0.0031)
(130,2)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(0.9767, 0.977)	(0.9767, 0.977)	(1, 1)	(1, 1)
ecoli-0-1-4-6_vs_5 (13)	(0.9873, 0.0049)	(0.9747, 0.0021)	(0.9873, 0.0156)	(0.943, 0.0155)	(0.9494, 0.0253)	(0.9684, 0.0064)	(0.9747, 0.0174)	(1, 0.0228)	(0.981, 0.0692)	(0.981, 0.0033)
(280,6)	(0.8333, 0.9873)	(0.7143, 0.9744)	(0.8333, 0.9873)	(0.5263, 0.9413)	(0.5556, 0.948)	(0.6667, 0.9678)	(0.7143, 0.9744)	(1, 1)	(0.7692, 0.9808)	(0.7692, 0.9808)
ecoli-0-1-4-7_vs_2-3-5-6 (10.59)	(0.6728, 0.0142)	(0.8641, 0.0053)	(0.9056, 0.0161)	(0.9228, 0.0182)	(0.8998, 0.043)	(0.8169, 0.0127)	(0.8157, 0.0264)	(0.8251, 0.0269)	(0.8699, 0.0636)	(0.8596, 0.0039)
(336,7)	(0.5, 0.5942)	(0.7333, 0.8606)	(0.8, 0.9043)	(0.8889, 0.9205)	(0.7742, 0.8988)	(0.6452, 0.8104)	(0.75, 0.7972)	(0.5455, 0.8245)	(0.7586, 0.8658)	(0.6316, 0.8596)
ecoli-0-1-4-7_vs_5-6 (12.28)	(0.928, 0.0065)	(0.9335, 0.0058)	(0.928, 0.0062)	(0.928, 0.0109)	(0.9225, 0.0289)	(0.9225, 0.0098)	(0.8004, 0.0205)	(0.9005, 0.0283)	(0.928, 0.0516)	(0.9335, 0.0051)
(332,6)	(0.8, 0.9271)	(0.8421, 0.9324)	(0.8, 0.9271)	(0.8, 0.9271)	(0.7619, 0.9219)	(0.7619, 0.9219)	(0.5714, 0.7891)	(0.64, 0.9004)	(0.8, 0.9271)	(0.8421, 0.9324)
ecoli-0-1_vs_5 (11)	(0.9929, 0.0019)	(0.9786, 0.0017)	(0.9786, 0.0085)	(0.95, 0.013)	(0.9786, 0.019)	(0.9786, 0.0036)	(0.9714, 0.0144)	(0.9786, 0.0236)	(0.9714, 0.0615)	(0.9857, 0.0051)
(240,6)	(0.8, 0.9928)	(0.5714, 0.9783)	(0.5714, 0.9783)	(0.3636, 0.9487)	(0.5714, 0.9783)	(0.5714, 0.9783)	(0.5, 0.971)	(0.5714, 0.9783)	(0.5, 0.971)	(0.6667, 0.9856)
ecoli-0-2-3-4_vs_5 (9.1)	(0.8061, 0.0017)	(0.8152, 0.0009)	(0.7409, 0.0036)	(0.7409, 0.0039)	(0.7227, 0.0163)	(0.8985, 0.0029)	(0.8333, 0.0103)	(0.7409, 0.01)	(0.8242, 0.0248)	(0.8152, 0.003)
(202,7)	(0.6154, 0.7939)	(0.6667, 0.8015)	(0.6, 0.7006)	(0.6, 0.7006)	(0.5, 0.6876)	(0.7692, 0.8961)	(0.8, 0.8165)	(0.6, 0.7006)	(0.7273, 0.809)	(0.6667, 0.8015)
ecoli-0-2-6-7_vs_3-5 (9.18)	(0.8125, 0.0037)	(0.9121, 0.0014)	(0.9375, 0.0079)	(0.8125, 0.0069)	(0.8125, 0.0232)	(0.8665, 0.0035)	(0.7415, 0.0127)	(0.8125, 0.0158)	(0.8496, 0.0284)	(0.9206, 0.0026)
(224,7)	(0.7692, 0.7906)	(0.7778, 0.9113)	(0.9333, 0.9354)	(0.7692, 0.7906)	(0.7692, 0.7906)	(0.8, 0.8587)	(0.6154, 0.7011)	(0.7692, 0.7906)	(0.7059, 0.8437)	(0.8235, 0.9194)
ecoli-0-3-4-7_vs_5-6 (9.28)	(1, 0.0023)	(0.9932, 0.0018)	(0.75, 0.0052)	(1, 0.0049)	(0.9932, 0.0219)	(0.9726, 0.0047)	(0.75, 0.0142)	(1, 0.016)	(0.9932, 0.0327)	(0.9795, 0.0034)
(257,7)	(1, 1)	(0.8889, 0.9931)	(0.6667, 0.7071)	(1, 1)	(0.8889, 0.9931)	(0.6667, 0.9722)	(0.6667, 0.7071)	(1, 1)	(0.8889, 0.9931)	(0.7273, 0.9792)
ecoli-0-4-6_vs_5 (9.15)	(0.8894, 0.0024)	(0.9909, 0.0017)	(0.9167, 0.0045)	(0.8333, 0.0041)	(0.8333, 0.0211)	(0.9909, 0.0033)	(0.8333, 0.0085)	(0.9818, 0.0129)	(1, 0.0229)	(1, 0.0017)
(203,6)	(0.7143, 0.8876)	(0.9231, 0.9909)	(0.9091, 0.9129)	(0.8, 0.8165)	(0.8, 0.8165)	(0.9231, 0.9909)	(0.8, 0.8165)	(0.8571, 0.9817)	(1, 1)	(1, 1)
ecoli0137vs26 (4.76)	(0.862, 0.0066)	(0.8803, 0.0018)	(0.8998, 0.0065)	(0.931, 0.0129)	(0.8998, 0.0281)	(0.862, 0.0043)	(0.8933, 0.022)	(0.8685, 0.0309)	(0.8933, 0.044)	(0.8998, 0.0015)
(311,7)	(0.8, 0.8547)	(0.7879, 0.8777)	(0.8667, 0.8955)	(0.9032, 0.9293)	(0.8667, 0.8955)	(0.8, 0.8547)	(0.8387, 0.8896)	(0.8276, 0.8604)	(0.8387, 0.8896)	(0.8667, 0.8955)
ecoli4 (15.8)	(0.8948, 0.0105)	(0.8896, 0.0021)	(0.8896, 0.0074)	(0.8896, 0.0113)	(0.8948, 0.0451)	(0.9948, 0.0056)	(0.9, 0.0208)	(0.8948, 0.0287)	(0.8948, 0.0397)	(0.9844, 0.0027)
(336,7)	(0.8, 0.8898)	(0.7273, 0.8851)	(0.7273, 0.8851)	(0.7273, 0.8851)	(0.8, 0.8898)	(0.9091, 0.9948)	(0.8, 0.8898)	(0.8889, 0.8944)	(0.7692, 0.9843)	(0.7692, 0.9843)
glass5 (22.78)	(0.85, 0.0019)	(0.75, 0.0012)	(0.625, 0.0038)	(0.625, 0.004)	(0.625, 0.0177)	(0.625, 0.0031)	(0.5, 0.0053)	(0.7333, 0.015)	(0.85, 0.023)	(0.7417, 0.0016)
(214,9)	(0.6, 0.8441)	(0.6667, 0.7071)	(0.4, 0.5)	(0.4, 0.5)	(0.4, 0.5)	(0.4, 0.5)	(0, 0)	(0.5, 0.6952)	(0.6, 0.8441)	(0.5714, 0.7012)
haber (2.78)	(0.6516, 0.0048)	(0.6778, 0.002)	(0.7162, 0.0042)	(0.7162, 0.0044)	(0.69, 0.0224)	(0.6547, 0.0045)	(0.5, 0.0121)	(0.6641, 0.0215)	(0.5459, 0.0249)	(0.7085, 0.0008)
(306,3)	(0.5091, 0.6378)	(0.5455, 0.6667)	(0.6, 0.698)	(0.6, 0.698)	(0.56, 0.6684)	(0.5172, 0.6472)	(0, 0)	(0.5373, 0.6641)	(0.4583, 0.475)	(0.5882, 0.6918)
new-thyroid1 (5.14)	(1, 0.0024)	(0.9583, 0.0021)	(1, 0.0036)	(1, 0.0041)	(1, 0.0203)	(1, 0.0031)	(1, 0.0139)	(0.9904, 0.011)	(1, 0.0191)	(1, 0.0014)
(215,5)	(1, 1)	(0.9565, 0.9574)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(1, 1)	(0.96, 0.9903)	(1, 1)	(1, 1)
aus (1.25)	(0.8687, 0.0229)	(0.8848, 0.0134)	(0.8784, 0.0414)	(0.8745, 0.043)	(0.8863, 0.1453)	(0.8763, 0.0276)	(0.8584, 0.0605)	(0.8803, 0.0868)	(0.8809, 0.7946)	(0.8978, 0.0121)
(690,14)	(0.8601, 0.8686)	(0.8788, 0.8847)	(0.8705, 0.8784)	(0.8673, 0.8745)	(0.877, 0.8854)	(0.8713, 0.8758)	(0.8482, 0.8581)	(0.8744, 0.8801)	(0.8756, 0.8805)	(0.8912, 0.8978)
brwiscosin (1.86)	(0.9598, 0.0159)	(0.9704, 0.015)	(0.9778, 0.0263)	(0.9778, 0.0322)	(0.9601, 0.1164)	(0.9598, 0.0296)	(0.9778, 0.0767)	(0.9706, 0.0648)	(0.9558, 0.2168)	(0.9778, 0.0076)
(683,9)	(0.966, 0.9597)	(0.9695, 0.9699)	(0.9773, 0.9775)	(0.9773, 0.9775)	(0.9701, 0.96)	(0.966, 0.9597)	(0.9773, 0.9775)	(0.9736, 0.9705)	(0.9579, 0.9154)	(0.9773, 0.9775)
checkerboard_Data (1.25)	(0.8687, 0.0193)	(0.8848, 0.0143)	(0.8784, 0.029)	(0.8745, 0.041)	(0.8863, 0.1047)	(0.8763, 0.027)	(0.8584, 0.066)	(0.8803, 0.0632)	(0.8809, 0.1706)	(0.8978, 0.0117)
(690,14)	(0.8601, 0.8686)	(0.8788, 0.8847)	(0.8705, 0.8784)	(0.8673, 0.8745)	(0.877, 0.8854)	(0.8713, 0.8758)	(0.8482, 0.8581)	(0.8744, 0.8801)	(0.8756, 0.8805)	(0.8912, 0.8978)
cmc (1.34)	(0.6536, 0.113)	(0.6657, 0.1031)	(0.6483, 0.1106)	(0.6557, 0.1102)	(0.6907, 0.7021)	(0.697, 0.1567)	(0.6248, 0.2579)	(0.6249, 0.2425)	(0.6913, 1.0054)	(0.6832, 0.3102)
(1473,9)	(0.732, 0.6394)	(0.7509, 0.6454)	(0.7293, 0.6329)	(0.7395, 0.6376)	(0.7452, 0.6863)	(0.7626, 0.6873)	(0.6874, 0.6197)	(0.6527, 0.6249)	(0.7387, 0.6888)	(0.7229, 0.6823)
haberman (2.78)	(0.6516, 0.0039)	(0.6795, 0.0019)	(0.5821, 0.0052)	(0.6977, 0.0113)	(0.6302, 0.0282)	(0.684, 0.0038)	(0.5, 0.0159)	(0.551, 0.0264)	(0.5459, 0.0715)	(0.684, 0.0012)
(306,3)	(0.7907, 0.6378)	(0.7563, 0.6794)	(0.7883, 0.5262)	(0.8444, 0.6743)	(0.7069, 0.6302)	(0.7805, 0.6819)	(0.828, 0)	(0.8267, 0.3759)	(0.4091, 0.475)	(0.7805, 0.6819)
heart-stat (1.25)	(0.8702, 0.0023)	(0.8416, 0.0015)	(0.8273, 0.0075)	(0.8165, 0.0051)	(0.8382, 0.0281)	(0.8158, 0.004)	(0.8416, 0.0219)	(0.813, 0.0193)	(0.8519, 0.0322)	(0.8811, 0.0012)
(270,13)	(0.8736, 0.8691)	(0.8539, 0.8415)	(0.8444, 0.8273)	(0.8315, 0.8164)	(0.8571, 0.8381)	(0.8049, 0.8099)	(0.8539, 0.8415)	(0.8352, 0.8129)	(0.8434, 0.847)	(0.8864, 0.8804)
ionn (1.79)	(0.9312, 0.0015)	(0.8974, 0.0016)	(0.9161, 0.0056)	(0.9312, 0.0039)	(0.8951, 0.0597)	(0.9079, 0.0037)	(0.8543, 0.0088)	(0.9516, 0.0164)	(0.9056, 0.0431)	(0.9062, 0.0021)
(351,33)	(0.9466, 0.9312)	(0.9429, 0.8916)	(0.9302, 0.9161)	(0.9466, 0.9312)	(0.9343, 0.892)	(0.9412, 0.9058)	(0.9051, 0.8501)	(0.9618, 0.9516)	(0.9323, 0.905)	(0.912, 0.9052)
monk1 (1.04)	(0.4675, 0.0085)	(0.5, 0.0089)	(0.4534, 0.0237)	(0.4599, 0.0134)	(0.4688, 0.1489)	(0.5, 0.0186)	(0.4654, 0.0459)	(0.5091, 0.0427)	(0.5, 0.0992)	(0.4763, 0.0105)
(556,6)	(0.4767, 0.462)	(0, 0)	(0.4204, 0.453)	(0.4304, 0.4598)	(0.4211, 0.4663)	(0.6255, 0)	(0.4331, 0.4651)	(0.5596, 0.4676)	(0, 0)	(0.4972, 0.4651)
monk2 (1.92)	(0.9343, 0.0247)	(0.9046, 0.0092)	(0.9032, 0.0167)	(0.9032, 0.0168)	(0.9109, 0.0845)	(0.9254, 0.0193)	(0.7336, 0.0394)	(0.8905, 0.051)	(0.9343, 0.1613)	(0.9631, 0.0587)
(601,7)	(0.912, 0.9341)	(0.8636, 0.9045)	(0.8929, 0.898)	(0.8929, 0.898)	(0.8966, 0.908)	(0.8923, 0.9253)	(0.6504, 0.7282)	(0.8696, 0.8865)	(0.912, 0.9341)	(0.9516, 0.963)
monk3 (1.01)	(0.4726, 0.0073)	(0.5, 0.0054)	(0.4433, 0.0227)	(0.4375, 0.0259)	(0.4767, 0.1448)	(0.5, 0.0093)	(0.4454, 0.0338)	(0.5195, 0.0466)	(0.4726, 0.1098)	(0.5, 0.0081)
(554,6)	(0.4238, 0.4678)	(0, 0)	(0.3158, 0.4062)	(0.3134, 0.4024)	(0.4727, 0.4764)	(0, 0)	(0.101, 0.2289)	(0.6376, 0.3259)	(0.4238, 0.4678)	(0.6449, 0)
pima (1.87)	(0.7365, 0.0388)	(0.728, 0.0175)	(0.7428, 0.0295)	(0.7428, 0.0262)	(0.73, 0.0857)	(0.7273, 0.0253)	(0.5146, 0.0629)	(0.7407, 0.077)	(0.7528, 0.203)	(0.7526, 0.031)
(768,8)	(0.6848, 0.7341)	(0.6944, 0.7238)	(0.7053, 0.7416)	(0.7053, 0.7416)	(0.6948, 0.727)	(0.6772, 0.7262)	(0.5831, 0.1709)	(0.707, 0.7368)	(0.7033, 0.7501)	(0.7059, 0.7514)
ripley1 (1)	(0.8854, 0.2039)	(0.8907, 0.0727)	(0.8933, 0.1028)	(0.8933, 0.2714)	(0.8959, 0.5827)	(0.8907, 0.1367)	(0.904, 0.2454)	(0.8721, 0.225)	(0.8933, 0.6607)	(0.904, 0.0911)
(1250,2)	(0.8835, 0.8852)	(0.8883, 0.8904)	(0.893, 0.8933)	(0.893, 0.8933)	(0.8982, 0.8957)	(0.8912, 0.8907)	(0.9048, 0.904)	(0.8689, 0.8717)	(0.8953, 0.8931)	(0.9048, 0.904)
wdbc (3.22)	(0.6979, 0.0053)	(0.7574, 0.0012)	(0.6979, 0.0052)	(0.6548, 0.0035)	(0.6979, 0.0297)	(0.7604, 0.0032)	(0.5818, 0.0108)	(0.7098, 0.0142)	(0.686, 0.0191)	(0.7649, 0.0013)
(194,33)	(0.5625, 0.6847)	(0.6667, 0.7319)	(0.5625, 0.6847)	(0.5, 0.6362)	(0.5625, 0.6847)	(0.6471, 0.7569)	(0.3, 0.4278)	(0.5806, 0.6944)	(0.5455, 0.6748)	(0.6667, 0.752)
yeast-0-2-5-6_vs_3-7-8-9 (9.14)	(0.7408, 0.0325)	(0.7015, 0.0314)	(0.7701, 0.0736)	(0.772, 0.0797)	(0.7706, 0.2384)	(0.7651, 0.0616)	(0.6712, 0.1029)	(0.7903, 0.1336)	(0.7811, 0.3816)	(0.7825, 0.0219)
(1004,8)	(0.4, 0.7343)	(0.5106, 0.6462)	(0.4865, 0.7596)	(0.4932, 0.7611)	(0.5397, 0.7531)	(0.5152, 0.7486)	(0.4762, 0.5932)	(0.5714, 0.7764)	(0.5294, 0.7688)	(0.4872, 0.7756)
yeast-0-2-5-7-9_vs_3-6-8 (9.14)	(0.8574, 0.0584)	(0.9149, 0.0304)	(0.9186, 0.053)	(0.9223, 0.0815)	(0.9113, 0.3055)	(0.889, 0.0628)	(0.7815, 0.1778)	(0.8889, 0.1209)	(0.9334, 0.4735)	(0.9131, 0.1383)
(1004,8)	(0.7719, 0.8484)	(0.7879, 0.9136)	(0.7297, 0.9184)	(0.75, 0.9221)	(0.6923, 0.9112)	(0.7143, 0.8872)	(0.7083, 0.7514)	(0.8, 0.8845)	(0.8182, 0.9328)	(0.7013, 0.913)
yeast-0-3-5-9_vs_7-8 (9.12)	(0.6798, 0.0125)	(0.6222, 0.006)	(0.7374, 0.0096)	(0.8355, 0.0224)	(0.7744, 0.0651)	(0.8483, 0.0123)	(0.5, 0.0292)	(0.7828, 0.0429)	(0.6833, 0.0824)	(0.8286, 0.0023)
(506,8)	(0.3, 0.6317)	(0.2667, 0.5233)	(0.32, 0.7185)	(0.3, 0.8352)	(0.2778, 0.772)	(0.2414, 0.8346)	(0, 0)	(0.1818, 0.752)	(0.3158, 0.634)	(0.2857, 0.8281)
yeast5 (32.73)	(0.9919, 0.0972)	(0.955, 0.0786)	(0.9456, 0.1686)	(0						

Table 4.26: Effect of noise and outliers on datasets.

Dataset	Noise and Outlier ratio	TWSVM [22] AUC	LSTSVM [33] AUC	IFTWSVM [49] AUC	IFTWSVM-ID [51] AUC	pin-IFTWSVM [111] AUC	RFLSTSVM [151] AUC	IIFTWSVM [50] AUC	LS-FLSTSVM-CIL [115] AUC	ACFTSVM [154] AUC	proposed BSFLSTSVM-ID AUC
crossplane130	0.05	1	1	0.9706	0.9706	1	1	0.9706	0.9706	1	1
	0.1	0.9773	1	0.9773	0.9773	1	0.9412	0.9412	0.9773	0.9412	0.9773
	0.15	0.9773	1	1	0.9706	0.9412	0.9318	0.9118	1	1	0.9318
	0.2	0.9251	0.9251	0.9479	0.9251	0.9024	0.9024	0.9184	0.9024	0.9706	0.9706
	0.25	0.8797	0.8797	0.9024	0.9024	0.9024	0.9184	0.8797	0.8797	0.9024	0.9024
ecoli-0-1-4-7_vs_2-3-5-6	0.3	0.8797	0.8797	0.8957	0.9024	0.873	0.766	0.8957	0.8824	0.8797	0.8957
	0.05	0.7085	0.8514	0.8871	0.8814	0.8998	0.8584	0.8571	0.8998	0.8641	0.9183
	0.1	0.6728	0.8641	0.8756	0.8814	0.8814	0.8157	0.8157	0.8883	0.8998	0.8399
	0.15	0.6371	0.6728	0.9056	0.7385	0.8941	0.78	0.8009	0.8998	0.9056	0.8826
	0.2	0.6856	0.6371	0.8756	0.8456	0.78	0.7443	0.5977	0.8042	0.8042	0.8399
ecoli-0-2-3-4_vs_5	0.25	0.7755	0.8099	0.7627	0.8514	0.7385	0.78	0.7028	0.7874	0.8099	0.8481
	0.3	0.5727	0.7385	0.7697	0.8666	0.7697	0.6856	0.7282	0.5956	0.7984	0.8157
	0.05	0.8242	0.8985	0.8061	0.8152	0.8242	0.8242	0.8333	0.797	0.797	0.797
	0.1	0.7409	0.8152	0.7409	0.8242	0.7318	0.7409	0.8242	0.8061	0.8061	0.7879
	0.15	0.8152	0.8242	0.8152	0.7409	0.8061	0.8152	0.8242	0.7409	0.8242	0.9076
ecoli-0-2-6-7_vs_3-5	0.2	0.8152	0.9076	0.8152	0.8803	0.8061	0.7697	0.8333	0.797	0.7879	0.7879
	0.25	0.8242	0.8712	0.8061	0.8242	0.8712	0.7697	0.8333	0.7409	0.8061	0.7879
	0.3	0.8242	0.7879	0.8333	0.8333	0.8333	0.797	0.5833	0.75	0.8061	0.8242
	0.05	0.8125	0.9036	0.875	0.875	0.8125	0.804	0.625	0.8125	0.8665	0.9576
	0.1	0.8581	0.8951	0.875	0.9375	0.875	0.929	0.6102	0.8665	0.8581	0.8242
ecoli-0-3-4-7_vs_5-6	0.15	0.8665	0.929	0.929	0.9121	0.9375	0.929	0.822	0.9746	0.8665	0.8665
	0.2	0.8125	0.8951	0.8125	0.8125	0.8665	0.875	0.8559	0.804	0.875	0.9746
	0.25	0.875	0.929	0.929	0.929	0.8125	0.875	0.6525	0.8125	0.875	0.9375
	0.3	0.8125	0.9206	0.929	0.679	0.8581	0.9036	0.9036	0.929	0.8581	0.8867
	0.05	1	1	0.9932	0.9863	0.9932	1	1	1	0.9795	1
ecoli-0-4-6_vs_5	0.1	0.9658	1	1	1	0.9932	1	0.75	1	0.9863	0.9726
	0.15	0.9658	1	0.9863	0.9932	1	1	1	0.9726	0.9795	0.9932
	0.2	0.9932	0.9932	0.9932	0.875	0.9932	0.9795	0.6575	0.9315	0.9863	1
	0.25	1	0.9863	0.9795	0.5	0.75	0.9795	0.75	0.9795	0.9863	0.9795
	0.3	0.9932	0.875	0.875	0.9932	1	0.9726	0.6952	1	0.9726	0.9521
ecoli0137vs26	0.05	1	0.8333	0.9909	0.8333	1	1	0.8333	1	1	0.9909
	0.1	0.9076	0.9167	0.9167	0.9076	0.9167	0.9909	0.75	1	1	0.9182
	0.15	0.8242	0.8333	0.9818	0.9818	0.9818	0.9636	0.7818	0.8333	0.8333	0.8803
	0.2	0.8333	0.9818	0.9076	0.8621	0.7318	0.8333	0.75	0.8333	0.8242	0.8894
	0.25	0.8242	0.9091	0.8985	0.8894	0.9076	0.8894	0.75	0.9076	0.8894	0.9455
ecoli4	0.3	0.9076	0.8258	0.9076	0.8348	0.9167	0.8985	0.6894	0.9076	0.9076	0.8894
	0.05	0.8243	0.918	0.8373	0.8685	0.8685	0.849	0.8685	0.8373	0.8685	0.8243
	0.1	0.8933	0.8933	0.8555	0.918	0.918	0.8933	0.862	0.7995	0.8868	0.849
	0.15	0.8803	0.8933	0.8803	0.806	0.905	0.8243	0.8438	0.8438	0.849	0.8373
	0.2	0.8308	0.8856	0.8438	0.8438	0.862	0.8373	0.7813	0.8373	0.862	0.8555
haber	0.25	0.8803	0.8803	0.8178	0.862	0.862	0.849	0.7813	0.7435	0.8738	0.8243
	0.3	0.8125	0.8933	0.8803	0.8373	0.8125	0.8438	0.7735	0.8555	0.8308	0.8791
	0.05	0.8844	0.8948	0.8844	0.8948	0.8948	0.8844	0.9	0.874	0.8844	0.8948
	0.1	0.8896	0.8896	0.8948	0.8948	0.8948	0.9844	0.8	0.974	0.8948	0.8896
	0.15	0.8948	0.7948	0.8792	0.8792	0.8948	0.8896	0.7948	0.8948	0.7948	0.7635
new-thyroid1	0.2	0.8948	0.8896	0.8896	0.8948	0.8	0.8896	0.8	0.7	0.8896	0.8896
	0.25	0.8896	0.8635	0.8948	0.8948	0.9583	0.8948	0.7	0.5	0.8896	0.8896
	0.3	0.8844	0.9688	0.9635	0.7896	0.8896	0.9844	0.6	0.9896	0.8896	0.9896
	0.05	0.6949	0.7085	0.6578	0.6547	0.6501	0.7009	0.4678	0.6519	0.6886	0.7009
	0.1	0.6453	0.6254	0.6855	0.6672	0.6932	0.5821	0.6097	0.4923	0.6547	0.6932
	0.15	0.5174	0.6687	0.6439	0.6348	0.6576	0.6704	0.586	0.6165	0.6085	0.6746
	0.2	0.5217	0.5504	0.6838	0.6117	0.5652	0.6345	0.5282	0.5687	0.6655	0.7117
	0.25	0.6239	0.6444	0.6704	0.6641	0.6977	0.6687	0.6105	0.5815	0.6379	0.7225
	0.3	0.5017	0.5698	0.61	0.6302	0.6117	0.6225	0.4954	0.4573	0.6162	0.5823
	0.05	1	0.9808	1	1	1	1	1	1	1	0.875
	0.1	1	0.9904	0.9391	0.8878	0.9808	1	0.9167	1	0.9904	1
	0.15	0.9904	0.9135	0.9904	0.9583	0.9391	0.9583	1	1	0.9904	1
	0.2	1	0.9423	0.9904	0.9167	0.9391	0.9583	0.9583	0.9712	1	0.9904
	0.25	0.9167	0.9167	0.9071	0.9167	0.875	0.9071	0.8333	0.9071	0.9167	0.9071
	0.3	0.9615	0.9615	0.9808	0.9904	1	0.9712	0.9231	0.9327	1	1

Table 4.27: Wilcoxon signed rank test for the proposed BSFLSTSVM-ID in comparison to the baseline models.

UCI dataset					KEEL dataset				
Model	R ₊	R ₋	p-value	Null hypothesis	Model	R ₊	R ₋	p-value	Null hypothesis
TWSVM	27	109	0.034	Rejected	TWSVM	137.5	642.5	0.00044	Rejected
LSTSVM	10	110	0.00452	Rejected	LSTSVM	104.5	525.5	0.00056	Rejected
IFTWSVM	6	165	0.00054	Rejected	IFTWSVM	82.5	583.5	0.00008	Rejected
IFTWSVM-ID	11	125	0.00318	Rejected	IFTWSVM-ID	87	579	0.00012	Rejected
pin-IFTWSVM	0	136	0.00044	Rejected	pin-IFTWSVM	77.5	588.5	<0.00001	Rejected
RFLSTSVM	28	125	0.02144	Rejected	RFLSTSVM	167	536	0.00544	Rejected
IIFTWSVM	0	136	0.00044	Rejected	IIFTWSVM	9	771	<0.00001	Rejected
LS-FLSTSVM-CIL	3	150	0.0005	Rejected	LS-FLSTSVM-CIL	126	694	0.00014	Rejected
ACFTSVM	46	90	.25428	Not rejected	ACFTSVM	176	527	0.00804	Rejected

Table 4.28: Win-tie-loss sign test for the proposed BSFLSTSVM-ID with respect to the baseline models.

Model	Dataset	TWSVM [22]	LSTSVM [33]	IFTWSVM [49]	IFTWSVM-ID [51]	pin-IFTWSVM [111]	RFLSTSVM-CIL [151]	IIFTWSVM [50]	LS-FLSTSVM-CIL [115]	ACFTSVM [154]
Proposed BSFLSTSVM-ID	UCI KEEL	[12,1,4] [30,3,9]	[11,2,4] [28,7,7]	[14,1,2] [29,6,7]	[14,1,2] [27,6,9]	[16,1,0] [31,6,5]	[13,0,4] [26,5,11]	[16,1,0] [37,3,2]	[16,0,1] [33, 2,7]	[10,1,6] [25,5,12]

where S denotes sensitivity, P and Sp signify precision and specificity, respectively, which are expressed as:

$$\text{Precision } (P) = \frac{tp}{tp + fp},$$

$$\text{Sensitivity } (S) = \frac{tp}{tp + fn}, \text{ Specificity } (Sp) = \frac{tn}{tn + fp}.$$

4.4.2.2 Experimental analysis on synthetic datasets

We performed numerical experiments on 12 synthetic KEEL datasets employing an RBF kernel to evaluate the performance of the proposed BSFLSTSVM-ID against baseline models. The results are presented in Table 4.23. The average AUC of the proposed BSFLSTSVM-ID is 0.8505, which is higher, surpassing baseline models which have average AUC of 0.8212, 0.8444, 0.8131, 0.8096, 0.6949, 0.8029, 0.8433, 0.7966, and 0.8357 for TWSVM, LSTSVM, IFTWSVM-ID, IFTWSVM, IIFTWSVM, pin-IFTWSVM, RFLSTSVM-CIL, LS-FLSTSVM-CIL, and ACFTSVM, respectively. The average AUC of the proposed BSFLSTSVM-ID is 0.8505, which is higher, surpassing baseline mod-

Table 4.29: AUC with training time, F-measure and G-mean for the proposed BSFLSTSVM-ID in comparison to baseline models over NDC datasets having RBF kernel.

Model (<i>ir</i>)	LSTSVM [33] (AUC,time(seconds)) (G-mean,F-measure)	RFLSTSVM-CIL [151] (AUC,time(seconds)) (G-mean,F-measure)	LS-FLSTSVM-CIL [115] (AUC,time(seconds)) (G-mean,F-measure)	proposed BSFLSTSVM-ID (AUC,time(seconds)) (G-mean,F-measure)
NDC-10k (2.11)	(0.8944, 6.84)	(0.9364, 9.45)	(0.842, 11.7)	(0.9159, 5.81)
NDC-15k (1.01)	(0.886, 0.8893)	(0.9394, 0.9355)	(0.8415, 0.8368)	(0.92160, 9.147)
NDC-20k (1.05)	(0.6606, 22.19)	(0.9003, 26.39)	(0.7736, 24.34)	(0.8794, 18.31)
NDC-25k (1.32)	(0.7435, 0.5692)	(0.9072, 0.8962)	(0.7926, 0.7671)	(0.87890, 8.794)
NDC-30k (1.09)	(0.8669, 46.74)	(0.9439, 56.2)	(0.8873, 78.64)	(0.9253, 39.49)
NDC-35k (1.4)	(0.8484, 0.8581)	(0.9424, 0.9433)	(0.8861, 0.8869)	(0.93090, 9.243)
NDC-40k (1.73)	(0.7988, 82.27)	(0.9469, 101.52)	(0.7889, 103.61)	(0.9283, 85.43)
NDC-45k (2.15)	(0.7916, 0.7737)	(0.9358, 0.9459)	(0.781, 0.7689)	(0.91920, 9.281)
NDC-50k (1.72)	(0.8392, 141.65)	(0.9339, 171.06)	(0.7165, 210.88)	(0.9344, 153.05)
NDC-60k (1.01)	(0.8099, 0.8247)	(0.9312, 0.9327)	(0.7422, 0.7137)	(0.93640, 9.344)
NDC-70k (2.07)	(0.8268, 208.94)	(0.955, 262.87)	(0.9532, 262.9)	(0.9725, 217.27)
	(0.7907, 0.8086)	(0.9531, 0.954)	(0.954, 0.9527)	(0.97610, 9.725)
	(0.8388, 290.55)	(0.9753, 377.55)	(0.6917, 435.74)	(0.9583, 389.38)
	(0.8093, 0.8238)	(0.9772, 0.9752)	(0.8248, 0.6516)	(0.9710, 9.581)
	(0.9188, 392.43)	(0.9474, 535.32)	(0.8338, 696.37)	(0.9334, 519.91)
	(0.8547, 0.9154)	(0.9027, 0.9463)	(0.7565, 0.8326)	(0.89590, 9.333)
	(0.7914, 549.13)	(0.8353, 695.37)	(0.6521, 935.51)	(0.9264, 660.83)
	(0.7365, 0.7635)	(0.803, 0.819)	(0.8225, 0.5708)	(0.94340, 9.264)
	(0.6017, 1040.51)	(0.8521, 1262.22)	(0.6103, 2350.614)	(0.918, 1162.64)
	(0.7103, 0.4609)	(0.8662, 0.8443)	(0.6668, 0.5841)	(0.9180, 9.18)
	(0.4959, 1401.679)	(0.5035, 1815.252)	*	(0.9238, 1983.59)
	(0.0005, 0.0163)	(0.0456, 0.1521)	*	(0.92860, 9.228)

* Experiment stopped due to taking larger training time of more than 2 days.

els which have average AUC of 0.8212, 0.8444, 0.8131, 0.8096, 0.6949, 0.8029, 0.8433, 0.7966, and 0.8357 for TWSVM, LSTSVM, IFTWSVM-ID, IFTWSVM, IIFTWSVM, pin-IFTWSVM, RFLSTSVM-CIL, LS-FLSTSVM-CIL, and ACFTWSVM, respectively. To mitigate potential biases in the average AUC caused by a model's smaller AUC value over some datasets being compensated by higher values over others, highlighting the limitation of relying solely on the average AUC, ranks are assigned on each dataset for all models. The model with the highest performance will be ranked lowest according to this ranking system, and vice versa. Calculating average of the assigned ranks, we obtain the proposed BSFLSTSVM-ID gets the smallest rank which is 2.75 in comparison to baseline models which have larger average ranks.

4.4.2.3 Experimental analysis on UCI and KEEL datasets

To evaluate the classification effectiveness of the proposed BSFLSTSVM-ID, we conducted experiments using 17 UCI and 42 KEEL datasets. Table 4.24 showcases the perfor-

mance metric of the models on UCI datasets. The descriptions of each dataset including the number of samples, features and imbalance ratio are presented in Table 4.24. Additionally, the F-measure and G-mean, along with training time for all models are presented. Furthermore, we also considered the rank of models across all datasets. The average AUC for proposed BSFLSTSVM-ID over the UCI dataset is 0.8172 followed by ACFTSVM with AUC 0.7977, RFLSTSVM-CIL with 0.7976, LSTSVM with 0.7966, IFTWSVM-ID with 0.7937, IFTWSVM with 0.7916, TWSVM with 0.7911, pin-IFTWSVM with 0.7798, LS-FLSTSVM-CIL with 0.7512, and IIFTWSVM with AUC 0.7012. With an average rank of 2.59, the proposed BSFLSTSVM-ID exhibits the lowest average rank, the second smallest average rank is acquired by ACFTSVM with 4.68, and third best is shown by RFLSTSVM-CIL which is 4.94. The remaining models in increasing order of average rank are as follows: LSTSVM (5), IFTWSVM (5.35), TWSVM (5.32), IFTWSVM-ID (5.65), pin-IFTWSVM (5.76), LS-FLSTSVM-CIL (6.56) and IIFTWSVM (9.15). Hence, the greatest average AUC with the least average rank demonstrate the supremacy of the proposed BSFLSTSVM-ID model over UCI datasets.

Table 4.25 displays the models' performance metrics corresponding to the KEEL datasets. It contains performance metric over each dataset for all the models along with the number of samples, features and *ir* (varying between 1 to 72.69) of the datasets contained in the first column of the Table. The proposed BSFLSTSVM-ID has an average AUC of 0.8529, that is larger in comparison to the baseline models with average AUC 0.8304, 0.8219, 0.8216, 0.8154, 0.8117, 0.8119, 0.8114, 0.8108 and 0.7418 for RFLSTSVM-CIL, LSTSVM, ACFTSVM, LS-FLSTSVM-CIL, IFTWSVM-ID, pin-IFTWSVM, IFTWSVM, TWSVM, and IIFTWSVM, respectively. At 3.15, the proposed BSFLSTSVM-ID possess the least average rank, followed by ACFTSVM, which has an average rank of 4.65, and RFLSTSVM-CIL with an average rank of 5.11. The average ranks of the remain-

ing baseline models in increasing order are as follows: 5.36, 5.56, 5.69, 5.71, 5.79, 5.81, 8.17 for LSTSVM, IFTWSVM, pin-IFTWSVM, TWSVM, IFTWSVM-ID, LS-FLSTSVM-CIL, IIFTWSVM, respectively. The training times of baseline algorithms and the proposed BSFLSTSVM-ID for different KEEL datasets are shown in Figure 4.10. The figure indicates that the proposed BSFLSTSVM-ID generally exhibits shorter training time than the baseline models, except with LSTSVM which presents a competitive training time. However, the proposed BSFLSTSVM-ID demonstrates superior performance across the majority of the datasets in comparison to all the baseline models.

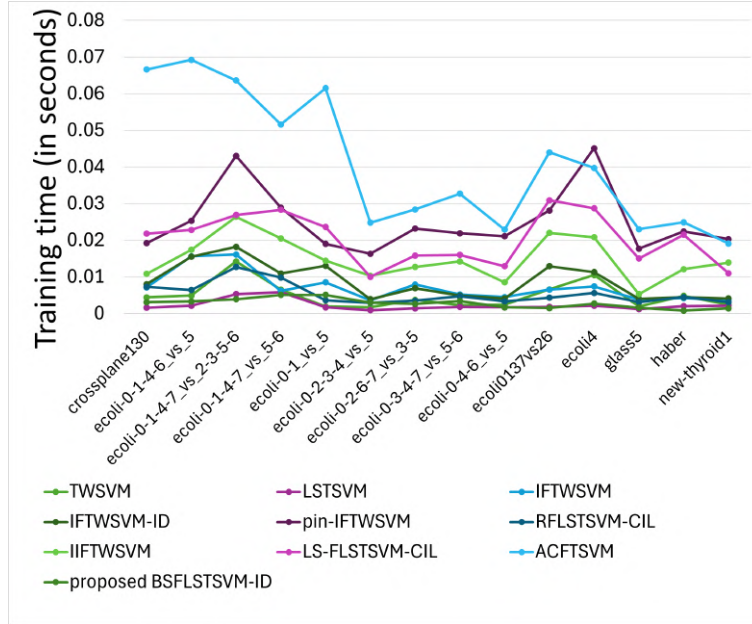


Figure 4.10: Comparison of time taken for training of the proposed BSFLSTSVM-ID and baseline models over KEEL datasets with RBF kernel.

We carried out numerical experiments of the proposed BSFLSTSVM-ID to assess its performance in the presence of impure data points. We injected noise and outliers into the KEEL datasets to execute the experiments, and the performance is compared with baseline algorithms. The numerical experiments' results are shown in Table 4.25. The experiments involved varying noise and outlier ratios to evaluate the model's robustness. The ratio of the variance of noise to that of each feature considered in the experiment

is 0.05, 0.1, 0.15, 0.2, 0.25, 0.3 [82]. Further, the ratio of outliers occupying the whole training set 5% to 30% with step size 5% [158]. Table 4.26 distinctly presents the model performance under different noise and outlier conditions, affirming that the proposed BSFLSTSVM-ID exhibits resistance to noise and outliers compared to the baseline models.

4.4.2.4 Parameter sensitivity

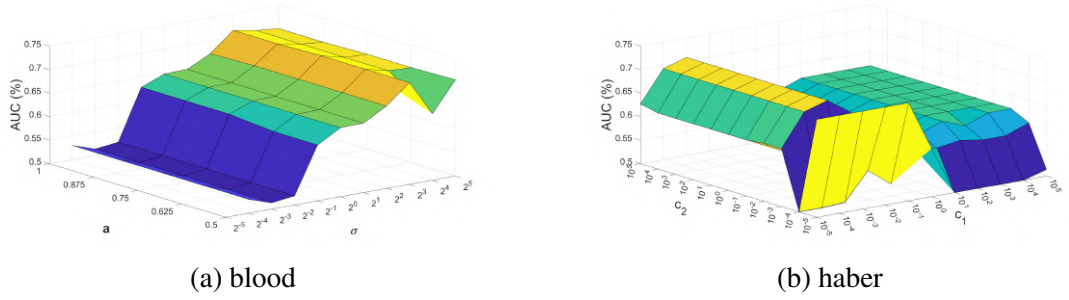


Figure 4.11: Evaluation of AUC values of proposed BSFLSTSVM-ID while varying the parameters C_1 , C_2 , σ and a .

In this section, we delve into the parameter sensitivity analysis of the proposed BSFLSTSVM-ID. The model incorporates hyperparameters C_1 and C_2 , a kernel parameter σ , and membership parameters a and λ . In Figure 4.11, we illustrate the variation in AUC values for UCI datasets blood and haber by systematically altering different pairs of parameters. In Figure 4.11a, we showcase the AUC variation for the blood dataset by adjusting the hyperparameter pair a and σ , while maintaining other parameters at their optimal values. Figure 4.11b illustrates the impact on the performance of the haber dataset by varying the hyperparameter pair C_1 and C_2 . These visualizations underscore the pivotal role of selecting optimal parameters in achieving superior performance within the proposed BSFLSTSVM-ID.

Furthermore, to unveil the dependency of the proposed BSFLSTSVM-ID on individual parameters, we conducted experiments on the synthetic data paw02a-800-7-60-BI. We

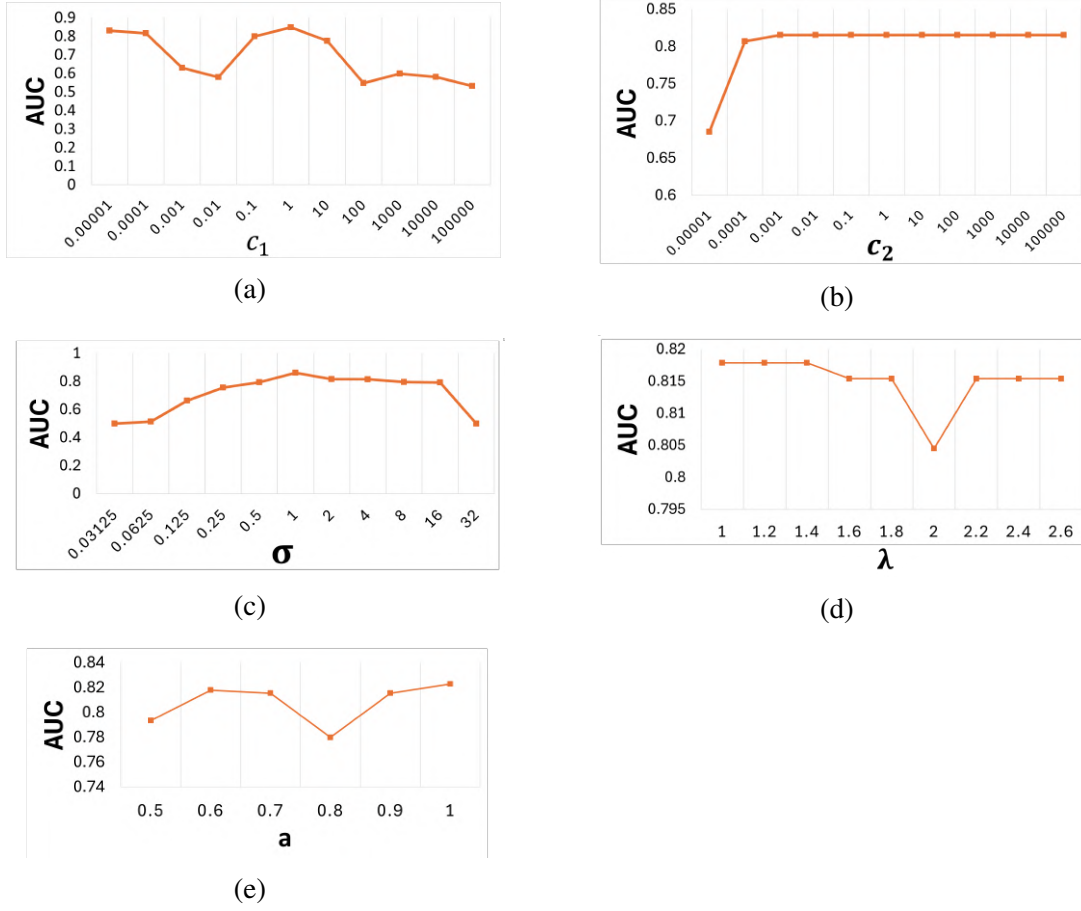


Figure 4.12: Effect of variation in the parameters C_1 , C_2 , σ , λ and a on the AUC values of proposed BSFLSTSVM-ID for synthetic dataset paw02a-800-7-60-BI.

systematically varied each parameter within a defined range, while maintaining other parameters at optimal values. Ranges of variation for different parameters are set as follows: C_1 and C_2 in the range $[10^{-5}, 10^{-4}, \dots, 10^4, 10^5]$, σ in the range $[2^{-5}, 2^{-4}, \dots, 2^4, 2^5]$, λ in the range $[1, 1.2, 1.4, \dots, 2.4, 2.6]$, and a in the range $[0.5, 0.6, 0.7, 0.8, 0.9, 1]$. Figure 4.12 illustrates the corresponding variation in AUC values for paw02a-800-7-60-BI with each parameter. These figures distinctly highlight the critical importance of parameter selection in achieving optimal performance.

