Ensembles of decision tree and random vector functional link neural network for classification problems

Ph.D. Thesis

by

Mudasir Ahmad Ganaie

Department of Mathematics

INDIAN INSTITUTE OF TECHNOLOGY INDORE

January 2022

Ensembles of decision tree and random vector functional link neural network for classification problems

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

in partial fulfillment of the requirements for

the award of the degree

of

DOCTOR OF PHILOSOPHY

by

Mudasir Ahmad Ganaie

Department of Mathematics

INDIAN INSTITUTE OF TECHNOLOGY INDORE

January 2022

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Ensembles of decision tree and random vector functional link neural network for classification 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 July 2019 to January 2022 under the supervision of Dr. M. Tanveer, Associate Professor, Indian Institute of Technology Indore, Indore, India.

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.

mudasir

14-June-2022

Signature of the Student with Date

(Mudasir Ahmad Ganaie)

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

14/06/2022

Signature of Thesis Supervisor with Date

(Dr. M. Tanveer)

Mudasir Ahmad Ganaie has successfully given his Ph.D. Oral Examination held on 10/06/2022

14/06/2022

Signature of Thesis Supervisor with Date

(Dr. M. Tanveer)

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed by making this time as learnable, enjoyable, and bearable. At first, I would like to thank my supervisor **Dr. M. Tanveer**, who was a constant source of inspiration during my work. Without his constant guidance and research directions, this research work could not be completed. His continuous support and encouragement has motivated me to remain streamlined in my research work.

I am also thankful to Dr. Md. Aquil Khan and Prof. Ram Bilas Pachori, my research progress committee members for their valuable feedback and comments which made significant improvements in my research work. I am also thankful to the Head, Department of Mathematics, for his continuous support and encouragement.

My sincere gratitude and respect to the worthy Director, Indian Institute of Technology Indore for providing the research facilities and support.

I would like to acknowledge the Indian Institute of Technology Indore for the institute fellowship during my PhD programme.

I am thankful to the lab members for the healthy discussion especially Ashwani Malik, M. Tabish, Bharat Richhariya, R.U. Khan, S. Sharma and Anuradha. I am also thankful to Dr Mohammed Wasid for his continuous support and assistance, whenever required. Moreover, I would like to thank the Department staff especially Jitendra for his support and assistance. I am also thankful to administrative and other staff members of the institute for their support and assistance during the PhD programme.

I would like to express my heartfelt respect to my parents for their love, care and support they have provided to me throughout my life. Special thanks to my parents, wife (Sumaira), daughter (Ayesha), brother (Aaqib), sister (Rubeena), brother-in-law (M. Tayub) and friends (Rasik, Abass, Anwer) as this thesis would not have been possible without their support and encouragements. Finally, I am thankful to all who directly or indirectly contributed, helped and supported me.

Mudasir Ahmad Ganaie

To my family and friends

Abstract

Decision tree, neural network and support vector machine are the powerful machine learning models widely used in classification problems. Decision tree is composed of terminal and non-terminal nodes. Each non-terminal node evaluates a series of decisions and optimizes the best split. The terminal nodes represent different class labels or their distributions. Intuitively, decision tree is a sequence of If-Then rules which are mostly understood by humans and are easy to implement. Neural network (NN) is another category of machine learning algorithms. It is composed of several interconnected computational units known as neurons. The neurons are interconnected via weights learned usually via back propagation algorithms. However, in randomized neural networks like random vector functional link network (RVFL) some weights are initialized randomly (fixed while training) and other weights are optimized via closed form solution or iterative algorithm. On the other hand, support vector machine (SVM) is an algorithm based on the concept of margin maximization. SVM implements the structural risk minimization and is a stable classifier. However, SVM is computationally inefficient and hence, twin SVMs (TWSVMs) have been formulated.

Ensemble learning is a well known approach in machine learning. Ensemble learning trains multiple base learners and collaborates them in a manner that the combined model is better compared to the individual models. Decision trees, neural networks, support vector machines and their combinations have been typically used in ensemble approach for better generalization performance.

An ensemble of decision trees, known as random forest (RaF), is a successful model widely used in the classification problems, and is considered as the best algorithm. Similarly, RVFL model is gaining its popularity across multiple domains due to its universal approximation capability. Moreover, it is faster compared to the networks trained via back-propagation. On the other hand, TWSVM have shown impressive performance and is faster compared to the SVM based models. TWSVM based models formulate the convex optimization problems, hence, guarantee the existence of global minima. The first part of the thesis is based on ensemble of decision trees i.e., RaF

and double RaF. The second part is based on shallow and deep ensembles of RVFL and the last part of this thesis is based on the ensemble of TWSVM based models with random projections.

Decision trees of standard RaF employ axis parallel splits at each non-terminal node. Oblique decision trees use oblique (linear) hyperplane for data partitioning at each non-terminal node. We present several variants of oblique RaF by using twin bounded SVM (TBSVM) at each non-terminal node to get the best separating hyperplane. Also, we present oblique rotation forest via TBSVM and oblique random subspace rotation forest via TBSVM. Moreover, recent study of double RaF showed that bootstrapping of the data at each non-terminal node results in better performance compared to standard RaF model which uses bootstrapping at the root node only. However, both standard RaF and standard double RaF use axis parallel decisions which ignore the geometric structure of the data. We propose oblique and rotation double RaF to improve the performance of the classification models.

Random vector functional link network is an efficient randomized neural network. Unlike back-propagation trained networks which suffer from local minima problem, slow convergence and learning rate sensitivity, RVFL overcomes these hurdles via closed form solution. We present minimum variance embedded RVFL and co-trained RVFL models. The former exploits the intra-class/total variance while as the later uses correlation concept to improve the generalization performance. Recently, deep learning models have been of great interest due to better feature representation. We present deep RVFL and ensemble deep RVFL with learning using privileged information.

TWSVM model use kernel functions for classification of non-linear data. However, as the size of the data increases, the kernel function leads to memory issues and also increase the complexity of the models. We present models wherein randomization based approach is used to project the data into non-linear space, this results in better control over complexity and the memory issues.

Keywords: Random forest, random vector functional link network, classification, randomization algorithms, ensemble learning, ensemble deep learning.

List of Publications

A. Published

A1. In Refereed Journals

1. **M. A. Ganaie**, M. Tanveer, P.N. Suganthan and V. Snasel "*Oblique and rotation double random forest*" Neural Networks, Elsevier **(SCI Indexed Impact Factor: 8.05, Q1)**

2. **M. A. Ganaie**, M. Tanveer "*Ensemble of deep random vector functional link network using privileged information for Alzheimer's disease diagnosis*" IEEE/ACM Transactions on Computational Biology and Bioinformatics **(SCI Indexed Impact Factor: 3.71, Q1)**

3. **M. A. Ganaie**, M. Tanveer, and P. N. Suganthan *"Oblique decision tree ensemble via twin bounded SVM."* Expert Systems with Applications, Elsevier, 143 (2020): 113072. **(SCI Indexed Impact Factor: 6.954, Q1)**

4. **M. A. Ganaie**, and M. Tanveer *"LSTSVM classifier with enhanced features from pre-trained functional link network."* Applied Soft Computing, Elsevier, 93 (2020): 106305. **(SCI Indexed Impact Factor: 6.725, Q1)**

5. M. Tanveer, **M. A. Ganaie**, and P. N. Suganthan *"Ensemble of classification models with weighted functional link network."* Applied Soft Computing, Elsevier, 107 (2021): 107322. **(SCI Indexed Impact Factor: 6.725, Q1)**

6. M. Tanveer, T. Rajani, R. Rastogi, Y. H. Shao, **M. A. Ganaie** "*Comprehensive review on twin support vector machines*" Annals of Operations Research, Springer **(SCI Indexed Impact Factor: 4.854, Q1)**

A2. In Refereed Conferences

1. **M. A. Ganaie**, M. Tanveer, and P. N. Suganthan *"Co-trained random vector functional link network."* 2021 International Joint Conference on Neural Networks, IJCNN. IEEE, 2021. **(Scopus Indexed, Core rank: A)**

2. **M. A. Ganaie**, M. Tanveer, and P. N. Suganthan *"Minimum variance embedded random vector functional link network."* International Conference on Neural Information Processing (ICONIP2020). **(Scopus Indexed, Core rank: A)**

B. Communicated

In Refereed Journals

UNDER REVISION

1. **M. A. Ganaie**, Minghui Hu, A.K. Malik, M. Tanveer, and P. N. Suganthan "*Ensemble deep learning: A review*" arXiv preprint arXiv:2104.02395. (Submitted after first revision to Engineering Applications of Artificial Intelligence, Elsevier) **(SCI Indexed Impact Factor: 6.212, Q1)**

C. Journal publications (other than thesis)

In Refereed Journals

[1] **M. A. Ganaie**, M. Tanveer, and C.T. Lin "*Large scale fuzzy least squares twin support vector machines for class imbalance learning*" IEEE Transactions on Fuzzy Systems **(SCI Indexed Impact Factor: 12.03, Q1)**.

[2] M. Tanveer, **M. A. Ganaie**, A Bhattacharjee and C.T. Lin "*Intuitionistic fuzzy weighted least squares twin SVMs*" IEEE Transactions on Cybernetics, (2022) **(SCI Indexed Impact Factor: 11.448, Q1)**.

[3] **M. A. Ganaie**, and M. Tanveer "*KNN weighted reduced universum twin SVM for class imbalance learning*" Knowledge-Based Systems, (2022) **(SCI Indexed Impact Factor: 8.038, Q1)**.

[4] M. Tanveer, Jatin Jangir, **M. A. Ganaie**, Iman Beheshti, M. Tabish, and Nikunj Chhabra "*Diagnosis of Schizophrenia: A comprehensive evaluation*" IEEE Journal of Biomedical and Health Informatics (2022) **(SCI Indexed Impact Factor: 5.36, Q1)**.

[5] **M. A. Ganaie**, M. Tanveer, and Iman Beheshti "*Brain age prediction with improved least squares TSVR*" IEEE Journal of Biomedical and Health Informatics (2022) **(SCI Indexed Impact Factor: 5.77, Q1)**.

[6] A. K. Malik, **M. A. Ganaie**, M. Tanveer, P.N. Suganthan "*Alzheimer's disease diagnosis via intuitionistic fuzzy random vector functional link network*" IEEE Transactions on Computational Social Systems (2022) **(SCI Indexed)**.

[7] M. Tanveer, A. Tiwari, R. Choudhary, and **M. A. Ganaie** "*Large-scale pinball twin support vector machines*" Machine Learning (2021): 1-24, Springer (DOI: 10.1007/s10994-021-06061-z) **(SCI Indexed Impact Factor: 2.94, Q1)**.

[8] **M. A. Ganaie**, and M. Tanveer "*Fuzzy least squares projection twin support vector machines for class imbalance learning*" Applied Soft Computing 113 (2021):

107933, Elsevier **(SCI Indexed Impact Factor: 6.725, Q1)**.

[9] I. Beheshti, **M. A. Ganaie**, V. Paliwal, A. Rastogi, I. Razzak, M. Tanveer (2021). *"Predicting brain age using machine learning algorithms: A comprehensive evaluation".* IEEE Journal of Biomedical and Health Informatics. (DOI: 10.1109/JBHI.2021.3083187) **(SCI Indexed Impact Factor: 5.77, Q1).**

[10] **M. A. Ganaie**, I. Beheshti, M. Tanveer "Brain age prediction using improved twin SVR." Neural Computing and Applications (2021): 1-11. **(SCI Indexed Impact Factor: 5.606, Q1)**.

[11] M. Tanveer, A. H. Rashid, **M. A. Ganaie**, M. Reza, I. Razzak, and K. L. Hua (2021). *"Classification of Alzheimer's disease using ensemble of deep neural networks trained through transfer learning."* IEEE Journal of Biomedical and Health Informatics. **(SCI Indexed Impact Factor: 5.77, Q1)**.

[12] **M. A. Ganaie**, S. Ghosh, N. Mendola, M. Tanveer, and S. Jalan *"Identification of chimera using machine learning."* Chaos: An Interdisciplinary Journal of Nonlinear Science 30.6 (2020): 063128. **(SCI Indexed Impact Factor: 3.642, Q1)**

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

BP Back propagation

DNN Deep neural network

dRVFL Deep random vector functional link network

dRVFL+ Deep random vector functional link network with LUPI framework

DT Decision tree

edRVFL Ensemble deep random vector functional link network

edRVFL+ Ensemble deep random vector functional link network with LUPI framework

ELM Extreme learning machine

GBT Gradient boosted tree

KRR Kernel ridge regression

LDA Linear discriminant analysis

LSTSVM Least squares twin support vector machines

LUPI Learning using privileged information

MPSVM Multi-surface proximal support vector machines

NCL Negative correlation learning

NN Neural network

PCA Principal component analysis

QPP Quadratic programming problem

RaF Random forest

RELSTSVM Robust energy based least squares twin support vector machines

RoF Rotation forest

RRoF Random sub-rotation forest

RVFL Random vector functional link network

RVFL+ Random vector functional link network with LUPI framework

SRM Structural risk minimization

SVM Support vector machine

TBRaF Twin bounded random forest

TBRoF Twin bounded rotation forest

TBRRoF Twin bounded random sub-rotation forest

TBSVM Twin bounded support vector machines

TWKSVC Twin k-class support vector classification

TWSVM Twin support vector machines

Chapter 1

Introduction

With the advent of technology, the generation of data has proliferated creating a need to develop the robust machine learning models. In the past few decades, the development of machine learning algorithms has brought significant changes in our daily life. The advancement of machine learning algorithms have attracted the research community especially in classification problems. Classification problem is the identification of a discrete category for the new testing sample. The classification learning algorithm is trained on a set of data with the observations of known categories. Mathematically, the classification problem is given as:

$$y = f(x, \theta) \in \mathbb{Y}, \tag{1.1}$$

where y is the label assigned to new observation x by the learning algorithm f, θ is the parameter of the learning algorithm and $\mathbb{Y}$ is the set of categories or class labels. The classification problem may be binary or multiclass depending on the cardinality of $\mathbb{Y}$. In binary class problems, there exist two categories while as in multiclass problems more than two categories are present. In literature, hundreds of classification problems have been proposed [62] from the computer science and mathematics to solve the classification problems.

In this thesis, we discuss decision trees (DTs), neural networks (NNs) and support vector machines (SVMs). We give the brief overview of these classification models in the following section, followed by the motivations and objectives of this thesis. Finally,

we give the contributions of this thesis followed by the organization of the thesis.

1.1 Background

"Many heads are better than one" "Many are smarter than the few". These proverbs have been studied in several sociological, psychological and other human aspects [227, 250]. Ensemble learning [221] is a widely used approach in classification and regression problems. The combination of multiple classifiers, which are unstable, into an ensemble model leads to better generalization performance compared to a single unstable classifier [57, 194, 225, 266]. Ensemble learning employ multiple base classifiers such that the ensemble prediction performance is better than any of the individual classifiers. Perturb and combine strategy on individual classifiers are used in ensemble methodology [19]. In perturb strategy, the classifiers are evaluated on the perturbed training datasets and in combine strategy, the outputs of these classifiers are aggregated in a suitable fashion such that the classification of an ensemble model is better compared to the individual baseline classifiers. In ensemble learning framework, the unstable classifiers are mostly sought ones. Decision trees and neural networks are unstable classifiers whose performance greatly varies with small perturbations in the training set or in construction [18]. Ensembles of support vector machines [134] have also been proposed wherein multiple support vector machines are generated. Thus, ensembles of random forest, random vector functional link network and support vector machines are generated to make the more accurate and robust classification models.

Decision tree algorithm is a commonly used classification model due to its simplicity and better interpretability. Decision tree uses divide and conquer approach to recursively partition the data. Decision tree enjoy multiple benefits. Decision tree uses a sequence of decisions from the root node to the leaf node via If-Then rules which are intuitive to humans. The recursive partition of the tree is sensitive to perturbation of the input data, and hence, results in an unstable classifier. Thus, it is said to have high variance and low bias which makes it most suitable algorithm for the ensemble methodology. The decision trees have been successfully employed in medical diagnosis

and legal analysis wherein interpretability is of utmost importance [33]. Mostly, ensemble of decision trees such as random forest (RaF) [23], gradient boosted tree (GBT) [67] are used which results in better performance compared to the individual decision trees. Due to the better generalization performance, random forest proved to be one of the best classification models among 179 classifiers evaluated on 121 datasets [62]. Based on the type of evaluation at each node, decision trees can be broadly divided into two categories: axis-parallel or orthogonal and oblique or multivariate decision trees. In orthogonal decision trees, the expression "$x_j < b_i$" (is j^{th} feature less than the threshold value b_i) while as in oblique decision trees "$w_i^T x < b_i$" expression is evaluated. In this thesis, we refer the standard RaF as an ensemble of orthogonal decision trees and oblique random forest as an ensemble of oblique decision trees.

Neural Network (NN) is another machine learning model widely used for the classification problems. NN consists of multiple computational units, known as neurons, interconnected to mimic the functionality of human brain. Typically, a neural network consists of an input layer, one or more hidden layers and an output layer. The hidden layer consists of multiple neurons or nodes that perform non-linear transformation from the input layer to the output layer. The hidden neurons are connected by the weights of a network. The training of neural networks is performed by minimizing the loss functions. Specifically, gradient of the loss function with respect to the network parameters is calculated and the model weights are updated iteratively in the negative direction of the gradient to minimize the loss function. However, it suffers from local minima problem, slow convergence and learning rate sensitivity [248]. To elevate the training procedure, randomization based neural networks have been putforth. Randomization based models avoid the above enumerated issues as the optimization problem is solved via closed form solution [84, 85, 231, 263]. These models show good generalization performance and training is faster [52, 284, 285]. Among the randomization based models, extreme learning machine [113] and RVFL [203] are the widely used architectures.

Support vector machine (SVM) [47, 151, 268] is a powerful method used mainly in classification as well as regression problems. SVM is based on the maximum margin

concept and shows better generalization performance as it implements the structural risk minimization (SRM) principle. Despite its better generalization performance, SVM owns high computational complexity of the order of $O(M^3)$ where M is the size of the training dataset. To overcome this drawback of higher complexity, TWSVM builds two hyperplanes such that each hyperplane is proximal to its own class and farthest from the samples of other class. Unlike SVM wherein a single large quadratic programming problem (QPP) is solved, TWSVM solves two QPPs of smaller size compared to SVM. Solving two smaller quadratic programming problems (QPPs) make TWSVM approximately 4 times faster than SVM. As TWSVM implements empirical risk minimization principle, twin bounded SVM (TBSVM) [233] implemented the structural risk minimization principle. Least squares twin SVM (LSTSVM) [73, 144] involves a system of linear equations with squared loss function instead of the convex QPP. As LSTSVM is sensitive to noise and outliers, hence energy-based least squares twin support vector machine [190] introduced energy term to reduce the effect of noise and outliers. Tanveer et al. [260] introduced an extra regularization term and proposed robust energy-based least squares twin support vector machine (RELSTSVM), resulting in the optimization problems to be positive definite and hence, better generalization. Recent study [261] shows that RELSTSVM is the best classifier among the twin support vector machine models. Different from twin bounded SVM, least squares twin SVM, energy based least squares twin SVM, and robust energy based least squares twin SVM, a new multiclass approach called twin k-class support vector classification (TWKSVC) [294] based on "1-versus-1-versus-rest" generates $k(k-1)/2$ binary classifiers for a k-class classification problem. To reduce the computational complexity of TWKSVC, least squares TWKSVC [191] introduced the equality constraints in the objective function of TWKSVC to solve linear system of equations instead of solving a QPP.

1.2 Motivation

Deep neural networks (DNNs) have been successfully applied in tasks like image, text and speech recognition, however, there are many other tasks wherein the DNNs perform lower compared to other models like random forest [23, 193, 236]. Recent study of evaluation of classification models on the UCI machine learning repository [60], wherein the datasets range from standard vision to speech dataset, reveal that random forest is a top ranked classifier [62, 306]. Thus, Chapter 3 and Chapter 4 of this thesis present the techniques to further improve the performance of ensemble of decision trees.

Training of artificial neural networks via back propagation (BP) has several disadvantages like slow convergence, local minima problem and sensitive to learning rate setting which results in lower generalization performance [248]. To avoid these issues, randomization based neural network models use closed form solutions [84, 85, 231, 263] for training the network. Apart from the neural networks trained via backpropagation, there is growing interest in randomized based neural networks [11, 84, 85, 201]. Also, neural network ensembles are more accurate than individual networks [112, 129, 220, 270]. In Chapter 5, we present a novel RVFL model and an ensemble of RVFL model. Recently, deep learning architectures have received quite a lot attention. In particular, non-iterative learning based deep randomized models have been proposed for the classification and regression problems. The deep randomized models especially deep RVFL with direct links have been quite successful. However, deep RVFL and its ensemble models are trained only on normal samples. In Chapter 6, we present deep RVFL and its ensembles which are enabled to incorporate privileged information, however, the standard RVFL model and its deep models are unable to use privileged information.

Support vector machines (SVMs) have been successfully employed in classification problems. The success of SVMs is attributed to three factors. First, SVM is a maximum margin classifier. Second, the dual form of SVM solves a quadratic programming problem which depends on number of data samples and not their dimensions. Third,

the dual form of the optimization problem involves the inner product of data points and hence, kernel trick can be easily plugged [64]. The SVMs need to choose kernel type, kernel parameters and the regularization parameter. The choice of these parameters determines the generalization performance of the models. Using kernel trick for the transformation of the data to a higher dimensional space leads to computation and memory issues. Using explicit randomized features instead of Gram matrix leads to smaller computational time as no kernel parameters are tuned [64]. Therefore, we propose ensembles of randomized feature based twin SVM model in Chapter 7 and Chapter 8, which provide better control over complexity and memory related issues.

1.3 Objectives

The objectives of this thesis are:

[1] To present literature review on ensemble learning and twin SVM based models.

[2] To develop novel ensembles of decision trees for the classification problems.

[3] To develop the rotation double random forest with diversified base learners and also develop the oblique double random forest models.

[4] To develop the novel RVFL and its ensemble.

[5] To develop the novel deep RVFL network using privileged information and its ensemble.

[6] To develop novel ensemble of least squares twin SVM model.

[7] To develop novel ensemble classifiers based on twin SVM models.

1.4 Contributions of the thesis

In this section, we give the brief overview of work contributed in this thesis. The contributions are as follows:

[1] To develop the novel machine learning algorithms, better understanding of the literature is important. Therefore, we reviewed different approaches of ensemble learning followed in the literature. Moreover, we reviewed different learning techniques followed in twin SVM to make the models more efficient and/or robust against noise and outliers. We presented detailed review of the ensemble learning approaches and the formulations of twin SVM based models in Chapter 2 of this thesis.

[2] Decision trees and their ensembles are well known machine learning algorithms applied in different domains to solve the classification problems. In Chapter 3, we present an ensemble of decision trees via twin bounded support vector machines. The proposed method overcomes the issues of invertibility while generating the decision trees. The proposed method doesn't require any regularization technique to render the solution and uses structural risk minimization principle for better generalization performance. Furthermore, we developed ensemble of decision trees known as twin bounded random forest, twin bounded rotation forest and twin bounded random subspace rotation forest for the classification problems.

[3] Recent study of double random forest [91] evaluated the effect of node size on the performance of the model. The study revealed that the prediction accuracy could be improved if there is a way to generate deeper decision trees. The authors showed that the largest tree grown on a given data by the standard random forest might not be sufficiently large to give the optimal performance. Hence, double random forest [91] generated decision trees that are bigger than the ones in standard random forest. Instead of training each decision tree with different bags of training set obtained via bagging approach at the root node, the authors in [91] generated each tree with the original training set and used bootstrap aggregation at each non-leaf node of the decision tree to obtain the best split. However, both the random forest and double random forest are ensembles of univariate decision trees and hence, ignore the geometric class distributions

resulting in lower generalization performance. In Chapter 4, we present oblique and rotation double random forest to overcome these issues. Oblique double random forest models integrate the benefits of double random forest and the geometric structure information of the class distribution for better generalization performance. For generating more diverse base learners in the double random forest, we present rotation double random forest wherein we transform or rotate the feature space at each non-leaf node using the principal component analysis or linear discriminant analysis transformations.

[4] We present minimum variance embedded RVFL and ensemble of RVFL. Minimum variance embedded RVFL exploits the training data dispersion and uses extra information for learning the network parameters. Furthermore, we present an ensemble of RVFL known as co-trained RVFL which trains two RVFL models jointly such that each RVFL model is constructed with different feature projections. We use randomly projected features and sparse-l_1 norm autoencoder based features to train the proposed coRVFL model, which resulted in better generalization performance.

[5] Recently, deep learning architectures have received quite a lot attention. In particular, non-iterative learning based deep randomized models have been proposed for the classification and regression problems. The deep randomized models especially RVFL with direct links have been successful. However, deep RVFL and its ensemble models are trained only on normal samples. We present deep RVFL and its ensembles which are enabled to incorporate privileged information, however, the standard RVFL model and its deep models are unable to use privileged information. Privileged information based approach is commonly seen in human learning. To fill this gap, we have incorporated learning using privileged information (LUPI) in deep RVFL model, and propose deep RVFL with LUPI framework (dRVFL+). Privileged information is available while training the models. In order to make the model more robust, we propose ensemble deep RVFL+ with LUPI framework.

[6] Inspired by the random feature projection in RVFL network, we present LSTSVM classifier with enhanced features from pre-trained functional link network. In addition to random projections, kernel trick has also been used for transforming the input data. The kernel trick transforms the data into non-linear space with kernel function being applied to each pair of the training sample. This kernel trick leads to computational and memory issues. Hence, we use LSTSVM projected features which results in improved performance and reduce the computational cost compared to kernel trick. Here, the input feature space is enhanced by the pre-trained functional link network. Weights are generated by LSTSVM, and a non-linear function is applied on the product between input features and the weights to get the enhanced features. The final classification is performed by least squares twin support vector machines based on the original and enhanced feature space.

[7] Kernel based machines such as twin SVM based models can approximate any arbitrary function with enough training data. However, these models suffer as the size of the data increases. The kernel trick transforms the data into non-linear space with kernel function being applied to each pair of the training samples. This kernel matrix leads to computational and memory issues. To overcome these issues, we propose two approaches for the non-linear projection of the data and hence, avoid the use of kernel trick which results in better efficiency compared to kernel based models. In the first approach, the input data points are mapped explicitly into a randomized feature space via neural network wherein the weights of the hidden layer are generated randomly. After feature projection, classification models twin bounded support vector machines (SVM), least squares twin SVM, twin k-class SVM, least squares twin k-class SVM and robust energy based least squares twin SVM are trained on the extended features (original features and randomized features). In the second approach, twin bounded support vector machines (SVM), least squares twin SVM, twin k-class SVM, least squares twin k-class SVM and robust energy based least squares twin SVM

models are used to generate the weights of the hidden layer architecture and the weights of output layer are optimized via closed form solution. The projection of data points via random process or twin SVM based models provide better control over complexity and memory issues.

1.5 Organization of the thesis

The work of this thesis is divided into nine chapters. Figure 1.1 gives the pictorial layout of the thesis. The brief description of the chapters included is as follows:

- In chapter 2, we review the basics of ensemble learning and different ensemble strategies. Moreover, we discuss the algorithms and the formulations of the variants of the ensembles of decision trees, RVFL and twin SVM based models.
- In chapter 3, we present the oblique decision tree ensemble via TBSVM.
- In chapter 4, we present the basics of double random forest. Also, we propose rotation double random forest and ensembles of oblique double random forest.
- In chapter 5, we present the minimum variance embedded RVFL and co-trained RVFL.
- In chapter 6, we present the deep RVFL with privileged information and ensemble deep RVFL with privileged information.
- In chapter 7, we present LSTSVM classifier with enhanced features from pre-trained functional link network.
- In chapter 8, we present ensemble of classification models with weighted functional link network.
- In chapter 9, we give brief contribution of this thesis and the possible future research directions.

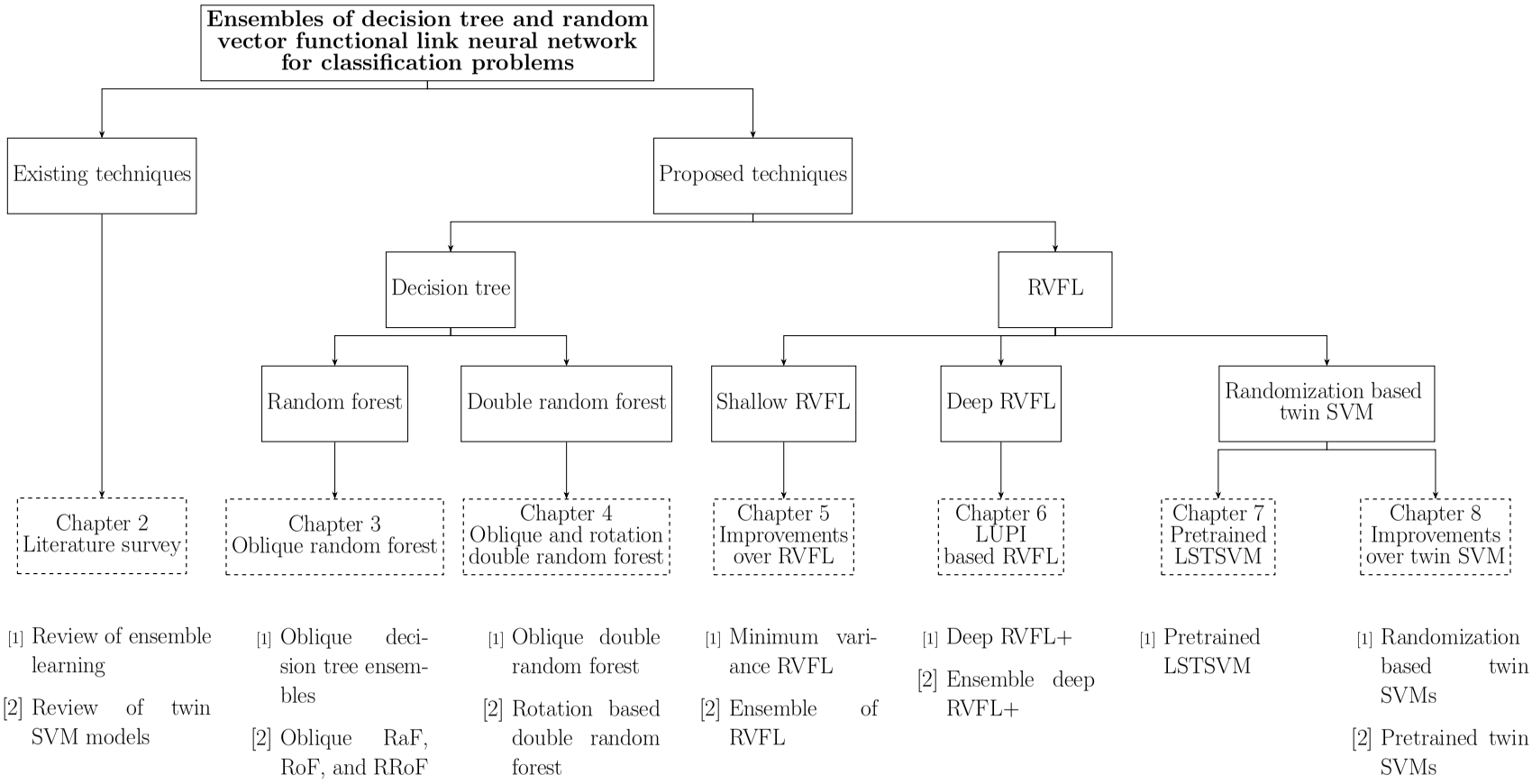


Figure 1.1: Layout of the thesis.

Chapter 2

Literature survey and research methodology

In this chapter, we review the literature on different research problems addressed in this thesis. We divided the literature into six sections. Section 2.1 introduces the ensemble learning. Moreover, a survey for ensemble learning strategies is presented in Section 2.2 and Section 2.3 discusses decision trees and different approaches to generate the ensemble of decision trees. Section 2.4 introduces the artificial neural networks and discusses the formulation of multiple artificial neural networks. Section 2.5 discusses the formulation of twin support vector machines and its variants.

2.1 Ensemble learning

Generally speaking, the goal of generating the hypothesis H in machine learning area is that it should perform better when applied to unknown data. The performance of the model is measured with respect to the area in which the model is applied. Combining the predictions from several models has proven to be an elegant approach for increasing the performance of the models. Combination of several different predictions from different models to make the final prediction is known as ensemble learning or ensemble model. The ensemble learning involves multiple models combined in some fashion like averaging, voting such that the ensemble model is better than any of the individual baseline models. To prove that ensemble is better than individual models,

Marquis de Condorcet proposed a theorem wherein he proved that if the probability of each voter being correct is above 0.5 and the voters are independent, then addition of more voters increases the probability of majority vote being correct until it approaches 1 [46]. Although Marquis de Condorcet proposed this theorem in the field of political science and had no idea of the field of machine learning, but it is the similar mechanism that leads to better performance of the ensemble models. Assumptions of Marquis de Condorcet theorem also holds true for ensembles [93].

Perturb and combine strategy [19] is core of the ensemble learning [57] and hence, it has been used across different domains like machine learning [287], computer vision tasks [79] and recognition of patterns. Both theoretical and empirical aspects of the ensemble learning have been explored in the literature. Multiple classifier systems [319] or ensemble learning perturbs the input data to induce diversity among the base learners of an ensemble and use combine strategy to aggregate the outputs of base learners such that the performance of the ensemble model is better in comparison with the individual learners.

To analyze how the ensemble models perform better compared to individual models, studies like variance reduction among the classifiers [19, 78, 299] have been put-forth. With the bias and variance reduction theory [19, 139], the classification error is given in terms of bias and variance. Bias measure gives information about how far is the average guess of each base learner from the target class over the perturbed training sets generated from a given training set and variance measures how much the base learners guess fluctuates with the perturbations of the given training set. Ensemble methods have been supported by several theories like bias-variance [139, 289], strength correlation [23], stochastic discrimination [137], and margin theory [229]. These theories provide the equivalent of bias-variance-covariance decomposition [212].

The reasons for the success of ensemble learning include: statistical, computational and representation learning [57], bias-variance decomposition [139] and strength-correlation [23].

2.1.1 Bias-variance decomposition

Initially, the success of ensemble methods was theoretically investigated in regression problems. The authors proved via ambiguity decomposition [26, 143] that the proper ensemble classifier guarantees a smaller squared error compared to the individual predictors of the classifier. Ambiguity decomposition was given for single dataset based ensemble methods, later on, multiple dataset bias-variance-covariance decomposition was introduced in [26, 27, 76, 208] and is given as:

$$\begin{aligned} E[o-y]^2 &= bias^2 + \frac{1}{L}var + (1-\frac{1}{L})covar, \\ bias &= \frac{1}{L}\sum_i (E[o_i]-y), \\ var &= \frac{1}{L}\sum_i E[o_i - E[o_i]]^2, \\ covar &= \frac{1}{L(L-1)}\sum_i \sum_{j\neq i} E[o_i - E[o_i]][o_j - E[o_j]], \end{aligned} \tag{2.1}$$

where y is target, o_i is the output of i^{th} model and L is the ensemble size. Here *bias* term measures the average difference between the base learner and the model output, *var* indicates their average variance, and *covar* is the covariance term measuring the pairwise difference of different base learners.

The above given equations of decomposition error can't be directly applied to the datasets with discrete class labels due to their categorical nature. However, alternate ways to decompose the error in classification problems have been given in [21, 69, 121, 139, 140].

2.1.2 Statistical, computational and representational aspects

Dietterich provided Statistical, Computational and Representational reasons [57] for success of ensemble models. The learning model is viewed as the search of the optimal hypothesis H among the several hypothesis in the search space. When the amount of data available for the training is smaller compared to the size of the hypothesis space, the statistical problem arises. Due to statistical problems, the learning algo-

rithm identifies the different hypothesis which give similar performance on the training samples. Ensembling of these hypothesis results in an algorithm which reduces the risk of being a wrong classifier. In computational aspect, a learning algorithm stucks in a local optima due to some form of local search. Ensemble model overcomes this issue by performing some form of local search via different starting points which leads to better approximation of the true unknown function. Another reason is representational wherein none of the hypotheses among the set of hypothesis is able to represent the true unknown function. Hence, ensembling of these hypothesis via some weighting technique results into the hypothesis which expands the representable function space.

2.1.3 Diversity

The success of ensemble methods depends on the diversity among the base classifiers and the same is highlighted in [57]. Different approaches have been proposed to generate diverse classifiers. Different methods like bootstrap aggregation (bagging) [18], adaptive boosting (AdaBoost) [66], random subspace [7], and random forest [23] approaches are followed for generating the multiple datasets from the original dataset to train the different predictors such that the outputs of predictors are diverse. Attempts have been made to increase diversity in the output data wherein multiple outputs are created instead of multiple datasets for the supervision of the base learners. 'Output smearing' [22] is one of this kind which induces random noise to introduce diversity in the output space.

2.2 Ensemble strategies

The different ensemble strategies have evolved over a period of time which results in better generalization of the learning models. The ensemble strategies are broadly categorised as follows:

2.2.1 Bagging

Bagging [18], also known as bootstrap aggregating, is one of the standard techniques for generating the ensemble-based algorithms, which is applied to enhance the performance of an ensemble classifier. The main idea in bagging is to generate a series of independent observations with the same size, and distribution as that of the original data. Given the series of observations/samples, generate an ensemble predictor which is better than the single predictor generated on the original data. Bagging increases two steps in the original models: First, generating the bagging samples and passing each bag of samples to the base models and second, strategy for combining the predictions of the multiple predictors. Bagging samples may be generated with or without replacement. Combining the output of base predictors may vary as mostly majority voting is used for classification problems while the averaging strategy is used in regression problems for generating the ensemble output.

Random forest [23] is an improved version of the decision trees that use the bagging strategy for improving the predictions of the base classifier (decision tree). At each non-leaf node of a decision tree in random forest, only a subset of features is randomly selected and considered for splitting. The purpose of ensembling the decision trees is to decorrelate the trees and prevent over-fitting. Breiman [23] showed heuristically that the variance of the bagged predictor is smaller than the original predictor and proposed that bagging is better in higher dimensional data. However, the analysis of the smoothing effect of bagging [29] revealed that bagging doesn't depend on the data dimensionality.

Bühlmann et al. [28] gave theoretical explanation of how bagging gives smooth hard decisions, small variance, and mean squared error. Since, bagging is computationally expensive, hence subbagging and half subbagging [28] were introduced. Half subbagging, being computationally efficient, is as accurate as bagging.

Several approaches have been proposed to combine bagging with other machine learning algorithms. Kim et al. [133] used bagging method to generate multiple bags of the dataset and multiple SVM are trained independently with each bag as the

input. The output of the models in an ensemble is combined via majority voting, least squares estimation weighting and double layer hierarchical approach. In the double layer hierarchical approach, another SVM is used to combine the outcomes of the multiple SVMs efficiently. In [262], asymmetric bagging strategy was used to generate the ensemble model to handle the class imbalance problems. A case study of bagging, boosting and basic ensembles [174] revealed that at higher rejection rates of samples, boosting is better compared to bagging and basic ensembles. However, as the rejection rate increases the difference disappears among the boosting, bagging and basic ensembles. Ha et al. [87] showed that bagging based ensemble models are better compared to individual multilayer perceptron. Gençay and Qi [77] analysed the bagging approach and other regularization techniques and showed that bagging regularizes the neural networks and hence provide better generalization. For predicting the short term load forecasting, an ensemble of bagging based neural networks [131] was proposed. Unlike random forest [23] which uses majority voting for aggregating the ensemble of decision trees, bagging based survival trees [104] used Kaplan–Meier curve to predict the ensemble output for breast cancer and lymphoma patients. In [4], ensembles of stacked denoising autoencoders for classification showed that bagging and switching technique in a general deep machine results in improved diversity.

Bagging has also been applied to solve the problems of imbalanced data. Roughly balanced bagging [99] tries to equalize each class's sampling probability in binary class problems wherein the negative class samples are sampled via negative binomial distribution, instead of keeping the sample size of each class the same number. Neighbourhood balanced bagging [15] incorporates the neighbourhood information for generating the bagging samples of the class imbalance problem.

The theoretical and experimental analysis of online bagging and boosting [198] showed that the online bagging algorithm can achieve similar accuracy as the batch bagging algorithm with only a little more training time, however, online bagging is an option when all training samples can't be loaded into the memory due to memory issues.

Although, ensembling may increase the computational complexity, however, bag-

Papers	Contribution
[18]	Proposed the idea of bagging
[23]	Bagging with random subspace decision trees and ensembling outputs via majority voting
[29]	Theoretical analysis of bagging
[28]	Theoretical justification of bagging, proposed subbagging and half subagging
[262]	Proposed assymmetric bagging with SVMs and ensembling outputs SVMs
[133]	Bagging with SVMs and ensembling outputs via SVMs, majority voting and least squares estimation
[174]	Case study of bagging, boosting and basic ensembles
[87, 131]	Bagging with neural networks and ensembling outputs via majority voting
[77]	Study of Bayesian regularization, early stopping and bagging
[104]	Bagging with decision trees and ensembling outputs via Kaplan–Meier curve
[99]	Roughly balanced bagging on decision trees and ensembling outputs via majority voting
[15]	Neighbourhood balanced bagging ensembling outputs via majority voting
[198]	Theoretical and experimental analysis of online bagging and boosting

Table 2.1: Bagging based ensemble models.

ging possesses the property that it can be paralleled. Hence, it can lead to effective reduction in the training time subject to the availability of hardware for running the parallel models. Since, deep learning models have high training time, hence, optimization of multiple deep models on different training bags is not a feasible option. To subsidise the memory issues, online bagging is used which avoids the complexity of loading all the data into memory.

2.2.2 Boosting

Boosting technique is a sequential approach used in ensemble models for improving the generalization performance of the base learners. The techniques such as majority voting in case of classification problems or linear combination of unstable learners in the regression problems results in better prediction compared to the single unstable learner. Boosting methods like AdaBoost [66] and gradient boosting [70] have been used across different domains. Adaboost uses a greedy technique for minimizing a convex surrogate function upper bounded by misclassification loss via augmentation at each iteration of the current model with the appropriately weighted predictor. AdaBoost learns an effective ensemble classifier as it leverages the incorrectly classified sample at each stage of the learning. AdaBoost minimizes the exponential loss function while as the gradient boosting generalized this framework to the arbitrary differential loss function.

Boosting, also known as forward stagewise additive modelling, was originally pro-

posed to improve the performance of the classification trees. It has been recently incorporated in the deep learning models keeping in view the performance of the deep learning models in applications across many domains.

Boosted deep belief network (DBN) [168] for facial expression recognition unified the boosting technique and multiple DBN's via objective function which results in a strong classifier. The model learns complex feature representation to build a strong classifier in an iterative manner. deep boosting [48] is an ensemble model that uses the deep decision trees or can be used in combination with any other rich family classifier and improves the generalization performance. In each stage of the deep boosting, the decisions of which classifier to add and what weights should be chosen depends on the (data-dependent) complexity of the classifier to which it belongs. The interpretation of the deep boosting classifier is given via structural risk minimization principle at each stage of the learning. Multiclass deep boosting [147] extended the deep boosting [48] algorithm to theoretical, algorithmic, and empirical results to the multiclass problems. Due to the limitation of the training data in each mini batch, Boosting convolutional neural network (CNN) may overfit the data. To avoid overfitting, incremental boosting CNN (IBCNN) [89] accumulated the information of multiple batches of the training data samples. The IBCNN uses decision stumps on the top of single neurons as the unstable learners and learns weights via AdaBoost method in each mini batch. Unlike DBN [168] which uses image patch for learning the unstable classifiers, incremental Boosting CNN trains the classifiers from the fully connected layer i.e. the whole image is used for learning the classifiers. To make the IBCNN model more efficient, the unstable learners loss functions are combined with the global loss function.

Boosted CNN [183] use boosting technique for training the deep CNN. Instead of averaging, least squares objective function was used to incorporate the boosting weights into CNN. The authors also showed that CNN can be replaced by network structure within their boosting framework for improving the performance of the base classifier. Boosting increases the complexity of training the networks, hence, the concept of dense connections was introduced in a deep boosting framework to overcome the problem of vanishing gradient problem for image denoising [37]. Deep boosting

framework was extended to image restoration in [38] wherein the dilated dense fusion network was used to boost the performance.

The convolutional channel features [296] generated the high level features via CNN and then used boosted forest for final classification. As CNN has high number of hyperparameters than the boosted forest, thus the ensemble model proved to be efficient than end-to-end training of CNN models both in terms of performance and time. The authors showed its application in edge detection, object proposal generation, pedestrian and face detection. A stagewise boosting deep CNN [272] trains several CNN models within the offline paradigm boosting framework. To extend the concept of boosting in online scenario's wherein only a chunk of data is available at given time, boosting independent embeddings robustly (BIER) [195] was proposed to cope up the online scenario's. In BIER, a single CNN model is trained end-to-end with an online boosting technique. The training set in the BIER is reweighted via the negative gradient of the loss function to project the input spaces (images) into a collection of independent output spaces. To make BIER more robust, hierarchical boosted deep metric learning [273] incorporated the hierarchical label information into the embedding ensemble which improves the performance of the model on the large scale image retrieval application. As deep boosting results in higher training time, thus, to reduce the warm-up phase of training which trains the classifier from scratch, deep incremental boosting [184] used transfer learning approach. This approach leveraged the initial warm-up phase of each incremental base model of the ensemble during the training of the network. To reduce the training time of boosting based ensembles, snapshot boosting [310] combined the merits of snapshot ensembling and boosting to improve the generalization performance without increasing the cost of training. Snapshot boosting trains each base network and combines the outputs via meta learner to combine the output of base learners more efficiently.

Literature shows that the boosting concept is the backbone behind well-known architectures like deep residual networks [96, 243] and AdaNet [49]. The theoretical background for the success of the deep residual network (DeepResNet) [96] was explained in the context of boosting theory [110]. Huang et al. [110] showed that the

Papers	Contribution
[168]	Boosted deep belief network (DBN) as base classifiers for facial expression recognition
[48]	Decision trees as base classifiers for binary class classification problems
[147]	Decision trees as base classifiers for multiclass classification problems
[89]	Boosting based CNN with incremental approach for facial action unit recognition
[183]	Boosted CNN
[37]	Deep boosting for image denoising with dense connections
[38]	Deep boosting for image restoration and image denoising
[296]	Ensemble of CNN and boosted forest for edge detection, object proposal generation, pedestrian and face detection
[272]	CNN Boosting applied to bacterila cell images and crowd counting
[195]	Boosted deep independent embedding model for online scenarios
[273]	Hierarchical boosted deep metric learning with hierarchical label embedding
[184]	Transfer learning based deep incremental boosting
[310]	Snapshot boosting

Table 2.2: Boosting based ensemble models.

output of the top layer is a layer-by-layer boosting method. Huang et al. [110] proposed multi-channel telescoping sum boosting learning framework, known as BoostResNet, wherein each channel has a scalar value updated during rounds of boosting to minimize the multi-class error rate. The fundamental difference between the AdaNet and BoostResnet is that the former maps the feature vectors to classifier space and boosts weak classifiers while the latter used multi-channel representation boosting. Huang et al. [110] showed that in terms of computational time, BoostResNet is more efficient than DeepResnet.

The theory of boosting was extended to online boosting in [12] and provided theoretical convergence guarantees. Online boosting shows improved convergence guarantees for batch boosting algorithms.

The ensembles of bagging and boosting have been evaluated in [80]. The study evaluated the different algorithms based on the concept of bagging and boosting along with the availability of software tools. The study highlighted the practical issues and opportunities of their feasibility in ensemble modeling.

Boosting based models are slower due to the sequential approach used in reweighing the samples after training of each base model.

2.2.3 Stacking

Ensembling can be done either by combining outputs of multiple base models in some fashion or using some method to choose the "best" base model. Stacking is one

of the integration techniques wherein the meta-learning model is used to integrate the output of base models. If the final decision part is a linear model, the stacking is often referred to as "model blending" or simply "blending". The concept of stacking or stacked regression was initially given by [288]. In this technique, the dataset is randomly split into J equal parts. For the j^{th}-fold cross-validation one set is used for testing and the rest are used for training. With these training testing pair subsets, we obtain the predictions of different learning models which are used as the meta-data to build the meta-model. Meta-model makes the final prediction, which is also called the winner-takes-all strategy.

Stacking is a bias reducing technique [150]. Following [288], deep convex net (DCN) [55], a deep learning architecture, composed of variable number of modules stacked together to form the deep architecture. Each learning module in DCN is convex. DCN is a stack of several modules consisting of linear input units, hidden layer non-linear units, and the second linear layer with the number of units as that of target classification classes. The modules are connected layerwise as the output of the lower module is given as input to the adjacent higher module in addition to the original input data. The deep stacking network (DSN) enabled parallel training on very large scale datasets [54], the network was named stacking based as it shared the concept of "stacked generalization" [288]. The kernelized version of DCN, known as kernel deep convex networks (K-DCN), was given in [56], here the number of hidden layer approach infinity via kernel trick. The authors showed that K-DCN performs better compared to DCN. However, due to kernel trick the memory requirements increase and hence, may not be scalable to large scale datasets also we need to optimize the hyperparameters like the number of levels in the stacked network, the kernel parameters to get the optimal performance of the network. To leverage the memory requirements, random Fourier feature-based kernel deep convex network [114] approximated the Gaussian kernel which reduces the training time and the memory requirements and thus helps in the evaluation of K-DCN over large scale datasets. A framework for parameter estimation and model selection in kernel deep stacking networks [283] based on the combination of model-based optimization and hill-climbing approaches

used data-driven framework for parameter estimation, hyperparameter tuning and model selection in kernel deep stacking networks. Another improvement over DSN was tensor deep stacking network (T-DSN) [116], here in each block of the stacked network large single hidden layer was split into two smaller ones and then mapped bilinearly to capture the higher-order interactions among the features. Comprehensive evaluation and detailed analysis of the learning algorithm and T-DSN implementation was given in [117]. Sparse coding is another popular method using in the deep learning area. The advantage of sparse representation is numerous, including robust to noise, effective for learning useful features, etc. Sparse deep stacking network (S-DSN) is applied in image classification and abnormal detection [153, 249]. The authors stack many sparse simplified neural network modules (SNNM) with mixed-norm regularization, in which weights are solved by using the convex optimization and the gradient descent algorithm. In order to make sparse SNNM learning the local dependencies between hidden units, [154] split the hidden units or representations into different groups, which is termed as group sparse DSN (GS-DSN). The DSN idea is also utilized in the deep reinforcement learning field. Zhang et al. [302] employ DSN method to integrate the observations from the formal network: grasp network and stacking network based on Q-learning algorithm to make an integrated robotic arm system to grasp and place actions. Convolutional neural networks (CNN) are widely used in the image classification task, the stacking method also plays a role in this field. Wang et al. [275] stack the evolved block multiple times to increase the performance of the neural architecture search task. Zhang et al. [301] presents a deep hierarchical multi-patch network for image deblurring, with the stacked method, they can reach a better result.

Since, there is no temporal representation of the data in DSNs, they are less effective to the problems where temporal dependencies exist in the input data. To embed the temporal information in DSNs, recurrent deep stacking networks (R-DSNs) [200] combined the advantages of DSNs and recurrent neural networks (RNN). Unlike RNN which uses back propagation through time for training the network, R-DSNs use echo state network (ESN) to initialize the weights and then fine-tuning them via batch-

mode gradient descent. A stacked extreme learning machine (ELM) was proposed in [315]. Here, at each level of the network ELM with the reduced number of hidden nodes was used to solve the large scale problems. The number of hidden nodes was reduced via the principal component analysis (PCA) reduction technique. Keeping in view the efficiency of stacked models, the number of stacked models based on support vector machine have been proposed [159, 276, 277]. Traditional models like random forests have also been extended to deep architecture, known as deep forests [317], via stacking concept.

In addition to DSNs, there are some novel network architectures proposed based on the stacked method, Low et al. [170] contributed a stacking-based deep neural network (S-DNN) which is trained without a backpropagation algorithm. Kang et al. [128] presented a model by stacking conditionally restricted Boltzmann machine and deep neural network, which achieve significant superior performance with fewer parameters and fewer training samples.

2.2.4 Negative correlation based ensembles

Negative correlation learning (NCL) [169] is an important technique for training the learning algorithms. The main concept behind the NCL is to encourage diversity among the individual models of the ensemble to learn the diverse aspects of the training data. NCL minimizes the empirical risk function of the ensemble models via minimization of error functions of the individual networks. NCL [169] was evaluated for regression as well as classification tasks. The evaluation used different measures like simple averaging and winner-takes-all measures on classification tasks and simple average combination methods for regression problems. The authors figured out that winner-takes-all is better compared to simple averaging in NCL ensemble models.

Shi et al. [240] proposed deep negative correlation learning architecture for crowd counting known as D-ConvNet i.e. decorrelated convolutional networks. Here, counting is done based on regression-based ensemble learning from a pool of convolutional feature mapped weak regressors. The main idea behind this is to introduce the NCL concept in deep architectures. Robust regression via deep NCL [307] is an extension

of [240] wherein theoretical insights about the Rademacher complexity are given and extended to more regression-based problems.

Buschjäger et al. [30] formulated a generalized bias-variance decomposition method to control the diversity and smoothly interpolation. They present the generalized negative correlation learning algorithm, which can encapsulate many existing works in literature and achieve superior performance.

2.2.5 Explicit/Implicit ensembles

Ensembling of deep neural networks doesn't seem to be an easy option as it may lead to increase in computational cost heavily due to the training of multiple neural networks. High performance hardware's with GPU acceleration may take weeks of weeks to train the deep networks. Implicit/Explicit ensembles obtain the contradictory goal wherein a single model is trained in such a manner that it behaves like ensemble of training multiple neural networks without incurring additional cost or to keep the additional cost as minimum as possible. Here, the training time of an ensemble is same as the training time of a single model. In implicit ensembles, the model parameters are shared and the single unthinned network at test times approximates the model averaging of the ensemble models. However, in explicit ensembles model parameters are not shared and the ensemble output is taken as the combination of the predictions of the ensemble models via different approaches like majority voting, averaging and so on.

Dropout [245] creates an ensemble network by randomly dropping out hidden nodes from the network during the training of the network. During the time of testing, all nodes are active. Dropout provides regularization of the network to avoid overfitting and introduces sparsity in the output vectors. Overfitting is reduced as it trains exponential number of models with shared weights and provides an implicit ensemble of networks during testing. Dropping the units randomly avoids coadaptation of the units by making the presence of a particular unit unreliable. The network with dropout takes 2-3 times more time for training as compared to a standard neural network. Hence, a balance is to be set appropriately between the training time of

the network and the overfitting. Generalization of DropOut was given in DropConnect [274]. Unlike DropOut which drops each output unit, DropConnect randomly drops each connection and hence, introduces sparsity in the weight parameters of the model. Similar to DropOut, DropConnect creates an implicit ensemble during test time by dropping out the connections (setting weights to zero) during training. Both DropOut and DropConnect suffer from high training time. To alleviate this problem, deep networks with stochastic depth [111] aimed to reduce the network depth during training while keeping it unchanged during testing of the network. Stochastic depth is an improvement on ResNet [96] wherein residual blocks are randomly dropped during training and bypassing these transformation blocks connections via skip connections. Swapout [242] is a generalization of DropOut and stochastic depth. Swapout involves dropping of individual units or to skip the blocks randomly. Embarking on a distinctive approach of reducing the test time, distilling the knowledge in a network [100] transferred the "knowledge" from ensembles to a single model. Gradual DropIn or regularised DropIn [244] of layers starts from a shallow network wherein the layers are added gradually. DropIn trains the exponential number of thinner networks, similar to DropOut, and also shallower networks.

All the aforementioned methods provided an ensemble of networks by sharing the weights. There have been attempts to explore explicit ensembles in which models do not share the weights. Snapshot ensembling [112] develops an explicit ensemble without sharing the weights. The authors exploited good and bad local minima and let the stochastic gradient descent (SGD) converge M-times to local minima along the optimization path and take the snapshots only when the model reaches the minimum. These snapshots are then ensembled by averaging at multiple local minima for object recognition. The training time of the ensemble is same as that of a single model. The ensemble out is taken as the average of the snapshot outputs at multiple local minimas. Random vector functional link network [203] has also been explored for creating the explicit ensembles [239] where different random initialization of the hidden layer weights in a hierarchy diversifies the ensemble predictions.

Explicit/Implicit approach produce ensemble out of a single network at the expense

Papers	Contribution
[245]	Introduced Dropout (Random skipping of units)
[274]	Introduced DropConnect (Random skipping of connections)
[111]	Deep networks with stochastic depth (Random skipping of blocks)
[242]	Introduced swapout (Hybrid of Dropout and stochastic depth approach)

Table 2.3: Implicit / Explicit ensembles.

of base model diversity [31] as the lower level features across the models are likely to be same. To alleviate this issue, branching based deep models [88] branch the network to induce more diversity. As different initializations of the neural network leads to different local minimas, hence, Xue et al. [295] proposed deep ensemble model wherein ensemble of fully convolution neural network over multiloss module with coarse fine compensation module resulted in better segmentation of central serous chorioretinopathy lesion. Multiple neural networks with different initializations or multiple loss functions resulted in better diversity of an ensemble.

2.2.6 Homogeneous/Heterogeneous ensembles

Homogeneous/Heterogeneous ensembles involve training a group of base learners either from the same family or different families, respectively. Hence, base learners of an ensemble must be as diverse as possible and each base model must be performing better than a random guess. The base learner can be a decision tree, neural network, or any other learning model.

In homogeneous ensembles, the same base learner is used multiple times to generate the family of base classifiers. However, the key issue is to train each base model such that the ensemble model is as diverse as possible i.e., no two models are making the same error on a particular data sample. The two most common techniques of inducing randomness in homogeneous ensemble is either sampling of training set multiple times thereby training each model on a different bootstrapped sample of the training data or sampling the feature space of the training data and train each model on different feature subset of the training data. In some ensemble models like random forest [23] which used both these techniques for introducing diversity in an ensemble of decision trees.

In neural networks, training each model independently with different initialization also induces diversity. However, deep learning models have high training costs and hence, training of multiple deep learning models is not a feasible option. Some attempts like horizontal vertical voting of deep ensemble [292] have been made to obtain ensemble of deep models without independent training. Temporal ensembling [149] trains multiple models with different input augmentation, different regularization and different training epochs. Training of multiple deep neural networks for image classification [45] and for disease prediction [82] showed that better performance is achieved via ensembling of multiple networks and averaging the outputs. Despite these models, training multiple deep learning models for ensembling is an uphill task as millions or billions of parameters need to be optimized. Hence, some studies have used deep learning in combination with the traditional models to build the heterogeneous ensemble models enjoying the benefits of lower computation and higher diversity. Heterogeneous ensemble for default prediction [157] is an ensemble of the extreme gradient boosting, deep neural network and logistic regression. Heterogeneous ensemble for text classification [132] is an ensemble of multivariate Bernoulli naïve Bayes, multinomial naïve Bayes, support vector machine, random forest, and convolutional neural network learning algorithms. Using different perspectives of data, model and decision fusion strategies, heterogeneous deep network fusion [254] showed that complex heterogeneous fusion architecture is more diverse and hence, shows better generalization performance.

2.2.7 Decision fusion strategies

Ensemble learning trains several base learners and aggregates the outputs of base learners using some rules. The rule used to combine the outputs determines the effective performance of an ensemble. Most of the ensemble models focus on the ensemble architectures followed by their naive averaging to predict the ensemble output. However, naive averaging of the models, followed in most of the ensemble models, is not data adaptive and leads to less optimal performance [126] as it is sensitive to the performance of the biased learners. As there are billions of hyperparameters in deep learning architecture, hence, the issues of overfitting may also lead to the failure of

some base learners. Hence, to overcome these issues, approaches like Bayes optimal classifier and super learner have been followed [126].

The different approaches followed in the literature for combining the outputs of the ensemble models are:

2.2.7.1 Unweighted model averaging

Unweighted averaging of the outputs of the base learners in an ensemble is the most followed approach for fusing the decisions in the literature. Here, the outcomes of the base learners are averaged to get the final prediction of the ensemble model. Deep learning architectures have high variance and low bias, thus, simple averaging of the ensemble models improve the generalization performance due to the reduction of the variance among the models.

The averaging of the base learners is performed either on the outputs of the base learners directly or on the predicted probabilities of the classes via softmax function:

$$P_i^j = softmax^j(O_i) = \frac{O_i^j}{\sum_{k=1}^{K} exp(O_k^j)}, \tag{2.2}$$

where P_i^j is the probability outcome of the i^{th} unit on the j^{th} base learner, O_i^j is the output of the i^{th} unit of the j^{th} base learner and K is the number of the classes.

Unweighted averaging is a reasonable choice when the performance of the base learners is comparable, as suggested in [96, 241, 253]. However, when the ensemble contains heterogeneous base learners naive unweighted averaging may result in sub-optimal performance as it is affected by the performance of the weak learners and the overconfident learners [126]. The adaptive metalearner should be good enough to adaptively combine the strengths of the base learners as some learners may have lower overall performance but may be good at the classification of certain subclasses and hence, leading to better overall performance.

2.2.7.2 Majority voting

Similar to unweighted averaging, majority voting combines the outputs of the base learners. However, instead of taking the average of the probability outcomes, majority voting counts the votes of the base learners and predicts the final labels as the label with the majority of votes. In comparison to unweighted averaging, majority voting is less biased towards the outcome of a particular base learner as the effect is mitigated by majority vote count. However, favouring of a particular event by most of the similar base learners or dependent base learners leads to the dominance of an event in an ensemble model. In majority voting, the analysis in [146] showed that the pairwise dependence among the base learners plays an important role and for the classification of images, the prediction of shallow networks is more diverse compared to the deeper networks [43]. Hence, in [126] hypothesised that the performance of the majority voting based shallow ensemble models is better compared to the majority based deep ensemble models.

2.2.7.3 Bayes optimal classifier

In Bayesian method, hypothesis h_j of each base learner with the conditional distribution of target label t given x. Let h_j be the hypothesis generated on the training data D evaluated on test data (x, t), mathematically, $h_j(t|x) = P[y|x, h_j, D]$. With Bayes rule, we have

$$P(t|x, D) \propto \sum_{h_j} P[t|h_j, x, D]P[D|h_j]P[h_j] \tag{2.3}$$

and the Bayesian optimal classifier is given as:

$$\underset{t}{argmax} \quad \sum_{h_j} P[t|h_j, x, D]P[D|h_j]P[h_j], \tag{2.4}$$

where $P[D|h_j] = \Pi_{(t,x)\in D} h_j(t|x)$ is the likelihood of the data under h_j. However, due to overfitting issues this might not be a good measure. Hence, training data is divided into two sets-one for training the model and the other for evaluating the

model. Usually the validation set is used to tune the hyperparameters of a model.

Choosing prior probabilities in Bayes optimal classifier is difficult and hence, usually set to uniform distribution for simplicity. With a large sample size, one hypothesis tends to give larger posterior probabilities than others and hence, the weight vector is dominated by a single base learner and the Bayes optimal classifier would behave as the discrete superlearner with a negative likelihood loss function.

2.2.7.4 Stacked generalization

Stacked generalization [288] works by reducing the biases of the generalizer(s) with respect to a learning set. To obtain the good linear combination of the base learners in regression, cross-validation data and least squares under non-negativity constraints are used to get the optimal weights of combination [20]. Consider the linear combination of the predictions of the base learners $f_1, f_2, \cdots, f_m$ given as:

$$f_{stacking}(x) = \sum_{j=1}^{m} w_j f_j(x), \tag{2.5}$$

where w is the optimal weight vector learned by the meta learner.

2.2.7.5 Super learner

Inspired by the cross validation for choosing the optimal classifier, Van der Laan et al. [267] proposed super learner which is weighted combination of the predictions of a base learner. Unlike the stacking approach, it uses cross validation approach to select the optimal weights for combining the predictions of the base learners.

With smaller datasets, cross validation approach can be used to optimize the weights. However, with the increase in size of the data and the number of base learners in a model, it may not be a feasible option. Instead of optimizing the five-fold cross validation, single split cross validation can also be used for optimizing the weights to get optimal combination [127]. In deep learning models, usually, a validation set is used to evaluate the performance instead of using the cross validation.

2.2.8 Unsupervised learning

Unsupervised learning is another group of machine learning techniques. The fundamental difference between it and supervised learning is that unsupervised learning usually handles training samples without corresponding labels. Therefore, the primary usage of unsupervised learning is to do clustering. Diversity among the base learners is important in the ensemble learning. To create diverse clusters, several approaches can be applied: using different sampling data, using different subsets of the original features, and employing different clustering methods. Sometimes, even some random noise can be added to these base models to increase randomness, which is good for ensemble methods according to [14]. After receiving all the outputs from each cluster, various consensus functions can be chosen to obtain the final output based on the user's requirement [269]. The ensemble clustering is also known as consensus clustering.

Zhou and Tang [318] explored ensemble methods for unsupervised learning and developed four different approaches to combine the outputs of these clusters. In recent years, some new ensemble clustering methods have been proposed that illustrated the priority of ensemble learning [108, 109, 314]. Most of the clustering ensemble methods are based on the co-association matrix solution, which can be regarded as a graph partition problem. Besides, there is some focus on integrating the deep structure and ensemble clustering method. Liu et al. [164, 165] showed that ensemble unsupervised representation learning with deep structure via auto-encoder can be applied in large scale data and extended it to the vision field. Shaham et al. [232] used restricted Boltzmann machine with a single hidden neuron for crowd sourcing and proposed an RBM-based Deep Neural Net (DNN) for unsupervised ensemble learning. The unsupervised ensemble methods have also been used in the field of Natural Language Processing. Alami et al. [2] demonstrate that an ensemble of unsupervised deep neural network models that use Sentence2Vec representation as the input has the best performance according to the experiments. Hassan et al. [94] included four semantic similarity measures which improve the performance on the semantic textual similarity (STS) task. The unsupervised ensemble method is also widely used for tasks that lack

Papers	Contribution
[318]	Develop several approaches for cluster ensemble
[164]	Large scale unsupervised ensemble clustering method based on graph partition method
[165]	Vision clustering method integrates the deep structure and ensemble clustering method
[232]	RBM-based deep neural net applied in crowd sourcing and unsupervised ensemble learning
[2]	Unsupervised text summarization via ensemble method
[94]	Unsupervised ensemble method for assessing the semantic similarity
[1]	Medical image classification via unsupervised feature learning and ensemble
[148, 163]	Unsupervised ensemble method for retinal vessel segmentation
[32]	Subcellular localization prediction for long non-coding RNAs

Table 2.4: Unsupervised ensemble models.

annotation, such as the medical image. Ahn et al. [1] proposed unsupervised feature learning method integrated ensemble approach with a traditional convolutional neural network. Lahiri et al. [148] employed unsupervised hierarchical feature learning with ensemble sparsely autoencoder on retinal blood vessels segmentation task, meanwhile, Liu et al. [163] also proposed an unsupervised ensemble architecture to automatically segment retinal vessel. Besides, there are also some ensemble deep methods working on localization predicting for long non-coding RNAs [32].

2.2.9 Semi-supervised and active learning

Semi-supervised learning [36, 320] is a machine learning method that falls between supervised and unsupervised learning. It allows the dataset to contain a small number of labeled data and a large number of unlabeled data. The exploiting of unlabeled data can help to achieve a strong generalization.

There is a point of view that using unlabeled data to boost is good enough to achieve acceptable results, so there is no need to employ ensemble methods. Another view claims that ensemble learning models can tackle different kinds of tasks. And therefore using semi-supervised learning is redundant. However, Zhou [316] illustrated the benefits of combining semi-supervised learning and ensemble learning with theoretical proof.

Bennett et al. [10] proposed a successful adaptive semi-supervised ensemble method that won the NIPS 2001 unlabeled data competition. It maximized the function space of both labeled and unlabeled data by assigning "pseudo-classes" to these unlabeled

data. And the experimental results showed that both neural network and decision tree are suitable for this method and have strong performance on benchmark datasets. Furthermore, in recent years, some new semi-supervised ensemble methods have been proposed to deal with various tasks.

With the development of semi-supervised deep learning, ensemble methods started to be integrated with it. Li et al. [156] proposed an ensemble semi-supervised deep neural network (DNN), acoustic models, for automatic speech recognition. Here, the sub-models are trained with different labels in different GPUs and the ensemble training framework is inspired by Kaldi toolkit. Wang et al. [279] proposed an ensemble self-learning method to enhance semi-supervised performance and extracting adverse drug events from social media. In the semi-supervised classification area, deep coupled ensemble learning method combined with complementary consistency regularization gave state of the art performance [152]. Some results have also been achieved with semi-supervised ensemble learning on some datasets where the annotation is costly. Pio et al. [211] employed an ensemble method to improve the reliability of micro RNA:micro RNA predicted interactions.

Active learning is another popular topic in the deep learning area, which is also often used in conjunction with semi-supervised learning and ensemble learning. The key sight of this is to train the algorithm from less annotated data. Some conventional active learning algorithms, such as Query-By-Committee, have already adopted the idea of ensemble learning. In [177, 178], an ensemble method generated a diverse committee. Beluch et al. [8] discussed the power of ensembles for active learning is significantly better than Monte-Carlo Dropout and geometric approaches. Sharma and Rani [234] shows some applications in drug-target interaction prediction. Ensemble active learning is also available to conquer the concept drift and class imbalance problem [300].

In the semi-supervised/active learning domain, the foremost purpose of ensembles is to construct diverse subspaces that can be used to fill in the effects of missing labeled data. Since, the ensemble approach is able to cover data with different tendencies at the same time, it can achieve excellent results in semi-supervised and active learning.

Papers	Contribution
[10]	Adaptive semi-supervised ensemble method
[156]	Semi-supervised automatic speech recognition
[279]	Ensemble of spatial and non-spatial transformations to train a semi-supervised network
[166]	Extracting adverse drug events from social media
[152]	Semi-supervised deep coupled ensemble learning for image classification
[211]	Semi-supervised ensemble learning to predict microRNA target
[8]	Ensemble-based uncertainties consistently outperforms than other active learning method
[234]	Drug target interaction prediction using active learning
[300]	Ensemble active learning for concept drift and class imbalance

Table 2.5: Semi-supervised ensemble models.

2.2.10 Reinforcement learning

Reinforcement learning (RL) [252] deals with problems which need an agent to take actions in an uncertain and complex environment. Unlike supervised learning or unsupervised learning which has training samples, the training process of reinforcement learning is based on a notion called reward. Each time when the agent acts, a corresponding reward or penalty will be received by the agent. Therefore, the object of reinforcement learning is to maximize the total reward.

Wiering and Van Hasselt [287] proposed different ensemble algorithms to combine several popular RL models: Q-learning [282], Sarsa [228, 251], actor-critic (AC) [252], QV-learning [286], and AC learning automaton (ACLA) [286]. They used the weighted majority voting method, the rank voting method, the Boltzmann multiplication method, and the Boltzmann addition method as the decision fusion strategies to reach the final output of an ensemble model. Their experimental results indicated that the ensembling via weighted majority voting method and Boltzmann multiplication method significantly outperforms the single RL model.

There have been other attempts that tried to combine ensemble learning and reinforcement learning. Partalas et al. [207] used RL to decide whether to include a particular classifier into the ensemble. Moreover, ensembles of neural networks are used to achieve a more robust learning process and more reliable near-optimal policies in [92].

With the development of deep learning, some researchers have implemented deep reinforcement learning, which combines deep learning with Q-learning algorithm [182].

Ensemble methods in deep Q learning have decent performance. Chen et al. [41] proposed an ensemble network architecture for deep reinforcement learning which integrated temporal ensemble and target values ensemble. Develop a human-like chat robot is a challenging job, by incorporating deep reinforcement learning and ensemble method, Cuayáhuitl et al. [50] integrated 100 deep reinforcement learning agents, the agents are trained based on clustered dialogues. They also demonstrate that ensemble of DRL agents have better performance than a single variant or Seq2Seq model. Stock trading is another topic where ensemble deep reinforcement learning has achieved a promising result.Carta et al. [34] found the single supervised classifier is inadequate to deal with complex and volatile stock market. They employed hundreds of neural networks to pre-process the data, then combined several reward-based meta learners as a trading agency. Moreover, Yang et al. [297] trained an ensemble trading agency based on three different metrics: proximal policy optimization (PPO), advantage actor-critic (A2C), and deep deterministic policy gradient (DDPG). The ensemble strategy combines the advantages of the three different algorithms. Besides, some researchers used ensemble strategy to solve the disease-prediction problem. Tang et al. [256] proposed a model that consists of several sub-models which are in response to different anatomical parts.

In this aspect of reinforcement learning, the ensemble is more of an auxiliary tool, by federating the decisions made by different agents to get a more consistent performance. At the same time, the ensemble approach shows generality in the field of reinforcement learning, and it is applicable to all kinds of different reinforcement learning strategies.

2.2.11 Online/Incremental, multi-label learning

In recent years, online/incremental learning has received more and more attention [5]. With the limitation of getting the complete data in real-world problems, online/incremental learning has been applied to various tasks like learning social representations [210], fog computing [293], identifying suspicious URLs [171], etc. Some conventional ensemble learning methods have also been extended to online versions

such as bagging and boosting [198]. These online versions proved to have similar results as batch models with theoretical guarantees in [12, 198]. Other examples that employed online ensemble learning models were used to deal with the presence of concept drift [181], power load forecasting [83, 217], myoelectric prosthetic hands surface electromyogram characteristics [61], etc. In [51], the author proposed an ensemble incremental learning with pseudo-outer-product fuzzy neural network for traffic flow prediction, real-life stock price, and volatility predictions, etc. Work done in [186, 204, 215] propose Learn++ and its variants in the data fusion area, which is known as the combination of data or information from several sources, that demonstrate the effectiveness of ensemble incremental learning methods. Besides, some scholars are also working on developing different algorithms for ensemble incremental learning, in [187], the author employ a dynamically modified weighted majority voting strategy to combine the sub-classifiers. Tang et al. [257] proposed negative correlation learning (NCL) based approach for ensemble incremental learning. Zhao et al. [313] suggests that heterogeneous bagging based ensemble strategy perform better than boosting based Learn++ algorithms and some other NCL methods.

With the continuous increase in availability of data, there have been problems that need to assign each instance multiple labels. For example, the famous movie *"The Shawshank Redemption"* is a drama, but it can also be classified as crime fiction or mystery. This kind of classification problem is named multi-label classification [264]. It can also be combined with ensemble learning, and a typical application is the RAndom k-labELsets (RAKEL) algorithm [265]. Here, several single-label classifiers are trained using small random subsets of actual labels. Then the final output is carried out by a voting scheme based on the predictions of these single classifiers. There are also many variants of RAKEL proposed in recent years [135, 185, 278]. Shi et al. [238] proposed a solution for multi-label ensemble learning problem, which construct several accurate and diverse multi-label based basic classifiers and employed two objective functions to evaluate the accuracy and diversity of multi-label base learners. Li et al. [155] proposed an ensemble multi-label classification framework based on variable pairwise constraint projection. Xia et al. [291] proposed weighted stacked ensemble scheme

Papers	Contribution
[181]	Analysis the impact of ensemble algorithms with the presence of concept drift
[83, 217]	Ensemble incremental learning for electric load forecasting
[51]	Ensemble pseudo outer-product fuzzy neural network for time series prediction ability
[186, 204, 215]	Learn++ and its variants
[187]	Dynamically modified weighted majority voting strategy to combines the sub-classifiers
[257]	Negative correlation learning based ensemble incremental learning
[313]	Heterogeneous bagging based ensemble strategy incremental learning

Table 2.6: Online/Incremental ensemble models.

and employed sparsity regularization to facilitate classifier selection and ensemble construction. Besides, there are many applications of ensemble multi-label methods. Some publications employ multi-label ensemble classifier to explore the protein, such as protein subcellular localization [86], protein function prediction [298], etc. Multi-label classifier are also utilized in predicting the drug side effects [309], predicting the gene prediction [230], etc. Moreover, there is another critical ensemble multi-label algorithm called ensemble classifier chains (ECC) [219]. This method involves binary classifiers linked along a chain, the first classifier is trained using only the input data, and then each subsequent classifier is trained on the input space and all previous classifiers in the chain. The final prediction is obtained by the integration of the predictions and selection above a manually set threshold. Chen et al. [39] proposed an ensemble application of convolutional and recurrent neural networks to capture both the global and the local textual semantics and to model high-order label correlations.

The ensemble approach plays an important role in online learning and multi-label tasks. Many ensemble approaches can be considered as an online learning strategy. By combining the predictions of the model trained on the initial data with the model trained on the added data, the ensemble approach demonstrates a solid boost. In the multi-label domain, ensemble is also one of the main mean, fusing the results of multiple single-label classifiers is very intuitive solution to the multi-label task.

2.3 Decision trees and their ensembles

Decision tree algorithm is a commonly used classification model due to its simplicity and better interpretability. Decision tree uses divide and conquer approach

Papers	Contribution
[135, 185, 265, 278]	Random k-labeLsets (RAKEL) algorithm and its variants.
[238]	Multi-label ensemble learning constrained by two different objective functions.
[155]	Multi-label classification framework based on variable pairwise constraint projection.
[291]	Multi-label classification with weighted classifier selection and stacked ensemble.
[86, 298]	Protein subcellular localization and Protein function prediction by ensemble multi-label classifier.
[309]	Predict the drug side effects.
[230]	Predict the gene function.
[219]	Classifier chains for multi-label classification.
[39]	Ensemble CNN and RNN for multi-label text categorization.

Table 2.7: Multi-label ensemble models.

to recursively partition the data. The recursive partition of the tree is sensitive to perturbation of the input data, and hence, results in an unstable classifier. Hence, it possess high variance and low bias. The ensemble methodology can be used with unstable classifiers to further improve their generalization performance.

In decision trees, mostly two approaches are used: bootstrap aggregation (bagging) [18] and boosting [65]. In bagging, independent classifiers are trained on different bootstrap versions of data points selected with replacement from the training data points. In boosting, individual classifiers are trained in sequential manner with next level classifier taking the training instances which were hard to classify for the current classifier. In each training iteration, weights are updated for each sample of the training set. The samples which are hard to classify will more likely to be sampled than the other ones.

In the literature, mostly random forest [23] and rotation forest [224] are used in ensemble methodology. In a recent comprehensive survey of 179 classifiers over 121 datasets, RaF emerged as the best classifier [62]. With the randomly chosen subspaces of data samples at each non-leaf node, classification ensembles are build using the concepts of bagging [18] and random-subspaces [103] in each decision tree. The decision of individual base classifiers are independent as each classifier is trained on different bootstrapped versions of the training data which follow the same distribution as that of the whole population. Variance of each base classifier is increased due to the random subspace strategy at each node. RaF has been applied across different problems like object recognition and image segmentation [68], micro-arrays [123], time series [237] and spectral data [179]. With little tuning of parameters [105], RaF achieves comparable performance as that of other non-linear algorithms. Additionally, RaF is

able to choose the relevant set of features even if large number of irrelevant features are present [123, 161, 162].

The rotation forest (RoF) [224] is similar to RaF with the difference of two points: First, each tree in RoF is trained in a rotated feature space of the whole training data. Second, the best splitting feature is searched among all the features at each node of the tree. According to Rodriguez et al. [224], a small rotation in feature axes generates very different tree models. Also, RoF works comparatively better than the RaF. RoF has been applied to various research areas like bioinformatics [246, 290], business, economics, and medical fields [167, 199].

To obtain the better generalization performance, various hyperparameters of the random forest need to be chosen optimally. These hyperparameters include number of decision trees in a forest (ntree), number of candidate features for evaluation at a given non-leaf node (mtry), and number of samples in an impure node (node size or minleaf) (we will use minleaf and node size interchangeably). To get these parameters optimally, different studies have been proposed. Analysis of tuning process [63, 216], sensitivity of the parameters [107], effect of number of trees in an ensemble [6, 98, 196] provide insight how these parameters affect the model performance. To obtain the optimal number of candidate features, different methods [17, 90] have been proposed. Analysis of optimal sample size in bagging [176] and estimation of tree size via combination of random forest with adaptive nearest neighbours [162] result in the better choice of the hyperparameters.

Mostly the decision trees have been classified into two categories: univariate (axis-parallel or orthogonal) [6] and multivariate (oblique) [188] decision trees. In univariate decision trees, best split among several features is chosen based upon some impurity criteria. While as in multivariate decision trees, all or part of the features are evaluated for the best split. Broadly speaking, univariate decision tree can approximate any oblique decision boundary using a large number of stair-like decision boundaries. At each non-leaf node of a decision tree, hyperplane separates the data such that separation at the child nodes in the next level is easier. At that stage, the hyperplane itself may or may not be a good classifier [173]. Several impurity measures like Gini

index, entropy, a towing rule are used by the decision trees to evaluate the best splitting feature. For each feature in the axis parallel decision trees, a predefined criteria is evaluated at each node to find the best cutting plane and impurity score. Impurity criterion measures the different class skewness distribution on the data samples reaching to the node. Low impurity score is assigned to distributions closer to the uniform distribution and high score is assigned to the distributions where the one class has majority over the others. Impurity measure is optimized in most of the decision tree algorithms for choosing the best split. However, some impurity measures may not be differentiable with respect to the hyperplane parameters. In such cases, other techniques are used for finding the best hyperplane. For instance, deterministic hill climbing algorithm is used by CART-LC [24], randomized search based on CART-LC is used by OC1 [189]. Both these approaches [24, 189] suffer from local optimum problem. Hence, multiple restarts or trails are done to reduce the chances of ending with local optima. In high dimensional feature space, both are computationally expensive as they search in one dimension at a time. Some evolutionary approaches [35, 209] have been employed to optimize in all dimensions. All these impurity measures depend on the class distribution on each side of the hyperplane [173]. Altering the class labels of the data without altering the features of each class on either side of hyperplane will not make any change in the impurity measures. It happens due to the reason that geometric structure is not captured by the impurity measures. However, geometric structure involves the internal data structure by measuring the distance between the data point and the decision hyperplane. Hence, any change to the relevant features will change the decision hyperplane. To generate the geometric decision trees, support vector machines were used to generate the hyperplane capturing the geometric structure of the class distributions. Decision tree based on support vector machines [308] generated optimal hyperplane with standard support vector machines using radial basis function (RBF) kernel. Multisurface proximal support vector machine (MPSVM) [172] finds the two hyperplanes such that each hyperplane is closer to the data samples of one class and farthest from the data samples of other class, and the test data samples are classified based on their distance from the hyperplanes.

In geometric decision tree [173], MPSVM finds the two clustering hyperplanes, and based on the impurity measure choose the angle bisector of these hyperplanes. Manwani and Sastry [173] grouped the majority class into one class and rest into other class to tackle the multi-class problem. Sample size problem arises when the data samples reaching to the children nodes are fewer. To reduce this problem, [40] used NULL space method. For more details regarding the decision trees, readers may refer to [226]. The oblique decision tree ensemble classifier [303] uses MPSVM to enhance the performance of the different base classifiers such as random forest (RaF), rotation forest (RoF), and random subspace rotation forest (RRoF) and named these classifiers as MPRaF, MPRoF, and MPRRoF, respectively. In each non-leaf node of the decision tree, MPSVM generates splitting plane of the decision trees. Different regularization techniques were applied in oblique decision trees based on MPSVM as the matrices appearing in the MPSVM formulation are positive semi-definite.

Recent study of double random forest [91] evaluated the effect of node size on the performance of the model. The study revealed that the prediction accuracy could be improved if there is a way to generate deeper decision trees. The authors showed that the largest tree grown on a given data by the standard random forest might not be sufficiently large to give the optimal performance. Hence, double random forest [91] generated decision trees that are bigger than the ones in standard random forest. Instead of training each decision tree with different bags of training set obtained via bagging approach at the root node, double random forest [91] generates each tree with the original training set and uses bootstrap aggregation at each non-leaf node of the decision tree to obtain the best split. However, both the random forest and double random forest are univariate decision trees and hence, ignore the geometric class distributions that results in lower generalization performance.

The brief overview of the related ensembles of decision trees is given as follows:

2.3.1 Random forest

Originally given by Breiman [23], RaF uses the idea of bagging and random subspaces. RaF relies on the series of tree predictors, where each tree takes values from

Algorithm 2.1 Random forest.

Training Phase:

Given:

$X = M \times n$ is the dataset with M samples each with feature length n.
$Y = M \times 1$ are data labels corresponding to the training dataset.
L is the ensemble size i.e. number of trees in the forest.
Each tree in the random forest is represented as T_i, where $i = 1, \cdots, L$.
"$mtry$" refers to the randomly selected features for splitting at each non-leaf node.
"$minleaf$" is the maximum number of samples in an impure node.

1) Each tree T_i is build using the bootstrapped versions of the training data X with replacement.
2) At each non-leaf node, the best feature split is selected among the "$mtry$" randomly selected features from the training data.
3) Repeatedly execute step 2 until one of the conditions is met:
- Node becomes pure.
- Node contains number of samples less than or equal to $minleaf$.

Classification Phase:

For classification of a test sample, it is pushed down each tree in the forest, and each tree in the forest assigns the vote to the given sample data. Then the predicted label for the sample data is the one with the highest number of votes among the forest.

a randomly initialized vectors which are sampled independently and have same distribution across all the trees in a forest. Diversity among the base classifiers is increased by combining the concepts of bagging and random subspaces. Each tree is trained on the bootstrapped version of the training data with random subspace of features. The number of random subspace features selected control the number of tests (split tests) to be performed at each node, as each feature is evaluated for the split. Among these features, the one which makes the node more pure is selected.

The algorithm of RaF is given in Algorithm 2.1.

2.3.2 Rotation forest

Before constructing each tree, RoF [224] uses PCA to transform or rotate the dataset. The different decision trees in the forest are uncorrelated as each tree uses a distinct rotation matrix. The heuristics used in RoF is that features are extracted from a subset of features and then a full feature set is reconstructed for each classifier.

The authors used all the principal components to construct the feature sub space. The diversity of the model comes with the difference in the possible feature subsets. With rotation heuristic, the number of different partitions of the feature set $T = \frac{n!}{K!(M!)^K}$, where K is the number of subsets of size M, and n is the sample feature length. Different classifiers are generated on different partitions. Under the assumption that partitions of the feature sets are equally likely, the probability that all classifiers will be different is P (different classifiers)$= \frac{T!}{(T-L)!T^L}$, where L is the ensemble size.

The algorithm of RaF is given in Algorithm 2.2.

Algorithm 2.2 Rotation forest.

Training Phase:

Given: $X = M \times n$ is the dataset with M samples each with feature length n.
$Y = M \times 1$ are class labels corresponding to the training data set.
$L :=$ Number of decision trees in an ensemble.
S is the number of subsets and $\{C_1, \cdots, C_k\}$ is the set of class labels.
F is the feature set.
For $i = 1, \cdots, L$

1) Prepare the rotation matrix R_i^a :
   - Split F into S subsets: $F_{i,j}$ with $j = 1, \cdots, S$.
   - For $j = 1, \cdots, S$
     - Let $X_{i,j}$ be the data set X for the features in $F_{i,j}$.
     - Eliminate a random subset of classes from $X_{i,j}$.
     - Select a bootstrap sample from $X_{i,j}$ of size 75% of the number of samples in $X_{i,j}$. Denote the new set as $X_{i,j}^{'}$.
     - Apply PCA on $X_{i,j}^{'}$ to get the coefficients in a matrix $Z_{i,j}$.
   - Arrange the $Z_{i,j}$, for $j = 1, \cdots, S$ in a rotation matrix R_i as in equation (2.6).
   - For construction of R_i^a, rearrange the columns of R_i to match order of the features in F.

2) With (XR_i^a, Q) as the training data, build the classifier D_i.

Classification Phase:

- For a given sample x, let the classifier D_i assigned the probability $d_{i,j}(xR_i^a)$ to the hypothesis that x belongs to class C_j. For each class C_j, calculate the confidence by the average combination method:

$$\mu_j(x) = \frac{1}{L} \sum_i d_{i,j}(xR_i^a), \quad j = 1, \cdots, k. \tag{2.6}$$

- Assign x to the class with the largest confidence.

2.3.3 Double random forest

Double random forest [91] is an ensemble of decision trees based on the concept of bagging and the random subspace method. Unlike standard random forest wherein each decision tree is generated on the boostrapped samples of the training set, double random forest trains each decision tree on the original training set. This results in more unique features in the data used in training the double random forest than standard forest. The more number of unique instances leads to larger decision trees and hence, better generalization performance. Double random forest uses bootstrap sampling momentarily at each non-leaf node. Once the best feature split is chosen among the randomly chosen subset of the features from the bootstrap samples, the splitting of the original data is done and hence, original data is sent down the decision tree resulting in more number of unique instances. The algorithm of the double random forest is given in Algorithm 2.3.

2.3.4 Rotation random forest

In rotation random forests [305], the objective is to rotate or transform the input feature space at a given node. Here, instead of applying the transformation to the whole data, the data is transformed at each node. The transformation at each node is different as different subspace of features is chosen at node while building CART. This results in the improved diversity among the weak learners of an ensemble. Here, we discuss the two rotations of random forest with PCA and linear discriminant analysis (LDA) known as PCA based random forest and LDA based random forest, respectively. The Algorithm of PCA based random forest is given in Algorithm 2.4.

The algorithm of LDA based random forest is similar to Algorithm 2.4, except at Step 3, where instead of calculating the total scatter matrix S_t, within class S_w and between class scatter matrix S_b are calculated. Then, all the generalized eigenvectors of (S_b, S_w) are calculated i.e. $S_b \times \alpha = \lambda \times S_w \times \alpha$ where λ is the generalized eigenvalue of the generalized eigenvector.

Algorithm 2.3 Double random forest.

Training Phase:

Given:

$X := M \times n$ be the training set with M number of samples with feature size n.
$X_i := M_i \times n_i$ be the training samples reaching to a node i, with M_i number of samples with feature size n_i.
$Y := M \times 1$ be the labels of the training data.
$L :=$ Number of decision trees in an ensemble.
"mtry": number of candidate features to be evaluated at each non-leaf node.
"nodesize" or *"minleaf"*: maximum number of samples in an impure node.

For each decision tree, T_i for $i = 1, 2, \cdots, L$

[1] Use training data X.

[2] Generate the decision tree T_i with random feature subset and random bootstrap instances using X:
For a given node d with data X_d:

(i) if $M_d > M \times 0.1$
Generate bootstrap sample X_d^* from X_d.
else
$X_d^* = X_d$.

(i) Choose *"mtry"*$= \sqrt{n}$ number of features from the given feature space of D_d^*.

(ii) Select the best split feature and the cutpoint among the random feature subset X_d^*.

(iii) With the optimal split feature and the cutpoint with X_d^*, split the data X_d into child nodes.

Repeat steps (i)-(iii), until one of the conditions is met:

- Node becomes pure.
- Number of samples reaching a given node is less than or equal to *minleaf*.

Classification Phase:

For a test sample x_i, use the decision trees of the forest to generate the label of the test sample. The predicted label of the test sample is given by the majority voting of labels generated by the decision trees of an ensemble.

Algorithm 2.4 PCA based random forest.

Training Phase:

Given:

$X := M \times n$ be the training set with M number of samples with feature size n.

$Y := M \times 1$ be the labels of the training data.

$L :=$ Number of decision trees in an ensemble.

"mtry": number of candidate features to be evaluated at each non-leaf node.

"nodesize" or *"minleaf"*: maximum number of samples in an impure node.

For each decision tree, T_i for $i = 1, 2, \cdots, L$

[1] Generate the decision tree T_i with random feature subset and random bootstrap instances using X.

[2] For a given node, choose *"mtry"*= $\sqrt{n}$ number of features from the given feature space.

[3] Calculate total scatter matrix S_d using "mtry" features.

[4] Calculate all the eigenvectors of S_d, denoted by V.

[5] Calculate the data transformation using all the eigenvectors V as, $X^*_{PCA} = X * V$.

[6] In the PCA space, search the best feature split.

[7] With the optimal split feature and the cutpoint, split the data X into the child nodes.

Repeat steps [2]-[7], until the stopping criteria is met.

Classification Phase:

For a test sample x_i, use the decision trees of the forest to generate the label of the test sample. At each non-leaf node, the test data sample is rotated with the same matrix V generated in the training stage. The predicted label of the test sample is given by the majority voting of labels generated by the decision trees of an ensemble.

2.3.5 Oblique decision tree ensemble via multisurface proximal support vector machine

The oblique decision tree ensemble classifier [303] uses MPSVM to enhance the performance of the different base classifiers such as random forest, rotation forest, and random subspace rotation forest and name these classifiers as MPRaF, MPRoF, and MPRRoF, respectively. The multisurface proximal support vector machines (MPSVM) generate the clustering hyperplanes such that each plane is closer to the samples of one class and as far as possible from the samples of another class [172]. In each non-leaf node of the decision tree, MPSVM generates splitting plane of the decision trees. Different regularization techniques were applied in oblique decision trees based on MPSVM as the matrices appearing in the MPSVM formulation are positive semi-definite. To avoid this issue two regularization approaches have been used Tikhonov regularization and axis-parallel split regularization. Tikhnonov regularization adds a small constant to the diagonal entries of the matrix to be regularised. Let the matrix G is rank deficient, then G is regularized as

$$G' = G + \delta I, \tag{2.7}$$

where δ is a small constant and I is an identity matrix of appropriate dimensions. If the matrix at a particular node becomes singular, then we can always continue the building of a tree via axis-parallel split. Thus, from the root node upto the current node, MPSVM is used to generate the splits and from the current node to the leaf node, the decision tree uses axis-parallel method. Hence, heterogeneous test splits are used to complete the decision tree.

2.4 Artificial neural networks

Neural networks have been widely used for classification and regression problems, however, there are issues like slow training, local minima problem and sensitivity to learning rate that need attention [248]. Backpropagation approach is used to tune the

model parameters of the neural network. Gradient of the loss function with respect to the model parameters is calculated and the model weights are updated iteratively in the negative direction of the gradient to minimize the loss function. It suffers from local minima problem, slow convergence and learning rate sensitivity [248].

To elevate the training procedure, randomization based neural networks have been putforth. Randomization based models avoid the above enumerated issues as the optimization problem is solved via closed form method [84, 85, 231, 263]. These models show good generalization performance and training is faster [52, 284, 285]. Among the randomization based models, extreme learning machine (ELM) [113] and random vector functional link networks (RVFL) [203] are the widely used architectures. RVFL/ELM are the single layer feedforward neural networks composed of input layer, hidden layer and the output layer. The weights and biases in the hidden layer are initialized randomly within a suitable range and kept fixed while as the output layer weights are optimized either via Moore penrose pseudoinverse or least squares ridge regression method.

RVFL has direct links from input layer to the output layer which boosts the generalization performance of the model [258, 271]. The direct links regularize the network, hence, RVFL model shows better generalization performance than ELM model [97, 180]. Compared to ELM network, the direct links make the RVFL model less complex and thinner network [304]. According to Occam's Razor principle and PAC theory [130], the simpler and lesser complex model make RVFL better compared to ELM. To further boost the performance of the RVFL model, the unsupervised learning based RVFL model [311] use sparse l_1-norm autoencoder for tuning the hidden layer weights to obtain better feature representation. The autoencoder helps to learn more superior network parameters for different learning tasks.

RVFL model has been successfully used in combination with other learning models. Statistical self-organizing learning system [42] combined RVFL with hypothesis testing. It is a two stage method wherein the nodes in the enhancement layer are added incrementally which leads to the learning of optimal number of nodes efficiently. RVFL in combination with expectation minimization [115] lead to the improvement

in the training process. Ensemble model with negative correlation learning (NCL) [169] have been widely used for improving the performance of RVFL model. Fast decorrelated neural network with random weights [3] trained multiple RVFL models such that each RVFL model is decorrelated from every other model of an ensemble. Contrary to NCL, the positively correlated kernel ridge regression (KRR) model [306] was proposed. RVFL model has also been in multitude of real world applications. An ensemble of RVFL model [158] for clinker free lime content estimate uses multisource data ensemble. The combination of RVFL model with radial basis neural net [205] for offline english language script recognition, RVFL in MPEG-4 [206], pedestrian detection [280, 281] and deep RVFL [235, 239] have proved the efficiency of the model.

Here, we briefly discuss the formulation of the randomization based neural networks as follows:

2.4.1 Random vector functional link network

Random vector functional link network (RVFL) [202] uses non-linear transformation on the input feature space to get the randomized features of the input data. The non-linear features are concatenated with original features to optimize the final layer weights. Hence, the RVFL network performs the classification based on both the original and enhanced features. RVFL is a three layer network consisting of input layer, enhancement layer and the final output layer. In this network, the weights of the hidden layer a_{ij}, between the i^{th} input node and j^{th} enhancement node, are generated randomly such that the activation function for the j^{th} node $g(\cdot)$ is not saturated. For each dataset, weights are chosen from the uniform distribution within $[_S, +S]$ interval, where S is the scaling parameter. The RVFL network is essentially same as the generalized delta rule (GDR) except that the hidden layer is moved to the enhancement layer of the input feature space and the weights a_{ij} are not learned but generated randomly. The output layer weights β of the network are obtained by solving

$$y_i = d_i^T \beta, \quad i = 1, 2, \cdots, M, \tag{2.8}$$

where M represents the number of data samples, y_i is the target, d_i is the concatenated vector of original and the random features. Once the β values are calculated then the RVFL network can be expressed as:

$$f^*(X) = \sum_j (\beta_j \cdot (g(a_j^T X + b_j))), \tag{2.9}$$

where $f^*(X)$ is the approximate function of $f(X)$ as it maps the input samples $X = [x_1, x_2, \cdots, x_M]$ to the target response $Y = [y_1, y_2, \cdots, y_M]$, b_j is the bias of the j^{th} hidden node and β_j are weights for the links to output node. Solving the equation (2.8) directly may lead to overfitting. Hence, Moore-penrose pseudoinverse and l_2 norm regularized least square (ridge regression) methods are used to avoid the overfitting issues while solving the equation (2.8). Using l_2-norm regularized least square method (or ridge regression), we have

$$\sum_i \left\| y_i - d_i^T \beta \right\|^2 + c||\beta||^2 \quad i = 1, 2, \cdots, M. \tag{2.10}$$

The optimal output layer weights are given as $\beta = (D^T D + cI)^{-1} D^t Y$, where c, D and Y are the regularization parameter, concatenated data and target responses, respectively.

2.4.2 Extreme learning machine

Extreme learning machine (ELM) [113] is a randomization based feed forward neural network. Unlike RVFL, ELM is without direct links between input and output layer and bias term in the output layer. The optimization problem of the ELM is given as:

$$O = \min_{\beta} \|H\beta - Y\|^2 + c\|\beta\|^2, \tag{2.11}$$

where H represent the randomized features, and c is a penalty parameter. The final output parameters in primal and dual space are given as follows:

$$\text{Primal space: } \beta = (H^T H + cI)^{-1} H^T Y, \tag{2.12}$$

$$\text{Dual space: } \beta = H^T (H H^T + cI)^{-1} Y. \tag{2.13}$$

2.4.3 Minimum class variance extreme learning machine

The minimum class variance extreme learning machine [119] is a randomized single layer feed-forward network which exploits the training data dispersion while optimizing the output layer weights of the objective function. The optimization problem of the minimum class variance extreme learning machine is given as:

$$\begin{aligned} \underset{\beta}{min} \quad & \left\| S_T^{\frac{1}{2}} \beta \right\|_F^2 + c\xi^T \xi \\ s.t. \quad & H\beta = Y - \xi, \end{aligned} \tag{2.14}$$

where β contains the optimized output layer weights, c is the penalty parameter, S is the training data scatter matrix and H represents the randomized features. To avoid the singularity issues in S, a regularization parameter c is added.

2.4.4 Autoencoder

An autoencoder [9] is a neural network consisting of two parts i.e. encoder and decoder part. The encoder transforms the input features into some other feature space wherein the decoder is able to reconstruct the original input space. Based on the number of neurons in the hidden layer, the encoder either compresses or expands the input features [9].

Encoder: The encoder $f(\cdot)$ maps the the input data point $x \in \mathbb{R}^n$ to the hidden feature space $f(x) \in \mathbb{R}^{d'}$. Mathematically, mapping is given as

$$h = f(x) = a_h(Wx + b_h), \tag{2.15}$$

where $a_h(\cdot)$ is the activation function, $W \in \mathbb{R}^{d' \times n}$ and $b_h \in \mathbb{R}^{d'}$ is the weight matrix and bias, respectively. The encoder parameter set is $\Theta = \{W, b_h\}$.

Decoder: The decoder $g(\cdot)$ maps the hidden feature representation to the representation y as

$$y = g(h) = a_g(W'h + b_g), \tag{2.16}$$

where $a_g(\cdot)$ is the activation function, W' and $b_h \in \mathbb{R}^{d'}$ is the weight matrix and bias of the decoder parameter set Θ', respectively.

The optimization problem to learn the parameter sets Θ and Θ' for the dataset X is given as

$$arg \min_{\Theta,\Theta'} \mathbb{E}(L(X, g(f(X)))), \tag{2.17}$$

where reconstruction error is represented by L. The optimization of the autoencoder can be done either through closed form solution or via iterative procedure.

2.4.5 Sparse pre-trained random vector functional link network

Once the hidden layer parameters are optimized via autoencoder, these pretrained weights are utilised for projecting the input data and the optimization problem is given as follows:

$$O_{RVFL} = \min_{\beta} ||\hat{H}\beta - Y|| + c||\beta||^2, \tag{2.18}$$

where $\hat{H}$ is the feature projection matrix optimized via l_1 norm autoencoder, c is the regularization parameter and β represents the output layer weights. Similar to RVFL model, the objective function (2.18) can be optimized via regularized least squares

ridge regression and the optimal output weights are given as:

$$\text{Primal space: } \beta = (\hat{H}^T\hat{H} + cI)^{-1}\hat{H}^T Y, \tag{2.19}$$

$$\text{Dual space: } \beta = \hat{H}^T(\hat{H}\hat{H}^T + cI)^{-1} Y. \tag{2.20}$$

Depending upon the number of samples and features of the training dataset, choosing optimally between the primal and dual space leads to reduction in training time.

2.5 Support vector machines

Support vector machines (SVMs) [47, 151, 268] are powerful methods used in classification as well as regression problems. SVM is based on the maximum margin concept and shows better generalization performance as it implements the structural risk minimization (SRM) principle. SRM principle is at the core of the statistical learning and hence, SVM and its variants have been successfully applied across different fields including detection of faces [197], categorization of text [125], classification of electroencephalogram signals [223], extraction of features [160, 222], identification of chimera [74] and so on. Despite its better generalization performance, SVM owns high computational complexity of the order of $O(M^3)$ where M is the size of the training dataset. To overcome this drawback of higher complexity, generalized eigenvalue proximal support vector machine (GEPSVM) [172] was proposed. GEPSVM builds two hyperplanes such that each hyperplane is proximal to its own class. Motivated by GEPSVM, twin support vector machine (TWSVM) [122] formulation was proposed for classification problems. Like GEPSVM, TWSVM builds two hyperplanes such that each hyperplane is proximal to its own class and farthest from the samples of other class. Unlike SVM wherein a single large quadratic programming problem (QPP) is solved, TWSVM solves two QPPs of smaller size as compared to SVM. Solving two smaller QPPs make TWSVM approximately 4 times faster than SVM. As TWSVM implements empirical risk minimization principle, twin bounded support vector machine (TBSVM) [233] implemented the structural risk minimization principle.