4.4.2.5 Statistical analysis

This subsection statistically analyses the experimental results. We perform statistical tests, including the win-tie-loss sign test, Wilcoxon signed rank test, Nemenyi-post-hoc test, and Friedman test [123].

Friedman's test assumes a null hypothesis that the proposed and baseline models are equivalent. It is calculated using the statistics shown in equation (2.56). The F_F statistic is calculated as shown in equation 2.57. Initially, we conduct a statistical analysis of the results obtained from the UCI datasets followed by presenting it on KEEL datasets. For the UCI data, we have $N = 17$ and $md = 10$, hence $\chi_F^2 = 45.0566$ and $F_F = 6.6785$ with d.o.f. 9 and (9, 144), respectively. $F_{(9,144)} = 1.9454$ which is comparatively less than F_F , leading to the rejection of the null hypothesis, indicating that the models are not equivalent.

We next use the Nemenyi post-hoc test to make pairwise comparisons in order to further examine the models' performance. The critical difference is determined by the Nemenyi test using equation (2.58) with $\alpha = 0.05$, i.e., at 5% level of significance. If the difference in the average ranks of the two models surpasses the critical difference, it signifies that the models are statistically distinct. Here, the calculated value of critical difference is 3.28. Hence, the baseline models IFTWSVM and LS-FLSTSVM-CIL exhibit statistical differences from the proposed BSFLSTSVM-ID. Nevertheless, additional tests, average AUC and average ranks consistently indicate the supremacy of the proposed BSFLSTSVM-ID with respect to remaining models. Next, we apply the Wilcoxon signed rank test on the AUC values across UCI datasets, assuming that each baseline model is equivalent to the proposed BSFLSTSVM-ID. The Wilcoxon test assigns ranks to the performance differences between two models based on their magnitudes. Let R_- and R_+ represent the sums of the corresponding negative and positive ranks. The null hypothesis is rejected if the calculated p -value for a pair of models is less than 0.05. Table 4.27 demonstrates the Wilcoxon

Table 4.30: AUC with training time (in seconds), F-measure and G-mean values of the proposed BSFLSTSVM-ID and baseline models on BreakHis and ADNI datasets with RBF kernel.

Dataset (<i>ir</i>) (samples,features)	TWSVM [22] (AUC,train time) (<i>G, F</i>)	LSTSVM [33] (AUC,train time) (<i>G, F</i>)	IFTWSVM [49] (AUC,train time) (<i>G, F</i>)	IFTWSVM-ID [51] (AUC,train time) (<i>G, F</i>)	pin-IFTWSVM [111] (AUC,train time) (<i>G, F</i>)	RFLSTSVM-CIL [151] (AUC,train time) (<i>G, F</i>)	IFTWSVM [50] (AUC,train time) (<i>G, F</i>)	LS-FLSTSVM-CIL (AUC,train time) (<i>G, F</i>)	ACFTSVM [154] (AUC,train time) (<i>G, F</i>)	proposed BSFLSTSVM-ID (AUC,train time) (<i>G, F</i>)
BreakHis dataset										
AD.vs_DC (1.96) (314,768)	(0.7985, 0.017) (0.7458, 0.784)	(0.8049, 0.0208) (0.7541, 0.7946)	(0.9103, 0.0535) (0.8684, 0.9083)	(0.8868, 0.0612) (0.8615, 0.8845)	(0.9098, 0.1013) (0.8923, 0.908)	(0.8049, 0.0331) (0.7541, 0.7946)	(0.8005, 0.068) (0.75, 0.7793)	(0.725, 0.0551) (0.6733, 0.6708)	(0.9078, 0.1482) (0.8824, 0.9075)	(0.9373, 0.0137) (0.9143, 0.9372)
AD.vs_LC (1.29) (243,768)	(0.5015, 0.0081) (0.4194, 0.4843)	(0.5323, 0.0049) (0.5, 0.5303)	(0.4628, 0.0247) (0.0952, 0.2228)	(0.5, 0.0238) (0.6481, 0)	(0.4628, 0.0361) (0.0952, 0.2228)	(0.4789, 0.02) (0.3793, 0.4637)	(0.5, 0.0197) (0, 0)	(0.5481, 0.0394) (0.6458, 0.4318)	(0.5481, 0.0364) (0.6458, 0.4318)	(0.6511, 0.0054) (0.6905, 0.6265)
AD.vs_MC (1.59) (275,768)	(0.5952, 0.0077) (0.4255, 0.5455)	(0.5661, 0.0047) (0.4286, 0.5492)	(0.6131, 0.0143) (0.4583, 0.5722)	(0.6131, 0.0132) (0.4583, 0.5722)	(0.6118, 0.0367) (0.4444, 0.5634)	(0.5847, 0.0161) (0.4789, 0.5611)	(0.5628, 0.0179) (0.4, 0.5079)	(0.5099, 0.0282) (0.4898, 0.5976)	(0.631, 0.0398) (0.4727, 0.5864)	(0.6025, 0.0057) (0.4727, 0.5864)
AD.vs_PC (1.3) (244,768)	(0.571, 0.0052) (0.4828, 0.5584)	(0.5795, 0.0047) (0.5161, 0.576)	(0.6286, 0.0109) (0.6111, 0.6234)	(0.6321, 0.0103) (0.597, 0.632)	(0.5829, 0.0221) (0.4912, 0.568)	(0.5795, 0.0129) (0.5161, 0.576)	(0.5223, 0.0195) (0.5682, 0.4382)	(0.6217, 0.0245) (0.6341, 0.5826)	(0.571, 0.0366) (0.4828, 0.5584)	(0.6617, 0.0066) (0.6667, 0.6277)
FB.vs_DC (1.14) (445,768)	(0.8107, 0.0133) (0.8252, 0.8104)	(0.7775, 0.0129) (0.7761, 0.7762)	(0.8348, 0.0389) (0.8429, 0.8348)	(0.8348, 0.0343) (0.8429, 0.8348)	(0.8258, 0.0446) (0.8392, 0.8255)	(0.7775, 0.0412) (0.7761, 0.7762)	(0.8208, 0.0692) (0.8261, 0.8206)	(0.6699, 0.068) (0.5741, 0.628)	(0.7503, 0.1435) (0.7344, 0.7451)	(0.8117, 0.0119) (0.8227, 0.8117)
FB.vs_LC (1.73) (374,768)	(0.6319, 0.0071) (0.795, 0.5774)	(0.6042, 0.0071) (0.807, 0.4895)	(0.5917, 0.0174) (0.8023, 0.4644)	(0.5792, 0.0168) (0.7977, 0.4378)	(0.6778, 0.0328) (0.7838, 0.6656)	(0.6042, 0.0244) (0.807, 0.4895)	(0.6583, 0.0417) (0.8148, 0.6055)	(0.6986, 0.043) (0.8079, 0.6826)	(0.6528, 0.0525) (0.8171, 0.5907)	(0.6597, 0.0029) (0.7299, 0.6588)
FB.vs_MC (1.4) (406,768)	(0.5201, 0.0069) (0.6864, 0.4187)	(0.5508, 0.0073) (0.7303, 0.4009)	(0.5923, 0.0282) (0.7089, 0.5547)	(0.569, 0.0155) (0.6434, 0.5621)	(0.5363, 0.0332) (0.7322, 0.3323)	(0.5508, 0.0285) (0.7303, 0.4009)	(0.5508, 0.0286) (0.5128, 0.5371)	(0.4981, 0.0444) (0.4706, 0.4883)	(0.6115, 0.0508) (0.4421, 0.5262)	(0.5907, 0.0069) (0.6475, 0.5883)
FB.vs_PC (1.72) (375,768)	(0.5139, 0.0071) (0.717, 0.4564)	(0.5943, 0.0067) (0.6715, 0.5942)	(0.6382, 0.0226) (0.7712, 0.623)	(0.6462, 0.0222) (0.7568, 0.6398)	(0.6586, 0.0375) (0.7843, 0.6455)	(0.6038, 0.0248) (0.7532, 0.5824)	(0.6637, 0.0498) (0.8125, 0.6355)	(0.5051, 0.0389) (0.6187, 0.5015)	(0.5738, 0.0536) (0.4771, 0.525)	(0.6564, 0.021) (0.8148, 0.6213)
PT.vs_DC (1.81) (323,768)	(0.8176, 0.0062) (0.7778, 0.8136)	(0.8392, 0.0086) (0.8056, 0.8358)	(0.8439, 0.0161) (0.8108, 0.8421)	(0.8702, 0.0153) (0.8421, 0.8698)	(0.857, 0.0343) (0.8267, 0.8561)	(0.8392, 0.0216) (0.8056, 0.8358)	(0.857, 0.0313) (0.8267, 0.8561)	(0.8307, 0.0395) (0.7945, 0.828)	(0.8401, 0.0525) (0.8052, 0.8397)	(0.8439, 0.006) (0.8108, 0.8421)
PT.vs_LC (1.19) (252,768)	(0.5161, 0.0042) (0.0625, 0.1796)	(0.519, 0.0043) (0.2727, 0.4043)	(0.5358, 0.009) (0.4776, 0.5355)	(0.4814, 0.0093) (0.4179, 0.4804)	(0.486, 0.0229) (0.5417, 0.3344)	(0.519, 0.0132) (0.2727, 0.4043)	(0.5025, 0.0145) (0.4571, 0.5023)	(0.5, 0.0252) (0.5794, 0)	(0.5294, 0.0298) (0.5773, 0.3748)	(0.4978, 0.002) (0.1951, 0.3344)
PT.vs_MC (1.47) (284,768)	(0.631, 0.0052) (0.5313, 0.6001)	(0.5867, 0.0046) (0.4, 0.5014)	(0.6516, 0.0111) (0.6027, 0.6457)	(0.6407, 0.0108) (0.5946, 0.6361)	(0.6664, 0.0222) (0.6316, 0.6644)	(0.5867, 0.0146) (0.4, 0.5014)	(0.6407, 0.0156) (0.5946, 0.6361)	(0.6497, 0.0308) (0.5915, 0.6401)	(0.631, 0.0368) (0.5313, 0.6001)	(0.6753, 0.0047) (0.6301, 0.6699)
PT.vs_PC (1.2) (253,768)	(0.5462, 0.0045) (0.3333, 0.4491)	(0.5784, 0.0041) (0.4231, 0.5192)	(0.5238, 0.0101) (0.4857, 0.5233)	(0.5357, 0.0095) (0.4928, 0.5345)	(0.631, 0.0257) (0.5574, 0.6172)	(0.6022, 0.0121) (0.44, 0.5339)	(0.5567, 0.0168) (0.6408, 0.3724)	(0.6989, 0.0239) (0.6452, 0.6901)	(0.5749, 0.0284) (0.4746, 0.5513)	(0.6646, 0.0073) (0.6824, 0.6373)
TA.vs_DC (1.6) (338,768)	(0.6455, 0.0067) (0.4906, 0.5945)	(0.6524, 0.0056) (0.4898, 0.585)	(0.6905, 0.0196) (0.566, 0.6489)	(0.6979, 0.0124) (0.5769, 0.6541)	(0.7161, 0.0272) (0.6216, 0.7159)	(0.6524, 0.0198) (0.4898, 0.585)	(0.707, 0.0277) (0.6032, 0.6947)	(0.6941, 0.0357) (0.5974, 0.6941)	(0.6854, 0.0475) (0.5797, 0.6808)	(0.6832, 0.0069) (0.5556, 0.6438)
TA.vs_LC (1.05) (267,768)	(0.7621, 0.0046) (0.7765, 0.7621)	(0.7913, 0.0045) (0.8298, 0.7828)	(0.7681, 0.0123) (0.8043, 0.7625)	(0.7467, 0.0142) (0.7727, 0.7454)	(0.7583, 0.0222) (0.7865, 0.7563)	(0.7913, 0.0131) (0.8298, 0.7828)	(0.6125, 0.015) (0.4211, 0.5138)	(0.7797, 0.0291) (0.8172, 0.7727)	(0.8067, 0.0284) (0.8352, 0.803)	(0.7913, 0.0024) (0.8298, 0.7828)
TA.vs_MC (1.3) (299,768)	(0.6824, 0.0048) (0.641, 0.6785)	(0.6926, 0.0051) (0.6494, 0.6877)	(0.6946, 0.0127) (0.6582, 0.692)	(0.6824, 0.0103) (0.641, 0.6785)	(0.6478, 0.0237) (0.5974, 0.642)	(0.6926, 0.015) (0.6494, 0.6877)	(0.5607, 0.0156) (0.3509, 0.4626)	(0.668, 0.0289) (0.6512, 0.6678)	(0.6391, 0.0346) (0.6667, 0.6102)	(0.6068, 0.0028) (0.6087, 0.602)
TA.vs_PC (1.06) (268,768)	(0.5683, 0.0048) (0.6222, 0.5657)	(0.5984, 0.0044) (0.6737, 0.5877)	(0.6317, 0.0149) (0.6341, 0.6294)	(0.5968, 0.0092) (0.6364, 0.5963)	(0.6079, 0.022) (0.6517, 0.6068)	(0.5905, 0.0137) (0.6522, 0.5855)	(0.6127, 0.0152) (0.4516, 0.5333)	(0.4937, 0.0292) (0.3385, 0.4261)	(0.6397, 0.0274) (0.7327, 0.6131)	(0.6698, 0.0139) (0.7736, 0.6249)
Average AUC	0.632	0.6417	0.6632	0.6571	0.6648	0.6424	0.6331	0.6307	0.662	0.6877
Average rank	7.47	6.16	4.25	5.28	4.63	6.34	5.97	6.5	4.81	3.59
ADNI dataset										
CN.vs_AD (1.22) (415,91)	(0.8736, 0.0206) (0.8515, 0.8731)	(0.8697, 0.0122) (0.8462, 0.8697)	(0.8834, 0.0444) (0.8627, 0.8832)	(0.8834, 0.0615) (0.8627, 0.8832)	(0.901, 0.0462) (0.8866, 0.8992)	(0.8864, 0.0232) (0.8654, 0.8864)	(0.8033, 0.067) (0.7769, 0.7945)	(0.8775, 0.0756) (0.8571, 0.8759)	(0.9128, 0.0797) (0.8952, 0.9128)	(0.903, 0.0176) (0.8846, 0.903)
CN.vs_MCI (1.75) (626,91)	(0.6219, 0.0158) (0.7941, 0.5784)	(0.635, 0.0146) (0.7833, 0.6101)	(0.673, 0.029) (0.7742, 0.6679)	(0.7006, 0.0359) (0.8078, 0.6918)	(0.6621, 0.0554) (0.6231, 0.6389)	(0.6642, 0.0251) (0.7843, 0.6527)	(0.6461, 0.0988) (0.6061, 0.6223)	(0.6665, 0.0819) (0.735, 0.6664)	(0.7071, 0.1037) (0.7586, 0.707)	(0.6743, 0.0372) (0.7458, 0.6741)
MCI.vs_AD (2.13) (585,91)	(0.6587, 0.0293) (0.5669, 0.6503)	(0.635, 0.0101) (0.5594, 0.6347)	(0.6615, 0.0209) (0.562, 0.6469)	(0.615, 0.0264) (0.4673, 0.5702)	(0.6462, 0.0513) (0.5378, 0.6276)	(0.6262, 0.0222) (0.5085, 0.6042)	(0.6455, 0.0441) (0.5049, 0.597)	(0.6682, 0.0567) (0.5865, 0.6647)	(0.6178, 0.0849) (0.5294, 0.6145)	(0.6643, 0.0222) (0.5565, 0.6417)
Average AUC	0.7181	0.7132	0.7393	0.733	0.7364	0.7256	0.6983	0.7374	0.7459	0.7472
Average Rank	7.33	8.33	4.17	5.83	5	6	8	4.33	3.67	2.33

tests' results over UCI datasets and rejects the null hypothesis assumption with each model except ACFTSVM. However, the numerical results depict the supremacy of the proposed BSFTSVM-ID over ACFTSVM. Hence, the proposed BSFLSTSVM-ID is statically different from the baseline models.

Further to strengthen our analysis, we employ the win-tie-loss sign test based on the performance of the models. If the number of wins of a model is greater than $\frac{N}{2} + \frac{1.96\sqrt{N}}{2}$ in comparison to that of another model, the model is significantly better than the other. If there's an even number of ties between the pair of models, they are divided equally. If the number of ties is odd, distribute them as evenly as possible, disregarding any remainder. For the case of UCI datasets, the calculated value is 12.54, i.e., if the number of wins of the proposed BSFLSTSVM-ID is greater than 12.54, then it is significantly better. Table 4.28 (first row) clearly illustrates the number of wins, ties and losses of the proposed BSFLSTSVM-ID over the baseline models. According to the win-tie-loss test, the proposed BSFLSTSVM-ID is statistically better concerning all the baseline models except TWSVM, LSTSVM and ACFTSVM. Hence, the proposed model is significantly better than baseline models over the UCI datasets.

For the results over the KEEL dataset, having $N = 42$ and $md = 10$, the calculated $\chi_F^2 = 63.2731$, $F_F = 8.2427$ with d.o.f. 9 and (9, 369), respectively. Since $F_F > F_{(9,369)} = 1.9053$ in this case, the Friedman test's null hypothesis assumption is rejected, and the models are not similar. The calculated critical difference value across KEEL datasets is 2.09. The rank difference is greater than the calculated critical difference, which implies baseline models except ACFTSVM and RFLSTSVM-CIL are statistically different from the proposed BSFLSTSVM-ID. The difference between the proposed model with ACFTSVM and RFLSTSVM-CIL is shown through other statistical tests and numerical experiments. Further, the Wilcoxon test is applied to the models over KEEL data which are depicted in Ta-

ble 4.27 right part which rejects the null hypothesis and concludes the statistical difference of the proposed BSFLSTSVM-ID. Next, using the win-tie-loss sign test, we calculate that 27.35 wins are needed for the proposed BSFLSTSVM-ID model to differ statistically from the baseline models. Table 4.28 (second row) presents the win-tie-loss statistic over the KEEL dataset, for each case, the proposed BSFLSTSVM-ID is statistically better as the number of wins is greater than 27.35. Hence, the overall statistical analysis and numerical results highlight the supremacy of the proposed BSFLSTSVM-ID.

4.4.2.6 Experimental analysis on NDC datasets

To demonstrate the applicability of the proposed BSFLSTSVM-ID on large-scale data, we conducted numerical experiments over NDC datasets having training and testing samples in the ratio of 50 : 50. We regarded the dataset as comprising samples ranging from 15,000 to 70,000. Due to the large sample size, the experiments are conducted in a PC with Intel(R) Xeon(R) Platinum 8260 CPU @ 2.30GHz 2.30 GHz processor and 256 GB RAM. We have compared the proposed BSFLSTSVM-ID with LSTSVM [33], RFLSTSVM-CIL [151] and LS-FLSTSVM-CIL [115], owing to their effectiveness over large-scale datasets. The parameters C_1 , C_2 (c_0 in case of RFLSTSVM-CIL) and σ of baseline models and the proposed BSFLSTSVM-ID are fixed at 10^{-2} , 10^{-5} and 2^4 , respectively. The membership parameter a and λ of the proposed BSFLSTSVM-ID are fixed at 0.5 and 1, respectively. Table 4.29 illustrates the AUC and training time for the models. For the data of sizes 30k, 35k, 50k and 60k, the proposed BSFLSTSVM-ID has greatest AUC. For the remaining data of sizes 10k, 15k, 20k, 25k, 40k and 45k, the proposed BSFLSTSVM-ID has the second-best performance following RFLSTSVM-CIL. However, the training time taken by the proposed BSFLSTSVM-ID is less than that of LS-FLSTSVM-CIL and RFLSTSVM-CIL. The training time for LSTSVM is the smallest for datasets having a size in the range of 30k to 60k,

owing to no membership assignment in the model. It reveals the proposed model has less training time and comparable AUC value as that of LSTSVM, RFLSTSVM-CIL and LS-FLSTSVM-CIL. Thus, the proposed BSFLSTSVM-ID model is efficient in terms of training time and gives tough competition to the performance of the baseline models.

4.4.3 Biomedical applications

In this section, we carried out numerical experiments involving the proposed BSFLSTSVM-ID and baseline models over BreakHis dataset for breast cancer tumor classification and the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset for identifying subjects with Alzheimer's disease.

Microscopic biopsy images of both benign and malignant breast cancers are available in the BreakHis database [130]. For the acquisition of images, the pathologist first identified the region of interest (ROI) then several images of ROI are captured using different magnifications, i.e., $40\times$, $100\times$, $200\times$ and $400\times$. In our analysis, we used 1240 images with $400\times$ magnification. The database is divided into two types of tumours, malignant (cancerous) tumors and benign (non-cancerous) tumors. Based on the characteristics of the tumoral cells under the microscope, benign and malignant breast tumors are further divided into multiple types. Benign tumors are divided into adenosis (AN), fibroadenoma (FB), phyllodes tumor (PT) and tubular adenoma (TA). Malignant tumors are divided into following four types: mucinous carcinoma (MC), ductal carcinoma (DC), papillary carcinoma (PC) and lobular carcinoma (LC). The images are transformed into grayscale versions, and each one is subjected to upto three stages of decomposition using the Daubechies-4 wavelet. 16 pairs of benign and malignant data are made to distinguish between the 4 subtypes of malignant cancer. The goal of classification models is to classify the malignant tumors from benign tumors.

The AD data used in this work belongs to the ADNI repository. Under the direction of Principal Investigator Michael W. Weiner, ADNI was established in 2003 as a public-private partnership. The major goal of ADNI is to examine how neuroimaging methods like magnetic resonance imaging (MRI), positron emission tomography (PET), and clinical neuropsychological tests are used to assess the development of AD from the condition of moderate cognitive impairment. For further information, please refer to www.adni-info.org. We followed the ADNI baseline dataset [159] and processed the data similar to [144]. We used 817 structural MRI images by utilising the recon-all pipeline (version 6.0.1) of FreeSurfer to perform volume-based morphometry on the images [160]. Four images out of 817 failed to process using the FreeSurfer pipeline. Thus, we utilise 398 subjects with mild cognitive impairment (MCI), 187 subjects with AD, and 228 subjects who are control normal (CN). The classification models are utilized to differentiate between distinct categories of subjects.

4.4.3.1 Numerical experiments

To conduct the experiments on the BreakHis data and AD data, we have followed the same experimental setup as discussed in subsection 4.4.2.1.

BreakHis dataset: Table 4.30 (first part) consists of the AUC, G-mean, F-measure and training time over the breast cancer dataset. For AD vs DC, AD vs LC, AD vs MC, PT vs MC, and TA vs PC, the proposed BSFLSTSVM-ID has the largest average AUC of 0.9373, 0.6511, 0.6025, 0.6753, and 0.6698, respectively.

The AUC value of FB vs MC, FB vs PC, PT vs PC, and TA vs LC for the proposed BSFLSTSVM-ID lies at the second position after baseline models. For the remaining cases, though the performance of the baseline models is better, Nevertheless, the proposed BSFLSTSVM-ID has the highest average AUC at 0.6877, than baseline models (IFTWSVM-ID, TWSVM, LSTSVM, IFTWSVM, pin-IFTWSVM, RFLSTSVM-CIL,

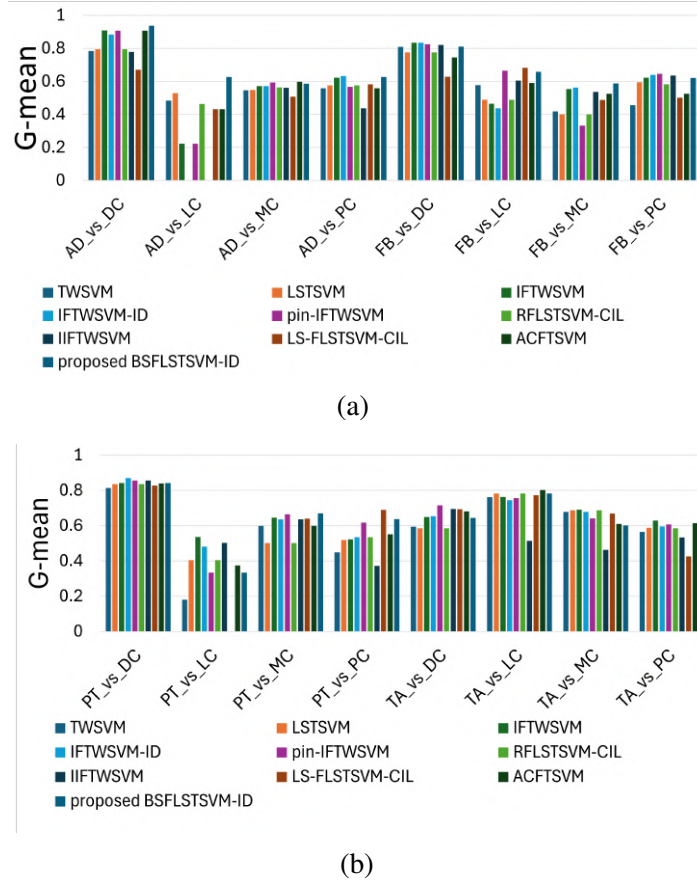


Figure 4.13: Comparison of G-mean values of the proposed BSFLSTSVM-ID and baseline models on BreakHis datasets.

IIFTWSVM, LS-FLSTSVM-CIL and ACFTSVM with average AUC 0.6571, 0.632, 0.6417, 0.6632, 0.6648, 0.6424, 0.6331, 0.6307, and 0.662, respectively). Additionally, the proposed BSFLSTSVM-ID has the smallest average rank of 3.59. followed by the baseline algorithm 5.28, 7.47, 6.16, 4.25, 4.63, 6.34, 5.97, 6.5, and 4.81 for the aforementioned order of models. Therefore, the average AUC and average rank demonstrate the superior performance of the proposed BSFLSTSVM-ID for the diagnosis of breast cancer. Figure 4.13 plots the G-mean of models using bar graph on the breast cancer dataset which demonstrates the better performance for the proposed BSFLSTSVM-ID.

AD dataset: The results of numerical experiments over the AD dataset are shown in Table 4.30 (second part). For the CN vs AD case, the proposed BSFLSTSVM-ID has the sec-

ond highest AUC 0.903 after ACFTSVM which has an AUC of 0.9128. The performance of the pin-IFTWSVM over CN vs AD is 0.901, RFLSTSVM-CIL with 0.8864. For the CN vs MCI case, the proposed BSFLSTSVM-ID has the third-best performance with AUC 0.6743 following ACFTSVM (0.7071) and IFTWSVM-ID (0.7006). The AUC of other baseline models over CV vs MCI in the decreasing order as IFTWSVM, LS-FLSTSVM-CIL, RFLSTSVM-CIL, pin-IFTWSVM, IIFTWSVM, LSTSVM and TWSVM. For the MCI vs AD case, the proposed BSFLSTSVM-ID has the second-best performance following LS-FLSTSVM-CIL with 0.6682. The average AUC of baseline algorithms are the following: 0.733, 0.7181, 0.7132, 0.7393, 0.7364, 0.7256, 0.6983, 0.7374, and 0.7459 for IFTWSVM-ID, TWSVM, LSTSVM, IFTWSVM, pin-IFTWSVM, RFLSTSVM-CIL, IIFTWSVM, LS-FLSTSVM-CIL, and ACFTSVM, respectively, whereas the proposed BSFLSTSVM-ID has the highest average AUC of 0.7472. The average rank of the proposed model is 2.33, which is less than the average rank of the baseline models. Thus, the proposed BSFLSTSVM-ID has superior performance in classifying breast cancer tumors and diagnosing AD, than the baseline models.

BSFLSTSVM-ID utilizes a smooth bell-shaped fuzzy function (CPBS) within an efficient least squares twin SVM framework. It ensures that representative samples are prioritized, while uncertainty and class imbalance are handled through intuitive scoring and reweighting.

4.5 Interpretability embedded in model design

This section highlights the interpretability of the proposed models in this chapter.

- [1] Intuitionistic fuzzy numbers (IFNs) and dual center-based IFNs: The models utilize IFNs and their dual-center extension to characterize each sample by membership and

non-membership values, which are interpretable measures. The model assigns higher weights to samples closer to their class center, down-weighting those with high non-membership, such as outliers or noisy points. This geometric and probabilistic reasoning not only improves robustness but also ensures transparency, as the contribution of each training sample is based on interpretable spatial criteria.

- [2] Class probability-based membership functions (CPGB and CPBS): In CGFTSVM-ID and BSFLSTSVM-ID, the CPGB and CPBS functions integrate distance, class probability, and imbalance ratio into the weighting process. These smooth, tunable functions provide an intuitive mechanism for emphasizing representative or minority samples and de-emphasizing outliers, all based on mathematically explainable and visualizable rules. This transparent scoring process significantly contributes to interpretability.
- [3] Handling class imbalance with transparent reweighting: The proposed models explicitly incorporate class-wise scaling in their weighting functions. This ensures that minority class samples receive proportionally higher weights, thus addressing imbalance in a rational and explainable manner. The incorporation of imbalance ratio scaling makes the classifier's decisions fairer and interpretable.
- [4] Simple, transparent decision rules: The decision functions used in all models, based on signed distances from hyperplanes or margin-based criteria, are inherently interpretable. These structures enable clear tracing of how decisions are made, enhancing user trust and understanding.

4.6 Intuitive explanation of key mathematical notations

To further enhance clarity, we provide the following intuitive interpretations of frequently used mathematical symbols in Chapter 4:

- [1] $K(x_i, x_j)$ (Kernel function): This function measures the similarity between data points x_i and x_j , often using a Gaussian kernel. It enables the transformation of non-linearly separable data into a higher-dimensional space, facilitating effective linear classification.
- [2] ρ_j (Slack variable): Represents the degree to which a training sample violates the margin condition. A value of zero indicates correct classification, while a positive value reflects a misclassification or a marginal violation, allowing the model to tolerate noisy data.
- [3] $s(x_j)$ or s_j (Intuitionistic fuzzy score): Denotes the intuitionistic fuzzy score for a sample x_j , reflecting its reliability. Samples near their class center receive higher scores, whereas ambiguous or noisy samples are down-weighted. This score guides the model's attention to trustworthy data.
- [4] κ_j (Final sample weight): Combines the fuzzy score $s(x_j)$ with a class imbalance scaling factor, ensuring that minority class samples are adequately emphasized. It captures both the individual reliability and the class distribution structure, balancing fairness and robustness.

4.7 Summary

This chapter presents a series of advanced hyperplane-based classifiers designed to address the challenges of noise, outliers, and class imbalance in classification problems.

By integrating fuzzy weighting techniques and novel membership functions, these models demonstrate remarkable performance across diverse datasets, including practical applications in medical diagnosis. The discussion begins with the intuitionistic fuzzy universum support vector machine (IFUSVM) [83], which employs intuitionistic fuzzy numbers (IFNs) [49] to reduce the influence of noise and outliers. The incorporation of universum data [25] further enhances the model by leveraging prior knowledge about data distribution, thereby improving generalization. Experimental evaluations on various datasets reveal that IFUSVM consistently achieves superior accuracy and the highest ranks compared to baseline models.

To address the computational and class imbalance limitations of IFUSVM, the dual center-based intuitionistic fuzzy support vector machine (DC-IFSVM) [114] and its least square variant, dual center-based intuitionistic fuzzy least square twin support vector machine (DC-IFLSTSVM), are introduced. These models utilize a dual-center weighting function that incorporates distances to the opposing class and remains robust to threshold variations. By including class-specific terms, they effectively handle imbalanced datasets. Performance evaluations on UCI and KEEL datasets establish DC-IFLSTSVM as the most effective among the proposed models and baseline methods, with additional experiments on the BreakHis breast cancer dataset demonstrating its strong practical applicability.

Building on these advancements, the class probability and generalized bell fuzzy twin support vector machine for imbalanced data (CGFTSVM-ID) [84] introduces the CPGB weighting function. This weighting function integrates generalized bell functions, class probabilities, and imbalance ratios to assign adaptive weights, minimizing the impact of outliers while emphasizing minority-class samples. Results from experiments on UCI, KEEL, and ADNI datasets underscore its robust performance, particularly in diagnosing Alzheimer's disease (AD). Notably, CGFTSVM-ID excels in distinguishing mild cognitive

impairment (MCI) from AD, a challenging classification task in the literature.

Finally, the chapter presents the bell-shaped fuzzy least square twin support vector machine for imbalanced data (BSFLSTSVM-ID) [85], which employs the novel CPBS membership function. Unlike CPGB, CPBS integrates a bell-shaped function with adjustable scale parameters, allowing finer control over weight assignment based on data distribution. This adaptability enhances the identification of borderline samples and outliers within the majority class, while assigning the highest weights to minority-class samples to ensure their significance during training. The BSFLSTSVM-ID model addresses optimization challenges using the conjugate gradient method and demonstrates outstanding performance across UCI, BreakHis, and ADNI datasets.

In summary, this chapter underscores the transformative impact of fuzzy weighting techniques and novel membership functions in advancing hyperplane-based classifiers. By addressing the challenges of noise, outliers, and class imbalance, these models offer robust and practical solutions for real-world classification problems.

The advancements discussed in this chapter highlight the importance of tailored solutions to address specific challenges like noise, outliers, and class imbalance in classification problems. However, as data continues to grow in scale and complexity, new challenges emerge that demand scalable and efficient models without compromising accuracy. This necessitates the exploration of innovative approaches that ensure computational feasibility for large-scale datasets. The next chapter delves into this critical aspect, aiming to bridge the gap between performance and scalability in ML.

Chapter 5

Breaking Complexity Barriers in Large-Scale Classification

In an era of rapidly expanding data across diverse domains, the demand for scalable ML algorithms has become increasingly critical [120]. Traditional classification models [3] often struggle with the complexities of large-scale datasets, facing challenges such as high computational overhead, memory limitations, and inefficiencies in processing massive amounts of data. To tackle these challenges, innovative methods are needed to achieve scalability and robust performance without sacrificing accuracy.

This chapter addresses these barriers by introducing a novel approach designed specifically for large-scale data classification. Central to this methodology is the novel proximity and distance scoring ensemble for universal representation (ProDiSE), a powerful pruning mechanism that identifies a subset of data points capable of effectively representing the entire dataset. By minimizing redundancy while preserving essential information, ProDiSE enables efficient data point selection and significantly reduces computational complexity.

Building on this foundation, the chapter presents the ProDiSE-LSSVM model, which seamlessly integrates ProDiSE with LSSVM. This combined framework delivers a scalable and efficient solution for datasets containing millions of samples. ProDiSE-LSSVM miti-

gates the inefficiencies of traditional models, achieving faster training times and improved performance without compromising classification accuracy.

This chapter underscores the importance of innovative data selection and processing strategies in advancing scalable ML models, paving the way for effective solutions in big data classification.

5.1 Proposed work

In this section, we introduce a novel method for selecting data points that universally represent the entire training set. This technique, known as ProDiSE, ensures a comprehensive and diverse representation of the data. We then seamlessly integrate this approach into the LSSVM framework, resulting in the proposed ProDiSE-LSSVM model. This combination enhances the sparsity of the model while improving computational efficiency and robustness, especially for large-scale datasets.

5.1.1 Proximity and distance scoring ensemble for universal representation (ProDiSE)

Universally representative data point sampling involves selecting data points that effectively capture the full diversity, characteristics, and distribution of the entire dataset, resulting in a comprehensive and reliable representation. It ensures that the selected points reflect the overall population in a balanced manner. As a result, the chosen samples contribute meaningfully to the model's learning process, minimizing the risk of overfitting to local patterns and enhancing the model's capacity to generalize across diverse and complex data distributions.

To select the universally representative data points, we begin as follows:

Nearest neighbour proximity ratio (NNPR): We start by selecting k samples from the

dataset, forming a subset T . Around each data point $t_i \in T$, a hypersphere is defined with radius $\mathcal{R}_{t_i}$, representing the distance between t_i and its nearest neighbour within T . Let η_{t_i} denotes the nearest neighbour, with its radius represented by $\mathcal{R}_{\eta_{t_i}}$.

The nearest neighbour proximity ratio (NNPR) for each t_i is defined as the ratio of its distance to its nearest neighbour in T to the distance of that nearest neighbour to its own nearest neighbour. In other words, it is the ratio of its radius to the ratio of radius of η_{t_i} :

$$\text{NNPR}(t_i) = \frac{\mathcal{R}_{t_i}}{\mathcal{R}_{\eta_{t_i}}} \quad \forall i = 1, 2, \dots, k. \quad (5.1)$$

This ratio quantifies the spatial relation between each point and its nearest neighbour in T , as illustrated in Figure 5.1 for points t_i , t_j , and t_k . A higher NNPR indicates a point that is relatively isolated within the subset T .

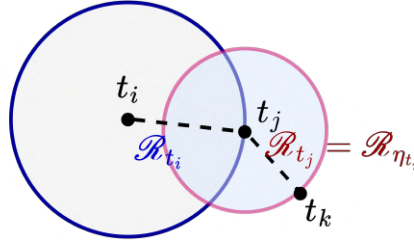


Figure 5.1: Illustration of radius of data points $t_i, t_j, t_k \in T$.

Proximity-based scoring (PBS): The PBS assigns scores to the training data points based on their relationship with points in set T through NNPR. The process involves identifying the closest neighbour for each of the l training samples among the k selected data points in T . For each training point x_j , a hypersphere is drawn with radius $\mathcal{R}_{x_j}$, representing the distance to its nearest neighbour t_i in T . The PBS assigned to x_j depends on comparing the radius of x_j with that of its nearest neighbour t_i in T . If the radius of x_j is greater than or equal to that of t_i , PBS of x_j is set to some large value. Conversely, if the radius of x_j is smaller than that of t_i , the PBS for x_j is equal to $\text{NNPR}(t_i)$, suggesting x_j represents a

distinct region in the dataset.

In other words, to assess the representativeness of each training point x_j , PBS compares its radius to the radius of its nearest neighbour in T . If x_j is closer to its neighbour than the neighbour is to its own nearest neighbour, x_j is considered well-represented and assigned a PBS based on the NNPR of x_j 's nearest neighbour. Otherwise, x_j is assigned a high PBS, indicating it's in a less-represented region. Mathematically, it can be defined as:

$$\text{PBS}(x_j) = \begin{cases} M, & \text{if } \mathcal{R}_{\eta_{x_j}} \leq \mathcal{R}_{x_j}, \\ \text{NNPR}(t_i), & \text{if } \mathcal{R}_{\eta_{x_j}} > \mathcal{R}_{x_j}, \end{cases} \quad (5.2)$$

where M represents a large positive real number and $\eta_{x_j} = t_j$. Figure 5.2 illustrates the training sample x_j (pink color) and its nearest neighbour sample t_i from subset T (black color), considering two different cases of computing PBS. Fig. 5.2a presents the scenario when $\mathcal{R}_{t_i} \leq \mathcal{R}_{x_j}$, resulting in $\text{PBS}(x_j) = M$. Conversely, Fig. 5.2b illustrates the scenario when $\mathcal{R}_{t_i} > \mathcal{R}_{x_j}$, leading to $\text{PBS}(x_j) = \text{NNPR}(t_i)$.

Distance-based scoring (DBS): DBS assigns a score to each training sample by measur-

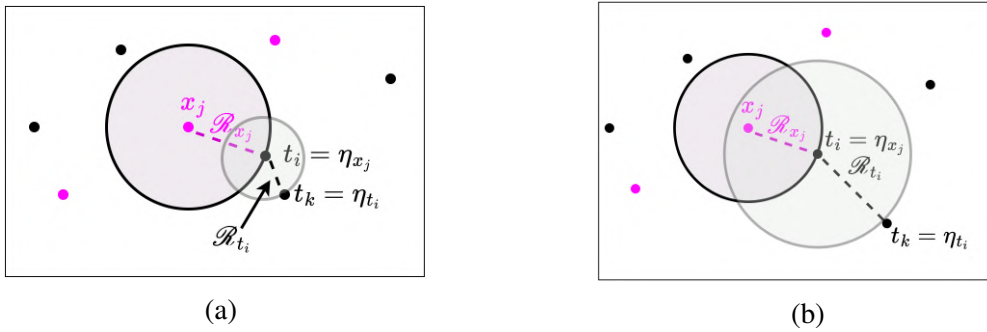


Figure 5.2: Representation of PBS of a training sample x_j based on equation (5.2). Pink data points represent the training samples, and black points represent samples of subset T . Sample x_j has radius $\mathcal{R}_{x_j}$. The radius of the nearest neighbour of x_j in T (denoted as $t_i \in T$) is given by $\mathcal{R}_{t_i}$. Fig. (a) illustrates $\text{PBS}(x_j) = M$ as $\mathcal{R}_{t_i} \leq \mathcal{R}_{x_j}$. Fig. (b) illustrates $\text{PBS}(x_j) = \text{NNPR}(t_i)$, where $\mathcal{R}_{t_i} > \mathcal{R}_{x_j}$.

ing its cumulative proximity to a subset of selected points, denoted by set T , which has k

representative samples. The goal of DBS is to quantify how closely each training sample, x_j , aligns with the diverse points in T by summing distances to each point in T .

For each training sample x_j , the distance to each point t_i in T is calculated as:

$$\mathcal{D}_{ji} = \|x_j - t_i\|, \quad j = 1, 2, \dots, l \text{ and } i = 1, 2, \dots, k.$$

This distance $\mathcal{D}_{ji}$ represents the Euclidean norm, giving a scalar that captures the closeness of x_j to each t_i .

The DBS for each training sample x_j is then computed by summing these distances over all k points in T , providing a cumulative measure of proximity. Mathematically, it is defined as:

$$\text{DBS}(x_j) = \sum_{i=1}^k \mathcal{D}_{ji} \text{ for } j = 1, 2, \dots, l. \quad (5.3)$$

- **Interpretation of PBS:** PBS focuses on the local density of data points, with a lower PBS indicating that a data point is situated within a dense cluster of similar points, making it highly representative of the local data distribution. Conversely, a higher PBS suggests that the point is relatively isolated or in a sparse region, potentially indicating an outlier or a data point from a less-represented area.
- **Interpretation of DBS:** DBS considers the global distribution of data points in relation to the subset T . A lower DBS implies that a data point is closer to the selected representative points, aligning well with the overall data distribution. Such points are more representative of the global data structure. Conversely, a higher DBS suggests that the data point is farther away from the representative points, potentially indicating an outlier or a data point that is less representative of the global data distribution.

5.2 Intuitive purpose of NNPR, PBS and DBS

NNPR quantifies the local isolation of a point within a randomly selected subset T . A high NNPR suggests that the point is not tightly clustered with others, implying sparsity in that region, while a low NNPR indicates local density. Thus, NNPR helps identify points that are critical for representing under-sampled or diverse regions of the input space.

PBS uses NNPR to evaluate the representativeness of each training sample relative to its nearest neighbor in T . If a training point is more isolated than its neighbor, a large constant score is assigned, de-emphasizing its selection. If it lies within a dense neighborhood, it is given a score based on that neighbor's NNPR, identifying it as part of a unique region needing better representation. This effectively prioritizes diversity in local structures.

DBS measures how globally close a sample is to all points in T . A lower DBS indicates the point is close to many representative samples, thus highly redundant, while a high DBS signals global isolation and potential value for inclusion. Therefore, DBS captures the global alignment of a point with the selected subset.

PBS and DBS together offer a complementary view of data representativeness: low PBS highlight points in dense local clusters, while low DBS indicate proximity to subset T . Points with both low PBS and DBS are thus identified as highly representative, capturing both local density and alignment with the dataset's overall structure.

To ensure a robust selection of universally representative data points, the subset T is repeatedly formed r times by randomly sampling k points from the l available training data points. This iterative formation of T helps in capturing a broader and more stable representation across different sample sets. For each of the r randomly selected subsets T , the computations for NNPR, PBS, and DBS are carried out individually, resulting in an $l \times r$ matrix for both PBS and DBS values. By averaging across each row of these matrices, a

final column vector of size l is derived, which represents the consolidated PBS and DBS for all training data points. This approach mitigates the risk of variability due to any particular random subset T , thereby providing a more reliable measure of representativeness across the entire dataset.