Twin bounded support vector machine (TBSVM) [233] implement structural risk minimization principle to embody the marrow of statistical learning. TBSVM solves two smaller QPPs to find the two non-parallel hyperplanes. By adding the regularization term in TBSVM, structural risk is minimized by maximizing the margin.

Least squares twin support vector machine (LSTSVM) [144] involves a system of linear equations with squared loss function instead of the convex QPP. As LSTSVM is sensitive to noise and outliers, hence, energy-based least squares twin support vector machine (ELSTSVM) [190] introduced energy term to reduce the effect of noise and outliers. Tanveer et al. [260] introduced an extra regularization term to the ELSTSVM formulation, known as robust energy based least squares twin support vector machines (RELSTSVM), resulting in the optimization problems to be positive definite and hence, better generalization performance. Recent study [261] shows that RELSTSVM is the best classifier among the twin support vector machine based models.

Different from TBSVM, LSTSVM, ELSTSVM, RELSTSVM and sparse linear programming twin support vector machines [259], a new multiclass approach called twin k-class support vector classification (TWKSVC) [294] based on "1-versus-1-versus-rest" generates $k(k-1)/2$ binary classifiers for a k-class classification problem. To reduce the computational complexity of TWKSVC, least squares TWKSVC [191] introduced the equality constraints in the objective function of TWKSVC to solve a linear system of equations instead of solving a QPP.

Here, we briefly discuss the formulation of variants of twin SVM models. Assume $H = [A \;\; e]$, $G = [B \;\; e]$, where e is vector of ones of appropriate dimensions. Also, c_i be the positive penalty parameters, with $i = 1, 2, 3, 4$.

2.5.1 Twin bounded support vector machine

The linear TBSVM [233] finds the nonparallel hyperplanes

$$f_1(x) = w_1^\top x + b_1 = 0 \quad \text{and} \quad f_2(x) = w_2^\top x + b_2 = 0, \tag{2.21}$$

such that each hyperplane is closer to the data points of one class and farthest from the data points of another class. Based upon the proximity of the data samples

from two hyperplanes, it is assigned with the class label $+1$ or -1. With the objective function of one class and constraints corresponding to the other class, TBSVM solves the pair of QPPs as

$$\begin{aligned} \min_{w_1,b_1,\xi,\xi^*} \ & \frac{1}{2}c_3(||w_1||^2 + b_1^2) + \frac{1}{2}\xi^{*T}\xi^* + c_1 e^T \xi \\ s.t. \ & Aw_1 + eb_1 = \xi^*, \\ & -(Bw_1 + eb_1) + \xi \geq e, \ \xi \geq 0 \end{aligned} \tag{2.22}$$

and

$$\begin{aligned} \min_{w_2,b_2,\eta,\eta^*} \ & \frac{1}{2}c_4(||w_2||^2 + b_2^2) + \frac{1}{2}\eta^{*T}\eta^* + c_2 e^T \eta \\ s.t. \ & Bw_2 + eb_2 = \eta^*, \\ & (Aw_2 + eb_2) + \eta \geq e, \ \eta \geq 0, \end{aligned} \tag{2.23}$$

The dual of the QPP (2.22) is given as

$$\begin{aligned} \max_{\alpha} \ & e^T\alpha - \frac{1}{2}\alpha^T G(H^T H + c_3 I)^{-1} G^T \alpha \\ s.t. \ & 0 \leq \alpha \leq c_1. \end{aligned} \tag{2.24}$$

Similarly, the dual of QPP (2.23) is given as

$$\begin{aligned} \max_{\gamma} \ & e_1^T\gamma - \frac{1}{2}\gamma^T H(G^T G + c_4 I)^{-1} H^T \gamma \\ s.t. \ & 0 \leq \gamma \leq c_2, \end{aligned} \tag{2.25}$$

where γ is the Lagrange multiplier.

The optimal separating hyperplanes are given as:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (H^T H + c_3 I)^{-1} G^T \alpha, \tag{2.26}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (G^T G + c_4 I)^{-1} H^T \gamma. \tag{2.27}$$

The test sample is assigned with one of the class labels i $(i = +1, -1)$ based upon its distance from the two hyperplanes as

$$\text{Class}(x) = \arg \quad \min_{k=1,2} \frac{|w_k^T x + b_k|}{||w_k||}, \tag{2.28}$$

where $|\cdot|$ is the absolute value.

2.5.2 Least squares twin support vector machines

Least squares twin support vector machines (LSTSVM) [144] solves a pair of primal problems of twin support vector machine instead of dual problems which is extremely fast for generating the two non-parallel hyperplanes. The primal problems of LSTSVM are given as:

$$\begin{aligned} \underset{w_1, b_1, \xi^*}{min} \quad & \frac{1}{2} \|Aw_1 + eb_1\|^2 + \frac{c_1}{2} \|\xi^*\|^2 \\ s.t. \quad & -(Bw_1 + eb_1) + \xi^* = e \end{aligned} \tag{2.29}$$

and

$$\begin{aligned} \underset{w_2, b_2, \eta^*}{min} \quad & \frac{1}{2} \|Bw_2 + eb_2\|^2 + \frac{c_2}{2} \|\eta^*\|^2 \\ s.t. \quad & (Aw_2 + eb_2) + \eta^* = e. \end{aligned} \tag{2.30}$$

The linear LSTSVM solves the inverse of matrix of the order of $(n+1) \times (n+1)$. Once the equations (2.29) and (2.30) are solved, the optimal hyperplanes are obtained

as follows:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -[c_1 G^T G + H^T H]^{-1} c_1 G^T e, \tag{2.31}$$

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = [c_2 H^T H + G^T G]^{-1} c_2 H^T e. \tag{2.32}$$

2.5.3 Robust energy based least squares twin support vector machines

The primal formulation of robust energy based least squares twin support vector machines (RELSTSVM) [260] is given as follows:

$$\begin{aligned} \underset{w_1,b_1,\xi^*}{min} \quad & \frac{1}{2}\left\|Aw_1 + eb_1\right\|^2 + \frac{c_1}{2}\left\|\xi^*\right\|^2 + \frac{c_3}{2}\left\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\right\|^2 \\ s.t. \quad & -(Bw_1 + eb_1) + \xi^* = E_1 \end{aligned} \tag{2.33}$$

and

$$\begin{aligned} \underset{w_2,b_2,\eta^*}{min} \quad & \frac{1}{2}\left\|Bw_2 + eb_2\right\|^2 + \frac{c_2}{2}\left\|\eta^*\right\|^2 + \frac{c_4}{2}\left\|\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}\right\|^2 \\ s.t. \quad & (Aw_2 + eb_2) + \eta^* = E_2, \end{aligned} \tag{2.34}$$

where E_1 and E_2 are energy parameters of the hyperplanes.

On substituting the equality constraints into the objective function, QPP (2.33) becomes:

$$L_1 = \frac{1}{2}\left\|Aw_1 + eb_1\right\|^2 + \frac{c_1}{2}\left\|Bw_1 + eb_1 + E_1\right\|^2 + \frac{c_3}{2}\left\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\right\|^2. \tag{2.35}$$

Setting the gradient of (2.35) with respect to w_1 and b_1 to zero gives the solution

of QPP (2.33) as follows:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -[c_1 G^T G + H^T H]^{-1} c_1 G^T E_1. \tag{2.36}$$

In the similar way, the solution of QPP (2.34) is given as follows:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = [c_2 H^T H + G^T G]^{-1} c_2 H^T E_2. \tag{2.37}$$

2.5.4 Twin k-class support vector classification

Twin k-class support vector classification (TWKSVC) [294] algorithm evaluates the training points into "1-versus-1-versus-rest" with the two kind of samples selected from k-classes as the focused partitions. Let l_1, l_2 be the number of samples in two focused class samples and l_3 be the samples in rest class, here $M = l_1 + l_2 + l_3$. The remaining samples are mapped into a region between the non-parallel hyperplanes.

The optimization problem of TWKSVC is given as:

$$\begin{aligned} \underset{w_1, b_1, \xi, \xi^*}{min} \quad & \frac{1}{2} \|Aw_1 + e_1 b_1\|^2 + c_1 e_2^T \xi^* + c_2 e_3^T \xi \\ s.t. \quad & -(Bw_1 + e_2 b_1) + \xi^* \geq e_2, \\ & -(Cw_1 + e_3 b_1) + \xi \geq e_3(1 - \epsilon), \\ & \xi^* \geq 0, \xi \geq 0 \end{aligned} \tag{2.38}$$

and

$$\begin{aligned} \underset{w_2, b_2, \eta, \eta^*}{min} \quad & \frac{1}{2} \|Bw_2 + e_2 b_2\|^2 + c_3 e_1^T \eta^* + c_4 e_3^T \eta \\ s.t. \quad & (Aw_2 + e_2 b_2) + \eta^* \geq e_2, \\ & (Cw_2 + e_3 b_2) + \eta \geq e_3(1 - \epsilon), \\ & \eta^* \geq 0, \eta \geq 0, \end{aligned} \tag{2.39}$$

where $w_1 \in R^{n\times 1}, w_2 \in R^{n\times 1}, b_1 \in R, b_2 \in R, \xi^* \in R^{l_2\times 1}, \xi \in R^{l_3\times 1}, \eta^* \in R^{l_1\times 1}, \eta \in R^{l_3\times 1}, e_1 \in R^{l_1\times 1}, e_2 \in R^{l_2\times 1}$, and $e_3 \in R^{l_3\times 1}$.

The dual of QPP (2.38) is given as follows:

$$\begin{aligned} \underset{\gamma}{max} \quad & -\frac{1}{2}\gamma^T N(H^T H)^{-1}N^T\gamma + e_4^T\gamma \\ s.t. \quad & 0 \leq \gamma \leq F, \end{aligned} \tag{2.40}$$

where $S = [C \quad e_3], e_4 = [e_2; e_3(1-\epsilon)], N = [G; S], \gamma = [\alpha; \beta]$ and $F = [c_1 e_2; c_2 e_3]$.

Similarly, the dual of QPP (2.39) is obtained as follows:

$$\begin{aligned} \underset{\rho}{max} \quad & -\frac{1}{2}\rho^T P(G^T G)^{-1}P^T\rho + e_5^T\rho \\ s.t. \quad & 0 \leq \rho \leq F^*, \end{aligned} \tag{2.41}$$

where $P = [H; S], F^* = [c_3 e_1; c_4 e_3], e_5 = [e_1; e_3(1-\epsilon)]$.

Once the QPP (2.40) is solved, the optimal hyperplane is given as:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(H^T H)^{-1}(N^T\gamma). \tag{2.42}$$

Similarly, as the QPP (2.41) is solved, the optimal hyperplane is given as:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (G^T G)^{-1}(P^T\rho). \tag{2.43}$$

2.5.5 Least squares twin k-class support vector classification

The optimization problem of least squares TWKSVC [191] is given as follows:

$$\begin{aligned} \underset{w_1,b_1,\xi,\xi^*}{min} \quad & \frac{1}{2}\left\|Aw_1 + e_1 b_1\right\|^2 + \frac{c_1}{2}\xi^{*T}\xi^* + \frac{c_2}{2}\xi^T\xi \\ s.t. \quad & -(Bw_1 + e_2 b_1) + \xi^* = e_2, \\ & -(Cw_1 + e_3 b_1) + \xi = e_3(1-\epsilon) \end{aligned} \tag{2.44}$$

and

$$\begin{aligned} \underset{w_2,b_2,\eta,\eta^*}{min} \quad & \frac{1}{2}\left\|Bw_2+e_2b_2\right\|^2+\frac{c_3}{2}\eta^{*T}\eta^*+\frac{c_4}{2}\eta^T\eta \\ s.t. \quad & (Aw_2+e_1b_2)+\eta^*=e_1, \\ & (Cw_2+e_3b_2)+\eta=e_3(1-\epsilon). \end{aligned} \tag{2.45}$$

On substituting the equality constraints into the objective function, QPP (2.44) becomes:

$$\underset{w_1,b_1}{min} \quad \frac{1}{2}\left\|Aw_1+e_1b_1\right\|^2+\frac{c_1}{2}\left\|Bw_1+e_2b_1+e_2\right\|^2+\frac{c_2}{2}\left\|Cw_1+e_3b_1+e_3(1-\epsilon)\right\|^2. \tag{2.46}$$

Setting the gradient of (2.46) with respect to w_1 and b_1 to zero gives the dual of QPP (2.44) as follows:

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = -(c_1G^TG+H^TH+c_2P^TP)^{-1}(c_1G^Te_5+c_2P^Te_6(1-\epsilon)), \tag{2.47}$$

where $H=[A \quad e_1], G=[B \quad e_2]$ and $P=[C \quad e_3]$.

Similarly, the solution of QPP (2.45) is given as follows:

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (c_3H^TH+G^TG+c_4P^TP)^{-1}(c_3H^Te_4+c_4P^Te_6(1-\epsilon)). \tag{2.48}$$

2.6 Statistical tests

In this section, we discuss the statistical tests used to evaluate the performance of the models statistically. We follow the following tests:

2.6.1 Friedman test

Friedman test has been proven to be more robust than other approaches and has been practiced by numerous researchers [53, 145] to check the statistical significance among classifiers. 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 k algorithms and N datasets. The Friedman test compares the average rank of algorithms, $R_j = \frac{1}{N}\sum_i r_i^j$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their rank R_j should be equal. If k is the number of algorithms and N is number of datasets then Friedman statistic

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right], \tag{2.49}$$

is distributed according to χ_F^2 with $(k-1)$ degrees of freedom, when N and k are big enough. In that case, Friedman statistic χ_F^2 is undesirably conservative and derived a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{2.50}$$

which is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom. If the null-hypothesis is rejected, Nemenyi test [192] can be used to check whether the performance of two among k classifiers are significantly different. Two classifiers are said to be statistically different if the rank of two classifiers differ by at least critical difference (CD). Mathematically, critical difference is given by

$$\text{CD} = q_\alpha\sqrt{\frac{k(k+1)}{6N}}, \tag{2.51}$$

where critical value q_α is based on the studentized range statistic divided by $\sqrt{2}$.

2.6.2 Win-tie-loss: sign test

To access the overall performances of classifiers, we count the number of datasets on which algorithm is the overall winner. We use sign test for the pairwise comparison of algorithms. In this test, under null-hypothesis two algorithms are equivalent if each wins on approximately $N/2$ out of N datasets. If the number of wins is at least $N/2 + 1.96\sqrt{N}/2$, the algorithm is significantly better with $p < 0.05$. If the two algorithms end with tie, then the number should be evenly splitted between the classifiers. However, if the number is odd we ignore one.

Chapter 3

Oblique decision tree ensemble via twin bounded SVM

Ensemble methods with "perturb and combine" strategy have shown improved performance in the classification problems. Recently, random forest algorithm was ranked one among 179 classifiers evaluated on 121 UCI datasets. Motivated by this, we present a new approach for the generation of oblique decision trees. At each non-leaf node, the training data samples are grouped in two categories based on the Bhattachrayya distance with randomly selected feature subset. Then, twin bounded support vector machine (TBSVM) is used to get two clustering hyperplanes such that each hyperplane is closer to data points of one group and as far as possible from the data points of other group. Based on these hyperplanes, each non-leaf node is splitted to generate the decision tree. We used different base models like random forest (RaF), rotation forest (RoF), random sub rotation forest (RRoF) to generate the different oblique decision tree forests named as twin bounded random forest (TBRaF), twin bounded rotation forest (TBRoF) and twin bounded random sub rotation forest (TBRRoF), respectively. In earlier oblique decision trees, like multisurface proximal support vector machine (MPSVM) based oblique decision trees, matrices are positive semi-definite and hence different regularization methods are required. However, no explicit regularization techniques need to be applied to the primal problems as the matrices in the proposed TBRaF, TBRoF and TBRRoF are positive definite. We evaluated the performance of the proposed models (TBRaF, TBRoF and TBRRoF) on 49 datasets

taken from the UCI repository and on some real-world biological datasets (not in UCI). The experimental results and statistical tests conducted show that TBRaF and TBRRoF outperform other baseline methods.

In the oblique decision trees where SVM based models are used for generating the separating hyperplanes, we need to handle the multiclass problem as SVM is basically a binary classification model. In literature, multiple approaches have been followed like one versus one, one versus all [95, 106, 141, 247], one-versus-one-versus-rest [294], DAG [214], ECOC [58], majority class versus rest class [173] and so on. However, these approaches being successfull lead to additional complexity as each method generates different number of hyperplanes like one-versus-one approach generates $\frac{k(k-1)}{2}$ hyperplanes for a $k-$class problem. In order to avoid any additional computational cost, we followed the same approach as in [303] to convert the multiclass problem into binary problem with a Bhattachrayya distance [13]. Bhattacharyya distance measures the similarity between the two discrete or continuous probability distributions, which is considered a good metric to measure the separability between two normal classes w_j and w_k with distributions $N(\mu_j, \sum_j)$ and $N(\mu_k, \sum_k)$. Here, multivariate Gaussian distribution is taken due to reasons given in [124]. First, it is the most natural distribution and the sum of a large number of independent random distributions follow Gaussian distribution. It has the maximum uncertainty of all distributions having a given mean and variance. Also, it is appropriate in many situations.

In MPSVM based oblique decision trees [303], the matrices that appear in MPSVM may or may not be invertible. Hence, in order to get solution, MPSVM assumes that either the inverse of the matrices exist or the matrices are non-singular. These prerequisites may or may not be satisfied. Hence, the dual problems are technically modified to handle the inverse matrices problem in MPSVM. In this chapter, we propose TBSVM based oblique decision trees. TBSVM has following advantages :

- TBSVM minimizes the structural risk by incorporating the regularization.
- No extra assumption or modification of the original dual problems is required.
- Matrices that appear in dual problems are positive definite.

This chapter presents TBRaF, TBRoF and TBRRoF ensemble classifiers for the multiclass classification problems. For each class, a decision hyperplane is learned based on the randomly selected feature set. The proposed model uses TBSVM based hyperplanes for splitting the data at each non-leaf node. Hyperplanes generated by TBSVM provide better classification down the hierarchy in the proposed TBRaF, TBRoF and TBRRoF ensemble classifiers. The proposed TBRaF, TBRoF and TBRRoF lead to the following improvements:

- The matrices in MPSVM based oblique decision tree ensemble are positive semi-definite as the generalized eigenvalue problems in MPSVM based decision tree may become singular. Thus, require a regularization technique to render their solutions. However, the matrices that appear in the dual formulation of the proposed TBRaF, TBRoF and TBRRoF are positive definite and thus primal problems in TBRaF, TBRoF and TBRRoF can be derived without any extra assumption and need not be modified any more.

- The significant advantage of the proposed TBRaF, TBRoF and TBRRoF is that the structural risk minimization principle is implemented by introducing the regularization term. This embodies the marrow of statistical learning theory which improves the generalization performance.

- At each node, the samples are grouped based on the Bhattacharyya distance. Based on the groups generated, splitting hyperplanes are generated via TBSVM.

- Unlike using univariate features in RaF, the proposed TBRaF, TBRoF and TBRRoF use multivariate features for the decision making at each non-leaf node. This provide better decision hyperplane.

- Numerical experiments on multiple datasets from the UCI repository and some real-world biological (not in UCI) datasets show the efficacy of the proposed TBRaF, TBRoF and TBRRoF models.

3.1 Oblique decision tree ensemble via twin bounded SVM

In order to use TBSVM based hyperplanes as a test criterion in the decision trees, following issue needs to be resolved.

- As TBSVM was originally designed for the binary classification, so we need to handle the scenarios of multiclass problem.

The approach we have followed to transform multiclass to binary class problem is given in Algorithm 3.1.

Algorithm 3.1 Multiclass to Binary class.

Input:
$X = M \times n$ is the dataset with M samples each with feature length n.
$Y = M \times 1$ are data labels corresponding to the training data set.
$\{L_1, \ldots, L_k\}$ is the set of data labels.
Output:
G_+ and G_- are the two hyperclasses or groups.

- For $j = 1, \ldots, k$
  - Calculate the Bhattacharyya distance between every pair of classes L_j and L_i, with $i = j + 1, \ldots, k$ as:

$$B(L_j, L_i) = \frac{1}{8}(\mu_i - \mu_j)^T \left(\frac{\sum_j + \sum_i}{2} \right)^{-1} (\mu_i - \mu_j) + \frac{1}{2} ln \frac{|(\sum_j + \sum_i)/2|}{\sqrt{|\sum_j||\sum_i|}}. \tag{3.1}$$

- Let L_+ and L_- be the pair of classes with largest Bhattacharyya distance, put them in groups G_+ and G_- respectively.
- For every other class, if $B(L_i, L_+) < B(L_i, L_-)$ then assign the L_i to the group G_+ else put in group G_-.

First we describe the construction of the proposed TBRaF, TBRoF and TBRRoF models, and then we will discuss about the motivation behind these ensemble models.

In the proposed TBRaF ensemble model, we use RaF as the base model. In this model, multiple decision trees are generated based on the different bootstrapped versions of the training data. In each decision tree, splitting hyperplane is generated at each non-leaf node. We use TBSVM for generating the hyperplanes. The splitting hyperplane depends on the randomly selected subset of features of the training

data samples. Each tree takes the decision independently based on the different bootstrapped version of the training data. During testing, each test sample is pushed down in each tree of the forest, and based on the majority voting final label is assigned to the test sample. The detailed algorithm of the proposed TBRaF model is given in Algorithm 3.2.

Algorithm 3.2 Proposed TBRaF.

Training Phase:

Given:

$X = M \times n$ is the dataset with M samples each with feature length n.

$Y = M \times 1$ are data labels corresponding to the training dataset.

L is the ensemble size i.e number of trees in the forest.

Each tree in the random forest is represented as T_i, where $i = 1, \ldots, L$.

"$mtry$" refers to the number of randomly selected features.

"$minleaf$" is the maximum number of samples in an impure node.

c_1, c_2, c_3, c_4 are the positive parameters, e is vector of ones, α and γ are the Lagrange multipliers.

[1] Each tree T_i is build using the bootstrapped versions of the training data P with replacement.

[2] At each non-leaf node,

(a) Based on the Bhattacharyya distance, group the samples into two groups (G_+ and G_-) using the Multiclass to Binary class algorithm given in Algorithm 3.1.

(b) The node samples are splitted based on the "$mtry$" randomly selected features from the training data by solving the following optimization problems with $H = [G_+, e]$ and $G = [G_-, e]$

$$\begin{aligned} \max_{\alpha} \quad & e^T\alpha - \frac{1}{2}\alpha^T G(H^T H + c_3 I)^{-1} G^T \alpha \\ s.t. \quad & 0 \leq \alpha \leq c_1 \end{aligned} \tag{3.2}$$

and

$$\begin{aligned} \max_{\gamma} \quad & e_1^T\gamma - \frac{1}{2}\gamma^T H(G^T G + c_4 I)^{-1} H^T \gamma \\ s.t. \quad & 0 \leq \gamma \leq c_2. \end{aligned} \tag{3.3}$$

[3] Repeatedly execute step 2 until one of the conditions is met:

- Node becomes pure.
- Node contains number of samples less than or equal to $minleaf$.

Classification Phase:

For classification of a test sample, it is pushed down each tree in the forest, and each tree in the forest assigns the vote to the given sample data. Then the predicted label for the sample data is the one with the highest number of votes among the forest.

In the proposed TBRoF ensemble model, we use RoF as the base model. In this model, multiple decision trees are generated based on the rotated feature space of the training data. In each decision tree, splitting hyperplane is generated at each non-leaf node. We use TBSVM for generating the hyperplanes. Each tree uses a different rotated feature space matrix for generating the tree. The decision hyperplane generated in each tree depends on all the features of the rotated feature space matrix being used for that tree. The proposed TBRoF algorithm is given in Algorithm 3.3.

In the proposed TBRRoF ensemble model, we use RRoF as the base model. In this model, decision trees are generated based on the randomly selected subset of features taken from rotated feature space of the training data. In each decision tree, splitting hyperplane is generated at each non-leaf node. We use TBSVM for generating the hyperplanes. Each tree uses a different rotated feature subspace matrix for generating the tree. The decision hyperplane generated in each tree depends on randomly selected subset of features of the rotated feature space matrix being used for that tree. The proposed TBRRoF models is given in Algorithm 3.3. Note that TBRoF and TBRRoF models differ in step 2 of Algorithm 3.3 wherein the TBRoF is trained on the entire feature space while as the TBRRoF model is generated based on the random subspace of the training data depending on the "mtry" parameter.

Zhang and Suganthan [303] generated the hyperplane using MPSVM. In MPSVM, the matrices of generalized eigen value problem are not always positive definite. Hence, the authors used different regularization techniques like Tikhonov regularization and axis-parallel split regularization to tackle this issue. However, in TBSVM the regularization term is added in the primal problem, thus the matrices that evolve in dual formulation are always positive definite. Thus, there is no need to handle the regularization problem explicitly.

Unlike base models (RaF, RoF and RRoF), the proposed TBRaF, TBRoF and TBRRoF apply TBSVM for generating the hyperplanes at each non-leaf node of the decision trees. At each non-leaf node of the ensemble model, more than one feature is used for generating the decision hyperplanes. The proposed TBRaF, TBRoF and TBRRoF models provides the following advantages:

- The proposed models handle multi-class problem naturally. There is no need to use a series of binary classification problems (like one-vs-one, one-vs-all, one-vs-one-vs-rest) as in other classifier models such as support vector machine.

- The decision tree construction process can be distributed among several workstations or cores, hence ideal for parallelization.

- Recently in literature [62], decision trees (RaF) emerged as the rank one classifier among 179 classifiers evaluated on 121 datasets. Thus, ensemble classifiers with RaF as one of the base models provide better performance.

- Structural risk minimization principle is implemented by introducing the regularization term. This embodies the marrow of statistical learning theory, so this modification improved the performance of classification.

3.2 Experiments

Here, we analyze the computational complexity of the given models and evaluate the effect of different algorithmic parameters, compare the proposed ensemble models with other baseline models and finally evaluate the statistical significance among the models.

3.2.1 Experimental setup

In all the experiments, we used Intel Xeon processor MATLAB R2010b on Windows10 with system configuration Intel Xeon processor (2×18cores, 2.3 GHZ CPU) with 128 GB of RAM. In the experimental work, we evaluate the performances of the proposed TBRaF, TBRoF and TBRRoF with other standard models and MPSVM based ensemble models. The different ensemble methods evaluated are MPSVM based RaFs (MPRaF-T, P, N), MPSVM based RoFs (MPRoF-T, P, N), MPSVM based RRoFs (MPRRoF-T, P, N), TBSVM based RaF (TBRaF), TBSVM based

Algorithm 3.3 Proposed TBRRoF and TBRoF.

Training Phase:

Given:

$X = M \times n$ is the dataset with M samples each with feature length n.

$Y = M \times 1$ are class labels corresponding to the training data set.

L is the ensemble size i.e. number of trees in the forest.

F is the feature set.

S is the number of subsets of feature set F and $\{C_1, \ldots, C_k\}$ is the set of class labels.

"mtry" refers to the number of randomly selected features.

For $i = 1, \ldots, L$

[1] Prepare the Rotation Matrix R_i^a :

- Split F into S subsets: $F_{i,j}$ with $j = 1, \ldots, S$.
- For $j = 1, \ldots, S$
  - Let $X_{i,j}$ be the data set X for the features in $F_{i,j}$.
  - Eliminate a random subset of classes from $X_{i,j}$.
  - Select a bootstrap sample from $X_{i,j}$ of size 75% of the number of samples in $X_{i,j}$. Denote the new set as $X'_{i,j}$
  - Apply Principle Component Analysis (PCA) on $X'_{i,j}$ to get the coefficients in a matrix $C_{i,j}$.
- Arrange the $C_{i,j}$, for $j = 1, \ldots, S$ in a rotation matrix R_i as in equation (3.4).
- For construction of R_i^a, rearrange the columns of R_i to match order of the features in F.

[2] *For TBRRoF method:* The node samples are splitted based on the "mtry" randomly selected features from the training data (XR_i^a, Q).
For TBRoF method: The node samples are splitted based on the training data (XR_i^a, Y) as the training data.
At each non-leaf node of the decision tree T_i with a given training data,

(a) Based on the Bhattacharyya distance, group the samples into two groups (G_+ and G_-) using the Multiclass to Binary class algorithm given in Algorithm 3.1.

(b) Solve the optimization problems (3.2) and (3.3) with $H = [G_+, e]$ and $G = [G_-, e]$.

[3] Repeatedly execute step 2 until one of the conditions is met:

- Node becomes pure.
- Node contains number of samples less than or equal to $minleaf$.

Classification Phase:

- For a given sample x, let each individual decision tree T_i assigned the probability $d_{i,j}(xR_i^a)$ to the hypothesis that x belongs to class C_j. For each class C_j, calculate the confidence by the average combination method:

$$\mu_j(x) = \frac{1}{L} \sum_i d_{i,j}(xR_i^a), j = 1, \ldots, k. \tag{3.4}$$

- Assign x to the class with the largest confidence.

RoF (TBRoF) and TBSVM based RRoF (TBRRoF). For all the ensemble methods, we used CART [24] as the base classifier. We have taken the codes of random forest from "https://github.com/P-N-Suganthan/CODES" and TBSVM from "http://www.optimal-group.org/Resource/TWSVM.html".
Gini impurity criteria is defined as

$$\text{Gini}(t) = \frac{n_t^l}{n_t}\left[1 - \sum_{i=1}^{c}\left(\frac{n_{w_i}^l}{n_t^l}\right)^2\right] + \frac{n_t^r}{n_t}\left[1 - \sum_{i=1}^{c}\left(\frac{n_{w_i}^r}{n_t^l}\right)^2\right], \tag{3.5}$$

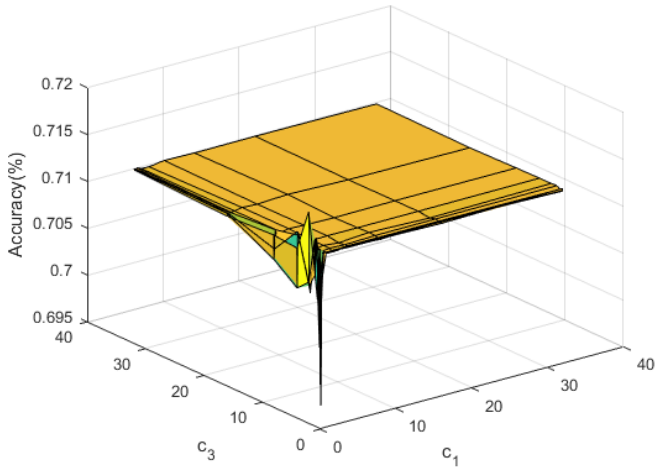
where c is the number of classes, n_t represents the number of data points at this node, n_t^l, n_t^r are the number of data points reaching to the left and right of the current node, respectively. $n_{w_i}^l$, $n_{w_i}^r$ are the number of data points belonging to class w_i that reached to the left node and right node of the current node, respectively.

For evaluating the performances of these ensemble methods, we conducted experiments on the benchmark datasets from the UCI repository and some real world datasets (not in UCI repository) [60, 62] . The four real world datasets not included in the UCI repository are about fecundity estimation [81] for fisheries: they are given as oocMer14D (2-class classification according to the presence/absence of oocyte nucleus), oocMer12F (3-class classification according to the stage of development of the oocyte) for fish species *Merluccius*, and oocTris2F (nucleus) and oocTris5B (stages) for fish species *Trisopterus*. The different parameters corresponding to different models are as follows

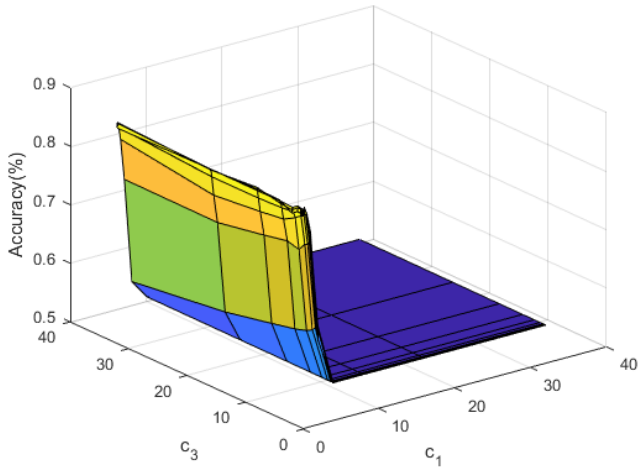
- *minleaf* is the parameter in the decision trees that controls the maximum number of data samples within an impure node. This value is usually set to 1 (as default).

- L represents the ensemble size in all the ensemble methods. With the increase in the value of L the computational time increases. Hence, to keep the computational time low we conducted all the experiments with $L = 50$.

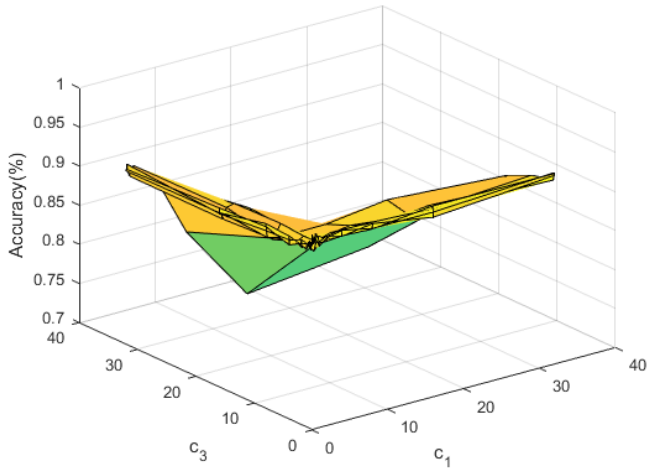
- *mtry* is another parameter in RaF and RRoF used to control randomness of the algorithm. *mtry* is the number of features used for splitting at each node. *mtry* is set to $round(\sqrt{n})$, where n represents the dimensions of the data samples.



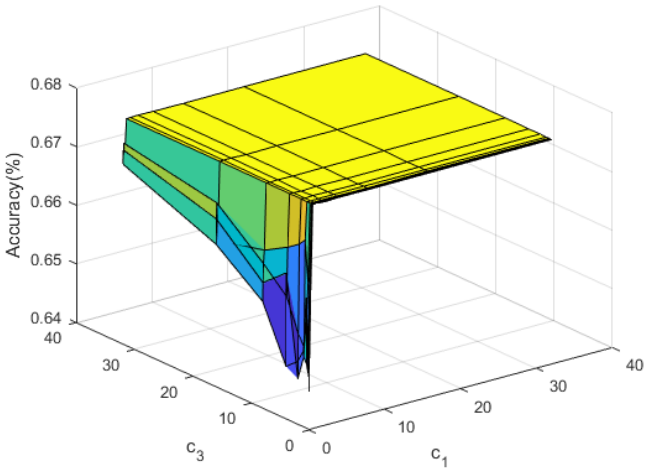
(a) Ilpd-indian-liver.



(b) Musk-1.



(c) Seeds.



(d) Statlog-australian-credit.

Figure 3.1: Influence of the parameters c_1 and c_3.

- For RoF, the number of subset features is set to three [224]. If n is not divisible by 3, randomly selected 1 or 2, as necessary, features from other features are used to complete the final feature subset. For all the experiments with regularization method as Tikhonov, we used $\delta = 0.01$.

- For TBRaF, TBRoF and TBRRoF, we performed grid search over the parameters of c_1 and c_3 with the parameters varying from $\{2^{-5}, \ldots, 2^5\}$. To reduce the computational complexity, we set the parameters as $c_1 = c_2$ and $c_3 = c_4$.

3.2.2 Computational complexity analysis

Consider X as training dataset with dimensions $M \times n$ where M is the number of samples and n is the number of features of each sample. Without any assumption about the structure of the decision tree as computational time is highly variable, we focus on partitioning the samples at a given node. In axis parallel split, each feature is split based on the impurity criterion. The computational complexity of finding such best split nodes is of the order of $nM \log M$ ($M \log M$ for each feature). For oblique decision trees based on MPSVM, complexity of generalized problem is of the order of n^3 [172]. For the oblique decision tree based on TBSVM, complexity of the generalized problem is of the order of $2(M/2)^3$. The training times corresponding to each model is given in the Table 3.1, 3.2 and 3.3 for TBRaF, TBRoF and TBRRoF, respectively.

3.2.3 Influence of the parameters c_1, c_2, c_3 and c_4

Figure (3.1(a)) to Figure (3.1(d)) show the sensitivity of the proposed TBRaF with respect to the parameters c_1, c_2, c_3, c_4 with $c_1 = c_2, c_3 = c_4$. Figure (3.1(a)) shows that the proposed TBRaF achieves maximum accuracy near lower values of c_1 and c_3. However, this model is insensitive to higher values of parameters c_1 and c_3. In Figure (3.1(b)), the model is insensitive to higher values of c_1. However, the model shows better performance at higher values of c_3. Figure (3.1(c)) shows that the performance of the proposed TBRaF model decreases if both values of c_1 and c_3 are higher. However, the model shows better performance if the value of c_1 is lower and the value of c_3 is higher or vice-versa. Similarly, Figure (3.1(d)) indicates that the performance is better for higher values of c_1 as compared to the lower values of c_1.

From the figures, it is clear that the choice of these parameters is a subjective matter. Thus, one must tune in these parameters to get the optimal accuracy.

3.2.4 Does TBSVM improve the decision tree ensemble?

In this subsection, we compare the classification accuracy of the proposed TBRaF, TBRoF and TBRRoF models with the existing standard RaF, RoF and RRoF models.

The results obtained on multiple datasets are presented in Tables 3.1, 3.2 and 3.3.

From Table 3.1, it is clear that the proposed TBRaF outperforms or achieves better performance as compared to the other baseline methods. The average accuracy of TBRaF is 84% while as among the baseline models MPRaF-P achieved the maximum accuracy of 83.49%. Also, the average rank of the proposed TBRaF model is lowest with 2.1122. In terms of overall win-tie-loss comparison, the proposed TBRaF has emerged as overall winner in 21 datasets while as other baseline models show win at less than or equal to 9 datasets.

From Table 3.2, one can see that the TBRoF does not show much improvement w.r.t. standard RoF and MPRoF-P. However, the proposed TBRoF achieves better performance than MPRoF-T and MPRoF-N. The average rank and overall win of the proposed TBRoF is second to RoF. In terms of overall-win-tie-loss, the proposed TBRoF is second to RoF with the number of wins 21 and 18 for RoF and TBRoF, respectively.

Similarly in Table 3.3, the proposed TBRRoF achieves better average accuracy and lower rank as compared to other baseline models. The proposed TBRRoF has emerged as the overall winner in 21 datasets while as other models show overall win in 9 or less than 9 datasets.

From the above discussion, we can conclude that the proposed TBRaF and TBRRoF with different base classifiers have improved the standard models in terms of average accuracy, average rank and the overall win-tie-loss performance.

Table 3.1: Classification accuracy[1] of RaF, MPRaF-T, MPRaF-P, MPRaF-N and proposed TBRaF.

Datasets	**RaF** Accuracy±Std Time(s)	**MPRaF-T** Accuracy±Std Time(s)	**MPRaF-P** Accuracy±Std Time(s)	**MPRaF-N** Accuracy±Std Time(s)	**Proposed TBRaF** Accuracy±Std Time(s), c_1, c_3
balance-scale	0.8531±0.021	0.8893±0.02	0.888±0.02	0.8925±0.022	**0.8979**±0.018
	1.53602	2.0714	2.82122	2.08447	**5.65959**, 0.25, 0.0625
balloons	0.5625±0.212	0.5125±0.189	0.55±0.237	**0.575**±0.183	0.5375±0.233
	0.0419774	0.0550788	0.0676103	0.0611679	**0.140481**, 0.25, 16
breast-cancer-wisc-diag	0.9525±0.014	**0.9722**±0.014	0.967±0.011	0.9708±0.013	0.9687±0.015
	5.64472	0.519897	2.37919	0.666441	**5.34323**, 0.25, 0.25
breast-cancer-wisc-prog	0.7741±0.052	0.7862±0.055	0.7841±0.063	0.7912±0.053	**0.8023**±0.054
	2.79874	0.594735	1.96744	0.709238	**1.54653**, 0.0625, 0.5
breast-cancer-wisc	0.966±0.01	0.9685±0.009	0.9651±0.009	**0.9691**±0.009	0.9674±0.009
	1.05521	0.629595	1.05363	0.718628	**3.78236**, 0.03125, 0.125
breast-cancer	0.7273±0.043	0.7398±0.05	0.7321±0.043	**0.7412**±0.045	0.7336±0.042
	1.13152	1.14917	1.67626	1.18845	**3.18361**, 0.03125, 0.0625
CTG-10clases	**0.8554**±0.014	0.8185±0.013	0.8491±0.014	0.7981±0.018	0.8253±0.016
	21.3329	14.3621	26.9488	16.4288	**71.4426**, 0.03125, 0.03125
CTG-3clases	**0.9371**±0.01	0.9166±0.009	0.9348±0.011	0.9056±0.009	0.9209±0.008
	13.0621	5.70305	14.2068	6.65877	**66.5011**, 0.03125, 0.03125

Continued on next page

[1] Average accuracy obtained in five times 4-fold cross validation

Table 3.1 – continued from previous page

Datasets	**RaF** Accuracy±Std Time(s)	**MPRaF-T** Accuracy±Std Time(s)	**MPRaF-P** Accuracy±Std Time(s)	**MPRaF-N** Accuracy±Std Time(s)	**Proposed TBRaF** Accuracy±Std Time(s), c_1, c_3
conn-bench-sonar-mines-rocks	0.7913±0.05	**0.8192**±0.057	0.7952±0.059	0.8096±0.057	0.8144±0.053
	3.56253	0.705381	2.77231	0.896123	**1.56348**, 0.03125, 4
cylinder-bands	**0.7746**±0.047	0.7422±0.051	0.7613±0.042	0.7215±0.047	0.7426±0.03
	3.56915	2.09743	4.8048	2.46726	**6.13782**, 0.125, 1
dermatology	0.9743±0.014	**0.9765**±0.011	0.9754±0.012	0.9738±0.013	0.9754±0.009
	1.18396	1.22938	2.25063	1.49392	**5.13847**, 0.5, 1
echocardiogram	0.8279±0.051	0.8431±0.05	0.8324±0.054	0.8336±0.057	**0.8444**±0.061
	0.571991	0.442826	0.780068	0.509125	**1.2295**, 0.5, 0.125
fertility	0.878±0.058	0.88±0.059	**0.884**±0.06	0.88±0.059	0.88±0.059
	0.233329	0.278645	0.389356	0.272389	**0.769963**, 0.03125, 1
haberman-survival	0.721±0.043	0.7138±0.045	0.7197±0.05	0.7256±0.049	**0.7405**±0.041
	1.18723	1.15411	1.50033	1.22006	**1.53548**, 0.0625, 32
heart-hungarian	0.8206±0.056	0.8193±0.054	0.8138±0.05	**0.8274**±0.044	0.824±0.044
	1.06171	0.909603	1.50164	0.978653	**2.66378**, 0.03125, 16
hepatitis	0.8195±0.045	0.8179±0.041	0.8209±0.032	**0.8286**±0.037	0.819±0.031
	0.582215	0.437568	0.782073	0.509688	**1.38997**, 0.0625, 4
ilpd-indian-liver	0.7117±0.032	**0.7176**±0.042	0.7087±0.037	0.7039±0.04	0.7156±0.028
	3.93123	2.24851	3.26439	2.19885	**4.52837**, 0.03125, 2
ionosphere	0.93±0.026	0.9277±0.029	**0.9453**±0.022	0.9237±0.029	0.9334±0.027
	4.68988	0.834947	3.25767	0.973408	**2.28928**, 0.03125, 0.25
iris	0.9412±0.031	0.9546±0.024	0.9506±0.034	**0.9642**±0.026	0.9547±0.024

Continued on next page

Table 3.1 – continued from previous page

Datasets	**RaF** Accuracy±Std Time(s)	**MPRaF-T** Accuracy±Std Time(s)	**MPRaF-P** Accuracy±Std Time(s)	**MPRaF-N** Accuracy±Std Time(s)	**Proposed TBRaF** Accuracy±Std Time(s), c_1, c_3
	0.260484	0.235475	0.280535	0.235618	**0.728634**, 0.03125, 0.0625
led-display	**0.7188**±0.021	0.7112±0.028	0.7146±0.025	0.711±0.026	0.7088±0.022
	1.65933	3.61142	4.34234	3.21508	**12.0478**, 0.03125, 0.125
libras	0.7672±0.033	0.8367±0.032	0.8128±0.034	0.8267±0.038	**0.8506**±0.031
	11.5398	4.59987	10.4651	5.31686	**10.117**, 0.25, 0.125
low-res-spect	0.8892±0.026	0.8968±0.026	0.8904±0.023	0.8825±0.026	**0.9024**±0.028
	12.9121	2.12658	13.88	2.87792	**13.5362**, 32, 0.03125
lymphography	0.8216±0.059	0.8297±0.06	0.827±0.068	0.8176±0.08	**0.8419**±0.077
	0.538933	0.69192	1.06478	0.750724	**1.71002**, 0.03125, 0.25
mammographic	0.8237±0.021	0.8237±0.022	0.8256±0.021	**0.8293**±0.018	0.8244±0.023
	2.68687	2.4751	3.57924	2.53127	**9.69897**, 0.0625, 0.0625
molec-biol-promoter	**0.8457**±0.085	0.7771±0.088	0.8266±0.07	0.7604±0.065	0.8078±0.066
	0.655743	0.408548	0.863222	0.49062	**0.711638**, 0.125, 0.5
musk-1	0.8685±0.027	0.8748±0.028	0.8634±0.029	**0.8786**±0.032	0.8761±0.025
	13.9999	1.99875	10.081	2.6516	**4.6471**, 0.0625, 32
oocytes-merluccius-nucleus-4d	0.7914±0.025	0.8346±0.021	0.8295±0.021	0.8233±0.015	**0.8389**±0.015
	31.7648	4.05487	13.44	4.56613	**12.5657**, 0.0625, 0.03125
oocytes-merluccius-states-2f	0.9225±0.015	0.9198±0.017	0.9235±0.015	0.9209±0.016	**0.9248**±0.016

Continued on next page

Table 3.1 – continued from previous page

Datasets	**RaF** Accuracy±Std Time(s)	**MPRaF-T** Accuracy±Std Time(s)	**MPRaF-P** Accuracy±Std Time(s)	**MPRaF-N** Accuracy±Std Time(s)	**Proposed TBRaF** Accuracy±Std Time(s), c_1, c_3
	19.6519	2.14772	8.01702	2.48453	**16.169**, 0.0625, 0.125
oocytes-trisopterus-nucleus-2f	0.7989±0.025	0.8243±0.022	0.8173±0.019	0.825±0.024	**0.832**±0.019
	23.325	3.35974	8.36373	3.83251	**10.8075**, 0.0625, 0.0625
oocytes-trisopterus-states-5b	0.9086±0.017	0.9235±0.013	0.9136±0.019	0.9169±0.014	**0.9279**±0.016
	25.0987	2.17577	15.7213	3.05768	**12.3638**, 0.0625, 0.03125
parkinsons	0.8903±0.038	**0.9202**±0.039	0.9017±0.037	0.9046±0.039	0.9108±0.048
	1.67672	0.47586	1.13345	0.53735	**1.29587**, 0.03125, 0.5
pima	0.7609±0.029	**0.7628**±0.035	0.757±0.029	0.7607±0.031	0.7576±0.028
	5.16697	2.72914	4.24306	2.96877	**8.50438**, 0.03125, 4
pittsburg-bridges-MATERIAL	0.8547±0.074	0.8565±0.064	0.8489±0.064	0.8659±0.057	**0.8696**±0.054
	0.274766	0.375906	0.532384	0.373841	**0.979578**, 0.125, 4
pittsburg-bridges-REL-L	0.7091±0.075	0.7197±0.055	0.7275±0.065	0.7195±0.056	**0.7369**±0.061
	0.429036	0.581914	0.872204	0.637329	**1.4768**, 0.03125, 4
pittsburg-bridges-SPAN	0.6522±0.085	0.6696±0.075	0.6652±0.063	0.6717±0.085	**0.6826**±0.091
	0.379814	0.534946	0.735547	0.571372	**1.32874**, 0.125, 8

Continued on next page

Table 3.1 – continued from previous page

Datasets	**RaF** Accuracy±Std Time(s)	**MPRaF-T** Accuracy±Std Time(s)	**MPRaF-P** Accuracy±Std Time(s)	**MPRaF-N** Accuracy±Std Time(s)	**Proposed TBRaF** Accuracy±Std Time(s), c_1, c_3
pittsburg-bridges-T-OR-D	0.8531±0.061	0.8573±0.059	0.855±0.064	**0.8653**±0.063	0.8611±0.07
	0.239393	0.264411	0.347137	0.240879	**0.715674**, 0.125, 0.03125
planning	0.695±0.058	0.7102±0.063	0.7146±0.069	0.7124±0.07	**0.7168**±0.061
	1.59859	0.745928	1.20486	0.847941	**2.22809**, 0.0625, 0.5
post-operative	0.6972±0.074	0.6926±0.067	0.6949±0.078	0.7038±0.075	**0.7106**±0.082
	0.296504	0.436419	0.623308	0.460275	**0.852061**, 0.25, 0.125
seeds	0.9192±0.034	0.9449±0.032	0.9286±0.037	0.9333±0.034	**0.9458**±0.035
	0.933655	0.39269	0.810089	0.434924	**1.82897**, 0.5, 0.03125
statlog-australian-credit	0.66±0.032	0.6464±0.045	0.6577±0.026	0.6643±0.031	**0.6782**±0.035
	4.944	3.15254	5.30835	3.44116	**3.22101**, 2, 2
statlog-german-credit	**0.7574**±0.024	0.7424±0.017	0.749±0.024	0.7364±0.03	0.7514±0.025
	5.49939	4.12323	7.33562	4.37016	15.1641, 0.125, 2
statlog-heart	0.8133±0.038	0.834±0.035	**0.84**±0.034	0.8369±0.033	0.8319±0.028
	1.33561	0.808507	1.5169	0.932894	**3.31014**, 0.25, 0.5
statlog-image	0.9729±0.005	0.9739±0.007	**0.9761**±0.006	0.9679±0.006	0.9697±0.007
	18.9464	6.79109	16.9491	7.43291	**70.1839**, 0.0625, 0.03125
statlog-vehicle	0.7357±0.026	0.7634±0.024	0.7579±0.034	0.7586±0.023	**0.7671**±0.025
	7.04925	3.92815	7.70733	4.47957	**12.5013**, 0.03125, 0.03125

Continued on next page

Table 3.1 – continued from previous page

Datasets	**RaF**	**MPRaF-T**	**MPRaF-P**	**MPRaF-N**	**Proposed TBRaF**
	Accuracy±Std	Accuracy±Std	Accuracy±Std	Accuracy±Std	Accuracy±Std
	Time(s)	Time(s)	Time(s)	Time(s)	Time(s), c_1, c_3
synthetic-control	0.9787±0.011	**0.992**±0.005	0.9863±0.007	0.9817±0.012	0.986±0.011
	11.6504	1.43298	9.54569	2.42161	**4.7429**, 0.03125, 1
tic-tac-toe	**0.9793**±0.011	0.977±0.013	0.9768±0.015	0.9572±0.02	0.976±0.016
	2.37135	2.81796	3.95174	2.98624	**9.75575**, 0.03125, 0.03125
vertebral-column-2clases	0.8263±0.043	0.8464±0.03	0.8458±0.035	0.8405±0.041	**0.8536**±0.033
	1.62642	0.967857	1.13434	0.912168	**2.42357**, 0.0625, 2
wine	0.9765±0.019	**0.9808**±0.016	0.9776±0.016	0.9763±0.015	0.9775±0.017
	0.921645	0.34843	0.736564	0.41319	**0.949454**, 0.03125, 0.125
zoo	0.9367±0.053	0.9387±0.041	0.9288±0.051	0.9348±0.046	**0.9465**±0.045
	0.234978	0.402426	0.546577	0.434458	**0.970172**, 0.0625, 0.0625
Average Accuracy	0.8294	0.8346	0.8349	0.8330	0.8401
Average Rank	3.7449	2.8265	3.2143	3.1020	2.1122
Overall Win-Tie-Loss	7-0-22	8-0-6	4-0-6	9-0-13	21-0-1

Here, CTG denotes cardiotocography.

Table 3.2: Classification accuracy[2] of RoF, MPRoF-T, MPRoF-P, MPRoF-N and proposed TBRoF.

Datasets	**RoF** Accuracy±Std Time(s)	**MPRoF-T** Accuracy±Std Time(s)	**MPRoF-P** Accuracy±Std Time(s)	**MPRoF-N** Accuracy±Std Time(s)	**Proposed TBRoF** Accuracy±Std Time(s), c_1, c_3
balance-scale	0.878±0.0219	0.8738±0.035	**0.8946**±0.0364	0.8873±0.0379	0.7946±0.1841
	6.34407	2.2999	6.15686	2.75258	**6.30545**, 0.0625, 2
balloons	0.65±0.1884	0.5±0.2294	0.5375±0.1862	0.5375±0.247	**0.6625**±0.2841
	0.143577	0.0878862	0.12783	0.0956771	**0.205192**, 8, 0.03125
breast-cancer-wisc-diag	0.9718±0.0125	0.9511±0.0206	0.9595±0.016	0.9437±0.0237	**0.9764**±0.0141
	84.4053	1.83243	50.3251	3.52055	**4.44207**, 0.0625, 8
breast-cancer-wisc-prog	0.8009±0.0462	0.6825±0.0714	0.7846±0.0457	0.729±0.0408	**0.8039**±0.0558
	32.0557	1.73	30.4958	3.2382	**3.07957**, 16, 1
breast-cancer-wisc	**0.9662**±0.0106	0.9453±0.0129	0.9588±0.0109	0.9593±0.0118	0.9599±0.0113
	11.217	0.83596	8.32655	1.23002	**7.8142**, 1, 1
breast-cancer	0.6993±0.0425	0.6672±0.0427	0.6866±0.051	0.6966±0.0506	**0.7246**±0.0662
	6.54187	1.34891	6.81218	1.7516	**4.96631**, 0.5, 0.5
CTG-10clases	**0.8666**±0.0115	0.716±0.0234	0.8626±0.0131	0.7811±0.0156	0.7786±0.0128
	554.614	39.5324	615.48	76.0299	**151.284**, 0.125, 0.125
CTG-3clases	**0.9404**±0.0091	0.8849±0.0141	0.9398±0.0098	0.8769±0.0144	0.8958±0.017
	492.875	13.914	520.895	22.9332	**77.2051**, 0.0625, 0.125
conn-bench-sonar-mines-rocks	**0.8307**±0.0452	0.7278±0.0519	0.7817±0.0596	0.748±0.0793	0.7875±0.046
	73.6051	3.40861	60.825	6.01374	**4.71071**, 1, 32

Continued on next page

[2]Average accuracy obtained in five times 4-fold cross validation

Table 3.2 – continued from previous page

Datasets	**RoF**	**MPRoF-T**	**MPRoF-P**	**MPRoF-N**	**Proposed TBRoF**
	Accuracy±Std	Accuracy±Std	Accuracy±Std	Accuracy±Std	Accuracy±Std
	Time(s)	Time(s)	Time(s)	Time(s)	Time(s), c_1, c_3
cylinder-bands	**0.7972**±0.0329	0.6425±0.0388	0.7953±0.0373	0.7273±0.0395	0.7031±0.0407
	90.6537	6.04663	101.946	9.012	**10.5082**, 0.5, 2
dermatology	0.9694±0.0182	0.9579±0.0155	0.9727±0.0155	0.959±0.0222	**0.9743**±0.0101
	13.4256	2.43307	16.3661	3.87778	**4.58952**, 0.0625, 4
echocardiogram	0.8174±0.0579	0.761±0.0683	0.8249±0.0615	**0.8511**±0.0592	0.8218±0.0708
	6.00092	0.654299	7.3419	0.8314	**2.11714**, 2, 8
fertility	0.874±0.0584	0.804±0.084	0.862±0.0615	0.83±0.0732	**0.88**±0.0467
	2.27835	0.450268	2.65816	0.560399	**0.474786**, 1, 32
haberman-survival	0.6993±0.0412	0.6922±0.0477	0.6875±0.0568	0.6795±0.0587	**0.7339**±0.0461
	7.86895	1.92986	3.50187	1.99036	**8.93732**, 0.5, 1
heart-hungarian	**0.8034**±0.0402	0.7675±0.0501	0.798±0.0365	0.8033±0.0438	0.802±0.0376
	14.7009	1.46215	12.1157	1.84952	**3.82945**, 1, 1
hepatitis	0.8025±0.0517	0.7709±0.0661	0.7983±0.0583	**0.8383**±0.0418	0.81±0.0504
	8.75994	1.01502	10.0063	1.25461	**3.05519**, 0.5, 16
ilpd-indian-liver	0.7009±0.0327	0.6619±0.038	0.6899±0.0379	0.67±0.0437	**0.7156**±0.0271
	41.7445	3.66013	16.5748	3.89046	**6.54913**, 0.5, 1
ionosphere	0.943±0.0255	0.8752±0.0351	**0.9436**±0.0237	0.886±0.0288	0.9031±0.0302
	77.4499	2.04324	82.3376	3.24659	**9.77862**, 0.5, 32
iris	0.9452±0.0417	0.968±0.0238	0.9706±0.0192	0.9733±0.0196	**0.9771**±0.0236
	1.44854	0.314701	0.769036	0.368306	**1.20041**, 0.5, 0.5
led-display	0.7106±0.0291	0.711±0.0299	**0.7134**±0.029	0.709±0.0321	0.709±0.0295
	6.40203	6.66245	11.4223	7.20944	**18.4051**, 0.03125, 2
libras	**0.8572**±0.0425	0.6888±0.0599	0.85±0.055	0.7538±0.0525	0.7511±0.0455
	220.033	65.2071	266.071	129.963	**34.0787**, 0.125, 0.5

Continued on next page

Table 3.2 – continued from previous page

Datasets	**RoF**	**MPRoF-T**	**MPRoF-P**	**MPRoF-N**	**Proposed TBRoF**
	Accuracy±Std	Accuracy±Std	Accuracy±Std	Accuracy±Std	Accuracy±Std
	Time(s)	Time(s)	Time(s)	Time(s)	Time(s), c_1, c_3
low-res-spect	**0.8942**±0.0194	0.8263±0.0374	0.8847±0.0317	0.8071±0.0425	0.8621±0.0272
	316.559	36.0088	272.133	61.7302	**31.3146**, 4, 32
lymphography	**0.8351**±0.0428	0.7256±0.0627	0.8216±0.0506	0.8013±0.08	0.7689±0.0597
	4.19793	1.2612	6.21308	1.83226	**1.68999**, 0.0625, 16
mammographic	0.7841±0.0233	0.7779±0.026	0.7735±0.0239	0.7894±0.0205	**0.8102**±0.0195
	23.1274	5.76322	18.9043	6.02513	**22.3329**, 0.25, 0.25
molec-biol-promoter	**0.8483**±0.0642	0.7105±0.0978	0.8387±0.0531	0.7184±0.1075	0.7125±0.0696
	20.1039	1.59418	24.7746	2.67736	**1.06812**, 0.25, 32
musk-1	0.8962±0.0355	0.8075±0.0387	**0.897**±0.0292	0.8815±0.0351	0.8558±0.0297
	728.169	49.8765	772.725	105.082	**17.7031**, 2, 32
oocytes-merluccius-nucleus-4d	**0.8389**±0.019	0.7385±0.0241	0.7919±0.029	0.7868±0.0328	0.8231±0.0184
	429.307	13.6073	254.303	21.5857	**33.0952**, 0.5, 0.03125
oocytes-merluccius-states-2f	**0.9281**±0.0131	0.8953±0.0176	0.9183±0.0153	0.9091±0.0134	0.9178±0.0113
	235.95	5.44743	156.404	9.11346	**35.8439**, 0.5, 0.25
oocytes-trisopterus-nucleus-2f	**0.8337**±0.0184	0.728±0.0323	0.7714±0.0253	0.7618±0.026	0.8015±0.022
	259.341	8.43164	124.478	14.1039	**21.0519**, 0.25, 1
oocytes-trisopterus-states-5b	0.9326±0.0115	0.9002±0.019	**0.9353**±0.0142	0.9287±0.0131	0.9309±0.0153
	284.709	4.92042	239.741	9.70073	**28.4211**, 1, 0.5

Continued on next page

Table 3.2 – continued from previous page

Datasets	**RoF** Accuracy±Std Time(s)	**MPRoF-T** Accuracy±Std Time(s)	**MPRoF-P** Accuracy±Std Time(s)	**MPRoF-N** Accuracy±Std Time(s)	**Proposed TBRoF** Accuracy±Std Time(s), c_1, c_3
parkinsons	**0.9106**±0.056	0.8438±0.0571	0.8859±0.0461	0.8802±0.0582	0.8928±0.0463
	24.4298	1.22282	14.4354	2.21427	**2.92968**, 0.125, 4
pima	0.7486±0.0273	0.683±0.0305	0.7263±0.0271	0.7192±0.0235	**0.7653**±0.0316
	76.3127	5.03872	30.7515	6.39343	**9.23343**, 1, 0.25
pittsburg-bridges-MATERIAL	0.8299±0.0688	0.8166±0.0704	0.8274±0.0813	**0.856**±0.0493	0.8151±0.0601
	2.73545	0.728494	2.93366	0.874335	**1.41142**, 0.03125, 0.25
pittsburg-bridges-REL-L	0.6671±0.0986	0.6188±0.1147	0.6585±0.0963	**0.6685**±0.0884	0.6011±0.1044
	3.27439	1.15099	3.94896	1.38253	**2.40118**, 0.03125, 2
pittsburg-bridges-SPAN	**0.626**±0.0696	0.5717±0.0884	0.6108±0.0791	0.6239±0.0763	0.5956±0.087
	3.08439	1.08511	3.37997	1.25927	**2.44113**, 0.0625, 32
pittsburg-bridges-T-OR-D	0.8545±0.0633	0.8605±0.0418	0.8389±0.0481	0.8297±0.0593	**0.8608**±0.0644
	2.1259	0.514897	1.40227	0.587455	**0.967282**, 0.25, 2
planning	0.6857±0.0499	0.6224±0.0663	0.6431±0.0628	0.6276±0.0606	**0.712**±0.0439
	18.8087	1.62626	10.8308	2.38968	**1.94542**, 0.5, 0.125
post-operative	0.6282±0.1088	0.5721±0.1306	0.6261±0.1235	0.6395±0.1163	**0.7094**±0.1065
	2.21755	0.917702	2.85467	1.0935	**0.394733**, 8, 8
seeds	0.9389±0.0302	0.9227±0.0403	0.9285±0.0377	0.9143±0.0377	**0.9523**±0.0286
	6.70914	0.653522	4.46574	0.965109	**3.15239**, 4, 0.5

Continued on next page

Table 3.2 – continued from previous page

Datasets	**RoF** Accuracy±Std Time(s)	**MPRoF-T** Accuracy±Std Time(s)	**MPRoF-P** Accuracy±Std Time(s)	**MPRoF-N** Accuracy±Std Time(s)	**Proposed TBRoF** Accuracy±Std Time(s), c_1, c_3
statlog-australian-credit	0.6377±0.0295	0.571±0.0301	0.5907±0.0336	0.5901±0.0275	**0.6782**±0.0249
	104.333	6.67357	56.661	9.82792	**4.75702**, 4, 8
statlog-german-credit	**0.7708**±0.0276	0.6796±0.0297	0.766±0.0313	0.7452±0.035	0.7512±0.032
	111.489	8.67586	110.458	12.7143	**23.2292**, 0.5, 0.0625
statlog-heart	**0.8206**±0.042	0.7386±0.0405	0.8021±0.0405	0.7865±0.0474	0.8112±0.0539
	18.717	1.64374	14.8313	2.00331	**3.2454**, 2, 2
statlog-image	**0.9833**±0.0061	0.9326±0.0121	0.9767±0.0063	0.9324±0.0175	0.9637±0.0076
	361.169	16.7608	230.238	32.9771	**96.6169**, 0.125, 0.03125
statlog-vehicle	0.7902±0.0237	0.7643±0.0285	**0.8002**±0.0324	0.7867±0.0347	0.7943±0.0225
	165.044	7.61175	106.054	13.576	**27.3681**, 0.5, 0.125
synthetic-control	**0.9906**±0.0073	0.949±0.0215	0.9716±0.0193	0.977±0.0126	0.98±0.0192
	231.585	6.37416	165.883	14.4046	**10.8487**, 0.25, 4
tic-tac-toe	**0.9895**±0.0055	0.9716±0.0076	0.9835±0.0072	0.9832±0.0067	0.9832±0.0067
	13.4263	1.26946	6.08404	1.3285	**10.1453**, 8, 0.0625
vertebral-column-2clases	**0.8676**±0.0345	0.8302±0.0361	0.8476±0.0339	0.8426±0.0425	0.8504±0.0451
	10.4226	1.17567	5.82557	1.44774	**4.82573**, 1, 4
wine	0.9754±0.0285	0.9742±0.0244	0.9754±0.025	0.9674±0.0247	**0.9943**±0.01
	8.86472	0.483222	2.849	0.586307	**0.957313**, 0.0625, 4
zoo	0.9403±0.0527	0.9366±0.0585	0.9364±0.0509	0.9386±0.0491	**0.9505**±0.0424

Continued on next page

Table 3.2 – continued from previous page

Datasets	**RoF** Accuracy±Std Time(s)	**MPRoF-T** Accuracy±Std Time(s)	**MPRoF-P** Accuracy±Std Time(s)	**MPRoF-N** Accuracy±Std Time(s)	**Proposed TBRoF** Accuracy±Std Time(s), c_1, c_3
	1.63271	0.920206	2.50957	1.16041	**1.30516**, 1, 1
Average Accuracy	0.8361	0.7800	0.8233	0.8068	0.8226
Average Rank	1.8469	4.5918	2.7347	3.4592	2.3673
Overall Win-Tie-Loss	21-0-1	0-0-34	6-0-2	4-0-8	18-0-3

Here, CTG denotes cardiotocography.

Table 3.3: Classification accuracy[3] of RRoF, MPRRoF-T, MPRRoF-P, MPRRoF-N and proposed TBRROF.

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
balance-scale	0.8646±0.0209	**0.903**±0.0169	0.8937±0.0201	0.9023±0.0203	0.902±0.0218
	4.09479	3.15363	4.34982	3.37573	**7.7355**, 0.125, 0.0625
balloons	0.5625±0.179	0.5375±0.2599	0.6375±0.2496	0.6375±0.2217	0.625±0.25
	0.124138	0.132158	0.148812	0.135168	**0.300394**, 0.03125, 0.125
breast-cancer-wisc-diag	0.9666±0.0101	0.9729±0.0118	0.9725±0.013	0.9715±0.0108	**0.9753**±0.0083
	15.6612	1.30613	5.7209	1.50981	**6.63948**, 0.0625, 2
breast-cancer-wisc-prog	0.8133±0.04	0.8061±0.0425	0.803±0.046	0.804±0.0435	0.8133±0.0395
	7.21432	1.24913	4.05133	1.59607	**3.16648**, 0.0625, 0.25

Continued on next page

[3]Average accuracy obtained in five times 4-fold cross validation

Table 3.3 – continued from previous page

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
breast-cancer-wisc	0.9662±0.0132	0.965±0.0116	0.9642±0.0126	0.9665±0.0116	**0.9668**±0.0119
	4.79869	1.16632	2.15567	1.29555	**11.276**, 0.125, 0.03125
breast-cancer	0.726±0.0519	0.7203±0.045	0.7294±0.0435	0.7246±0.0474	**0.7357**±0.047
	3.61585	2.09501	3.3657	2.31028	**6.4252**, 0.5, 32
CTG-10clases	**0.8509**±0.0159	0.8133±0.017	0.8401±0.013	0.802±0.0204	0.8257±0.0183
	116.953	32.0436	95.0748	34.6305	**87.5706**, 0.03125, 0.03125
CTG-3clases	**0.932**±0.0105	0.9174±0.0097	0.9242±0.0096	0.9151±0.0103	0.9201±0.0116
	91.7823	10.5951	43.0608	12.1037	**88.621**, 0.03125, 0.03125
conn-bench-sonar-mines-rocks	0.8298±0.048	0.8201±0.0437	0.8211±0.0409	0.8057±0.0521	0.8298±0.0455
	12.0361	1.72983	6.60695	2.1017	**4.01408**, 0.25, 2
cylinder-bands	**0.7972**±0.0227	0.7718±0.0365	0.7773±0.0315	0.7453±0.032	0.78±0.0332
	17.5184	4.02551	12.5728	4.52613	**10.5769**, 0.125, 1
dermatology	**0.9764**±0.0171	0.9742±0.0143	0.9742±0.0168	0.9743±0.0147	0.9748±0.0151
	3.88823	2.47363	4.8165	3.06688	**6.07008**, 0.0625, 0.5
echocardiogram	0.814±0.0461	0.8348±0.0477	0.8321±0.0502	0.8231±0.0484	**0.8487**±0.0465
	1.89361	0.727468	1.26649	0.753632	**1.86393**, 0.5, 8
fertility	0.878±0.0922	0.88±0.083	0.874±0.0802	0.88±0.083	0.88±0.082
	0.832361	0.498083	0.795095	0.476865	**1.32369**, 0.0625, 1
haberman-survival	0.7071±0.0456	0.7052±0.0447	0.7083±0.0419	0.7071±0.0435	**0.7391**±0.0456
	5.14339	2.01874	2.44577	1.85193	**2.42135**, 0.03125, 8

Continued on next page

Table 3.3 – continued from previous page

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
heart-hungarian	0.8081±0.045	0.813±0.0431	**0.8169**±0.0451	0.8082±0.0435	0.813±0.0441
	3.97653	1.62576	2.84685	1.63621	**3.93937**, 0.5, 0.03125
hepatitis	0.8078±0.0673	0.8206±0.0621	0.8232±0.0624	**0.8296**±0.0579	0.8246±0.0474
	2.09076	0.854563	1.72104	0.920929	**2.13331**, 0.03125, 0.25
ilpd-indian-liver	0.7108±0.0343	0.7108±0.0325	**0.7129**±0.0374	0.7036±0.0418	0.7125±0.0279
	14.5554	3.65307	5.81799	3.71275	**9.51685**, 0.25, 0.25
ionosphere	0.9452±0.0257	0.9429±0.0223	**0.9498**±0.0203	0.9436±0.0212	0.9469±0.015
	15.4361	1.68565	8.05488	1.74154	**5.1604**, 0.0625, 0.25
iris	0.9498±0.0371	0.9644±0.0294	0.963±0.0271	0.9655±0.0268	**0.9683**±0.0248
	1.02457	0.442621	0.619184	0.463659	**1.65091**, 0.25, 0.03125
led-display	0.7036±0.0326	0.7044±0.0327	**0.7046**±0.0324	0.7016±0.0336	0.662±0.0724
	3.96341	6.77295	8.0428	5.88161	**16.0665**, 0.0625, 0.5
libras	0.865±0.0401	0.8683±0.0378	0.8688±0.0261	0.8533±0.0372	**0.8744**±0.029
	32.6966	9.97352	21.7712	10.8252	**15.0519**, 0.125, 0.03125
low-res-spect	0.8914±0.0314	0.8944±0.025	0.8899±0.03	0.8895±0.0286	**0.9012**±0.028
	38.8281	5.48187	30.9212	7.00611	**12.2346**, 0.0625, 0.0625
lymphography	0.8351±0.0567	0.8418±0.052	0.8405±0.0649	0.8297±0.0632	**0.8445**±0.0643
	1.73855	1.31614	2.36874	1.46099	**2.84016**, 0.0625, 0.03125
mammographic	0.8054±0.0265	0.8214±0.0272	0.8035±0.0224	0.8204±0.026	**0.827**±0.0225
	9.01402	5.00246	7.33527	4.93632	**9.75736**, 0.0625, 2
molec-biol-promoter	**0.8473**±0.0588	0.78±0.0821	0.8076±0.0633	0.7789±0.0923	0.7968±0.0614

Continued on next page

Table 3.3 – continued from previous page

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
	3.77228	1.00445	2.93334	1.14846	**1.72298**, 0.0625, 1
musk-1	**0.9004**±0.0293	0.8932±0.0265	0.8886±0.03	0.8886±0.0293	0.8865±0.0285
	47.0719	4.72229	23.9045	6.00834	**7.38054**, 0.125, 0.25
oocytes-merluccius-nucleus-4d	0.834±0.0246	0.8387±0.024	0.8399±0.0221	**0.842**±0.0278	0.8395±0.0302
	67.1065	7.52705	24.1753	8.59568	**21.8265**, 0.03125, 0.125
oocytes-merluccius-states-2f	0.9228±0.0129	0.9252±0.0137	0.925±0.0146	0.9252±0.0132	**0.9273**±0.0119
	41.5749	4.35347	16.9693	4.90871	**19.6346**, 0.03125, 0.0625
oocytes-trisopterus-nucleus-2f	0.8241±0.0191	0.8208±0.0203	0.826±0.0182	0.8245±0.0218	**0.8364**±0.0174
	50.7754	6.49082	16.6738	7.25101	**17.8984**, 0.03125, 0.03125
oocytes-trisopterus-states-5b	0.9285±0.0144	0.9287±0.016	0.9287±0.0142	0.9296±0.0136	**0.9313**±0.0156
	56.9417	4.75703	20.3494	5.75159	**19.7663**, 0.03125, 0.0625
parkinsons	0.908±0.0439	**0.9344**±0.0325	0.9232±0.0346	0.923±0.0463	0.9203±0.042
	4.75873	0.920433	2.52871	1.06274	**2.07906**, 0.03125, 0.0625
pima	0.7507±0.029	0.7468±0.028	0.7523±0.0323	0.7541±0.0329	**0.7601**±0.0314

Continued on next page

Table 3.3 – continued from previous page

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
	20.4454	4.9798	8.63783	5.10208	**15.4454**, 0.5, 0.25
pittsburg-bridges-MATERIAL	0.8247±0.0433	0.8416±0.0513	0.8322±0.0499	0.8436±0.0467	**0.8605**±0.0488
	1.12882	0.753009	1.30153	0.798122	**1.53054**, 0.25, 0.25
pittsburg-bridges-REL-L	0.6856±0.077	0.696±0.0783	**0.7038**±0.087	0.699±0.0724	0.7025±0.0811
	1.67203	1.22721	1.9887	1.24888	**2.38282**, 0.03125, 8
pittsburg-bridges-SPAN	0.6413±0.0976	0.6521±0.0858	**0.6804**±0.0917	0.6586±0.1067	0.6652±0.0998
	1.5292	1.09759	1.80332	1.19927	**2.14966**, 0.03125, 32
pittsburg-bridges-T-OR-D	0.8665±0.0661	0.88±0.0676	**0.8825**±0.0712	0.8708±0.0687	0.8763±0.0712
	1.05365	0.56043	0.907649	0.586204	**1.42341**, 0.125, 0.25
planning	0.7091±0.0656	0.7209±0.0617	0.7208±0.0574	0.722±0.0669	**0.7253**±0.063
	6.05221	1.59899	2.56125	1.56499	**3.59153**, 0.0625, 0.03125
post-operative	0.6299±0.0665	0.6592±0.0681	0.6456±0.0613	0.6702±0.0649	**0.7104**±0.0652
	1.1116	0.991297	1.63455	1.01075	**0.455685**, 8, 0.0625
seeds	0.9418±0.0352	0.9426±0.029	0.9428±0.0301	0.9399±0.0367	**0.9457**±0.0292
	3.19883	0.777223	1.93635	0.865785	**3.64155**, 8, 0.03125
statlog-australian-credit	0.6423±0.0285	0.6203±0.0354	0.6266±0.0276	0.6368±0.0306	**0.6783**±0.0259
	23.5354	6.00389	11.3536	6.48367	**2.41396**, 8, 0.0625

Continued on next page

Table 3.3 – continued from previous page

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
statlog-german-credit	**0.7648**±0.024	0.7434±0.0269	0.7566±0.0288	0.7412±0.022	0.7548±0.0285
	20.3747	7.23834	15.0714	7.58658	**19.6344**, 0.03125, 0.5
statlog-heart	0.8356±0.0542	0.8238±0.0504	0.8259±0.0498	0.8334±0.0493	**0.8394**±0.053
	4.79391	1.50302	3.42737	1.74863	**3.73004**, 0.5, 1
statlog-image	0.9779±0.0064	0.9774±0.0048	**0.9796**±0.0068	0.9759±0.0062	0.9758±0.0062
	76.0733	12.0994	37.9945	13.4508	**71.504**, 0.03125, 0.03125
statlog-vehicle	0.7793±0.0325	0.7725±0.0371	0.7784±0.0391	0.7744±0.0359	**0.7853**±0.0361
	34.0603	7.85545	20.1122	8.66157	**20.941**, 0.03125, 0.03125
synthetic-control	0.9863±0.0085	**0.9923**±0.0075	0.9893±0.0079	0.987±0.0073	0.9886±0.0061
	30.6564	3.40665	22.1319	4.9698	**10.9711**, 0.125, 0.5
tic-tac-toe	**0.9893**±0.0054	0.9868±0.0089	0.9866±0.0092	0.9843±0.0114	0.9839±0.01
	6.55748	4.60064	6.91135	5.0801	**13.8609**, 0.03125, 0.0625
vertebral-column-2clases	**0.8612**±0.0231	0.8574±0.0214	0.8464±0.0293	0.8541±0.0299	0.8541±0.0343
	3.43212	1.35993	1.82605	1.43099	**4.21239**, 0.0625, 0.0625
wine	0.9775±0.0206	0.9797±0.0216	0.9785±0.0225	**0.9808**±0.0184	0.9797±0.0191
	2.43472	0.586922	1.73049	0.721739	**2.60316**, 4, 0.03125
zoo	0.9424±0.0507	0.9403±0.0526	**0.9462**±0.0554	0.9424±0.0584	0.9423±0.0556
	0.528149	0.747051	0.990783	0.827133	**1.61777**, 0.125, 0.0625

Continued on next page

Table 3.3 – continued from previous page

Datasets	**RRoF** Accuracy±Std Time(s)	**MPRRoF-T** Accuracy±Std Time(s)	**MPRRoF-P** Accuracy±Std Time(s)	**MPRRoF-N** Accuracy±Std Time(s)	**Proposed TBRRoF** Accuracy±Std Time(s), c_1, c_3
Average Accuracy	0.8363	0.8359	0.8396	0.8364	0.8440
Average Rank	3.3980	3.3061	2.8163	3.3673	2.1122
Overall Win-Tie-Loss	9-0-18	3-0-10	9-0-5	3-0-11	21-0-4

Here, CTG denotes cardiotocography.

3.2.5 Comparison among the proposed ensemble models and MPSVM based ensemble models

The main motive in this subsection is to verify whether the proposed TBRaF, TBRoF and TBRRoF improve the performance with respect to other classifiers. Thus, we want to ensure that the improvement if any observed is due to different random strategies or due to ensemble methodology.

We first rank the performance of RaF, MPRaF-T, MPRaF-P, MPRaF-N and the proposed TBRaF on each dataset and take average across all the datasets. After calculations, the average ranks of RaF, MPRaF-T, MPRaF-P, MPRaF-N and the proposed TBRaF are $3.7449, 2.8265, 3.2143, 3.1020$ and 2.1122, respectively. Then after evaluation, $\chi_F^2 = 28.0064$ and $F_F = 8.0021$. with five algorithms and 49 datasets, F_F is distributed according to the F-distribution with $5-1 = 4$ and $(5-1)(49-1) = 192$ degrees of freedom. The critical value of $F_{(4,192)}$ for $\alpha = 0.05$ is 2.42. So, we reject the null hypothesis. Based on the Nemenyi test with $\alpha = 0.05$, the critical difference (CD) for $q_\alpha = 2.728, k = 5, N = 49$ is 0.8714. Since the difference among the average ranks of pairs (RaF, MPRaF-T), (RaF, TBRaF), (MPRaF-P, TBRaF) and (MPRaF-N, TBRaF) are $0.9184, 1.6327, 1.1021, 0.9898$ which is larger than the critical difference. We can conclude that the proposed model is significantly better than the other baseline models except MPRaF-T. However, the Nemenyi test fails to detect the significant difference between the TBRaF and MPRaF-T. Also, MPRaF-T shows better performance as compared to the RaF model. The proposed TBRaF achieves maximum average

accuracy and lower average rank among all the baseline ensemble classifier models.

Similarly, the average ranks of RoF, MPRoF-T, MPRoF-P, MPRoF-N and the proposed TBRoF are $1.8469, 4.5918, 2.7347, 3.4592$ and 2.3673, respectively. Then after evaluation, $\chi_F^2 = 89.0707$ and $F_F = 39.9834$. With five algorithms and 49 datasets, F_F is distributed according to the F-distribution with $5-1 = 4$ and $(5-1)(49-1) = 192$ degrees of freedom. The critical value of $F_{(4,192)}$ for $\alpha = 0.05$ is 2.42. So, we reject the null hypothesis. Based on the Nemenyi test with $\alpha = 0.05$, the critical difference (CD) for $q_\alpha = 2.728, k = 5, N = 49$ is 0.8714. The significant difference among the average ranks of pairs (MPRoF-T, RoF), (MPRoF-P, RoF), (MPRoF-N, RoF), (MPRoF-T, MPRoF-P), (MPRoF-T, MPRoF-N), (MPRoF-T, TBRoF) and (MPRoF-N, TBRoF) are $2.7449, 0.8878, 1.6123, 1.8571, 1.1326, 2.2245$ and 1.0919, respectively. Based on this, the significant difference among the algorithms is shown in Table 3.5. One can see from Table 3.5 that Nemenyi test shows that the TBRoF is better than MPRoF-T and MPRoF-N. However, the Nemenyi test fails to detect the significant difference among the TBRoF and other baseline methods like RoF and MPRoF-P. The proposed TBRoF achieves lower average rank as compared to the other baseline methods except RoF model. However, in terms of average accuracy no improvement is shown compared to RoF and MPRoF-P models.

Also, the average ranks of RRoF, MPRRoF-T, MPRRoF-P, MPRRoF-N and TBRRoF are $3.3980, 3.3061, 2.8163, 3.3673$ and 2.1122, respectively. Then after evaluation $\chi_F^2 = 23.6836$ and $F_F = 6.5972$. The significant difference among the average ranks of pairs (RRoF, TBRRoF), (MPRRoF-T, TBRRoF) and (MPRRoF-N, TBRRoF) are $1.2858, 1.1939$ and 1.2551, respectively. In the Table 3.4-3.6, the numbers in the bracket represent the average rank for the algorithm, the numbers in different cells represent the statistical difference between the method in the corresponding column and its corresponding row. Empty entry means there is no significant difference between the row method and column method.

From the analysis given in Table 3.6, one can see that the proposed TBRRoF is significantly better as compared to other baseline methods. However, Nemenyi test fails to detect any significant difference between the proposed TBRRoF and MPRRoF-P. The proposed TBRRoF achieves lower rank and highest average accuracy as compared to MPRRoF-P. Also, the proposed TBRRoF emerged as the overall winner in more number of datasets as compared to other baseline models.

From Tables 3.1, 3.2 and 3.3, one can see that the average accuracy and the average rank

Method	RaF (3.7449)	MPRaF-T (2.8265)	MPRaF-P (3.2143)	MPRaF-N (3.1020)	TBRaF (2.1122)
RaF (3.7449)					
MPRaF-T (2.8265)	**0.9184**				
MPRaF-P (3.2143)					
MPRaF-N (3.1020)					
TBRaF (2.1122)	**1.6327**		**1.1021**	**0.9898**	

Table 3.4: Significant difference for RaF and its ensembles at $\alpha = 0.05$.

Method	RoF (1.8469)	MPRoF-T (4.5918)	MPRoF-P (2.7347)	MPRoF-N (3.4592)	TBRoF (2.3673)
RoF (1.8469)		**2.7449**	**0.8878**	**1.6123**	
MPRoF-T (4.5918)					
MPRoF-P (2.7347)		**1.8571**			
MPRoF-N (3.4592)		**1.1326**			
TBRoF (2.3673)		**2.2245**		**1.0919**	

Table 3.5: Significant difference for RoF and its ensembles at $\alpha = 0.05$.

Method	RRoF (3.3980)	MPRRoF-T (3.3061)	MPRRoF-P (2.8163)	MPRRoF-N (3.3673)	TBRRoF (2.1122)
RRoF (3.3980)					
MPRRoF-T (3.3061)					
MPRRoF-P (2.8163)					
MPRRoF-N (3.3673)					
TBRRoF (2.1122)	**1.2858**	**1.1939**		**1.2551**	

Table 3.6: Significant difference for RRoF and its ensembles at $\alpha = 0.05$.

Table 3.7: Significant difference between the TBRaF, TBRoF, TBRRoF ensemble methods and MPRaF, MPRoF, MPRRoF ensemble methods based on win-tie-loss: sign test.

Method	Significance
(TBRaF, MPRaF-T) (34,14)	✔
(TBRoF, MPRoF-T) (45,4)	✔
(TBRRoF, MPRRoF-T) (37,9)	✔

The numbers in the bracket for (A,B) (say) method represent the number of times A wins w.r.t. method B , ✔ means there is significant difference between this pair of algorithms.

of the classifiers of the proposed TBRaF, TBRoF and TBRRoF are better or comparable as compared to other classifier models. The proposed TBRaF, TBRoF and TBRRoF showed consistent performance with all the three standard base models of decision trees (RaF, RoF and RRoF). However, the MPSVM based decision trees especially MPRoF-T and MPRoF-N does not show the consistent performance.

Table 3.8: Average rank of the ensemble methods for different minleaf parameter.

Method	$minleaf = 1$	$minleaf = 2$	$minleaf = 3$	F_F
RaF	1.51	2.3	2.2	**12.3812**
MPRaF-T	1.84	2.06	2.1	0.9596
MPRaF-P	1.61	2.3	2.1	**8.2094**
MPRaF-N	1.66	2.06	2.28	**5.2623**
TBRaF	1.7	2	2.3	**4.7473**
RRoF	1.51	2.3	2.2	**12.3812**
MPRRoF-T	1.84	2.06	2.1	0.9596
MPRRoF-P	1.61	2.29	2.1	**6.7383**
MPRRoF-N	1.66	2.06	2.28	**5.2623**
TBRRoF	1.7	2	2.3	**4.7473**

3.2.6 Win-tie-loss: sign test

From Table 3.7, one can see that the sign test shows significant differences among (TBRaF, MPRaF-T), (TBRoF, MPRoF-T) and (TBRRoF, MPRRoF-T) pair of classifiers. The proposed models achieve better performance as compared to the Tikhonov regularized MPSVM based oblique decision trees.

3.2.7 On the effect of minleaf

Parameter minleaf in decision trees represents the maximum number of data samples within an impure node. Generally, smaller trees are generated with the large values of minleaf. Zhang and Zhang [299] reported that the accuracy of decision tree ensemble is robust to minleaf parameter. However, Lin and Jeon [162] reported that the optimal values of minleaf parameter is situation dependent. In this subsection, we evaluate the results obtained by varying the minleaf parameter from 1 to 3 on all the 49 datasets. The average ranks of each method with different minleaf parameters are given in Table 3.8. With 3 parameter variations ($minleaf = 1, 2, 3$) and 49 datasets, F_F is distributed according to the F-distribution with $3 - 1 = 2$ and $(3 - 1)(49 - 1) = 96$ degrees of freedom. The critical value of $F_{(2,96)}$ for $\alpha = 0.05$ is 3.09. Thus, if $F_F > 3.09$ then there is a significant difference among the values corresponding to different minleaf parameters of the method in that row. The methods in which significant difference exist among the different minleaf parameters are highlighted.

From Table 3.8, one can see that in most of the cases Null hypothesis is rejected

while as in some methods no significant difference exist among varying minleaf parameters. Based on the Nemenyi test, critical difference (CD) for $\alpha = 0.05, q_\alpha = 2.343, k = 3$ and $N = 49$ is 0.4734. In RaF method, the significant difference exist among the varying minleaf parameters as the difference among the pairs $(RaF_{\text{minleaf}} = 2, RaF_{\text{minleaf}} = 1)$ and $(RaF_{\text{minleaf}} = 3, RaF_{\text{minleaf}} = 1)$ is greater than 0.4734. Further, $RaF_{\text{minleaf}} = 1$ achieves lowest rank and hence shows better performance. Likewise, the significant difference exist among the varying minleaf parameter of the methods where F_F values are highlighted. One can see from Table 3.8 that in most of the cases smaller minleaf parameter leads to lower rank and better performance.

3.3 Summary

In this chapter, we presented a novel approach to generate the decision tree ensembles with different base models like RaF, RoF and RRoF. Here, the splitting plane is decided based on multivariate features in each non-leaf node. This hyperplane is based on the hyperplanes generated by TBSVM. Unlike MPSVM based oblique decision trees, the proposed TBRaF, TBRoF and TBRRoF require no explicit regularization techniques. This is due to the reason that in TBSVM based oblique decision trees, the matrices appearing in the dual formulation are positive-definite. Also, the structural risk minimization principle is implemented in the proposed models. The proposed models show consistent performance with all the three baseline methods (RaF, RoF, RRoF). Among all the models, the proposed TBRaF and TBRRoF models emerge as the overall winner in 21 terms of accuracy while as other models show lower number of overall wins. The proposed TBRaF, TBRoF and TBRRoF show consistent performance with different base classifiers.

In this chapter, we presented the variants of the ensembles of oblique decision tree via TBSVM. Standard RaF and the ensembles of oblique decision tree via TBSVM used bootstrapping only at the root node. Recent study of double random forest revealed that the application of bootstrapping at each non-leaf node results in better performance. Moreover, the presented models uses TBSVM which solves QPP at each non-leaf node, which is computationally expensive. Hence, next chapter presents double RaF based ensemble which overcomes these issues.

Chapter 4

Oblique and rotation double random forest

In previous chapter, we presented several variants of oblique ensembles of decision tree. In this chapter, we present oblique and rotation double random forest. The proposed work is inspired by the recent study of double random forest [91]. We used the concepts of rotations (principle component analysis and linear discriminant analysis) and oblique hyperplane via multi-surface proximal support vector machine (MPSVM). As suggested by Breiman [23], the strength of unstable learners and the diversity among them are the ensemble models' core strength. In this chapter, we present two approaches known as oblique and rotation double random forests. In the first approach, we propose a rotation based double random forest. In rotation based double random forests, transformation or rotation of the feature space is generated at each node. At each node different random feature subspace is chosen for evaluation, hence the transformation at each node is different. Different transformations result in better diversity among the base learners and hence, better generalization performance. With the double random forest as base learner, the data at each node is transformed via two different transformations namely, principal component analysis and linear discriminant analysis. In the second approach, we propose oblique double random forest. Decision trees in random forest and double random forest are univariate, and this results in the generation of axis parallel split which fails to capture the geometric structure of the data. Also, the standard random forest may not grow sufficiently large decision trees resulting in suboptimal performance. To capture the geometric properties and to grow the decision trees of sufficient depth, we propose oblique double random forest. The oblique double random forest models are multivariate decision trees. At each non-leaf node, multisurface proximal support vector machine generates the optimal plane for better generalization performance. Also, different

regularization techniques (Tikhonov regularisation and axis-parallel split regularisation) are employed for tackling the small sample size problems in the decision trees of oblique double random forest. The evaluation of the baseline models and the proposed oblique and rotation double random forest models is performed on benchmark UCI datasets and real-world fisheries datasets. Both statistical analysis and the experimental results demonstrate the efficacy of the proposed oblique and rotation double random forest models compared to the baseline models on the benchmark datasets.

Recent study of double random forest [91] evaluated the effect of node size on the performance of the model. The study revealed that the prediction performance may improve if deeper decision trees are generated. The authors showed that the largest tree grown on a given data by the standard random forest might not be sufficiently large to give the optimal performance. Hence, double random forest [91] generated decision trees that are bigger than the ones in standard random forest. Instead of training each decision tree with different bags of training set obtained via bagging approach at the root node, the authors in [91] generated each tree with the original training set and used bootstrap aggregation at each non-terminal node of the decision tree to obtain the best split. However, both the random forest and double random forest are univariate decision trees and hence ignore the geometric class distributions resulting in lower generalization performance. To overcome these issues, we propose oblique double random forest. oblique double random forest models integrate the benefits of double random forest and the geometric structure information of the class distribution for better generalization performance. For generating more diverse ensemble learners in the double random forest, feature space is rotated or transformed at each non leaf node using two transformations known as linear discriminant analysis and principal component analysis. Using transformations at each non-leaf node on different randomly chosen feature subspaces improves diversity among the base models and leads to better generalization performance.

The main highlights of this chapter are:

- We use different rotations (principal component analysis and linear discriminant analysis) at each non-leaf node to generate diverse double random forest ensembles (DRaF-PCA and DRaF-LDA).

- The proposed oblique double random forest (MPDRaF-T, MPDRaF-P and MPDRaF-

N) variants use MPSVM for obtaining the optimal separating hyperplanes at each non-terminal node of the decision tree ensembles.

- The proposed ensemble of double Random forest generate larger trees compared to the variants of standard Random forest.

- Statistical analysis reveals that the average rank of the proposed double random forest models is superior than the standard random forest. Moreover, the average accuracy of the proposed DRaF-LDA, DRaF-PCA, and MPDRaF-P is superior than the standard random forest and standard double random forest models. Also, the average rank of the proposed DRaF-LDA, MPDRaF-P and DRaF-PCA is better compared to the standard double Random forest.

4.1 Handling multiclass problems

MPSVM is a binary classification model and finding the optimal separating hyperplanes at each non-terminal node of a decision tree may be a multiclass problem. To handle the multiclass problem via binary class approach, different methods like one-versus-all [16], one-versus-one [138], decision directed acyclic graph [213], error correcting output codes [59] and so on have been proposed. Data partitioning rule of the decision trees at each non-leaf node proves handy over other binary classification models [303]. Separating the classes with majority samples as one class and rest samples as another class results in an inefficient model as it fails to capture the geometric structure of the data samples [173]. To incorporate the geometric structure, the authors in [303] decomposed the multiclass problem into a binary one by using class separability information. The authors used Bhattacharyya distance for decomposition. In statistics, Bhattacharyya distance gives the measure of similarity between the two discrete probability distributions or continuous probability distributions as it is deemed to be a good insight about separability of classes between two normal classes $C_1 \sim N(\mu_1, \nu_1), C_2 \sim N(\mu_2, \nu_2)$, where μ_i and ν_i are the parameters of the normal distribution of class C_i, for $i = 1, 2$. Following the similar approach as in [303], we used multivariate Gaussian distribution [124]. Motivated by [124, 303], we use Bhattacharyya distance to measure the class separability for decomposing the multiclass problem into a binary class problem (Algorithm 4.1).

Algorithm 4.1 Decomposition of multiclass problem to a binary class problem

Input:

$D := N \times n$ be the training dataset with N number of data points with feature size n.

$Y := N \times 1$ be the target labels.

$\{L_1, L_2, \ldots, L_C\}$ be the target labels.

Output:

C_p and C_n are two hyperclasses or groups

For each class $j = 1, 2, \ldots, C$.

[1] For each pair of L_j and L_k, for $k = j + 1, \ldots, C$ as:

$$F(L_j, L_k) = \frac{1}{8}(\mu_k - \mu_j)^t \Big(\frac{\nu_j + \nu_k}{2}\Big)^{-1}(\mu_k - \mu_j) + \frac{1}{2} ln \frac{|(\nu_j + \nu_k)/2|}{\sqrt{|\nu_j||\nu_k|}} \tag{4.1}$$

[2] Find the pair L_p and L_n of classes with the maximum Bhattacharyya distance, and assign them to C_p and C_n respectively.

[3] For every other class, if $F(L_k, L_p) < F(L_k, L_n)$ then group L_k to C_p otherwise group in C_n.

Algorithm 4.2 Null space regularization

Input: P (Positive class) and H (Negative class).

Output: Clustering hyperplane $\begin{bmatrix} w \\ b \end{bmatrix}$.

[1] Suppose P is rank deficit with rank $r < n+1$, calculate $O = [\alpha_1, \alpha_2, \cdots, \alpha_{n+1-r}]$ whose columns are the orthonormal basis for the Null space of P.

[2] Project the matrix Q in the Null space of P. For each vector (row) p in matrix P, the projection is given as pOO^t. Hence, the projection of matrix Q is given as $\bar{Q} = \sum_{p \in Q} OO^t p^t pOO^t = OO^t QOO^t$. In the similar manner, the projeection of matrix P is given as $\bar{P} = OO^t POO^t$.

[3] Since the columns of O span the Null space of P, hence $\bar{P}$ would be zero. Thus, the desired plane is the eigen vector corresponding to the largest eigenvector of $\bar{Q}$.

4.2 Proposed oblique and rotation double random forest

This chapter presents two approaches for generating the double random forest known as oblique double random forest models and the rotation based double random forest models. Two approaches are given as follows:

4.2.1 Oblique double random forest with MPSVM

Univariate decision trees don't capture properties of the data geometrically. Both standard random forest and double random forest are univariate decision tree ensembles. Also, decision trees in the standard random forest may not be large enough for the datasets to get the better generalization. To overcome these limitations, we propose oblique double random forest with MPSVM. Unlike standard random forest, the oblique double random forest models with MPSVM use bootstrapping samples at every non-terminal node (until some condition is met as given in Algorithm 4.3) for generating the optimal oblique splits and divide the original data instead of bootstrapped samples among the children nodes. To incorporate the geometric structure in the splitting hyperplane, the proposed oblique double random forest uses MPSVM wherein optimal split at each non-leaf node is generated based on the clustering hyperplanes. As the decision tree size increases, the data points arriving at a particular node decreases and hence, the issues of sample size may arise. To overcome this issue, we use different regularization techniques to obtain a better generalization performance. The regularization approaches used are Tikhonov regularization, axis parallel split regularization and null space approach. If the model uses Tikhonov regularization then the proposed model is named as oblique double random forest via MPSVM with Tikhonov regularization (MPDRaF-T), if the model uses axis parallel split regularization then the proposed model is known as oblique double random forest via MPSVM with axis parallel split regularization (MPDRaF-P) and if the model uses null space approach then the proposed models is known as oblique double random forest via MPSVM with null space approach (MPDRaF-N). In Tikhonov regularization, the small positive number is added along the diagonal elements to regularize the data matrix (say, H) i.e., if data matrix H is

rank deficient, then regularize H as :

$$H = H + \delta \times I, \tag{4.2}$$

where δ is a small positive number and I is appropriate dimensional identity matrix. In axis-parallel split regularization, if the data matrix (say, H) is rank deficient at a given node then we follow axis parallel approach to complete the growth of decision tree. Thus, heterogeneous test functions are used for growing the decision trees. i.e., till the current node MPSVM is used for generating the optimal splits and now onwards axis parallel approach is followed for growing the decision tree. In order to handle the sampling issues, Manwani and Sastry [173] proposed the Null space approach (given in Algorithm 4.2) for regularizing the matrices. For the proposed MPDRaF-N, we follow the Algorithm 4.2 for regularizing the matrices.

Algorithm 4.3 summarises the oblique double random forest with MPSVM.

4.2.2 Double random forest with PCA/LDA

For generating the diverse learners in an ensemble, we propose rotation based double random forest ensemble models. Rotation or transformation on different random feature subspaces results in different projections leading to better generalization performance. In this method, the objective is to rotate or transform the data for better diversity among the base learners. At each non-leaf node, the rotation is applied on random feature subspace which results in improved diversity among the base classifiers. We use two approaches for rotation of feature subspace i.e., principal component analysis (PCA) and linear discriminant analysis (LDA).

The proposed double random forest with PCA (DRaF-PCA) is given in Algorithm 4.4. At each non-leaf node, rotation or transformation is applied on the bootstrapped samples reaching a given node with random feature subspace.

The algorithm of the proposed double random forest with LDA (DRaF-LDA) varies from Algorithm 4.4 at step (ii) and (iii). In DRaF-LDA model, instead of calculating total scatter matrix S_d at each node, within class scatter matrix S_d^w and between class scatter matrix S_d^b are calculated. Then, generalized eigenvectors of (S_d^w, S_d^b) are calculated ($S_d^b \times \alpha = \lambda \times S_d^w$, where α is the generalized eigenvector corresponding to the generalized eigenvalue λ).

Algorithm 4.3 Oblique Double Random Forest with MPSVM

Training Phase:

Given:

$D := N \times n$ be the training set with N number of samples with feature size n.
$D_i := N_i \times n_i$ be the training samples reaching to a node i, with N_i number of samples with feature size n_i.
$Y := N \times 1$ be the target labels.
L : is number of base learners.
"mtry": number of candidate features to be evaluated at each non-leaf node.
"nodesize" or *"minleaf"*: maximum number of data samples to be placed in an impure node.

For each decision tree, T_i for $i = 1, 2, \ldots, L$

[1] Use training data D.

[2] Generate the decision tree T_i with randomly chosen subset of features and randomised bootstrap instance using D:
For a given node d with data D_d:

(i) if $N_d > N \times 0.1$
Generate bootstrap sample D_d^* from D_d.
else
$D_d^* = D_d$

(i) Choose *"mtry"*$= \sqrt{n}$ number of features from the given feature space of D_d^*

(ii) Using Algorithm 4.1 group the dataset D_d^* into C_p and C_n.

(iii) Use MPSVM (with different regularization's) for generating the optimal split with C_p and C_n as input, and split the data D_d into child nodes.

Repeat steps (i)-(iii), until the stopping criteria is one of the conditions is met:

- Node reaches to purest form.
- Samples reaching a given node are lesser or equal than *minleaf*

Classification Phase:

For a test data point x_i, use the decision trees of the forest to generate the label of the test sample. The predicted class of the test data point is given by the majority voting of the decision trees of an ensemble.

Algorithm 4.4 Double Random Forest with PCA

Training Phase:

Given:

$D := N \times n$ be the training set with N number of samples with feature size n.
$D_i := N_i \times n_i$ be the training samples reaching to a node i, with N_i number of samples with feature size n_i.
$Y := N \times 1$ be the target labels.
L : is number of base learners.
"mtry": number of candidate features to be evaluated at each non-leaf node.
"nodesize" or *"minleaf"*: maximum number of data samples to be placed in an impure node.

For each decision tree, T_i for $i = 1, 2, \ldots, L$

[1] Use training data D.

[2] Generate the decision tree T_i with randomly chosen subset of features and randomised bootstrap instance using D:
For a given node d with data D_d:

(i) if $N_d > N \times 0.1$
Generate bootstrap sample D_d^* from D_d.
else
$D_d^* = D_d$

(i) Choose *"mtry"*$= \sqrt{n}$ number of features from the given feature space of D_d^*

(ii) Calculate total scatter matrix S_d using D_d^*.

(iii) Calculate all the eigenvectors of S_d, denoted by V.

(iv) Calculate the data transformation using all the eigenvectors V as, $D_{PCA}^* = D_d^* * V$.

(v) In the PCA space, search the best feature split.

(iii) With the optimal split feature and the cutpoint, split the data D_d into the child nodes.

Repeat steps (i)-(iii), until the stopping criteria is met.

Classification Phase:

For a test sample x_i, generate labels via decision trees of the forest. At every non-terminal node, the test data sample is rotated with the same matrix V generated in the training stage. The predicted class of the test data point is given by the majority voting of decision trees of an ensemble.

4.3 Comparison of the proposed oblique and rotation based double random forest models with the existing baseline models

The main differences of the proposed models with respect to the existing models are given as follows:

[1] MPDRaF-T, P, N are the oblique double random forest variants which employ bagging at each non leaf node to allow the generation of bigger trees. Unlike standard variants like RaF, MPRaF-T, MPRaF-P and MPRaF-N, the proposed models use the training bags which have more unique instances of the samples which results in generation of bigger trees. Moreover, MPDRaF-T,P,N capture the geometric properties of the data which is ignored by the standard RaF and double RaF models.