5.3 RepScore

To ensure a fair comparison between the scores for PBS and DBS, which have different units, min-max normalization is applied. This normalization scales the values into a common range, typically $[0, 1]$, making them comparable without altering the underlying relationships. Let the PBS value for the data point x_j be denoted as PBS_j and the DBS value as DBS_j . The min-max normalization for these values is applied as follows:

$$\bar{\text{PBS}}_j = \frac{\text{PBS}_j - \text{PBS}_{\min}}{\text{PBS}_{\max} - \text{PBS}_{\min}}, \quad (5.4)$$

$$\bar{\text{DBS}}_j = \frac{\text{DBS}_j - \text{DBS}_{\min}}{\text{DBS}_{\max} - \text{DBS}_{\min}}, \quad (5.5)$$

where $\text{DBS}_{\min}$ and $\text{DBS}_{\max}$ denote the minimum and maximum values of vector DBS, respectively. Similarly, $\text{PBS}_{\min}$ and $\text{PBS}_{\max}$ represent the minimum and maximum values of vector PBS, respectively. For simplicity, we continue to use PBS_j to denote $\bar{\text{PBS}}_j$ and DBS_j to denote $\bar{\text{DBS}}_j$.

RepScore: To universally represent the training data points, we define a representation score, abbreviated as RepScore, s_j for each training data point x_j . The value of s_j is determined by the following expression:

$$s_j = \max\{\text{DBS}_j, \text{PBS}_j\}, \quad (5.6)$$

where $j = 1, 2, \dots, l$. The RepScore quantifies the degree of universal representation of

each data point. Specifically, a smaller value of s_j indicates a data point that is more universally representative within the dataset. Let P denotes the set of most representative data points which are the top-ranked data points based on the sorted s_j values in ascending order.

The allocation of points in set P from both classes in training set, is made in proportion to the class distribution of the training data. Let l_+ and l_- represent the number of data points in the positive and negative classes, respectively. The number of data points chosen from each class for subset P , denoted by m_+ and m_- , is calculated using the following formulas:

$$m_+ = \max\{1, \text{round}(m \times l_+/l)\}, \quad (5.7)$$

$$m_- = m - m_+, \quad (5.8)$$

where m is the total number of data points selected for the subset P . The steps of the ProDiSE algorithm are outlined in Algorithm [5.1](#).

Principles of the ProDiSE selection approach: Universally representative nearest neighbour ensemble reflects the methodology’s focus on capturing a comprehensive, unbiased subset of data points by leveraging multiple samples. ProDiSE builds a globally representative collection of points by iteratively constructing subsets of nearest neighbours. This ensemble approach thus ensures a robust, consistent subset that reflects the overall data structure, justifying both the “representative” and “ensemble” elements in its name.

Connection to classical ensemble learning: ProDiSE draws philosophical inspiration from ensemble learning, particularly in how it constructs multiple random subsets T and aggregates information from them to form a robust selection. Like ensemble models that combine weak learners to enhance predictive performance, ProDiSE repeatedly selects different dataset and averages representativeness scores across them. This approach reduces

sampling bias, stabilizes the representation process, and ensures diversity in the final subset, a core principle in ensemble learning. The “ensemble” in ProDiSE does not refer to combining multiple classifiers, but to the ensemble of perspectives obtained by examining how each data point behaves across different subsets of the training data. By reframing NNPR, PBS, and DBS as local and global representativeness indicators, and linking their computation across multiple subsets to ensemble-style data modeling, the intuitive rationale behind ProDiSE becomes better. This not only enriches interpretability but also showcases its novelty as a non-iterative, ensemble-inspired data pruning technique that balances diversity and representativeness for scalable learning.

5.3.1 Proposed ProDiSE-LSSVM

By seamlessly incorporating ProDiSE technique with LSSVM, we propose a sparse variant of LSSVM which is non-iterative and prunes the data points based on their characteristics in the feature space and call it as ProDiSE-LSSVM. $P \subset X$ represents the set of universally represented data points based on the nearest neighbour ensemble such that $|P| = m$. The optimization problem for the proposed ProDiSE-LSSVM by considering data points in subset P is defined as:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2} \|u^2\| + \frac{\lambda}{2} \sum_{j=1}^m \xi_j^2 \\ \text{subject to} \quad & u^t \psi(x_j) + b = y_j - \xi_j, \forall j = 1, 2, \dots, m, \end{aligned} \quad (5.9)$$

where ξ_j represents the slack variable. u and b represent the weight vector and bias for the hyperplane. The proposed ProDiSE-LSSVM leverages the ProDiSE technique to select a more representative subset of data points for training, thus enhancing efficiency by focusing on a reduced, yet informative, set of points. The optimization framework, when restricted

to the subset P , ensures that the model maintains its predictive accuracy while achieving a more compact and sparse representation of the data. By reducing the number of data points involved in the training process, the proposed ProDiSE-LSSVM improves computational efficiency, making it a suitable choice for large-scale datasets while retaining robustness against noise and outliers.

By employing the representer theorem [161, 162, 163], we have

$$u = \sum_{j=1}^m \alpha_j \psi(x_j). \quad (5.10)$$

Substituting equation (5.10) in problem (5.9) and eliminating ξ_j with the equality constraints, we get

$$\mathcal{F} = \frac{1}{2} \sum_{j=1}^m \sum_{k=1}^m \alpha_j K(x_j, x_k) \alpha_k + \frac{\lambda}{2} \sum_{j=1}^m (y_j - \sum_{k=1}^m \alpha_k K(x_k, x_j) - b)^2. \quad (5.11)$$

On solving and rearranging the terms, we get

$$\mathcal{F} = \frac{1}{2} \alpha^t (K + \lambda K K^t) \alpha + \lambda \alpha^t K (e b - Y) + \lambda b e^t Y + \frac{\lambda}{2} N b^2. \quad (5.12)$$

To solve for α and b , differentiate the equation (5.12) with respect to α and b which leads to the following equations:

$$\frac{\partial \mathcal{F}}{\partial \alpha} = (K + \lambda K K^t) \alpha + \lambda (e b - Y)^t K = 0, \quad (5.13)$$

$$\frac{\partial \mathcal{F}}{\partial b} = \lambda \alpha^t K e - \lambda e^t Y + \lambda N b = 0. \quad (5.14)$$

Rearranging the equations (5.13) and (5.14), we obtain:

$$\begin{bmatrix} \frac{1}{\lambda} K + K K^t & e^t K \\ e^t K^t & N \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} Y^t K \\ e^t Y \end{bmatrix}. \quad (5.15)$$

The equation (5.15) involves inversion of $(m + 1) \times (m + 1)$ order matrix. Upon obtaining the unknowns α and b , the prediction of the label for a new input data x is accomplished through the following function:

$$F(x) = \text{sign} \left(\sum_{k=1}^m \alpha_k K(x_k, x) + b \right). \quad (5.16)$$

Algorithm 5.2 outlines the procedural steps involved in the proposed ProDiSE-LSSVM model. Further Figure 5.3 represents the flow of ProDiSE-LSSVM.

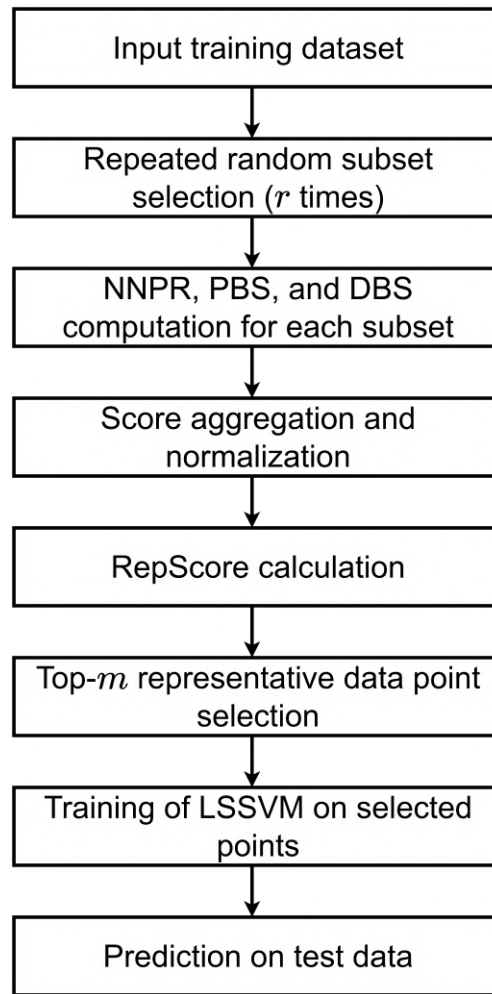


Figure 5.3: Flow diagram of ProDiSE-LSSVM.

5.4 Complexity

The time and space complexity of the proposed ProDiSE methodology and the proposed ProDiSE-LSSVM model can be explained as follows:

Time complexity: For the proposed ProDiSE process, the time complexity is derived from three primary steps. First, calculating NNPR for the k data points in the subset T has a complexity of $O(k^2)$. Second, computing the PBS requires determining the nearest neighbour of each of the l training samples among the k data points of set T , which incurs a computational cost of $O(lk)$. Third, the DBS calculation also involves finding the distance of l training samples from the k data points in T , resulting in another $O(lk)$ cost. Therefore, the total cost for identifying the universally representative data points becomes $O(lk) + O(lk) + O(k^2)$, which simplifies to $O(lk)$, for $k \ll l$. The ProDiSE process is executed r times, resulting in a total time complexity of $O(rlk)$. Given that r is a constant, the overall time complexity of the ProDiSE process simplifies to $O(lk)$, which is linear with respect to the number of training samples.

The computational complexity of proposed ProDiSE-LSSVM involves ProDiSE technique and solving optimization problem. The ProDiSE selection incurs a cost of $O(lk)$. Further, solving the ProDiSE-LSSVM optimization problem requires calculating the inverse of the matrix in equation (5.15), which has a complexity of $O(m^{1.95})$ [164, 165], approximately $O(m^2)$. Therefore, the total complexity of ProDiSE-LSSVM is $O(m^2) + O(lk)$. Since $m \ll l$, the overall complexity is simplified to $O(lk)$, giving ProDiSE-LSSVM linear time complexity for large datasets.

Space complexity: For the ProDiSE process, the space complexity is $O(rl)$, which simplifies to $O(l)$ when $r \ll l$. This accounts for the storage required for intermediate computations during the selection of universally representative data points. On the other hand, the

space complexity of ProDiSE-LSSVM is $O(m^2)$ due to the need to store the matrix associated with the system of linear equations. Combining these, the total space complexity of the proposed ProDiSE-LSSVM is $O(m^2) + O(l)$, and since $m \ll l$, the overall space complexity is dominated by $O(l)$. As a result, ProDiSE-LSSVM exhibits linear space complexity, making it efficient and suitable for large-scale datasets.

5.5 Numerical experiments

This section outlines the numerical experiments performed for the proposed ProDiSE-LSSVM and the baseline models. The considered baseline models include, sparse LSSVM [41], FSA-LSSVM [34], PCP-LSSVM [166], RoBoSS [167], and GRS-LSSVM [42]. We conduct a comprehensive analysis of the experimental results to assess the performance of these models.

5.5.1 Experimental setup

All the experiments are conducted on a Desktop PC equipped with 2023a MATLAB, processor Intel(R) Xeon(R) Platinum 8260 CPU, and 256GB RAM. For experiments, we used real-world UCI datasets [126] and synthetic NDC datasets [168]. The real-world datasets are organized into two categories based on size: Category A, with 28 datasets containing up to 19,020 samples; Category B with 5 datasets containing over 20,000 samples, up to a maximum of 581,012 samples. Each dataset was partitioned into 70% for training and 30% for testing, and the RBF kernel ($K(x_j, x_k) = \|x_j - x_k\|^2 / 2\sigma^2$) is applied in non-linear model evaluations. The baseline models and the proposed ProDiSE-LSSVM model involve key parameters: the regularization hyperparameter λ and the kernel function parameter σ . For datasets in category A, we performed five-fold cross-validation for parameter selection. Specifically, λ is selected from the range $[10^{-2}, 10^{-1}, 10^0, 10^1, 10^2]$, while

Algorithm 5.1 Proposed ProDiSE

Input: Training data set, number of reserved SVs m .

Output: Universally represented data points.

- [1] For $k = 1$ to r do:
 - [2] Randomly choose k training samples to form subset T .
 - [3] for $i = 1$ to k do:
 - [4] Compute $\text{NNPR}(t_i)$ using equation (5.1).
 - [5] end For
 - [6] for $j = 1$ to l do
 - [7] Compute $\text{PBS}(x_j)$ using equation (5.2).
 - [8] Compute $\text{DBS}(x_j)$ using equation (5.3).
 - [9] end For
 - [10] Store vectors PBS and DBS as the column of matrix $\tilde{P}$ and $\tilde{D}$,
 respectively, which results in $l \times r$ matrix.
 - [11] end For
 - [12] Compute mean along each row, resulting in column vectors PBS and DBS.
 - [13] Normalize vectors PBS and DBS using min-max normalization based on equation
 (5.4) and (5.5).
 - [14] for $j = 1$ to l do
 - [15] Compute indicator variable s_j using equation (5.6).
 - [16] end For
 - [17] Sort s in descending order.
 - [18] Choose first m data points from vector s .
-

Algorithm 5.2 Proposed ProDiSE-LSSVM

Input: Training data point $(x_1, y_1), (x_2, y_2), \dots, (x_l, y_l)$, testing data, hyperparameters λ , number of reserved SVs m .

Output: Universally represented data points, labels of testing data set.

- [1] Determine m_+ and m_- from equations (5.7) and (5.8).
 - [2] Apply ProDiSE to obtain m_+ positive and m_- negative data points, forming the subset P .
 - [3] Compute the kernel matrix using the subset P and solve the system of linear equation (5.15).
 - [4] Determine the label of the test data point using equation (5.16).
-

σ is chosen from $[1, 10, 100, 200]$. Further, the ProDiSE-LSSVM model introduces additional parameters: k , representing the size of $T \subset X$, and r , denoting the number of subset formations for enhanced robustness in selecting universally representative data points. The parameter k is varied over the range $[2, 2^2, \dots, 2^7]$, while r was held constant at 100 for all datasets. For RoBoSS, the ranges for loss function hyperparameters c and d were set to $[2:0.1:5]$ and $[0.1:0.1:2]$, respectively. For the value of M in equation (2), which is part of the PBS definition, any sufficiently large value is appropriate. In our experiments, we set M as greater than the maximum element of the DBS vector. After normalization, this choice ensures that M is the largest value in the dataset. In category B datasets, to reduce tuning time, we fixed parameter values to the first in each range ($\lambda = 10^{-2}$, $\sigma = 1$, and $k = 2$). The metrics considered for model performance include training time (in seconds), testing time (in seconds), and accuracy. Testing time signifies the sparsity of the model and accuracy represents the percentage of correctly classified samples out of the total samples.

Each baseline model implements unique, rigorously tested stopping criteria. FSA-LSSVM caps sparsity by limiting a randomly selected subset to 2% of the total sample count, with an approximation error tolerance of 0.05 [34]. PCP-LSSVM achieves similar control by capping the kernel matrix rank at 2% of the sample size and setting an excep-

Table 5.1: Accuracy, training time (in seconds) and testing time (in seconds) of the proposed ProDiSE-LSSVM and baseline models on category A datasets.

Dataset (samples,features)	sparse LSSVM [41] Accuracy (Train time,Test time)	FSA-LSSVM [34] Accuracy (Train time,Test time)	PCP-LSSVM [166] Accuracy (Train time,Test time)	RoBoSS [167] Accuracy (Train time,Test time)	GRS-LSSVM [42] Accuracy (Train time,Test time)	Proposed ProDiSE-LSSVM Accuracy (Train time,Test time)
breast_cancer (286, 9)	73.2558 (0.0689, 0.0097)	26.7442 (0.1594, 0.0089)	74.4186 (0.1784, 0.0028)	73.2558 (0.0187, 0.001)	73.2558 (0.0647, 0.0037)	73.2558 (0.0587, 0.0056)
congressional_voting (435, 16)	56.9231 (0.1095, 0.002)	43.0769 (0.1282, 0.0065)	56.9231 (0.0681, 0.0011)	58.4615 (0.0203, 0.0029)	56.9231 (0.0502, 0.0038)	56.9231 (0.0701, 0.0019)
cylinder_bands (512, 35)	44.8052 (0.0116, 0.0034)	59.7403 (0.0401, 0.0099)	59.7403 (0.0426, 0.0016)	58.4416 (0.014, 0.0051)	59.7403 (0.0385, 0.0031)	59.7403 (0.0675, 0.0002)
haberman_survival (306, 3)	70.6522 (0.077, 0.0052)	70.6522 (0.1036, 0.0017)	70.6522 (0.1355, 0.0019)	70.6522 (0.0117, 0.0039)	70.6522 (0.0471, 0.004)	70.6522 (0.0575, 0.0085)
hepatitis (155, 19)	84.7826 (0.0057, 0.0025)	86.9565 (0.607, 0.0012)	86.9565 (0.515, 0.0009)	86.9565 (0.0188, 0.007)	86.9565 (0.1228, 0.0031)	86.9565 (0.0659, 0.0013)
horse_colic (368, 25)	63.6364 (0.0161, 0.0082)	69.0909 (0.2263, 0.0059)	66.3636 (0.3561, 0.0069)	65.4546 (0.0107, 0.0063)	66.3636 (0.1333, 0.0044)	67.2727 (0.1066, 0.0017)
ilpd_indian_liver (583, 9)	73.1429 (0.1101, 0.0023)	73.1429 (0.0393, 0.0019)	73.1429 (0.0476, 0.0012)	72 (0.0094, 0.0146)	73.1429 (0.0288, 0.004)	73.1429 (0.0628, 0.0004)
monks_3 (554, 6)	46.3855 (0.0408, 0.003)	55.4217 (0.0454, 0.0002)	55.4217 (0.0667, 0.0008)	72.8916 (0.0112, 0.0164)	55.4217 (0.0419, 0.0005)	60.241 (0.0705, 0.0004)
BCW (683, 9)	65.8537 (0.043, 0.0044)	87.3171 (0.0183, 0.004)	91.2195 (0.0429, 0.008)	89.2683 (0.0017, 0.0004)	65.8537 (0.1726, 0.0055)	92.6829 (0.1914, 0.0045)
oocytes_trisopterus_nucleus_2f (912, 25)	57.2993 (0.0998, 0.002)	59.4891 (0.0286, 0.0006)	59.4891 (0.0295, 0.0032)	58.7591 (0.0136, 0.0002)	57.2993 (0.0213, 0.0003)	59.4891 (0.0775, 0.0013)
spectf (267, 44)	76.25 (0.0146, 0.0023)	80 (0.2082, 0.0006)	75 (0.1944, 0.0027)	80 (0.0186, 0.0003)	80 (0.0521, 0.0004)	80 (0.0617, 0.001)
statlog_australian_credit (690, 14)	60.8696 (0.0975, 0.0027)	66.1836 (0.0516, 0.0005)	66.1836 (0.0388, 0.0013)	66.1836 (0.019, 0.0003)	66.1836 (0.0284, 0.0017)	66.1836 (0.0658, 0.0004)
brwiscinsin (683, 9)	65.8537 (0.0147, 0.0002)	65.8537 (0.0668, 0.0007)	91.2195 (0.164, 0.0017)	96.5854 (0.0236, 0.0003)	65.8537 (0.1081, 0.0002)	88.2927 (0.0952, 0.0011)
haber (306, 3)	70.6522 (0.0874, 0.0014)	70.6522 (0.1328, 0.0021)	70.6522 (0.0834, 0.0013)	70.6522 (0.0256, 0.0002)	70.6522 (0.7262, 0.0004)	70.6522 (0.0646, 0.0112)
haberman (306, 3)	68.4783 (0.0124, 0.0001)	36.9565 (0.2599, 0.0027)	70.6522 (0.0477, 0.0005)	70.6522 (0.0188, 0.0002)	70.6522 (0.1015, 0.0002)	70.6522 (0.0961, 0.0011)
monk1 (556, 6)	54.491 (0.045, 0.0008)	54.491 (0.0215, 0.0004)	45.509 (0.2423, 0.0043)	45.509 (0.0363, 0.0002)	45.509 (0.0675, 0.0006)	49.7006 (0.0566, 0.0012)
wdbc (194, 33)	72.4138 (0.0403, 0.0006)	72.4138 (0.2068, 0.0029)	72.4138 (0.0339, 0.0005)	60.3448 (0.0373, 0.0002)	72.4138 (0.0519, 0.0005)	72.4138 (0.0702, 0.0001)
blood (748, 4)	72.7679 (0.2364, 0.0003)	73.2143 (0.0019, 0.0008)	72.7679 (0.0026, 0.0014)	72.7679 (0.0017, 0.0004)	72.7679 (0.0385, 0.0009)	72.7679 (0.095, 0.0009)
tic_tac_toe (958, 9)	60.9756 (0.2191, 0.0007)	47.7352 (0.0025, 0.0006)	66.899 (0.0034, 0.0004)	66.899 (0.0017, 0.0005)	66.899 (0.0319, 0.0004)	66.899 (0.0773, 0.0003)
oocytes_merluccius_nucleus_4d (1022, 41)	64.8209 (0.1168, 0.0027)	67.101 (0.0218, 0.0013)	67.101 (0.0212, 0.0013)	62.8665 (0.0829, 0.0027)	67.101 (0.0111, 0.0031)	67.101 (0.0851, 0.0021)
yeast1 (2968, 8)	71.9101 (1.5847, 0.0165)	28.0899 (0.1102, 0.0011)	71.9101 (1.308, 0.0136)	72.5843 (0.1054, 0.0008)	71.9101 (0.5366, 0.0011)	71.9101 (0.0961, 0.0056)
bank (4521, 16)	88.3481 (5.4923, 0.0038)	87.3156 (0.1897, 0.0057)	88.2743 (0.1445, 0.0057)	88.3481 (0.0016, 0.0007)	88.3481 (0.8195, 0.0036)	88.3481 (0.0873, 0.0035)
banknote (1372, 4)	53.3981 (0.6258, 0.0003)	46.6019 (0.0059, 0.001)	46.6019 (0.0094, 0.0017)	89.5631 (0.0017, 0.0004)	45.3884 (0.1026, 0.0019)	62.8641 (0.1166, 0.0015)
musk_2 (6598, 166)	85.8009 (14.7686, 0.0035)	85.9525 (1.3346, 0.0114)	76.7054 (1.3453, 0.0128)	82.9207 (0.0028, 0.0027)	85.8009 (24.8822, 0.0091)	85.902 (0.5301, 0.0069)
ozone (2536, 72)	97.3719 (1.9627, 0.0003)	97.3719 (0.045, 0.0013)	97.3719 (0.0443, 0.001)	94.7438 (0.0017, 0.0007)	97.3719 (0.641, 0.0011)	97.3719 (0.0948, 0.001)
ringnorm (7400, 20)	52.973 (1.8521, 0.1103)	51.0811 (0.1754, 0.0079)	51.0811 (0.2351, 0.0093)	55.4955 (0.0018, 0.0008)	51.0811 (1.0976, 0.002)	96.26 (0.1242, 0.0013)
titanic (2201, 3)	68.1818 (1.2793, 0.0001)	31.8182 (0.0147, 0.0007)	31.8182 (0.006, 0.0003)	76.2121 (0.0017, 0.0006)	68.1818 (0.1411, 0.0005)	76.52 (0.0829, 0.0003)
magic (19020, 10)	64.2657 (82.0547, 0.0005)	29.0922 (1.2505, 0.0217)	25.0964 (1.2607, 0.0227)	66.2636 (0.0018, 0.0045)	64.2657 (7.2478, 0.0018)	64.2657 (0.1395, 0.0017)
Average accuracy	67.38	61.56	67.2	72.73	68.51	73.16
Average rank	4.18	3.77	3.63	3.25	3.52	2.66

tionally tight error tolerance of 10^{-10} [166]. For GRS-LSSVM, the number of SVs is fixed at 20, while θ is calibrated to 0.4, as recommended in [42]. To ensure a fair and direct comparison, the ProDiSE-LSSVM is also configured with 20 SVs.

Table 5.2: Optimal parameters corresponding to maximum accuracy for baseline and proposed ProDiSE-LSSVM across category A datasets.

Dataset	sparse LSSVM [41] (λ, σ)	FSA-LSSVM [34] (λ, σ)	PCP-LSSVM [166] (λ, σ)	RoBoSS [167] (λ, σ, a, b)	GRS-LSSVM [42] (λ, σ)	Proposed ProDiSE-LSSVM (λ, σ, k)
breast_cancer	(0.01, 1)	(10, 1)	(1, 1)	(0.1, 1, 2.3, 0.3)	(0.01, 1)	(0.01, 1, 2)
congressional_voting	(0.01, 1)	(100, 200)	(10, 1)	(0.01, 1, 2, 1.7)	(0.01, 1)	(0.01, 1, 2)
cylinder_bands	(1, 1)	(0.01, 10)	(0.01, 10)	(100, 1, 4.2, 0.3)	(0.01, 1)	(0.01, 1, 2)
haberman_survival	(0.01, 1)	(100, 1)	(10, 1)	(10, 1, 2.4, 0.2)	(0.01, 1)	(0.01, 1, 2)
hepatitis	(0.01, 10)	(0.01, 10)	(0.1, 10)	(0.1, 1, 2, 1.5)	(0.01, 1)	(0.01, 1, 2)
horse_colic	(10, 1)	(1, 1)	(0.01, 1)	(1, 1, 3.5, 1.6)	(0.01, 1)	(10, 1, 8)
ilpd_indian_liver	(0.01, 1)	(100, 100)	(0.01, 1)	(0.1, 1, 3.5, 0.1)	(0.01, 1)	(0.01, 1, 2)
monks_3	(0.1, 1)	(10, 1)	(0.01, 1)	(1, 1, 3.5, 0.3)	(0.01, 10)	(10, 1, 16)
BCW	(0.1, 1)	(0.01, 1)	(10, 1)	(0.01, 1, 3.8, 2)	(0.0001, 1)	(100, 1, 4)
oocytes_trisopterus_nucleus_2f	(100, 1)	(0.01, 1)	(10, 1)	(10, 1, 4, 1.4)	(0.01, 1)	(0.01, 1, 2)
spectf	(0.01, 10)	(0.01, 10)	(0.01, 10)	(10, 1, 3.4, 1.1)	(0.01, 1)	(0.01, 1, 2)
statlog_australian_credit	(100, 1)	(0.01, 10)	(0.01, 1)	(100, 100, 2.3, 1.2)	(0.01, 1)	(0.01, 1, 2)
brwiscosin	(0.1, 1)	(1, 1)	(10, 1)	(0.01, 1, 4, 1.8)	(0.01, 1)	(1, 1, 2)
haber	(0.01, 1)	(100, 1)	(10, 1)	(10, 1, 2.4, 0.2)	(0.01, 1)	(0.01, 1, 2)
haberman	(0.1, 1)	(0.1, 1)	(0.1, 1)	(0.1, 1, 4.4, 0.6)	(0.01, 1)	(100, 1, 4)
monk1	(1, 1)	(100, 100)	(1, 10)	(0.01, 200, 4.4, 0.2)	(10, 100)	(0.1, 1, 2)
wpbc	(0.01, 1)	(0.01, 100)	(0.01, 1)	(0.1, 1, 2.1, 1.7)	(0.01, 1)	(0.01, 1, 2)
blood	(0.01, 1)	(10, 10)	(100, 1)	(1, 1, 4.5, 0.3)	(0.0001, 1)	(0.01, 1, 2)
tic_tac_toe	(100, 1)	(0.1, 1)	(0.01, 1)	(1, 1, 2.2, 1.2)	(0.0001, 1)	(0.01, 1, 2)
oocytes_merluccius_nucleus_4d	(10, 1)	(0.01, 10)	(0.1, 1)	(0.01, 1, 3.7, 0.5)	(1, 10)	(0.01, 1, 2)
yeast1	(0.01, 1)	(100, 1)	(100, 1)	(1, 1, 4.6, 0.3)	(0.01, 1)	(0.01, 1, 2)
bank	(0.01, 1)	(10, 1)	(0.01, 1)	(0.01, 10, 2, 0.1)	(0.0001, 1)	(0.01, 1, 2)
banknote	(1, 1)	(100, 100)	(0.01, 1)	(0.1, 1, 2.4, 0.6)	(0.001, 1)	(0.01, 1, 128)
musk_2	(0.01, 1)	(1, 1)	(100, 10)	(10, 1, 3.3, 0.3)	(0.0001, 1)	(100, 1, 4)
ozone	(0.01, 1)	(10, 1)	(0.01, 1)	(0.1, 1, 4.9, 1.1)	(0.0001, 1)	(10, 1, 64)
ringnorm	(100, 1)	(10, 1)	(1, 200)	(1, 1, 2.8, 0.7)	(0.0001, 1)	(0.01, 1, 2)
titanic	(0.01, 10)	(100, 1)	(0.01, 100)	(10, 1, 4.6, 0.5)	(0.0001, 1)	(0.01, 1, 2)
magic	(0.01, 10)	(100, 1)	(100, 1)	(0.1, 1, 4.7, 1.7)	(0.0001, 1)	(0.1, 1, 16)

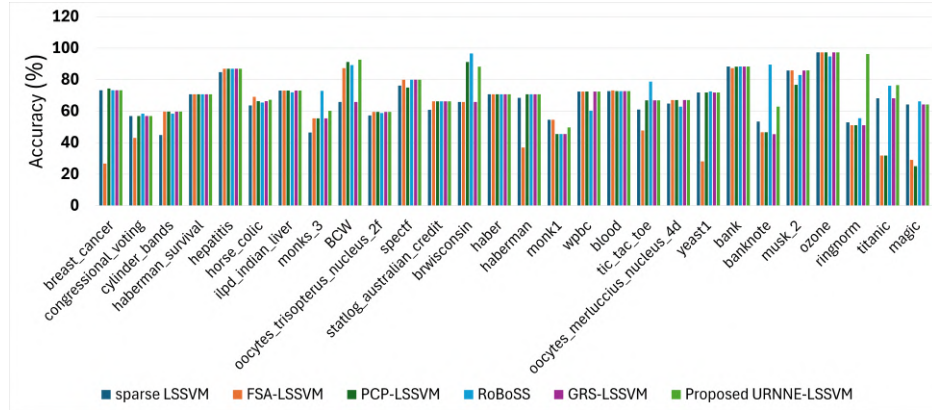


Figure 5.4: Demonstration of performance on category A datasets.

5.5.2 Experimental results on category A datasets

Table 5.1 presents a detailed comparison of the proposed ProDiSE-LSSVM with baseline models, including sparse LSSVM, FSA-LSSVM, PCP-LSSVM, RoBoSS, and GRS-LSSVM. The proposed ProDiSE-LSSVM achieves an average accuracy of 73.16% across 28 datasets. In comparison, the baseline models exhibit the average accuracy of 72.73% for RoBoSS, 68.51% for GRS-LSSVM, 67.38% for sparse LSSVM, 67.2% for PCP-LSSVM

and 61.56% for FSA-LSSVM. These results demonstrate that the proposed ProDiSE-LSSVM outperforms all baseline models in terms of average accuracy. Further, to avoid potential biases where higher accuracy on specific datasets might obscure poorer performance on others, we calculate the ranks of each model. The rank is assigned so that the best-performing model receives the lowest rank and vice-versa. The proposed ProDiSE-LSSVM achieves an average rank of 2.66. In comparison, the baseline models have the following average ranks in increasing order: RoBoSS with an average rank of 3.25, GRS-LSSVM with 3.52, PCP-LSSVM with 3.63, FSA-LSSVM with 3.77, and sparse LSSVM with 4.18. These results further validate the superior performance of the proposed ProDiSE-LSSVM, as it achieves the highest accuracy and lowest average rank. Thus, the proposed ProDiSE-LSSVM demonstrates enhanced effectiveness and reliability across diverse datasets, making it a promising model for classification tasks. Figure 5.4 demonstrates the performance comparison between the proposed ProDiSE-LSSVM and various baseline models on category A datasets.

The optimal parameters for both the proposed ProDiSE-LSSVM and the baseline models are detailed in Table 5.2, providing a clear comparison of their configurations.

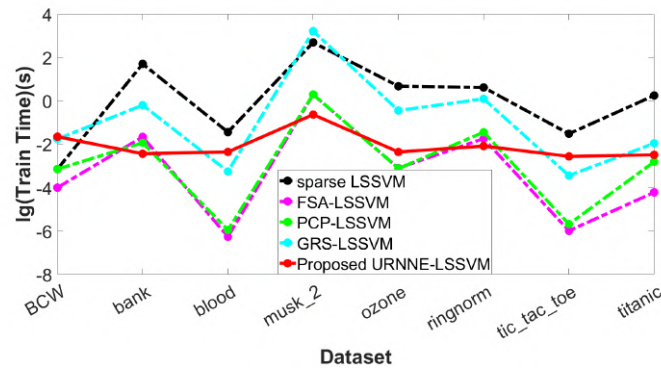


Figure 5.5: Comparison of training time (in seconds) for different datasets.

5.5.3 Comparison of training time of the proposed ProDiSE-LSSVM and the baseline models

For a more comprehensive comparison of training times across various datasets and models, Figure 5.5 presents a graph that plots the logarithmic values of training time against different datasets. The results indicate that for datasets such as bank, musk_2, and ring-norm, the training time of the proposed ProDiSE-LSSVM is notably lower than that of the baseline algorithms, showcasing its efficiency. However, for other datasets, the training time of models like PCP-LSSVM and FSA-LSSVM is comparatively lower than that of the proposed ProDiSE-LSSVM. This observation is consistent with our analysis of time complexity, which suggests that the ProDiSE-LSSVM model incurs a time complexity of $O(rlk)$. If the dataset size l is equivalent to k , the complexity becomes $O(r l^2)$, with r being constant. Therefore, the proposed model is especially effective for datasets with larger training sizes, where its time efficiency significantly outperforms other models. This makes ProDiSE-LSSVM an ideal choice for large-scale applications, where both training time and accuracy are critical for performance.

5.5.4 Sensitivity analysis of the proposed ProDiSE-LSSVM across various datasets

Figure 5.6 presents the sensitivity analysis of the proposed ProDiSE-LSSVM model across various hyperparameters. Specifically, Figure 5.6a and Figure 5.6b illustrate the performance variations on the bank and musk_2 datasets, respectively, by varying λ and k , with σ held at its optimal value. Similarly, Figure 5.6c shows the effect of varying k and σ on the ozone dataset, while Figure 5.6d displays the impact of adjusting λ and σ for titanic dataset. These curves highlight that the performance of the proposed ProDiSE-LSSVM model is influenced by changes in the hyperparameters, underscoring the importance of

selecting optimal parameter values to achieve the best performance.

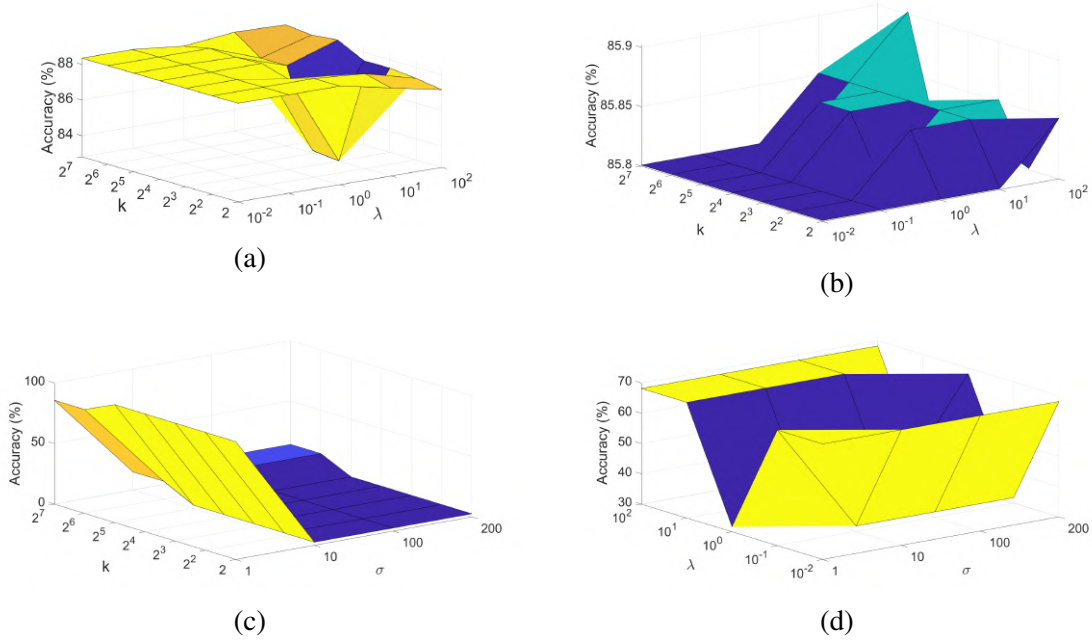


Figure 5.6: (a) Sensitivity analysis of the proposed ProDiSE-LSSVM with respect to different parameters on real-world datasets. (a) bank (b) musk_2 (c) ozone (d) titanic.

5.5.5 Impact of hyperparameters m and r on the performance of the proposed ProDiSE-LSSVM

To analyze the impact of the number of SVs, m , on the model's accuracy, experiments are conducted on the BCW, banknote, and blood datasets, varying m from 10 to 100 with a step size of 10. The accuracy of the models is then plotted against m . Sparse LSSVM [41] is excluded from this analysis as it cannot accommodate arbitrary numbers of SVs. Figure 5.7a illustrates the accuracy fluctuation of the proposed ProDiSE-LSSVM as m varies for the BCW dataset. Although the accuracy fluctuates, the proposed model outperforms others up to $m = 60$, after which GRS-LSSVM surpasses it. Figure 5.7b presents the accuracy plot for the banknote dataset, where the proposed model shows superior accuracy across various values of m up to 80. Figure 5.7c depicts the results for the blood dataset, where

the proposed model maintains a consistently higher accuracy after $m = 20$, with minimal variation. From Figures 5.7a to 5.7c, it can be inferred that the proposed ProDiSE-LSSVM consistently delivers superior performance across a range of SV values, demonstrating its reliability in handling different dataset characteristics.

To further investigate the effect of the hyperparameter r on the performance of the proposed ProDiSE-LSSVM, we conducted experiments across a range of UCI datasets, including BCW, bank, banknote, titanic, tic_tac_toe, musk_2, ozone, and blood. For each dataset, we varied r from 50 to 150 and plotted the corresponding accuracy, as shown in Figure 5.7d (where n denotes same value as r). The results indicate that changes in the value of r have a minimal impact on the accuracy of the proposed model. This demonstrates the consistency and stability of the ProDiSE-LSSVM, suggesting that its performance is not highly sensitive to variations in r .

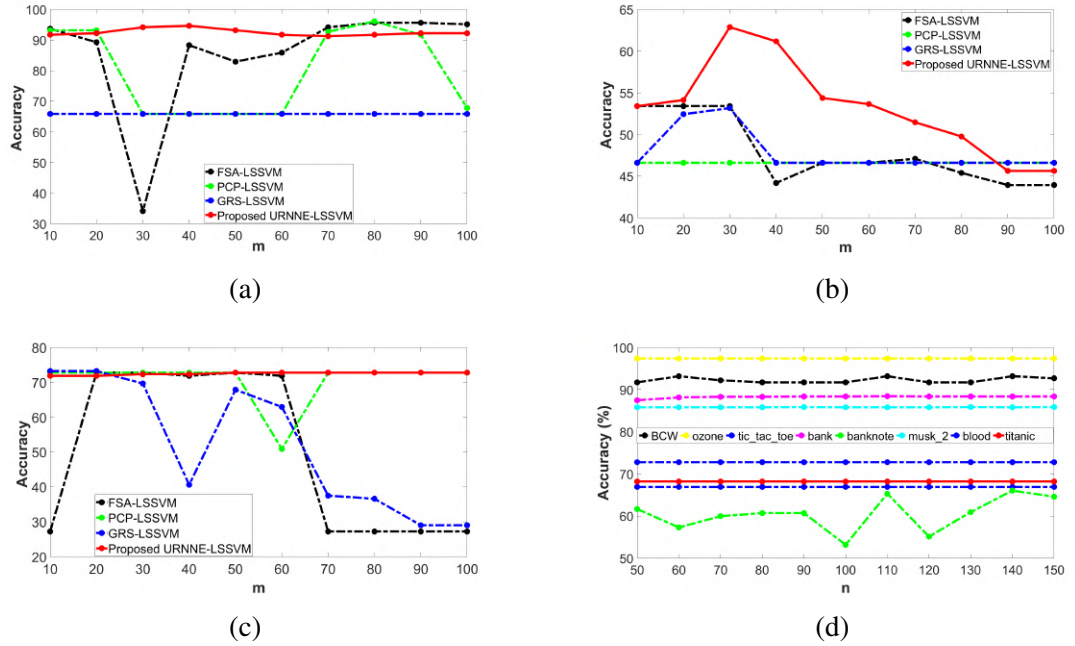


Figure 5.7: Fig. (a)-(c) represent the accuracy of datasets by varying the number of SVs (m). (a) BCW (b) banknote (c) blood. Figure (d) illustrates the impact of n on the performance of the model.

5.5.6 Statistical analysis on category A datasets

To further strengthen our analysis, we perform statistical tests, including the Friedman test and the Wilcoxon signed-rank test [123]. The Friedman test evaluates the null hypothesis that all models are equivalent in performance. The Friedman statistic, denoted as χ_F^2 , is calculated using the formula shown in equation (2.56). To maintain the conservative nature of the test, we compute the F_F statistic as shown in equation (2.57). This statistical test allows us to determine whether there are significant differences in performance among the models across multiple datasets, providing evidence to support our inference.

In our case, we have $N = 28$ and $md = 6$ which results in $\chi_F^2 = 11.1256$ and $F_F = 2.3309$ having degree of freedom 5 and $(5, 135)$, respectively. Further, $F_{(5,135)} = 2.2813 < F_F$, we reject the null hypothesis, indicating that there are statistically significant differences among the models.

To further strengthen our analysis, we perform pairwise Wilcoxon signed-rank test [123] to compare models on a pairwise basis, with the null hypothesis asserting equivalence between two models. This test ranks differences in performance metrics by magnitude, where the sum of positive ranks is denoted as R_+ and the sum of negative ranks as R_- . Table 5.3 presents the Wilcoxon test results, which reveal that for all models except RoBoSS, the null hypothesis is rejected, indicating significant performance differences. In comparison to RoBoSS, the average accuracy and average rank infer the superiority of the proposed model. Thus, the statistical analysis further reinforces the superiority of the proposed ProDiSE-LSSVM, validating its effectiveness and reliability over the competing models.

5.5.7 Experimental results on category B datasets

The performance of the proposed ProDiSE-LSSVM is evaluated on large-scale UCI datasets, specifically category B datasets, with sample sizes ranging from 20,000 to 581,012.

Table 5.3: Wilcoxon signed rank test of the baseline models with respect to the proposed ProDiSe-LSSVM.

Model	$R+$	$R-$	p -value	Null hypothesis
sparse LSSVM [41]	145	8	0.0012	rejected
FSA-LSSVM [34]	141	12	0.00228	rejected
PCP-LSSVM [166]	83	8	0.0088	rejected
RoBoSS [167]	110	80	0.5485	not rejected
GRS-LSSVM [42]	45	0	< 0.05	rejected

Table 5.4: Accuracy, training time (in seconds) and testing time (in seconds) of the proposed ProDiSE-LSSVM and baseline models on the category B datasets.

Dataset (samples, features)	FSA-LSSVM [34] Accuracy (Train time, Test time)	PCP-LSSVM [166] Accuracy (Train time, Test time)	RoBoSS [167] Accuracy (Train time, Test time)	GRS-LSSVM [42] Accuracy (Train time, Test time)	Proposed ProDiSE-LSSVM Accuracy (k) (Train time, Test time)
adult2 (48842, 14)	33.1673 (12.2666, 0.1344)	76.5714 (10.5468, 0.1609)	74.1282 (0.1723, 0.158)	76.1346 (48.1587, 0.0202)	76.1346 (1.0656, 0.0134)
connect_4 (67557, 42)	58.9826 (32.2451, 0.2221)	75.5317 (28.7412, 0.2669)	63.1865 (0.0306, 0.029)	75.3688 (130.2458, 0.0261)	75.3688 (1.3843, 0.0207)
miniboone (130064, 50)	72.1469 (183.2995, 0.8207)	72.1546 (131.9929, 0.9141)	78.449 (1.0058, 0.0303)	72.088 (855.8048, 0.0441)	72.088 (2.2493, 0.0392)
skin_segment (245057, 3)	*	*	90.8729 (1.1495, 0.0509)	79.4279 (636.8612, 0.03)	79.4279 (1.9323, 0.0244)
covtype (581012, 54)	43.785 (61858.66, 29.3838)	51.1922 (21697.75, 29.1151)	47.0328 (15.7807, 8.8841)	51.8927 (41268.97, 30.6144)	51.9217 (63.3557, 25.8027)

* denotes out of memory.