[2] The standard RaF and DRaF models use the concepts of random subspace and bagging for introducing the diversity among the base learners of an ensemble. However, the proposed DRaF-PCA and DRaF-LDA employ PCA and LDA transformations at non-leaf nodes in addition to the random subspace and bagging concepts for producing more diverse base learners. Thus, the proposed DRaF-PCA and DRaF-LDA models possess better diversity compared to the RaF and DRaF models. Unlike RaF-PCA and RaF-LDA, the proposed DRaF-PCA and DRaF-LDA models use bagging concept at each non-leaf node which allow greater depth of the tree and hence better performance.

4.4 Experiments

Here, we discuss the setup followed in experiments and analyze the performance of the proposed oblique and rotation double random models and baseline models or existing models (here, standard RaF [23], standard DRaF [91], MPRaF-T [303], MPRaF-P [303], MPRaF-N [303], RaF-PCA [305] and RaF-LDA [305]).

4.4.1 Experimental setup

We evaluated the classification models on UCI datasets [60] and real world fisheries datasets [81]. We follow the preprocessing scripts of [136] wherein the partitions of the training and testing sets are publicly available for evaluation. The sample size of the datasets varies from 10 to 130064. Also, the dimensions of the feature samples vary from 3 to 262 and the number of classes vary from 2 to 100.

In all the ensemble models, 50 is the number of base learners. At each non-terminal node, we evaluated $\sqrt{n}$ number of features, here n is the dimension of feature set and the minleaf parameter is set to default. We used CART [25] as the base classifier.

4.4.2 Statistical analysis

Table 8.2 summarizes the classification performance of each ensemble model on 121 datasets. From the given table, it is evident that the average accuracy of the proposed DRaF-LDA, MPDRaF-P and DRaF-PCA are superior compared to the existing classifiers. Following [62], we rank each classifier based on its performance on each dataset. Every classifier in Friedman test is given a rank on a dataset with the worse performing classifier assigned higher rank and vice versa. Hence, a lower rank indicates better generalization performance of the model. The average rank of each classification model is presented in Table 4.3. It is evident that the average rank of the proposed ensemble models DRaF-LDA, DRaF-PCA, and MPDRaF-P is better as compared to all the existing classifiers. Furthermore, the rank of the proposed MPDRaF-T is better in comparison to existing classifiers (except standard DRaF and DRaF-LDA).

The average ranks of the classification models RaF, MPRaF-T, MPRaF-P, MPRaF-N, RaF-PCA, RaF-LDA, DRaF, MPDRaF-T, MPDRaF-P, MPDRaF-N, DRaF-PCA and DRaF-LDA are $6.99, 6.81, 6.48, 8, 7.31, 6.12, 6.27, 6.38, 5.45, 7.3, 5.84$ and 5.04 respectively. With simple calculations, we get $\chi_F^2 = 71.0559$ and $F_F = 6.7675$. At 5% level of significance i.e. $\alpha = 5\%$, F_F follows F-distribution with $(n-1) = 11$ and $(n-1)(N-1) = 1320$. From Statistical table, $F_F(11, 1320) = 1.8$. Since $6.7675 > 1.8$, hence we reject the null hypothesis. Thus, significant difference exists among the classification models. To get the significant difference, we use Nemenyi post hoc test. With simple calculations, critical difference $CD = 1.5149$ with $q_\alpha = 3.268$ at 5% level of significance. From Figure 4.1, one can

	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T	MPDRaF-P	MPDRaF-N	DRaF-PCA	DRaF-LDA
RaF									$r-$			$r-$
MPRaF-T												$r-$
MPRaF-P				$r+$								
MPRaF-N			$r-$			$r-$	$r-$	$r-$	$r-$		$r-$	$r-$
RaF-PCA									$r-$			$r-$
RaF-LDA				$r+$								
DRaF				$r+$								
MPDRaF-T				$r+$								
MPDRaF-P	$r+$			$r+$	$r+$					$r+$		
MPDRaF-N									$r-$			$r-$
DRaF-PCA				$r+$								
DRaF-LDA	$r+$	$r+$		$r+$	$r+$					$r+$		

Here, $r+$ denotes that the the row model is significantly better than the column model. $r-$ denotes that the row model is significantly worse than the corresponding column model. Empty entries denote that no significant difference exists among the models of a cell.

Table 4.1: Significance difference of classification performance of the baseline models and the proposed oblique and rotation double random forest with Nemenyi posthoc tests based on the accuracy.

see the statistically significant difference exists among the models which are not connected by a line. Table 4.1 summarizes the Nemenyi post-hoc test results. From the table, it is evident that the proposed DRaF-LDA is significantly better in comparison to RaF, MPRaF-T, MPRaF-N, RaF-PCA and MPDRaF-N classifiers. Also, the proposed DRaF-PCA is significantly better compared to the DRaF-PCA model.

Table 4.2: Classification accuracy of RaF [23], MPRaF-T [303], MPRaF-P [303], MPRaF-N [303], RaF-PCA [305], RaF-LDA [305], DRaF [91], MPDRaF-T, MPDRaF-P, MPDRaF-N, DRaF-PCA and DRaF-LDA classification models.

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
abalone	64.68	64.99	65.54	65.06	64.85	65.4	64.18	65.33	63.94	65.33	64.06	65.35
acute-inflammation	100	100	100	100	100	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100	100	100	100	100	100
adult	85.79	85.04	85.54	84.5	85.6	85.47	85.63	85.12	85.45	84.41	85.58	85.47
annealing	54.25	76	38.75	76	62.25	65	37	76	56	76	65.25	64
arrhythmia	73.01	63.05	73.23	61.5	65.49	67.04	73.67	63.05	73.89	60.62	70.35	70.35
audiology-std	75	70	76	24	55	48	78	59	78	25	60	58
balance-scale	86.7	89.42	88.94	89.42	88.62	89.42	82.69	87.82	86.86	89.58	85.42	86.38
balloons	81.25	87.5	87.5	93.75	81.25	75	87.5	93.75	81.25	81.25	81.25	87.5
bank	89.6	88.61	89.2	88.63	89.45	89.91	89.93	88.87	89.29	88.83	89.6	89.82
blood	76.6	76.74	77.27	77.81	76.6	77.01	75.67	77.14	75.94	77.14	75.8	76.34
breast-cancer	73.94	73.94	73.94	73.94	76.06	76.76	75.35	73.24	74.65	76.06	72.89	75
breast-cancer-wisc	97.29	97.71	97.43	97	97.14	97.43	97.14	97.86	97.71	97.57	97.43	97.29
breast-cancer-wisc-diag	95.6	96.83	96.83	97.71	95.6	97.01	95.77	97.01	96.65	96.3	96.83	97.01
breast-cancer-wisc-prog	80.1	80.61	79.59	81.12	80.1	80.61	81.63	82.65	82.14	83.67	82.14	82.14
breast-tissue	70.19	69.23	71.15	71.15	73.08	75	73.08	69.23	69.23	68.27	73.08	70.19

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
car	96.93	95.31	96.99	88.08	96.76	96.76	97.05	95.37	97.8	87.96	97.16	98.15
cardiotocography-10clases	86.11	82.44	85.59	79.8	84.37	84.84	87.15	83.47	86.53	81.87	84.93	85.64
cardiotocography-3clases	94.02	92.75	94.26	91.57	92.33	93.27	94.92	93.22	94.73	92.23	92.7	93.69
chess-krvk	69.6	65.97	70.12	52.02	73.33	71.72	69.92	66.34	71.36	52.22	75.18	73.24
chess-krvkp	98.25	97.97	98.62	97.43	98.25	98.59	98.56	98.53	98.94	97.84	98.75	98.81
congressional-voting	62.39	61.24	61.01	61.24	60.55	61.01	61.7	62.39	61.7	60.78	60.78	61.7
conn-bench-sonar-mines-rocks	76.92	78.37	78.85	78.37	76.44	78.85	79.33	77.4	80.77	79.33	80.77	84.13
conn-bench-vowel-deterding	98.48	99.78	99.13	99.62	99.57	99.46	98.97	100	99.73	99.68	99.95	99.95
connect-4	83.54	76	81.2	75.41	82.63	82.31	84.01	75.91	81.7	75.4	83.37	82.83
contrac	53.94	50.41	53.13	49.8	51.9	50.68	53.33	48.78	51.15	52.31	51.15	51.56
credit-approval	87.5	86.92	86.05	88.08	88.08	87.5	87.5	86.63	87.21	85.61	87.21	87.06
cylinder-bands	81.25	76.17	80.47	73.05	78.71	77.73	82.03	76.95	82.03	79.3	80.86	80.47
dermatology	98.35	98.08	98.35	96.7	97.8	97.8	98.08	97.8	97.8	97.8	97.53	97.25
echocardiogram	84.85	84.85	85.61	84.09	84.09	83.33	84.09	85.61	84.85	84.85	84.09	84.09
ecoli	86.31	87.2	88.99	87.8	87.5	87.5	88.1	86.01	88.1	88.39	86.61	86.61

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
energy-y1	94.79	92.58	94.92	94.14	94.53	95.7	95.83	92.19	95.96	94.27	96.09	96.22
energy-y2	89.06	89.45	89.84	89.32	89.97	89.71	88.28	89.06	88.8	89.71	89.84	89.45
fertility	88	89	89	88	88	88	88	88	88	88	88	88
flags	67.19	55.21	64.58	56.77	56.25	57.29	66.67	54.69	63.02	54.69	62.5	62.5
glass	73.11	69.34	75.47	70.28	70.75	72.17	76.89	69.34	75.47	67.92	71.7	74.06
haberman-survival	71.05	71.71	71.05	72.37	70.07	71.38	69.74	70.39	69.08	70.72	69.41	68.75
hayes-roth	87.5	86.61	84.82	81.25	89.29	87.5	89.29	85.71	90.18	75	89.29	88.39
heart-cleveland	57.89	61.51	57.57	59.21	58.22	59.21	55.59	59.21	58.22	59.54	60.2	57.57
heart-hungarian	83.9	84.93	84.25	84.25	84.59	84.59	84.25	83.56	83.56	85.27	84.59	84.59
heart-switzerland	41.13	43.55	41.13	44.35	43.55	45.16	41.94	39.52	41.94	41.13	45.97	47.58
heart-va	35.5	34.5	35.5	36.5	34	37.5	36	32.5	36.5	39.5	33.5	34.5
hepatitis	83.33	82.05	82.05	86.54	82.69	84.62	82.69	82.69	81.41	82.69	81.41	80.77
hill-valley	53.84	66.75	63	65.88	64.03	66.25	54.17	70.09	66.79	66.58	67.2	66.75
horse-colic	86.4	86.03	87.87	87.5	82.35	85.29	86.76	83.46	86.76	84.56	80.51	81.62
ilpd-indian-liver	71.4	70.72	71.23	71.23	73.29	71.23	71.23	72.6	72.95	71.23	73.46	71.75
image-segmentation	93.8	94.18	94.75	92.46	94.96	95.07	94.85	94.63	95.15	92.57	95.93	96.06
ionosphere	91.76	93.75	93.47	93.18	94.03	94.6	91.48	93.75	94.03	94.32	94.89	93.18
iris	95.27	97.3	97.3	97.97	95.95	96.62	95.95	97.3	95.95	97.3	96.62	96.62
led-display	74.3	72	73.7	72.4	73.9	73.6	71.7	72.1	72.4	71.2	71.6	72.1

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
lenses	83.33	79.17	83.33	79.17	79.17	87.5	83.33	79.17	75	83.33	79.17	79.17
letter	95.31	95.35	95.18	94.83	94.74	95.62	95.86	95.85	96	95.02	96.06	96.73
libras	76.94	84.17	79.17	79.44	80.28	81.39	79.72	86.11	84.72	86.67	85.28	85.83
low-res-spect	90.79	91.17	91.35	89.47	90.6	91.54	91.54	91.17	91.73	90.79	91.17	91.35
lung-cancer	46.88	46.88	50	53.13	40.63	43.75	50	31.25	53.13	50	46.88	50
lymphography	79.05	85.14	83.11	83.78	84.46	84.46	85.81	86.49	83.11	85.81	84.46	86.49
magic	87.01	86.37	86.26	86.69	87.38	87.06	87.19	86.51	86.78	86.88	87.79	87.57
mammographic	81.98	81.67	80.63	81.67	80.63	80.42	79.9	81.46	80.63	81.35	80.1	80.42
miniboone	93.33	93.07	93.24	92.76	93.21	93.46	93.69	93.5	93.64	93.3	93.65	93.88
molec-biol-promoter	84.62	79.81	84.62	82.69	71.15	78.85	91.35	84.62	87.5	80.77	83.65	85.58
molec-biol-splice	94.2	86.57	93.1	85.01	84.1	89.9	94.7	87.23	93.22	87.14	87.05	90.56
monks-1	59.95	60.59	58.39	57.52	58.04	58.16	60.65	60.47	58.8	58.97	58.22	59.32
monks-2	66.78	66.9	66.9	67.01	66.84	67.01	66.55	66.96	66.61	67.13	66.9	66.72
monks-3	53.01	56.6	52.78	54.34	53.36	52.89	52.78	54.17	52.95	52.78	53.01	53.76
mushroom	100	100	100	100	100	100	100	100	100	100	100	100
musk-1	86.13	86.97	83.82	87.18	86.13	83.82	86.34	89.29	86.97	87.82	88.24	85.92
musk-2	97.21	96.12	95.94	95.69	95.98	96.12	98.12	96.53	96.45	96.07	96.71	96.95
nursery	99.28	98.58	99.21	96.74	99.22	99.33	99.31	98.9	99.53	96.95	99.66	99.76
OM_nucleus_4d	77.65	81.57	82.55	80.69	82.55	82.65	79.8	84.31	83.24	82.84	82.45	84.31

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
OM_states_2f	91.37	91.57	92.16	91.86	91.86	92.16	92.35	92.35	92.35	92.55	92.06	92.45
OT_nucleus_2f	79.39	81.58	82.79	83.11	82.57	82.24	80.7	83.44	82.79	82.89	83.77	83.22
OT_states_5b	90.9	92.54	92	92.65	92.65	93.31	92.21	93.64	93.09	93.09	92.87	93.64
optical	96.08	95.66	96.26	84.65	95.72	91.62	96.91	96.37	96.74	83.85	96.59	94.3
ozone	97.08	97.2	97.16	97.16	97.16	97.16	97.08	97.16	97.16	97.16	97.16	97.2
page-blocks	97.08	96.98	97.3	96.78	97.09	97.13	97.08	97.08	97.09	97.08	97.09	97.28
parkinsons	88.78	92.35	89.8	91.84	87.76	90.82	90.31	92.86	92.35	92.35	90.82	91.84
pendigits	95.05	96.76	95.75	96.06	96.38	96.48	95.48	96.96	96.22	96.18	96.53	96.58
pima	76.69	75.26	75.52	75.13	74.61	74.87	73.96	74.74	74.22	74.48	74.09	75
pittsburg-bridges-MATERIAL	91.35	93.27	91.35	92.31	92.31	92.31	88.46	93.27	89.42	92.31	90.38	91.35
pittsburg-bridges-REL-L	74.04	75.96	73.08	75.96	75	73.08	73.08	74.04	71.15	78.85	73.08	75
pittsburg-bridges-SPAN	61.96	72.83	63.04	67.39	69.57	71.74	60.87	67.39	61.96	67.39	66.3	66.3
pittsburg-bridges-T-OR-D	88	88	88	88	88	90	89	88	88	88	88	88
pittsburg-bridges-TYPE	68.27	69.23	67.31	66.35	71.15	66.35	67.31	69.23	68.27	69.23	70.19	71.15
planning	70	67.78	70	70	70.56	69.44	69.44	71.67	71.11	72.22	70.56	70.56

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
plant-margin	79.25	75.06	72.25	72.56	76	75.5	81.56	76.88	77.5	73.06	79.75	82.44
plant-shape	59.44	66.13	62.13	65.25	65.31	68.44	61.94	67.75	66.31	66.13	70	73.44
plant-texture	77.94	77.25	76.06	75.06	75.81	76.81	80.56	79.13	79.06	75.56	79	81.63
post-operative	72.73	71.59	70.45	71.59	69.32	67.05	70.45	72.73	69.32	68.18	69.32	68.18
primary-tumor	54.88	51.83	55.18	52.44	53.05	56.1	54.88	53.05	53.66	53.35	54.57	54.57
ringnorm	95.19	90.41	90.81	90.85	97.01	97.09	95.46	91.99	92.15	92.54	97.24	97.15
seeds	93.27	94.71	91.83	91.83	93.75	92.31	93.75	93.75	95.19	92.31	92.79	93.75
semeion	92.4	89.51	91.52	89.13	88.69	91.96	92.46	89.7	92.09	90.52	91.14	93.22
soybean	90.29	89.23	90.56	82.71	89.83	86.3	90.36	88.63	90.76	83.38	90.36	87.43
spambase	94.39	94.5	94.15	94.11	94.8	94.48	94.72	94.91	94.83	94.3	95.3	95.17
spect	68.95	61.56	65.46	59.68	60.75	61.29	65.05	60.75	63.04	60.22	61.42	62.1
spectf	91.98	91.98	91.98	91.98	91.98	91.84	91.98	91.84	91.98	91.98	91.98	91.84
statlog-australian-credit	67.3	65.26	66.57	67.15	63.66	63.23	64.39	63.37	65.55	63.52	64.53	63.08
statlog-german-credit	77.5	74.8	75.4	73.9	75.3	77.7	77.4	73.9	75.9	72.8	76.1	76.1
statlog-heart	85.45	87.31	86.19	86.19	85.45	85.45	85.07	85.82	85.45	83.21	85.45	85.45
statlog-image	97.27	97.66	97.57	96.66	97.88	97.92	97.88	97.92	98.31	97.18	98.09	98.27
statlog-landsat	89.94	89.99	89.99	89.04	89.78	89.88	90.78	90.89	90.73	89.44	90.76	90.98
statlog-shuttle	99.96	99.87	99.95	99.76	99.94	99.96	99.99	99.9	99.97	99.78	99.97	99.97

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
statlog-vehicle	73.58	76.3	77.73	75.95	78.08	79.03	75.71	76.18	77.61	77.25	78.32	80.45
steel-plates	78.04	78.04	76.75	75.15	75.05	76.49	78.4	78.2	78.76	76.86	77.94	77.99
synthetic-control	97.67	99.83	98.5	98.33	97.17	99.17	98.5	99.33	99.33	98.83	98.5	99.67
teaching	59.21	58.55	60.53	57.24	55.92	60.53	58.55	58.55	59.87	57.24	59.21	59.87
thyroid	98.88	95.86	98.89	93.26	98.7	97.65	98.96	96.05	98.93	93.46	98.87	98.16
tic-tac-toe	97.91	97.49	97.7	94.77	97.07	98.01	98.64	98.95	98.85	98.22	98.43	99.06
titanic	78.95	78.68	78.95	78.32	78.95	78.95	78.95	78.95	78.95	78.5	78.95	78.95
trains	87.5	100	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5	87.5
twonorm	96.8	97.59	97.68	97.57	97.68	97.55	96.8	97.57	97.53	97.42	97.68	97.66
vertebral-column-2clases	83.77	86.69	86.04	86.04	85.06	86.69	82.14	86.36	86.36	87.01	83.44	85.06
vertebral-column-3clases	83.44	84.09	83.44	83.77	83.77	83.77	84.74	86.04	84.42	85.71	85.39	86.36
wall-following	99.3	94.24	98.41	93.71	96.17	96.19	99.52	94.54	98.57	94.68	96.87	96.92
waveform	84.54	85.4	85.04	85.44	84.8	85.4	83.76	85.26	85.9	85.48	85.12	85.78
waveform-noise	85.5	85.2	86.24	85.74	85.08	85.84	85.22	85.44	85.44	85.52	85.6	86.14
wine	97.73	98.86	99.43	97.73	97.16	98.86	97.73	97.73	99.43	98.3	97.16	99.43
wine-quality-red	65.81	68	67.38	68.19	68.31	67.56	68	67.5	69.19	67.31	68.81	67.88
wine-quality-white	67.01	67.28	67.57	66.14	67.87	67.69	68.2	67.85	67.97	66.91	68.57	68.4
yeast	61.52	62.06	61.79	62.2	62.53	62.53	60.58	61.39	61.39	60.98	60.98	61.79

Continued on next page

Table 4.2 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
zoo	99	99	98	98	99	99	98	97	98	99	98	98
Average Accuracy	81.86	81.98	81.96	80.83	81.48	81.9	82.09	81.79	82.39	80.96	82.2	82.55

Here, * denotes the methods introduced in this chapter.

OM denotes oocytes_merluccius, OT denotes oocytes_trisopterus.

	Rank	Average Rank	Average Accuracy	Average Time(s)
DRaF-LDA*	1	5.04	82.55	758.68
MPDRaF-P*	2	5.45	82.39	80.83
DRaF-PCA*	3	5.84	82.2	765.81
RaF-LDA	4	6.12	81.9	732.63
DRaF	5	6.27	82.09	523.32
MPDRaF-T*	6	6.38	81.79	30.66
MPRaF-P	7	6.48	81.96	56.3
MPRaF-T	8	6.81	81.98	24.64
RaF	9	6.99	81.86	383.43
MPDRaF-N*	10	7.3	80.96	32.12
RaF-PCA	11	7.31	81.48	719.97
MPRaF-N	12	8	80.83	26.76

Here * denotes the methods introduced in this chapter.

Table 4.3: Overall comparison of the baseline classification models, proposed oblique and rotation double random forest models.

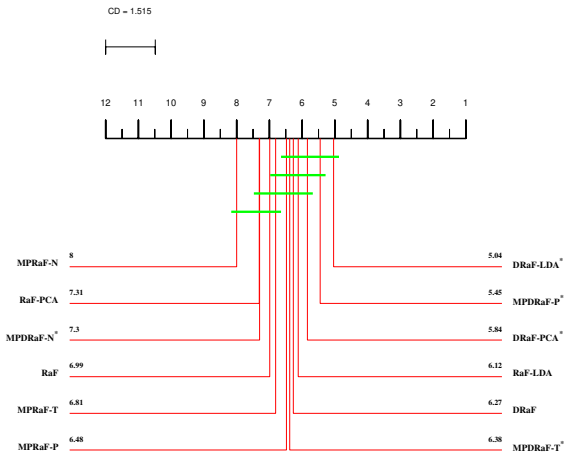


Figure 4.1: Nemenyi test based post hoc evaluation of classification models at $\alpha = 5\%$ level of significance. The classification models which are not statistically different are connected.

	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*
MPRaF-T	[57, 10, 54]										
MPRaF-P	[60, 17, 44]	[60, 13, 48]									
MPRaF-N	[50, 10, 61]	[36, 14, 71]	[40, 16, 65]								
RaF-PCA	[49, 16, 56]	[51, 8, 62]	[47, 11, 63]	[63, 15, 43]							
RaF-LDA	[67, 10, 44]	[63, 10, 48]	[61, 14, 46]	[77, 13, 31]	[70, 17, 34]						
DRaF	[69, 14, 38]	[65, 8, 48]	[58, 17, 46]	[64, 11, 46]	[60, 15, 46]	[54, 11, 56]					
MPDRaF-T*	[56, 14, 51]	[61, 15, 45]	[54, 10, 57]	[70, 15, 36]	[60, 14, 47]	[48, 12, 61]	[52, 12, 57]				
MPDRaF-P*	[69, 13, 39]	[74, 10, 37]	[70, 14, 37]	[76, 12, 33]	[68, 18, 35]	[67, 9, 45]	[63, 16, 42]	[59, 16, 46]			
MPDRaF-N*	[46, 14, 61]	[44, 12, 65]	[45, 13, 63]	[74, 13, 34]	[55, 14, 52]	[40, 12, 69]	[47, 14, 60]	[36, 17, 68]	[37, 13, 71]		
DRaF-PCA*	[61, 14, 46]	[65, 10, 46]	[56, 16, 49]	[77, 11, 33]	[72, 22, 27]	[64, 14, 43]	[57, 14, 50]	[59, 10, 52]	[49, 20, 52]	[70, 11, 40]	
DRaF-LDA*	[69, 11, 41]	[70, 10, 41]	[62, 15, 44]	[79, 11, 31]	[71, 16, 34]	[77, 12, 32]	[67, 12, 42]	[68, 15, 38]	[62, 13, 46]	[79, 8, 34]	[66, 22, 33]

Here, * denotes the proposed methods, $[a, b, c]$ entry in each cell denotes that row method wins a-times, loses c-times and ties b-times with respect to column method.

Table 4.4: Pairwise win-tie-loss count

	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
RaF						$r-$	$r-$		$r-$			$r-$
MPRaF-T									$r-$			$r-$
MPRaF-P									$r-$			
MPRaF-N						$r-$		$r-$	$r-$	$r-$	$r-$	$r-$
RaF-PCA						$r-$			$r-$		$r-$	$r-$
RaF-LDA	$r+$			$r+$	$r+$				$r-$			$r-$
DRaF	$r+$											$r-$
MPDRaF-T*				$r+$								$r-$
MPDRaF-P*	$r+$	$r+$	$r+$	$r+$	$r+$	$r+$						
MPDRaF-N*				$r+$							$r-$	$r-$
DRaF-PCA*				$r+$	$r+$					$r+$		$r-$
DRaF-LDA*	$r+$	$r+$		$r+$	$r+$	$r+$	$r+$	$r+$		$r+$	$r+$	

Here, * denotes the methods introduced in this chapter, $r+$ denotes that the method in the corresponding row is significantly better as compared to the method given in the corresponding column. $r-$ denotes that the row method is significantly worse than the method given in the corresponding column. Blank entries denote that no significant difference exists among the methods in the cell's corresponding row and column.

Table 4.5: Pairwise win-tie-loss: sign test

4.4.3 Win-tie-loss: sign test

Under the null hypothesis, the pair of classifiers is significantly different if each classification model wins $N/2$ in N datasets. The number of wins follow binomial distribution. When N is large enough, the number of wins follow $N(N/2, \sqrt{N}/2)$, and hence, z-test can be used: two models are significantly better with $p < 0.05$ if any model has least $N/2 + 1.96\sqrt{N}/2$ wins. Since tied matches favor of null hypothesis, hence, we split the number of ties between the models evenly and if the number is odd we ignore one.

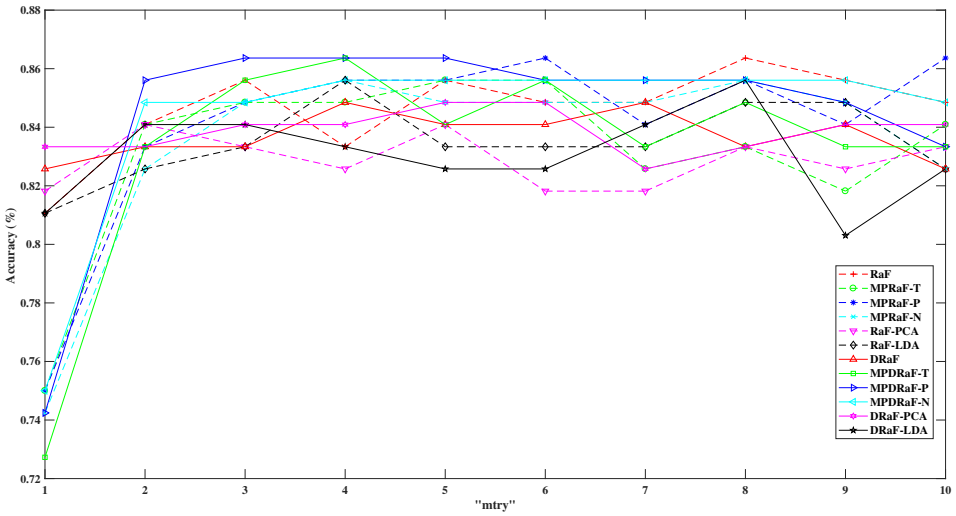
Table 4.4 summarizes the count of win tie loss results among the given classification models. One can see that the proposed rotation double random forest (DRaF-PCA and DRaF-LDA) achieved more wins as compared to the existing models. Compared to the existing MPRaF-N and RaF-PCA models, the proposed MPDRaF-N emerged as winner in more datasets. Also, the proposed MPDRaF-P model emerged as the winner in more datasets in comparison to the given baseline models. Table 4.5 shows that the proposed DRaF-LDA model is significantly better as compared to the RaF, MPRaF-T, MPRaF-N, RaF-PCA, RaF-LDA and DRaF models. The proposed DRaF-PCA model is significantly better compared to the existing MPRaF-N and RaF-PCA models. Also, the proposed MPDRaF-P is significantly better as compared to the existing models except DRaF model.

4.4.4 Effect of "mtry" parameter

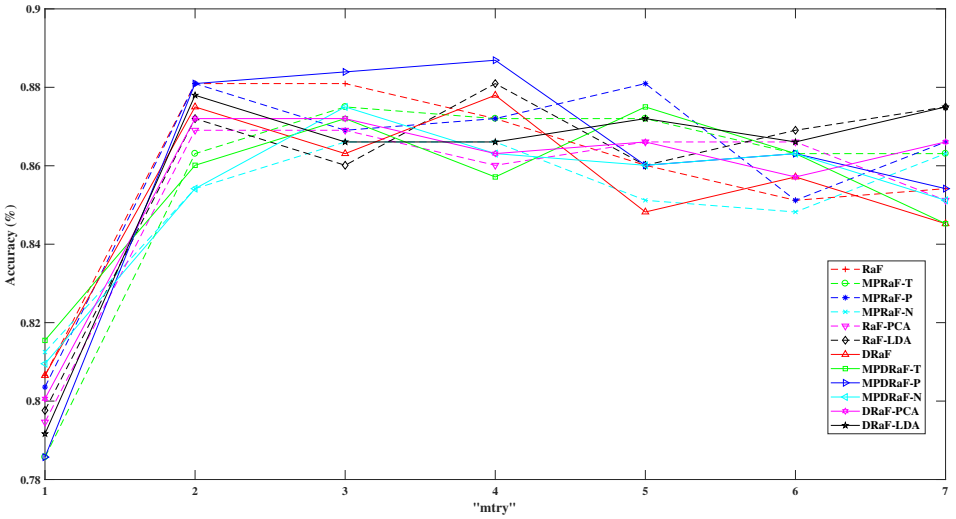
The parameter *"mtry"* denotes the number of candidate features to be evaluated at each non-leaf node. In a given problem, the smaller *"mtry"* results in stronger randomization among the trees and weaker dependency of their structures on the output. However, if the *"mtry"* is small, the random subset of features selected at a given node may fail to get the geometry of the data points. To see the effect of *"mtry"* parameter, we varied it to different values on the datasets given in Figure 4.2. From the Figure 4.2, it is clear that at very low values of *"mtry"*, the performance is lower. However, as the size of the *"mtry"* parameter increases, the performance starts increasing and becomes stable very quickly. Setting *"mtry"* to $round(\sqrt{n})$ leads to satisfactory performance.

Figure 4.2: Effect of the *"mtry"* parameter.

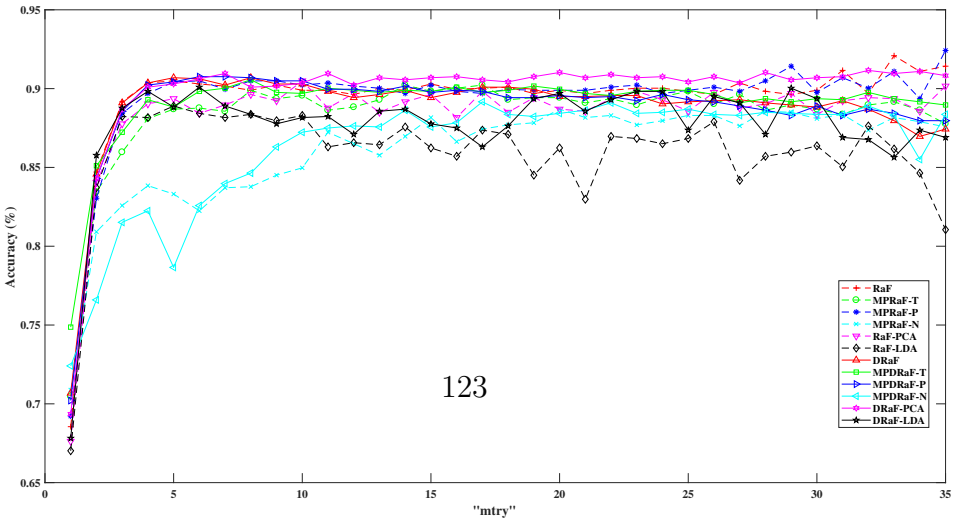
(a) Echocardiogram



(b) Ecoli



(c) Soybean



4.4.5 Effect of *"minleaf"* parameter

In the ensembles of decision tree *"minleaf"* denotes the maximum number of data samples to be placed in an impure node. In general, smaller trees are generated with higher minleaf which results in higher bias and lower variance. Zhang and Zhang [299] suggested that performance ensembles of decision tree are robust to this parameter while as Lin and Jeon [162] suggested that its optimal value varies in different situations. To analyse the effect of this parameter, we evaluated the effect of *"minleaf"* parameter with its value varying from 1 to 3 on 120 datasets (leaving miniboone dataset as it took huge time to compute for all these parameters). The average rank of each model across different parameters corresponding to each model are given in Table 4.6. With $N = 120$, $K = 3$ (as minleaf=1,2,3), $F_F(2, 238) = 3.03$. Significant difference exist among the different performances based on the minleaf value of the model if $F_F > 3.03$ (Table 4.6). From the given table, it is clear that significant difference exists among the performances of the all the models (except DRaF-LDA) with different minleaf parameters. However, in most of the cases smaller minleaf parameter results in better performance. This study is in consensus with the observation that decision trees of an ensemble should grow as much as possible for better performance.

Table 4.6: Average rank of the classification models with different *minleaf* parameters.

Method	$minleaf = 1$	$minleaf = 2$	$minleaf = 3$	F_F
RaF	1.87	1.87	2.26	6.3555
MPRaF-T	1.79	1.93	2.29	11.2753
MPRaF-P	1.66	2.11	2.23	11.8124
MPRaF-N	1.73	1.95	2.32	11.6113
RaF-PCA	1.73	2.01	2.27	12.1945
RaF-LDA	1.77	2.01	2.22	6.3555
DRaF	1.65	2.08	2.27	13.3546
MPDRaF-T*	1.87	1.94	2.19	3.4658
MPDRaF-P*	1.74	1.98	2.28	9.3988
MPDRaF-N*	1.76	1.93	2.3	7.0927
DRaF-PCA*	1.68	2	2.32	13.5758

Continued on next page

Table 4.6 – continued from previous page

Method	$minleaf = 1$	$minleaf = 2$	$minleaf = 3$	F_F
DRaF-LDA*	1.8	2.08	2.11	1.111

4.4.6 Average number of nodes

As seen in the above section that smaller minleaf results in better performance, hence, the performance of the models can be increased if there is a way to generate the bigger trees [91]. Thus, greater the size of the tree better the performance is. Here, we analyse the size of the tree via number of nodes. Average number of nodes denote that the average number of nodes in an ensemble. Table 4.7 gives the average of the nodes present in different ensembles of the classification models. From Figure 4.3 represents the average of mean nodes in different classification models. Figure 4.3, it is clear that double variants of the random forest have higher number of nodes compared to the standard variants of the random forest. Hence, the proposed variants of the double random forest show better performance due to larger size of the trees.

Table 4.7: Average number of nodes in RaF [23], MPRaF-T [303], MPRaF-P [303], MPRaF-N [303], RaF-PCA [305], RaF-LDA [305], DRaF [91], MPDRaF-T, MPDRaF-P, MPDRaF-N, DRaF-PCA and DRaF-LDA classification models.

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
abalone	349.81	481.93	392.75	372.79	304.43	296.08	470.07	447.08	491.77	324.92	444.69	439.34
acute-inflammation	4.9	5.04	4.97	4.74	4.46	4.44	5.05	4.99	5.18	4.72	4.32	4.54
acute-nephritis	3.89	4.46	4.03	4.09	3.57	3.66	4.01	4.48	4.3	4.27	3.52	3.77
adult	1731.66	2113.55	1843.9	1846.68	1480.15	1489.83	2132.73	2456.92	2321.47	2021.68	2016.47	1965.98
annealing	34.1	55.84	34.65	46.51	32.62	34.05	40.54	68.02	40.73	54.3	38.96	40.03
arrhythmia	36.6	63.65	36.12	63.92	34.81	35.39	43.64	58.73	43.54	58.12	46.24	47.53
audiology-std	17.26	23.98	17.22	23.18	16.63	17.89	14.43	26.65	13.98	23.34	15.89	16.14
balance-scale	38.45	38.18	37.65	34.35	32.13	30.73	35.96	36.71	36.53	30.1	33.8	32.89
balloons	2.1	1.84	2.13	1.78	2.23	2.33	1.25	1.28	1.38	1.32	1.92	1.86
bank	167.83	245.75	185.29	218.47	140.31	138.89	232.43	294.41	250.61	264.36	207.68	203.55
blood	44.22	39.85	47.36	44.86	41.17	40.84	54.81	41.74	54.41	46.77	56.34	56.92
breast-cancer	26.38	28.39	26.35	26.41	21.9	21.39	29.49	31.43	31.22	26.8	31.53	28.97
breast-cancer-wisc	14.94	17.11	14.83	17.75	11.36	10.34	19.77	19.92	18.68	22.86	16.07	15.21
breast-cancer-wisc-diag	11.41	14.47	11.79	18.57	11.87	8.36	16.65	18.06	15.71	23.04	17.41	11.69
breast-cancer-wisc-prog	12.66	17.26	13.81	19.59	11.65	9.77	19.42	21.51	20.14	23.98	19.2	17.14
breast-tissue	10.22	13.94	11.58	12.88	10.03	9.45	12.42	11.86	13.4	9.65	12.85	12.45

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
car	52.05	53.99	59.95	38.74	56.29	51.94	48.68	51.09	57.2	32.38	55.22	51.35
cardiotocography-10clases	134.08	242.91	144.6	253.69	134.46	113.43	161.43	256.29	174.09	270.15	176.92	148.62
cardiotocography-3clases	63.94	110.23	71.05	122.75	60.33	54.43	80.75	127.33	90.16	144.95	85.3	75.46
chess-krvk	922.92	1341.55	1138.44	651.32	1250.65	1226.39	720.29	897.58	876.76	390.42	1014.06	982.35
chess-krvkp	91.09	158.26	96.72	130.73	94.32	90.97	98.88	190.97	106.51	149.3	110.76	102.34
congressional-voting	9.03	11.01	8.65	8.95	8.6	8.12	6.46	13.07	6.49	10.67	8.65	8.58
conn-bench-sonar-mines-rocks	11.95	17.62	12.94	19.71	11.59	8.2	19.3	21.92	21.08	24.68	19.88	15.11
conn-bench-vowel-deterding	61.24	92.06	71.79	94.91	58.61	52.83	77.37	103.21	89.43	94.97	76.15	67.79
connect-4	1967.07	2058.1	1891.47	1183.77	2060.35	2069.17	2042.64	2231.71	1938.48	1325.44	2340.46	2169.71
contrac	132.21	143.02	134.2	112.51	122.42	118.87	131.37	124.59	137.45	94	144.96	138.52
credit-approval	38.91	57.35	43.62	52.44	34.06	32.43	52.69	67.08	56.72	61.46	51.09	47.16
cylinder-bands	41.29	57.62	42.98	58.21	32.63	30.92	59.68	63.21	63.02	63.14	52.22	51.39
dermatology	16.57	21.36	16.72	22.93	15.93	15.07	18.75	25.76	18.88	26.48	18.59	19.01
echocardiogram	9.39	12.91	10.54	13.51	8.47	8.16	12.5	13.4	13.18	13.64	12.37	11.55
ecoli	21.06	29.67	22.12	29.22	19.55	19.53	24.27	30.18	25.9	29.94	24.87	24.44

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
energy-y1	19.81	28.57	21.12	23.27	21.05	19.8	20.84	28.92	22.49	22.94	22.88	20.51
energy-y2	20.16	27.39	21.02	21.62	19.94	18.93	19.06	26.41	19.82	19.59	20.55	20.12
fertility	6.59	7.89	6.98	7.71	5.3	4.98	7.55	8.45	7.9	8.06	6.39	6.25
flags	22.14	29.94	23.32	30.2	19.74	19.1	20.38	24.02	20.65	23.11	22.19	22.11
glass	19.94	28.3	20.21	27.43	18.29	17.5	23.35	26.87	24.81	24.85	24.46	24.13
haberman-survival	25.08	30.43	27.5	30.54	22.39	22.72	31.43	30.7	29.78	29.58	31.97	31.9
hayes-roth	9.48	11.65	10.47	11.06	9.27	9.57	9.25	12.94	10.4	12.14	9.38	10.1
heart-cleveland	29.28	38.23	30.57	38.94	24.89	23.78	31.77	30.35	31.28	31.75	32.14	31.8
heart-hungarian	19.39	24.65	20.9	22.62	17.24	16.58	23.73	27.59	25.09	23.16	24.27	22.77
heart-switzerland	16.19	18.03	15.94	16.19	13.97	14.5	13.85	11.5	14.31	10.72	15.2	15
heart-va	24.8	29.49	24.4	25.87	22.4	22.69	21.02	17.98	21.08	16.34	25.44	25.66
hepatitis	10.18	13.48	10.88	12.77	8.45	7.74	13.07	14.49	13.59	13.81	11.91	11.54
hill-valley	66.19	35.12	78.26	66.38	44.66	33.52	107.02	48.96	113.11	81.06	71.38	57.43
horse-colic	24.01	35.35	26.48	36.79	21.26	18.99	30.83	42.35	35.99	43.91	32.1	29.08
ilpd-indian-liver	42.55	62.23	52.69	59.59	37.81	36.36	64.22	66.84	70.02	61.38	60.39	60.05
image-segmentation	15.59	27.32	18.32	28.83	15.1	13.31	18.55	29.1	22.34	31.26	19.57	17.15
ionosphere	13.52	21.57	16.08	23.27	12.08	10.21	18.44	27.29	22.89	29.9	18.13	15.86
iris	5.29	7.04	5.75	6.28	5.98	4.31	5.84	6.99	6.15	6.31	7.45	5.29
led-display	12.41	13.45	12.54	12.22	13.08	13.42	10.19	11.76	9.96	9.72	11.3	10.27

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
lenses	2.8	2.61	2.75	2.78	2.78	2.77	2.15	1.85	2.01	2.12	2.19	2.29
letter	1190.77	1941.31	1545.76	2019.17	1105.98	930.41	1277.75	1958.65	1592.73	1954.44	1307.54	1096.94
libras	35.05	52.7	41.16	56.74	33.61	24.04	46.59	58.94	50.94	60.77	45.32	35.67
low-res-spect	21.64	30.63	24.64	41.84	22.72	15.78	29.46	37.97	32.4	54.2	29.92	22.45
lung-cancer	4.49	4.47	4.37	4.87	4.04	3.63	4.23	3.11	4.42	3.11	4.3	4.2
lymphography	13.44	16.54	12.94	16.01	10.45	10.4	15.29	17.56	15.17	16.81	14.91	13.9
magic	778.89	1291.33	1027.86	1307.15	753.48	712.13	1111.39	1482.07	1405	1449.84	1107.32	1065.07
mammographic	44.23	41.93	46.28	37.9	38.79	40.24	55.53	50.51	55.49	44.05	51.62	52.76
miniboone	2878.18	3186	3325.18	3415.87	2829.45	2159.2	4173.28	3917.6	4949.34	4402.77	4223.4	3380.49
molec-biol-promoter	8.25	11.47	9.13	11.27	7.16	5.22	11.85	12.52	12.61	12.61	12.22	9.04
molec-biol-splice	176.93	313.37	203.18	340.19	189.1	121.92	222.68	344.12	267.96	367.3	270.48	185.86
monks-1	13.76	14.75	14.34	13.17	13.46	13.17	13.82	14.26	14.8	12.42	14.68	14.39
monks-2	18.58	19.18	19.15	17.15	19.37	20.08	17.63	15.94	17.98	13.79	21.23	20.87
monks-3	11.2	12.73	12.65	11.99	10.42	10.54	12.02	12.08	12.44	11.3	11.67	11.34
mushroom	22.75	40.82	28.52	40.07	25.59	22.62	22.69	41.7	29.31	40.72	25.97	22.32
musk-1	24.84	38.13	27.21	42.76	23.55	14.11	38.03	49.24	42.81	56.82	39.26	25.85
musk-2	122.19	209.28	154.56	269.33	131.13	80.8	155.73	272.22	218.71	358.09	185.13	119.86
nursery	251.01	304.95	292.95	227.6	284.02	229	238.47	317.62	291.65	207.74	284.79	238.4
OM_nucleus_4d	60.42	82	67.16	77.37	49.74	41.21	90.32	96.09	97.68	97.48	78.41	65.48

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
OM_states_2f	28.9	43.9	34.34	50.86	29.16	22.92	41.53	54.12	47.79	63.07	41.68	34.74
OT_nucleus_2f	54.49	81	67.18	87.08	49.45	43.09	81.65	97.52	96.57	105.17	78.13	69.05
OT_states_5b	35.15	48.35	36.61	60.54	32.36	24.35	50.86	61.42	54.25	80.15	47.17	36.41
optical	210.94	383.82	219.43	467.41	204.48	183.58	244.67	454.88	253.86	562.79	246.3	218.09
ozone	28.65	41.55	31.06	52.34	27.68	20.37	42.9	56.45	45.01	74.71	40.87	31.9
page-blocks	63.65	91.79	72.87	85.22	59.51	59	78.39	102.96	92.8	89.51	77.12	77.25
parkinsons	8.53	12.85	10.47	13.85	8.53	7.28	12.53	15.65	14.38	16.36	12.63	10.9
pendigits	210.57	320.17	237.25	385.79	200.55	153.25	254.88	371.52	279.76	434.7	245.63	188.28
pima	49.77	73.48	61.52	77.61	45.37	43.15	71.98	79.12	81.43	81.48	71.59	69
pittsburg-bridges-MATERIAL	7.64	8.8	7.63	8.55	6.54	6.35	8.78	9.04	8.5	8.33	7.39	7.36
pittsburg-bridges-REL-L	11.41	12.56	11.42	12.48	9.7	9.55	11.93	12.62	12.23	11.47	11.32	10.99
pittsburg-bridges-SPAN	10.1	11.39	10.42	11.4	8.78	8.55	10.38	10.78	10.49	9.89	9.76	9.72
pittsburg-bridges-T-OR-D	6.56	7.48	6.65	7.78	5.28	4.93	7.27	8.55	7.66	8.49	6.45	5.95
pittsburg-bridges-TYPE	12.21	14.65	12.47	14.11	11.06	11.04	11.24	11.1	11.46	11.3	10.85	10.98
planning	14.76	21.19	18.04	22.72	14.07	13.81	22.81	22.24	24.48	22.64	22.64	22.4

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
plant-margin	185.17	278.12	226.79	284.17	177.44	162.35	208.88	237.69	224.82	215.9	214.61	198.11
plant-shape	179.46	269.87	225.2	266.63	172.3	151.38	211.7	223.83	224.9	208.34	211.32	185.95
plant-texture	183.56	280.18	229.79	288.43	185.23	174.11	204.49	239.06	224.63	228.76	213.6	203.7
post-operative	9.47	10.15	9.36	9.94	8.33	8.2	8.78	9.8	9.02	8.52	9.1	8.89
primary-tumor	30.61	35.14	30.93	29.84	29.14	27.48	28.67	28.18	27.73	25.73	26.43	23.9
ringnorm	192.09	94.06	78.06	97.7	184.54	177.42	278.44	94.96	81.93	106.52	276.47	270.66
seeds	7.75	9.65	8.57	9.61	7.46	5.71	10.24	10.99	10.03	10.33	9.25	7.65
semeion	133.92	200.29	140.83	211.13	112.95	74.38	134.27	205.83	142.29	217.42	129.69	95.45
soybean	34.87	43.67	34.94	42.07	31.01	30.76	38.45	48.22	38.59	46.51	35.89	34.99
spambase	145.86	266.25	155.26	264.33	126.5	128.33	185.34	359.57	195.29	330.51	172.87	172.18
spect	8.87	10.66	9.01	10.16	7.56	7.34	7.92	11.14	8.36	9.83	9.03	8.89
spectf	7.22	9.79	7.9	8.53	6.88	5.17	10.53	12.95	11.62	11.17	11.03	8.53
statlog-australian-credit	61.65	84.09	67.34	80.9	51.56	50.41	88.45	86.61	93.81	81.66	82.81	81.69
statlog-german-credit	83.54	104.95	86.89	98.17	62.34	61.28	114.18	115.94	116.87	111.89	96.46	93.76
statlog-heart	19.09	23.63	19.61	24.06	15.01	13.64	25.98	27	25.5	27.88	22.85	21.07
statlog-image	53.56	104.54	66.08	110.84	55.86	44.66	64.94	122.42	82.5	126.87	70.88	57.08
statlog-landsat	212.51	282.05	215.65	348.32	183.51	147.53	279.69	349.92	291.84	441.74	255.52	214.28
statlog-shuttle	43.13	85.75	50.42	88.78	50.5	46.36	46.61	96.3	55.25	92.36	56.93	53.09

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
statlog-vehicle	62.92	86.28	69.01	86.57	54.83	48.37	84.86	88.85	89.87	86.25	80.76	72.17
steel-plates	137.59	219.51	151.26	223.54	127.45	116.3	174.26	212	191.89	216.55	172.86	163.19
synthetic-control	22.45	30.57	24.67	47.6	22.3	14.81	31.6	39.99	33.12	68.44	31.62	23.2
teaching	16.53	15.55	15.79	13.96	14.67	15.2	17.06	13.03	16.26	10.53	16.72	16.82
thyroid	35.33	97.28	38.4	86.65	43.65	40.25	40.6	125.71	41.32	102.93	53.44	49.49
tic-tac-toe	63.59	74.08	68.68	70.4	53.15	49.43	69.82	89.25	79.65	78.96	68.94	60.88
titanic	2.11	1.79	1.96	1.22	1.89	1.92	1.78	1.57	1.67	1.02	1.66	1.58
trains	1.86	1.83	1.82	1.84	1.85	1.87	1.86	1.33	1.83	1.29	1.62	1.79
twonorm	228.12	218.97	174.37	246.81	148.81	117.17	338.36	276.83	242.89	304.57	213.91	179.02
vertebral-column-2clases	16.76	24.45	21.93	22.62	17.19	16.12	23.89	26.32	26.1	24.54	24.36	23.27
vertebral-column-3clases	17.79	27.44	22.68	23.81	19.01	17.86	24.66	27.57	27	25.11	26.38	24.15
wall-following	75.12	447.26	206.66	457	200.73	172.75	87.98	525.26	292.24	542.77	260.06	226.96
waveform	235.29	341.47	263.81	393.89	218.79	171.81	345.63	422.09	393.9	497.19	334.64	274.58
waveform-noise	248.63	394.91	283.88	450.01	252.12	178.42	364.23	477.08	430.04	541.41	371.84	285.08
wine	6.59	8.7	7.12	11.46	6.78	5.45	8.63	10.95	8.85	12.68	9.05	7.46
wine-quality-red	145.57	216.6	160.1	214.39	129.81	128.37	187.58	205.59	199.63	196.07	182.78	180.84
wine-quality-white	469.5	680.6	513.65	660.92	417.41	409.4	596.95	643.95	635.97	596.16	573.59	569.44
yeast	154.13	205.48	159.04	180.27	139.59	140.98	173.4	158.29	176.88	145.17	175.34	176.23

Continued on next page

Table 4.7 – Continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
zoo	7.05	8.55	7.22	8.02	6.7	6.63	7.21	9	7.41	8.29	7.29	7.32
Average of Mean Nodes	135.98	183.01	152.44	172.21	132.99	119.17	166.91	198.03	185.75	187.43	171.87	153.07

Here, OM denotes oocytes_merluccius, OT denotes oocytes_trisopterus.