Table 5.5: Training time and testing time (in seconds) of the proposed ProDiSE-LSSVM and baseline models on NDC datasets.

Dataset	sparse LSSVM [41] (Train time, Test time)	FSA-LSSVM [34] (Train time, Test time)	PCP-LSSVM [166] (Train time, Test time)	GRS-LSSVM [42] (Train time, Test time)	Proposed ProDiSE-LSSVM (Train time, Test time)
NDC 10k	63.75 (42.6281, 0.0032)	37.15 (6.5781, 0.0258)	63.75 (1.7899, 0.0222)	63.75 (16.8149, 0.0215)	63.75 (0.5615, 0.0184)
NDC 50k	61.31 (1791.749, 0.0086)	38.76 (61.982, 0.2495)	61.66 (51.5151, 0.2412)	61.31 (406.5682, 0.1919)	61.31 (2.07, 0.1555)
NDC 1l	*	39.1 (515.5119, 0.6523)	63.89 (172.9243, 0.571)	60.98 (1027.614, 0.6305)	60.99 (2.6556, 0.5404)
NDC 3l	*	38.88667 (11012.69, 6.4853)	49.22167 (4382.3045, 5.2514)	61.21667 (7931.072, 5.0805)	61.73333 (13.9827, 4.3547)
NDC 5l	*	39.061 (49285.73, 21.9725)	43.958 (20616.54, 19.0569)	61.065 (22530.89, 21.7334)	61.563 (40.9168, 11.3753)
NDC 1m	*	*	*	61.0485 (76060.32, 167.8278)	60.447 (162.8723, 70.9912)

* denotes out of memory.

These datasets include adult2, connect_4, miniboone, skin_segment, and covtype. To optimize tuning time, we fix parameter values at the starting points of their respective ranges: $\lambda = 10^{-2}$ and $\sigma = 1$, and $k = 2$. Table 5.4 summarizes the performance comparisons of the models. Sparse LSSVM is excluded due to out-of-memory errors on these large datasets. As indicated in Table 5.4, the proposed ProDiSE-LSSVM demonstrates notably

lower training times across datasets compared to baseline algorithms, except RoBoSS. This aligns with the theoretical time complexity of the proposed ProDiSE technique and is confirmed by our experimental results. Moreover, ProDiSE-LSSVM achieves the lowest testing time among all models, underscoring its efficiency. While RoBoSS requires slightly less training time, ProDiSE-LSSVM delivers superior performance, making it a strong choice in terms of both generalization ability and computational efficiency. Figure 5.8 illustrates the performance of the proposed ProDiSE-LSSVM model alongside baseline models across category B datasets.

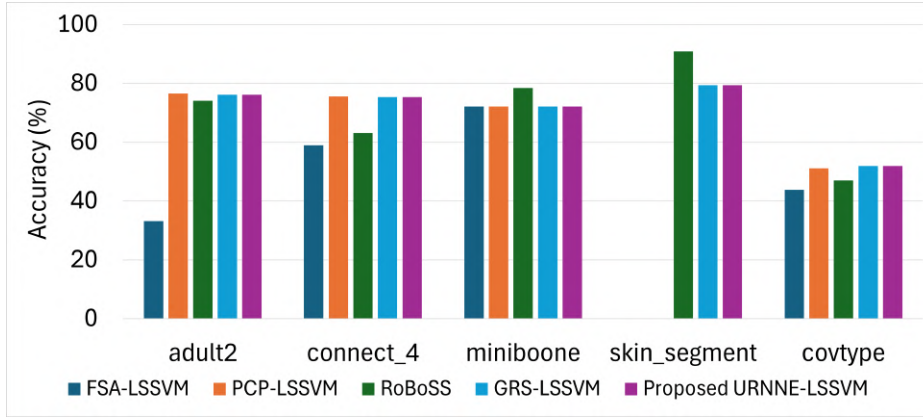


Figure 5.8: Demonstration of performance on category B datasets.

5.5.8 Experiments over NDC dataset

The performance of the proposed ProDiSE-LSSVM against baseline models is further analyzed on artificially generated NDC datasets [168], which vary in size from 10,000 to 1 million instances, each with 32 features. The datasets are denoted as follows: NDC 1m (1 million samples), NDC 1l (100,000 samples), and NDC 10k (10,000 samples). Hyperparameters for both ProDiSE-LSSVM and the baseline models are kept consistent with those used for large-scale datasets, as described in Section 5.5.7. Table 5.5 presents the accuracy, training time, and testing time (in seconds) for each model. The results indicate that

ProDiSE-LSSVM outperforms, particularly on NDC 3l and NDC 5l, and maintains comparable accuracy to GRS-LSSVM on NDC 1m. On NDC 50k and NDC 1l, PCP-LSSVM shows slightly higher accuracy than the proposed ProDiSE-LSSVM. However, the proposed model consistently demonstrates the shortest training times across datasets, underscoring its computational efficiency. These results highlight the proposed model’s capability to deliver both high performance and efficiency, making it a strong choice for large-scale data scenarios.

5.6 Summary

This chapter introduces a novel approach for efficient data selection and classification, known as the proximity and distance scoring ensemble for universal representation (ProDiSE), designed to address the challenges posed by large-scale datasets. ProDiSE focuses on selecting a compact yet highly representative subset of data points from the training set, ensuring scalability without sacrificing the quality of representation. The method leverages the nearest neighbour proximity ratio (NNPR) to capture spatial relationships among data points, combined with proximity-based scoring (PBS) and distance-based scoring (DBS) to assess local density and global distribution. Data points are assigned a representative score (RepScore), allowing the selection of highly representative samples while filtering outliers and redundant points.

To demonstrate its practical applicability, ProDiSE is integrated with least squares support vector machines (LSSVM) to form the ProDiSE-LSSVM model. This integration introduces a scalable classification framework that efficiently solves systems of linear equations with linear computational and memory complexity relative to the dataset size, making it particularly suitable for large-scale datasets. The effectiveness of ProDiSE-LSSVM is validated through extensive experimentation on diverse large-scale real-world datasets. Results

consistently demonstrate the model's superior performance and reduced training times compared to traditional methods.

The challenges posed by noise, outliers, and non-sparsity in classification models become particularly critical in OCC tasks or anomaly detection. As existing classification models utilise binary data, the next chapter explores robust and scalable solutions tailored to enhance performance and efficiency in OCC frameworks, paving the way for improved anomaly detection in diverse applications.

Chapter 6

Robust and sparse one-class classification framework

This chapter presents a transformative approach to one-class classification (OCC) by addressing two key challenges faced in hyperplane-based anomaly detection models: robustness and sparsity for large datasets. OCC refers to the task of identifying whether a data sample belongs to a known (target) class using target class data during training. It is commonly used for anomaly detection when non-target data is scarce or undefined. Robustness is a critical requirement in OCC because real-world data often contains imperfections such as noise, mislabeled entries, or measurement errors. These issues introduce outliers even within the target class. Since the OCC model is trained on the target data, it has no information about what constitutes an anomaly. Consequently, any contamination within the training data can mislead the model and result in a poorly defined decision boundary. Traditional models like least squares one-class support vector machine (LS-OCSVM) [80] is effective for clean training data, a constant mean modeling error, and Gaussian noise assumptions. Moreover, the use of squared loss in LS-OCSVM amplifies the effect of outliers, making the model particularly sensitive to corrupted data. Sparsity, on the other hand, is crucial for efficiency. Non-sparse models such as LS-OCSVM typically rely on all train-

ing samples for defining the decision function, which leads to high memory usage, increased computational cost during inference, and potential overfitting. A sparse formulation enables the model to focus only on the most representative support vectors, resulting in a compact, efficient, and generalizable solution, especially important when deploying models on large-scale or resource-constrained systems. To overcome these limitations, we introduce two novel models that redefine the OCC paradigm: the robust least squares one-class support vector machine (RLS-1SVM) and the sparse robust least squares one-class support vector machine (SRLS-1SVM).

The RLS-1SVM leverages the distributional information of the data by minimizing both the mean and variance of modeling errors, ensuring reliable performance in the presence of random noise. This integration of robustness ensures a more resilient classification framework, particularly for noisy datasets. Taking a step further, the SRLS-1SVM pioneers the introduction of sparsity into the LS-OCSVM framework for batch learning. By applying the representer theorem and pivoted Cholesky decomposition [81], the SRLS-1SVM achieves significant computational efficiency, transforming the LS-OCSVM into a scalable model capable of handling large-scale datasets with ease. This marks the first adaptation of sparse least squares modeling to OCC, addressing the memory and scalability concerns prevalent in existing approaches.

Together, the RLS-1SVM and SRLS-1SVM provide a unified solution for modern OCC challenges, advancing the machine-learning for anomaly detection.

6.1 Proposed work

The LS-OCSVM assigns equal weights to all samples, making it ineffective in the presence of outliers or non-Gaussian noise. Apart from its limited ability to handle outliers, LS-OCSVM suffers from a significant limitation in its non-sparse nature, as it treats all

training data as support vectors. This lack of sparseness increases computational complexity and extends the testing time, thus hindering the deployment of the model in real-time applications that require swift responses. This challenge becomes particularly pronounced when dealing with extensive datasets. To handle this problem, the development of a sparse LS-OCSVM variant is much needed. Further, techniques to create a robust LS-OCSVM, such as assigning weights to training data and using alternative loss functions, enhance generalization. However, weighting individual data points introduces significant computational costs [169], while alternative loss functions require extra parameters and complex optimization.

To handle the anomaly detection of an unknown non-linear system with random noise, in this paper, we propose robust least squares one-class support vector machine (RLS-1SVM) by reducing modeling error's mean and variance. Further, to address another major concern of the non-sparseness of LS-OCSVM, we propose sparse RLS-1SVM (SRLS-1SVM).

6.1.1 Proposed RLS-1SVM

The proposed RLS-1SVM includes the mean and variance of the modeling error of training data in its optimization problem, which is expressed as follows:

$$\begin{aligned} \min_{w, \rho} \quad & \frac{1}{2} \|w\|^2 - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho + \frac{1}{N} \frac{C_1}{2} \sum_{j=1}^N (\zeta_j - \tilde{\zeta})^2 + C_2 \tilde{\zeta}^2 \\ \text{subject to} \quad & w^t \psi(x_j) = (1 + \sqrt{C_2/C_1}) \rho - \zeta_j, \quad \forall j = 1, 2, \dots, N, \end{aligned} \quad (6.1)$$

where ζ_j denote the error corresponding to the j^{th} datapoint; C_1 , C_2 correspond to the regularization term. The factor $\tilde{\zeta}$ serves as a global error adjustment factor, establishing a bridge among errors and redistributes the weights of various points. The terms in the optimization problem (6.1) hold specific significance: the first term represents the regularization term,

enforcing adherence to the structural risk minimization (SRM) principle to prevent overfitting. The second term corresponds to the weighted bias of separating hyperplane. The third term accounts for the global variance observed among the training data points. The fourth term is linked to the square of the global error, equivalent to the square of the mean of errors ζ (as demonstrated in Theorem 6). Consequently, the optimization problem proposed for RLS-1SVM incorporates considerations related to the mean and variance of the error terms ζ_j .

On taking $\tilde{\zeta} = 0$ and $C_2 \rightarrow 0$, the optimization problem for the proposed RLS-1SVM simplifies to a form that is analogous to that of LS-OCSVM with a weighted bias term. Thus, LS-OCSVM is a special case of the proposed RLS-1SVM.

Assuming $\zeta_j - \tilde{\zeta} = \gamma_j$ ($j = 1, 2, \dots, N$) and substituting in the optimization problem (6.1), the problem can be formulated as follows:

$$\begin{aligned} \min_{w, \rho} \quad & \frac{1}{2} \|w\|^2 - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho + \frac{1}{N} \frac{C_1}{2} \sum_{j=1}^N \gamma_j^2 + C_2 \tilde{\zeta}^2 \\ \text{subject to} \quad & w^t \psi(x_j) = (1 + \sqrt{C_2/C_1}) \rho - \zeta_j, \quad \forall j = 1, 2, \dots, N, \end{aligned}$$

which can be equivalently written as:

$$\begin{aligned} \min_{w, \rho} \quad & \frac{1}{2} \|w\|^2 + \frac{1}{N} \frac{C_1}{2} \left(\sum_{j=1}^N \gamma_j^2 + \frac{C_2}{C_1} \sum_{j=1}^N \tilde{\zeta}^2 \right) + \frac{C_2}{2} \tilde{\zeta}^2 - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho \\ \text{subject to} \quad & w^t \psi(x_j) = (1 + \sqrt{C_2/C_1}) \rho - \zeta_j, \quad \forall j = 1, \dots, N. \end{aligned} \quad (6.2)$$

Assuming $u = \begin{bmatrix} w_1 & w_2 & \dots & w_d & \sqrt{C_2} \tilde{\zeta} \end{bmatrix}^t$; $\gamma_j = \zeta_j - \tilde{\zeta}$, for $j = 1, \dots, N$ and $\gamma_j = \sqrt{\frac{C_2}{C_1}} \tilde{\zeta}$ for $j = N + 1, N + 2, \dots, 2N$; $\Psi = \begin{bmatrix} \psi(x_1) & \psi(x_2) & \dots & \psi(x_N) \end{bmatrix}$ is $d \times N$ matrix and e denotes a column vector of ones of order N . We define $\tilde{\Psi} = \begin{bmatrix} \Psi & \mathbf{0} \\ \frac{1}{\sqrt{C_2}} e^t & \frac{-1}{\sqrt{C_1}} e^t \end{bmatrix}$, where $\mathbf{0}$ denotes a matrix of zeros having size $d \times N$. Using the above notations, the optimization

problem (6.2) reduces to

$$\begin{aligned} \min_{u, \rho} \quad & \frac{1}{2} \|u\|^2 - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho + \frac{1}{N} \frac{C_1}{2} \sum_{j=1}^{2N} \gamma_j^2 \\ \text{subject to} \quad & u^t \tilde{\psi}(x_j) = m_j \rho - \gamma_j, \quad \forall j = 1, 2, \dots, 2N, \end{aligned} \quad (6.3)$$

where $m_j = \begin{cases} 1 + \sqrt{\frac{C_2}{C_1}}, & \text{for } j = 1, 2, \dots, N, \\ 0, & \text{for } j = N + 1, N + 2, \dots, 2N, \end{cases}$; $\tilde{\psi}(x_j)$ corresponds to the j^{th} column of matrix $\tilde{\Psi}$. The Lagrangian for the optimization problem (6.3) is written as:

$$\Gamma = \frac{1}{2} \|u\|^2 - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho + \frac{1}{N} \frac{C_1}{2} \sum_{j=1}^{2N} \gamma_j^2 - \sum_{j=1}^{2N} \beta_j (u^t \tilde{\psi}(x_j) - m_j \rho + \gamma_j), \quad (6.4)$$

where β_j corresponds to the Lagrange multipliers. Further, differentiating the Lagrangian (6.4) with respect to u , ρ , γ_j and β_j , and equating to zero, we obtain:

$$\frac{\partial \Gamma}{\partial u} = u - \sum_{j=1}^{2N} \beta_j \tilde{\psi}(x_j) = 0 \implies u = \sum_{j=1}^{2N} \beta_j \tilde{\psi}(x_j), \quad (6.5)$$

$$\begin{aligned} \frac{\partial \Gamma}{\partial \rho} = -\left(1 + \frac{\sqrt{C_1}}{2}\right) + \sum_{j=1}^{2N} m_j \beta_j &= 0 \implies \sum_{j=1}^{2N} m_j \beta_j = \left(1 + \frac{\sqrt{C_1}}{2}\right) \\ \implies \sum_{j=1}^N \beta_j &= \left(1 + \frac{\sqrt{C_1}}{2}\right) / \left(1 + \sqrt{\frac{C_2}{C_1}}\right) = R \text{ (assume)}, \end{aligned} \quad (6.6)$$

$$\frac{\partial \Gamma}{\partial \gamma_j} = \frac{C_1}{N} \gamma_j - \beta_j = 0 \implies \beta_j = \frac{C_1}{N} \gamma_j, \quad \forall j = 1, 2, \dots, 2N, \quad (6.7)$$

$$\frac{\partial \Gamma}{\partial \beta_j} = u^t \tilde{\psi}(x_j) - m_j \rho + \gamma_j = 0 \quad \forall j = 1, 2, \dots, 2N. \quad (6.8)$$

Substituting equations (6.5) and (6.7) to equation (6.8), we get,

$$\sum_{i=1}^{2N} \beta_i \tilde{\psi}(x_i)^t \tilde{\psi}(x_j) - m_j \rho + \frac{N \beta_j}{C_1} = 0 \quad \forall j = 1, 2, \dots, 2N. \quad (6.9)$$

Further, from equations (6.6) and (6.9), we get the following system of linear equations:

$$\begin{bmatrix} 0 & \tilde{a}^t \\ m & \tilde{\mathbf{K}} + I \cdot N/C_1 \end{bmatrix} \begin{bmatrix} -\rho \\ \beta \end{bmatrix} = \begin{bmatrix} R \\ \mathbf{0} \end{bmatrix}. \quad (6.10)$$

Here, $\tilde{a} = \begin{bmatrix} e ; z \end{bmatrix}$, z denotes column vector of zeros of size N ; $m = \begin{bmatrix} m_1 & m_2 & \dots & m_{2N} \end{bmatrix}$; $\tilde{\mathbf{K}}$ denotes the extended kernel matrix of order $2N \times 2N$ such that

$$\tilde{\mathbf{K}} = \begin{bmatrix} K + \frac{1}{C_2}E & -\frac{1}{\sqrt{C_1 C_2}}E \\ -\frac{1}{\sqrt{C_1 C_2}}E & \frac{1}{C_1}E \end{bmatrix}, \quad (6.11)$$

where E denotes a $N \times N$ matrix having all the entries as 1. K denotes the kernel matrix.

On solving the system of linear equations, we get,

$$\beta = R \cdot \frac{(I \cdot N/C_1 + \tilde{\mathbf{K}})^{-1}m}{\tilde{a}^t(I \cdot N/C_1 + \tilde{\mathbf{K}})^{-1}m} \text{ and } \rho = \frac{R}{\tilde{a}^t(I \cdot N/C_1 + \tilde{\mathbf{K}})^{-1}m}. \quad (6.12)$$

Algorithm (1) presents the pseudocode of the proposed RLS-1SVM. Further, the system of

Algorithm 6.1 Proposed RLS-1SVM

Input: Training data $\mathcal{T}$, regularization parameters C_1, C_2 , kernel parameter σ .

Output: Labels corresponding to the testing data.

- [1] Obtain the kernel matrix K using the kernel function.
 - [2] Obtain the extended kernel matrix $\tilde{\mathbf{K}}$ using equation (6.11).
 - [3] Solve the equations (6.12) to obtain β, ρ .
 - [4] Obtain the label for test sample using equation (2.46).
-

linear equations (6.10) obtained for the proposed RLS-1SVM has an order of $(2N + 1) \times (2N + 1)$ which may incur high computational complexity of matrix inversion to get β and ρ . Moreover, owing to the non-sparseness of $(I \cdot N/C_1 + \tilde{\mathbf{K}})^{-1}$, β obtained in equation (6.12) is non-sparse. Hence, the proposed RLS-1SVM is non-sparse.

6.1.2 Proposed SRLS-1SVM

To further improve the sparseness of the proposed RLS-1SVM, we use the pivoted Cholesky factorization scheme [81, 170]. By identifying and retaining only those vectors that are essential for defining the decision boundary, the model reduces the number of support vectors needed, thereby minimizing computational complexity and memory usage. Overall, the pivoted Cholesky factorization enhances the model's efficiency with minimum compromise in the performance. To train the proposed SRLS-1SVM model, the optimization problem (6.3) is solved in its primal form. Following the representer theorem [161], the solution u of the optimization problem (6.3) is given by:

$$u = \sum_{j=1}^{2N} \beta_j \tilde{\psi}(x_j). \quad (6.13)$$

On substituting the constraint of the optimization problem (6.3) into its objective function, we obtain the equation:

$$L = \frac{1}{2} \|u\|^2 - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho + \frac{C_1}{2N} \sum_{j=1}^{2N} (m_j \rho - u^t \tilde{\psi}(x_j))^2. \quad (6.14)$$

Substituting u from equation (6.13) and solving, the equation (6.14) changes into

$$L = \frac{1}{2} \beta^t \tilde{\mathbf{K}} \beta + \frac{C_1}{2N} \beta^t \tilde{\mathbf{K}}^t \tilde{\mathbf{K}} \beta - \frac{C_1}{N} \rho m^t \tilde{\mathbf{K}} \beta - \left(1 + \frac{\sqrt{C_1}}{2}\right) \rho + \frac{C_1}{2N} m^t m \rho^2.$$

Differentiating L with respect to β and ρ , and equating to zero, we obtain

$$\frac{\partial L}{\partial \beta} = \tilde{\mathbf{K}} \beta + \frac{C_1}{N} \tilde{\mathbf{K}}^t \tilde{\mathbf{K}} \beta - \frac{C_1}{N} \rho \tilde{\mathbf{K}}^t m = 0, \quad (6.15)$$

$$\begin{aligned} \frac{\partial L}{\partial \rho} &= -\frac{C_1}{N} \beta^t \tilde{\mathbf{K}} m - \left(1 + \frac{\sqrt{C_1}}{2}\right) + \frac{C_1}{N} m^t m \rho = 0 \\ \implies \rho &= \frac{N}{C_1 m^t m} \left(1 + \frac{\sqrt{C_1}}{2} + \frac{C_1}{N} \beta^t \tilde{\mathbf{K}} m\right). \end{aligned} \quad (6.16)$$

Substituting (6.16) in (6.15) and solving, we get

$$\beta = \left(\tilde{\mathbf{K}} + \frac{C_1}{N} \tilde{\mathbf{K}}^t \tilde{\mathbf{K}} - \frac{C_1}{Nm^t m} \tilde{\mathbf{K}}^t m m^t \tilde{\mathbf{K}} \right)^{-1} \frac{1}{m^t m} \tilde{\mathbf{K}}^t m \left(1 + \frac{\sqrt{C_1}}{2} \right). \quad (6.17)$$

It is important to note that the matrix inside the inverse operator in equation (6.17) is singular due to the low-rank structure of the kernel matrix $\tilde{\mathbf{K}}$. Direct inversion of this matrix is therefore not possible. To address this issue, we approximate the kernel matrix using the Nyström approximation, which not only alleviates the singularity problem but also reduces the computational burden. Let the rank of matrix $\tilde{\mathbf{K}}$ be r , such that any r linearly independent columns (say $\mathbb{A}$) can represent $\tilde{\mathbf{K}}$. Assume $\mathbb{I} = \{1, 2, \dots, 2N\}$ be the index set. The matrix $\tilde{\mathbf{K}} = \begin{bmatrix} K_{\mathbb{I}\mathbb{A}} & K_{\mathbb{I}\mathbb{P}} \end{bmatrix}$ such that $\mathbb{P} = \mathbb{I} \setminus \mathbb{A}$ denote the non-basic indices, $K_{\mathbb{I}\mathbb{A}} \in \mathbb{R}^{2N \times r}$ and $K_{\mathbb{I}\mathbb{P}} = K_{\mathbb{I}\mathbb{A}} T$ with $T \in \mathbb{R}^{r \times (2N-r)}$. Approximate the kernel matrix using the Nyström approximation, $\tilde{\mathbf{K}} = K_{\mathbb{I}\mathbb{A}} K_{\mathbb{A}\mathbb{A}}^{-1} K_{\mathbb{A}\mathbb{I}}^t$ [171], we have

$$\begin{aligned} \beta_{\mathbb{A}} &= \left(K_{\mathbb{A}\mathbb{A}} + \frac{C_1}{N} K_{\mathbb{I}\mathbb{A}}^t K_{\mathbb{I}\mathbb{A}} - \frac{C_1}{Nm^t m} K_{\mathbb{I}\mathbb{A}}^t m m^t K_{\mathbb{I}\mathbb{A}} \right)^{-1} \frac{1}{m^t m} K_{\mathbb{I}\mathbb{A}}^t m \left(1 + \frac{\sqrt{C_1}}{2} \right) \\ &= \left(K_{\mathbb{A}\mathbb{A}} + \frac{C_1}{N} K_{\mathbb{I}\mathbb{A}}^t \left[\mathbf{I} - \frac{1}{m^t m} m m^t \right] K_{\mathbb{I}\mathbb{A}} \right)^{-1} \frac{1}{m^t m} K_{\mathbb{I}\mathbb{A}}^t m \left(1 + \frac{\sqrt{C_1}}{2} \right). \end{aligned} \quad (6.18)$$

Note: Prove that $(K_{\mathbb{A}\mathbb{A}} + \frac{C_1}{N} K_{\mathbb{I}\mathbb{A}}^t [\mathbf{I} - \frac{1}{m^t m} m m^t] K_{\mathbb{I}\mathbb{A}})$ is invertible.

Proof. To prove the inverse of the matrix exist, we will show that the matrix is positive definite.

Define $D = I - c m m^t$, where $c = \frac{1}{m^t m}$. First, we need to show that D is positive semi-definite.

For any $x \in \mathbb{R}^{2N}$, $x^t D x = x^t (I - c m m^t) x = \|x\|^2 - c \|m^t x\|^2$. Using the Cauchy-Schwarz inequality, $\|m^t x\|^2 \leq \|m\|^2 \|x\|^2$, we obtain $x^t D x \geq \|x\|^2 - c \|m\|^2 \|x\|^2 = \|x\|^2 (1 - c \|m\|^2)$. Substitute $c = \frac{1}{m^t m}$, we have $1 - c \|m\|^2 = 1 - \frac{1}{m^t m} \|m\|^2 \geq 0$. Thus, D is positive semi-definite. Now consider the second term in the matrix expression, $K_{\mathbb{I}\mathbb{A}}^t D K_{\mathbb{I}\mathbb{A}}$. For any

$x \in \mathbb{R}^r$, $x^t K_{\mathbb{I}\mathbb{A}}^t D K_{\mathbb{I}\mathbb{A}} x = (K_{\mathbb{I}\mathbb{A}} x)^t D (K_{\mathbb{I}\mathbb{A}} x)$. Since D is positive semi-definite, the above expression is non-negative. Hence, $K_{\mathbb{I}\mathbb{A}}^t D K_{\mathbb{I}\mathbb{A}}$ is positive semi-definite.

The matrix $K_{\mathbb{A}\mathbb{A}}$ is full rank, implying it is positive definite. The second term $\frac{C_1}{N} K_{\mathbb{I}\mathbb{A}}^t D K_{\mathbb{I}\mathbb{A}}$ is positive semi-definite. Since the sum of a positive definite matrix and a positive semi-definite matrix is positive definite, the given matrix is positive definite. $\square$

It can be observed that to find β in equations (6.12) and (6.17), one has to compute the inverse of $2N \times 2N$ matrix. However, after applying the approximation technique to $\tilde{\mathbf{K}}$, we need to compute inverse of $r \times r$ matrix which results in $\beta_{\mathbb{A}}$ and the remaining $\beta_{\mathbb{P}} = 0$. Thus, the proposed SRLS-1SVM is sparse.

To achieve a robust low-rank approximation of $\tilde{\mathbf{K}}$, we determine $\mathbb{A} \subset \mathbb{I}$ that captures the essential structure within the data. The determination of core set $\mathbb{A}$ enables the reduced representation while preserving the performance of the model. Building on the concept of pivoted Cholesky factorization [166], we adapt and refine this approach to precisely identify $\mathbb{A}$. This careful selection process underpins the low-rank structure of $\tilde{\mathbf{K}}$, allowing our model to achieve computational efficiency without compromising performance.

We present the pseudocode of the proposed SRLS-1SVM in Algorithm 6.2.

Algorithm 6.2 Proposed SRLS-1SVM

Input: Training data $\mathcal{T}$, regularization parameters C_1 , C_2 , kernel parameter σ .

Output: Labels corresponding to the testing data.

- [1] Obtain the basis set using pivoted Cholesky factorization.
 - [2] Calculate $\beta_{\mathbb{A}}$ using equation (6.18) and $\beta_{\mathbb{P}}$ being zero.
 - [3] Calculate ρ using equation (6.16).
 - [4] Obtain the label for the test sample using equation (2.46).
-

6.1.3 Theoretical properties

In this section, we discuss the theoretical properties of the proposed models.

Theorem 6. *The global error adjustment factor in the optimization problem (6.1) is close to the mean of the error corresponding to the data points. Further, for $C_2 \rightarrow 0$, the equality holds.*

Proof. From equation (6.5), we have

$$\begin{aligned} u &= \sum_{j=1}^{2N} \beta_j \tilde{\psi}(x_j) = \sum_{j=1}^{2N} \frac{C_1}{N} \gamma_j \tilde{\psi}(x_j) \quad (\text{using equation (6.7)}) \\ &= \sum_{j=1}^N (\zeta_j - \tilde{\zeta}) \frac{C_1}{N} \tilde{\psi}(x_j) + \sum_{j=N+1}^{2N} \left(\sqrt{\frac{C_2}{C_1}} \right) \frac{C_1}{N} \tilde{\zeta} \tilde{\psi}(x_j). \end{aligned}$$

Substituting the vectors u and $\tilde{\psi}(x_j)$ (following from assumption after problem (6.2) in the above equation, and equating the $(d+1)^{th}$ row, we get:

$$\sqrt{C_2} \tilde{\zeta} = \sum_{j=1}^N (\zeta_j - \tilde{\zeta}) \frac{C_1}{N} \frac{1}{\sqrt{C_2}} - \sum_{j=N+1}^{2N} \left(\sqrt{\frac{C_2}{C_1}} \right) \frac{C_1}{N} \tilde{\zeta} \frac{1}{\sqrt{C_1}} \implies \tilde{\zeta} = \frac{C_1}{C_1 + 2C_2} \frac{1}{N} \sum_{j=1}^N \zeta_j \quad (6.19)$$

When $C_2 \rightarrow 0$, we get $\tilde{\zeta} = (1/N) \sum_{j=1}^N \zeta_j$. Hence proved. $\square$

Theorem 7. [172]: *The consideration of mean and variance terms in the proposed RLS-1SVM and SRLS-1SVM models enhances their robustness in comparison to LS-OCSVM.*

Proof. In LS-OCSVM, the sum of modeling errors is $\sum_{j=1}^N \zeta_j^2$, where ζ_j denotes the error corresponding to j^{th} sample. The weight for the j^{th} sample's contribution to the optimization process in the original LS-OCSVM is given by:

$$\mathcal{S}_{LS-OCSVM}^j = \frac{\zeta_j^2}{\sum_{j=1}^N \zeta_j^2}.$$

The modeling error in the proposed RLS-1SVM and SRLS-1SVM is given by $(\zeta_j - \tilde{\zeta})^2 + \eta \tilde{\zeta}^2$ shown in equation (6.1), where $\eta = 2C_2/C_1$, thus the weight for j^{th} sample's contribution

to the optimization process in the proposed models is given by:

$$\mathcal{S}_{proposed}^j = \frac{(\zeta_j - \tilde{\zeta})^2 + \eta \tilde{\zeta}^2}{\sum_{j=1}^N (\zeta_j - \tilde{\zeta})^2 + N \eta \tilde{\zeta}^2}.$$

Assume $\Delta^j = \mathcal{S}_{proposed}^j - \mathcal{S}_{LS-OCSVM}^j$ and $Z = \sum_{j=1}^N \zeta_j^2 \left(\sum_{i=1}^N (\zeta_i - \tilde{\zeta})^2 + N \eta \tilde{\zeta}^2 \right)$, then

$$\begin{aligned} \Delta^j &= \frac{1}{Z} \left(((\zeta_j - \tilde{\zeta})^2 + \eta \tilde{\zeta}^2) \sum_{i=1}^N \zeta_i^2 - \zeta_j^2 \left(\sum_{i=1}^N (\zeta_i - \tilde{\zeta})^2 + N \eta \tilde{\zeta}^2 \right) \right) \\ \Rightarrow \Delta^j &= \frac{1}{Z} \left((\tilde{\zeta}^2 - 2\tilde{\zeta}\zeta_j + \eta \tilde{\zeta}^2) \sum_{i=1}^N \zeta_i^2 - \zeta_j^2 (N \tilde{\zeta}^2 + N \eta \tilde{\zeta}^2 - 2\tilde{\zeta} \sum_{i=1}^N \zeta_i) \right) \\ \Rightarrow \Delta^j &= \frac{1}{Z} \left((\tilde{\zeta}^2 - 2\tilde{\zeta}\zeta_j + \eta \tilde{\zeta}^2) \sum_{i=1}^N \zeta_i^2 + (\tilde{\zeta}^2 + 2\eta \tilde{\zeta}^2 - \eta \tilde{\zeta}^2) N \zeta_j^2 \right) \quad (\text{using relation (6.19)}). \end{aligned}$$

Further, the numerator of the above equation satisfies the following inequality for $|\zeta_j| < 0.5|\tilde{\zeta}|$

$$(\tilde{\zeta}^2 - 2\tilde{\zeta}\zeta_j + \eta \tilde{\zeta}^2) \sum_{i=1}^N \zeta_i^2 + (\tilde{\zeta}^2 + 2\eta \tilde{\zeta}^2 - \eta \tilde{\zeta}^2) N \zeta_j^2 \geq \eta \tilde{\zeta}^2 \sum_{i=1}^N \zeta_i^2 + (\tilde{\zeta}^2 + \eta \tilde{\zeta}^2) N \zeta_j^2 \geq 0.$$

Thus, when $|\zeta_j| < 0.5|\tilde{\zeta}|$, $\mathcal{S}_{LS-OCSVM}^j < \mathcal{S}_{proposed}^j$, that is, the proposed models assign greater weight to training samples having smaller errors than original LS-OCSVM. $\square$

Definition 8. [173]: The term robustness of a model implies that if a training sample and a testing sample are similar, then their associated losses are also close.

Definition 9. [173]: An algorithm $\mathcal{A}$ is considered robust with parameters $(P, \theta(\cdot))$, where $P \in \mathbb{N}$ and $\theta(\cdot) : \mathcal{Z}^N \rightarrow \mathbb{R}$, with $\mathcal{Z}$ corresponding to the sample space $Y \times X$ and if $\mathcal{Z}$ can be partitioned into P disjoint subsets, represented as $\{\mathcal{C}_j\}_{j=1}^P$. Then, $\forall \mathbf{a} \in \mathcal{Z}^N$ and for the given loss function $l(\mathcal{A}_{\mathbf{a}}, a)$, the following holds:

$$\forall \mathbf{a} \in \mathbf{a}, \forall z \in \mathcal{Z}, \forall j = 1, 2, \dots, P : \text{ if } a, z \in \mathcal{C}_j, \text{ then } |l(\mathcal{A}_{\mathbf{a}}, a) - l(\mathcal{A}_{\mathbf{a}}, z)| \leq \theta(\mathbf{a}),$$

where $\mathbf{a} \in \mathcal{Z}^N$ signifies the N training samples.

In the following theorem, we establish the robustness of the baseline LS-OCSVM using Definition 9.

Theorem 10. Let X be a compact set and $l(w, z) = (\rho - w^t \psi(z_x))^2$, where z_x and z_y denote the x and y component of $z \in \mathcal{Z}$, respectively. Assume $\mathcal{S}$ denote the kernel space and $f_{\mathcal{S}}(\lambda) = \max_{p, q \in X, \|p-q\| \leq \lambda} (K(p, p) + K(q, q) - 2K(p, q))$ and $g_{\mathcal{S}}(\lambda) = \max_{p, q \in X, \|p+q\| \leq \lambda} (K(p, p) + K(q, q) + 2K(p, q))$. For a continuous kernel function $K(\cdot, \cdot)$ and non-negative λ , $f_{\mathcal{S}}(\lambda)$ is finite and LS-OCSVM is $\left(2\mathcal{N}(\frac{\lambda}{2}, X, \|\cdot\|), \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*} \cdot (\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + 2|\rho^*|)\right)$ robust with optimal bias ρ^* .

Proof. Let (w^*, ρ^*) be the optimal solution for the optimization problem (2.41) which is expressed as:

$$\begin{aligned} \min_{w, \rho} J(w, \rho) &= \frac{1}{2} \|w\|^2 - \rho + \frac{C}{2} \sum_{j=1}^N (\rho - w^t \psi(x_j))^2 \\ \implies J(w^*, \rho^*) &= \frac{1}{2} \|w^*\|^2 - \rho^* + \frac{C}{2} \sum_{j=1}^N (\rho^* - w^{*t} \psi(x_j))^2 \\ \implies J(w^*, \rho^*) &\leq J(0, 0) = 0 \implies \frac{1}{2} \|w^*\|^2 - \rho^* \leq 0. \end{aligned}$$

Thus, we get $\|w^*\|^2 \leq 2\rho^*$. Following the definition of covering number [174], the input space X can be partitioned into $\mathcal{N}(\lambda/2, X, \|\cdot\|)$ disjoint subsets ensuring that the radius of each subset does not exceed $\frac{\lambda}{2}$. Further, for OCC problem, Y can be partitioned into $\{1, -1\}$, leading to the partition of $\mathcal{Z}$ into $2\mathcal{N}(\lambda/2, X, \|\cdot\|)$ disjoint subsets. Suppose z_1 and z_2 belong to the same subset, and denote $z_{i(y)} = y_i$ and $z_{i(x)} = x_i$ such that $y_1 = y_2$ and $\|x_1 - x_2\| \leq \lambda$. Therefore, we have,

$$|l(w^*, z_1) - l(w^*, z_2)| = |(\rho^* - w^{*t} \psi(x_1))^2 - (\rho^* - w^{*t} \psi(x_2))^2|$$

$$\begin{aligned}
&= |w^{*t}(\psi(x_1) - \psi(x_2))(w^{*t}(\psi(x_1) + \psi(x_2)) - 2\rho^*)| \\
&\leq |w^{*t}(\psi(x_1) - \psi(x_2))| \cdot (|w^{*t}(\psi(x_1) + \psi(x_2))| + |2\rho^*|) \\
&\leq \|w^*\| \sqrt{\langle \psi(x_1) - \psi(x_2), \psi(x_1) - \psi(x_2) \rangle} \\
&\quad \cdot (\|w^*\| \sqrt{\langle \psi(x_1) + \psi(x_2), \psi(x_1) + \psi(x_2) \rangle} + 2|\rho^*|) \\
&\leq \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*} \cdot (\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + 2|\rho^*|).
\end{aligned}$$

The last second step follows through Cauchy-Schwarz inequality. Hence, Theorem 10 holds. $\square$

Further, we prove the robustness of the proposed RLS-1SVM and SRLS-1SVM utilizing the Definition 9. The loss imposed on each sample x_j for the proposed models is given by $l_{\text{proposed}} = (\zeta_j - \tilde{\zeta})^2 + \frac{2C_2}{C_1}\tilde{\zeta}^2$ from equation (6.1), which can be further simplified using equation (6.19) as:

$$\begin{aligned}
l_{\text{proposed}} &= \left(\zeta_j - \frac{C_1}{C_1 + 2C_2} \frac{1}{N} \sum_{i=1}^N \zeta_i \right)^2 + \frac{2C_2}{C_1} \left(\frac{C_1}{C_1 + 2C_2} \frac{1}{N} \sum_{i=1}^N \zeta_i \right)^2 \\
&= \zeta_j^2 - \frac{C_1}{C_1 + 2C_2} \frac{2}{N} \zeta_j \sum_{i=1}^N \zeta_i + \left(\frac{C_1}{C_1 + 2C_2} \right)^2 \left(\frac{1}{N} \sum_{i=1}^N \zeta_i \right)^2 \\
&\quad + \frac{2C_1C_2}{(C_1 + 2C_2)^2} \left(\frac{1}{N} \sum_{i=1}^N \zeta_i \right)^2.
\end{aligned} \tag{6.20}$$

Substituting $\zeta_j = (\rho - u^t \tilde{\psi}(x_j))$, we obtain:

$$\zeta_j^2 = (u^t \tilde{\psi}(x_j))^2 + \rho^2 - 2\rho u^t \tilde{\psi}(x_j), \tag{6.21}$$

$$\zeta_j \sum_{i=1}^N \zeta_i = (\rho - u^t \tilde{\psi}(x_j)) \left(\sum_{i=1}^N (\rho - u^t \tilde{\psi}(x_i)) \right). \tag{6.22}$$

Theorem 11. *The proposed models RLS-1SVM and SRLS-1SVM with loss function l_{proposed} is*

$(2\mathcal{N}(\frac{\lambda}{2}, X, \|\cdot\|), \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*}(\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + M + 2|\rho^*|))$ robust, where $f_{\mathcal{S}}(\lambda) = \max_{p,q \in X, \|p-q\| \leq \lambda} (K(p,p) + K(q,q) - 2K(p,q))$, $g_{\mathcal{S}}(\lambda) = \max_{p,q \in X, \|p+q\| \leq \lambda} (K(p,p) + K(q,q) + 2K(p,q))$ and M denotes a constant.

Proof. Let l_{proposed} denotes the loss function of the proposed RLS-1SVM and SRLS-1SVM. Assume two samples z_1 and z_2 belong to the same partition with the same label. By utilizing equations (6.21) and (6.22) in equation (6.20), we obtain:

$$\begin{aligned}
 & |l_{\text{proposed}}(u^*, z_1) - l_{\text{proposed}}(u^*, z_2)| \\
 &= |(u^{*t}\psi(x_1))^2 - (u^{*t}\psi(x_2))^2 - 2\rho^*(u^{*t}\psi(x_1) - u^{*t}\psi(x_2)) - \frac{2C_1}{C_1 + 2C_2} \frac{1}{N} (u^{*t}\psi(x_2) - u^{*t}\psi(x_1))\alpha_1| \\
 &= |u^{*t}(\psi(x_1) - \psi(x_2)) \left(u^{*t}(\psi(x_1) + \psi(x_2)) - 2\rho^* + \frac{2C_1}{C_1 + 2C_2} \frac{1}{N} \alpha_1 \right)| \\
 &\leq |u^{*t}(\psi(x_1) - \psi(x_2))| \cdot \left(|u^{*t}(\psi(x_1) + \psi(x_2))| + |2\rho^*| + \frac{2C_1}{C_1 + 2C_2} \frac{1}{N} |\alpha_1| \right) \\
 &\leq \|u^*\| \sqrt{\langle \psi(x_1) - \psi(x_2), \psi(x_1) - \psi(x_2) \rangle} \cdot \left(\|u^*\| \sqrt{\langle \psi(x_1) + \psi(x_2), \psi(x_1) + \psi(x_2) \rangle} + M + 2|\rho^*| \right) \\
 &\leq \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*} \left(\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + M + 2|\rho^*| \right),
 \end{aligned}$$

where $M = \frac{2C_1}{C_1 + 2C_2} \frac{1}{N} |\alpha_1|$ with $\alpha_1 = \sum_{i=1}^N (\rho - u^{*t}\psi(x_i))$ being constant for any choice of samples. Therefore, Theorem 11 holds. $\square$

Theorem 12. [173]: Let $E_{ex}(\mathcal{A})$ and $E_{tr}(\mathcal{A})$ denote the expected and the training error of an algorithm $\mathcal{A}$, and the loss function has upper bound U . If the algorithm is $(P, \theta(\cdot))$ robust. Then for any $\nu > 0$, with probability at least $1 - \nu$, the following holds:

$$|E_{ex}(\mathcal{A}) - E_{tr}(\mathcal{A})| \leq \theta(\mathbf{a}) + U \sqrt{\frac{2P \ln 2 + 2 \ln(1/\nu)}{N}}. \quad (6.23)$$

Following the above theorem, we can bound the generalization error of the proposed RLS-1SVM and SRLS-1SVM, which can be stated as follows:

Theorem 13. For any $\nu > 0$ having probability $1 - \nu$, the generalization error of the proposed RLS-1SVM and SRLS-1SVM satisfies the following inequality:

$$|E_{ex}(\mathcal{A}) - E_{tr}(\mathcal{A})| \leq \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*} \cdot \left(\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + M + 2|\rho^*| \right) \quad (6.24)$$

$$+ V \sqrt{\frac{4\mathcal{N}(\frac{\lambda}{2}, X, \|\cdot\|) \ln 2 + 2 \ln(1/\nu)}{N}}, \quad (6.25)$$

where V denotes the upper bound of loss function $l_{proposed}$.

Proof. The loss function considered in the proposed RLS-1SVM and SRLS-1SVM is the mean and variance of the modeling error, which are finite quantities. Thus, $l_{proposed}$ is finite quantity, bounded by V . By Theorem 11, one obtains the proposed models are $(2\mathcal{N}(\frac{\lambda}{2}, X, \|\cdot\|), \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*}(\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + M + 2|\rho^*|))$ robust. Substituting $P = (2\mathcal{N}(\frac{\lambda}{2}, X, \|\cdot\|), \theta(\mathbf{a}) = \sqrt{2f_{\mathcal{S}}(\lambda)\rho^*}(\sqrt{2g_{\mathcal{S}}(\lambda)\rho^*} + M + 2|\rho^*|))$ into equation (6.23), we obtain the required inequality (6.24). $\square$

6.1.4 Time complexity

In this subsection, we delve into the time complexity analysis of the proposed RLS-1SVM and SRLS-1SVM by assuming N input samples with d features. The training phase of RLS-1SVM solves a system of linear equations (6.10), which leads to the inversion of $(2N + 1) \times (2N + 1)$ matrix (shown in equation (6.12)). The time complexity associated with the inversion of this matrix is $O((2N + 1)^3)$ [175]. Thus, the proposed RLS-1SVM has the complexity of $O((2N)^3)$. On the other hand, SRLS-1SVM involves approximating the extended kernel matrix $\tilde{\mathbf{K}}$ to a matrix of rank r ($r \ll N$) using pivoted Cholesky decomposition. The determination of the basis set has a complexity of $O(Nr^2)$ [166]. Moreover, the training of SRLS-1SVM includes the inversion of an $r \times r$ matrix, which has the complexity of $O(r^3)$. Therefore, the total time complexity of the proposed SRLS-1SVM is

$O(Nr^2) + O(r^3)$ which is lower and demonstrates improved computational efficiency with respect to the proposed RLS-1SVM.

6.2 Assessment of proposed models versus baseline approaches

In this section, we compare different existing models with the proposed RLS-1SVM and SRLS-1SVM models.

6.2.1 Proposed models versus OCSVM

- **Computational efficiency:** OCSVM uses hinge loss and solves a QPP, leading to higher training times [60]. In contrast, the proposed models solve a system of linear equations, significantly reducing computational cost.
- **Robustness to outliers and noise:** OCSVM is sensitive to outliers and random noise [176], whereas the proposed models incorporate distribution information, enhancing their ability to handle noise and resist the influence of outliers effectively.

6.2.2 Proposed models versus LS-OCSVM

- **Robustness to outliers:** LS-OCSVM uses a least squares loss, which is sensitive to outliers [158]. However, the proposed RLS-1SVM and SRLS-1SVM consider the mean and variance of modeling error which improves robustness and reduces the influence of outliers.
- **Sparsity:** LS-OCSVM lacks sparsity [80], leading to a larger number of support vectors. RLS-1SVM improves robustness but doesn't enforce sparsity, whereas SRLS-

1SVM incorporates sparsity, focusing on key samples and leading to fewer support vectors and a more compact model.

- **Computational efficiency:** LS-OCSVM and proposed RLS-1SVM have higher computational requirements due to the lack of enforced sparsity, especially on larger datasets. However, the proposed SRLS-1SVM, with its sparse structure, achieves faster training and testing times, making it more efficient.

6.2.3 Proposed models versus ROCSVM-RHHQ

- **Optimization efficiency:** ROCSVM-RHHQ employs a rescaled hinge loss with a half-quadratic optimization technique, resulting in a high computational burden [177], whereas RLS-1SVM achieves efficiency by solving a system of linear equations.
- **Robustness to outliers:** While the rescaled hinge loss in ROCSVM-RHHQ improves robustness, its effectiveness decreases as the proportion of outliers in the training set increases [177]. In contrast, the proposed models incorporate distributional information, effectively maintaining performance in the presence of outliers.

6.2.4 Proposed models versus MKCLS-OCSVM

- **Computational efficiency:** The proposed models solve a system of linear equations, leading to faster training times and reduced computational overhead compared to the dynamic half-quadratic optimization technique used in MKCLS-OCSVM [178].
- **Enhanced robustness and sparsity:** The proposed SRLS-1SVM incorporates sparsity, allowing it to focus on key samples and reduce sensitivity to outliers, which MKCLS-OCSVM lacks [178]. This sparsity improves generalization and reduces reliance on large data samples.

- **Versatile robustness mechanism:** While MKCLS-OCSVM relies on multi-kernel similarity measures for robustness [178], the proposed models achieve robustness through regularization. These approaches allow effective handling of noise and outliers without the need for complex kernel combinations.

6.3 Numerical experiments

To analyze the performance of the proposed RLS-1SVM and SRLS-1SVM for OCC, we conduct numerical experiments of the proposed models and the baseline models, OCSVM [60], LS-OCSVM [80], ROCSVM-RHHQ [177] and MKCLS-OCSVM [178]. In this section, we give a detailed discussion about the numerical experiments.

6.3.1 Experimental setup

All numerical experiments are conducted using MATLAB 2022b on a desktop PC equipped with an 11th Gen Intel(R) Core(TM) i7-11700 processor running at 2.50GHz and 16GB of RAM. We selected publicly available real-world UCI [126] datasets to execute the experiments, initially normalizing them using the Z-score technique. Furthermore, we carried out 5 execution on each dataset and reported the average results. For each execution, the training, validation and testing datasets are divided by following the method mentioned in [178]: the training set comprises 60% target class samples and 10% non-target class samples, with the labels of the non-target class samples reversed to simulate the influence of outliers. The validation set consists of 20% target class samples and 45% non-target class samples. The remaining samples constitute the testing set, which is employed for the final performance evaluation.

The regularization parameters C_1 , C_2 for the proposed models and ROCSVM-RHHQ, and C for the baseline models OCSVM, LS-OCSVM and MKCLS-OCSVM are selected

from the range $[10^{-3}, 10^{-2}, \dots, 10^2, 10^3]$. The kernel parameter σ is selected from the range $[2^{-3}, 2^{-2}, \dots, 2^2, 2^3]$. The remaining parameters range of MKCLS-OCSVM is followed same as in [178]. The optimal parameters for all models, excluding MKCLS-OCSVM, are determined through grid search within the specified ranges. This process involves using the training set to find the optimal parameters and validating them on the separate validation set. Subsequently, the testing performance is evaluated using the designated testing set based on the obtained optimal parameter values. For the proposed SRLS-1SVM, we empirically chose the rank r as 20% of the total samples which corresponds to the support vectors.

The metric used to analyze the performance of the models is G-mean [179] which utilizes precision $(p) = tp/(tp + fp)$ and recall $(r) = tp/(tp + fn)$, and is defined as follows: $G\text{-mean} = \sqrt{p \times r}$ where tp represents correctly predicted target samples, tn represents correctly predicted non-target samples, fp and fn represent incorrectly predicted target and non-target samples, respectively.

6.3.2 Experimental analysis

The information pertaining to the real-world dataset is outlined in Table 6.1 which includes the dataset's name, sample and feature counts, and domain. It further specifies the target and non-target classes of datasets, along with the corresponding sample count for the target class. We have considered 33 UCI datasets from various domains such as health and medicine, business, social science, physics and chemistry, and others. Out of the 33 datasets, 22 have a size less than 1000 and 11 datasets have a size between 1000 to 19020. Further, the performances of the models are shown in Table 6.2, which contains the average G-mean value across 5 iteration along with the standard deviation (std) training time (in seconds (s)) and testing time (in seconds) of the models. Table 6.2 represents the proposed SRLS-1SVM has superior performance with average performance across all the datasets being 0.6876, fol-

Table 6.1: Details of the datasets.

Dataset	Abr.*	#samples	# features	Domain	Target class	# target	non-target class
breast_cancer	DS1	286	9	health and medicine	nonrecurrence of disease	201	recurrence of disease
breast_cancer_wisc_diag	DS2	569	30	health and medicine	benign	357	malignant
breast_cancer_wisc_prog	DS3	198	33	health and medicine	nonrecurrence of disease	151	recurrence of disease
congressional_voting	DS4	435	16	Social science	democrat	267	republican
credit_approval	DS5	690	15	Business	credit card not approved	383	credit card approved
cylinder_bands	DS6	512	35	physics and chemistry	no band	312	band
echocardiogram	DS7	131	10	health and medicine	died before 1 year	88	alive for 1 year
fertility	DS8	100	9	health and medicine	normal diagnosis	88	altered diagnosis
haberman_survival	DS9	306	3	health and medicine	patient survived for ≥ 5 year	225	patient survived < 5 year
hepatitis	DS10	155	19	health and medicine	alive	123	die
horse_colic	DS11	368	25	biology	pathology data absent	232	pathology data present
ilpd_indian_liver	DS12	583	9	health and medicine	not suffer	416	suffer
molec_biol_promoter	DS13	106	57	biology	not promoter	53	promoter
monks_2	DS14	601	6	other	two features are not 1	395	exactly two features has 1
monks_3	DS15	554	6	other	-1	266	1
musk_1	DS16	476	166	physics and chemistry	non musk molecule	207	musk molecule
oocytes_trisopterus_nucleus_2f	DS17	912	25	-	1	527	-1
planning	DS18	182	12	computer science	planning state of brain	130	realzed state of brain
spect	DS19	265	22	health and medicine	abnormal heart	110	normal heart
spectf	DS20	267	44	health and medicine	normal heart	55	abnormal heart
statlog_australian_credit	DS21	690	14	business	not granted credit	468	granted credit
vertebral_column_2clases	DS22	310	6	health and medicine	abnormal patients	210	normal patients
statlog_german_credit	DS23	1000	24	social science	good credit risk	700	bad credit risk
oocytes_merlucius_nucleus_4d	DS24	1022	41	-	1	685	-1
bank	DS25	4521	16	business	term deposit	4000	no term deposit
hill_valley	DS26	1212	100	other	hill	606	valley
magic	DS27	19020	10	physics and chemistry	gamma	12332	hadron
mushroom	DS28	8124	21	biology	edible	4208	poisonous
titanic	DS29	2201	3	other	not survived	1490	
chess_krvkp	DS30	3196	36	games	white won	1669	white lost
musk_2	DS31	6598	166	physics and chemistry	non musk molecule	207	musk molecule
ozone	DS32	2536	72	climate and environment	Ozone day	2463	normal day
spambase	DS33	4601	57	computer science	non-spam mail	2788	spam mail

Abr.* denotes abbreviation.

lowed by proposed RLS-1SVM which has an average G-mean value of 0.6823 across all the 33 datasets. The remaining models have average G-mean values of 0.6539, 0.5868, 0.5822 and 0.5757 for OCSVM, ROCSVM-RHHQ, LS-OCSVM, and MKCLS-OCSVM, respectively. To mitigate the impact of biasness in the average performance across all datasets, where superior performance on one dataset may overshadow poor performance on another, we assigned ranks to the performance of each dataset. The model with the best performance receives the lowest rank, and vice versa. The average rank for the proposed SRLS-1SVM is 2.15, followed by the proposed RLS-1SVM with an average rank of 2.42. OCSVM holds the third position with an average rank of 3, while the remaining models have ranks of 4.27, 4.48, and 4.67 for LS-OCSVM, ROCSVM-RHHQ, and MKCLS-OSVM, respectively. These results suggest that the integration of sparsity-promoting mechanisms in the OCSVM

CHAPTER 6. ROBUST AND SPARSE ONE-CLASS CLASSIFICATION FRAMEWORK

Table 6.2: Average G-mean with standard deviation (std), training time (Tr) and testing (Ts) time (in seconds) of the proposed RLS-1SVM, SRLS-1SVM, and the baseline models on UCI datasets (bold values represent the best G-mean values).

Dataset	OCSVM [60] G-mean $\pm$ std (Tr time, Ts time)	LS-OCSVM [80] G-mean $\pm$ std (Tr time, Ts time)	ROCSVM-RHHQ [177] G-mean $\pm$ std (Tr time, Ts time)	MKCLS-OCSVM [178] G-mean $\pm$ std (Tr time, Ts time)	proposed RLS-1SVM G-mean $\pm$ std (Tr time, Ts time)	proposed SRLS-1SVM G-mean $\pm$ std (Tr time, Te time)
DS1	0.6597 $\pm$ 0.1128 (0.0162, 0.0009)	0.6095 $\pm$ 0.0288 (0.0014, 0.0012)	0.5598 $\pm$ 0.0485 (0.0564, 0.0007)	0.5591 $\pm$ 0.0512 (0.0074, 0.0016)	0.706 $\pm$ 0.0233 (0.0011, 0.0014)	0.7127 $\pm$ 0.0115 (0.0007, 0.0015)
DS2	0.5521 $\pm$ 0.0716 (0.0488, 0.0036)	0.6756 $\pm$ 0.0327 (0.0025, 0.0079)	0.6216 $\pm$ 0.0345 (0.2748, 0.0043)	0.6971 $\pm$ 0.0959 (0.0054, 0.0027)	0.664 $\pm$ 0.015 (0.0037, 0.0044)	0.6664 $\pm$ 0.0163 (0.0013, 0.002)
DS3	0.767 $\pm$ 0 (0.0103, 0.0005)	0.5681 $\pm$ 0.0606 (0.0004, 0.0005)	0.5606 $\pm$ 0.0665 (0.0361, 0.0005)	0.5026 $\pm$ 0.0207 (0.0025, 0.0005)	0.7633 $\pm$ 0.0073 (0.0006, 0.0008)	0.7633 $\pm$ 0.0073 (0.0005, 0.0008)
DS4	0.5154 $\pm$ 0.0875 (0.0098, 0.0013)	0.5734 $\pm$ 0.017 (0.001, 0.0018)	0.585 $\pm$ 0.0364 (0.0717, 0.0013)	0.5697 $\pm$ 0.0177 (0.0059, 0.0028)	0.649 $\pm$ 0.0019 (0.002, 0.0018)	0.647 $\pm$ 0 (0.0007, 0.0014)
DS5	0.4587 $\pm$ 0.1886 (0.0653, 0.0038)	0.4979 $\pm$ 0.0629 (0.0025, 0.0055)	0.5773 $\pm$ 0.0322 (0.3213, 0.0049)	0.5344 $\pm$ 0.0285 (0.0083, 0.0097)	0.5497 $\pm$ 0.0498 (0.0062, 0.0062)	0.5836 $\pm$ 0.0142 (0.0012, 0.0022)
DS6	0.5134 $\pm$ 0.2567 (0.0467, 0.0017)	0.5801 $\pm$ 0.0494 (0.0012, 0.0018)	0.601 $\pm$ 0.0321 (0.2425, 0.0023)	0.5327 $\pm$ 0.0564 (0.0068, 0.0033)	0.6219 $\pm$ 0.0151 (0.0037, 0.0032)	0.6391 $\pm$ 0.0155 (0.0009, 0.0018)
DS7	0.5736 $\pm$ 0.1253 (0.0037, 0.0003)	0.6672 $\pm$ 0.0622 (0.0001, 0.0002)	0.5333 $\pm$ 0.0626 (0.0238, 0.0003)	0.6541 $\pm$ 0.1126 (0.0029, 0.0006)	0.6595 $\pm$ 0.0708 (0.0002, 0.0005)	0.6791 $\pm$ 0.0117 (0.0002, 0.0002)
DS8	0.8597 $\pm$ 0 (0.0043, 0.0003)	0.7167 $\pm$ 0.0882 (0.0001, 0.0002)	0.6592 $\pm$ 0.0592 (0.0227, 0.0002)	0.5226 $\pm$ 0.1229 (0.0031, 0.0002)	0.8532 $\pm$ 0.013 (0.0002, 0.0002)	0.8597 $\pm$ 0 (0.0001, 0.0002)
DS9	0.5971 $\pm$ 0.0707 (0.0215, 0.0011)	0.6027 $\pm$ 0.027 (0.0016, 0.0012)	0.6616 $\pm$ 0.0666 (0.0641, 0.0007)	0.5882 $\pm$ 0.0614 (0.0043, 0.0009)	0.7403 $\pm$ 0.0068 (0.0015, 0.0011)	0.7416 $\pm$ 0.0077 (0.0009, 0.0009)
DS10	0.7845 $\pm$ 0 (0.0104, 0.0006)	0.7382 $\pm$ 0.111 (0.0002, 0.0003)	0.6886 $\pm$ 0.0755 (0.065, 0.0006)	0.6862 $\pm$ 0.0689 (0.0013, 0.0005)	0.7742 $\pm$ 0.0175 (0.0005, 0.0006)	0.7805 $\pm$ 0.0528 (0.0013, 0.0006)
DS11	0.6597 $\pm$ 0 (0.0221, 0.0012)	0.3737 $\pm$ 0.0498 (0.001, 0.0009)	0.5088 $\pm$ 0.0754 (0.0843, 0.001)	0.6034 $\pm$ 0.069 (0.0041, 0.0011)	0.6597 $\pm$ 0 (0.0015, 0.0019)	0.6565 $\pm$ 0.0064 (0.0048, 0.0013)
DS12	0.7248 $\pm$ 0 (0.0917, 0.0046)	0.5528 $\pm$ 0.0245 (0.0029, 0.0054)	0.6249 $\pm$ 0.0163 (0.3248, 0.0049)	0.486 $\pm$ 0.0455 (0.0147, 0.003)	0.7248 $\pm$ 0 (0.0117, 0.0146)	0.7252 $\pm$ 0.0009 (0.0015, 0.0019)
DS13	0.5318 $\pm$ 0.021 (0.0039, 0.0003)	0.3586 $\pm$ 0.0878 (0.0001, 0.0003)	0.4972 $\pm$ 0.0972 (0.0233, 0.0005)	0.5137 $\pm$ 0.0528 (0.0012, 0.0005)	0.5421 $\pm$ 0.0116 (0.0001, 0.0004)	0.5395 $\pm$ 0.0584 (0.0001, 0.0003)
DS14	0.6797 $\pm$ 0 (0.0491, 0.0042)	0.5063 $\pm$ 0.0685 (0.0016, 0.0039)	0.4502 $\pm$ 0.0535 (0.328, 0.005)	0.5317 $\pm$ 0.0479 (0.0054, 0.0021)	0.619 $\pm$ 0.1215 (0.0063, 0.0077)	0.6776 $\pm$ 0.0097 (0.0012, 0.0023)
DS15	0.5266 $\pm$ 0 (0.0753, 0.0046)	0.3737 $\pm$ 0.0416 (0.0011, 0.0073)	0.4161 $\pm$ 0.0308 (0.1039, 0.0028)	0.5255 $\pm$ 0.0497 (0.0035, 0.0023)	0.5145 $\pm$ 0.0392 (0.0027, 0.005)	0.5122 $\pm$ 0.0196 (0.0014, 0.0013)
DS16	0.6061 $\pm$ 0 (0.0222, 0.0028)	0.3699 $\pm$ 0.0346 (0.002, 0.0035)	0.5155 $\pm$ 0.0418 (0.0582, 0.0021)	0.6258 $\pm$ 0.0254 (0.0056, 0.0046)	0.6086 $\pm$ 0.002 (0.0025, 0.0033)	0.6069 $\pm$ 0.0017 (0.0012, 0.0021)
DS17	0.6164 $\pm$ 0 (0.2109, 0.0087)	0.4919 $\pm$ 0.0199 (0.0049, 0.0081)	0.4911 $\pm$ 0.0345 (0.8283, 0.0145)	0.5213 $\pm$ 0.083 (0.0128, 0.0104)	0.6171 $\pm$ 0.0013 (0.009, 0.0133)	0.6157 $\pm$ 0.002 (0.002, 0.0042)
DS18	0.5995 $\pm$ 0.2432 (0.0102, 0.001)	0.5041 $\pm$ 0.0937 (0.0002, 0.0005)	0.4825 $\pm$ 0.0726 (0.0478, 0.0017)	0.4713 $\pm$ 0.0501 (0.0027, 0.0011)	0.7211 $\pm$ 0 (0.0008, 0.0007)	0.7211 $\pm$ 0 (0.0003, 0.0004)
DS19	0.5434 $\pm$ 0.1581 (0.0086, 0.0012)	0.5228 $\pm$ 0.0338 (0.0004, 0.0008)	0.4742 $\pm$ 0.0254 (0.0432, 0.0007)	0.5782 $\pm$ 0.0334 (0.0054, 0.0021)	0.6199 $\pm$ 0.0074 (0.0006, 0.001)	0.6208 $\pm$ 0.0079 (0.0005, 0.0009)
DS20	0.8011 $\pm$ 0 (0.0121, 0.0008)	0.6546 $\pm$ 0.0519 (0.0008, 0.0031)	0.6525 $\pm$ 0.0553 (0.0904, 0.0011)	0.6906 $\pm$ 0.0894 (0.0043, 0.0022)	0.7998 $\pm$ 0.0062 (0.0012, 0.0012)	0.7986 $\pm$ 0.0051 (0.0006, 0.0023)
DS21	0.6942 $\pm$ 0 (0.1155, 0.0081)	0.568 $\pm$ 0.02 (0.0017, 0.0043)	0.576 $\pm$ 0.0176 (0.5128, 0.0073)	0.4711 $\pm$ 0.0503 (0.0076, 0.0029)	0.693 $\pm$ 0.0024 (0.0069, 0.0058)	0.6942 $\pm$ 0 (0.0023, 0.0022)
DS22	0.6948 $\pm$ 0 (0.0296, 0.0012)	0.5442 $\pm$ 0.0472 (0.0009, 0.0012)	0.5014 $\pm$ 0.0489 (0.058, 0.0007)	0.5427 $\pm$ 0.0987 (0.0034, 0.0009)	0.6939 $\pm$ 0.0066 (0.0012, 0.0011)	0.6956 $\pm$ 0.0016 (0.0006, 0.0013)
DS23	0.7135 $\pm$ 0 (0.1205, 0.0079)	0.5902 $\pm$ 0.0262 (0.0063, 0.0175)	0.5417 $\pm$ 0.0231 (0.9354, 0.0114)	0.5203 $\pm$ 0.0269 (0.0326, 0.0105)	0.7112 $\pm$ 0.002 (0.0471, 0.0381)	0.7127 $\pm$ 0.0015 (0.0028, 0.0055)
DS24	0.6897 $\pm$ 0.638228 (0.2216, 0.013)	0.5638 $\pm$ 0.0205 (0.0084, 0.0132)	0.5869 $\pm$ 0.0188 (0.9289, 0.0228)	0.5547 $\pm$ 0.041 (0.0302, 0.0114)	0.6897 $\pm$ 0.0024 (0.0214, 0.0158)	0.6897 $\pm$ 0 (0.003, 0.0053)
DS25	0.8796 $\pm$ 0 (4.8569, 0.2056)	0.768 $\pm$ 0.0075 (0.3164, 0.1835)	0.769 $\pm$ 0.0039 (90.7636, 0.4677)	0.65 $\pm$ 0.0031 (0.6678, 0.1868)	0.8792 $\pm$ 0 (2.2591, 0.478)	0.8789 $\pm$ 0.0003 (0.126, 0.0853)
DS26	0.4905 $\pm$ 0.0165 (0.2725, 0.0253)	0.513 $\pm$ 0.0169 (0.007, 0.014)	0.532 $\pm$ 0.0112 (1.2289, 0.0184)	0.5002 $\pm$ 0.0919 (0.0219, 0.0174)	0.5542 $\pm$ 0.0013 (0.0191, 0.0229)	0.5549 $\pm$ 0.0145 (0.0041, 0.0086)
DS27	0.6713 $\pm$ 0.0003 (178.8672, 14.7217)	0.7157 $\pm$ 0.0025 (18.7071, 19.4843)	0.625 $\pm$ 0.01 (1852.022, 16.0919)	0.6692 $\pm$ 0.0301 (24.5179, 20.869)	0.7352 $\pm$ 0.0033 (96.2685, 54.0958)	0.6712 $\pm$ 0 (12.8824, 9.2946)
DS28	0.7004 $\pm$ 0.0097 (11.2205, 0.6425)	0.7435 $\pm$ 0.0125 (0.651, 0.6195)	0.7387 $\pm$ 0.0136 (96.3938, 0.6292)	0.7224 $\pm$ 0.0112 (0.8466, 0.6227)	0.5547 $\pm$ 0.0037 (2.2633, 1.7395)	0.5684 $\pm$ 0 (0.4134, 0.3167)
DS29	0.6926 $\pm$ 0.009 (0.4202, 0.0742)	0.7411 $\pm$ 0.0084 (0.0372, 0.0457)	0.7543 $\pm$ 0.0079 (1.2486, 0.0561)	0.6904 $\pm$ 0.0151 (0.0863, 0.0453)	0.6945 $\pm$ 0.0002 (0.2298, 0.1049)	0.6945 $\pm$ 0.0002 (0.0004, 0.0019)
DS30	0.572 $\pm$ 0 (1.4868, 0.0907)	0.5046 $\pm$ 0.0181 (0.057, 0.0854)	0.4924 $\pm$ 0.0231 (6.6096, 0.0947)	0.5523 $\pm$ 0.0153 (0.1188, 0.1085)	0.4547 $\pm$ 0.167 (0.0406, 0.0461)	0.5381 $\pm$ 0 (0.3915, 0.2178)
DS31	0.8109 $\pm$ 0.0027 (29.6761, 0.3975)	0.7221 $\pm$ 0.0058 (1.0483, 0.5201)	0.7276 $\pm$ 0.0292 (24.707, 0.4119)	0.6621 $\pm$ 0.0301 (1.3068, 0.398)	0.8428 $\pm$ 0.0004 (0.3674, 0.2273)	0.8423 $\pm$ 0 (4.5426, 1.1037)
DS32	0.7824 $\pm$ 0.3687 (5.8947, 0.0753)	0.7746 $\pm$ 0.0154 (0.0813, 0.0599)	0.7647 $\pm$ 0.0108 (8.1596, 0.0676)	0.5182 $\pm$ 0.1533 (0.2428, 0.0537)	0.9681 $\pm$ 0 (0.0463, 0.0413)	0.9681 $\pm$ 0 (0.4046, 0.1095)
DS33	0.6179 $\pm$ 0.0018 (4.9689, 0.1718)	0.5249 $\pm$ 0.0129 (0.1878, 0.1753)	0.5939 $\pm$ 0.0052 (19.9818, 0.1756)	0.5515 $\pm$ 0.0119 (0.3195, 0.2077)	0.6361 $\pm$ 0.0009 (0.0905, 0.1095)	0.636 $\pm$ 0.0012 (1.1648, 0.6311)
Average G-mean	0.6539	0.5822	0.5868	0.5757	0.6823	0.6876
Average Rank	3	4.27	4.48	4.67	2.42	2.15

framework contributes significantly to improved performance. Moreover, the rank-based evaluation provides a more robust comparison, accounting for the variance in model performance across datasets. Thus, the experiments infer that the proposed SRLS-1SVM and RLS-1SVM models demonstrate strong overall performance, making them viable options for OCC tasks.

Table 6.3: Win-tie-loss of the baseline column models with respect to the proposed RLS-1SVM and SRLS-1SVM across UCI datasets. The representation is in the form of (win,tie,loss).

Proposed model	OCSVM [60]	LS-OCSVM [80]	ROCSVM-RHHQ [177]	MKCLS-OCSVM [178]
RLS-1SVM	(18,3,12)	(28,0,5)	(29,0,4)	(28,0,5)
SRLS-1SVM	(18,3,12)	(29,0,4)	(31,0,2)	(28,0,5)

6.3.3 Win-tie-loss assessment

To further assess the performance of the models, we analyze the number of wins, ties, and losses of the proposed models in comparison to baseline models. In Table 6.3, we present the counts of wins, ties, and losses (win, tie, loss) for the proposed model against baseline models. The first row of the table corresponds to the comparison with the proposed RLS-1SVM, and the second row corresponds to the comparison with the proposed SRLS-1SVM. Across all baseline models, the number of wins of the proposed models is consistently greater than the number of losses. The proposed RLS-1SVM outperforms OCSVM, LS-OCSVM, ROCSVM-RHHQ, and MKCLS-OCSVM with respective win counts of 18, 28, 29, and 28. Similarly, for SRLS-1SVM, the number of wins against the same models, in the given order, are 18, 29, 31, and 28, respectively. This indicates that the proposed RLS-1SVM and SRLS-1SVM exhibit superior performance compared to the baseline models.

Throughout all datasets, OCSVM and ROCSVM-RHHQ exhibit comparatively high training times, primarily due to solving the QPP. Conversely, the least squares variant models, namely LS-OCSVM, MKCLS-OCSVM, and the proposed RLS-1SVM and SRLS-

1SVM, demonstrate reduced training times as they involve solving linear systems of equations. Furthermore, OCSVM and ROCSVM-RHHQ showcase relatively lower testing times owing to the sparseness of the model. In contrast, LS-OCSVM, MKCLS-OCSVM, and the proposed RLS-1SVM exhibit higher testing times. The testing time of the proposed RLS-1SVM is notably high, attributed to doubling the number of variables in its formulation and its non-sparse nature. However, by enhancing the sparseness of the proposed RLS-1SVM, the proposed SRLS-1SVM delivers improved performance with enhanced training and testing times compared to the baseline models as well as the proposed RLS-1SVM.

6.3.4 Impact of rank parameter r on proposed SRLS-1SVM

To assess the sensitivity of the rank parameter r in the proposed SRLS-1SVM model, we evaluated its performance by setting r to 5%, 10%, 15%, 20%, 25%, and 30% of the total sample size. Figure 6.1 presents the G-mean performance results for four benchmark datasets: `breast_cancer`, `congressional_voting`, `haberman_survival`, and `ilpd_indian_liver` with varying r . As shown, the G-mean values exhibit only slight variation as r changes across datasets, indicating that SRLS-1SVM is robust to different rank parameter settings considered. This analysis suggests that the choice of r has minimal influence on the model's overall performance across diverse datasets.

6.3.5 Performance of the models in the presence of outliers

Figure 6.2 demonstrates the relationship between testing performance and varying ratios of outliers in the training data for several baseline models, including the proposed RLS-1SVM. The analysis spans four benchmark UCI datasets: `breast_cancer`, `congressional_voting`, `haberman_survival`, and `ilpd_indian_liver`, with outlier ratios incrementally set from 5% to 30% in 5% steps, following the methodology in [158].

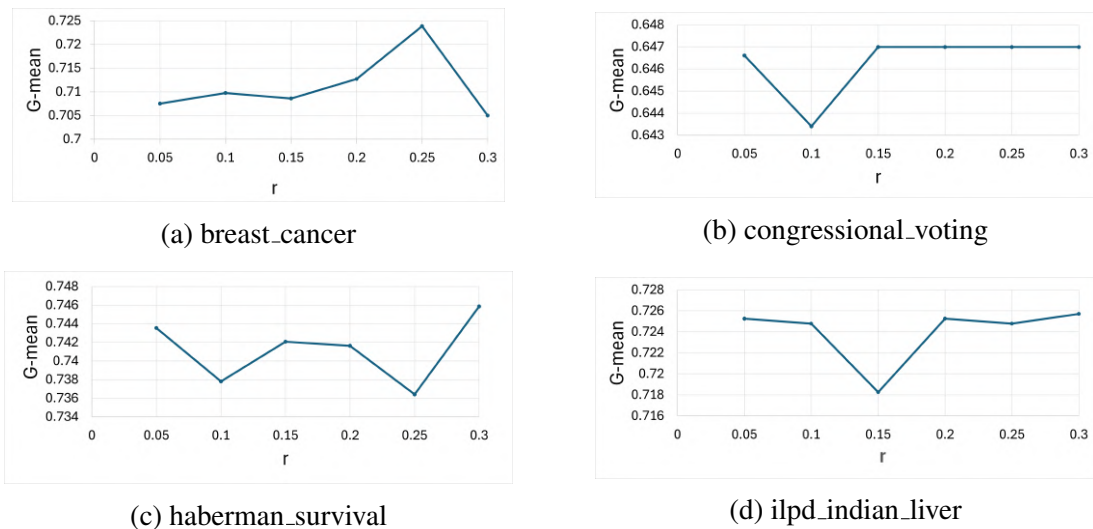


Figure 6.1: Sensitivity analysis of the proposed SRLS-1SVM by varying the rank parameter r across four benchmark datasets. The datasets include (a) breast_cancer (b) congressional_voting (c) haberman_survival (d) ilpd_indian_liver.

The figures provide insights into the resilience of each model in handling outliers across different datasets. As observed from the performance curves, the proposed RLS-1SVM model consistently shows the most robust performance, maintaining stability and performance as outlier levels increase. Notably, the proposed RLS-1SVM's stability is evident across datasets, where it effectively minimizes the impact of outliers on model performance. While the baseline OCSVM model performs comparable to the proposed RLS-1SVM in outlier tolerance, RLS-1SVM's performance remains slightly superior, suggesting that it manages to generalize better despite the presence of significant noise in the training data.

These trends collectively emphasize the advantage of the RLS-1SVM model in applications where data contamination by outliers is a concern, making it a more robust choice than the other tested baseline models.

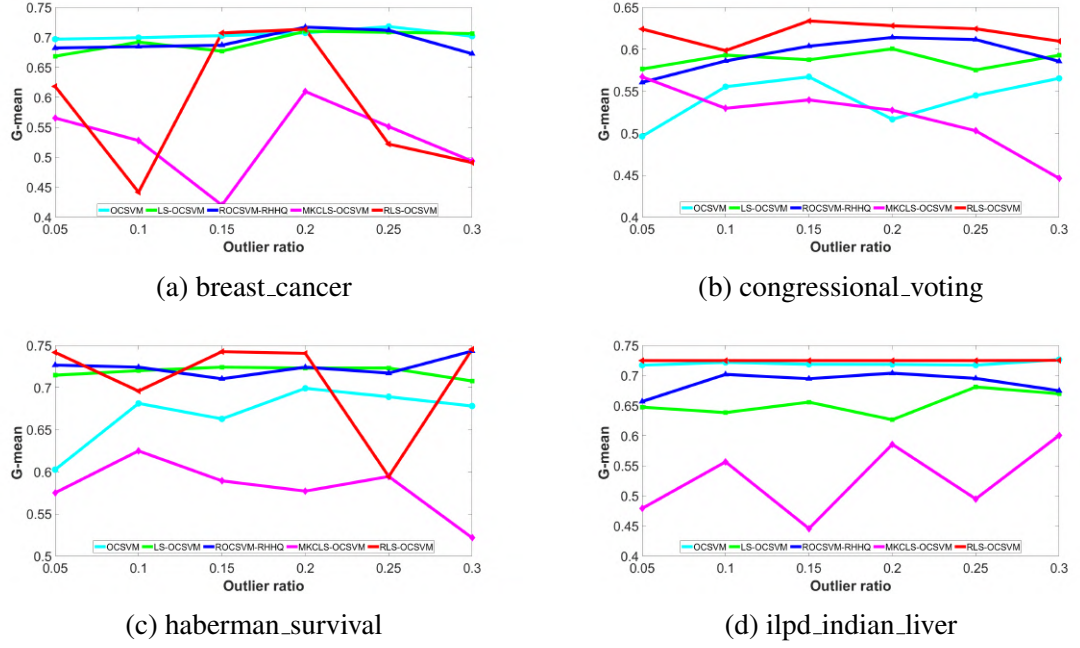


Figure 6.2: The testing performances of the four baseline models and proposed RLS-1SVM by varying ratios of outliers occupying the entire training set across four benchmark datasets. The datasets include (a) breast_cancer (b) congressional_voting (c) haberman_survival (d) ilpd_indian_liver.

6.4 Statistical analysis on real-world data

In this section, we discuss various statistical tests employed to strengthen the analysis. We conducted the Friedman test, Nemenyi post-hoc test and Wilcoxon signed rank test. Friedman's test has a null hypothesis assumption of equivalence among the models. The statistic of Friedman is calculated as shown in equation (2.56). Further, F_F statistic is calculated to overcome the conservative nature of the Friedman statistic. The equation (2.57) is utilized to calculate F_F statistic. Moreover, we also employ the pairwise Nemenyi-post-hoc test which calculates the critical difference (cd) between each pair of models using equation (2.58). If the calculated cd is greater than the difference in average ranks of the concerned models, then the models are significantly different.

For the case of UCI datasets, we have $N = 33$ and $md = 6$, which results in $\chi_F^2 =$

Table 6.4: Nemenyi post-hoc significant difference between the baseline column models and the proposed RLS-1SVM and SRLS-1SVM across UCI datasets.

	OCSVM [60]	LS-OCSVM [80]	ROCSVM-RHHQ [177]	MKCLS-OCSVM [178]
Proposed RLS-1SVM	No	Yes	Yes	Yes
Proposed SRLS-1SVM	No	Yes	Yes	Yes

57.4304, and $F_F = 17.0845$ with degree of freedom 5, and (5, 160), respectively. At a significance level of 5%, the calculated value of $F_{(5,160)} = 2.2707$ which is less than the critical value F_F , thereby leading to the rejection of the null hypothesis. This rejection implies the non-equivalence of the models. Moving to the pairwise Nemenyi test, we obtain $cd = 1.3126$ with the level of significance 5% and $q_{0.05} = 2.850$ [123]. Table 6.4 represents models which are significantly different in comparison to the proposed RLS-1SVM and SRLS-1SVM. While the baseline model, OCSVM, does not exhibit statistically significant differences compared to the proposed models in the Nemenyi post-hoc test, other analyses and numerical results consistently highlight the superior performance of proposed models over OCSVM.

Table 6.5: Wilcoxon signed rank test of the column models with respect to the proposed RLS-1SVM and SRLS-1SVM across UCI datasets.

Model	Proposed RLS-1SVM				Proposed SRLS-1SVM			
	R^+	R^-	p -value	Null hypothesis	R^+	R^-	p -value	Null hypothesis
OCSVM [60]	330.5	134.5	0.04338	rejected	335	130	0.03486	rejected
LS-OCSVM [80]	515	46	< 0.00001	rejected	520	41	< 0.00001	rejected
ROCSVM-RHHQ [177]	518	43	< 0.00001	rejected	526	35	< 0.00001	rejected
MKCLS-OCSVM [178]	504	57	0.00006	rejected	517	44	< 0.00001	rejected

Furthermore, we perform the Wilcoxon test to examine the assumption of the null hypothesis, i.e., equivalence of two models. This test assigns ranks to the dissimilarity in performance metrics of two models based on their magnitudes. Representing the sum of positive ranks as R^+ and the sum of negative ranks as R^- , Table 6.5 provides the results of the Wilcoxon signed rank tests comparing the performance of the proposed RLS-1SVM and SRLS-1SVM with baseline algorithms. In instances where the associated p -value for a model is below 0.05, the null hypothesis assumption is rejected. As a result, Table 6.5

indicates that each of the proposed models is not equivalent to the baseline models.

Therefore, the statistical tests provide evidence of the proposed models' superiority over the baseline models, reinforcing the reliability and robustness of our approach. These findings further substantiate the effectiveness of our models, lending additional confidence to their performance across various datasets.

6.5 Application

To further examine the application of the proposed models, we conducted experiments on the CIFAR-10 dataset. The dataset comprises 60,000 color images, each with a resolution of 32x32 pixels, distributed across 10 classes. There are 6,000 images per class, with 50,000 images allocated for training and 10,000 for testing. The 10 classes of the dataset consist of airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

6.5.1 Experimental setup

In the forthcoming experiments, each of the ten class labels is chosen once as the target class, with the remaining nine class labels considered as the non-target class. This process leads to the generation of ten reallocated subclass datasets for OCC. To train the various models, we follow the strategy mentioned in [178], we randomly select 1000 training images from the target class and 200 training images from the non-target class (with reversed labels as target class) in each reallocated subclass dataset to form a training set. Additionally, we randomly choose another set of 1000 training images from the target class and 5000 training images from the non-target class to create a validation set. After obtaining the model for each learning method, its efficacy is assessed on a distinct test set, consisting of all test images from the target class and the initial 1000 test images from the non-target class. This entire process is repeated 30 times to account for inherent randomness. Moreover, the

parameters are chosen by utilizing the grid search in the range: $[10^{-2}, 10^{-1}, \dots, 10^1, 10^2]$ for C_1 , C_2 , C , and $[10^2, 10^3, \dots, 10^7, 10^8]$ for σ . Other parameters have the same range as mentioned in Section 6.3.

Table 6.6: Average G-mean with standard deviation, training (Tr) time and testing (Ts) time (in seconds) of the proposed RLS-1SVM, SRLS-1SVM, and the baseline models on CIFAR-10 dataset (bold values represent the best G-mean values).

Dataset	OCSVM [60] G-mean $\pm$ std (Tr time, Ts time)	LS-OCSVM [80] G-mean $\pm$ std (Tr time, Ts time)	ROCSVM-RHHQ [177] G-mean $\pm$ std (Tr time, Ts time)	MKCLS-OCSVM [178] G-mean $\pm$ std (Tr time, Ts time)	proposed RLS-1SVM G-mean $\pm$ std (Tr time, Ts time)	proposed SRLS-1SVM G-mean $\pm$ std (Tr time, Ts time)
airplane	0.4995 $\pm$ 0.2936 (0.666, 4.1319)	0.4946 $\pm$ 0.0044 (0.0386, 5.2561)	0.5952 $\pm$ 0.0104 (72.5792, 4.7355)	0.4761 $\pm$ 0.0632 (0.7478, 5.1111)	0.7068 $\pm$ 0.0006 (0.1982, 5.3443)	0.7071 $\pm$ 0.0001 (0.1525, 1.1216)
automobile	0.7537 $\pm$ 0.0026 (0.6337, 4.7612)	0.5502 $\pm$ 0.1372 (0.0348, 5.0177)	0.6208 $\pm$ 0.0443 (71.2645, 3.54)	0.625 $\pm$ 0.0307 (0.6756, 4.8584)	0.7071 $\pm$ 0 (0.1993, 5.442)	0.7071 $\pm$ 0 (0.0578, 1.0389)
bird	0.6328 $\pm$ 0.105 (0.5398, 5.4399)	0.6139 $\pm$ 0.0381 (0.0424, 4.8033)	0.5122 $\pm$ 0.1565 (61.0285, 8.491)	0.4446 $\pm$ 0.118 (0.6716, 4.9226)	0.7071 $\pm$ 0.0002 (0.1938, 5.2702)	0.7057 $\pm$ 0.0021 (0.053, 1.053)
cat	0.2568 $\pm$ 0.3184 (0.5213, 4.2152)	0.5789 $\pm$ 0.1047 (0.0341, 4.7566)	0.5566 $\pm$ 0.0402 (91.6418, 4.7694)	0.5362 $\pm$ 0.0124 (0.6969, 4.7678)	0.7071 $\pm$ 0 (0.1988, 5.4849)	0.7071 $\pm$ 0.0001 (0.0626, 1.0206)
deer	0.3781 $\pm$ 0.2305 (0.5797, 4.0549)	0.68 $\pm$ 0.0306 (0.0526, 5.4896)	0.5353 $\pm$ 0.0377 (61.0672, 4.1877)	0.4107 $\pm$ 0.1288 (0.6946, 4.9058)	0.7242 $\pm$ 0.0031 (0.191, 5.398)	0.7105 $\pm$ 0.0019 (0.0781, 1.0343)
dog	0.5736 $\pm$ 0.2312 (0.5466, 3.9081)	0.5628 $\pm$ 0.1179 (0.0287, 5.1743)	0.5717 $\pm$ 0.0837 (43.3611, 4.1464)	0.5838 $\pm$ 0.0136 (0.6936, 4.6968)	0.7071 $\pm$ 0 (0.1902, 5.8374)	0.7071 $\pm$ 0 (0.0557, 1.0335)
frog	0.2598 $\pm$ 0.3167 (0.5344, 3.8519)	0.6464 $\pm$ 0.0568 (0.0308, 4.9924)	0.5216 $\pm$ 0.0942 (65.5769, 4.8948)	0.4296 $\pm$ 0.0984 (0.6785, 4.8793)	0.6785 $\pm$ 0.0024 (0.1894, 5.3493)	0.7057 $\pm$ 0.0016 (0.0611, 1.0498)
horse	0.6278 $\pm$ 0.1433 (0.5148, 3.7753)	0.5555 $\pm$ 0.0759 (0.0472, 4.8724)	0.5931 $\pm$ 0.1252 (58.5924, 4.1459)	0.4415 $\pm$ 0.1308 (0.6908, 4.9147)	0.7071 $\pm$ 0 (0.1856, 5.421)	0.7071 $\pm$ 0.0001 (0.0552, 1.0399)
ship	0.4021 $\pm$ 0.2784 (0.5921, 4.0781)	0.5284 $\pm$ 0.0598 (0.0296, 5.0358)	0.5934 $\pm$ 0.0344 (61.6087, 4.6947)	0.5058 $\pm$ 0.069 (0.6856, 4.9565)	0.7072 $\pm$ 0.0005 (0.1873, 5.319)	0.7072 $\pm$ 0.0003 (0.0599, 1.0598)
truck	0.6991 $\pm$ 0.0888 (0.5682, 3.9443)	0.4437 $\pm$ 0.0707 (0.043, 4.9578)	0.6247 $\pm$ 0.0925 (65.8867, 4.329)	0.6217 $\pm$ 0.0088 (0.6864, 4.914)	0.7075 $\pm$ 0.0006 (0.19, 5.5088)	0.7071 $\pm$ 0 (0.0761, 1.0139)
Average G-mean	0.5083	0.5654	0.5725	0.5075	0.706	0.7072
Average Rank	4.2	4.5	4.1	5	1.55	1.65

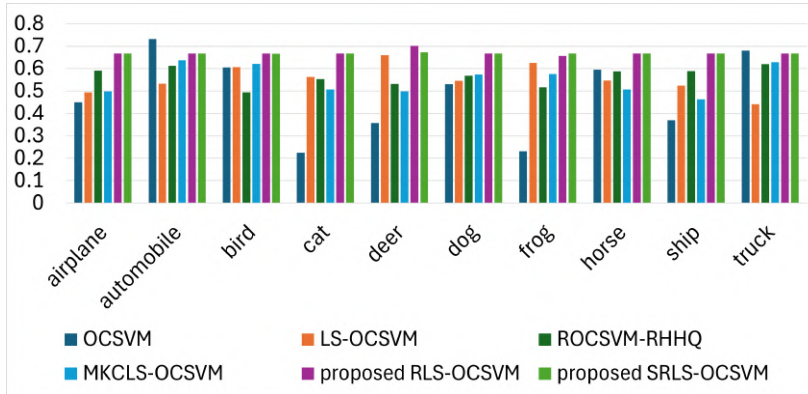


Figure 6.3: F-measure representation of models over different cases of CIFAR-10 dataset.