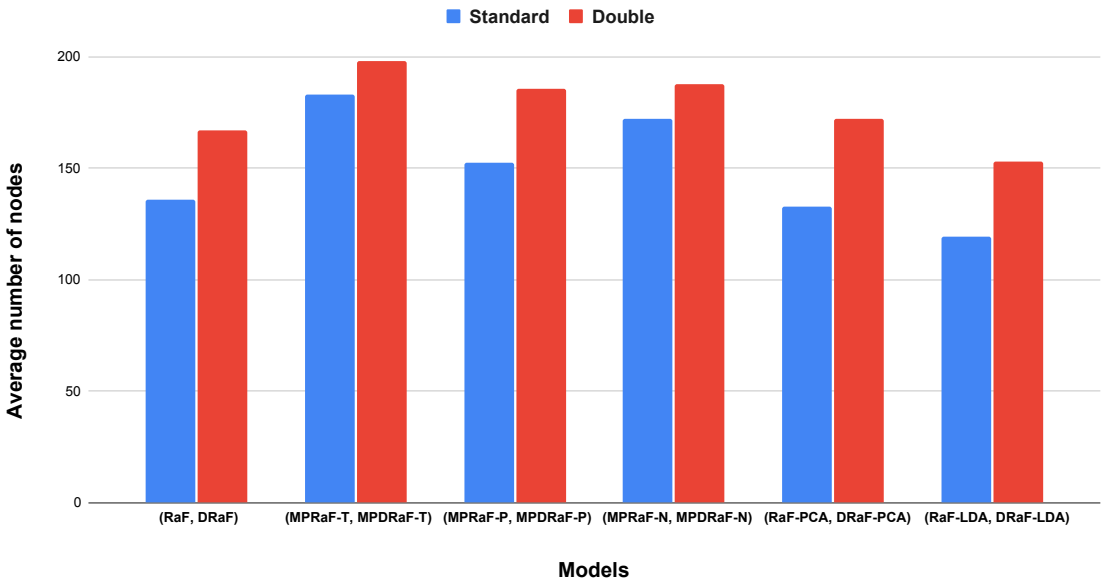


Figure 4.3: Mean node analysis of the proposed oblique and rotation double random forest models and the baseline models.

4.5 Diversity error diagrams

In this section, we analyze the existing baseline models and the proposed oblique and rotation double random forest in terms of "diversity" among the individual decision tree classifiers and their classification accuracy or error. To visualise both the models in terms of these measures, visualization approach known as kappa-error diversity diagrams are used [175]. Kappa error diagrams use $2D$ plot for visualisation of individual accuracy and diversity of the members of the base learner. For L number of base learners (here, decision trees) in an ensemble, a diagram is shown as a scatter plot of $L(L-1)/2$ points with each point corresponding to a pair of classifiers being analysed. The x-coordinate represents the diversity among the pair of base learners, also known as Kappa (κ) coefficient and the y-coordinate represents the average error of the pair of base learners. Kappa gives the level of agreement between the two base learners and while correcting for chance. For T target labels of given dataset, κ is defined on the $T \times T$ coincidence matrix C of two classifiers. Each entry in the c_{ij} represents the proportion of the testing data which one classifier predicted as k^{th} class while the other base learner classifies it as the j^{th} class. Kappa coefficient κ represents the

level of agreement between the two classifiers and is given as follows:

$$\kappa = \frac{p_r(a) - p_r(e)}{1 - p_r(e)} \tag{4.3}$$

where $p_r(a)$ is the observed agreement between the two classifiers i.e. probability that both classifiers predicted the same label and the $p_r(e)$ is the hypothetical probability of agreement by chance. Mathematically,

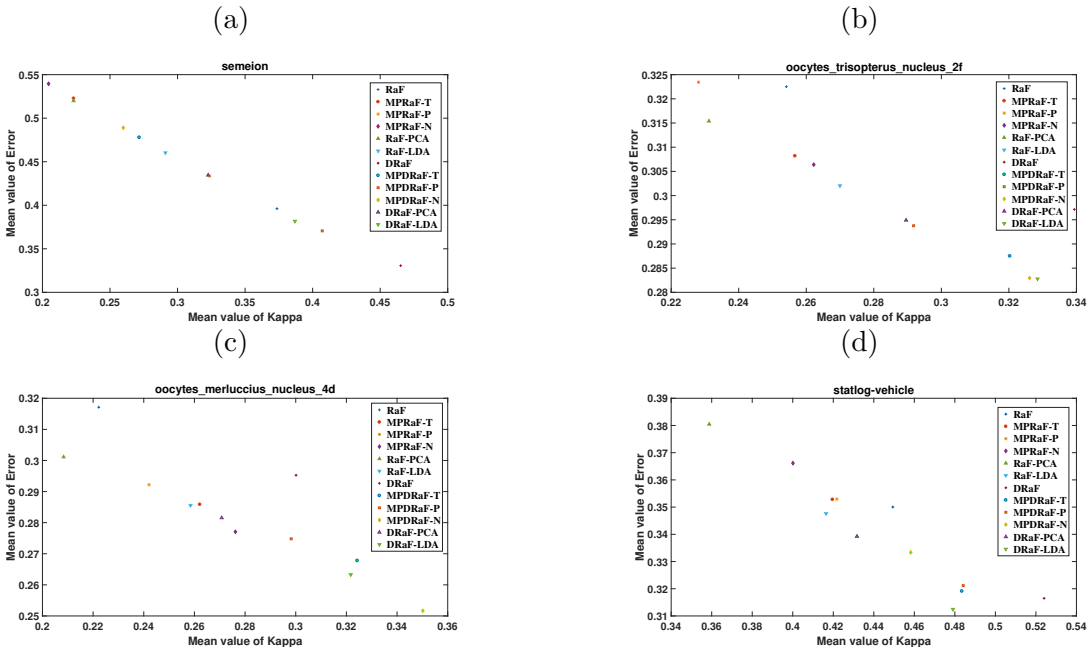
$$p_r(a) = \sum_i c_{ii}, \tag{4.4}$$

$$p_r(e) = \sum_k \left[\Big(\sum_i m_{ki} \Big) \Big(\sum_j m_{jk} \Big) \right]. \tag{4.5}$$

If the two decision trees are in complete agreement, then the kappa coefficient (κ) is 1 and the two trees are identical. If the trees are independent, then the kappa coefficient (κ) is 0. As mentioned above, we evaluate $L(L-1)/2$ pairs of kappa coefficients. Also, averaged error of the individual classifiers $E_{i,j} = (E_i + E_j)/2$. The smaller κ value indicates better diversity or low correlation while as the smaller averaged error E represents the more accurate or better strength classifier. The most desirable pair of classifiers is the one in the bottom left corner of Figure 4.4.

Figure 4.4 plots the kappa error diagram for some datasets. The ensemble size is 50, hence, 1225 dots in each plot. All the classification models are trained on the training data samples and κ-error diagrams are plotted based on the performance of the classification models on the testing samples (in some diagrams the axis are adjusted for better view). Figure 4.4 represents the centroid of the scatter points for each classification model corresponding to the semeion, oocytes_merluccius_nucleus_4d, oocytes_trisopterus_nucleus_2f and statlog-vehicle datasets. From the given plots, different models of the random forest possess different characteristics. Figure 4.4(a) plot shows that MPRaF-N is the most diverse classifier (least mean value of kappa) and DRaF is the most accurate classifier (least mean value of error). However, DRaF-LDA ensemble classifiers possess the best overall generalization performance on this dataset. From the plot, one can see that the proposed DRaF-LDA have the better combination of diversity and error. Similarly in other datasets, the models with better combination results in better performance.

Figure 4.4: Centroid of Kappa error diagrams on different datasets.



4.6 Analysis of computational complexity

Here, we evaluate the computational complexity of the classifiers. Without assuming any structure of decision trees, we focus on the complexity involved at a given node. Let a given node receives m number of samples with n number of features. In axis parallel splits, the optimum threshold is chosen based on some impurity criteria via ranking of each feature. Despite the complexity of the gini impurity, the complexity of the search involved in optimal split is $O(nm\ log\ m)$ [303]. For MPSVM based oblique decision trees, the computational complexity of generalized problem is $O(n^3)$ [173]. In decision trees wherein the feature transformations (PCA and LDA) are used for projecting the input features, additional computational time is involved for calculating the projection matrix. The complexity of the PCA is $O(mn \times min(m,n) + n^3)$ [142] while as for LDA the complexity is $O(mn^2)$ [44]. MPSVM based decision tree ensembles are faster as compared to the standard ensemble models. The reason is that in most of the cases, particularly for the nodes near the root, MPSVM method is faster compared to the exhaustive search. The training time of the proposed DRaF-PCA and DRaF-LDA is more as compared to the RaF-PCA and RaF-LDA, respectively, due to the reason that the bootstrapping at each non-leaf node of the proposed DRaF-PCA and DRaF-LDA leads to more number of unique samples to be sent down the tree resulting in

more deeper decision trees. The average training time of each classification model is given in Table 4.3.

4.7 Bias variance analysis

In this section, we discuss the bias-variance analysis of the ensemble models. Bias-variance analysis is the main reason for the success of ensemble models. The concept of bias-variance is well known in the regression problems for the squared loss functions [76]. However, this analysis is inappropriate as the labels of the classes are categorical. Thus, it is not feasible to transplant the decomposition of error in regression problems to classification problems. In classification problems, several studies have provided the ways to decompose the classification error into bias-variance terms [69, 121, 140]. Each of these studies provide some insight into the models performance.

In this study, we consider $0-1$ loss function to analyse the performance of the models [139]. Let D and Y be spaces representing the input and output, respectively. Suppose $|D|$ represents the cardinality of D and $|Y|$ represents the cardinality of Y. Also, let $d \in D$ and $y \in Y$ be the element its label respectively. The conditional probability distribution of target f is $P(Y_F = y_F|d)$ where Y_F is the Y-valued random variable. Then for a single test data sample:

$$E(C) = \sum_{d} P(D)[(bias_d)^2 + \sigma_d^2 + variance_d], \tag{4.6}$$

where

$$(bias_d)^2 = \frac{1}{2}\sum_{y \in Y}[P(Y_F = y) - P(Y_H = y)]^2, \tag{4.7}$$

$$variance_d = \frac{1}{2}[1 - \sum_{y \in Y} P(Y_H = y)^2], \tag{4.8}$$

$$\sigma_d^2 = \frac{1}{2}[1 - \sum_{y \in Y} P(Y_H = y)^2]. \tag{4.9}$$

Here, $(bias_d)^2$ and $variance_d$ are calculated are each model and for each dataset. $(bias_d)^2$ is abbreviated as $bias_d$. Theoretically, the error should be decomposed into squared bias, variance and noise (also known as irreducible error). However, given the real-world tasks

Methods	Average Rank	Average Rank Difference	Significance
(RaF, DRaF)	$(7, 3.02)$	3.98	Yes
(MPRaF-T, MPDRaF-T)	$(8.52, 4.37)$	4.15	Yes
(MPRaF-P, MPDRaF-P)	$(7.96, 3.55)$	4.41	Yes
(MPRaF-N, MPDRaF-N)	$(9.73, 5.9)$	3.83	Yes
(RaF-PCA, DRaF-PCA)	$(10.03, 5.5)$	4.53	Yes
(RaF-LDA, DRaF-LDA)	$(8.42, 4.01)$	4.41	Yes

$\chi_F^2 = 615.0719, F_F = 103.0950, q_{0.05} = 3.2680$. The two models are significantly different if the average ranks of the two models differ at least by the critical difference, $CD = 1.5149$.

Table 4.8: Significant difference among the standard and double variants of the ensembles of decision trees based on the bias analysis.

wherein the true underlying probability distribution is unknown, estimation of noise is difficult task. In commonly used approach, the noise is generally aggregated into bias and variance or the only bias term as the noise in invariant across the learning models for a given task and hence not a significant factor for the comparative analysis of the algorithms. Table 4.10 gives the bias-variance values for each model corresponding to the 121 datasets. In most of the cases the double variant ensembles of decision trees have the best bias-variance values compared to the standard ensembles of the decision trees.

We evaluate the bias-variance of the classification models via statistical tests. In this test, the lower value of bias/variance gets lower rank and vice versa. The analysis of the results for bias and variance are given in Table 4.8 and Table 4.9, respectively. From the given tables, it is clear that the double variants of the random forest achieve lower average rank compared to the standard variants of random forest for both bias and variance performance. Hence, the proposed double variants of random forest show better bias-variance results compared to the standard variants of the random forest. Moreover, the all the proposed variants of the double random forest are significantly better compared to the standard variants of the random forest.

Methods	Average Rank	Average Rank Difference	Significance
(RaF, DRaF)	$(6.61, 1.93)$	4.68	Yes
(MPRaF-T, MPDRaF-T)	$(8.96, 4.03)$	4.93	Yes
(MPRaF-P, MPDRaF-P)	$(8.13, 3.36)$	4.77	Yes
(MPRaF-N, MPDRaF-N)	$(9.88, 5.64)$	4.24	Yes
(RaF-PCA, DRaF-PCA)	$(10.68, 5.64)$	5.04	Yes
(RaF-LDA, DRaF-LDA)	$(8.93, 4.2)$	4.73	Yes

$\chi_F^2 = 809.8335, F_F = 186.4664, q_{0.05} = 3.2680$. The two models are significantly different if the average ranks of the two models differ at least by the critical difference, $CD = 1.5149$.

Table 4.9: Significant difference among the standard and double variants of the ensembles of decision trees based on the variance analysis.

Table 4.10: Bias variance analysis of RaF [23], MPRaF-T [303], MPRaF-P [303], MPRaF-N [303], RaF-PCA [305], RaF-LDA [305], DRaF [91], MPDRaF-T, MPDRaF-P, MPDRaF-N, DRaF-PCA and DRaF-LDA classification models.

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias	Bias	Bias	Bias	Bias	Bias	Bias	Bias	Bias	Bias	Bias	Bias
	Var.	Var.	Var.	Var.	Var.	Var.	Var.	Var.	Var.	Var.	Var.	Var.
abalone	422.19	421.95	422.16	409.26	424.42	420.62	415.77	411.11	417.33	399.97	415.42	413.34
	220.38	224.11	226.61	206.18	228.12	225.96	202.89	204.21	214.17	190.48	215.17	214.64
acute-inflammation	2.21	1.48	2.14	1.99	1.72	1.74	1.81	1.44	1.64	2.07	1.56	1.53
	1.84	1.31	1.84	1.74	1.52	1.48	1.53	1.25	1.4	1.83	1.31	1.32
acute-nephritis	1.14	0.86	1.14	1.53	0.74	0.72	0.53	0.86	0.71	1.18	0.58	0.53
	0.99	0.8	1.02	1.41	0.66	0.66	0.49	0.8	0.66	1.09	0.54	0.5
adult	3032.87	3261.33	3192.13	3669.3	3341.57	3802.34	2798.83	2993.5	2967.71	3432.11	3175.48	3539.3
	1423.89	1552.82	1555.3	1908.44	1718.95	2123.73	1110.31	1241.92	1287.36	1646.2	1553.7	1886.05
annealing	56.94	76.64	54.6	80.44	62.27	63.7	52.45	78.1	58.66	80.84	63.66	64.95
	24.71	23.7	25.63	21.81	33.16	34.4	23.87	21.74	24.56	22.91	32.91	34.78
arrhythmia	48.16	55.61	47.75	54.86	53.39	53.65	45.18	52.32	45.5	50.95	49.35	50.5
	30.55	35.78	30.37	35.2	34.99	35.65	26.16	30.44	26.79	28.58	29.82	32.12
audiology-std	11.27	14.81	10.96	15.59	15.04	14.59	11	14.25	11.12	15.71	14.47	14.25

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	6.9	10.19	6.73	10.3	10.22	9.98	5.79	9.92	6	10.03	9.82	9.83
balance-scale	34.84	30.63	31.25	30.69	32.12	30.29	32.76	28.41	28.69	27.46	30.59	27.93
	22.44	20.42	20.46	20.4	21.08	19.99	17.75	17.64	17.18	16.99	17.8	16.84
balloons	1.42	1.42	1.41	1.35	1.43	1.33	1.26	1.19	1.27	1.31	1.11	1.14
	0.8	0.82	0.82	0.83	0.84	0.77	0.66	0.67	0.65	0.75	0.65	0.66
bank	158.45	171.54	164.99	165.29	168.34	166.13	146.42	151.74	152.84	148.85	155.18	153.4
	79.65	85.26	83.55	76.1	87.21	84.7	67.42	64.8	70.35	59.45	75	73.58
blood	52.18	50.54	51.52	50.64	52.67	52.28	47.96	44.88	48.01	46.45	49.84	49.51
	19.29	18.11	19.18	19.06	19.8	19.75	11.12	10.79	11.85	12.1	13.16	12.94
breast-cancer	24.34	24.42	24.36	23.73	24.75	24.51	21.58	22.02	21.87	20.98	22.56	22.05
	11.77	11.85	11.78	11.64	12.35	12.23	8.8	9.28	9	8.51	9.87	9.6
breast-cancer-wisc	9.94	9.37	8.91	9.06	9.99	9.29	9.31	8.12	8.07	8.01	8.91	8.75
	5.92	5.6	5.22	4.98	6.03	5.35	5.25	4.5	4.5	4.38	4.99	4.89

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
breast-cancer-wisc-diag	11.58 6.82	9.6 5.79	10.32 6.28	10.57 6.57	13.13 8.5	10.09 6.47	10.13 5.65	8.5 4.93	8.84 5.16	8.47 4.91	11.23 7.17	8.46 5.07
breast-cancer-wisc-prog	16.76 8.9	15.82 8.65	16.42 8.63	16.32 8.73	17.21 9.24	16.64 8.94	15.98 8.11	14.64 7.55	15.08 7.98	14.58 7.41	15.74 8.29	16.16 8.43
breast-tissue	9.3 5.28	9.45 5.81	9.77 5.84	9.95 6.32	10.14 6.47	9.38 5.91	8.33 4.45	9.1 4.87	8.89 4.85	9.2 5.37	8.96 5.18	8.71 4.9
car	53.62 40.16	66.26 49.16	57.3 43.61	82.54 52.95	57.16 43.64	51.75 39.4	47.24 35.52	60.48 44.72	48.71 37.55	76.95 46.94	46.33 35.5	42.38 33.03
cardiotocography-10clases	131.33 89.8	173.39 125.16	137.05 96.6	184.31 132.33	167.74 124.55	146.51 106.39	109.53 70.11	152.42 106.29	116.46 76.81	171.16 121.06	142.7 102.89	125.47 86.74
cardiotocography-3clases	57.33	73.66	62.12	78.34	73.06	69.02	47.51	62.48	51.28	68.54	62.08	57.57

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	37.12	47.85	40.8	50.63	47.74	45.32	28.08	38.44	31.89	42.3	38.38	36.22
chess-krvk	3034.05	3362.72	3124.59	3727.55	2940.16	2994.79	2827.31	3183.27	2887.18	3648.43	2662.02	2724.99
	2059.91	2326.91	2178.24	2416.27	2080.07	2093.71	1831.25	2132.02	1957.51	2302.68	1831.15	1848.16
chess-krvkp	80.61	120.16	83.73	155.37	102.15	90.43	58.94	98.25	64.73	138.61	79.92	73.46
	61.57	90.21	64.48	113.06	77.99	69.75	44.63	74.42	50.15	102.97	61.82	57.61
congressional-voting	44.37	44.87	45.13	44.68	46.33	45.13	42.7	42.79	42.63	43.49	43.21	42.87
	16.03	14.65	15.34	13.96	18.56	17.31	3.25	3.8	3.4	6.23	5.88	4.65
conn-bench-sonar-mines-rocks	17.83	17.62	18.6	18	19.54	17.24	16.33	16.47	16.78	16.82	17.18	15.5
	10.12	10.15	10.55	10.38	11.02	9.62	9.37	9.19	9.77	9.4	9.89	8.92
conn-bench-vowel-deterding	116.57	95.58	112.38	124.23	121.65	109.1	76.15	60.54	70.67	101.47	78.76	68.95
	105.4	92.13	105.05	117.06	113.98	102.3	71.13	61.26	69.17	99.09	78.68	68.13

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
connect-4	3786.62 1936.33	4350.55 1787.5	3949.26 1932.43	4354.23 1384.57	4067.8 2150.1	3991.2 2055.04	3372.93 1546.41	4027.03 1353.85	3573.11 1556.88	4160.29 1148.35	3628 1772.59	3581.14 1673.69
contrac	178 91.6	186.21 96.67	182.14 94.68	187 98.58	181.85 94.61	181.28 93.99	175.47 75.21	182.93 82	178.55 80.1	180.44 87.81	179.98 80.62	179.21 80.66
credit-approval	35.78 19.87	38.42 21.82	38.92 22.12	37.32 20.93	39.4 22.69	38.34 21.69	33.89 17.87	34.78 17.97	34.9 19.05	35.15 18.67	37.12 20.57	35.42 19.26
cylinder-bands	44.55 25.53	49.18 27.29	46.05 26.23	48.93 26.87	47.79 27.08	46.45 26.09	40.46 23.47	47.67 26.66	42.35 24.57	46.89 26.55	44.39 25.48	43.68 24.91
dermatology	9.98 8.43	10.83 9.44	9.38 7.97	11.47 9.97	13.46 11.84	11.9 10.41	7.52 5.96	9.16 7.71	7.64 6.03	9.71 8.33	10.38 9.03	9.35 8.01
echocardiogram	9.15 4.68	8.86 4.69	9.1 4.87	9.38 4.96	9.44 5.04	9.19 4.95	8.03 3.72	7.87 3.77	8.28 4.15	8.04 3.79	8.78 4.38	8.11 4.03
ecoli	18.23 10.84	19.64 11.67	18.28 10.89	20.9 13.62	19.4 12.08	18.85 11.52	15.41 7.47	17.79 9.45	15.86 8.21	19.05 11.62	16.9 8.97	16.7 8.87
energy-y1	16.04	24.28	17.47	23.22	20.83	17.95	12.73	19.82	13.21	20.27	15.18	13.49

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	8.06	14.04	9.48	13.82	12.6	10.71	5.35	9.79	5.86	10.98	8.25	7.04
energy-y2	23.08	26.22	23.5	27.07	24.83	23.9	22.61	23.18	21.97	26.04	23.07	22.62
	10.96	13.58	11.08	14.46	12.98	12.13	7.2	9.18	7.58	12.87	9.78	8.9
fertility	4.31	4.93	4.55	4.56	4.73	4.57	3.67	3.86	3.91	3.7	3.9	3.95
	1.94	2.47	2.06	2.06	2.27	2.09	1.12	1.31	1.38	1.08	1.43	1.47
flags	21.28	25.33	22.76	25.36	24.8	25	19.63	24.64	20.77	24.74	23.38	23.12
	14.72	16.74	15.71	17.01	16.75	17.04	12.68	15.6	13.78	15.65	15.34	15.29
glass	19.46	20.25	19.09	20.78	20.5	20.51	16.67	18.63	17.23	19.99	18.63	18.57
	12.45	12.55	12.2	12.87	13.1	12.98	9.7	10.73	10.14	11.45	11.37	11.18
haberman-survival	25.47	25.79	25.48	25.05	26.5	26.28	25.04	24.18	24.92	24.09	25.51	25.24
	11.01	11.38	11	10.92	11.64	11.61	7.83	8.31	8.69	8.14	8.86	9.16
hayes-roth	5.92	9.26	8.03	10.3	8.09	8.58	4.57	7.82	5.92	9.55	6.86	7.24
	3.43	5.99	5.21	6.48	5.45	5.44	2.37	4.98	3.56	5.76	4.52	4.61
heart-cleveland	32.34	32.18	31.56	32.01	32.1	31.78	31.08	30.46	30.52	30.2	31	31.05

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	19.01	19.59	18.77	19.27	19.53	19.35	16.28	16.73	16.64	16.38	17.69	17.68
heart-hungarian	17.72	17.95	18.06	17.68	18.68	17.26	15.66	15.94	16.59	15.6	16.65	16.44
	9.05	9.54	9.56	9.41	10.05	9.19	7.06	7.67	7.92	7.52	8.34	8.12
heart-switzerland	17.46	17.32	17.33	17.45	17.62	17.52	17.16	17.5	17.15	17.44	17.58	17.24
	9.94	10.18	10.02	10.2	10.31	10.23	9.04	9.31	8.9	9.19	9.78	9.66
heart-va	29.78	29.76	29.87	29.57	29.55	29.72	29.61	30.17	29.46	29.33	29.59	29.76
	17.39	17.41	17.36	17.5	17.44	17.49	15.93	16.11	16.02	16.16	16.35	16.34
hepatitis	8.91	9.79	9.41	9.57	9.48	9.05	8.82	9.19	8.89	9.06	9.04	8.96
	4.91	5.54	5.26	5.5	5.45	5.18	4.71	5.11	4.81	5.04	4.92	4.88
hill-valley	293.58	272.38	281	268.79	281.38	273.96	290.39	264.8	274.21	261.01	271.99	267.73
	135.21	143.5	144.06	137.19	145	138.52	132.08	141.33	142.91	131.17	142.78	137.4
horse-colic	18.23	20.99	19.67	20.73	22.68	22.33	16.51	19.16	18.7	20.2	21.22	22.24
	10.34	11.9	11.27	11.99	12.96	12.69	8.78	10.51	10.31	11.22	11.82	12.32
ilpd-indian-liver	48.37	48.88	49.08	50.59	48.99	49.22	46.68	46.75	46.17	47.11	46.68	47.47
	23.29	23.55	23.93	24.73	24.41	24.09	20.95	20.99	21.37	21.25	22.14	22.45

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
image-segmentation	380.59	466.36	424.88	623.46	457.6	468.05	298.42	393.75	338.4	588.04	378.66	392.93
	295.25	377.07	348.3	520.25	375.6	398.97	227.28	315.87	270.57	490.47	315.18	339.2
ionosphere	13.06	14.02	14.4	14.62	14.84	13.84	10.89	12.21	12.45	12.83	13.34	12.7
	7.97	9.19	9.42	9.32	9.71	8.95	6.4	8	8.19	8.4	8.69	8
iris	2.02	2.11	1.77	2.02	2.77	1.55	1.95	1.52	1.59	1.39	2.28	1.54
	1.08	1.33	1.13	1.42	2.02	0.88	0.84	0.78	0.86	0.81	1.46	0.88
led-display	79.9	84.24	80.89	90.26	78.53	79.14	78.8	83.67	80.07	90.53	77.87	78.75
	38.74	45.57	40.56	55.39	38.18	38.21	28.95	35.1	31.03	49.77	26.71	29.18
lenses	1.93	1.85	1.77	2.06	2.13	1.96	1.64	1.8	1.94	1.57	1.9	1.79
	1.32	1.28	1.17	1.31	1.39	1.33	1.02	1.15	1.17	1.05	1.2	1.14
letter	875.1	1169.82	1122.19	1298.45	1182.43	1040.52	684.66	988.71	890.24	1159.74	940.64	816.4
	764.88	1047.64	1001.99	1159.21	1055.78	932.48	594	890.77	802.66	1044.75	852.52	743.86
libras	39.03	34	39.06	36.55	38.12	37.04	30.97	27.25	31.19	30.93	29.77	28.47
	29.95	27.67	31.79	29.51	30.76	29.78	23.66	22.48	25.89	25.5	24.2	23.26

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
low-res-spect	21.91 14.56	21.87 14.63	23.23 16.17	26.01 18.25	25.24 18.37	23.35 16.53	20.2 12.33	20.31 12.7	21.19 13.87	23.63 16.04	21.42 14.45	19.92 13.34
lung-cancer	4.26 2.41	4.17 2.45	4.33 2.38	4.34 2.48	4.44 2.55	4.39 2.51	3.98 2.14	4.31 2.45	4.06 2.13	4.23 2.43	4.2 2.43	4.26 2.39
lymphography	29.68 6.14	29.16 7.17	29.39 6.35	28.59 7.45	29.28 7.16	30 6.57	30.62 4.95	29.47 6.21	30.51 5.02	29.1 6.48	29.72 6.17	29.84 5.89
magic	966.3 505.97	1008.49 533.54	1008.6 537.45	1004.6 534.69	996.78 542.88	976.48 522.07	898 448	916.77 448.36	941.86 481.86	896.96 437.62	920.04 486.41	910.56 474.26
mammographic	54.54 23.19	53.46 21.7	57.6 24.86	52.97 21.59	56.38 23.8	57.43 24.96	52.55 14.66	50.14 15.31	52.68 16.96	50.64 17.48	52.91 16.02	53.6 16.37
miniboone	4121.21 2422.28	3859.11 2162.95	4216.63 2504.94	3803.82 2050.18	4551.71 2765.54	4175.43 2493.98	3858.53 2254.23	3504.21 1914.7	3965.63 2351.13	3479.97 1853.21	4206.87 2542.82	3913.31 2335.05
molec-biol-promoter	9.42 5.6	10.33 5.9	9.78 5.76	10.49 6.01	11.02 6.1	9.94 5.79	8.33 5.11	9.79 5.66	9.17 5.46	9.58 5.6	10.18 5.87	9.34 5.53

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
molec-biol-splice	181.66	290.79	213.17	297.37	306.32	244.77	145.8	261.31	180.77	267.25	269.7	214.22
	138.02	201.95	159.45	205.03	212.57	177.08	109.67	182.44	135.65	187.34	189.93	154.95
monks-1	184.7	188.87	190.15	193.74	190.17	189.89	179.06	181.61	184.26	191.03	188.55	182.31
	52.93	87.64	69.57	80.62	77.94	75.1	47.2	75.36	67.2	83.85	68.68	66.87
monks-2	153.7	157.68	155.9	162.69	163.26	160.78	148.77	150.26	151.06	154.24	154.44	154.19
	30.91	44.29	38.65	54.25	51.21	47.28	16.71	26.01	23.34	35.95	33.11	30.23
monks-3	194.02	172.46	188.45	184.22	188.32	192.86	200.1	169.54	186.25	188.42	181.59	190.13
	57.03	73.83	61.84	74.99	76.46	74.71	41.86	66.7	57.22	63.91	59.58	66.64
mushroom	3.18	6.41	3.36	27.9	9.02	7.78	2.21	4.61	2.23	22.76	9.97	4.58
	2.96	6.01	3.19	26.48	8.51	7.39	2.11	4.41	2.16	21.69	9.63	4.32
musk-1	36.02	37.26	39.74	38.63	41.52	38.78	30.47	32.76	34.15	34.71	35.13	34.42
	21.41	22.39	23.37	23.26	24.97	22.56	17.96	19.98	20.56	21.08	21.42	20.02
musk-2	115.57	146.03	149.62	159.35	175.74	157.47	74.74	114.79	115.99	122.98	125.67	120.31
	74.39	94.63	96.47	104.31	117.35	103.39	48.23	70.91	72.46	75.96	82.9	78.09
nursery	2030.12	1962.02	2014.15	1920.23	2004.34	2021.52	2058.12	1985.2	2046.31	1939.8	2050.82	2058

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	161.98	238.37	178.16	324.16	198.46	171.28	132.12	209.75	140.91	314.17	150.08	132.66
OM_nucleus_4d	81.66	74.48	76.05	72.5	78.9	74.33	76.11	68.82	71.97	65.53	74.18	68
	42.31	40.62	41.99	37.79	44.34	42.02	38.87	37.36	39.42	33.5	41.79	37.88
OM_states_2f	34.01	34.28	34.58	35.49	37.72	32.94	31.06	30.45	31.41	31.39	33.8	29.99
	19.83	19.9	20.75	21.12	23.55	20.02	17.22	17.43	18.2	18.1	20.25	17.49
OT_nucleus_2f	74.58	72.35	73.19	71.7	73.64	69.96	68.6	66.13	68.06	66.72	68.15	64.49
	40.36	40.87	41.87	40.8	41.99	39.73	36.25	37.3	38.51	37.36	39.01	35.9
OT_states_5b	39.31	37.43	39.03	40.35	42.03	37.07	34.83	31.38	34.19	33.6	35.96	31.24
	23.74	23.71	24.49	25.67	27.71	24.22	20.25	19.51	21.27	20.76	23.37	20.16
optical	347.14	471.68	351.36	932.06	525.93	854.82	293.25	424.12	301.73	941.64	473.15	746.27
	311.63	429.48	317.48	724.69	482.75	698.24	265.14	391.18	272.43	729.29	443.37	643.02
ozone	30.14	30.46	30.64	29.86	35	33.74	28.36	26.49	27.92	24.97	30.81	30.67
	15	16.01	15.45	15.3	19.46	18.5	13.23	11.87	13.25	10.05	15.93	15.73
page-blocks	61.43	65.49	62.18	71.87	67.36	65.3	54.61	56.4	55.72	65.77	58.29	56.73
	34.01	36.47	35.83	42.22	39.94	38.53	24.99	27.94	27.59	36.35	30.5	29.61

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
parkinsons	9.43	9.15	9.53	9.47	10.49	9.77	7.89	7.72	8.06	8.14	8.98	8.24
	5.16	5.53	5.5	5.56	6.1	6.06	4.3	4.73	4.78	4.91	5.11	4.88
pendigits	432.84	349.16	411.69	461.57	442.92	392.22	371.49	296.03	353.34	415.33	390.87	340.92
	334.57	280.17	325.47	379.57	375	322.83	281.79	233.79	278.04	338.84	328.11	276.92
pima	62.07	63.41	63.57	64.1	64.12	63	60.56	60.16	61.79	59.96	62.36	61.22
	30.1	31.52	31.52	31.97	31.71	31.04	27.11	27.75	29.4	28.04	29.64	28.43
pittsburg-bridges-MATERIAL	4.32	3.93	4.51	4	4.56	4.33	3.65	3.28	3.64	3.27	3.69	3.34
	2.45	2.53	2.76	2.51	2.96	2.71	1.56	1.71	1.58	1.7	2.03	1.62
pittsburg-bridges-REL-L	9.87	9.77	9.89	9.77	10.05	10.19	9.04	8.99	9.16	8.98	9.28	9.05
	5.66	5.84	5.88	5.88	5.92	6.01	4.78	4.92	4.75	5	5.09	4.96
pittsburg-bridges-SPAN	9.73	9.39	9.7	9.57	9.61	9.55	9.36	8.72	9.08	8.87	9.26	9.19

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	5.02	4.94	5	5.26	5.25	5.11	3.85	3.88	3.9	3.98	4.27	4.19
pittsburg-bridges-T-OR-D	3.8	4.23	3.76	3.95	4.11	4.08	3.13	3.13	3.18	3.29	3.6	3.51
	1.89	2.16	1.87	2	2.12	2.03	1.18	1.05	1.19	1.24	1.51	1.45
pittsburg-bridges-TYPE	11.06	11.16	11.43	11.38	11.39	11.44	10.23	9.92	10.1	10.34	10.45	10.18
	6.52	7.2	6.83	7.13	7.19	7.09	4.72	5.46	5.01	5.61	5.67	5.47
planning	18.94	18.72	19.12	18.83	19.18	18.78	18.43	17.87	17.96	16.92	18.36	18.3
	8.34	8.66	8.61	8.58	8.95	8.67	7.44	7.37	7.76	6.7	7.82	7.91
plant-margin	207.32	213.59	225.57	224.65	214.8	220.86	185.65	203.73	208.48	217.32	189.21	191.76
	165.7	168.02	175.83	174.65	169.32	173.42	150.92	162.44	166.6	170.34	153.6	156.81
plant-shape	208.39	200.15	218.05	213.26	207.39	210.35	187.89	186.74	199.96	201.49	181.49	179.02
	151.83	151.54	165.52	162.81	157.17	161.98	131.84	142.14	152.98	154.54	136.95	138.34
plant-texture	200.01	204.59	217.09	218.8	209.02	215.93	172.62	191.51	197.74	208.96	183.31	187.08
	161.71	165.22	173.28	173.36	167.88	172.22	142.61	156.98	161.7	167.84	150.72	155.16

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
post-operative	9.14	9.2	9.15	9.45	9.25	9.94	7.73	8.02	7.79	8.18	8.5	8.18
	3.58	3.53	3.47	3.78	3.54	3.92	1.69	1.94	1.81	2.07	2.28	2.12
primary-tumor	41.25	43.21	41.32	43.23	42.36	41.73	38.62	40.04	38.99	40.65	39.66	39.88
	26.24	28.4	26.7	29.11	27.75	27.34	20.98	23.19	21.99	25.14	23.35	24.13
ringnorm	256.67	306.79	304.97	304.71	263.7	276.46	213.73	285.23	281	275.01	220.26	231.47
	175.88	168.89	170.48	169.11	191.62	198.7	141.68	163.36	161.3	159.62	160.21	167
seeds	6.82	5.48	6.61	6.06	7.12	5.76	5.84	5.11	5.38	5.35	6.1	5.2
	3.77	3.24	3.87	3.42	4.32	3.24	3.1	2.75	3.02	2.81	3.47	2.92
semeion	126.93	163.75	139.3	167.68	162.94	144.83	106.49	150.44	118.57	154.24	138.95	121.39
	108.94	135.69	118.51	138.09	134.91	123.66	91.81	126.4	102.45	129.21	118.36	106.29
soybean	85.7	125.43	86.86	169.31	145.14	168.43	65.71	118.51	66.27	167.24	129.05	160.1
	70.2	109.22	71.29	140.08	126.74	141.95	49.24	104.9	50.77	139.17	115.31	137.44
spambase	135.46	161.62	143.8	179.6	156.4	149.42	113.87	140.49	121.61	156.76	131.21	128.29
	83.45	104.31	90.36	117.42	101.55	95.58	66.39	87.83	73.1	99.73	82.8	80.36
spect	74.55	78.75	75.48	81.66	81.93	79.26	70.13	75.54	71.13	76.95	77.76	76.27

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	35.82	37.59	36.46	37.47	39.29	36.74	28.16	29.38	28.01	29.5	32.29	32.7
spectf	37.38	44.39	42.58	28.7	50.73	60.26	33.26	39.24	36.48	23	45.36	47.13
	22.54	28.01	26.71	15.24	32.63	37.78	18.92	24.42	21.91	9.16	28.63	29.82
statlog-australian-credit	72.45	76.02	74.31	72.89	75.22	75.5	73.17	75.86	75.58	75.18	75.13	75.66
	35.72	36.79	36.61	35.72	36.67	36.65	35.08	36.52	36.82	35.68	35.9	36.08
statlog-german-credit	82.59	86.24	83.99	85.12	85.53	83.99	76.37	80.24	79.5	80.47	80.94	79.28
	43.05	44.83	43.79	43.73	44.69	43.93	37.2	38.65	39.35	38.63	40.84	39.73
statlog-heart	17.4	17.22	17.67	17.85	17.62	17	16.33	15.9	16.24	16.6	16.49	16.19
	9.81	9.82	10	10.02	9.98	9.37	8.18	8.39	8.55	8.77	9.02	8.69
statlog-image	42.92	53.07	46.48	67.83	51.13	42.32	30.38	41.77	34.76	53.36	36.73	30.74
	32.37	41.95	36.4	54.87	41.79	33.69	22.42	32.39	26.73	42.74	29.24	23.59
statlog-landsat	330.98	325.12	327.45	343.26	351.89	334.89	301.92	298.76	298.83	317.8	316.78	306.57
	215.27	205.78	210.92	213.63	232.29	215.5	191.15	186.25	188.09	193.7	205.19	194.98

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
statlog-shuttle	14.63	62.49	26.19	169.51	59.99	77.66	8.09	69.82	21.17	159.44	55.01	53.09
	10.99	51.74	21.32	145.12	54.77	70.31	6.06	58.72	16.2	134.38	51.13	48.18
statlog-vehicle	68.72	68.8	67.81	70.73	72.12	67.28	62.6	64.3	62.07	65.36	65.54	60.85
	41.53	43.94	43.43	45.73	48.4	44.34	35.74	39.18	38.16	41.03	42.64	39.28
steel-plates	158.94	168.25	167.32	176.17	174.42	169.94	143.81	157.85	150.39	163.29	156.3	156.1
	102.96	111.27	110.38	115.95	116.19	112.44	89.65	101.28	97.22	107.58	102.84	100.54
synthetic-control	24.33	20.63	24.89	28.59	27.24	21.6	18.64	16.33	19.11	22.57	20.27	17.68
	19.91	17.23	20.25	23.2	21.5	17.6	15.27	13.42	15.76	18.25	16.36	14.55
teaching	18.3	18.48	18.38	18.72	18.77	18.5	17.13	17.95	17.34	18.54	17.15	17.22
	9.98	10.03	10.09	10.11	10.23	9.88	7.15	7.57	7.47	8.44	7.54	7.61
thyroid	96.4	235.15	103.95	764.03	327.21	473.01	85.6	217.64	84.42	714.99	347.09	433.88
	64.68	152.4	71.55	587.65	277.09	396.42	54.73	137.61	53.7	558.06	307.19	369.07
tic-tac-toe	47.05	49.97	49.88	53.77	51	50.03	37.48	39.81	40.57	47.01	39.98	39.69
	33.66	35.23	35.05	36.53	35.73	35.22	27.83	29.77	30.07	33.41	29.71	29.61
titanic	119.26	120.25	119.76	122.17	118.52	118.64	118.31	119.18	118.84	121.15	118.62	117.76

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
	13.4	15.83	14.35	17.7	12.64	12.65	7	11.63	8.12	16.16	8.8	7.18
trains	0.75	0.69	0.75	0.73	0.83	0.83	0.59	0.71	0.55	0.75	0.62	0.69
	0.44	0.44	0.45	0.44	0.47	0.47	0.38	0.42	0.35	0.45	0.4	0.43
twonorm	322.13	174.14	179.14	175.13	196.7	174.59	300.72	154.18	163.55	152.77	173.41	156.68
	228.03	123.14	128.34	123.87	140.69	123.59	212.1	107.54	114.87	106.02	122.73	109.67
vertebral-column-2clases	18.23	17.96	18.07	18.48	19.07	17.83	17.34	16.3	16.55	16.32	17.53	16.7
	9.64	9.9	9.85	9.84	10.22	9.72	8.37	8.37	8.7	8.23	8.81	8.62
vertebral-column-3clases	18.24	19.38	18.97	19.38	19.65	18.88	16.69	17.36	17.24	17.18	18.4	16.97
	10.39	11.58	11.49	11.48	11.94	11.35	9	9.59	9.7	9.74	10.76	9.47
wall-following	53.4	263.94	147.78	279.8	217.02	205.71	34.78	209.91	116.49	228.6	161.01	150.81
	44.51	195.14	121.8	208.18	171.9	161.23	28.59	152.31	94.1	169.25	125.62	116.59
waveform	338.3	330.24	329.75	329.77	342.18	304.07	325.41	310.76	310.6	307.43	324.3	292.42
	200.61	195.75	195.75	196.45	207.55	177.82	185.88	180.54	181.45	179.29	192.39	166.3

Continued on next page

Table 4.10 – continued from previous page

Datasets	RaF	MPRaF-T	MPRaF-P	MPRaF-N	RaF-PCA	RaF-LDA	DRaF	MPDRaF-T*	MPDRaF-P*	MPDRaF-N*	DRaF-PCA*	DRaF-LDA*
	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.	Bias Var.
waveform-noise	365.29	396.83	373.3	402.08	402.48	346.01	346.85	370.86	356.66	371.62	373.77	323.59
	229.31	250.02	234.66	254.27	259.66	215.75	212.51	231	220.16	232.58	237.8	199.23
wine	4.7	4.16	4.63	4.9	5.94	4.44	3.9	3.97	3.44	4.12	4.59	3.28
	3.7	3.32	3.72	3.85	4.53	3.6	3.03	3.04	2.72	3.2	3.55	2.66
wine-quality-red	170.47	172.33	170.37	172.22	170.12	169.33	152.43	159.31	152.86	160.16	150.89	148.68
	98.56	102.89	99.29	101.99	101.58	101.11	82.49	89.32	85.4	89.21	85.87	83.8
wine-quality-white	538.74	544.35	539.74	550.6	540.62	538.74	474.95	498.56	478.36	509.12	470.59	466.43
	323.98	333.7	327.45	333.24	333.04	330.1	268.98	290.93	274.29	295.78	274.59	268.83
yeast	171.29	178.95	172.73	179.09	173.81	173.4	164.8	172.82	165.84	173.92	166.39	165.68
	99.69	109.78	102.5	109.3	104.37	103.6	85.57	97.63	88.63	99.42	91.59	91.27
zoo	2.46	2.62	2.37	3.11	3.11	2.94	1.62	2.02	1.49	2.16	2.22	1.93
	2.11	2.35	2.03	2.65	2.78	2.65	1.37	1.68	1.26	1.85	2.01	1.72

Here, OM denotes oocytes_merluccius, OT denotes oocytes_trisopterus.

Var. denotes the variance.

4.8 Summary

In this chapter, we propose two approaches for generating the double random forest models. In the first model, we propose oblique double random forest ensemble models and in the second approach, we propose rotation based double random forest ensemble models. In oblique double random forest models, the splitting hyperplane at each non-leaf node is generated via MPSVM. This leads to the incorporation of geometric structure and hence, leads to better generalization performance. As the decision tree grows, the problem of sample size may arise. Hence, we use Tikhonov regularisation, axis parallel split regularisation null space regularisation for generating decision trees to full depth. In rotation based double random forest models, we used two transformations- principal component analysis and linear discriminant analysis, on randomly chosen feature subspace at each non-leaf node. Rotations on different random subspace features lead to more diverse decision tree ensembles and better generalization performance. Unlike standard random forest where the bootstrap aggregation is used at root node only, the proposed oblique and rotation double random forest use bootstrap aggregation at each non-terminal node for choosing the best split and then the original samples are sent down the decision trees. The proposed double variants of the ensemble of decision trees results in bigger trees compared to the standard variants of the ensemble of decision trees. Experimental results and the statistical analysis show the efficacy of the proposed oblique and rotation double random forest ensemble models over standard baseline classifiers.

Chapter 5

Minimum variance embedded RVFL and co-trained RVFL network

In this chapter, we present two variants of random vector functional link network (RVFL) model known as minimum variance embedded RVFL and co-trained RVFL model. The minimum variance embedded RVFL includes total variance minimization based RVFL (Total-Var-RVFL) and intraclass variance minimization based RVFL (Class-Var-RVFL). Total-Var-RVFL exploits the training data dispersion by minimizing the total variance while as Class-Var-RVFL minimizes the intraclass variance of the training data. To further improve the performance of the RVFL, we ensemble the projections from randomised networks and l_1 norm autoencoder based projections. The coRVFL trains two RVFL models jointly such that each RVFL model is constructed with different feature projection matrix and hence, shows better generalization performance. We use randomly projected features and sparse-l_1 norm autoencoder based features to train the proposed coRVFL model. The proposed formulations are given as follows:

5.1 Minimum variance embedded RVFL network

Motivated by the simplicity and better generalization performance of the RVFL model [203] and minimum variance extreme learning machine [120], we propose novel multiclass classifiers known as total variance minimization based random vector functional link network (Total-Var-RVFL) and intraclass variance minimization based random vector functional link network (Class-Var-RVFL) to optimize the output layer weights via minimization of both

training data dispersion and norm of output layer weights.

5.1.1 Proposed formulation

In this section, we discuss variance embedded variants of random vector functional link network. The proposed Total-Var-RVFL and Class-Var-RVFL involve two step learning process wherein the first step requires the generation of enhanced features via randomized feature mapping. The second step involves incorporation of within class scatter matrix for generating the optimal output weights. We exploit the training data dispersion for generating the optimal classifier. Unlike RVFL network, the proposed approach minimizes the output layer weights norm and the dispersion of the training data in the projected feature space and original feature space.

$$\begin{aligned} \underset{\beta}{Min} \quad & \frac{1}{2}Tr(\beta^T S\beta) + \frac{c_2}{2}\left\|\beta\right\|_2^2 + \frac{c_1}{2}\left\|\xi\right\|_2^2 \\ s.t. \quad & H\beta - Y = \xi, \end{aligned} \tag{5.1}$$

where first term minimizes the variance of the training samples and second term of the objective function is the regularisation term and H is combination of original and randomized features. The variance term S is given as S_T for total variance and S_w for within class variance both are defined as follows:

The within class variance of the training data is given as:

$$S_w = \sum_k \sum_{i \in C_k} (x_i - m_k)(x_i - m_k)^T, \tag{5.2}$$

where m_k is the mean of k^{th} class training data, C_k represents the k^{th} class, $x_i \in \mathbb{R}^{(d+J)}$ is the training data sample.

The total variance of the training data is given as:

$$S_T = \sum_{i=1}^{M} (x_i - \mu)(x_i - \mu)^T, \tag{5.3}$$

where μ is the mean of the training data samples.

Substituting the constraints in the objective function of (5.1) and taking the gradient

with respect to β, we have

$$L = S\beta + c_2\beta + c_1 H^T(H\beta - Y). \tag{5.4}$$

Setting the gradient to zero, we get the optimal output layer weights as

$$\beta = (\frac{1}{c_1}S + \frac{c_2}{c_1}I + H^T H)^{-1} H^T Y, \tag{5.5}$$

where I is an identity matrix of appropriate dimension. Testing data sample x is assigned the class based on the maximum probability as $f(x) = h(x)\beta$.

5.1.2 Experiments

In this section, we analyze the experimental results of the given baseline models. We evaluate the performance of the given methods on the datasets available from the UCI repository [60]. We followed the same experimental setup and naming convention as given in the experimental study [62]. We used grid search approach to tune the optimal parameters corresponding to different given classification methods. The parameters are chosen from the range given as: $c_1 = [10^{-6}, 10^{-5}, \cdots, 10^5, 10^6]$, $c_2 = [10^{-6}, 10^{-5}, \cdots, 10^5, 10^6]$ and number of hidden neurons=$2 : 20 : 300$. We used relu activation function in the hidden layer. All experiments are performed on Windows-10 platform with 3-GB RAM and MATLAB-2019b.

Based on the previous discussion, variants of the proposed method include Total variance minimization based random vector functional link network (Total-Var-RVFL) and intraclass variance minimization based random vector functional link network (Class-Var-RVFL). The experimental results obtained by the given classification methods on different datasets is given in Table 5.1.

From the Table 5.1, the average accuracy of the classification models extreme learning machine (ELM), minimum variance ELM (MVELM), minimum class variance ELM (MCVELM), random vector functional link network (RVFL), proposed Total-Var-RVFL and proposed Class-Var-RVFL are $75.83, 73.01, 74.97, 75.88, 77.94$, and 77.88, respectively. One can observe that the proposed Total-Var-RVFL and proposed Class-Var-RVFL classification models achieved better average accuracy in comparison to given baseline models. We use Friedman test [53] to analyze the statistical significance of the classification models.

We assign rank R_i^j to the i^{th} classifier of the p classifiers on the j^{th} dataset with the best performing classifier getting the lower rank. Based on this evaluation, the average rank of the classification models ELM, MVELM, MCVELM, RVFL, proposed Total-Var-RVFL and proposed Class-Var-RVFL are $4.25, 4.22, 3.22, 4.06, 2.67$, and 2.58, respectively. Under the null-hypothesis, all classifiers are performing equally and hence their average ranks are equal. After simple calculations, we have $\chi_F^2 = 15.4707, F_F = 3.5289$. Hence, we reject the Null hypothesis. We use Nemenyi posthoc test to evaluate the significant difference among the models. After simple calculations at $\alpha = 0.05$, one can see that Nemenyi test fails to detect the significant difference among the models, however, the proposed models achieved higher average accuracy and lower average rank as compared to the baseline models.

Table 5.1: Performance of ELM, MVELM, MCVELM, RVFL, proposed Total-Var-RVFL and proposed Class-Var-RVFL.