6.5.2 Experimental and statistical analysis

We report the average G-mean values over 30 executions across different classes in Table 6.6. The average of G-mean values reported in the Table across all 10 cases for the proposed

SRLS-1SVM is 0.7072, which is the highest, followed by the proposed RLS-1SVM having an average G-mean of 0.706. The remaining baseline models have average G-mean values of 0.5725 for ROCSVM-RHHQ, 0.5654 for LS-OCSVM, followed by OCSVM with an average G-mean of 0.5083, and MKCLS-OCSVM having average G-mean of 0.5075. Moreover, we calculate the average ranks, which is 1.55 for the proposed RLS-1SVM, second best for the proposed SRLS-1SVM which is 1.65, followed by the baseline models in the order of increasing rank as ROCSVM-RHHQ, OCSVM, LS-OCSVM, and MKCLS-OCSVM with rank 4.1, 4.2, 4.5, and 5, respectively. Furthermore, Table 6.7 displays the performance of models across 10 cases of CIFAR-10 in terms of count on the number of wins, ties, and losses with respect to the baseline models. The number of wins for both the proposed models is greater than the baseline models.

For better visualisation, we compared the average F-measure values of 30 iterations for 10 cases of the CIFAR-10 dataset reflected in Figure 6.3, which illustrates the proposed RLS-1SVM and SRLS-1SVM have better F-measure values across most of the cases except automobile and truck being the target data.

Table 6.7: Win-tie-loss of the baseline column models with respect to the proposed RLS-1SVM and SRLS-1SVM over CIFAR-10 dataset.

Proposed model	OCSVM [60]	LS-OCSVM [80]	ROCSVM-RHHQ [177]	MKCLS-OCSVM [178]
RLS-1SVM	(9,0,1)	(10,0,0)	(10,0,0)	(10,0,0)
SRLS-1SVM	(9,0,1)	(10,0,0)	(10,0,0)	(10,0,0)

Table 6.8: Nemenyi post-hoc significant difference between the baseline column models and the proposed RLS-1SVM and SRLS-1SVM over CIFAR-10 dataset.

	OCSVM [60]	LS-OCSVM [80]	ROCSVM-RHHQ [177]	MKCLS-OCSVM [178]
Proposed RLS-1SVM	Yes	Yes	Yes	Yes
Proposed SRLS-1SVM	Yes	Yes	Yes	Yes

We conducted a series of statistical tests namely, the Friedman test, Nemenyi post-hoc test, and Wilcoxon signed-rank test on numerical results from 10 cases involving CIFAR-10 dataset. With parameters $md = 6$ and $N = 10$, and utilizing the corresponding relations, we

Table 6.9: Wilcoxon signed rank test of the baseline column models with respect to the proposed RLS-1SVM and SRLS-1SVM over the CIFAR-10 dataset.

Model	Proposed RLS-1SVM				Proposed SRLS-1SVM			
	R^+	R^-	p -value	Null hypothesis	R^+	R^-	p -value	Null hypothesis
OCSVM [60]	53	2	0.00932	rejected	53	2	0.00932	rejected
LS-OCSVM [80]	55	0	0.00512	rejected	55	0	0.00512	rejected
ROCSVM-RHHQ [177]	55	0	0.00512	rejected	55	0	0.00512	rejected
MKCLS-OCSVM [178]	55	0	0.00512	rejected	55	0	0.00512	rejected

obtain $\chi_F^2 = 32.3571$, $F_F = 16.5061$, and $cd = 2.3845$. The degrees of freedom for χ_F^2 and F_F are 5 and $(5, 45)$, respectively. With $F_{(5,45)} = 2.422 < F_F$, we reject the null hypothesis, indicating that the proposed RLS-1SVM is non-equivalent to baseline algorithms. Further, Table 6.8 presents statistical differences between the proposed models and baseline models, revealing that the test holds for all the baseline models with respect to the proposed RLS-1SVM and SRLS-1SVM. Further, the results of the Wilcoxon-signed rank test depicted in Table 6.9 indicate the assumption of the null hypothesis is rejected. In summary, experimental results and statistical analysis infer that the proposed RLS-1SVM and SRLS-1SVM outperform the baseline algorithms in terms of generalization performance. Additionally, as a result of the sparseness of the proposed SRLS-1SVM, it has the least training and testing times compared to other models.

6.6 Summary

This chapter introduces a robust and scalable framework for OCC, addressing the challenges of noise, outliers, and scalability inherent in traditional models. Two novel models are proposed: robust least squares one-class support vector machine (RLS-1SVM) and sparse robust least squares one-class support vector machine (SRLS-1SVM). The RLS-1SVM enhances robustness by incorporating distributional information of modeling errors, allowing it to detect anomalies in complex non-linear systems while mitigating the impact of noise and outliers. However, the increased number of variables and its non-sparse for-

mulation result in longer training and testing times, limiting its scalability. To overcome these computational inefficiencies, the SRLS-1SVM is introduced as the first sparse variant of LS-OCSVM for batch learning. By employing pivoted Cholesky decomposition to approximate the kernel matrix, SRLS-1SVM significantly reduces computational complexity, improving scalability and efficiency without sacrificing performance.

The chapter also explores the theoretical properties of both models, providing an upper bound for their generalization error. Extensive experiments and statistical analyses on UCI and CIFAR-10 datasets demonstrate the superiority of the proposed models. The RLS-1SVM excels in robustness against noise and outliers, while the SRLS-1SVM delivers efficient, scalable performance with reduced training and testing times. This framework provides a benchmark in robust and scalable OCC, offering significant advancements in anomaly detection for large, noisy datasets.

Looking ahead, the next chapter shifts focus to regression tasks, where models aim to learn relationships in continuous data. While OCC deals with anomaly detection, regression models often face challenges arising from incomplete information. The upcoming chapter introduces the LUPI paradigm, which enriches regression models by incorporating additional information during training. By expanding the capabilities of traditional regression, this approach aims to improve model performance and generalization, offering new insights and methods for handling complex predictive tasks.

Chapter 7

Leveraging Privileged Information in Regression Model

This chapter explores the integration of the LUPI paradigm [70] into regression models, highlighting its potential to enhance traditional regression frameworks by incorporating additional knowledge available during training. This mirrors an educational setting, where a teacher provides crucial guidance that helps students learn more effectively.

Utilizing the concept of LUPI in regression, we propose a refined framework that incorporates PI into the learning process. The framework builds upon the SRM principle and provides a generalization error bound based on Rademacher complexity, ensuring theoretical soundness and reliability. By embedding PI into regression model, this work advances ML methodologies, aligning them with the contextual and adaptive aspects of human learning to achieve improved generalization and decision-making capabilities.

7.1 Proposed work

The conventional LSTSVR [67] relies solely on the training features for learning. However, learning in the presence of PI enhances the generalization performance of the model.

The LUPI paradigm leverages additional information during the training phase. Building upon this concept and integrating PI into LSTSVR, in this section, we propose least square twin support vector regression with privileged information (LSTSVR+). We thoroughly discuss the linear and non-linear variants of the proposed LSTSVR+. In accordance with [70], the slack variable (ζ) is formed as the correcting function for incorporating PI into the proposed optimization problems. Further, following [180], we minimize ζ instead of ζ^2 for ease in the incorporation of LUPI paradigm. Throughout this chapter, x_i^* denotes the PI corresponding to the i^{th} data sample.

7.1.1 Proposed LSTSVR+: linear case

In this subsection, we discuss the linear case of the proposed LSTSVR+. The equations of two ϵ -insensitive proximal linear functions are given as:

$$r_1(x) = u_1^t x + b_1 \text{ and } r_2(x) = u_2^t x + b_2,$$

and correction functions of each linear function are given as:

$$p_1(x^*) = u_1^{*t} x^* + b_1^* \text{ and } p_2(x^*) = u_2^{*t} x^* + b_2^*,$$

respectively. Here, $u_i, u_i^* \in \mathbb{R}^n$ and $b_i, b_i^* \in \mathbb{R}$ for $(i = 1, 2)$. The optimization problems for the linear case can be expressed as:

$$\begin{aligned} \min_{u_1, b_1, u_1^*, b_1^*} & \frac{C_1}{2} (\|u_1\|^2 + b_1^2) + \frac{C_2}{2} (\|u_1^*\|^2 + b_1^{*2}) \\ & + \frac{1}{2} \|Y - (Mu_1 + eb_1)\|^2 + C_3 e^t (M^* u_1^* + eb_1^*) \\ \text{such that } & Y - (Mu_1 + eb_1) = -\epsilon_1 e - (M^* u_1^* + eb_1^*) \end{aligned} \quad (7.1)$$

and

$$\begin{aligned} \min_{u_2, b_2, u_2^*, b_2^*} & \frac{C_4}{2}(u_2^t u_2 + b_2^2) + \frac{C_5}{2}(u_2^{*t} u_2^* + b_2^{*2}) \\ & + \frac{1}{2} \|((Mu_2 + eb_2) - Y)\|^2 + C_6 e^t (M^* u_2^* + eb_2^*) \\ \text{such that } & (Mu_2 + eb_2) - Y = -\epsilon_2 e - (M^* u_2^* + eb_2^*). \end{aligned} \quad (7.2)$$

Here, C_i ($i = 1, 2, 3, 4, 5, 6$) corresponds to the positive regularization parameters. e signifies a column vector of the appropriate dimension. The first term in the optimization problem (7.1) is the regularization term corresponding to the first proximal linear function; the second term leads to the regularization term corresponding to the first correcting function; the third term signifies the sum of the squared distance between the estimated function and the actual targets. The fourth term corresponds to the correcting function values using the PI of data. The regularization terms in the optimization problem (7.1) result in the flatness of the function $r_1(x)$ and $p_1(x^*)$. The significance of the terms in the optimization problem (7.2) can be explained similarly.

In order to solve the problems (7.1) and (7.2), convert them to their dual forms using the Lagrangian function. For problem (7.1), it can be expressed as:

$$\begin{aligned} \mathcal{L} = & \frac{C_1}{2}(\|u_1\|^2 + b_1^2) + \frac{C_2}{2}(\|u_1^*\|^2 + b_1^{*2}) + \frac{1}{2} \|Y - (Mu_1 + eb_1)\|^2 \\ & + C_3 e^t (M^* u_1^* + eb_1^*) + \alpha^t (Y - (Mu_1 + eb_1) + \epsilon_1 e + (M^* u_1^* + eb_1^*)), \end{aligned} \quad (7.3)$$

where α corresponds to the Lagrangian multipliers. Using the Karush Kuhn Tucker (K.K.T.) conditions, differentiating equation (7.3) with respect to u_1, b_1, u_1^*, b_1^* , we get:

$$\frac{\partial \mathcal{L}}{\partial u_1} = C_1 u_1 - M^t (Y - (Mu_1 + eb_1)) - M^t \alpha = 0, \quad (7.4)$$

$$\frac{\partial \mathcal{L}}{\partial b_1} = C_1 b_1 - e^t (Y - (Mu_1 + eb_1)) - e^t \alpha = 0, \quad (7.5)$$

$$\frac{\partial \mathcal{L}}{\partial u_1^*} = C_2 u_1^* + C_3 M^{*t} e + M^{*t} \alpha = 0, \quad (7.6)$$

$$\frac{\partial \mathcal{L}}{\partial b_1^*} = C_2 b_1^* + C_3 e^t e - e^t \alpha = 0. \quad (7.7)$$

Combining equations (7.4) and (7.5), we obtain

$$C_1 v_1 - G^t(Y - G v_1) - G^t \alpha = 0, \quad (7.8)$$

where $v_1 = \begin{bmatrix} u_1 \\ b_1 \end{bmatrix}$ and $G = \begin{bmatrix} M & e \end{bmatrix}$. Similarly, combining equations (7.6) and (7.7), gives

$$C_2 v_1^* + C_3 G^{*t} e + G^{*t} \alpha = 0, \quad (7.9)$$

where $v_1^* = \begin{bmatrix} u_1^* \\ b_1^* \end{bmatrix}$ and $G^* = \begin{bmatrix} M^* & e \end{bmatrix}$. From the constraint of the optimization problem (7.1), we have

$$Y - G v_1 + \epsilon_1 e + G^* v_1^* = 0. \quad (7.10)$$

Solving equations (7.8), (7.9) and (7.10), we obtain

$$\begin{aligned} \alpha = & \left(G G^t + \frac{C_1}{C_2} G^* G^{*t} + \frac{1}{C_2} G G^t G^* G^{*t} \right)^{-1} [C_1 Y + C_1 \epsilon_1 e \\ & - \frac{C_1 C_3}{C_2} G^* G^{*t} e + G G^t \epsilon_1 e - \frac{C_3}{C_2} G G^t G^* G^{*t} e]. \end{aligned}$$

Following the same procedure as for optimization problem (7.2) gives following equations:

$$C_4 v_2 + G^t(G v_2 - Y) + G^t \beta = 0, \quad (7.11)$$

$$C_5 v_2^* + C_6 G^{*t} e + G^{*t} \beta = 0, \quad (7.12)$$

$$G v_2 - Y + \epsilon_2 e + G^* v_2^* = 0, \quad (7.13)$$

where β corresponds to the Lagrangian multiplier. From equations (7.11)-(7.13), we get

$$\begin{aligned} \beta = & (GG^t + \frac{C_4}{C_5}G^*G^{*t} + \frac{1}{C_5}GG^tG^*G^{*t})^{-1}(-C_4Y + C_4\epsilon_2e \\ & - \frac{C_4C_6}{C_5}G^*G^{*t}e + GG^t\epsilon_2e - \frac{C_6}{C_5}GG^tG^*G^{*t}e). \end{aligned} \quad (7.14)$$

Further, the parameters of the hyperplanes $r_1(x)$ and $r_2(x)$ obtained using equations (7.8) and (7.11) are as follows:

$$v_1 = (G^tG + C_1I)^{-1}G^t(Y + \alpha), \text{ and } v_2 = (G^tG + C_4I)^{-1}G^t(Y - \beta),$$

respectively. The target value corresponding to an unknown sample is given by the function:

$$r(x) = \frac{1}{2}(r_1(x) + r_2(x)) = \frac{1}{2}x^t(u_1 + u_2) + \frac{1}{2}(b_1 + b_2).$$

While the linear variant of LSTSVR+ offers a conceptually elegant solution by leveraging privileged information within a least-squares twin support vector regression framework, real-world data often exhibit non-linear patterns that cannot be captured adequately by linear models. These patterns are particularly prominent in datasets where the relationship between input features and outputs is inherently complex. To address this, we extend the linear formulation to a non-linear one using kernel functions, allowing the model to operate effectively in high-dimensional feature spaces via the kernel trick. This transition preserves the computational advantages of the linear model namely, solving systems of linear equations instead of quadratic programming problems, while significantly enhancing flexibility and expressiveness. Hence, this extension is not merely a model variant but a necessary progression to ensure broader applicability and stronger performance on non-linear tasks.

7.1.2 Proposed LSTSVR+: Non-Linear case

The presence of inseparable data points, which can occur when linear separation is not feasible, can be addressed using the proposed non-linear LSTSVR+. The ϵ_1 -insensitive down bound and ϵ_2 -insensitive up bound regressor for non-linear cases are given by:

$$\tilde{r}_1(x) = K(x^t, M^t)u_1 + b_1 \text{ and } \tilde{r}_2(x) = K(x^t, M^t)u_2 + b_2,$$

respectively. The non-linear correcting function for each hyperplane is given as:

$$\tilde{p}_1(x^*) = K(x^{*t}, M^{*t})u_1^* + b_1^* \text{ and } \tilde{p}_2(x^*) = K(x^{*t}, M^{*t})u_2^* + b_2^*, \quad (7.15)$$

respectively. As discussed for the linear case, the optimization problems for the proposed LSTSVR+ are written as:

$$\begin{aligned} \min_{u_1, b_1, u_1^*, b_1^*} & \frac{C_1}{2}(\|u_1\|^2 + b_1^2) + \frac{C_2}{2}(\|u_1^*\|^2 + b_1^{*2}) \\ & + \frac{1}{2}\|Y - (K(M, M^t)u_1 + eb_1)\|^2 + C_3e^t(K(M^*, M^{*t})u_1^* + eb_1^*) \\ \text{such that } & Y - (K(M, M^t)u_1 + eb_1) = -\epsilon_1e - (K(M^*, M^{*t})u_1^* + eb_1^*), \end{aligned} \quad (7.16)$$

and

$$\begin{aligned} \min_{u_2, b_2, u_2^*, b_2^*} & \frac{C_4}{2}(\|u_2\|^2 + b_2^2) + \frac{C_5}{2}(\|u_2^*\|^2 + b_2^{*2}) + \frac{1}{2}\|(K(M, M^t)u_2 + eb_2) - Y\| \\ & + C_6e^t(K(M^*, M^{*t})u_2^* + eb_2^*) \\ \text{such that } & (K(M, M^t)u_2 + eb_2) - Y = -\epsilon_2e - (K(M^*, M^{*t})u_2^* + eb_2^*). \end{aligned} \quad (7.17)$$

The significance of the terms in the primal problems (7.16) and (7.17) follows in a similar way as in the linear case. The Lagrangian function for problem (7.16) is as follows:

$$\begin{aligned} \mathcal{L} = & \frac{C_1}{2}(\|u_1\|^2 + b_1^2) + \frac{C_2}{2}(\|u_1^*\|^2 + b_1^{*2}) + \frac{1}{2}\|Y - (K(M, M^t)u_1 + eb_1)\|^2 + C_3e^t(eb_1^* \\ & + K(M^*, M^{*t})u_1^*) + \gamma^t(Y - (K(M, M^t)u_1 + eb_1) + \epsilon_1e + K(M^*, M^{*t})u_1^* + eb_1^*), \end{aligned} \quad (7.18)$$

where γ is the Lagrangian multiplier. Moving similar to the linear case, from equation (7.18), we obtain:

$$C_1v_1 - G^t(Y - Gv_1) - G^t\gamma = 0, \quad (7.19)$$

$$C_2v_1^* + C_3G^{*t}e + G^{*t}\gamma = 0, \quad (7.20)$$

$$Y - Gv_1 + \epsilon_1e + G^*v_1^* = 0, \quad (7.21)$$

for the optimization problem (7.16). Likewise the following equations are obtained for the optimization problem (7.17):

$$C_4v_2 + G^t(Gv_2 - Y) + G^t\lambda = 0, \quad (7.22)$$

$$C_5v_2^* + C_6G^{*t}e + G^{*t}\lambda = 0, \quad (7.23)$$

$$Gv_2 - Y + \epsilon_2e + G^*v_2^* = 0, \quad (7.24)$$

λ being the Lagrangian multiplier. Using the aforementioned equations, (7.19)-(7.21), we obtain

$$\begin{aligned} \gamma = & (GG^t + \frac{C_1}{C_2}G^*G^{*t} + \frac{1}{C_2}GG^tG^*G^{*t})^{-1}[C_1Y + C_1\epsilon_1e \\ & - \frac{C_1C_3}{C_2}G^*G^{*t}e + GG^t\epsilon_1e - \frac{C_3}{C_2}GG^tG^*G^{*t}e] \end{aligned} \quad (7.25)$$

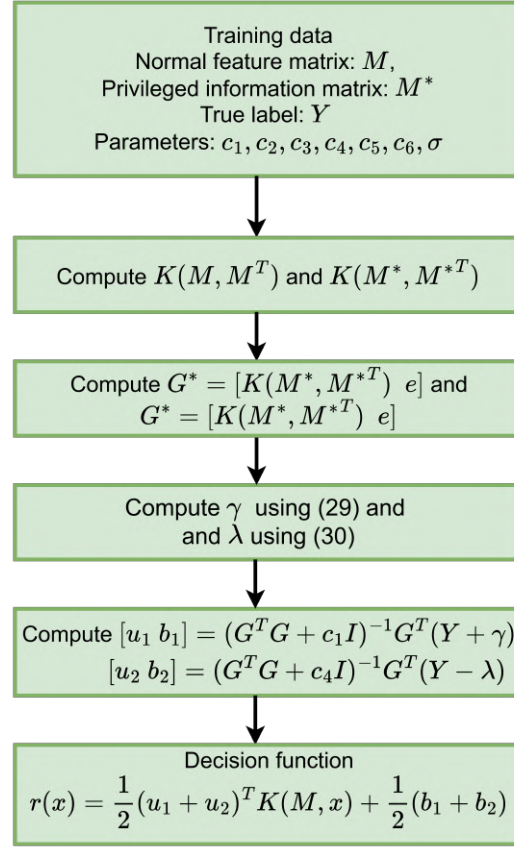


Figure 7.1: Graphical view to represent the flow of algorithm for the non-linear proposed LSTSVR+.

and by utilizing equations (7.22)-(7.24), we get,

$$\begin{aligned} \lambda = & (GG^t + \frac{C_4}{C_5} G^* G^{*t} + \frac{1}{C_5} GG^t G^* G^{*t})^{-1} (-C_4 Y + C_4 \epsilon_2 e \\ & - \frac{C_4 C_6}{C_5} G^* G^{*t} e + GG^t \epsilon_2 e - \frac{C_6}{C_5} GG^t G^* G^{*t} e) \end{aligned} \quad (7.26)$$

with $G = [K(M, M^t) \ e]$ and $G^* = [K(M^*, M^{*t}) \ e]$. Other notations have the same meaning as defined for the linear case.

Using γ and λ , the parameters of the hyperplane $\tilde{r}_1(x)$ and $\tilde{r}_2(x)$ are obtained. The target value of the unknown samples is given by the following function:

$$r(x) = \frac{1}{2} (r_1(x) + r_2(x)) = \frac{1}{2} (u_1 + u_2)^t K(M, x) + \frac{1}{2} (b_1 + b_2).$$

The optimization problems for the linear and non-linear cases of the proposed LSTSVR+ lead to the solution of system of linear equations, which has comparatively improved efficiency as compared to solving QPPs. Figure 7.1 represents the graphical view demonstrating the algorithmic flow of the proposed LSTSVR+.

7.1.3 Complexity

SVR+ solves QPP which has a complexity of $O(l^3)$, where l denote the number of samples. The incorporation of PI introduces additional computations, augmenting the time complexity in both SVR+ and the proposed LSTSVR+. However, unlike SVR+, which solves a QPP, the proposed LSTSVR+ solves a system of equations, leading to improved efficiency. The practical efficiency is improved due to the simpler nature of linear systems and the reduced reliance on optimization toolboxes. Incorporating PI does add some computational overhead, but this is manageable and does not significantly alter the overall complexity. Furthermore, the space complexity of the proposed LSTSVR+ is mainly due to storing matrices G and G^* which is $O(l^2)$.

7.1.4 Comparison between SVR+ and the proposed LSTVR-PI

The relationship between the proposed LSTSVR+ and SVR+ can be analyzed from various perspectives, including their underlying principles, computational complexity, and practical implications.

[1] Underlying principles

- Both LSTSVR+ and SVR+ aim to enhance the performance of SVR by incorporating PI during training.
- Both utilize the concept of correcting functions derived from privileged data to

improve the regression model's accuracy.

- Both algorithms adhere to the SRM principle by incorporating regularization terms in their optimization problems to prevent overfitting.

[2] Computational complexity

- SVR+ employs two correcting hyperplanes within the same QPP, leading to high computational complexity which gets enhanced in the presence of extra information. In contrast, LSTSVR+ constructs separate correcting hyperplanes for each optimization problem, resulting in a more manageable computational burden.
- LSTSVR+ further reduces computational complexity by solving systems of linear equations instead of QPPs, which is the approach adopted by SVR+.

[3] Practical implications:

- While SVR+ may offer superior performance under certain conditions, its high computational complexity limits its scalability. While LSTSVR+ provides a more computationally efficient alternative by leveraging separate correcting hyperplanes and solving systems of linear equations, making it more practical for real-world applications.

Hence, while both LSTSVR+ and SVR+ share the objective of enhancing SVR performance through PI, LSTSVR+ offers a more computationally efficient approach by employing separate correcting hyperplanes and solving systems of linear equations, thus addressing the complexity challenges associated with SVR+.

7.1.5 Impact of PI in proposed LSTSVR+

The performance of the proposed LSTSVR+ is influenced by two factors related to PI. First, the method used to acquire PI can impact performance. Second, the nature of the PI itself also plays a role. Here, we examine the effect of both factors on the model's performance.

Importance of PI acquisition: The method and quality of PI acquisition can significantly impact the performance of the algorithm. The relevance of PI to the target task is crucial; highly relevant PI that provides valuable insights into the data's underlying patterns can enhance the algorithm's performance, whereas irrelevant or noisy PI can lead to suboptimal results or even degrade performance. Additionally, the completeness and quality of PI play a vital role. When PI captures essential aspects of the data not present in the regular features, it can improve model generalization and performance. Conversely, incomplete or low-quality PI may fail to provide meaningful insights and could hinder the algorithm's effectiveness. Furthermore, the method used to integrate PI into the algorithm also affects performance. Effective integration methods that meaningfully combine regular features and PI can lead to improved model performance. However, inappropriate or ineffective integration methods may not fully leverage the advantages of PI, thereby limiting its impact. Overall, different approaches to PI acquisition can indeed influence the performance of the proposed LSTSVR+, making it essential to carefully consider the relevance, completeness, quality, and integration method of PI to maximize its benefits.

Importance of PI: Further, PI plays a crucial role in improving the performance of proposed LSTSVR+ by providing additional insights and context that regular features may not capture. Different ways through which PI contributes to enhancing the performance of the proposed LSTSVR+ are as follows:

- **Enhanced feature representation:** PI often contains valuable features or attributes that are not present in the regular feature set. By incorporating PI into the algorithm, the feature representation becomes more comprehensive and expressive, capturing a broader range of data characteristics.
- **Better generalization:** PI may provide supplementary knowledge or context about the underlying data distribution. By incorporating this additional information, the proposed LSTSVR+ can learn robust and generalized patterns, leading to improved performance on unseen data.
- **Domain-specific knowledge incorporation:** PI often encapsulates domain-specific knowledge or expertise that is valuable for the task at hand. Incorporating the domain knowledge into the proposed LSTSVR+ allows for more informed decision-making and better performance in real-world applications.

Overall, PI plays a critical role in improving algorithm performance by enriching feature representation, promoting better generalization, reducing overfitting, and incorporating domain-specific knowledge. By leveraging PI effectively, proposed LSTSVR+ achieve higher generalization performance.

Theorem 14. *For any given positive C_2, C_3 the solution of the problem (7.1) is always unique.*

Proof. Following [181], we have F as the objective function:

$$F = \frac{1}{2}C_1(\|u_1\|^2 + b_1^2) + \frac{1}{2}C_2(\|u_1^*\|^2 + b_1^{*2}) + \frac{1}{2}\|Y - (Mu_1 + eb_1)\|^2 + C_3e^t(M^*u_1^* + eb_1^*).$$

Define $w = (u_1, b_1, u_1^*, b_1^*)^t$. Assume w_1 and w_2 are two solutions problem (7.1). Due to the

convexity of the problem, a family of the solution is given by $w_t = (1-t)w_1 + tw_2$, $t \in [0, 1]$, and $F(w_1) = F(w_2) = F(w_t)$.

Thus, $F(w_t) - F(w_1) = 0$. Expanding we get,

$$\begin{aligned} & \left[\frac{1}{2}C_1(\|u_t\|^2 + b_t^2) + \frac{1}{2}C_2(\|u_t^*\|^2 + b_t^{*2}) + \frac{1}{2}\|Y - (Mu_t + eb_t)\|^2 \right. \\ & \left. + C_3e^t(M^*u_t^* + eb_t^*) \right] - \left[\frac{1}{2}C_1(\|u_1\|^2 + b_1^2) + \frac{1}{2}C_2(\|u_1^*\|^2 + b_1^{*2}) \right. \\ & \left. + \frac{1}{2}\|Y - (Mu_1 + eb_1)\|^2 + C_3e^t(M^*u_1^* + eb_1^*) \right] = 0. \end{aligned}$$

Differentiating the above equation with respect to t twice, we get

$$\begin{aligned} & C_1\|(u_1 - u_2)\|^2 + C_1(b_1 - b_2)^2 + C_2\|u_1^* - u_2^*\|^2 + C_2(b_1^* - b_2^*)^2 \\ & + \|(Mu_1 - Mu_2)(b_1e - b_2e)\|^2 = 0. \end{aligned}$$

Hence, from the above equation, we get $u_1 = u_2$, $b_1 = b_2$, $u_1^* = u_2^*$, $b_1^* = b_2^*$. □

The same follows for the remaining optimization problems (7.2), (7.16) and (7.17).

7.1.6 Theoretical properties

In this section, we discuss the theoretical properties of the proposed LSTSVR+. We provide a generalization error bound for the proposed LSTSVR+. The bound on generalization error relies on the Rademacher complexity, which is defined as follows:

Definition 7.1.1. [182] For a sample set $S = \{x_1, x_2, \dots, x_m\}$ generated by distribution $\mathcal{D}$ on set X and $\mathcal{F}$ represents a family of functions which maps from X to output space. Then the Rademacher complexity of $\mathcal{F}$ is defined as:

$$\mathcal{R}_m(\mathcal{F}) = \mathbb{E} \left[\frac{1}{l} \sup_{f \in \mathcal{F}} \sum_{i=1}^l \sigma_i f(x_i) \right],$$

where σ_i corresponds to the independent random variable chosen from $\{-1, 1\}$ with equal probability, and are known as Rademacher variables. $\mathbb{E}$ corresponds to the expectation.

Theorem 15. Let $K : X \times X \rightarrow \mathbb{R}$ be a kernel function and $S = \{x_1, x_2, \dots, x_m\}$ be a sample set such that $\sup_{x \in X} K(x, x) < \infty$. Assume $\phi : X \rightarrow H$ be the feature map for the kernel function K . Let $\mathcal{F}$ be the class of functions in kernel space such that

$$\mathcal{F}_{\mathcal{B}} = \{f | f : x \rightarrow \frac{1}{2}(u_1^t x + u_2^t x) : \|u_1\| \leq B, \|u_2\| \leq B\}, B \in \mathbb{R}. \quad (7.27)$$

Then, the Rademacher complexity is given by

$$R_m(\mathcal{F}_{\mathcal{B}}) = \frac{B}{l} \sqrt{\sum_{i=1}^l K(x_i, x_i)}. \quad (7.28)$$

Proof. Using the definition of Rademacher complexity, we have

$$\begin{aligned} \mathcal{R}_m(\mathcal{F}_{\mathcal{B}}) &= \frac{1}{l} \mathbb{E} \left[\sup_{f \in \mathcal{F}_{\mathcal{B}}} \sum_{i=1}^l \sigma_i f(x_i) \right] \\ &= \frac{1}{l} \mathbb{E} \left[\sup_{\|u_1\| \leq B, \|u_2\| \leq B} \sum_{i=1}^l \frac{\sigma_i}{2} (u_1^t \phi(x_i) + u_2^t \phi(x_i)) \right] \\ &\leq \frac{1}{l} \mathbb{E} \left[\sup_{\|u_1\| \leq B} \sum_{i=1}^l \frac{\sigma_i}{2} u_1^t \phi(x_i) + \sup_{\|u_2\| \leq B} \sum_{i=1}^l \frac{\sigma_i}{2} u_2^t \phi(x_i) \right] \\ &= \frac{B}{2l} \mathbb{E} \left[\left\| \sum_{i=1}^l \sigma_i \phi(x_i) \right\|_{\infty} + \left\| \sum_{i=1}^l \sigma_i \phi(x_i) \right\|_{\infty} \right] \quad (\text{using dual norm}) \\ &= \frac{B}{l} \mathbb{E} \left[\left\| \sum_{i=1}^l \sigma_i \phi(x_i) \right\| \right] \quad (\text{using the norm equivalence}) \\ &= \frac{B}{l} \mathbb{E} \left[\sqrt{\left\langle \sum_{i=1}^l \sigma_i \phi(x_i), \sum_{i=1}^l \sigma_i \phi(x_i) \right\rangle} \right] \quad (\text{Cauchy Schwarz inequality}) \\ &= \frac{B}{l} \mathbb{E} \left[\sqrt{\sum_{1 \leq i, j \leq l} \sigma_i \sigma_j \langle \phi(x_i), \phi(x_j) \rangle} \right] \end{aligned}$$

$$\begin{aligned}
 &= \frac{B}{l} \left[\sqrt{\mathbb{E} \left[\sum_{1 \leq i \leq l} \langle \phi(x_i), \phi(x_j) \rangle \right]} \right] \quad (\text{using Jensen's inequality}) \\
 &= \frac{B}{l} \left[\sqrt{\sum_{1 \leq i \leq l} K(x_i, x_i)} \right].
 \end{aligned}$$

□

Lemma 1. *The loss function l in LSTSVR+ exhibits Lipschitz continuity, i.e., there exists a positive Lipschitz constant L , such that, $\forall x, y \in \mathbb{R}^n$*

$$\|\mathcal{L}(x) - \mathcal{L}(y)\| \leq L\|x - y\|. \quad (7.29)$$

Proof. For the two hyperplanes, we have defined the loss function in two different ways: $\mathcal{L}_1(x) = x - \epsilon_1 e$ and $\mathcal{L}_2(x) = x - \epsilon_2 e$. It is straightforward that both $\mathcal{L}_1$ and $\mathcal{L}_2$ are Lipschitz continuous with Lipschitz constant L_1 and L_2 , respectively. Thus, the loss function l of LSTSVR+ is Lipschitz continuous. □

The general formula for the generalization error bound based on the Rademacher complexity can be expressed using the following theorem.

Theorem 16. [182] *Consider $\mathcal{G}(f)$ and $\mathcal{E}(f)$ as the generalization error bound and the empirical error bound, respectively, and $\mathcal{F}$ as a family of functions, with a probability of at least $1 - \delta$ ($\delta \in (0, 1)$) over the m samples and for all $f \in \mathcal{F}$, the Lipschitz continuous loss function l with Lipschitz constant L , the following relation holds.*

$$\mathcal{G}(f) \leq \mathcal{E}(f) + 2L\mathcal{R}_m(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2l}}. \quad (7.30)$$

Using the above theorems, the generalization error of LSTSVR+ is bounded which is given by the following theorem.

Theorem 17. Let $\tilde{\mathcal{H}}$ be a family of functions such that $\tilde{\mathcal{H}} = \{\tilde{h}|\tilde{h} : x \rightarrow f(x) - g(x) + (\epsilon_2 e - \epsilon_1 e)\}$, where $\epsilon_1, \epsilon_2 \in \mathbb{R}$, $f \in \mathcal{F}_B$ and $g(x)$ corresponds to the true label of x . For Lipschitz continuous loss functions $\mathcal{L}$ of proposed LSTSVR+ with Lipschitz constant L , with probability at least $1 - \delta$ ($\delta \in (0, 1)$) over the m samples. The generalization error for all $\tilde{h} \in \tilde{\mathcal{H}}$ is bounded by

$$\mathcal{G}(\tilde{h}) \leq \mathcal{E}(\tilde{h}) + \frac{2LB}{l} \sqrt{\sum_{i=1}^l K(x_i, x_i)} + \sqrt{\frac{\ln(1/\delta)}{2l}}, \quad (7.31)$$

where the definition of kernel function follows from Theorem 2.

Proof. By definition of Rademacher complexity, we have

$$\begin{aligned} \mathcal{R}_m(\tilde{\mathcal{H}}) &= \frac{1}{l} \mathbb{E} \left[\sup_{\tilde{h} \in \tilde{\mathcal{H}}} \sum_{i=1}^l \sigma_i \tilde{h}_i \right] \\ &= \frac{1}{l} \mathbb{E} \left[\sup_{\tilde{h} \in \tilde{\mathcal{H}}} \sum_{i=1}^l (\sigma_i f(x_i) - \sigma_i g(x_i) + \sigma_i (\epsilon_2 e - \epsilon_1 e)) \right] \\ &= \frac{1}{l} \mathbb{E} \left[\sup_{f \in \mathcal{F}_B} \sum_{i=1}^l (\sigma_i f(x_i) - \sigma_i g(x_i) + \sigma_i (\epsilon_2 e - \epsilon_1 e)) \right] \\ &= \frac{1}{l} \mathbb{E} \left[\sup_{f \in \mathcal{F}_B} \sum_{i=1}^l \sigma_i f(x_i) \right] - E \left[\sum_{i=1}^l \sigma_i (g(x_i) + (\epsilon_2 e - \epsilon_1 e)) \right] \\ &= \frac{1}{l} \mathbb{E} \left[\sup_{f \in \mathcal{F}_B} \sum_{i=1}^l \sigma_i f(x_i) \right] \\ &= \mathcal{R}_m(\mathcal{F}_B). \end{aligned}$$

From Theorem 3, the relation between the generalization error bound and the empirical error found is obtained. Besides using theorem 2, substitute the upper bound of Theorem Rademacher complexity. Then the desired result is obtained. $\square$

7.2 Numerical experiments

In this section, we focus our attention on the numerical experiments performed to compare the performance of the proposed LSTSVR+ with the baseline models such as TSVR [62], TWSVR [183], LSTSVR [68], TSVQR [184], and SVR+ [70].

Experimental Setup: All the experiments are carried out in MATLAB 2022b on a PC with 11th Gen Intel(R) Core(TM) i7-11700 @ 2.50GHz processor and 16 GB RAM. Initially, datasets are divided into training and testing datasets. Within the training data, 5-fold cross-validation with grid search is applied to determine the optimal parameters minimizing the training RMSE value. Subsequently, employing these optimal parameters and utilizing the remaining testing data, the RMSE values are computed across various datasets. To reduce the computational cost of tuning of parameters, we considered $C_1 = C_4$, $C_2 = C_5$, $C_3 = C_6$ and $\epsilon_1 = \epsilon_2$ for the proposed LSTSVR+. All the datasets are normalized using min-max normalization. For the non-linear variant of our proposed model, we incorporated the Gaussian (RBF) kernel function, $(\exp(\|x_i - x_j\|^2)/2\sigma^2)$, σ denotes the kernel parameter). In the evaluation and comparison of the proposed LSTSVR+ with several baseline models, including TSVR, TWSVR, LSTSVR, TSVQR, and SVR+, we aim to gauge its effectiveness and superiority across diverse regression datasets. To incorporate PI into the datasets, we adopt a systematic approach. Specifically, we partition the total number of features (# features) in each dataset into two distinct subsets. We designate $\lceil \frac{\# \text{ features}}{2} \rceil$ features as the normal features, representing the conventional input variables and the remaining features are treated as privileged features. While the baseline models are trained solely on the normal features, the proposed LSTSVR+ model utilizes the privileged features as well. This standardized methodology ensures consistency in the integration of PI across all datasets, facilitating comparisons between the proposed LSTSVR+ and the baseline models.

We carried out the experiments on artificially generated datasets and real-world UCI [126], and stock datasets. The metrics considered to compare the performance of the models are

$$\text{RMSE} = \sqrt{\sum_j \frac{(y_j - \tilde{y}_j)^2}{l}}, \quad \text{SSE} = \sum_j (y_j - \tilde{y}_j)^2, \quad \frac{\text{SSE}}{\text{SST}} = \frac{\sum_j (y_j - \tilde{y}_j)^2}{\sum_j (y_j - \bar{y})^2}$$

and time required (in seconds) for training, where y_j is the actual target and $\tilde{y}_j$ is the predicted target. $\bar{y} = \frac{1}{l} \sum_j y_j$ corresponds to the mean of actual target values. SSE and SST denote the sum of squared error and sum of square total, respectively; RMSE denotes the root mean squared error.

7.2.1 Parameter selection

The different parameters involve in the proposed LSTSVR+ are regularization hyperparameters $C_1 = C_4$, $C_2 = C_5$, $C_3 = C_6$, $\epsilon_1 = \epsilon_2$ and the kernel parameter σ . Following [63], all the parameters (except ϵ_1 , ϵ_2) are tuned in the range $\{2^i | i = -8, -7, \dots, 7, 8\}$ for the synthetic, real-world datasets. The respective regularization parameters $C_1 = C_3$, $C_2 = C_4$, and σ for the baseline models, i.e., TSVR, TWSVR, LSTSVR, TSVQR and SVR+ are selected from the range $\{2^i | i = -8, -7, \dots, 7, 8\}$. The quantile parameter τ of TSVQR is chosen from the range $[0.1, 0.25, 0.5, 0.75, 0.9]$ following the article [184]. The value of ϵ_1 and ϵ_2 are fixed at 0.01 [185].

7.2.2 Artificially generated datasets

To facilitate a comprehensive comparison between the baseline models and the proposed LSTSVR+, we conducted experiments using four artificially generated datasets. The functions employed for these datasets are listed in Table 7.1. In our experimental setup, we generated 100 random training samples and 200 testing samples for each type of function.

To facilitate a robust comparison, the training samples are intentionally contaminated with three distinct types of noise levels [186], as follows:

- [1] Uniform noise over the interval $[-0.2, 0.2]$;
- [2] Gaussian noise with mean 0 and standard deviation 0.05;
- [3] Gaussian noise with mean 0 and standard deviation 0.2;

This approach allows us to assess the performance of the models under different noise conditions for a comprehensive evaluation. To comprehend the proposed LSTSVR+, we examine its execution using the artificially generated dataset. To simplify our explanation, we utilize a dataset comprising two samples generated by Function 1 with Type I noise, where each sample contains two features: x_1 and x_2 . We designate x_1 as the normal feature and x_2 as the privileged feature. To illustrate the framework of the proposed LSTSVR+ more clearly, we provide a schematic diagram in Figure 7.2.

Table 7.1: Different functions to generate artificial datasets.

Name	Function	Domain of the function
Function 1	$f(x_1, x_2) = \frac{\sin(\sqrt{x_1^2 + x_2^2})}{\sqrt{x_1^2 + x_2^2}}$	$x \in [-4\pi, 4\pi]$
Function 2	$f(x_1, x_2, x_3, x_4, x_5) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$
Function 3	$f(x_1, x_2) = \exp(x_1 \sin(\pi x_2))$	$x_1, x_2 \in [-1, 1]$
Function 4	$f(x_1, x_2) = 1.9[1.35 + \exp(x_1) \sin(13(x_1 - 0.6)^2) + \exp(3(x_2 - 0.5)) \sin(4\pi(x_2 - 0.9)^2)]$	$x_1, x_2 \in [0, 1]$

The training samples are contaminated with the different generated noise and the test samples are noise-free. Four independent experiments are performed on each dataset with different types of noise and the average values are reported in Table 7.2. The least RMSE, SSE, SSE/SST value correspond to the best performance. The average RMSE values for artificially generated datasets are 0.1433, 0.1406, 0.1433, 0.1405, 0.1499, and 0.135 for TSVR, TWSVR, LSTSVR, TSVQR, SVR+ and proposed LSTSVR+, respectively. It's important to note that the average RMSE value may not always provide an optimal assessment

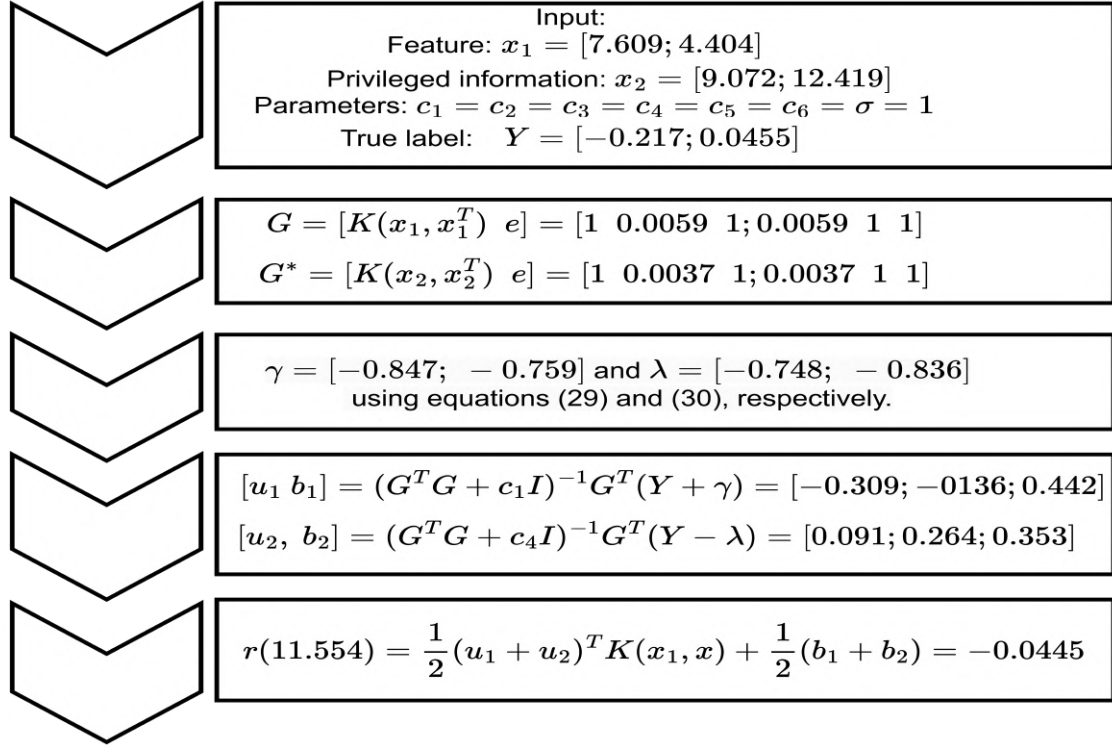


Figure 7.2: A framework diagram illustrating the proposed LSTSVR+ applied to Function 1 with Type I noise using two samples.

because a higher RMSE value for one dataset may be offset by a lower value for another. To address this, each dataset is ranked according to the performance across different models and metrics such that the best model gets the least rank and vice versa. The average ranks corresponding to RMSE values in the aforementioned order of models are 4.83, 2.58, 4.75, 2.67, 5, and 1.17, respectively. It reflects that the proposed LSTSVR+ performs well on the artificially generated datasets. Associated with the optimal parameters of RMSE across all the datasets, we calculate the SSE and SSE/SST metrics. The average SSE and SSE/SST of the proposed LSTSVR+ are 0.4 and 0.6833, respectively, which are the least values among all the models. The average rank of the proposed LSTSVR+ corresponding to SSE is 1.08, which is the best. Further, Figure 7.3 presents a visual comparison of the performance between the baseline models and the proposed LSTSVR+ across various artificially generated functions with different types of noise.

Table 7.2: RMSE values along with SSE, SSE/SST and training time (in seconds) across the artificially generated synthetic datasets with different types of noise for TSVR, TWSVR, LSTSVR, TSVQR, SVR+ and the proposed LSTSVR+.

Dataset	Metric	TSVR [62]	TWSVR [183]	LSTSVR [68]	TSVQR [184]	SVR+ [70]	Proposed LSTSVR+
Type 1: Uniform noise over the interval $[-0.2, 0.2]$.							
function 1	RMSE	0.1341	0.1329	0.1347	0.131	0.144	0.1264
	SSE	0.38	0.3737	0.3825	0.3626	0.4452	0.3305
	SSE/SST	0.9925	0.9914	0.9927	0.9891	0.9887	0.9998
	Time	0.0119	0.007	0.0048	0.0088	0.0468	0.0082
function 2	RMSE	0.1431	0.1397	0.1419	0.1428	0.143	0.1298
	SSE	0.4147	0.3974	0.4106	0.4154	0.4178	0.3405
	SSE/SST	0.5898	0.5685	0.5855	0.5926	0.6033	0.4944
	Time	0.0105	0.008	0.0034	0.0182	0.0352	0.0044
function 3	RMSE	0.1974	0.1951	0.199	0.1908	0.1947	0.1928
	SSE	0.8003	0.7876	0.8117	0.7573	0.7805	0.7644
	SSE/SST	0.9727	0.9587	0.9897	0.9463	0.9818	0.9845
	Time	0.0133	0.0043	0.002	0.0023	0.0621	0.0042
function 4	RMSE	0.1045	0.1025	0.1053	0.104	0.1301	0.0979
	SSE	0.2212	0.2124	0.2247	0.2196	0.3454	0.1945
	SSE/SST	0.2827	0.2709	0.2871	0.2809	0.4323	0.2505
	Time	0.0155	0.0113	0.0048	0.0127	0.0385	0.0037
Type 2: Gaussian noise with mean 0 and standard deviation 0.05.							
function 1	RMSE	0.1047	0.1023	0.1052	0.1022	0.1034	0.0963
	SSE	0.2457	0.2296	0.2478	0.228	0.2377	0.2025
	SSE/SST	0.9826	0.9269	0.9892	0.9326	0.9637	0.9889
	Time	0.0148	0.0036	0.0042	0.0081	0.0427	0.005
function 2	RMSE	0.1433	0.1401	0.1421	0.1424	0.1439	0.1301
	SSE	0.4157	0.399	0.4111	0.4131	0.4226	0.3418
	SSE/SST	0.5863	0.5655	0.5813	0.5869	0.6058	0.4934
	Time	0.011	0.0077	0.0022	0.0153	0.0369	0.0045
function 3	RMSE	0.1997	0.197	0.2009	0.1934	0.1986	0.1918
	SSE	0.8224	0.8062	0.8292	0.7823	0.8141	0.7608
	SSE/SST	0.9848	0.9726	0.9948	0.9688	0.9821	0.9754
	Time	0.0125	0.0034	0.0015	0.014	0.072	0.0042
function 4	RMSE	0.1042	0.1008	0.1043	0.1027	0.1301	0.0973
	SSE	0.2193	0.2055	0.2202	0.2152	0.345	0.1921
	SSE/SST	0.2843	0.2671	0.2858	0.2798	0.4313	0.2509
	Time	0.0173	0.0065	0.0017	0.0097	0.0397	0.004
Type 3: Gaussian noise with mean 0 and standard deviation 0.2.							
function 1	RMSE	0.1452	0.1432	0.1445	0.1422	0.1443	0.1411
	SSE	0.4363	0.4243	0.4332	0.4199	0.4332	0.4125
	SSE/SST	1.01	0.9811	1.0024	0.9895	0.9902	1.0102
	Time	0.0112	0.005	0.0058	0.0078	0.0723	0.0045
function 2	RMSE	0.1425	0.1397	0.1414	0.1415	0.1432	0.1299
	SSE	0.4108	0.3969	0.4072	0.4084	0.4191	0.341
	SSE/SST	0.5828	0.5654	0.5794	0.5854	0.6031	0.4948
	Time	0.011	0.005	0.0045	0.0126	0.032	0.0041
function 3	RMSE	0.1957	0.1914	0.1948	0.1875	0.1921	0.1878
	SSE	0.7864	0.7549	0.7762	0.7207	0.7593	0.7185
	SSE/SST	1.0072	0.9675	0.9981	0.9609	0.9904	0.9901
	Time	0.0119	0.0049	0.0039	0.0055	0.0708	0.0034
function 4	RMSE	0.105	0.1019	0.1053	0.1051	0.1319	0.0993
	SSE	0.2233	0.2104	0.2253	0.2247	0.3544	0.2004
	SSE/SST	0.2949	0.2777	0.2978	0.2958	0.4616	0.2662
	Time	0.0152	0.0056	0.0075	0.0117	0.0393	0.0043
Average	RMSE	0.1433	0.1406	0.1433	0.1405	0.1499	0.135
	SSE	0.448	0.4332	0.4483	0.4306	0.4812	0.4
	SSE/SST	0.7142	0.6928	0.7153	0.7007	0.7529	0.6833
Average rank	RMSE	4.83	2.58	4.75	2.67	5	1.17
	SSE	4.67	2.58	4.71	2.83	5.13	1.08

7.2.3 Real-world datasets

In this subsection, we present the results of numerical experiments involving the proposed LSTSVR+ and baseline models across 21 real-world UCI, KEEL and

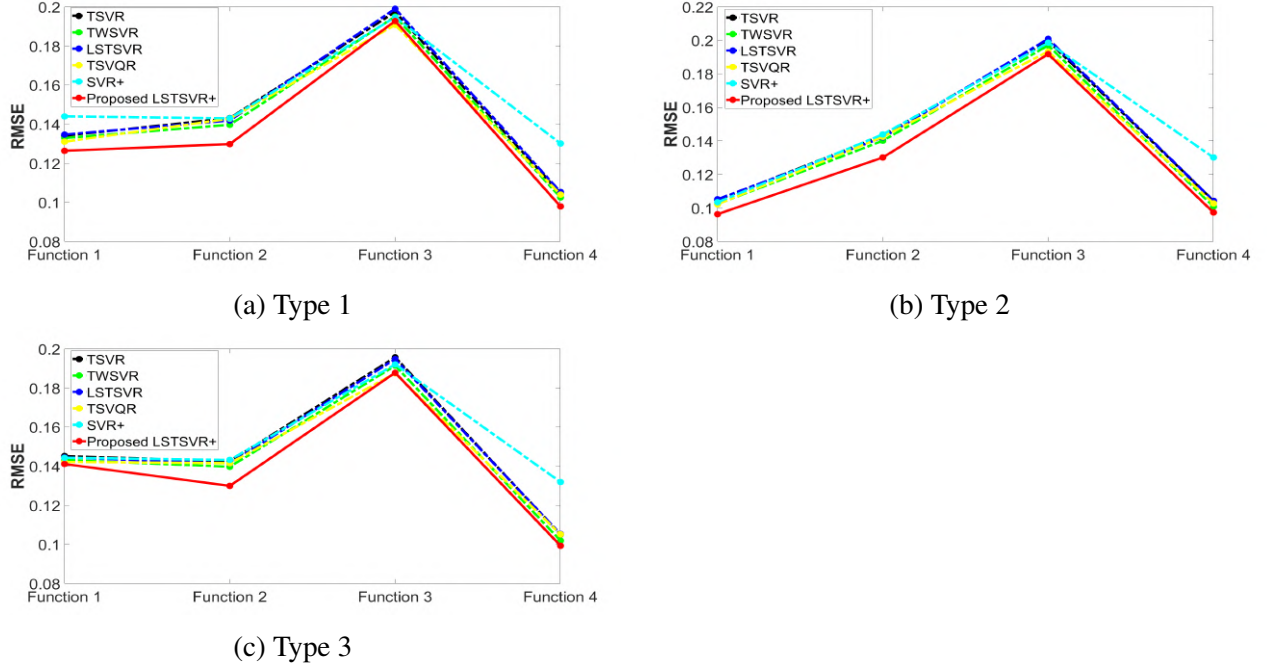


Figure 7.3: Performance of baseline models and proposed LSTSVR+ across artificially generated datasets on various types of noise.

stock datasets. Following UCI and KEEL datasets are considered: Auto-price, Daily_Demand_Forecasting_Orders, auto-original, bodyfat, hungary chickenpox, Istanbul stock exchange data, qsar aquatic toxicity, servo, slump_test, wpbc, yacht hydrodynamics, autoMPG, baseball, ele-1, dee, machineCPU which are split in a 70 : 30 ratio for training and testing, respectively.

The stock datasets are IBM, NVDA, citigroup, Infosys, microsoft and wipro, which are publicly available at [\[Link\]](#). The aforementioned stock index financial datasets are derived from the low prices of stocks and are designed to predict the current value using the five previous values. These datasets span from February 2, 2018, to December 30, 2020, encompassing a total of 738 or 755 samples. In our experimental setup, the first 200 samples are designated as the training data, while the remaining are reserved for testing the performance of the model.

Table [7.3](#) depicts the performance of the proposed LSTSVR+ and existing models

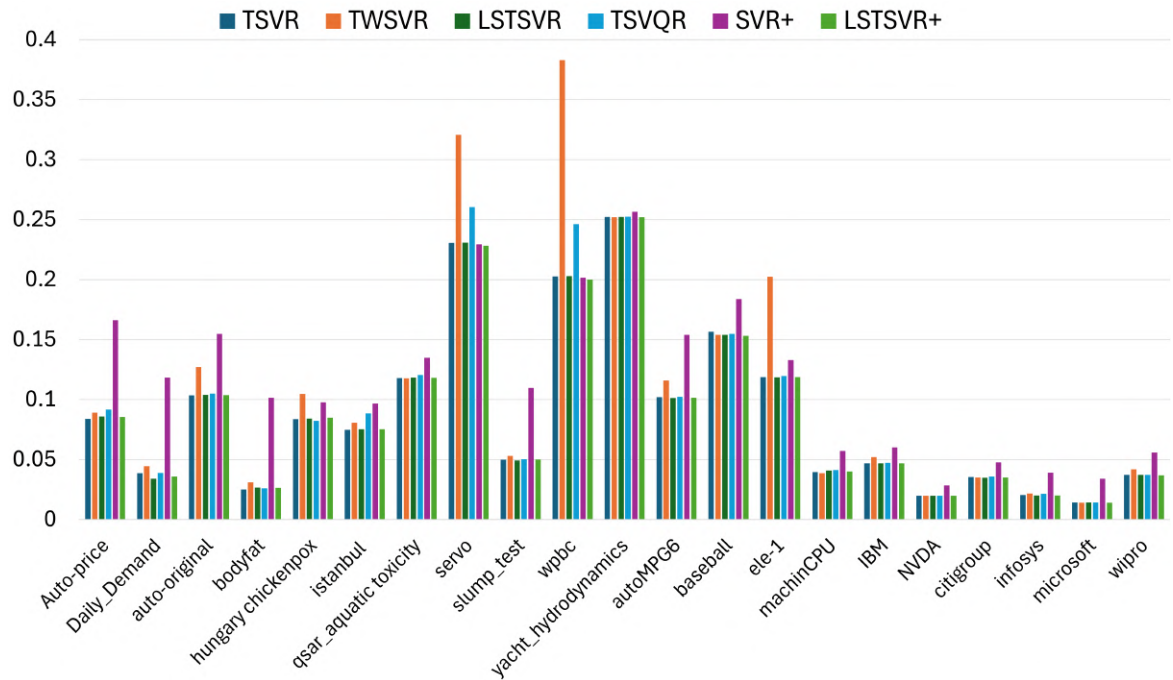


Figure 7.4: Comparison of RMSE values across different datasets for various models using bar graph.

TSVR, TWSVR, LSTSVR, TSVQR, and SVR+ in terms of metrics RMSE, SSE, SSE/SST and training time (in seconds). Table 7.3 also includes the number of samples and features for each of the datasets. The performance evaluation based on the RMSE indicates that lower RMSE values correspond to better model performance. The proposed LSTSVR+ model achieves the least average RMSE value of 0.0879, followed by LSTSVR, TSVR, TSVQR, TWSVR, and SVR+ with an average RMSE of 0.0881, 0.0883, 0.0931, 0.1095, and 0.1171, respectively. These values indicate that the proposed model exhibits the best performance among all the models. Figure 7.4 compares the RMSE values across different datasets for various models using bar graph. For the ease of representation of the figure, we shorten the name of Daily_Demand.Forecasting.Orders and istanbul stock exchange data to Daily_Demand and istanbul, respectively. The average rank corresponding to the RMSE value for the proposed LSTSVR+ is 2.14, which is the least. Further, we calculated the average of SSE and SSE/SST metrics obtained across all the datasets corresponding to op-

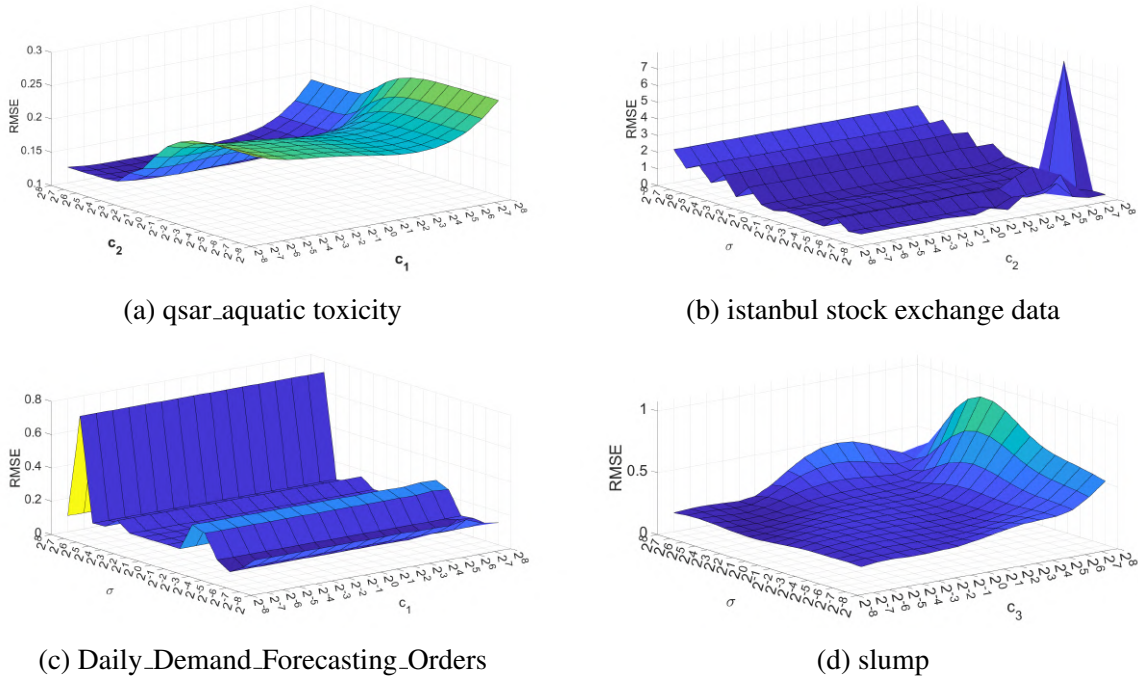


Figure 7.5: Influence of parameters on the RMSE values of the proposed LSTSVR+ across different UCI datasets.

timal parameters of RMSE. The average SSE value for the proposed LSTSVR+ is 0.5267, the second-best average SSE value is 0.5295 for LSTSVR, then 0.5311 for TSVR. The proposed LSTSVR+ has an average rank of 2.29, analogous to SSE which is the least among the models. The average rank corresponding to SSE/SST for the proposed LSTSVR+ is 2.26, then 2.55 for TSVR. The average rank of the remaining models in increasing order are as follows: 3.1 for LSTSVR, 3.4 for TSVQR, 3.93 for TWSVR, and 5.76 for SVR+. Thus, the average ranks average RMSE, and SSE values depict the proposed LSTSVR+ has superior performance.

Figure 7.5 illustrates the dependence of RMSE values on various hyperparameters across different UCI datasets for proposed LSTSVR+. Specifically, Figure 7.5a showcases the RMSE variation concerning hyperparameters C_1 and C_2 on the UCI dataset qsar_aquatic toxicity. Furthermore, Figure 7.5b demonstrates the impact of hyperparameter variation on C_2 and σ across the Istanbul Stock Exchange dataset. Similarly, Figure 7.5c portrays the

Table 7.3: RMSE values along with SSE, SSE/SST and training time (in seconds) across the real-world datasets for TSVR, TWSVR, LSTSVR, TSVQR, SVR+ and the proposed LSTSVR+.

Dataset	r	pin SVM [17]	TWSVM [22]	UTSVM [26]	Pin-GTSVM [101]	Tpin-UTWSVM	
Dataset (#samples, #features)	Metric	TSVR [62]	TWSVR [183]	LSTSVR [68]	TSVQR [184]	SVR+ [70]	Proposed LSTSVR+
Auto-price (159,8)	RMSE	0.084	0.0892	0.0858	0.0916	0.1661	0.0855
	SSE	0.1596	0.1827	0.1663	0.1895	0.6381	0.1692
	SSE/SST	0.1695	0.1938	0.1779	0.1953	0.6678	0.1788
	Time	0.0075	0.0092	0.0032	0.009	0.0201	0.0039
Daily_Demand_Forecasting_Orders (60,6)	RMSE	0.0386	0.0444	0.0339	0.0388	0.1182	0.0359
	SSE	0.0131	0.0177	0.0104	0.0129	0.134	0.011
	SSE/SST	0.0448	0.0606	0.0357	0.0444	0.4031	0.0378
	Time	0.0038	0.0018	0.0011	0.0095	0.0066	0.0022
auto-original (392,4)	RMSE	0.1035	0.1272	0.104	0.1049	0.1548	0.1023
	SSE	0.5891	0.89	0.5953	0.6075	1.3182	0.5929
	SSE/SST	0.2602	0.3689	0.2636	0.2676	0.5727	0.262
	Time	0.0281	0.0352	0.0138	0.0193	0.2344	0.0099
bodyfat (252,7)	RMSE	0.0249	0.031	0.0266	0.026	0.1016	0.0265
	SSE	0.0361	0.0445	0.0377	0.0369	0.3817	0.0372
	SSE/SST	0.038	0.0468	0.0397	0.0389	0.3938	0.0393
	Time	0.013	0.0164	0.0051	0.0133	0.0585	0.0066
hungary_chickenpox (522,10)	RMSE	0.0838	0.1047	0.0841	0.0823	0.0976	0.0848
	SSE	0.5652	0.8193	0.5675	0.5419	0.7435	0.5787
	SSE/SST	0.72	0.8981	0.7338	0.6915	0.9422	0.7358
	Time	0.0495	0.0596	0.0172	0.0267	0.3751	0.0167
istanbul_stock_exchange_data (536,4)	RMSE	0.0747	0.0807	0.0752	0.0884	0.0967	0.0751
	SSE	0.4211	0.4898	0.4272	0.6032	0.7093	0.4262
	SSE/SST	0.4139	0.4773	0.4193	0.498	0.6167	0.4183
	Time	0.0424	0.0373	0.0264	0.0256	0.357	0.0178
qsar_aquatic_toxicity (546,4)	RMSE	0.1178	0.1177	0.1182	0.1205	0.1349	0.118
	SSE	1.0672	1.069	1.0739	1.1196	1.4029	1.0685
	SSE/SST	0.5142	0.5119	0.5176	0.5292	0.6795	0.5146
	Time	0.0357	0.0411	0.0218	0.0201	0.4544	0.0166
servo (167,2)	RMSE	0.2306	0.3206	0.2307	0.2604	0.2294	0.2282
	SSE	1.288	2.4065	1.2823	1.6016	1.2539	1.2562
	SSE/SST	0.9578	1.0269	0.9703	0.9564	0.9587	0.9454
	Time	0.0057	0.0086	0.0025	0.0009	0.0217	0.003
slump_test (103,5)	RMSE	0.0499	0.053	0.0493	0.0502	0.1098	0.05
	SSE	0.038	0.043	0.0386	0.04	0.1774	0.0392
	SSE/SST	0.0823	0.0932	0.0859	0.0868	0.3598	0.0845
	Time	0.0071	0.0013	0.0015	0.0111	0.015	0.0013
wpbc (194,17)	RMSE	0.2027	0.383	0.2028	0.2463	0.2015	0.2001
	SSE	1.1782	3.9919	1.1768	1.7166	1.1626	1.1495
	SSE/SST	1.0004	0.9998	1.0062	0.9964	1	0.9998
	Time	0.0092	0.0171	0.0051	0.0095	0.1974	0.0049
yacht_hydrodynamics (308,3)	RMSE	0.2522	0.2521	0.2522	0.2525	0.2564	0.2519
	SSE	2.7814	2.779	2.7814	2.7852	2.881	2.7723
	SSE/SST	1	1	1.0006	1	1.0037	0.9977
	Time	0.017	0.0157	0.0078	0.0124	0.1063	0.0064
autoMPG6 (392,3)	RMSE	0.102	0.1159	0.1014	0.1023	0.154	0.1016
	SSE	0.5817	0.7413	0.5687	0.5801	1.3189	0.5712
	SSE/SST	0.2622	0.3228	0.2566	0.2603	0.5894	0.2574
	Time	0.0202	0.0266	0.0111	0.015	0.1812	0.0133
baseball (337,8)	RMSE	0.1566	0.1539	0.154	0.1547	0.1838	0.1531
	SSE	1.1605	1.1193	1.121	1.1312	1.5987	1.108
	SSE/SST	0.5788	0.5592	0.5649	0.5677	0.7888	0.5541
	Time	0.0191	0.0158	0.0091	0.0134	0.1185	0.0068
ele-1 (495,1)	RMSE	0.1187	0.2024	0.1186	0.1195	0.1329	0.1188
	SSE	0.9822	3.5617	0.9803	0.9953	1.231	0.9839
	SSE/SST	0.7295	1.9749	0.7398	0.7412	0.9115	0.7313
	Time	0.0312	0.0718	0.0236	0.0279	0.3444	0.0102

RMSE variation with hyperparameters C_1 and σ on the Daily_Demand_Forecasting_Orders dataset. Lastly, Figure 7.5d illustrates the RMSE variation with C_3 and σ across the slump dataset. Notably, all other hyperparameters remain fixed at their optimal values. Hence, these figures collectively highlight the significant influence of parameters on the perfor-

Table 7.3: Continued from previous page-RMSE values along with SSE, SSE/SST and training time (in seconds) across the real-world datasets for TSVR, TWSVR, LSTSVR, TSVQR, SVR+ and the proposed LSTSVR+.

Dataset	r	pin SVM [17]	TWSVM [22]	UTSVM [26]	Pin-GTSVM [101]	Tpin-UTWSVM	
Dataset	Metric	TSVR [62]	TWSVR [183]	LSTSVR [68]	TSVQR [184]	SVR+ [70]	Proposed LSTSVR+
(#samples, #features)							
machinCPU (209,3)	RMSE	0.0395	0.0386	0.0409	0.0413	0.0571	0.0399
	SSE	0.0508	0.0488	0.0531	0.0596	0.1092	0.0575
	SSE/SST	0.2102	0.2025	0.2207	0.2448	1.0084	0.2399
	Time	0.0109	0.0026	0.0042	0.0138	0.043	0.0054
IBM (750,3)	RMSE	0.0469	0.0519	0.0469	0.0473	0.0601	0.0469
	SSE	0.0889	0.1083	0.0892	0.0908	0.1473	0.0893
	SSE/SST	0.221	0.2617	0.2222	0.2249	0.3632	0.2219
	Time	0.0147	0.0169	0.0086	0.0121	0.0999	0.0057
NVDA (750,3)	RMSE	0.0197	0.0198	0.0198	0.0197	0.0283	0.0198
	SSE	0.0162	0.0162	0.0163	0.016	0.0327	0.0163
	SSE/SST	0.3116	0.3129	0.3155	0.3092	0.5905	0.3136
	Time	0.0122	0.0145	0.007	0.0134	0.1222	0.008
citigroup (750,3)	RMSE	0.0355	0.035	0.0349	0.0359	0.0477	0.035
	SSE	0.0523	0.052	0.0506	0.0542	0.0921	0.0515
	SSE/SST	0.2476	0.2464	0.2597	0.2531	0.4125	0.2441
	Time	0.0252	0.0074	0.008	0.0117	0.1172	0.0046
infosys (750,3)	RMSE	0.0205	0.0218	0.0201	0.0215	0.0392	0.0201
	SSE	0.0169	0.0193	0.0163	0.0187	0.0626	0.0163
	SSE/SST	0.1063	0.1204	0.1024	0.1157	0.3818	0.1021
	Time	0.0122	0.0211	0.008	0.0114	0.1308	0.0054
microsoft (750,3)	RMSE	0.0143	0.0141	0.0143	0.0143	0.034	0.014
	SSE	0.0083	0.0082	0.0084	0.0083	0.0463	0.0081
	SSE/SST	0.0719	0.0702	0.0722	0.0717	0.3918	0.0698
	Time	0.0107	0.005	0.0062	0.0116	0.0988	0.0062
wipro (750,3)	RMSE	0.0372	0.0419	0.0372	0.0371	0.0559	0.0368
	SSE	0.0588	0.0742	0.0584	0.0588	0.1259	0.0579
	SSE/SST	0.1743	0.2172	0.1731	0.1743	0.3644	0.1715
	Time	0.0144	0.0217	0.007	0.0117	0.0911	0.0047
Average	RMSE	0.0883	0.1095	0.0881	0.0931	0.1171	0.0879
	SSE	0.5311	0.8801	0.5295	0.5842	0.7413	0.5267
	SSE/SST	0.3864	0.4745	0.3874	0.3937	0.6381	0.3867
Average rank	RMSE	2.43	4.1	2.81	4	5.52	2.14
	SSE	2.57	4.19	2.62	3.86	5.48	2.29
	SSE/SST	2.55	3.93	3.1	3.4	5.76	2.26

mance of the proposed LSTSVR+ algorithm.

7.2.4 Statistical analysis

In this subsection, we conduct a statistical assessment of the performance of the baseline models TSVR, TWSVR, LSTSVR, TSVQR, SVR+, and proposed LSTSVR+ by employing Friedman test and Nemenyi post hoc test [123]. By conducting the tests, we can thoroughly evaluate the models' performance, eliminating potential biases and enabling us to make general conclusions about their ability to generalize. The Friedman test statistically compares the models with the null hypothesis assumption that all models are equivalent and

Table 7.4: Nemenyi post hoc significant difference for the proposed LSTSVR+ with respect to the baseline models on artificially generated datasets corresponding to different metrics.

Metric	TSVR [62]	TWSVR [183]	LSTSVR [68]	TSVQR [184]	SVR+ [70]
RMSE	Yes	No	Yes	No	Yes
SSE	Yes	No	Yes	No	Yes

the Nemenyi test compares the models pairwise. The Friedman statistic is calculated using the relation (2.56). The undesirable conservative nature of Friedman statistics can be overcome by F-distribution [123], which is calculated using the relation (2.57). We analyze the models in a pairwise way by using the Nemenyi test which concludes that the models are significantly different if the difference in their ranks is greater than the critical difference (cd), where it is calculated using 2.58 with $q_\alpha = 2.850$ at 5% level of significance, i.e., $\alpha = 0.05$.

Artificially generated datasets: First, we statistically analyse the artificially generated datasets for which we have $md = 6$ and $N = 12$. On calculation using the average of ranks corresponding to RMSE, we obtain $\chi_F^2 = 43.0135$ and $F_F = 27.8544$ with degree of freedom 5 and (5, 55), respectively. $F_{(5,55)} = 2.3828 < F_F$. Thus, the null hypothesis does not hold and the models are not equivalent. Further, we employ the Nemenyi test and calculate $cd = 2.1767$. Corresponding to the average of ranks assigned using SSE values, we have $\chi_F^2 = 43.3426$, $F_F = 28.6221$ and $cd = 2.1767$. We observe the F_F corresponding to average of the ranks assigned using SSE values is greater than $F_{(5,55)}$, thus the null hypothesis is rejected for SSE case. Table 7.4 represents the models which are significantly different from the proposed LSTSVR+ using the Nemenyi test over artificially generated datasets for RMSE and SSE metrics. Table 7.4 infers that TWSVR and TSVQR are not significantly different than the proposed LSTSVR+, however, statistical tests on other datasets establish the significant difference of the model.

Real-world datasets: For the numerical experiments on real-world datasets, we have

Table 7.5: Nemenyi post hoc significant difference for the proposed LSTSVR+ with respect to the baseline models on real-world datasets corresponding to different metrics.

Metric	TSVR [62]	TWSVR [183]	LSTSVR [68]	TSVQR [184]	SVR+ [70]
RMSE	No	Yes	No	Yes	Yes
SSE	No	Yes	No	No	Yes
SSE/SST	No	Yes	No	No	Yes

$N = 21$ and $md = 6$. Thus degree of freedom of χ_F^2 and F_F are obtained as 5 and (5, 100), respectively. At 5% level of significance, $F_{(5,100)} = 2.3053$. Using the average ranks of the model based on the RMSE values depicted in Table 7.3, we calculate $\chi_F^2 = 48.9660$, $F_F = 17.4772$ and $cd = 1.6454$. Corresponding to the average ranks assigned to the models based on SSE values, we have $\chi_F^2 = 46.1970$, $F_F = 15.7125$, and $cd = 1.6454$. Emphasizing the average of the ranks assigned using SSE/SST values, we have $\chi_F^2 = 47.4156$, $F_F = 16.4682$ and $cd = 1.6454$. For the three F_F values obtained using RMSE, SSE, and SSE/SST metric, we have $F_F > F_{(5,100)}$, thus the null hypothesis of the Friedman test is rejected and models are not equivalence. Further Table 7.5 shows which models are significantly different from the proposed LSTSVR+ corresponding to the metric RMSE, SSE, SSE/SST. Table 7.5 depicts TSVR and LSTSVR fails to be significantly different from the proposed LSTSVR+ corresponding to the three metrics. However, statistical tests on different datasets prove the significant difference between the models. Thus, stronger numerical results and consistently favourable outcomes from statistical tests provide significant support for the claim that the proposed LSTSVR+ performs better in generalization performance than the baseline models.

7.3 Application

In this section, we present the results of numerical experiments conducted for the proposed LSTSVR+ and the existing TSVR [62], TWSVR [183], LSTSVR [68], TSVQR

Table 7.6: RMSE values along with SSE, SSE/SST and training time (in seconds) across the real-world time-series datasets for TSVR, TWSVR, LSTSVR, TSVQR, SVR+ and the proposed LSTSVR+.

Dataset (#samples, #features)	Metric	TSVR [62]	TWSVR [183]	LSTSVR [68]	TSVQR [184]	SVR+ [70]	Proposed LSTSVR+
NNGC1_dataset_D1.V1.003 (430,3)	RMSE	0.2272	0.1564	0.1569	0.2371	0.1657	0.1439
	SSE	3.1422	1.4744	1.493	3.3962	1.6633	1.2493
	SSE/SST	0.62	0.7314	0.7362	0.8578	0.7734	0.6191
	Time	0.078	0.0459	0.0118	0.0295	0.3981	0.0237
NNGC1_dataset_D1.V1.004 (545,3)	RMSE	0.1629	0.1674	0.1687	0.2734	0.1724	0.1629
	SSE	2.0459	2.1598	2.1962	5.9796	2.2944	2.0508
	SSE/SST	0.8326	0.8815	0.8915	0.9394	0.9204	0.8252
	Time	0.082	0.0557	0.0214	0.0586	0.7138	0.0571
NNGC1_dataset_D1.V1.005 (430,3)	RMSE	0.1274	0.1126	0.1186	0.1714	0.1234	0.1083
	SSE	0.9772	0.7654	0.8482	1.7797	0.92	0.7111
	SSE/SST	0.2824	0.3117	0.3449	0.512	0.362	0.2897
	Time	0.0392	0.0672	0.0091	0.0348	0.2226	0.0451
NNGC1_dataset_D1.V1.006 (610,3)	RMSE	0.0895	0.0875	0.0899	0.0971	0.1055	0.0882
	SSE	0.6848	0.6566	0.6914	0.8083	0.9524	0.6666
	SSE/SST	0.2711	0.264	0.2777	0.3217	0.3665	0.2679
	Time	0.0916	0.1702	0.0348	0.0736	0.7409	0.0795
NNGC1_dataset_D1.V1.007 (610,3)	RMSE	0.1274	0.1322	0.1269	0.2388	0.1363	0.125
	SSE	1.3961	1.5047	1.3864	5.0833	1.5996	1.3414
	SSE/SST	0.6505	0.7312	0.6737	1.1004	0.7797	0.6503
	Time	0.0909	0.0792	0.0262	0.0874	0.9921	0.0452
NNGC1_dataset_D1.V1.009 (540,3)	RMSE	0.121	0.1229	0.1226	0.1365	0.1206	0.1179
	SSE	1.11	1.1456	1.1389	1.4172	1.1032	1.0535
	SSE/SST	0.5401	0.5813	0.5778	0.686	0.5575	0.5354
	Time	0.0911	0.0912	0.0187	0.0264	0.7775	0.0823
NNGC1_dataset_D1.V1.010 (585,3)	RMSE	0.1432	0.1416	0.1423	0.2485	0.1438	0.1419
	SSE	1.6854	1.6483	1.6631	5.0693	1.701	1.6537
	SSE/SST	0.7497	0.7493	0.7557	0.9095	0.7621	0.7506
	Time	0.1053	0.0349	0.0236	0.0637	1.0016	0.1168
NNGC1_dataset_E1.V1.001 (370,3)	RMSE	0.2218	0.2185	0.2219	0.3373	0.2201	0.2146
	SSE	2.5633	2.5002	2.5672	5.9456	2.5204	2.4116
	SSE/SST	0.6469	0.6649	0.6846	0.8248	0.6739	0.6409
	Time	0.0428	0.0941	0.0096	0.0146	0.2745	0.1288
NNGC1_dataset_E1.V1.008 (740,3)	RMSE	0.1377	0.1297	0.1375	0.1713	0.1317	0.1353
	SSE	1.9722	1.7544	1.9658	3.0604	1.8061	1.9128
	SSE/SST	0.7887	0.7201	0.8043	0.9674	0.744	0.788
	Time	0.149	0.0914	0.0342	0.0529	1.7184	0.0424
NNGC1_dataset_E1.V1.009 (740,3)	RMSE	0.1241	0.1127	0.1304	0.1662	0.1295	0.1327
	SSE	1.6196	1.3371	1.7705	2.8728	1.7508	1.8535
	SSE/SST	0.71	0.5871	0.7752	0.9684	0.7633	0.8134
	Time	0.158	0.0843	0.0382	0.1167	1.6502	0.0427
Average	RMSE	0.1482	0.1382	0.1416	0.2078	0.1449	0.1371
	SSE	1.7197	1.4947	1.5721	3.5412	1.6311	1.4904
Average rank	RMSE	3.55	2.2	3.6	5.9	3.9	1.85
	SSE	3.5	2.2	3.6	5.9	3.9	1.9
	SSE/SST	2.2	2.5	4.1	5.9	4.3	2

[184], SVR+ [70] on the time-series real-world datasets [127] to demonstrate the application of the proposed LSTSVR+ model.

Across various fields such as statistics, signal processing, and econometrics, time-series data represent a sequential arrangement of data points, typically measured at uniform intervals. The characteristics of time-series dataset used are as follows:

- **Temporal proximity and dependence:** Time-series data exhibit a strong temporal

proximity, where observations closer in time tend to have stronger relationships than those farther apart. This characteristic reflects the inherent dependence among consecutive observations. The acknowledgment of temporal proximity underscores the importance of considering the sequential order of observations in time-series analysis.

- **Unidirectional progression of time:** Time-series models leverage the unidirectional progression of time, where values for a given period are typically derived from preceding values rather than future ones. This approach aligns with the natural flow of time and the causal relationships between past and present observations. By focusing on past observations to predict future values, time-series models account for the cumulative effect of historical data on the current state of the system.

Thus, the acknowledgement of temporal proximity and the unidirectional progression of time are fundamental characteristics of time-series datasets. These characteristics highlight the sequential nature of time-series data and emphasize the importance of capturing temporal dependencies to make predictions about future trends.

In selecting applications for evaluating the efficiency of LSTSVR+, the choice of time-series datasets is deliberate and strategic. Time-series data are prevalent across various domains, offering real-world scenarios with diverse characteristics such as finance, economics, and environmental science. By applying the proposed LSTSVR+ to these datasets, we aim to demonstrate its practical utility and generalizability in capturing the complex temporal dynamics inherent in time-series data. This approach allows us to showcase the proposed LSTSVR+ effectiveness across different domains and provide valuable insights into its performance in real-world settings.

Time-series datasets often exhibit patterns, trends, and dependencies over time, making them suitable for incorporating supplementary data to enhance model performance. The chosen application possesses features or attributes that allow for the efficient utilization of

PI, such as seasonality, trends, or past observations' dependencies. By leveraging PI in such contexts, the proposed LSTSVR+ can potentially improve its generalization performance, and ability to capture complex temporal dynamics inherent in time-series data. Hence, selecting applications with similarities that facilitate the effective integration of PI ensures the algorithm's effectiveness and applicability in real-world time-series analysis.

7.3.1 Experimental analysis

We conducted random 70 : 30 splitting for training and testing, respectively, and the ranges of different parameters considered are consistent with the experimental setup defined in Section 7.2. The datasets are normalized using min-max normalization. Table 7.6 represents the performance of the models on the time-series dataset along with the details of the datasets. The proposed LSTSVR+ has the lowest average RMSE value of 0.1371, the second lowest value for TWSVR which is 0.1382, followed by LSTSVR, SVR+, TSVR, and TSVQR with RMSE 0.1416, 0.1449, 0.1482, and 0.2078, respectively. Using the optimal hyperparameters corresponding to the least RMSE for each model across all datasets, we obtain the SSE and SSE/SST metrics. The average SSE is the smallest for the proposed LSTSVR+ among all the models. As average RMSE or SSE values are not reliable always, we obtain the ranks of all the models across different datasets, considering different metrics. The average ranks corresponding to the RMSE values are 1.85, 2.2, 3.55, 3.6, 3.9, and 5.9 for proposed LSTSVR+, TWSVR, TSVR, LSTSVR, SVR+, and TSVQR, respectively. Considering SSE metric, the average rank assigned to the proposed LSTSVR+ is 1.9, which is the best among all the models.

Employing the statistical tests, i.e., Friedman and Nemenyi post hoc test as mentioned in subsection 7.2.4, we obtain $\chi_F^2 = 29.5571$ and $F_F = 13.0126$ with degree of freedom 5 and (5, 45), respectively. At 5% level of significance, we have $F_{(5,45)} = 2.422 < F_F$, thus the

null hypothesis of the Friedman test is rejected and models are not equivalent. Hence, the numerical results and statistical analysis prove the superiority of the proposed LSTSVR+.

7.4 Contribution

The key contribution of LSTSVR+ lies in its ability to incorporate privileged information (PI) during training, enabling improved generalization and more informed decision-making. Unlike classical SVR+ approaches, which often involve computationally expensive quadratic programming, our model achieves efficiency through a least-squares formulation. It is introduced in both linear and non-linear variants, making it versatile across diverse regression settings. The non-linear model's kernel-based structure allows it to capture complex patterns while retaining tractability. Theoretical analysis confirms the uniqueness of the solution and establishes a generalization error bound via Rademacher complexity. Empirical evaluations on synthetic and real-world datasets, including time-series and financial data, demonstrate that LSTSVR+ consistently outperforms baseline methods (TSVR, TWSVR, LSTSVR, TSVQR, and SVR+) in terms of RMSE, SSE, and SSE/SST. These results underscore the model's robustness, computational efficiency, and the practical advantage gained by integrating privileged features.

7.5 Summary

This chapter introduces the least squares twin support vector regression with privileged information (LSTSVR+), which integrates the LUPI paradigm into the framework of LSTSVR. The proposed model combines the benefits of both LUPI and LSTSVR, enhancing the learning process by incorporating additional privileged data, acting as a teacher during the training phase. This approach improves model performance and generalization

by incorporating regularization terms in both the regressor and correction functions, adhering to the SRM principle to prevent overfitting.

The LSTSVR+ model is efficient as it solves a system of linear equations, and its non-linear variant effectively captures relationships in high-dimensional input data. The chapter also establishes a generalization error bound for LSTSVR+ based on Rademacher complexity, ensuring theoretical robustness. Through experiments conducted on artificially generated and real-world datasets, the proposed model consistently outperforms baseline models. Additionally, experiments on time-series datasets highlight the superior performance of LSTSVR+ in practical applications.

Overall, the proposed LSTSVR+ framework demonstrates enhanced performance, robustness, and efficiency, making it a valuable tool for regression tasks, particularly in situations where PI is available.

Chapter 8

Conclusions and Future Works

This thesis addresses key challenges in hyperplane-based learning, including noise, outliers, imbalanced data, scalability, and efficiency in classification and regression tasks. It introduces adaptability in the classification model by extending the pinball loss while retaining its core benefits. It also enhances sparsity in non-hyperplane models using pinball loss while preserving its theoretical advantages. Additionally, fuzzy-based weighting schemes are proposed to mitigate the adverse effects of noise, outliers, and imbalanced datasets. To further address scalability issues in large-scale datasets, a novel pruning technique, ProDiSE is developed, ensuring efficient and sparse model training. Robust and sparse one-class classification models are also introduced to enhance anomaly detection performance. Furthermore, privileged information is effectively integrated into regression frameworks, leading to improved generalization and computational efficiency. Collectively, these contributions advance hyperplane-based machine learning, offering robust, scalable, and efficient solutions to real-world challenges. Further, Table [8.1](#) compares different proposed models.

Table 8.1: Comparison of different proposed models.