Dataset	ELM [113] (Acc., Time(s)) (N, c)	MVELM [120] (Acc., Time(s)) (N, c_2, c_1)	MCVELM [119] (Acc., Time(s)) (N, c_2, c_1)	RVFL [203] (Acc., Time(s)) (N, c)	Total-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)	Class-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)
arrhythmia	(68.81, 0.0108) (162, 0.1)	(71.46, 0.0387) (202, 1000, 100)	(70.35, 0.07) (282, 100, 10)	(69.47, 0.0145) (102, 1)	(69.47, 0.0536) (142, 1000, 100)	(69.47, 0.067) (142, 1000, 100)
balloons	(75, 0.0002) (102, 0.000001)	(50, 0.0003) (2, 0.000001, 1000)	(50, 0.0005) (22, 0.000001, 100)	(75, 0.0003) (2, 0.1)	(81.25, 0.0006) (22, 0.000001, 0.001)	(81.25, 0.0006) (22, 0.000001, 0.001)
blood	(78.61, 0.0073)	(76.47, 0.0261)	(79.28, 0.0031)	(77.81, 0.0218)	(79.01, 0.0174)	(79.01, 0.0188)

Continued on next page

Table 5.1 – continued from previous page

Dataset	ELM [113] (Acc., Time(s)) (N, c)	MVELM [120] (Acc., Time(s)) (N, c_2, c_1)	MCVELM [119] (Acc., Time(s)) (N, c_2, c_1)	RVFL [203] (Acc., Time(s)) (N, c)	Total-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)	Class-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)
breast-cancer	(102, 10000) (69.37, 0.0041) (122, 1000000)	(202, 0.001, 100) (69.37, 0.0153) (162, 100, 10000)	(62, 0.000001, 100000) (70.77, 0.0044) (82, 0.00001, 1)	(262, 100000) (68.66, 0.004) (262, 100)	(222, 0.1, 1000) (69.37, 0.0028) (102, 0.0001, 1)	(222, 0.1, 1000) (69.37, 0.0031) (102, 0.0001, 1)
breast-cancer-wisc-prog	(81.12, 0.0023) (82, 1)	(83.16, 0.0104) (142, 0.1, 0.1)	(80.61, 0.0272) (242, 0.01, 0.01)	(76.53, 0.0021) (142, 100)	(82.14, 0.0071) (202, 0.1, 0.1)	(81.12, 0.0013) (2, 10, 10000)
energy-y1	(85.16, 0.006) (162, 10000)	(87.63, 0.0227) (202, 0.00001, 1000)	(86.72, 0.0092) (122, 0.0001, 100000)	(86.33, 0.024) (282, 1000000)	(87.11, 0.0078) (122, 0.000001, 1000)	(87.37, 0.0316) (282, 0.000001, 100)
glass	(68.87, 0.0023) (242, 10)	(62.26, 0.0153) (182, 1, 1000)	(60.38, 0.0045) (82, 0.00001, 100000)	(66.04, 0.0026) (222, 1000)	(64.15, 0.0046) (142, 100, 100000)	(66.04, 0.002) (62, 0.00001, 1000)
heart-hungarian	(82.53, 0.0047) (182, 100)	(83.22, 0.0014) (42, 0.000001, 100000)	(84.93, 0.0037) (62, 10, 1)	(81.16, 0.0038) (162, 100000)	(84.25, 0.0031) (82, 1, 1000)	(83.56, 0.0144) (282, 1000, 100000)

Continued on next page

Table 5.1 – continued from previous page

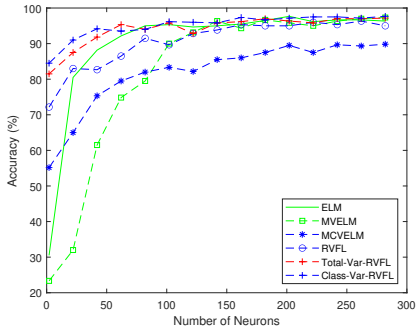
Dataset	ELM [113] (Acc., Time(s)) (N, c)	MVELM [120] (Acc., Time(s)) (N, c_2, c_1)	MCVELM [119] (Acc., Time(s)) (N, c_2, c_1)	RVFL [203] (Acc., Time(s)) (N, c)	Total-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)	Class-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)
hepatitis	(78.21, 0.0007) (62, 1000)	(73.72, 0.0206) (222, 0.01, 10)	(78.21, 0.0336) (282, 0.000001, 0.00001)	(78.85, 0.0012) (82, 1000)	(82.05, 0.01) (282, 10, 1000)	(80.13, 0.0057) (162, 0.1, 10)
hill-valley	(56.44, 0.0282) (282, 10000)	(68.48, 0.0256) (162, 0.0001, 0.1)	(56.77, 0.0338) (202, 100, 1000000)	(58.25, 0.0075) (62, 10000)	(59.74, 0.0094) (22, 0.0001, 10)	(65.51, 0.009) (42, 0.00001, 0.1)
horse-colic	(66.18, 0.0078) (262, 0.1)	(61.76, 0.0207) (202, 1, 0.1)	(63.24, 0.0299) (242, 10, 1)	(64.71, 0.0012) (22, 1)	(66.18, 0.0011) (2, 0.01, 0.1)	(66.18, 0.0169) (222, 100000, 10000)
lenses	(70.83, 0.0002) (22, 0.1)	(79.17, 0.0014) (62, 10000, 1000)	(70.83, 0.0008) (42, 10, 1)	(75, 0.0003) (42, 0.1)	(83.33, 0.0021) (122, 1000, 100)	(83.33, 0.0017) (122, 100, 10)
mammographic	(81.35, 0.012) (182, 100000)	(81.67, 0.0254) (182, 0.000001, 0.01)	(81.56, 0.0075) (102, 0.0001, 10000)	(82.29, 0.0145) (182, 1000)	(82.5, 0.0036) (62, 0.1, 1000000)	(81.98, 0.0023) (22, 0.001, 10000)

Continued on next page

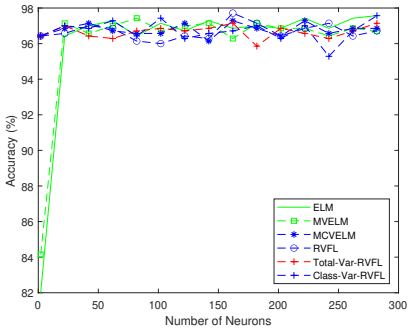
Table 5.1 – continued from previous page

Dataset	ELM [113] (Acc., Time(s)) (N, c)	MVELM [120] (Acc., Time(s)) (N, c_2, c_1)	MCVELM [119] (Acc., Time(s)) (N, c_2, c_1)	RVFL [203] (Acc., Time(s)) (N, c)	Total-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)	Class-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)
monks-3	(71.99, 0.0042) (122, 100000)	(72.92, 0.0084) (142, 0.001, 100)	(83.1, 0.0317) (262, 0.001, 0.01)	(75.93, 0.0008) (42, 1000000)	(82.41, 0.0063) (222, 0.001, 0.01)	(75.46, 0.0057) (202, 0.00001, 0.01)
planning	(71.11, 0.0002) (2, 1000000)	(65.56, 0.009) (142, 0.00001, 0.01)	(71.67, 0.0129) (182, 0.000001, 0.001)	(71.11, 0.0004) (2, 0.000001)	(71.11, 0.0005) (2, 0.000001, 0.000001)	(71.11, 0.0006) (2, 0.000001, 0.000001)
seeds	(94.23, 0.0005) (42, 10)	(65.38, 0.0007) (22, 0.000001, 0.00001)	(95.19, 0.0038) (82, 0.000001, 1000000)	(94.23, 0.0022) (162, 100000)	(94.71, 0.0008) (22, 0.00001, 1000000)	(95.67, 0.0008) (22, 0.00001, 100)
statlog-german-credit	(77.1, 0.005) (102, 100)	(75.8, 0.0144) (142, 0.1, 100000)	(78.1, 0.0295) (202, 0.000001, 0.01)	(77.6, 0.0299) (262, 10)	(77.2, 0.0096) (102, 0.01, 100000)	(77.6, 0.0068) (62, 1, 10000)
statlog-heart	(88.06, 0.0021) (122, 0.1)	(86.19, 0.0012) (22, 1000, 1000)	(87.69, 0.0032) (82, 10, 1)	(86.94, 0.0016) (82, 0.1)	(86.94, 0.0029) (82, 1, 0.1)	(87.69, 0.0025) (82, 100, 10)

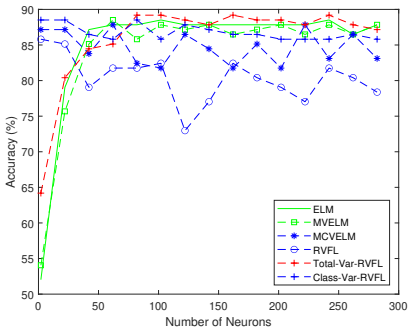
Continued on next page



(a) Synthetic-control



(b) Breast-cancer-wisc



(c) Lymphography

Figure 5.1: Effect of enhanced features on the performance of the classification models.

Table 5.1 – continued from previous page

Dataset	ELM [113] (Acc., Time(s)) (N, c)	MVELM [120] (Acc., Time(s)) (N, c_2, c_1)	MCVELM [119] (Acc., Time(s)) (N, c_2, c_1)	RVFL [203] (Acc., Time(s)) (N, c)	Total-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)	Class-Var-RVFL (Acc., Time(s)) (N, c_2, c_1)
Average-Accuracy	75.83	73.01	74.97	75.88	77.94	77.88
Average-Rank	4.25	4.22	3.22	4.06	2.67	2.58

5.1.3 Analysis of the number of hidden neurons

In this subsection, we analyze the significance of the number of hidden neurons on the generalization performance of the classification models. From Figure 5.1(a) one can see that as the number of hidden neurons increases, the generalization performance of the given

baseline methods also increases. Also, the generalization performance of the proposed Class-Var-RVFL is better than the given baseline models. In Figure 5.1(b), it is visible that the generalization performance of the proposed models is almost consistent with varying number of hidden neurons. In Figure 5.1(c), it is evident that RVFL model initially performs better and its performance decreases with the increase of hidden neurons. However, the performance of the proposed Total-Var-RVFL is better than other baseline models.

Given the above analysis, it is imperative that the number of hidden neurons should be chosen properly to obtain better generalization performance across different classification datasets.

5.2 Co-trained RVFL network

Motivated by the random feature projection matrix in RVFL [203], unsupervised feature learning based RVFL [311] and positively correlated KRR [306], we propose co-trained random vector functional link network (coRVFL). The proposed coRVFL model is a positively correlated ensemble model. It is effective as each RVFL model is constructed with different feature projection matrix and hence it is unlikely that the outcome of each model will give wrong prediction for a particular data sample. The positive correlation forces the output of bad learner to be as close as possible to the good learner in the ensemble.

5.2.1 Proposed formulation

In this section, we discuss the formulation and computational complexity of the proposed co-trained random vector functional link network. The proposed co-trained random vector functional link network (coRVFL) trains two RVFL models jointly. In the proposed coRVFL model, both random feature projection matrix and the sparse l_1-norm autoencoder features are used to learn the network robustly. Hence, the proposed coRVFL model can be regarded as marriage of random feature weighted RVFL model and unsupervised feature learning based RVFL model. The latent feature representation of the input for random and pre-

trained network is given as follows:

$$F_1 = \begin{bmatrix} g(x_1.w_1 + b_1) & \dots & g(x_1.w_L + b_L) \\ g(x_2.w_1 + b_1) & \dots & g(x_2.w_L + b_L) \\ \vdots & \ddots & \vdots \\ g(x_N.w_1 + b_1) & \dots & g(x_N.w_L + b_L) \end{bmatrix}, \tag{5.6}$$

where each $w_i \in \mathbb{R}^n$, $W = [w_1, w_2, w_3, \dots, w_L]^T$ is the random matrix and b_i the biases initialized from a suitable range.

Let

$$H_1 = [F_1, X]. \tag{5.7}$$

and

$$F_2 = \begin{bmatrix} g(x_1.\beta_1 + d_1) & \dots & g(x_1.\beta_L + d_L) \\ g(x_2.\beta_1 + d_1) & \dots & g(x_2.\beta_L + d_L) \\ \vdots & \ddots & \vdots \\ g(x_N.\beta_1 + d_1) & \dots & g(x_N.\beta_L + d_L) \end{bmatrix}, \tag{5.8}$$

where each $\beta_i \in \mathbb{R}^n$, $\beta = [\beta_1, \beta_2, \beta_3, \dots, \beta_L]^t$ is the pre-trained weight matrix obtained by l_1-norm autoencoder and d_i are the biases.

Also assume

$$H_2 = [F_2, X]. \tag{5.9}$$

The optimization problem of the proposed co-trained RVFL model is as follows:

$$\min_{\alpha} \ \frac{1}{2}\sum_{i=1}^{2}(||H_i\alpha_i - Y||^2 + c_i||\alpha_i||_F^2) + \frac{c_3}{2}\sum_{i,j=1}^{2}||H_i\alpha_i - H_j\alpha_j||^2, \tag{5.10}$$

where α_i are the output layer weights, for $i = 1, 2$ and $||\cdot||_F$ denotes the Frobenius norm. The first part of the objective function ensures that each model predicts the output label as closely as possible, the second term is the regularization term to control the model complexity, and the third term regularizes the pairwise models to minimize the disagreements between

the models. The objective function leads to better performance as it is unlikely for the two models to agree on an incorrect label as each is trained on a different latent feature representation of the input data. When $c_3 = 0$, the proposed coRVFL model reduces to the training of two independent RVFL models.

Taking derivative of the objective function with respect to α_i and setting this gradient to zero, we get the closed form solution as:

$$\begin{aligned}\alpha_1 = &((1+c_3)H_1^tH_1 + c_1I + c_3^2H_1^tH_2((1+c_3)H_2^tH_2 + c_2I)^{-1}H_2^tH_1)^{-1} \\ &(H_1^tY + c_3H_1^tH_2((1+c_3)H_2^tH_2 + c_2I)^{-1}H_2^tY)\end{aligned} \tag{5.11}$$

and

$$\begin{aligned}\alpha_2 = &((1+c_3)H_2^tH_2 + c_2I + c_3^2H_2^tH_1((1+c_3)H_1^tH_1 + c_1I)^{-1}H_1^tH_2)^{-1} \\ &(H_2^tY + c_3H_2^tH_1((1+c_3)H_1^tH_1 + c_1I)^{-1}H_1^tY).\end{aligned} \tag{5.12}$$

Let

$$G_i = (1+c_3)H_i^tH_i + c_iI, \tag{5.13}$$

where $i = 1, 2$, then we have

$$\alpha_1 = (G_1 + c_3^2H_1^tH_2G_2^{-1}H_2^tH_1)^{-1}(H_1^tY + c_3H_1^tH_2G_2^{-1}H_2^tY), \tag{5.14}$$

$$\alpha_2 = (G_2 + c_3^2H_2^tH_1G_1^{-1}H_1^tH_2)^{-1}(H_2^tY + c_3H_2^tH_1G_1^{-1}H_1^tY). \tag{5.15}$$

The outcome of the proposed coRVFL is combined in two ways, maximum and average of probability outcomes, to get the final output label and the coRVFL models are accordingly named as coRVFL-max and coRVFL-avg. In coRVFL-max, the maximum of the probability outcomes is taken as the final outcome of the model while as in coRVFL-avg the average of the probability outcomes is taken as final outcome of the model.

5.2.2 Experiments

In this section, experimental analysis is performed to demonstrate the performance of the proposed coRVFL model. All the experiments are conducted on MATLAB-2017*b* on a

PC with 8 GB RAM Intel(R) Core(TM) $i7-6700$ CPU 3.40GHz 3.41 GHz.

For comparison of the baseline models (here, standard RVFL [203] and sparse pre-trained RVFL (Sp-RVFL) [311]) and the proposed models, we have taken datasets from UCI repository [60] and some real-world fisheries datasets [81]. The details of the datasets used is given in Table 5.3. For comparison, we have followed the same setting for data partitions (training, validation and testing) as given in [62]. Stratified sampling is done to divide the data into training and testing datasets (with approx. 50% of the available patterns). We used grid search approach to obtain the optimal parameters, for number of hidden neurons $L = 3 : 203$ with 20 step size and $c_i = (\frac{1}{2})^{\lambda}$ with λ chosen from the range $-5 : 14$.

Table 5.4 gives the classification accuracies of the proposed coRVFL models and the baseline models. One can see, the proposed coRVFL models (coRVFL-max and coRVFL-avg) achieve highest average accuracies. However, comparing classifiers with average accuracy is susceptible as lower performance in one dataset can be compensated by higher in other one. Hence, we followed [62, 306], and use Friedman ranking method for performance evaluation. In Friedman testing, each model is assigned a rank with the higher performance model being assigned a lower rank and lower performance model being assigned higher rank. From Table 5.4, one can see that the top ranked model is the proposed coRVFL-avg model followed by coRVFL-max model. This shows that the proposed method leads to better performance compared to the given baseline models.

5.2.3 Computational complexity analysis

Let M be the number of samples in a dataset wherein each feature $x_i \in \mathbb{R}^n$ and L be the number of hidden neurons in each architecture. Also, let $d = n + L$. In standard RVFL and Sp-RVFL models, the primal space involves the inversion of matrix of size d while as in dual space the matrix inversion of size M is involved. Let d be the number of features in each model of co-trained RVFL. In proposed co-trained RVFL models, two matrix inversions of size d are involved in calculation of each α. By standard matrix inversion procedure, the complexity of inverting the matrix of size d is $O(d^3)$. Thus, the proposed co-trained RVFL model involves more computation compared to standard RVFL and Sp-RVFL models. The additional complexity comes at the cost of training two models in a cooperative manner.

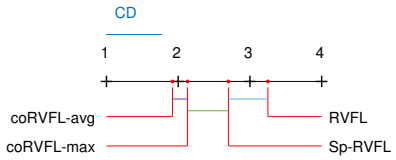


Figure 5.2: Statistical difference between the RVFL, Sp-RVFL, and the proposed coRVFL models based on pairwise Nemenyi test.

5.2.4 Friedman test

To analyse the performance of the classification models statistically, we use Friedman test [71, 72]. With simple calculation, we get $\chi_F^2 = 19.3644, F_F = 7.9502$. With $z = 4$ and $D = 30$, F_F is distributed with 3 and 87 degrees of freedom. At $\alpha = 0.10$, the critical value of $F_F(3, 87) = 2.15$. Hence, we reject the null hypothesis, i.e., significant difference exists among the compared models. With Nemenyi test, critical difference

$$\begin{aligned} CD =& q_\alpha \sqrt{\frac{z(z+1)}{6D}}, \\ =& 2.291 * \sqrt{4 * 5/(6 * 30)} = 0.7. \end{aligned} \tag{5.16}$$

From Figure 5.2, one can see that the proposed coRVFL-avg model is significantly better as compared to the RVFL and Sp-RVFL models. Also, coRVFL-max is significantly better as compared to the RVFL model. The statistical test is also reported in Table 5.2. The empty spaces denote that the significant difference doesn't exist among the methods given in the corresponding row and $s+$ denotes that the row method is significantly better as compared to the column method. Similarly, $s-$ denotes that the row method is statistically worse than the method in the corresponding column.

	RVFL	Sp-RVFL	coRVFL-max	coRVFL-avg
RVFL			s-	s-
Sp-RVFL				s-
coRVFL-max	s+			
coRVFL-avg	s+	s+		

Table 5.2: Statistical comparison of the classification models. Here s+ means row method is better as compared to the method in the corresponding column while as s- means that the row method is worse as compared to the method in the corresponding column. Empty entry means no significant difference exist among the methods given in the corresponding row and column of a cell.

Dataset	#pat.	#inputs	#classes	%Majority
bank	45211	17	2	88.5
cardiotocography-10clases	2126	21	10	27.2
cardiotocography-3clases	2126	21	3	77.8
chess-krvkp	3196	36	2	52.2
connect-4	67557	42	2	75.4
letter	20000	16	26	4.1
magic	19020	10	2	64.8
miniboone	130064	50	2	71.9
molec-biol-splice	3190	60	3	51.9
mushroom	8124	21	2	51.8
musk-2	6598	166	2	84.6
oocytes_merluccius_nucleus_4d	1022	41	2	68.7
oocytes_merluccius_states_2f	1022	25	3	67
ozone	2536	72	2	97.1
page-blocks	5473	10	5	89.8
plant-margin	1600	64	100	1
plant-shape	1600	64	100	1
plant-texture	1600	64	100	1
ringnorm	7400	20	2	50.5
semeion	1593	256	10	10.2
spambase	4601	57	2	60.6
statlog-german-credit	1000	24	2	70
statlog-image	2310	18	7	14.3
steel-plates	1941	27	7	34.7
twonorm	7400	20	2	50
wall-following	5456	24	4	40.4
waveform	5000	21	3	33.9
waveform-noise	5000	40	3	33.8
wine-quality-red	1599	11	6	42.6
wine-quality-white	4898	11	7	44.9

Naming convention is adapted from [62].

Table 5.3: Dataset details.

Dataset	RVFL	Sp-RVFL	coRVFL-max	coRVFL-avg
bank	89.8	89.23	**89.96**	**89.96**
cardiotocography-10clases	79	80.32	79.76	**80.79**
cardiotocography-3clases	90.82	91.15	91.24	**91.34**
chess-krvkp	96.87	96.65	**97.59**	**97.59**
connect-4	75.89	**77.23**	77.06	77.06
letter	79.71	82.56	83.46	**84.1**
magic	85.02	85.26	**85.69**	**85.69**
miniboone	90.36	89.28	**91.07**	**91.07**
molec-biol-splice	81.18	81.12	**82.59**	82.56
mushroom	**100**	**100**	**100**	**100**
musk-2	95.72	96.13	**96.15**	**96.15**
oocytes_merluccius_nucleus_4d	83.82	83.04	**84.8**	**84.8**
oocytes_merluccius_states_2f	**93.04**	92.35	92.16	91.37
ozone	**97.24**	97.08	97.12	97.12
page-blocks	95.76	96.11	96.13	**96.18**
plant-margin	78.31	**84.69**	83.56	83.44
plant-shape	56.69	59.81	61.19	**61.88**
plant-texture	80.63	**81.94**	81.06	81.75
ringnorm	95.64	96.32	**96.5**	**96.5**
semeion	88.32	**94.03**	93.15	91.21
spambase	91.48	92	**92.33**	**92.33**
statlog-german-credit	75.8	78.4	**79**	**79**
statlog-image	**94.06**	93.33	93.24	93.59
steel-plates	**75.62**	73.14	75.31	75.36
twonorm	97.77	**97.89**	97.76	97.76
wall-following	83.87	84.49	85.12	**85.41**
waveform	86.58	87.18	86.88	**86.92**
waveform-noise	86	**86.78**	**86.78**	86.76
wine-quality-red	60.13	61.63	60.63	**61.81**
wine-quality-white	55.58	55.25	**55.86**	55.47
Average Accuracy	84.69	85.48	85.77	**85.83**
Average Rank	3.25	2.7	2.13	**1.92**
Win-Tie-Loss	4-1-18	6-1-8	2-1-1	7-1-1

Table 5.4: Classification accuracy of RVFL, Sp-RVFL and the proposed coRVFL models.

Table 5.5: Comparison of RVFL, Sp-RVFL, and the proposed coRVFL models based on pairwise win-tie-loss: sign test.

	RVFL	Sp-RVFL	coRVFL-max	coRVFL-avg
Sp-RVFL	19-1-10			
coRVFL-max	24-1-5	18-2-10		
coRVFL-avg	23-1-6	21-1-8	11-13-6	

Here, a-b-c entry in each cell denotes that row method in each cell win a-times, ties b-times and loses c-times with respect to the method in the corresponding column.

Table 5.6: Significant difference between the RVFL, Sp-RVFL, and the proposed coRVFL models based on pairwise win-tie-loss: sign test.

	RVFL	Sp-RVFL	coRVFL-max	coRVFL-avg
RVFL			$s-$	$s-$
Sp-RVFL				$s-$
coRVFL-max	$s+$			
coRVFL-avg	$s+$	$s+$		

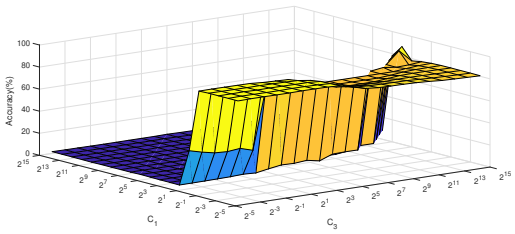
Here $s+$ means row method is better as compared to the column method while as $s-$ means that the row method is worse as compared to the column method in the corresponding cell. Empty entry means no significant difference exist among the methods given in the corresponding row and column of a cell.

5.2.5 Pairwise win-tie-loss: sign test

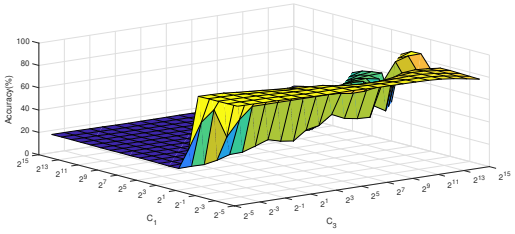
Another statistical method counts the number of wins a particular algorithm is the overall winner. In case of multiple algorithms, pairwise comparisons are given in a matrix form. Table 5.5 gives the number of wins, ties and losses in a matrix form. Table 5.6 gives the pairwise significant difference among the classification models. One can see that the proposed coRVFL-avg is better as compared to the given RVFL and Sp-RVFL classification models. Also, the proposed coRVFL-max is significantly better as compared to the standard RVFL model.

5.2.6 Parameter sensitivity

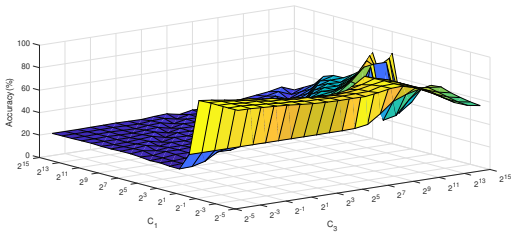
In this subsection, we analyze the effect of hyperparameters c_1, c_3 on the performance of the proposed coRVFL models. Figure 5.3(a) to Figure 5.3(d) show the effect of the parameters c_1 and c_3 on the performance of the proposed coRVFL-max models and Figure 5.3(e) to Figure 5.3(h) show the effect of the parameters on the performance of the proposed coRVFL-avg model. One can see that to obtain the optimal performance one needs to tune



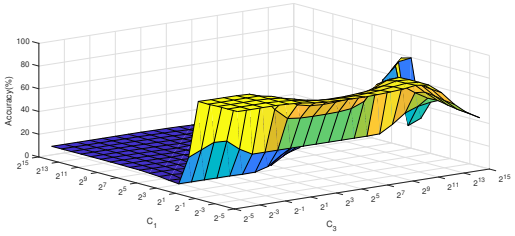
(a) Cardiotocography-3clases



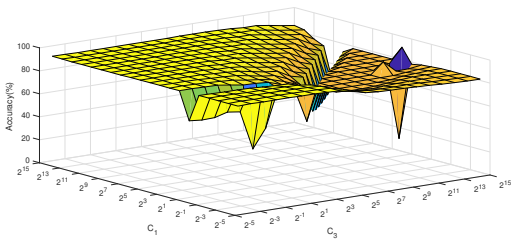
(b) Magic



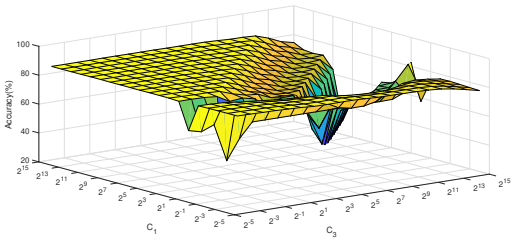
(c) Oocytes_merluccius_nucleus_4d



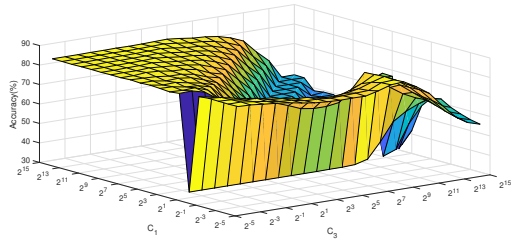
(d) Molec-biol-splice



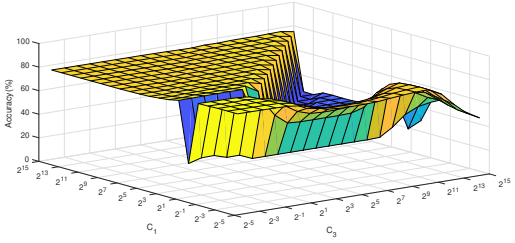
(e) Cardiotocography-3clases



(f) Magic



(g) Oocytes_merluccius_nucleus_4d



(h) Molec-biol-splice

Figure 5.3: Analysis of classification performance of the proposed coRVFL models with the varying hyperparameters c_1 and c_3 (Figures 5.3(a)-5.3(d) corresponds to the coRVFL-max while as the figures 5.3(e)-5.3(h) corresponds to the coRVFL-avg model).

these hyperparameters carefully.

5.3 Summary

In this chapter, we presented variance embedded RVFL and ensemble of RVFL, known as Co-Trained RVFL model. We present total variance minimization based random vector functional link network (Total-Var-RVFL) and intraclass variance minimization based random vector functional link network (Class-Var-RVFL). The proposed methods exploit the training data dispersion in the original feature space as well as the randomized feature projection space while optimizing the output layer weights. From the experimental analysis, one can see that incorporation of total variance and class variance improved the generalization performance of the proposed models. In comparison to given baseline models, the proposed Total-Var-RVFL and Class-Var-RVFL models achieved better average accuracy. Also, the average rank of the proposed Class-Var-RVFL is better than other baseline models except MCVELM.

In co-trained random vector functional link network (coRVFL), two RVFL models are trained jointly such that each RVFL model is predicting the target label as closely as possible. Since two RVFL models are trained on different feature representations and hence, it is unlikely for the outcome of two models to agree on a particular data sample resulting in forcing the less accurate model to be as close as possible to the more accurate model. Experimental results conducted on publicly available datasets show that the proposed coRVFL is better as compared to the baseline models. Furthermore, statistical analysis show that the proposed coRVFL-avg is statistically significantly better as compared to the baseline models.

This chapter presented the improvements over shallow RVFL models. Recently, there has been surge in the deep learning architecture, especially randomised based deep network architectures. In the next chapter, we present deep architecture of RVFL with learning using privileged information.

Chapter 6

Ensemble deep random vector functional link network using privileged information

In the previous chapter, we presented the improvements over shallow RVFL model. Recently, deep learning architectures have received quite a lot attention. In particular, non-iterative learning based deep randomized models have been proposed for the classification and regression problems. The deep randomised models especially random vector functional link network (RVFL) with direct links have proved to be successful. However, deep RVFL (dRVFL) and its ensemble models are trained only on normal samples. In this chapter, deep RVFL with privileged information and ensemble deep RVFL with privileged information are enabled to incorporate privileged information, however, the standard RVFL model and its deep models are unable to use privileged information. Privileged information based approach is commonly seen in human learning. To fill this gap, we have incorporated learning using privilaged information (LUPI) in dRVFL model, and propose dRVFL with LUPI framework called dRVFL+. In LUPI framework, half of the available features are used as normal features and rest as the privileged features. However, we propose a novel approach for generating the privileged information. We utilise different activation functions while processing the normal and privileged information in the proposed deep architectures. To the best of our knowledge, this is first time that a separate privileged information is generated.

The formulation of the proposed architectures is given as:

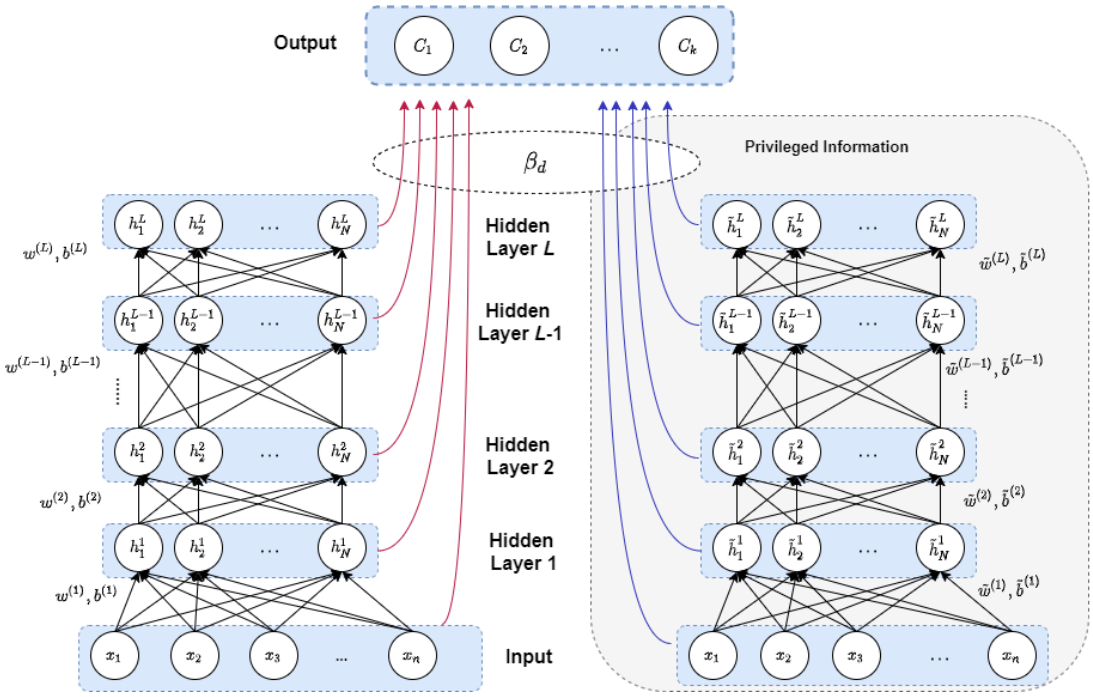


Figure 6.1: Deep random vector functional link network using privileged information.

6.1 Proposed deep RVFL+ and its ensemble

In this section, we introduce deep RVFL using privileged information (dRVFL+) and its ensemble deep RVFL (edRVFL) using privileged information known as edRVFL+.

6.1.1 Deep random vector functional link network using privileged information

Deep RVFL using privileged information (dRVFL+) is a deep architecture framework based on RVFL+ model. The architecture of the proposed dRVFL+ is given by Figure 6.1. From the given architecture, one can see that the proposed dRVFL+ consists of two components: normal information and the privileged information. The normal information is processed through L number of hidden layers, with each layer receiving the input from the previous layer. Here, we follow a rectangular structure wherein the number of nodes in each hidden layer is fixed to N. The weights and biases, given by w^i, b^i for $i = 1, 2, \ldots, L$ are generated randomly and kept fixed.

For mathematical simplicity, we skip the bias from the notation. The first hidden layer

output is given by

$$H^{(1)} = g(XW^{(1)}), \tag{6.1}$$

and the output of higher hidden layers ($i > 1$) is given as

$$H^{(i)} = g(H^{(i-1)}W^{(i)}), \tag{6.2}$$

where $W^{(1)} \in \mathbb{R}^{n \times N}$ and $W^{(i)} \in \mathbb{R}^{N \times N}$, and $g(\cdot)$ is the activation function.

The concatenated input data matrix to the output layer is given as

$$D = [H^{(1)}, H^{(2)}, \cdots, H^{(L-1)}, H^{(L)}, X]. \tag{6.3}$$

The privileged information is also generated by the L layer architecture wherein we input the original features. However, the privileged component of the given architecture uses separate set of weights and biases, denoted by $\tilde{w}^i, \tilde{b}^i$ for $i = 1, 2, \ldots, L$, which are generated randomly and kept fixed and different activation $f(\cdot)$ is used in the privileged component. Similar to normal component, we fix the rectangular component with N number of hidden nodes in each hidden layer of the L layer architecture.

The output of the first privileged hidden layer is given by

$$\tilde{H}^{(1)} = f(X\tilde{W}^{(1)}), \tag{6.4}$$

and the output of higher privileged hidden layers ($i > 1$) is given as

$$\tilde{H}^{(i)} = f(\tilde{H}^{(i-1)}\tilde{W}^{(i)}), \tag{6.5}$$

where $\tilde{W}^{(1)} \in \mathbb{R}^{n \times N}$ and $\tilde{W}^{(i)} \in \mathbb{R}^{N \times N}$, and $f(\cdot)$ is the activation function.

The concatenated input to the output layer is given as

$$\tilde{D} = [\tilde{H}^{(1)}, \tilde{H}^{(2)}, \cdots, \tilde{H}^{(L-1)}, \tilde{H}^{(L)}, X]. \tag{6.6}$$

Similar to RVFL+, the output weights (β) of the proposed dRVFL+ are obtained by solving closed form solution.

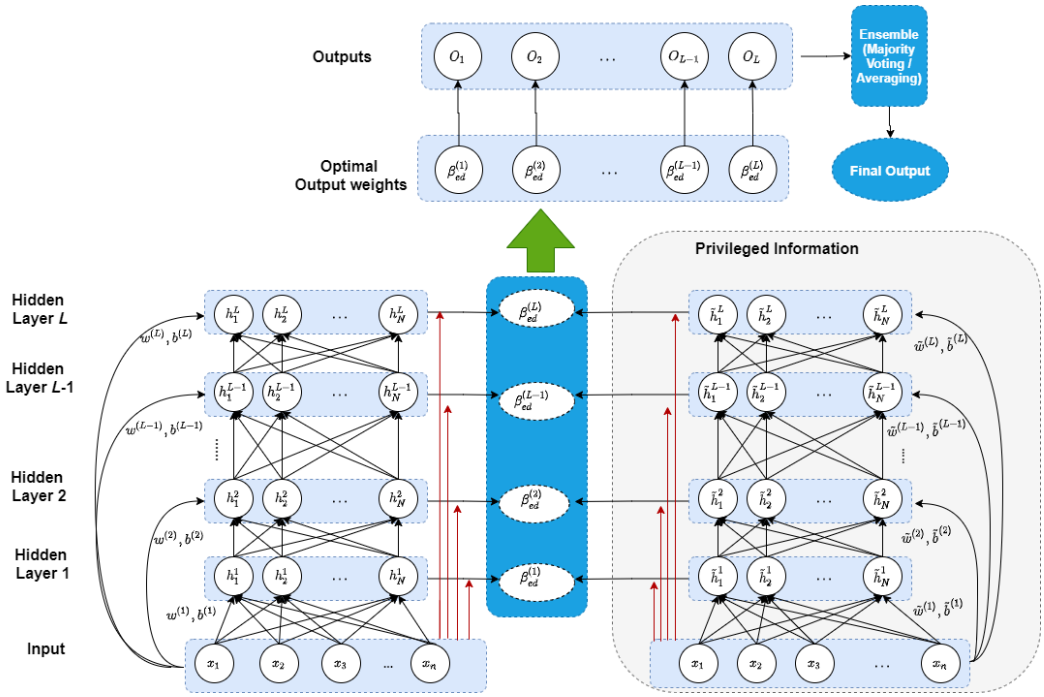


Figure 6.2: Ensemble deep random vector functional link network using privileged information.

6.1.2 Ensemble deep RVFL network using privileged information

The concatenation of all the projections of hidden layer in a single matrix in dRVFL+ exaggerated the complexity of the algorithm. To overcome this issue, we proposed ensemble of deep Random Vector Functional Link Network (edRVFL+). The proposed edRVFL+ utilises the intermediate features unlike the conventional multilayer architecture [255], which results in improved generalization. The architecture of the proposed edRVFL+ is given in Figure 6.2. The proposed edRVFL+ is composed of two components- normal and privileged information processing components. The input to normal component are the original features, processed by the L number of hidden layer, each having N number of nodes. Each hidden layer receives the input as the output of the preceding layer and the original features. The output of first hidden layer is

$$H^{(1)} = g(XW^{(1)}), \tag{6.7}$$

and for the other higher layers ($i > 1$)

$$H^{(i)} = g([H^{i-1}, X]W^{(i)}). \tag{6.8}$$

The privileged information is also generated by the L layer architecture wherein we input the original features. However, the privileged component of the given architecture uses separate set of weights and biases, denoted by $\tilde{w}^i, \tilde{b}^i$ for $i = 1, 2, \ldots, L$, are generated randomly and kept fixed and different activation $f(\cdot)$ is used in the privileged component. Similar to normal part, we fix the rectangular component with N number of hidden nodes in each hidden layer of the L layer architecture. Each privileged hidden layer receives the input as the output of the previous privileged layer and the original features. The output of first hidden layer is

$$\tilde{H}^{(1)} = f(X\tilde{W}^{(1)}), \tag{6.9}$$

and for the other higher layers ($i > 1$)

$$\tilde{H}^{(i)} = f([\tilde{H}^{i-1}, X]\tilde{W}^{i}). \tag{6.10}$$

Similar to RVFL+, with $D = H^{(i)}$ and $\tilde{D} = \tilde{H}^{(i)}$ the output weights $\beta^{(i)}$ are optimised via closed form solution.

6.2 Experiments

In this section, we analyse the performance of the baseline models and the proposed dRVFL+ and edRVFL+ models. The analysis is performed for the diagnosis of Alzheimer's disease.

6.2.1 Experimental setup

The experimental evaluation was performed using MATLAB R2017b and the workstation with Intel Xenon(R) CPU E5-2697 v4 2.30 GHZ and 128 GB RAM. The datasets are randomly partitioned into 70 : 30 ratio. Here, 30% samples are reserved for testing while as 70% of the samples are used for training the models. Also, we use five fold cross validation

on the training set for optimizing the hyperparameters corresponding to different classification models. In five fold cross validation the data is portioned randomly into 5 disjoint sets wherein one is used for testing and the rest are employed for training the model.

The hyperparameters corresponding to different classification models are obtained via grid search approach in the following range $\gamma = C = \{2^{-6}, 2^{-5}, \cdots, 2^{12}\}$ and the number hidden neurons from $\{1028, 2048, 4096\}$. We followed a 2-stage method for obtaining the optimal hyperparameters. In first stage, the number of hidden layers is fixed to two and optimal number of hidden nodes N^* and regularisation parameter C^* is searched from the fine range. In the second stage, tune the number of layers and then fine tune the regularisation parameter C and number of hidden nodes N.

The performance measures used to evaluate the algorithms are given as follows:

$$\text{Accuracy, AUC} = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.11}$$

$$\text{Sensitivity or Recall} = \frac{TP}{TP + FN} \tag{6.12}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6.13}$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.14}$$

$$\text{G-mean} = \sqrt{\text{Precision} \times \text{Recall}} \tag{6.15}$$

where false positive, true positive, false negative and true negative are denoted by FP, TP, FN and TN, respectively.

6.2.2 Evaluation on ADNI dataset

The scans from the ADNI repository (adni.loni.usc.edu) are used in this study. In 2003, the Principal Investigator of ADNI project, Michael W. Weiner, launched the project. The aim of this project is to analyse the neuroimaging techniques like MRI (Magnetic Resonance Imaging), PET (Positron Emission Tomography) and other tests for the early diagnosis of the AD from the MCI (Mild Cognitive Impairment) stage. For detailed information, we refer the interested readers to `www.adni-info.org`. In this study, Volume based Morphometry (VolBM) based features are used. The feature extraction pipeline followed is same as in [75]. The classification models are evaluated in terms of the classification performance in

	RVFL	dRVFL	edRVFL	RVFL+	dRVFL+	edRVFL+
	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)
Activation Function	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)
(Selu, Sin)	(0.8432, 0.9057)	(0.8997, 0.8679)	(0.8929, 0.8679)	(0.886, 0.8679)	(**0.9186**, 0.9057)	(0.8834, 0.8491
	(0.7808, 0.75)	(0.9315, 0.902)	(0.9178, 0.8846)	(0.9041, 0.8679)	(0.9315, 0.9057)	(0.9178, 0.8824)
(Selu, Hardlim)				(0.8321, 0.9245)	(0.772, 0.7358)	(0.8834, 0.8491
				(0.7397, 0.7206)	(0.8082, 0.7358)	(0.9178, 0.8824)
(Selu, Tribas)				(0.8484, 0.9434)	(0.8535, 0.8302)	(0.9049, 0.9057
				(0.7534, 0.7353)	(0.8767, 0.8302)	(0.9041, 0.8727)
(Selu, Radbas)				(0.8416, 0.9434)	(0.8834, 0.8491)	(0.886, 0.8679
				(0.7397, 0.7246)	(0.9178, 0.8824)	(0.9041, 0.8679)
(Selu, Sign)				(0.8201, 0.8868)	(0.772, 0.7358)	(0.8834, 0.8491
				(0.7534, 0.7231)	(0.8082, 0.7358)	(0.9178, 0.8824)
(Relu, Sin)	(0.8295, 0.9057)	(0.8535, 0.8302)	(0.9049, 0.9057)	(0.9117, 0.9057)	(**0.9186**, 0.9057)	(**0.9254**, 0.9057
	(0.7534, 0.7273)	(0.8767, 0.8302)	(0.9041, 0.8727)	(0.9178, 0.8889)	(0.9315, 0.9057)	(0.9452, 0.9231)
(Relu, Hardlim)				(0.8227, 0.9057)	(0.856, 0.8491)	(0.8723, 0.8679
				(0.7397, 0.7164)	(0.863, 0.8182)	(0.8767, 0.8364)
(Relu, Tribas)				(0.8484, 0.9434)	(0.916, 0.8868)	(0.8792, 0.8679
				(0.7534, 0.7353)	(0.9452, 0.9216)	(0.8904, 0.8519)
(Relu, Radbas)				(0.8304, 0.9623)	(0.8586, 0.8679)	(0.8483, 0.7925
				(0.6986, 0.6986)	(0.8493, 0.807)	(0.9041, 0.8571)
(Relu, Sign)				(0.8432, 0.9057)	(0.856, 0.8491)	(0.8723, 0.8679
				(0.7808, 0.75)	(0.863, 0.8182)	(0.8767, 0.8364)
(Sigmoid, Sin)	(0.8903, 0.8491)	(0.9023, 0.8868)	(0.8603, 0.8302)	(0.8714, 0.8113)	(0.8603, 0.8302)	(0.8886, 0.8868
	(0.9315, 0.9)	(0.9178, 0.8868)	(0.8904, 0.8462)	(0.9315, 0.8958)	(0.8904, 0.8462)	(0.8904, 0.8545)
(Sigmoid, Hardlim)				(0.8903, 0.8491)	(0.8792, 0.8679)	(0.8792, 0.8679
				(0.9315, 0.9)	(0.8904, 0.8519)	(0.8904, 0.8519)
(Sigmoid, Tribas)				(0.8414, 0.7925)	(0.8818, 0.8868)	(0.8509, 0.8113
				(0.8904, 0.84)	(0.8767, 0.8393)	(0.8904, 0.8431)
(Sigmoid, Radbas)				(0.8903, 0.8491)	(0.8346, 0.7925)	(0.8886, 0.8868
				(0.9315, 0.9)	(0.8767, 0.8235)	(0.8904, 0.8545)
(Sigmoid, Sign)				(0.8808, 0.8302)	(0.8792, 0.8679)	(0.8792, 0.8679
				(0.9315, 0.898)	(0.8904, 0.8519)	(0.8904, 0.8519)

Here, bold face denotes the performance of top two models.

Table 6.1: Performance evaluation of the algorithms on CN versus AD case.

Control Normal (CN) versus Mild Cognitive Impairment (MCI) subjects, Control Normal (CN) versus Alzheimer's disease (AD) subjects and Mild Cognitive Impairment (MCI) versus Alzheimer's disease subjects.

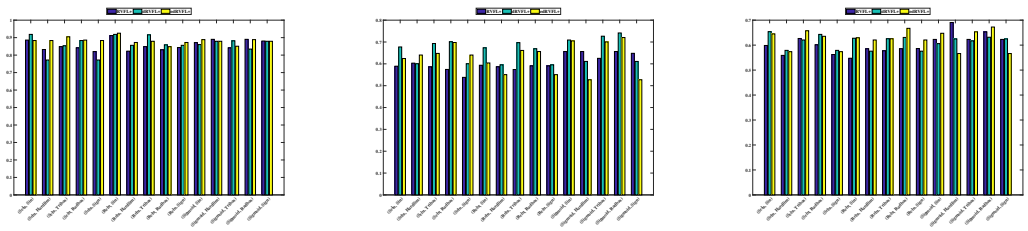
6.2.3 Results and discussion

In this section, we analyse the performance of the models for the diagnosis of the Alzheimer's disease. The architecture of the models involved are based on the processing of normal information and the privileged information. The standard RVFL, dRVFL and edRVFL models are capable of using only the normal information while as the standard RVFL+, dRVFL+ and edRVFL+ models can process both the normal and privileged information. In the literature of LUPI framework, the original feature space is divided into two disjoint subsets wherein one is used as the normal information and the other being used

	RVFL	dRVFL	edRVFL	RVFL+	dRVFL+	edRVFL+
	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)
Activation Function	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)
(Selu, Sin)	(0.5381, 0.459)	(0.5873, 0.5574)	(0.6181, 0.541)	(0.5893, 0.4754)	(0.6774, 0.7377)	(0.6244, 0.6393
	(0.6172, 0.3636)	(0.6172, 0.4096)	(0.6953, 0.4583)	(0.7031, 0.4328)	(0.6172, 0.4787)	(0.6094, 0.4382)
(Selu, Hardlim)				(0.6033, 0.5738)	(0.6005, 0.623)	(0.6404, 0.6557
				(0.6328, 0.4268)	(0.5781, 0.413)	(0.625, 0.4545)
(Selu, Tribas)				(0.5861, 0.5082)	(0.6931, 0.7377)	(0.6478, 0.6393
				(0.6641, 0.4189)	(0.6484, 0.5)	(0.6563, 0.4699)
(Selu, Radbas)				(0.5744, 0.5082)	(0.7013, 0.7541)	(0.6977, 0.7705
				(0.6406, 0.4026)	(0.6484, 0.5055)	(0.625, 0.4947)
(Selu, Sign)				(0.5381, 0.459)	(0.6005, 0.623)	(0.6404, 0.6557
				(0.6172, 0.3636)	(0.5781, 0.413)	(0.625, 0.4545)
(Relu, Sin)	(0.5908, 0.541)	(0.5951, 0.5574)	(0.615, 0.5738)	(0.5936, 0.4918)	(0.6735, 0.7377)	(0.6041, 0.6066
	(0.6406, 0.4177)	(0.6328, 0.4198)	(0.6563, 0.443)	(0.6953, 0.4348)	(0.6094, 0.4737)	(0.6016, 0.4205)
(Relu, Hardlim)				(0.5865, 0.5246)	(0.5955, 0.5738)	(0.5506, 0.4918
				(0.6484, 0.4156)	(0.6172, 0.4167)	(0.6094, 0.375)
(Relu, Tribas)				(0.574, 0.4918)	(0.697, 0.7377)	(0.6614, 0.7213
				(0.6563, 0.4054)	(0.6563, 0.5056)	(0.6016, 0.4632)
(Relu, Radbas)				(0.5908, 0.541)	(0.6692, 0.7213)	(0.6564, 0.6721
				(0.6406, 0.4177)	(0.6172, 0.4731)	(0.6406, 0.4713)
(Relu, Sign)				(0.5908, 0.541)	(0.5955, 0.5738)	(0.5506, 0.4918
				(0.6406, 0.4177)	(0.6172, 0.4167)	(0.6094, 0.375)
(Sigmoid, Sin)	(0.6564, 0.5082)	(0.6181, 0.541)	(0.6744, 0.6066)	(0.6564, 0.5082)	(0.7091, 0.7541)	(0.7052, 0.7541
	(0.8047, 0.5536)	(0.6953, 0.4583)	(0.7422, 0.5286)	(0.8047, 0.5536)	(0.6641, 0.5169)	(0.6563, 0.5111)
(Sigmoid, Hardlim)				(0.6564, 0.5082)	(0.6111, 0.5738)	(0.5272, 0.4918
				(0.8047, 0.5536)	(0.6484, 0.4375)	(0.5625, 0.3488)
(Sigmoid, Tribas)				(0.6252, 0.5082)	(0.7266, 0.8361)	(0.7005, 0.7213
				(0.7422, 0.4844)	(0.6172, 0.51)	(0.6797, 0.5176)
(Sigmoid, Radbas)				(0.6564, 0.5082)	(**0.7415**, 0.8033)	(**0.7204**, 0.7377
				(0.8047, 0.5536)	(0.6797, 0.5444)	(0.7031, 0.5422)
(Sigmoid, Sign)				(0.6482, 0.4918)	(0.6111, 0.5738)	(0.5272, 0.4918
				(0.8047, 0.5455)	(0.6484, 0.4375)	(0.5625, 0.3488)

Here, bold face denotes the performance of top two models.