Model Name	Problem type	Key features	Strengths	Weaknesses
FP-SVM (flexible pinball loss SVM)	Binary classification	Extended pinball loss with τ_1, τ_2	Convex under constraint, insensitive to noise	High computational complexity of solving QPP
Tpin-UTWSVM (truncated pinball universum TWSVM)	Binary classification	Combines truncated pinball loss with universum data	Sparse, robust to outliers and noise	Non-convex optimization
IFUSVM (intuitionistic fuzzy universum SVM)	Binary classification	Intuitionistic fuzzy weights with universum data	Robust to noise and outliers	Not effective for class imbalance
DC-IFSVM (dual center based intuitionistic fuzzy SVM)	Binary classification	Dual-center based fuzzy weights	Class-specific handling, robust to noise, outliers, and class imbalance	High computational complexity
CGFTSVM-ID (class probability and generalized bell fuzzy twin SVM for imbalanced data)	Imbalanced classification	Uses class probability and generalized bell (CPGB) fuzzy weights	Effective against for noise, outliers and imbalanced data	Solves QPP which is computationally expensive
BSFLSTSVM-ID (bell-shaped fuzzy least square twin SVM for imbalance data)	Imbalanced classification for large scaled data	Uses class probability and bell-shaped (CPBS) fuzzy weights	Solves using conjugate gradient, reduces QPP to linear system; efficient	Limited to vector form of data
ProDiSE-LSSVM (proximity and distance scoring ensemble for universal representation least squares SVM)	Large-scale classification	Pruning of data points via proximity and distance scoring	Linear time; scalable to massive data	Limited to shallow architecture
RLS-1SVM (robust least squares one-class support vector machine)	One-class classification / anomaly detection	Minimizes mean & variance of modeling error	Robust to non-Gaussian noise, non-sparse	High testing time
SRLS-1SVM (sparse robust least squares one-class support vector machine)	One-class classification	Sparse via pivoted Cholesky decomposition	Improves RLS-1SVM, sparse; faster training/testing	Contains approximation error
LSTSVR+ (least square twin support vector regression with privileged information)	Regression	Uses privileged data in LSTSVR framework	Better generalization	Requires privileged information

8.1 Conclusions

- [1] **Literature Review:** We provided an extensive review of hyperplane-based models, including support vector machine (SVM) and twin support vector machine (TWSVM). Our analysis explored various weighting strategies in the literature designed to mitigate the effects of noise, outliers, and class imbalance learning (CIL). Additionally, we examined the literature on large-scale data learning, one-class classification (OCC), and regression models that leverage hyperplane-based models, highlighting their key methodologies and contributions.
- [2] **Enhanced Pinball Loss Frameworks for Robust Classification and Generalization:** This chapter introduces two advanced models addressing limitations in pinball loss and enhancing classification performance. First, the flexible pinball loss SVM (FP-SVM) is proposed to overcome the restrictions of $\tau \in [0, 1]$ in traditional pinball loss and $\tau_1, \tau_2 \in [0, 1]$ in generalized pinball loss. FP-SVM extends the parameter range by allowing negative values for τ_1 and τ_2 , which control penalties for incorrectly and correctly classified data points, respectively. This extension improves the accuracy and generalization performance of FP-SVM compared to baseline models, with the constraint that τ_1 remains between $-\tau_2$ and 1 to maintain convexity. Extensive experiments on UCI and KEEL benchmark datasets statistically validate the superior performance of FP-SVM. Additionally, tests on the BreakHis dataset for breast cancer diagnosis demonstrate the model's effectiveness, achieving significant improvements over existing methods. Second, the universum twin support vector machine with truncated pinball loss (Tpin-UTWSVM) is introduced for classification tasks. Though models with pinball loss offer noise insensitivity and resampling stability, they lack the sparsity provided by hinge loss. The proposed Tpin-UTWSVM combines the

strengths of both loss functions, yielding a model that is sparse, noise-insensitive, robust to outliers, and stable to resampling, without compromising the theoretical properties of pinball loss. The non-convex, non-differentiable optimization problem of Tpin-UTWSVM is efficiently solved using the concave-convex procedure (CCCP) and successive over-relaxation (SOR) technique, which reduces memory usage during training. To evaluate Tpin-UTWSVM, experiments on 19 binary UCI datasets under varying noise levels demonstrate its superior average accuracy compared to baseline models, including pin-SVM, Pin-GTSVM, UTSVM, and TWSVM. Additionally, the influence of loss function parameters τ and s on model sparsity is thoroughly analyzed. In biomedical applications, the model's performance is validated using EEG data and Alzheimer's disease detection. Results show that Tpin-UTWSVM achieves the highest accuracy in classifying control versus Alzheimer's subjects, further substantiating its robustness and utility in practical domains.

- [3] **Fuzzy Weight-Based Robust Classification: Addressing Noise, Outliers, and Class Imbalance:** We proposed several novel models to address the challenges of noise, outliers, and class imbalance learning (CIL) in classification problems. First, we introduced the intuitionistic fuzzy universum support vector machine (IFUSVM), which assigns score to data points using intuitionistic fuzzy scheme (IFS), effectively reducing the influence of noise and outliers on the hyperplane construction. Additionally, the incorporation of universum data provides prior information about the data distribution, enhancing model robustness. However, IFUSVM is not effective for CIL. To address the limitation of IFUSVM, we proposed dual center-based intuitionistic fuzzy support vector machine (DC-IFSVM) and its least square variant, dual center-based intuitionistic fuzzy least square twin support vector machine (DC-IFLSTSVM). These models utilize dual centre-based IFS (DC-IFS) that account

for the distance of data points from the center of both classes. It also uses class-specific weighting for CIL. This approach enhances the handling of noise, outliers, and class imbalance while incorporating structural risk minimization through regularization terms. Numerical experiments on 54 UCI and KEEL datasets revealed that DC-IFLSTSVM consistently outperformed DC-IFSVM and baseline models. Furthermore, experiments on the BreakHis dataset for breast cancer diagnosis confirmed the practical utility of DC-IFLSTSVM, showcasing its superior classification performance. The existing IFS and DC-IFS are not effective for data distribution having majority of the samples at the class center or boundary. To handle this limitation, we introduced the class probability and generalized bell fuzzy twin SVM for imbalanced data (CGFTSVM-ID). This model employs a novel CPGB membership function that assigns weights to majority class samples by combining the generalized bell function, class probability, and imbalance ratio. The bell-shaped function effectively identifies potential outliers, while class probability and imbalance ratio ensure less importance to noise and a balanced contribution of samples, respectively. Minority class samples are assigned higher membership values to preserve their importance in learning. Experimental results on KEEL and UCI datasets demonstrated the superiority of CGFTSVM-ID, with additional experiments on ADNI datasets showcasing its efficacy in challenging real-world scenarios such as classifying mild cognitive impairment (MCI) and Alzheimer's disease (AD). The CGFTSVM-ID though effective against the resistance of noise and outliers, it solves QPP which has high complexity. Therefore, we developed the bell-shaped fuzzy least square twin support vector machine for imbalanced data (BSFLSTSVM-ID), which addresses the complexity of QPP solving in CGFTSVM-ID to system of linear equations. BSFLSTSVM-ID utilizes a novel membership function, class probability and bell-shaped (CPBS), which

provides robust resistance to noise, outliers, and imbalanced data. This function assigns varying weights to majority class samples by integrating class probability, bell-shaped function, and imbalance ratio. To resolve invertibility issues in solving the optimization problem in BSFLSTSVM-ID, we employed the conjugate gradient method, enhancing computational efficiency. Numerical experiments on 59 UCI and KEEL datasets, along with BreakHis and ADNI datasets, demonstrated the superior performance of BSFLSTSVM-ID over baseline models, reinforcing its effectiveness in handling complex real-world datasets.

- [4] **Breaking Complexity Barriers in Large-Scale Classification:** This chapter introduces ProDiSE: proximity and distance scoring ensemble for universal representation, an innovative approach for efficiently handling large-scale datasets by selecting a highly representative subset of data points. ProDiSE ensures a compact yet effective representation of data without compromising its overall integrity. The algorithm randomly selects a subset of training data and introduces the nearest neighbour proximity ratio (NNPR) to quantify the spatial relationships among points within the subset accurately. Building on NNPR, proximity-based scoring (PBS) and distance-based scoring (DBS) are employed to evaluate the local density and global distribution of each training sample. Samples with lower PBS and DBS values indicate points from dense clusters, while higher values signify outliers. Using these values, a representative score (RepScore) is computed for samples, with the most representative data points selected based on their lowest RepScores. Further, by integrating ProDiSE with LSSVM, we propose ProDiSE-LSSVM, a model that efficiently solves systems of linear equations while maintaining linear computational and memory complexity relative to the number of training samples, making it highly scalable for large datasets. Rigorous experimentation on 33 diverse real-world datasets, including those with up

to 581,012 samples, and 8 NDC datasets with up to 1 million instances, demonstrates ProDiSE-LSSVM's superior performance and its reduced training times. Statistical analysis of the real-world datasets confirms the superiority of the proposed model, and performance evaluations across different parameter settings highlight the proposed model's adaptability and sensitivity to different parameters.

- [5] **Robust and Sparse One-Class Classification Framework:** Least squares one-class SVM (LS-OCSVM) plays a significant role in OCC, however, it assumes the mean of the modeling error as constant and gets affected in the presence of noise, outliers. To address the challenge posed by noise and outliers in training data for one-class classification (OCC), this chapter introduces robust least squares one-class support vector machine (RLS-1SVM). The proposed RLS-1SVM detects anomalies in unknown non-linear systems characterized by random noise. The proposed RLS-1SVM, incorporates the distribution information of the modeling error, which makes it resilient to random noise. While RLS-1SVM solves systems of linear equations, it has increased training time due to more variables. Further, the non-sparse nature of RLS-1SVM contributes to heightened testing times. To address these challenges, we propose another model, namely sparse robust least squares one-class support vector machine (SRLS-1SVM). The proposed SRLS-1SVM leverages the pivoted Cholesky decomposition scheme to approximate the kernel matrix, thereby enhancing efficiency in both training and testing times compared to baseline models and the proposed RLS-1SVM. This is the first attempt in the literature to propose a sparse variant of LS-OCSVM tailored for batch learning. Furthermore, the chapter delves into the theoretical properties of the proposed RLS-1SVM and SRLS-1SVM, providing an upper bound for their generalization error. To assess the performance of the models, we conducted numerical experiments and statistical analysis on 33 UCI datasets, which

underscores the proposed models have superior performance. We conducted an in-depth experimental analysis on the impact of outliers and evaluated our models on the publicly available CIFAR-10 dataset as a practical application. The results on CIFAR-10, complemented by detailed statistical analysis, demonstrate the superiority of the proposed models. Notably, SRLS-1SVM achieves significantly reduced training and testing times while enhancing sparsity, underscoring its efficiency and effectiveness.

- [6] **Leveraging Privileged Information in Regression Model:** This chapter proposed least square twin support vector regression with privileged information (LSTSVR+) which integrates the paradigm of learning using privileged information (LUPI) in the framework of least square twin support vector regression. The proposed LUPI-based regression model constitutes the advantages of both. Most importantly, the LSTSVR+ provides a learning process, wherein the extra privileged data is taken into account as a teacher in the training phase. The proposed LSTSVR+ considers the structural risk minimization principle by incorporating regularization terms corresponding to both the regressor function as well as the correction function, thus preventing overfitting. Consequently, the LSTSVR+ has improved performance. Further LSTSVR+ solves a system of linear equations, which adds to the efficiency. The nonlinear variant of the proposed LSTSVR+ deals with the more intricate nonlinear relationships present in high-dimensional input data. We also established a generalization error bound based on the Rademacher complexity of the proposed model. Furthermore, to assess the performance of the proposed LSTSVR+, we conducted the experiments over artificially generated datasets and 21 real-world datasets. The average RMSE value of the proposed LSTSVR+ over real-world data is the smallest among the baseline models. Thus, the numerical results and its statistical analysis demonstrate the superiority of the proposed model. As an application of the proposed LSTSVR+, we conducted

the experiments on time-series datasets, which infers the proposed model is the best-performing one.

8.2 Future research directions

- [1] The works presented in this thesis primarily focus on binary classification. However, these frameworks can be further extended to handle multi-class classification [187] and multi-task learning [188], enabling the models to effectively address more complex problems involving diverse categories and tasks. For multi-class classification, the existing models can be adapted by incorporating techniques such as one-vs-all or one-vs-one, allowing the framework to distinguish between multiple classes while maintaining the advantages of the current approach. Additionally, multi-task learning can be integrated into the model, enabling it to address multiple related tasks simultaneously.
- [2] While the models in this thesis currently focus on vector-based input, they can be expanded to handle matrix input data by incorporating support matrix machines and their variants, as explored in [189] and [190]. This expansion would make the models even more versatile, enabling them to tackle a wider range of real-world problems.
- [3] While ProDiSE-LSSVM offers a powerful and scalable approach for classification tasks, its shallow architecture may limit its ability to model complex, high-dimensional, or sequential data effectively. Future research directions focused on incorporating the ProDiSE algorithm for regression tasks, and for deep learning architectures. These directions would address the limitations of the shallow model and broaden the applicability of ProDiSE to more intricate and diverse problems.
- [4] One can focus on developing more efficient and automated hyperparameter tuning

strategies to enhance model performance while reducing computational costs. The models proposed in this thesis involve various hyperparameters, and selecting the optimal set can be time-consuming and computationally expensive. To address this, future research could explore adaptive strategies that automatically select the most suitable hyperparameters based on the specific data and task. Techniques such as Bayesian optimization [191] or genetic algorithms could be integrated to streamline the tuning process, making it more efficient and scalable. This would improve the performance of the models and make them more applicable to large-scale, real-world problems, ultimately enhancing their overall efficiency and scalability.

- [5] One can focus on adapting hyperplane-based models of the thesis for online [192] and incremental learning, enabling them to continuously update and refine their parameters as new data becomes available. This approach would eliminate the need for complete retraining, enhancing computational efficiency and making the models more suitable for dynamic environments where data is constantly evolving. By integrating online learning capabilities, these models could offer real-time adaptability, improving their performance and scalability in practical, real-world applications.
- [6] The models presented in this thesis primarily focus on supervised learning, which may limit their applicability in scenarios with limited labeled data. Extending the current framework to incorporate semi-supervised and unsupervised learning could enhance model performance by leveraging both labeled and unlabeled data, or entirely unlabeled data, to discover underlying patterns. By integrating hyperplane-based methods into these learning paradigms, it would be possible to improve generalization, reduce reliance on labeled data, and potentially achieve better performance in real-world applications where labeled data is scarce or expensive to obtain.

Bibliography

- [1] C. Zhang, Y. Lu, Study on artificial intelligence: The state of the art and future prospects, *Journal of Industrial Information Integration* 23 (2021) 100224.
- [2] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, *The journal of machine learning research* 15 (2014) 3133–3181.
- [3] V. N. Vapnik, *The nature of statistical learning, Theory* (1995).
- [4] S.-C. Wang, S.-C. Wang, *Artificial neural network, Interdisciplinary computing in java programming* (2003) 81–100.
- [5] L. Zhang, P. N. Suganthan, A survey of randomized algorithms for training neural networks, *Information Sciences* 364 (2016) 146–155.
- [6] S. Wang, X. Yang, Y. Zhang, P. Phillips, J. Yang, T.-F. Yuan, Identification of green, oolong and black teas in china via wavelet packet entropy and fuzzy support vector machine, *Entropy* 17 (2015) 6663–6682.
- [7] S. Wang, Y. Li, Y. Shao, C. Cattani, Y. Zhang, S. Du, Detection of dendritic spines using wavelet packet entropy and fuzzy support vector machine, *CNS & Neurological Disorders-Drug Targets (Formerly Current Drug Targets-CNS & Neurological Disorders)* 16 (2017) 116–121.

- [8] B. Richhariya, M. Tanveer, EEG signal classification using universum support vector machine, *Expert Systems with Applications* 106 (2018) 169–182.
- [9] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, *Statistics and Computing* 14 (2004) 199–222.
- [10] R. Chen, C.-Y. Liang, W.-C. Hong, D.-X. Gu, Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm, *Applied Soft Computing* 26 (2015) 435–443.
- [11] Z. Zhang, W.-C. Hong, Application of variational mode decomposition and chaotic grey wolf optimizer with support vector regression for forecasting electric loads, *Knowledge-Based Systems* 228 (2021) 107297.
- [12] V. Ranković, N. Grujović, D. Divac, N. Milivojević, Development of support vector regression identification model for prediction of dam structural behaviour, *Structural Safety* 48 (2014) 33–39.
- [13] J. Bi, T. Zhang, Support vector classification with input data uncertainty, *Advances in Neural Information Processing Systems* 17 (2004).
- [14] X. Zhang, Using class-center vectors to build support vector machines, in: *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop* (Cat. No. 98TH8468), IEEE, 1999, pp. 3–11.
- [15] Y. Ma, L. Li, X. Huang, S. Wang, Robust support vector machine using least median loss penalty, *IFAC Proceedings Volumes* 44 (2011) 11208–11213.
- [16] X. Yang, Q. Song, Y. Wang, A weighted support vector machine for data classification, *International Journal of Pattern Recognition and Artificial Intelligence* 21 (2007) 961–976.

- [17] X. Huang, L. Shi, J. A. Suykens, Support vector machine classifier with pinball loss, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2013) 984–997.
- [18] E. Osuna, F. Girosi, Reducing the run-time complexity of support vector machines, in: *International Conference on Pattern Recognition* (submitted), Citeseer, 1998.
- [19] J. Platt, Sequential minimal optimization: A fast algorithm for training support vector machines (1998).
- [20] T. Joachims, 11 making large-scale support vector machine learning practical, in: *Advances in Kernel Methods: Support Vector Learning*, MIT press, 1999, p. 169.
- [21] O. L. Mangasarian, E. W. Wild, Multisurface proximal support vector machine classification via generalized eigenvalues, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2005) 69–74.
- [22] Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (2007) 905–910.
- [23] Y.-H. Shao, C.-H. Zhang, X.-B. Wang, N.-Y. Deng, Improvements on twin support vector machines, *IEEE Transactions on Neural Networks* 22 (2011) 962–968.
- [24] Y.-H. Shao, N.-Y. Deng, A coordinate descent margin based-twin support vector machine for classification, *Neural networks* 25 (2012) 114–121.
- [25] J. Weston, R. Collobert, F. Sinz, L. Bottou, V. Vapnik, Inference with the universum, in: *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 1009–1016.

- [26] Z. Qi, Y. Tian, Y. Shi, Twin support vector machine with universum data, *Neural Networks* 36 (2012) 112–119.
- [27] B. Richhariya, A. Sharma, M. Tanveer, Improved universum twin support vector machine, in: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2018, pp. 2045–2052.
- [28] Y. Xu, M. Chen, G. Li, Least squares twin support vector machine with universum data for classification, *International Journal of Systems Science* 47 (2016) 3637–3645.
- [29] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, *Neural Processing Letters* 9 (1999) 293–300.
- [30] R. Cheng, D. Chen, X. Ma, Y. Cheng, H. Cheng, Intelligent quantitative safety monitoring approach for atp using lssvm and probabilistic model checking considering imperfect fault coverage, *IEEE Transactions on Intelligent Transportation Systems* (2023, [10.1109/TITS.2023.3332348](https://doi.org/10.1109/TITS.2023.3332348)).
- [31] V. Mitra, C.-J. Wang, S. Banerjee, Text classification: A least square support vector machine approach, *Applied Soft Computing* 7 (2007) 908–914.
- [32] T. Van Gestel, J. A. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, J. Vandewalle, Benchmarking least squares support vector machine classifiers, *Machine Learning* 54 (2004) 5–32.
- [33] M. A. Kumar, M. Gopal, Least squares twin support vector machines for pattern classification, *Expert Systems with Applications* 36 (2009) 7535–7543.
- [34] L. Jiao, L. Bo, L. Wang, Fast sparse approximation for least squares support vector machine, *IEEE Transactions on Neural Networks* 18 (2007) 685–697.

- [35] B. J. De Kruif, T. J. De Vries, Pruning error minimization in least squares support vector machines, *IEEE Transactions on Neural Networks* 14 (2003) 696–702.
- [36] A. Kuh, P. De Wilde, Comments on pruning error minimization in least squares support vector machines, *IEEE Transactions on Neural Networks* 18 (2007) 606–609.
- [37] X. Zeng, X.-w. Chen, SMO-based pruning methods for sparse least squares support vector machines, *IEEE Transactions on Neural Networks* 16 (2005) 1541–1546.
- [38] Y.-H. Shao, C.-N. Li, L.-W. Huang, Z. Wang, N.-Y. Deng, Y. Xu, Joint sample and feature selection via sparse primal and dual LSSVM, *Knowledge-Based Systems* 185 (2019) 104915.
- [39] X. Chen, Z. Lai, Jointly sparse least square support vector machine, *Computers and Electrical Engineering* 101 (2022) 108078.
- [40] Q. Tang, G. Li, Sparse l_0 -norm least squares support vector machine with feature selection, *Information Sciences* 670 (2024) 120591.
- [41] J. A. Suykens, L. Lukas, J. Vandewalle, Sparse approximation using least squares support vector machines, in: *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, IEEE, 2000, pp. 757–760.
- [42] Y. Ma, X. Liang, G. Sheng, J. T. Kwok, M. Wang, G. Li, Noniterative sparse LS-SVM based on globally representative point selection, *IEEE Transactions on Neural Networks and Learning Systems* 32 (2020) 788–798.
- [43] M. Hanmandlu, Mamta, A new entropy function and a classifier for thermal face recognition, *Engineering Applications of Artificial Intelligence* 36 (2014).

- [44] C.-F. Lin, S.-D. Wang, Fuzzy support vector machines, *IEEE Transactions on Neural Networks* 13 (2002) 464–471.
- [45] R. Batuwita, V. Palade, FSVM-CIL: fuzzy support vector machines for class imbalance learning, *IEEE Transactions on Fuzzy Systems* 18 (2010) 558–571.
- [46] X. Tao, Q. Li, C. Ren, W. Guo, Q. He, R. Liu, J. Zou, Affinity and class probability-based fuzzy support vector machine for imbalanced data sets, *Neural Networks* 122 (2020) 289–307.
- [47] D. M. Tax, R. P. Duin, Support vector data description, *Machine Learning* 54 (2004) 45–66.
- [48] M. Tanveer, A. Tiwari, R. Choudhary, M. Ganaie, Large-scale pinball twin support vector machines, *Machine Learning* (2021) 1–24. doi:<https://doi.org/10.1007/s10994-021-06061-z>.
- [49] S. Rezvani, X. Wang, F. Pourpanah, Intuitionistic fuzzy twin support vector machines, *IEEE Transactions on Fuzzy Systems* 27 (2019) 2140–2151.
- [50] M. A. Ganaie, A. Kumari, A. K. Malik, M. Tanveer, EEG signal classification using improved intuitionistic fuzzy twin support vector machines, *Neural Computing and Applications* (2022) 1–17.
- [51] S. Rezvani, X. Wang, Intuitionistic fuzzy twin support vector machines for imbalanced data, *Neurocomputing* 507 (2022) 16–25.
- [52] S. S. Khan, M. G. Madden, One-class classification: taxonomy of study and review of techniques, *The Knowledge Engineering Review* 29 (2014) 345–374.

- [53] H.-J. Xing, Y.-J. Liu, Z.-C. He, Robust sparse coding for one-class classification based on correntropy and logarithmic penalty function, *Pattern Recognition* 111 (2021) 107685.
- [54] S. Chen, X. Ouyang, F. Luo, Ensemble one-class support vector machine for sea surface target detection based on K-means clustering, *Remote Sensing* 16 (2024) 2401.
- [55] M. Kim, J. Kim, J. Yu, J. K. Choi, Active anomaly detection based on deep one-class classification, *Pattern Recognition Letters* 167 (2023) 18–24.
- [56] M. A. Pimentel, D. A. Clifton, L. Clifton, L. Tarassenko, A review of novelty detection, *Signal Processing* 99 (2014) 215–249.
- [57] S. Alam, S. K. Sonbhadra, S. Agarwal, P. Nagabhushan, One-class support vector classifiers: A survey, *Knowledge-Based Systems* 196 (2020) 105754.
- [58] M. Akhtar, M. Tanveer, M. Arshad, A. D. N. Initiative, Advancing supervised learning with the wave loss function: A robust and smooth approach, *Pattern Recognition* (2024) 110637.
- [59] D. M. Tax, R. P. Duin, Support vector domain description, *Pattern Recognition Letters* 20 (1999) 1191–1199.
- [60] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, Support vector method for novelty detection, *Advances in Neural Information Processing Systems* 12 (1999).
- [61] F. Zhang, L. J. O'Donnell, Support vector regression, in: *Machine learning*, Elsevier, 2020, pp. 123–140.

- [62] X. Peng, TSVR: an efficient twin support vector machine for regression, *Neural Networks* 23 (2010) 365–372.
- [63] Y.-H. Shao, C.-H. Zhang, Z.-M. Yang, L. Jing, N.-Y. Deng, An ε -twin support vector machine for regression, *Neural Computing and Applications* 23 (2013) 175–185.
- [64] Y. Xu, L. Wang, A weighted twin support vector regression, *Knowledge-Based Systems* 33 (2012) 92–101.
- [65] X. Peng, D. Xu, J. Shen, A twin projection support vector machine for data regression, *Neurocomputing* 138 (2014) 131–141.
- [66] Y.-P. Zhao, J. Zhao, M. Zhao, Twin least squares support vector regression, *Neurocomputing* 118 (2013) 225–236.
- [67] H.-j. Huang, S.-f. Ding, Z.-z. Shi, Primal least squares twin support vector regression, *Journal of Zhejiang University SCIENCE C* 14 (2013) 722–732.
- [68] H. Huang, X. Wei, Y. Zhou, A sparse method for least squares twin support vector regression, *Neurocomputing* 211 (2016) 150–158.
- [69] H. Huang, X. Wei, Y. Zhou, An overview on twin support vector regression, *Neurocomputing* 490 (2022) 80–92.
- [70] V. Vapnik, A. Vashist, A new learning paradigm: Learning using privileged information, *Neural Networks* 22 (2009) 544–557.
- [71] C. Gautam, A. Tiwari, M. Tanveer, AEKOC+: Kernel ridge regression-based auto-encoder for one-class classification using privileged information, *Cognitive Computation* 12 (2020) 412–425.

- [72] J. Tang, Y. Tian, D. Liu, G. Kou, Coupling privileged kernel method for multi-view learning, *Information Sciences* 481 (2019) 110–127.
- [73] X. Huang, L. Shi, J. A. Suykens, Solution path for pin-SVM classifiers with positive and negative τ values, *IEEE Transactions on Neural Networks and Learning Systems* 28 (2016) 1584–1593.
- [74] X. Shen, L. Niu, Z. Qi, Y. Tian, Support vector machine classifier with truncated pinball loss, *Pattern Recognition* 68 (2017) 199–210.
- [75] C. Gautam, A. Tiwari, M. Tanveer, KOC+: Kernel ridge regression based one-class classification using privileged information, *Information Sciences* 504 (2019) 324–333.
- [76] A. Kumari, M. Akhtar, M. Tanveer, M. Arshad, Diagnosis of breast cancer using flexible pinball loss support vector machine, *Applied Soft Computing* 157 (2024) 111454.
- [77] C. Cortes, V. Vapnik, Support-vector networks, *Machine Learning* 20 (1995) 273–297.
- [78] P. Anand, R. Rastogi, S. Chandra, Improvement over pinball loss support vector machine, *arXiv preprint arXiv:2106.01109* (2021).
- [79] A. Kumari, M. Tanveer, Dual center based intuitionistic fuzzy plane based classifiers, in: *2024 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2024, pp. 1–8.
- [80] Y.-S. Choi, Least squares one-class support vector machine, *Pattern Recognition Letters* 30 (2009) 1236–1240.

- [81] G. H. Golub, C. F. Van Loan, Matrix computations, JHU press, 2013.
- [82] A. Kumari, M. Tanveer, A. D. N. Initiative, Universum twin support vector machine with truncated pinball loss, *Engineering Applications of Artificial Intelligence* 123 (2023) 106427.
- [83] A. Kumari, M. Ganaie, M. Tanveer, Intuitionistic fuzzy universum support vector machine, in: *International Conference on Neural Information Processing*, Springer, 2022, pp. 236–247.
- [84] A. Kumari, M. Tanveer, C.-T. Lin, Class probability and generalized bell fuzzy twin svm for imbalanced data, *IEEE Transactions on Fuzzy Systems* (2024, [10.1109/TFUZZ.2024.3366936](#)).
- [85] A. Kumari, M. Tanveer, C.-T. Lin, Bell-shaped fuzzy least square twin svm with biomedical applications, *IEEE Transactions on Fuzzy Systems* (2024).
- [86] A. Kumari, M. Tanveer, Lstsvr+: Least square twin support vector regression with privileged information, *Engineering Applications of Artificial Intelligence* 136 (2024) 108964.
- [87] J. Han, J. Pei, H. Tong, *Data mining: concepts and techniques*, Morgan kaufmann, 2022.
- [88] L. I. Kuncheva, J. J. Rodriguez, Classifier ensembles with a random linear oracle, *IEEE Transactions on Knowledge and Data Engineering* 19 (2007) 500–508.
- [89] S. R. Gunn, *Support vector machines for classification and regression*, ISIS Technical Report 14 (1998) 5–16.

- [90] V. Vapnik, The nature of statistical learning theory, Springer Science & Business Media, 1999.
- [91] G. Guo, S. Z. Li, K. Chan, Face recognition by support vector machines, in: Proceedings fourth IEEE International Conference on Automatic Face and Gesture Recognition (cat. no. PR00580), IEEE, 2000, pp. 196–201.
- [92] C. Bahlmann, B. Haasdonk, H. Burkhardt, Online handwriting recognition with support vector machines-A kernel approach, in: Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition, IEEE, 2002, pp. 49–54.
- [93] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: European Conference on Machine Learning, Springer, 1998, pp. 137–142.
- [94] N. H. Sweilam, A. Tharwat, N. A. Moniem, Support vector machine for diagnosis cancer disease: A comparative study, *Egyptian Informatics Journal* 11 (2010) 81–92.
- [95] I. Steinwart, A. Christmann, Estimating conditional quantiles with the help of the pinball loss, *Bernoulli* 17 (2011) 211–225.
- [96] L. Yang, H. Dong, Support vector machine with truncated pinball loss and its application in pattern recognition, *Chemometrics and Intelligent Laboratory Systems* 177 (2018) 89–99.
- [97] X. Huang, L. Shi, J. A. Suykens, Sequential minimal optimization for SVM with pinball loss, *Neurocomputing* 149 (2015) 1596–1603.
- [98] R. Rastogi, A. Pal, S. Chandra, Generalized pinball loss SVMs, *Neurocomputing* 322 (2018) 151–165.

- [99] M. Tanveer, T. Rajani, R. Rastogi, Y.-H. Shao, M. Ganaie, Comprehensive review on twin support vector machines, *Annals of Operations Research* (2022). <https://doi.org/10.1007/s10479-022-04575-w>) 1–46.
- [100] B. Mei, Y. Xu, Multi-task least squares twin support vector machine for classification, *Neurocomputing* 338 (2019) 26–33.
- [101] M. Tanveer, A. Sharma, P. N. Suganthan, General twin support vector machine with pinball loss function, *Information Sciences* 494 (2019) 311–327.
- [102] J. A. Goguen, La zadeh. fuzzy sets. information and control, vol. 8 (1965), pp. 338–353.-la zadeh. similarity relations and fuzzy orderings. information sciences, vol. 3 (1971), pp. 177–200., *The Journal of Symbolic Logic* 38 (1973) 656–657.
- [103] H.-J. Zimmermann, Fuzzy set theory, *Wiley interdisciplinary reviews: computational statistics* 2 (2010) 317–332.
- [104] D. Dubois, H. Prade, *Fundamentals of fuzzy sets*, volume 7, Springer Science & Business Media, 2012.
- [105] C. Kahraman, *Fuzzy applications in industrial engineering*, volume 201, Springer, 2006.
- [106] M. Yong, X. Zheng, Y. Zheng, S. Youxian, W. Zheng, Fault diagnosis based on fuzzy support vector machine with parameter tuning and feature selection, *Chinese Journal of Chemical Engineering* 15 (2007) 233–239.
- [107] B. K. Sriwastava, S. Basu, U. Maulik, Predicting protein-protein interaction sites with a novel membership based fuzzy svm classifier, *IEEE/ACM transactions on computational biology and bioinformatics* 12 (2015) 1394–1404.

- [108] V. E. Mirzakhonov, Value of fuzzy logic for data mining and machine learning: A case study, *Expert Systems with Applications* 162 (2020) 113781.
- [109] M. Ha, C. Wang, J. Chen, The support vector machine based on intuitionistic fuzzy number and kernel function, *Soft Computing* 17 (2013) 635–641.
- [110] M. Tanveer, M. Ganaie, A. Bhattacharjee, C. Lin, Intuitionistic fuzzy weighted least squares twin SVMs, *IEEE Transactions on Cybernetics* (2022,doi=10.1109/TCYB.2022.3165879).
- [111] Z. Liang, L. Zhang, Intuitionistic fuzzy twin support vector machines with the insensitive pinball loss, *Applied Soft Computing* 115 (2022) 108231.
- [112] M. Ganaie, A. Kumari, A. Girard, J. Kasa-Vubu, M. Tanveer, A. D. N. Initiative, Diagnosis of Alzheimer’s disease via intuitionistic fuzzy least squares twin SVM, *Applied Soft Computing* 149 (2023) 110899.
- [113] S. Rezvani, J. Wu, Handling multi-class problem by intuitionistic fuzzy twin support vector machines based on relative density information, *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023. [10.1109/TPAMI.2023.3310908](https://doi.org/10.1109/TPAMI.2023.3310908)).
- [114] L. Zhang, Q. Jin, S. Fan, D. Liu, A novel dual-center based intuitionistic fuzzy twin bounded large margin distribution machines, *IEEE Transactions on Fuzzy Systems* (2023).
- [115] M. A. Ganaie, M. Tanveer, C. T. Lin, Large-scale fuzzy least squares twin SVMs for class imbalance learning, *IEEE Transactions on Fuzzy Systems* 30 (2022) 4815–4827.
- [116] J. Laurikkala, Improving identification of difficult small classes by balancing class

- distribution, in: *Artificial Intelligence in Medicine: 8th Conference in Europe, AIME*, Springer, 2001, pp. 63–66.
- [117] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, *Journal of Artificial Intelligence Research* 16 (2002) 321–357.
- [118] S. Rezvani, X. Wang, A broad review on class imbalance learning techniques, *Applied Soft Computing* (2023). <https://doi.org/10.1016/j.asoc.2023.110415>) 110415.
- [119] X. Shao, K. Wu, B. Liao, et al., Single directional SMO algorithm for least squares support vector machines, *Computational Intelligence and Neuroscience* 2013 (2013).
- [120] Z. Akata, F. Perronnin, Z. Harchaoui, C. Schmid, Good practice in large-scale learning for image classification, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2013) 507–520.
- [121] J. Zhao, X. Chen, Z. Xiong, Z.-J. Zha, F. Wu, Graph representation learning for large-scale neuronal morphological analysis, *IEEE Transactions on Neural Networks and Learning Systems* 35 (2022) 5461–5472.
- [122] G. Chryssolouris, M. Lee, A. Ramsey, Confidence interval prediction for neural network models, *IEEE Transactions on Neural Networks* 7 (1996) 229–232.
- [123] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *The Journal of Machine Learning Research* 7 (2006) 1–30.
- [124] D. Makmuang, W. Ratiphaphongthon, R. Wangkeeree, Smooth support vector machine with generalized pinball loss for pattern classification, *The Journal of Supercomputing* (2023) 1–23.

- [125] A. K. Malik, M. Ganaie, M. Tanveer, P. Suganthan, A. D. N. I. Initiative, et al., Alzheimer's disease diagnosis via intuitionistic fuzzy random vector functional link network, *IEEE Transactions on Computational Social Systems* (2022).
- [126] D. Dua, C. Graff, UCI machine learning repository (2017).
- [127] J. Derrac, S. Garcia, L. Sanchez, F. Herrera, KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, *J. Mult. Valued Log. Soft Comput* 17 (2015) 255–287.
- [128] L. Zhang, P. N. Suganthan, A comprehensive evaluation of random vector functional link networks, *Information Sciences* 367 (2016) 1094–1105.
- [129] P. B. Nemenyi, *Distribution-free multiple comparisons.*, Princeton University, 1963.
- [130] F. A. Spanhol, L. S. Oliveira, C. Petitjean, L. Heutte, A dataset for breast cancer histopathological image classification, *IEEE Transactions on Biomedical Engineering* 63 (2015) 1455–1462.
- [131] C. Gautam, P. K. Mishra, A. Tiwari, B. Richhariya, H. M. Pandey, S. Wang, M. Tanveer, for the Alzheimer's Disease Neuroimaging Initiative, Minimum variance-embedded deep kernel regularized least squares method for one-class classification and its applications to biomedical data, *Neural Networks* 123 (2020) 191–216.
- [132] I. E. Yen, N. Peng, P.-W. Wang, S.-D. Lin, On convergence rate of concave-convex procedure, in: *Proceedings of the NIPS 2012 Optimization Workshop*, 2012, pp. 31–35.
- [133] Y. Zhang, J. Duchi, M. Wainwright, Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates, *The Journal of Machine Learning Research* 16 (2015) 3299–3340.

- [134] B. Richhariya, M. Tanveer, A. D. N. Initiative, A fuzzy universum least squares twin support vector machine (FULSTSVM), *Neural Computing and Applications* (2021) 1–12. doi:<https://doi.org/10.1007/s00521-021-05721-4>.
- [135] O. L. Mangasarian, D. R. Musicant, Successive overrelaxation for support vector machines, *IEEE Transactions on Neural Networks* 10 (1999) 1032–1037.
- [136] D. J. Newman, UCI repository of machine learning database, <http://www.ics.uci.edu/~mllearn/MLRepository.html> (1998).
- [137] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, C. E. Elger, Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, *Physical Review E* 64 (2001) 061907.
- [138] B. Richhariya, M. Tanveer, A reduced universum twin support vector machine for class imbalance learning, *Pattern Recognition* 102 (2020) 107150.
- [139] H. Wang, Y. Xu, Z. Zhou, Twin-parametric margin support vector machine with truncated pinball loss, *Neural Computing and Applications* 33 (2021) 3781–3798.
- [140] Z. Wang, Y.-H. Shao, L. Bai, N.-Y. Deng, Twin support vector machine for clustering, *IEEE Transactions on Neural Networks and Learning Systems* 26 (2015) 2583–2588.
- [141] S. Theodoridis, K. Koutroumbas, Feature selection, *Pattern recognition* (2009) 261–322.
- [142] S. Kullback, *Information theory and statistics*, Courier Corporation, 1997.
- [143] F. Wilcoxon, *Individual comparisons by ranking methods*, Springer, 1992.

- [144] B. Richhariya, M. Tanveer, A. H. Rashid, A. D. N. Initiative, Diagnosis of alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE), *Biomedical Signal Processing and Control* 59 (2020) 101903.
- [145] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, *IEEE Transactions on Neural Networks* 13 (2002) 415–425.
- [146] P. Murphy, D. Aha, UCI repository of machine learning databases available at: <http://www.ics.uci.edu/mllearn, MLRepository.html> (1992).
- [147] B. S. Raghuwanshi, S. Shukla, Class-specific extreme learning machine for handling binary class imbalance problem, *Neural Networks* 105 (2018) 206–217.
- [148] J. Nocedal, S. J. Wright, *Conjugate Gradient Methods*, Springer New York, New York, NY, 2006, pp. 101–134. doi:[10.1007/978-0-387-40065-5_5](https://doi.org/10.1007/978-0-387-40065-5_5).
- [149] D. Tomar, S. Agarwal, Twin support vector machine: a review from 2007 to 2014, *Egyptian Informatics Journal* 16 (2015) 55–69.
- [150] J. R. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, 1994.
- [151] B. Richhariya, M. Tanveer, A robust fuzzy least squares twin support vector machine for class imbalance learning, *Applied Soft Computing* 71 (2018) 418–432.
- [152] W. Gao, Z.-H. Zhou, On the doubt about margin explanation of boosting, *Artificial Intelligence* 203 (2013) 1–18.
- [153] A. Dorzhigulov, A. P. James, Generalized bell-shaped membership function generation circuit for memristive neural networks, in: *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2019, pp. 1–5.

- [154] B. B. Hazarika, D. Gupta, P. Borah, Fuzzy twin support vector machine based on affinity and class probability for class imbalance learning, *Knowledge and Information Systems* (2023) 1–30.
- [155] J.-S. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics* 23 (1993) 665–685.
- [156] S. Rezvani, X. Wang, Class imbalance learning using fuzzy ART and intuitionistic fuzzy twin support vector machines, *Information Sciences* 578 (2021) 659–682.
- [157] D. Musicant, NDC: normally distributed clustered datasets, 1998, ????
- [158] H.-J. Xing, W.-T. Liu, Robust adaboost based ensemble of one-class support vector machines, *Information Fusion* 55 (2020) 45–58.
- [159] C. Lian, M. Liu, J. Zhang, D. Shen, Hierarchical fully convolutional network for joint atrophy localization and alzheimer’s disease diagnosis using structural MRI, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2018) 880–893.
- [160] E. Westman, J.-S. Muehlboeck, A. Simmons, Combining MRI and CSF measures for classification of Alzheimer’s disease and prediction of mild cognitive impairment conversion, *Neuroimage* 62 (2012) 229–238.
- [161] B. Schölkopf, R. Herbrich, A. J. Smola, A generalized representer theorem, in: *International Conference on Computational Learning Theory*, Springer, 2001, pp. 416–426.
- [162] S. S. Keerthi, O. Chapelle, D. DeCoste, K. P. Bennett, E. Parrado-Hernández, Building support vector machines with reduced classifier complexity., *Journal of Machine Learning Research* 7 (2006).

- [163] O. Chapelle, Training a support vector machine in the primal (2007, <https://doi.org/10.7551/mitpress/7496.001.0001>).
- [164] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2022.
- [165] V. Strassen, Gaussian elimination is not optimal, *Numerische Mathematik* 13 (1969) 354–356.
- [166] S. Zhou, Sparse LSSVM in primal using Cholesky factorization for large-scale problems, *IEEE Transactions on Neural Networks and Learning Systems* 27 (2015) 783–795.
- [167] M. Akhtar, M. Tanveer, M. Arshad, RoBoSS: a robust, bounded, sparse, and smooth loss function for supervised learning, *arXiv preprint arXiv:2309.02250* (2023).
- [168] D. R. Musicant, NDC: normally distributed clustered datasets, 1998. www.cs.wisc.edu/dmi/svm/ndc/.
- [169] F. Zhu, J. Yang, C. Gao, S. Xu, N. Ye, T. Yin, A weighted one-class support vector machine, *Neurocomputing* 189 (2016) 1–10.
- [170] H. Harbrecht, M. Peters, R. Schneider, On the low-rank approximation by the pivoted Cholesky decomposition, *Applied Numerical Mathematics* 62 (2012) 428–440.
- [171] P. Drineas, M. W. Mahoney, N. Cristianini, On the Nystrom method for approximating a gram matrix for improved kernel-based learning., *Journal of Machine Learning Research* 6 (2005).
- [172] X. Lu, W. Liu, C. Zhou, M. Huang, Robust least-squares support vector machine

- with minimization of mean and variance of modeling error, *IEEE Transactions on Neural Networks and Learning Systems* 29 (2017) 2909–2920.
- [173] H. Xu, S. Mannor, Robustness and generalization, *Machine Learning* 86 (2012) 391–423.
- [174] A. N. Kolmogorov, ε -entropy and ε -capacity of sets in function space, *Amer. Math. Soc. Translation, Ser. 2* (1961) 277–364.
- [175] Y. Zhang, J. Duchi, M. Wainwright, Divide and conquer kernel ridge regression, in: *Conference on Learning Theory*, PMLR, 2013, pp. 592–617.
- [176] S. Yin, X. Zhu, C. Jing, Fault detection based on a robust one class support vector machine, *Neurocomputing* 145 (2014) 263–268.
- [177] H.-J. Xing, M. Ji, Robust one-class support vector machine with rescaled hinge loss function, *Pattern Recognition* 84 (2018) 152–164.
- [178] Y. Zheng, S. Wang, B. Chen, Multikernel correntropy based robust least squares one-class support vector machine, *Neurocomputing* 545 (2023) 126324.
- [179] C. Gautam, R. Balaji, K. Sudharsan, A. Tiwari, K. Ahuja, Localized multiple kernel learning for anomaly detection: One-class classification, *Knowledge-Based Systems* 165 (2019) 241–252.
- [180] P.-B. Zhang, Z.-X. Yang, A new learning paradigm for random vector functional-link network: RVFL+, *Neural Networks* 122 (2020) 94–105.
- [181] C. J. Burges, D. Crisp, Uniqueness of the svm solution, *Advances in Neural Information Processing Systems* 12 (1999).

- [182] P. L. Bartlett, S. Mendelson, Rademacher and gaussian complexities: Risk bounds and structural results, *Journal of Machine Learning Research* 3 (2002) 463–482.
- [183] R. Khemchandani, K. Goyal, S. Chandra, TWSVR: regression via twin support vector machine, *Neural Networks* 74 (2016) 14–21.
- [184] Y. Ye, Z. Xu, J. Zhang, W. Chen, Y. Shao, Twin support vector quantile regression, *Expert Systems with Applications* 237 (2024) 121239.
- [185] M. Tanveer, K. Shubham, A regularization on lagrangian twin support vector regression, *International Journal of Machine Learning and Cybernetics* 8 (2017) 807–821.
- [186] M. Tanveer, K. Shubham, M. Aldhaifallah, S.-S. Ho, An efficient regularized k-nearest neighbor based weighted twin support vector regression, *Knowledge-Based Systems* 94 (2016) 70–87.
- [187] S. Dong, Multi class svm algorithm with active learning for network traffic classification, *Expert Systems with Applications* 176 (2021) 114885.
- [188] Y. Zhang, Q. Yang, A survey on multi-task learning, *IEEE Transactions on Knowledge and Data Engineering* 34 (2021) 5586–5609.
- [189] L. Luo, Y. Xie, Z. Zhang, W.-J. Li, Support matrix machines, in: *International conference on machine learning*, PMLR, 2015, pp. 938–947.
- [190] A. Kumari, M. Akhtar, R. Shah, M. Tanveer, Support matrix machine: A review, *Neural Networks* (2024) 106767.
- [191] X. Wang, Y. Jin, S. Schmitt, M. Olhofer, Recent advances in bayesian optimization, *ACM Computing Surveys* 55 (2023) 1–36.

- [192] S. C. Hoi, D. Sahoo, J. Lu, P. Zhao, Online learning: A comprehensive survey, *Neurocomputing* 459 (2021) 249–289.