Table 6.2: Performance evaluation of the algorithms on CN versus MCI case.



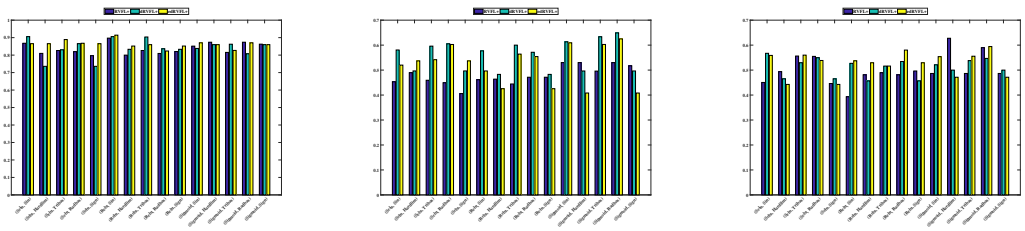
(a) CN-vs-AD (b) CN-vs-MCI (c) MCI-vs-AD

Figure 6.3: AUC analysis of the classification models for AD.

	RVFL	dRVFL	edRVFL	RVFL+	dRVFL+	edRVFL+
	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)	(AUC, Sens.)
Activation Function	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)	(Spec, Prec.)
(Selu, Sin)	(0.5624, 0.4462)	(0.6308, 0.5385)	(0.6601, 0.5077)	(0.5986, 0.3846)	(0.6539, 0.5846)	(0.645, 0.5846
	(0.6786, 0.4462)	(0.7232, 0.5303)	(0.8125, 0.6111)	(0.8125, 0.5435)	(0.7232, 0.5507)	(0.7054, 0.5352)
(Selu, Hardlim)				(0.5589, 0.6)	(0.579, 0.4615)	(0.5738, 0.4154
				(0.5179, 0.4194)	(0.6964, 0.4688)	(0.7321, 0.4737)
(Selu, Tribas)				(0.6266, 0.6462)	(0.6207, 0.5538)	(0.6576, 0.5385
				(0.6071, 0.4884)	(0.6875, 0.507)	(0.7768, 0.5833)
(Selu, Radbas)				(0.6014, 0.7385)	(0.643, 0.5538)	(0.6353, 0.5385
				(0.4643, 0.4444)	(0.7321, 0.5455)	(0.7321, 0.5385)
(Selu, Sign)				(0.5624, 0.4462)	(0.579, 0.4615)	(0.5738, 0.4154
				(0.6786, 0.4462)	(0.6964, 0.4688)	(0.7321, 0.4737)
(Relu, Sin)	(0.5862, 0.5385)	(0.5968, 0.4615)	(0.6102, 0.4615)	(0.5475, 0.3538)	(0.6276, 0.5231)	(0.6296, 0.5538
	(0.6339, 0.4605)	(0.7321, 0.5)	(0.7589, 0.5263)	(0.7411, 0.4423)	(0.7321, 0.5313)	(0.7054, 0.5217)
(Relu, Hardlim)				(0.5854, 0.4923)	(0.5758, 0.4462)	(0.6207, 0.5538
				(0.6786, 0.4706)	(0.7054, 0.4677)	(0.6875, 0.507)
(Relu, Tribas)				(0.5773, 0.5385)	(0.6256, 0.4923)	(0.6256, 0.4923
				(0.6161, 0.4487)	(0.7589, 0.5424)	(0.7589, 0.5424)
(Relu, Radbas)				(0.5854, 0.4923)	(0.6308, 0.5385)	(0.6673, 0.5846
				(0.6786, 0.4706)	(0.7232, 0.5303)	(0.75, 0.5758)
(Relu, Sign)				(0.5862, 0.5385)	(0.5758, 0.4462)	(0.6207, 0.5538
				(0.6339, 0.4605)	(0.7054, 0.4677)	(0.6875, 0.507)
(Sigmoid, Sin)	(0.6229, 0.4154)	(0.6661, 0.6)	(0.6365, 0.5231)	(0.6229, 0.4154)	(0.606, 0.5692)	(0.6475, 0.5538
	(0.8304, 0.587)	(0.7321, 0.5652)	(0.75, 0.5484)	(0.8304, 0.587)	(0.6429, 0.4805)	(0.7411, 0.5538)
(Sigmoid, Hardlim)				(**0.6907**, 0.7385)	(0.6249, 0.4462)	(0.5663, 0.5077
				(0.6429, 0.5455)	(0.8036, 0.5686)	(0.625, 0.44)
(Sigmoid, Tribas)				(0.6229, 0.4154)	(0.617, 0.6)	(0.6532, 0.5385
				(0.8304, 0.587)	(0.6339, 0.4875)	(0.7679, 0.5738)
(Sigmoid, Radbas)				(0.653, 0.7077)	(0.6316, 0.5846)	(**0.6725**, 0.6308
				(0.5982, 0.5055)	(0.6786, 0.5135)	(0.7143, 0.5616)
(Sigmoid, Sign)				(0.6229, 0.4154)	(0.6249, 0.4462)	(0.5663, 0.5077
				(0.8304, 0.587)	(0.8036, 0.5686)	(0.625, 0.44)

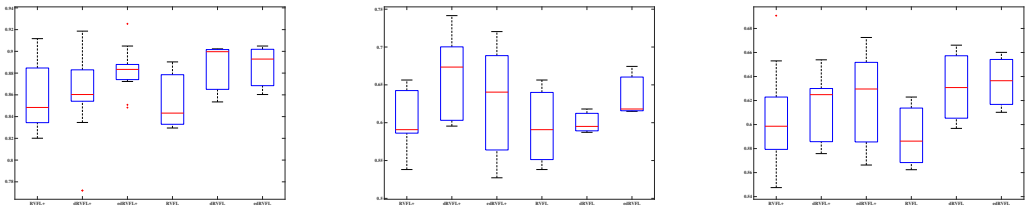
Here, bold face denotes the performance of top two models.

Table 6.3: Performance evaluation of the algorithms on MCI versus AD case.



(a) CN-vs-AD (b) CN-vs-MCI (c) MCI-vs-AD

Figure 6.4: F-Measure analysis of the classification models for AD.



(a) CN-vs-AD (b) CN-vs-MCI (c) MCI-vs-AD

Figure 6.5: AUC analysis of the classification models for AD.

as the privileged information. In this chapter, we introduced a novel strategy to generate the privileged information. Instead of dividing the features into two disjoint subset of features, we generated the privileged information via different activation functions. In standard RVFL+ and proposed deep RVFL+ models, different activation functions are employed to generate the privileged information. We evaluate the models using selu, relu, sigmoid, sin, hardlim, tribas, radbas and sign activation functions. The first three activation functions i.e., selu, relu and sigmoid are used in the processing of normal information while as rest of the activation functions are employed for generating the privileged information.

The performance of the models is evaluated for the classification of different stages of Alzheimer's disease. Here, the evaluation of the models on Control Normal (CN) cases versus Alzheimer disease (AD) cases, Control Normal (CN) cases versus Mild Cognitive Impairment (MCI) cases and Mild Cognitive Impairment (MCI) cases versus Alzheimer disease (AD) cases are presented in Tables 6.1, 6.2 and 6.3 respectively. The first column in each of the given Tables represents the activation functions used in the architecture. Each entry (a, b) in first column denotes that a activation function is used for processing the normal information while b activation function is used for generating the privileged information. For the models like RVFL, dRVFL and edRVFL which are not capable to handle the privileged information, the entry (a, b) denotes that a activation function is used in the architecture while as the models are not capable to use the activation function b.

6.2.3.1 CN versus AD

Table 6.1 gives the evaluation of the models on CN versus AD case in terms of AUC, sensitivity (Sens.), specificity (Spec.), and precision (Prec.). One can see that the proposed dRVFL+ and edRVFL+ emerged as the top two classifiers with AUC of 91.86%

and 92.54%, respectively, followed by RVFL+, edRVFL, dRVFL and standard RVFL with 91.17%, 90.49%, 90.23% and 89.03%, respectively. With respect to activation functions, RVFL+, proposed dRVFL+ and edRVFL+ showed best performance with 'relu' as the activation function for normal processing and 'sin' as the activation function for processing the privileged information. Also, 'sigmoid' activation function proved to be better for RVFL and dRVFL models while as edRVFL model showed best performance with 'relu' activation function. Figure 6.3(a) and Figure 6.4(a) gives the performance of the models in terms of AUC and F-Measure, respectively across different activation functions. From the figures, it is evident that the proposed dRVFL+ and edRVFL+ demonstrate better performance compared to the baseline models across different activation functions. Moreover, the overall effect of different activation functions is shown by Figure 6.5(a). One can see that proposed dRVFL+ and edRVFL+ models show competitive or better performance compared to the given baseline models.

6.2.3.2 CN versus MCI

Table 6.2 gives the performance of the models in terms of AUC, Sens., Spec., and Prec. for the CN versus MCI subjects. Among the given models, the proposed dRVFL+ and edRVFL+ showed highest performance with AUC of 74.05% and 72.04%, respectively. Standard RVFL and RVFL+ showed similar performance with AUC equal to 65.64%, respectively. Moreover, dRVFL and edRVFL models showed AUC of 61.81% and 67.44%, respectively. With respect to the activation functions, the proposed dRVFL+ and edRVFL+ showed best performance with 'sigmoid' and 'radbas' as the activation for normal and privileged information components, respectively. The performance of the models across different activation functions on CN versus MCI case in terms of AUC and F-Measure are presented in Figures 6.3(b) and 6.4(b), respectively. It is clear that the proposed dRVFL+ and edRVFL+ models showed best performance figures. Overall analysis of the models across different activation functions is presented in Figure 6.5(b). It is clear that the proposed dRVFL+ and edRVFL+ models proved to better compared to existing baseline models across different activation functions.

6.2.3.3 MCI versus AD

Table 6.3 gives the evaluation of the models on MCI versus AD case. The metrics evaluated are AUC, Sens., Spec., and Prec. RVFL+ showed best performance with AUC

of 69.07% followed by the proposed edRVFL+ model with AUC equal to 67.25%, respectively. The proposed edRVFL model emerged as the second highest performing classifier. The RVFL+ model showed best figures with 'sigmoid' and 'hardlim' as the activation functions for the normal and privileged information components respectively. However, for the proposed edRVFL+, 'sigmoid' and 'radbas' activation for the normal and privileged information, respectively, proved better in comparison with the existing baseline models. Figure 6.3(c) and 6.4(c) gives the evaluation of the algorithms in terms of AUC and F-Measure. Except RVFL+, the propose edRVFL+ model emerged as the best classifier compared to the existing models. Figure 6.5(c) gives the overall comparison of the models across multiple activation functions. It is evident from the figure that proposed edRVFL+ showed best figures compared to the baseline models (except RVFL+).

6.3 Summary

Alzheimer's disease (AD) is a progressive neuro-degenerative disease. The conditions of AD become worse with each passing day thereby interfering with the basic activities like talking, swallowing etc. Hence, there is a need to develop novel tools for the early diagnosis of AD. In this direction, we proposed deep random vector functional link using privileged information (dRVFL+) and ensemble of deep random vector functional link network using privileged information (edRVFL+) for the diagnosis of Alzheimer's disease. Standard RVFL, deep architectures of RVFL and its ensemble versions have shown better generalization, however, these models use only normal information for optimizing the network parameters. The proposed deep RVFL+ and its ensemble versions are enabled to incorporate the privileged information, which is sidelined by the standard RVFL and its deep models. Both dRVFL+ and edRVFL+ efficiently utilise the privileged information in combination with the original features to get better generalization performance. Unlike the standard LUPI based models (like RVFL+) which normally utilise the half of the available features as normal features and rest as the privileged features. We propose a novel approach for the privileged information. We utilise different activation functions while processing the normal and privileged information in the proposed deep architectures. To the best of our knowledge, this is first time that all the features are utilised in the LUPI framework and a separate information is generated as the privileged information. The proposed dRVFL+ and edRVFL+ models are employed

for the diagnosis of Alzheimer's disease. Experimental results demonstrate the superiority of the proposed dRVFL+ and edRVFL+ models over baseline models. Thus, the proposed edRVFL+ model can be utilised in clinical setting for the diagnosis of AD.

In this chapter, we exploited the randomised feature mapping across different hidden layers and used them for generating the deep RVFL and its ensemble via LUPI framework. Random projection of the data provides better control over complexity and memory issues. In addition to random projections, kernel functions are also used for non-linear projection of the data. Kernel based machines like support vector machines and twin support vector machines use kernel function for generating the non-linear feature representation. The kernel matrix generated by the kernel functions corresponds to all pairs of data points. This results in large computational and storage costs. To overcome these issues, the next chapter presents randomised feature projections based models which provide better control over complexity and memory issues in twin SVM based models.

Chapter 7

LSTSVM classifier with enhanced features from pre-trained functional link network

In previous chapter, we used random projections for non-linear projection of the data. In this chapter, we present the randomized feature projection based least squares twin SVM model. Inspired by the fact that kernel based models suffer from higher computation and memory issues, this chapter presents least squares twin support vector machines (LSTSVM) and pre-trained functional link to enhance the feature space. LSTSVM algorithm is used in many real world classification problems as it has lower computational complexity and solves system of linear equations instead of solving quadratic programming problems (QPPs). Since neural network models provide implicit feature representation and is one of the reasons for the success of neural networks. Here, we present a model wherein the input feature space is enhanced by the pre-trained functional link network. Weights are generated by LSTSVM, and a non-linear function is applied on the product between input features and the weights to get the enhanced features. These features are concatenated with the input features to get the extended feature space. Final classification is done by LSTSVM based on these extended features.

Motivated by [64, 218], direct link benefits of the feed forward network [304] and impact of randomization range on the performance [304], we propose LSTSVM classifier with enhanced features from pre-trained functional link network (ELSTSVM) for classification problems. In the proposed model, the hidden layer weights are initialized randomly from the pre-trained weights of the decision hyperplanes obtained via LSTSVM. Then a non-linear activation function is applied over the product of input features and weights generated. Af-

ter concatenating these enhanced features with the input features, LSTSVM generates the optimal decision hyperplanes for the final classification. Note that the proposed model is different from randomized SVM [64], random vector functional link network RVFL [118] and LSTSVM [144] as follows:

- In randomized SVM [64], the hidden layer weights of extreme learning machine (ELM) network are randomly generated and the final classification is based on SVM. However, the weights in the proposed model are chosen randomly from the weight space generated by LSTSVM and final classification is based on LSTSVM.
- The hidden layer weights in RVFL are generated randomly from some pre-defined range $[-S, S]$. However, the weights in the proposed model are chosen randomly from the weight space generated by LSTSVM.
- The LSTSVM generates the hyperplanes based on the original input features only while as the proposed model generates the hyperplane based on the extended input feature space.

The architecture of the proposed model is discussed as follows:

7.1 LSTSVM classifier with enhanced features from pre-trained functional link network

We propose an improved classification model known as LSTSVM classifier with enhanced features from pre-trained functional link network (ELSTSVM). Unlike RVFL network, pre-trained weights are used in the hidden layer instead of the random weights. We used one-vs-all technique to evaluate the linear LSTSVM on the original input features and obtain the weights of hyperplanes for initializing the hidden layer. The weights generated by the LSTSVM are put in a column vector and from this column vector, the weights are chosen randomly to initialize the hidden layer. Non-linear activation function $y = exp(-s^2)$ is then applied on the product of the weights and input features (here, s represents the input variable and y represents the output variable) to get the enhanced features. These enhanced features provide more information about the input domain. Based on the extended features (enhanced + original features), the LSTSVM generates the hyperplanes for final classification. To

Algorithm 7.1 ELSTSVM.

Input: Let $X = [x_1, x_2, \cdots, x_M]$ be an $M \times n$ training set, $Y = [y_1, y_2, \ldots, y_M]$ represents the corresponding labels, c_1, c_2 parameters and J the number of hidden neurons.

Step 1: Let $A = [X_A]$, $B = [X_B]$ be the samples of 'one' class and 'rest' class, respectively. Then, the LSTSVM is solved as follows:

$$\begin{aligned} &\underset{(w_1,b_1)\in\mathbb{R}^{n+1}}{min} \frac{1}{2}\left\|Aw_1 + eb_1\right\|^2 + \frac{c_1}{2}\left\|\xi_1\right\|^2 \\ &s.t. \quad -(Bw_1 + eb_1) + \xi_1 = e, \\ &\underset{(w_2,b_2)\in\mathbb{R}^{n+1}}{min} \frac{1}{2}\left\|Bw_2 + eb_2\right\|^2 + \frac{c_2}{2}\left\|\xi_2\right\|^2 \\ &s.t. \quad (Aw_2 + eb_2) + \xi_2 = e. \end{aligned}$$

Step 2: Solve the optimization problems in Step 1 and concatenate the weights into a column vector $W' = [w_1; b_1; w_2; b_2]$.

Step 3: Choose the weights V randomly from W' for initializing the weights in the hidden layer. Apply the non-linear function (radbas function) as:

$$X' = g(V_j^T X + b_j), \tag{7.1}$$

where V_j is a weight vector from the input to the j^{th} hidden node, b_j is the bias, X represents the input data, $X' = [x'_1, x'_2, ..., x'_M]$ represents the enhanced features where each $x'_i \in \mathbb{R}^J$ for $i = 1, 2, ..., M$.

Step 4: Let the extended feature sets be represented as $A^* = [X_A, X'_A], B^* = [X_B, X'_B]$. Based on these extended features, LSTSVM is solved as follows:

$$\begin{aligned} &\underset{(w'_1,b'_1)\in\mathbb{R}^{n+J+1}}{min} \frac{1}{2}\left\|A^*w'_1 + eb'_1\right\|^2 + \frac{c_1}{2}\left\|\xi_1\right\|^2 \\ &s.t. \quad -(B^*w'_1 + eb'_1) + \xi_1 = e, \qquad (7.2) \\ &\underset{(w'_2,b'_2)\in\mathbb{R}^{n+J+1}}{min} \frac{1}{2}\left\|B^*w'_2 + eb'_2\right\|^2 + \frac{c_2}{2}\left\|\xi_2\right\|^2 \\ &s.t. \quad (A^*w'_2 + eb'_2) + \xi_2 = e. \qquad (7.3) \end{aligned}$$

Solving optimization problems (7.2) and (7.3) generate the optimal decision hyperplanes via one-vs-all approach.

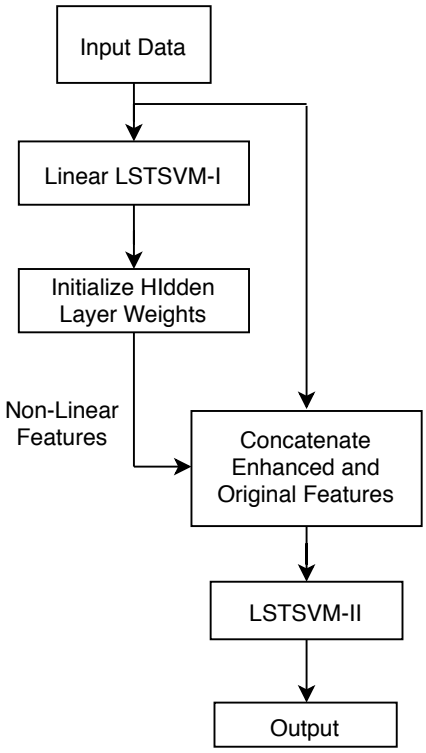


Figure 7.1: Flowchart of proposed enhanced feature based LSTSVM model.

handle the multi-class problems, we used one-vs-all approach for generating the decision hyperplanes. The flowchart of the proposed model is given in Figure 7.1 and the Algorithm 7.1 gives the model in detail.

Comparison with existing Models

The proposed ELSTSVM model is different from existing models as follows:

- In SVM using ELM approach [64] hidden layer input of the feedforward network is initialized randomly and the output layer weights are optimized by SVM for final classification. Also, final decision is based on the transformed input features. In proposed model, the weights are taken randomly from the weight space generated by the LSTSVM network initially and the final classification is done by the LSTSVM model. In addition of transformed features, original input features are also used by proposed model for final classification.

- In RVFL network, the hidden layer weights are initialized randomly and the output layer weights are optimized in closed form solution based on the transformed and

original input features. In the proposed model, weights are chosen from the weight space generated by solving the LSTSVM model on original input feature space and the output layer weights are optimized by the LSTSVM model based on both the original and transformed features.

- In LSTSVM model, the classification hyperplanes are generated based on the original input features space while as the proposed model generates the classification hyperplanes based on the extended input feature space.

7.2 Experiments

In this section, we discuss the experimental protocol followed, analyse and discuss the performance of the classification models.

7.2.1 Experimental setup

In this subsection, we provide the experimental analysis of the models. All the experiments were performed with Intel Xeon Processor (2.3 GHZ) with 128 GB RAM in MATLAB R2010b environment with Windows 10 platform. For evaluating the performances of the given models, we conducted experiments on the benchmark datasets from the UCI repository [60] and some real world datasets (not in UCI) [62, 81]. The real-world datasets not included in UCI repository are about fecundity estimation for fisheries: they are given as oocMerl4d (2-class classification according to the presence/absence of oocyte nucleus) for fish species *Merluccius*, and oocTris2F (nucleus) for fish species *Trisopterus* [81]. N is the parameter that controls the maximum number of enhanced patterns and chosen from the range $[2 : 20 : 302]$. We used *radbas* activation function in all our experiments as it performs better as compared to other activation functions [304]. To handle the multi-class problem, we used one-vs-all approach. In hidden layer, we evaluate LSTSVM once to generate the weights for initializing the hidden layer. In the output layer, LSTSVM based on extended features generates the decision hyperplanes in one-vs-all approach. In $k-$class problem, LSTSVM in the output layer generates k hyperplanes corresponding to each class. For LSTSVM and ELSTSVM, the values of c_1 and c_2 are chosen from the range of values $\{2^{-5}, \cdots, 2^5\}$. Also, for avoiding the computational overhead we used $c_1 = c_2$.

Table 7.1: Classification accuracy[1] of RVFL, LSTSVM and proposed ELSTSVM.

Datasets	**RVFL** (N, Train time)	**LSTSVM** ($c_1 = c_2$, Train time)	**Proposed ELSTSVM** ($c_1 = c_2, N$, Train time)
ionosphere	**89.3525**	86.0996	86.7989
	(282, 0.003386)	(0.03125, 0.003754)	(16, 2, 0.006414)
balance-scale	**91.1994**	87.5231	89.2175
	(142, 0.002542)	(0.5, 0.006866)	(32, 202, 0.055909)
balloons	71.25	65	**86.25**
	(162, 0.00038)	(8, 0.000321)	(0.03125, 162, 0.003972)
blood	77.6203	77.4866	**79.4652**
	(282, 0.007962)	(8, 0.003094)	(1, 262, 0.039643)
lenses	83.3333	81.6667	**85**
	(282, 0.000425)	(32, 0.00084)	(1, 102, 0.004598)
low-res-spect	83.0741	81.6094	**84.9646**
	(102, 0.002821)	(0.0625, 0.080298)	(0.0625, 22, 0.099788)
lung-cancer	39.375	33.75	**42.5**
	(302, 0.00063)	(4, 0.002093)	(0.25, 202, 0.015731)
mammographic	80.374	80.1451	**82.9135**
	(242, 0.008085)	(32, 0.004224)	(1, 62, 0.018453)
musk-1	**85.5882**	81.6807	83.1933
	(282, 0.011921)	(16, 0.010593)	(1, 42, 0.023578)
oocytes-merluccius-nucleus-4d	72.4253	80.9612	**84.0722**
	(182, 0.007133)	(0.25, 0.010624)	(8, 182, 0.053457)
oocytes-trisopterus-nucleus-2f	71.5351	78.7719	**83.114**
	(222, 0.007288)	(0.5, 0.006334)	(16, 302, 0.054958)
oocytes-trisopterus-states-5b	83.7061	92.0614	**93.8816**

Continued on next page

[1]Average accuracy and average training time obtained in 5-times four fold cross validation

Table 7.1 – continued from previous page

Datasets	**RVFL** (N, Train time)	**LSTSVM** ($c_1 = c_2$, Train time)	**Proposed ELSTSVM** ($c_1 = c_2, N$, Train time)
	(82, 0.002218)	(2, 0.023235)	(4, 162, 0.103041)
pima	**78.2552**	76.0938	78.2292
	(302, 0.008921)	(0.5, 0.003748)	(1, 22, 0.009555)
pittsburg-bridges-T-OR-D	87.2815	86.6074	**87.4222**
	(22, 0.000293)	(0.0625, 0.000632)	(0.0625, 2, 0.000998)
planning	**71.461**	69.3688	69.4752
	(2, 0.000238)	(1, 0.000967)	(1, 2, 0.001782)
car	90.8681	77.2106	**96.2037**
	(262, 0.014531)	(1, 0.032452)	(32, 302, 2.69924)
cardiotocography-10clases	72.5944	66.4889	**80.7028**
	(302, 0.020928)	(32, 0.222755)	(0.25, 282, 11.9054)
cardiotocography-3clases	88.2864	86.4522	**91.9557**
	(302, 0.021185)	(16, 0.064246)	(0.125, 282, 3.19593)
congressional-voting	61.5728	**62.9542**	62.769
	(302, 0.004425)	(1, 0.00262)	(1, 2, 0.004761)
cylinder-bands	70.6641	**73.7109**	73.5938
	(62, 0.001239)	(0.25, 0.00347)	(8, 42, 0.009301)
ecoli	80.3571	84.4048	**87.7976**
	(282, 0.002273)	(16, 0.010527)	(1, 22, 0.017743)
energy-y1	88.6719	83.5417	**96.0677**
	(302, 0.009105)	(8, 0.009983)	(0.25, 282, 0.116303)
energy-y2	89.1146	86.1198	**92.7083**
	(42, 0.000928)	(0.125, 0.008758)	(0.5, 202, 0.070699)
glass	59.5146	56.8405	**63.9794**
	(202, 0.0013)	(0.03125, 0.005071)	(8, 22, 0.008152)
heart-va	**36.6**	33.4	35.7
	(242, 0.001263)	(0.0625, 0.004301)	(2, 22, 0.007655)
hepatitis	**82.8466**	80.5071	81.7843

Continued on next page

Table 7.1 – continued from previous page

Datasets	**RVFL** (N, Train time)	**LSTSVM** ($c_1 = c_2$, Train time)	**Proposed ELSTSVM** ($c_1 = c_2, N$, Train time)
	(202, 0.000943)	(0.03125, 0.00106)	(0.5, 2, 0.002004)
statlog-image	84.9524	84.3732	**94.84**
	(302, 0.02231)	(8, 0.149564)	(0.5, 262, 8.00946)
statlog-vehicle	72.5695	76.3095	**82.4529**
	(222, 0.006938)	(16, 0.019444)	(32, 242, 0.14479)
synthetic-control	**95.5667**	80.3	93.7
	(162, 0.00329)	(4, 0.042724)	(16, 82, 0.083962)
teaching	**57.6385**	55.7365	55.8716
	(282, 0.001034)	(1, 0.002089)	(0.5, 222, 0.017315)
tic-tac-toe	93.6317	98.3293	**99.4156**
	(202, 0.00675)	(1, 0.004479)	(1, 302, 0.079665)
vertebral-column-2clases	83.2722	84.8159	**85.808**
	(202, 0.001777)	(0.25, 0.001308)	(1, 22, 0.003362)
vertebral-column-3clases	80.7529	82.313	**83.735**
	(202, 0.001949)	(8, 0.003832)	(0.5, 42, 0.010193)
Average Accuracy	77.4335	76.14041818	**81.07826667**
Overall Win-Tie-Loss	9-0-10	2-0-23	**22-0-0**

7.2.2 Computational complexity analysis

Let $M \times n$ be a binary class training dataset with M_1 number of samples belonging to positive class and M_2 number of samples belonging to negative class such that $M = M_1 + M_2$. By standard mathematical implementation, the inversion of $n \times n$ matrix requires $O(n^3)$ time complexity [312]. In case of RVFL, single matrix inversion of size $(n+J) \times (n+J)$ is required, thus time complexity is given as

$$T = O((n+J)^3) \cong O((n')^3), \tag{7.4}$$

where J is the number of hidden neurons and $n' = n + J$.

In LSTSVM model, two matrix inversions are required each of size $(n+1) \times (n+1)$. Thus, time complexity of LSTSVM is given as:

$$T = 2 \times O((n+1)^3) \cong 2 \times O(n^3). \tag{7.5}$$

The proposed ELSTSVM model involves four matrix inversions with two in hidden layer and two in output layer. In hidden layer, two matrix inversions of size $(n+1) \times (n+1)$ are required and in output layer two matrix inversions of size $(n+J+1) \times (n+J+1)$ are required. Hence, the time complexity of the proposed ELSTSVM is given as:

$$\begin{aligned} T &= 2 \times O((n+1)^3) + 2 \times O((n+J+1)^3), \\ &\cong 2 \times O(n^3) + 2 \times O((n'+1)^3), \\ &\cong 2(O(n^3) + O((n')^3)), \end{aligned} \tag{7.6}$$

where J is the number of enhanced features and $n' = n + J$.

One can see from above equations that the complexity of the proposed model is a function of original features and the number of hidden nodes. Thus, it provides better control over complexity and memory issues compared to the kernel trick.

7.2.3 Significant differences among classifiers with Nemenyi test

The main motive in this subsection is to verify whether the proposed ELSTSVM improves with respect to other baseline classifiers. Thus, we want to ensure that the improvement, if any observed, is due to different random strategies or not. Researchers readily use Friedman test for evaluating the performance of multiple classifiers as it proved to be more robust among other statistical tests [53].

Table 7.1 represents the average values of accuracy and training time obtained from five times four-fold cross validation.

We first rank the performance of RVFL, LSTSVM and the proposed ELSTSVM on each dataset and take average across all the datasets. After calculations, the average ranks of RVFL, LSTSVM and the proposed ELSTSVM are $2.0303, 2.6364$ and 1.3333, respectively.

Then

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right],$$
$$= \frac{12\times 33}{3\times 4}(2.0303^2 + 2.6364^2 + 1.3333^2 - \frac{3\times 4\times 4}{4}) = 28.0636,$$
$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2},$$
$$= \frac{32\times 28.0636}{33\times 2 - 28.0636} = 23.6721,$$

with three algorithms and 33 datasets, F_F is distributed according to the F-distribution with $(3-1) = 2$ and $(3-1)(33-1) = 64$ degrees of freedom. The critical value of $F_{(2,64)}$ for $\alpha = 0.5$ is 2.895. So, we reject the null hypothesis. Based on the Nemenyi test with $\alpha = 0.5$, the critical value is given as:

$$CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}},$$
$$= 2.343\times\sqrt{\frac{3\times 4}{6\times 33}},$$
$$= 0.5768.$$

The difference between the average ranks of (RVFL, ELSTSVM), (LSTSVM, RVFL) and (LSTSVM, ELSTSVM) are (2.0303-1.3333), (2.6364-2.0303) and (2.6364-1.3333), respectively which is greater than 0.5768. So, we conclude that RVFL is significantly better than LSTSVM while the proposed ELSTSVM is significantly better than both LSTSVM and RVFL models.

From Table 7.1, one can observe that the proposed ELSTSVM takes more time as compared to the other models. This is due to the time taken in the generation of extended features and output layer weight optimization in enhanced feature space.

Table 7.2 shows the statistically significant differences among the different classifier models. From the table, it is clear that the proposed ELSTSVM shows statistically significant difference from the other baseline methods. The proposed ELSTSVM achieves better average accuracy among the baseline methods. Also, the proposed ELSTSVM achieves lower rank and hence better performance as compared to other classifier models.

Table 7.2: Statistical difference among the RVFL, LSTSVM and proposed ELSTSVM model based on the Friedman ranking and Nemenyi test.

Method	RVFL (2.0303)	LSTSVM (2.6364)	ELSTSVM (1.3333)
RVFL (2.0303)		✔	
LSTSVM (2.6364)			
ELSTSVM (1.3333)	✔	✔	

The average ranks of the algorithms are given in braces. The rows with ✔ entry shows that the two methods are statistically different and the method in row is better than the method in column. Blank entries show that no statistical difference exists among the methods given in the column and row.

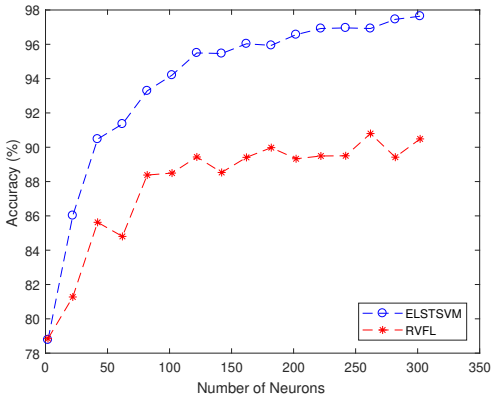
Table 7.3: Statistical difference among the RVFL, LSTSVM and proposed ELSTSVM model based on pairwise sign test.

Method	Significance
(RVFL, LSTSVM) (23, 10)	✔
(ELSTSVM, LSTSVM) (31, 2)	✔
(ELSTSVM, RVFL) (24, 9)	✔

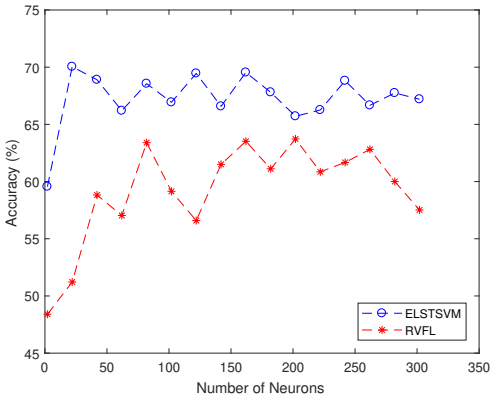
The method in the bracket for (A,B) (say) method represent the number of times A wins w.r.t. method B, ✔ means there is significant difference between this pair of algorithms.

Figure 7.2: Impact of varying the number of enhanced patterns on the performance of RVFL network and the proposed model.

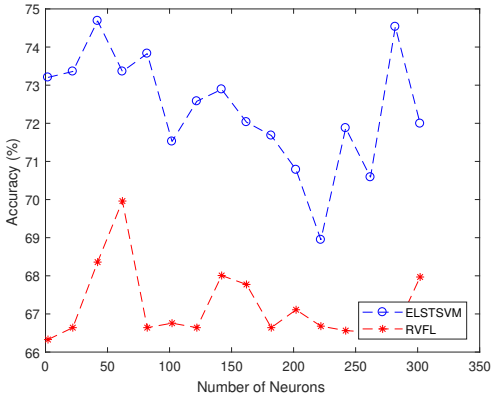
(a) Car.



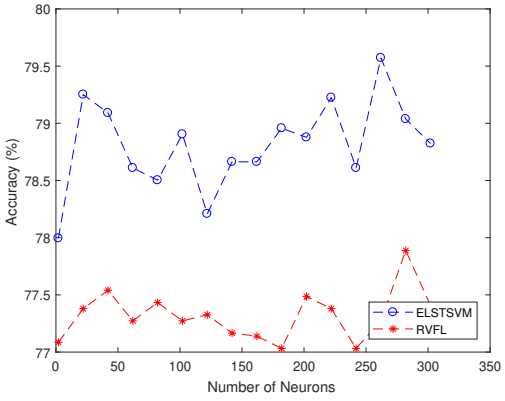
(b) Glass.

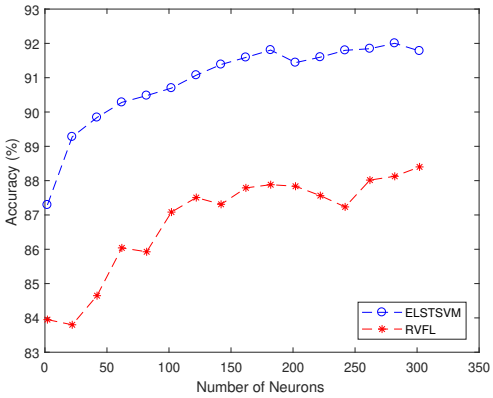


(c) Cylinder-bands.

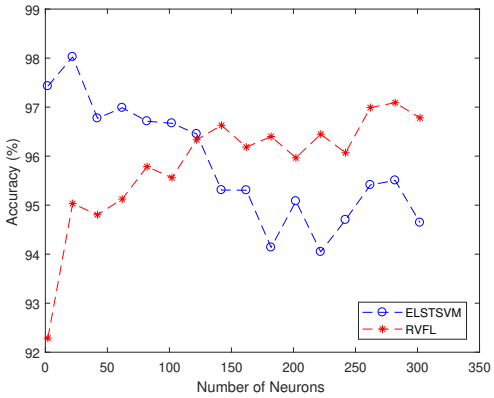


(d) Blood.

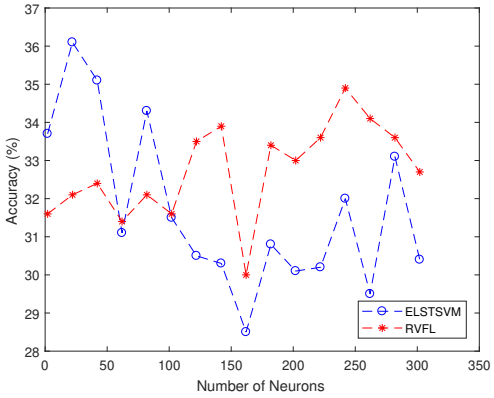




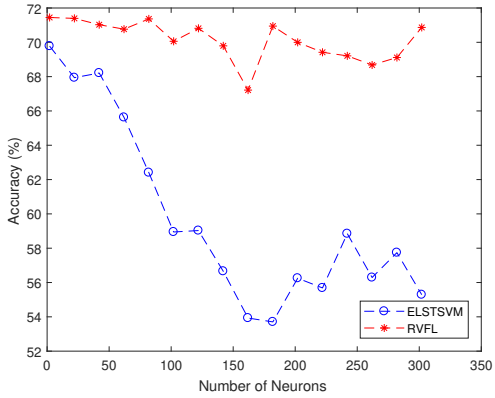
(e) Cardiotocography-3clases.



(f) Dermatology.



(g) Heart-va.



(h) Planning.

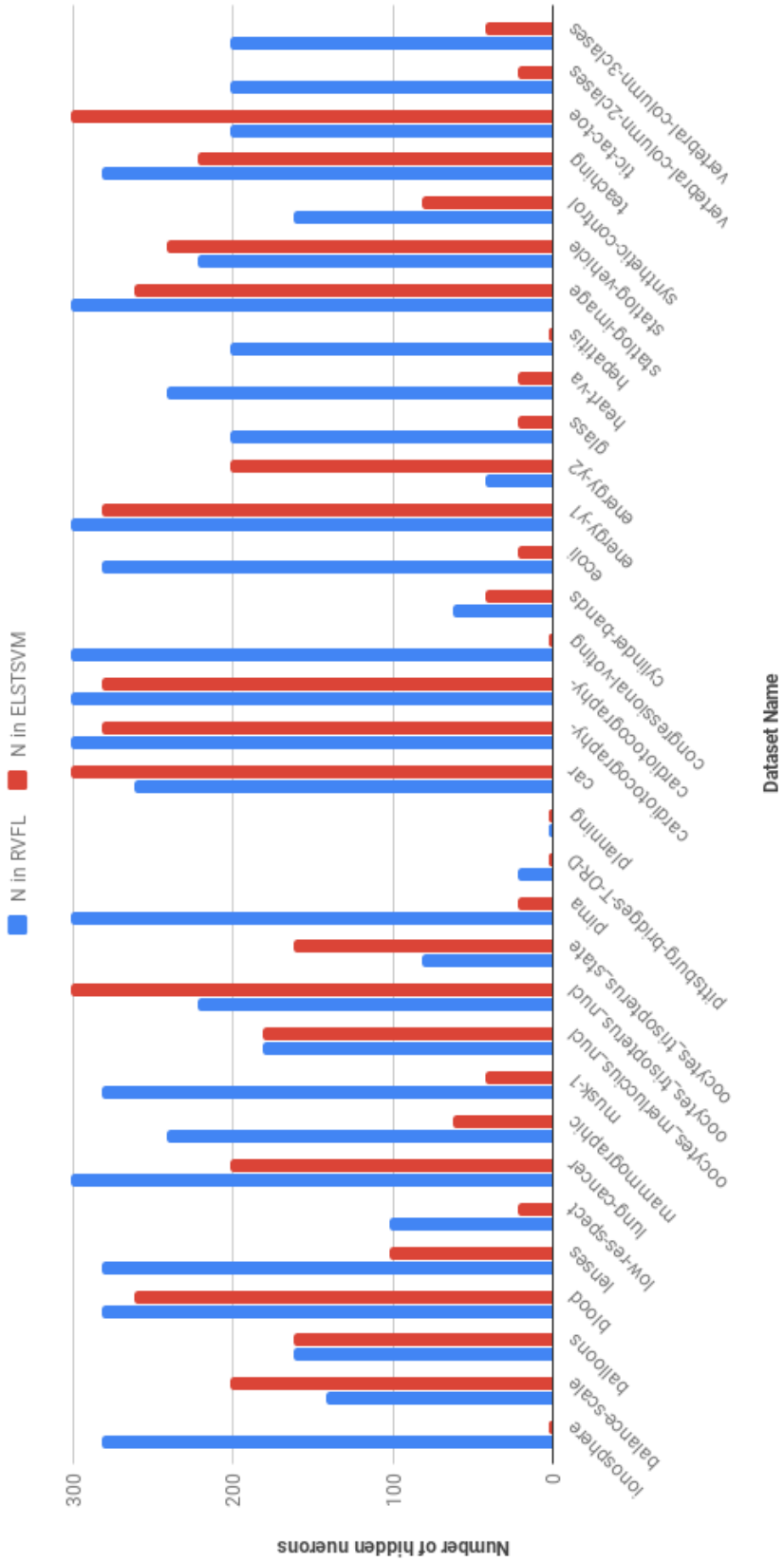


Figure 7.3: Comparison of the number of enhanced features used by RVFL network and the proposed model corresponding to maximum accuracy.

7.2.4 Win-tie-loss: sign test

To analyze the classification performance of different classifier models, we count the number of datasets on which algorithm is the overall winner. We use sign test or the pairwise comparison of algorithms. Table 7.3 gives the pairwise win-tie-loss comparison among the methods. From Table 7.3, it is clear that significant difference exits in (RVFL, LSTSVM), (ELSTSVM, LSTSVM) and (ELSTSVM, RVFL). Also, in most of the datasets the proposed ELSTSVM model emerged as a winner in comparison to other baseline methods.

7.2.5 Comparative analysis of number of enhanced patterns on the RVFL network and the proposed ELSTSVM model

The objective here is to analyze the effect of the number of enhanced patterns introduced in both RVFL network and the proposed ELSTSVM. Figure (7.2(a)) to Figure (7.2(h)) show the effect of number of enhanced patterns on the performance of the proposed ELSTSVM model and RVFL network.

From Figures 7.2(a) and 7.2(e), one can see the effect of increase in number of hidden neurons. It is clear that increasing the number of hidden neuron increases the performance of the classifiers. At higher number of hidden neurons the performance is better as compared to the lower number of hidden neurons.

From Figure (7.2(b)), one can see that the performance of the methods is fluctuating across all the number of hidden neurons. However, from the given figure it is clear that the proposed ELSTSVM outperforms the RVFL network across all the number of neurons.

From Figure (7.2(c)), one can see that the performance of the proposed ELSTSVM initially high and decreases with the increase of enhanced patterns and then increases again at the end. The possible reason for this could be that the enhanced patterns doesn't increase the information given to the model. However, the proposed ELSTSVM model perform better as compared to the RVFL network.

From Figure (7.2(d)), it is clear that performance is initially low and varies with varying the number of hidden neurons. At higher number of hidden neurons the performance is better.

In Figures 7.2(f) and 7.2(g), one can see that the performance of the proposed ELSTSVM model is initially high while as that of RVFL is low. However, with the increase in number of hidden neurons the performance of RVFL increases while as the performance of the proposed ELSTSVM model decreases. Figure (7.2(h)) corresponding to the Planning dataset shows that RVFL outperforms ELSTSVM model across all the number of datasets.

From the given plots, one can see that models are sensitive to the number of hidden neurons. Hence, this parameter should be chosen appropriately to obtain reasonable performance on different datasets.

With respect to RVFL network, the proposed ELSTSVM gives better performance with lower number of hidden neurons in most of the datasets. One can see from the Figure (7.3) that out of 33 datasets, the proposed ELSTSVM uses lesser number of hidden neurons in most of the datasets and at the same time achieves better or comparable performance.

7.3 Summary

In this chapter, we presented an improved model for the classification problems using the LSTSVM and enhanced features from the pre-trained functional link network. Inspired by the direct link benefits and impact of randomization range on the performance of the random vector functional link network. We constructed a model with LSTSVM as the basic unit for classification. The proposed ELSTSVM provides the advantages of implicit feature representation. The original input data is fed into the input layer. At hidden layer, LSTSVM generates the weights and a non-linear function is applied to get the enhanced feature space. Finally, the optimal decision boundary is generated by LSTSVM based on the extended features (input features+enhanced features) to get the final labels. In oocytes-merluccius-nucleus-4d dataset, the proposed ELSTSVM achieved approximately 12% and 4% increase in accuracy with respect to RVFL and LSTSVM model, respectively. In statlog-image dataset, the proposed ELSTSVM achieved approximately 10% increase in accuracy with respect to both the baseline models. Furthermore, the proposed ELSTSVM uses lesser number of hidden neurons and achieves better or comparable performance as that of RVFL network. The proposed model provides better control over computation time and complexity compared to the kernel trick approach. The numerical experiments and the statistical tests performed show that the proposed ELSTSVM gives improved performance as compared to

RVFL and LSTSVM models. From experimental results, one can see that the enhanced features in combination with the original features improved the classification performance.

In this chapter, we explored the LSTSVM model for the projection of the data and optimised the weights via solving the system of equations. However, the hidden layer weights and biases have a great impact on the performance of the models. The next chapter presents variants of the twin SVM based models wherein the hidden layer weights and biases are generated from a specific randomised range and evaluates its effect over multiple twin SVM based models.

Chapter 8

Ensemble of classification models with weighted functional link network

In the previous chapter, least squares twin support vector machine (LSTSVM) model determined the initialization of the hidden layer weights and avoided the computation and memory issues. In this chapter, we present the ensemble of original and the non-linearly projected features via randomization for multiple twin SVM based models.

Ensemble classifiers with random vector functional link network have shown improved performance in classification problems. In this chapter, we present two approaches to solve the classification problems. In the first approach, the original input space's data points are mapped explicitly into a randomized feature space via neural network wherein the weights of the hidden layer are generated randomly. After feature projection, classification models twin bounded support vector machine (TBSVM), least squares twin SVM (LSTSVM), twin k-class support vector classification (TWKSVC), least squares TWKSVC (LSTWKSVC) and robust energy based least squares twin SVM are trained on the extended features (original features and randomized features). In the second approach, twin bounded support vector machine, least squares twin SVM, twin k-class SVM, least squares twin k-class SVM and robust energy based least squares twin SVM models are used to generate the weights of the hidden layer architecture and the weights of output layer are optimized via closed form solution. The performance of both the proposed architectures is evaluated on 33 datasets- including datasets from the UCI repository and fisheries data (not in UCI). Both the experimental results and statistical tests conducted demonstrate that the proposed approaches perform significantly better than the other baseline models. We also analyze the effect of the number

of enhanced features on the performance of the given models.

Generally, input data samples provide a multitude of information from different feature representations, such as compressed feature representation obtained from a lower dimensional feature space representation and the sparse feature representation obtained via higher dimensional feature space [101, 102]. With different feature representations, plenty of underlying information is explored by the different learning algorithms. An ensemble of feature spaces with random forest [305] used different feature representations: principal component analysis (PCA) and linear discriminant analysis (LDA), at each node to provide diverse information and hence, diverse decisions improve the overall performance. Zhang and Suganthan [305] showed that with feature transformations, the information is increased which leads to better generalization performance. Also, random vector functional link network (RVFL) [202] uses random feature transformation in combination with the original input space and has been successfully used in classification and regression tasks. Motivated by the success of different feature representations, we propose two architectures for the classification problems. In the first method, we investigate the performance of the proposed ensemble classifiers with random weighted network based enhanced features and different classification models. Since the proposed approach maps the original input space to a randomized feature space, we name the models as random vector classification models, like if we use random weighted features and LSTSVM, we name it as random vector LSTSVM (RV-LSTSVM) and so on. RV-classification models consist of two phases: Enhancement phase and Classification phase. In the enhancement phase, enhanced feature vector pattern is generated. In this phase, random weights H are generated from the input layer to the enhancement layer. Then, an activation function $g([X \;\; e][H \;\; s]')$ is applied to obtain the enhanced feature vector, here X represents the original features, e is a vector of ones, H is a random weight matrix and s is a bias vector. Both the input feature vector and the enhanced feature vector are concatenated to get the extended feature space. Based on this extended feature space, different classification models are applied thereafter for generating the decision hyperplanes. In the second method, we use different classification models to initialize the hidden layer of the proposed architecture. Based on the different models used for initializing the hidden layer, we name the models accordingly, like in TBSVM with functional link (TBSVM-FL model), TBSVM is used for generation of hidden layer weights. Closed form solution is used to optimize the weights of the output layer architecture.

8.1 Proposed random weighted models

The proposed method can be elaborated in two steps: first step is initial feature mapping, from the input data to the implicit feature representation and the second step is the generation of classification models. Here, we apply the transformation on the original input feature pattern to map onto randomized feature space. Figure 8.1 shows the proposed architecture. Initially, nonlinear transformation $g(\cdot)$ on the random weighted input feature pattern X is applied to map it into a randomized feature space. After the random feature projection, the weights of the output layer are optimized to get the optimal decision hyperplane. In the proposed architecture, we used the enhanced feature pattern (randomized feature space) together with the original features and fed them as input feature vector to the classification model for the optimization of the output layer weights. The addition of the enhanced pattern provides additional knowledge about the sample and hence aids in classification. The concatenation of original features and the randomized features, known as extended features, are used for training. Thus, the rationale of this architecture is to enhance the generalization ability of the models via randomized mapping that are obtained by applying the nonlinear transformation on product of input features and weights between the input and the hidden layer. The weight matrix H from the input layer to the hidden layer are randomly generated such that the activation function $g([A \quad e][H \quad s]')$ is not saturated, where $g(\cdot)$ is an activation function operating element wise on the resulting matrix. To be more specific, let the dimensions of the matrices be $A_{M\times n}, H_{L\times n}$ and s is the bias vector of size L and e is the vector of ones of appropriate dimensions. Then, the resulting enhanced feature matrix (say, K) will be of size $M \times L$. Activation function $g(\cdot)$ is applied elementwise on the matrix K to get the output matrix of the size $M \times L$. Let n represent the number of features of each data sample and L represents the number of hidden neurons (features in the enhanced pattern), then the total number of features corresponding to each sample after concatenation is $n + L$.

Based on the different classification base models, we name the proposed models as the random vector TBSVM (RV-TBSVM), random vector twin k-class support vector classification (RV-TWKSVC), random vector least squares TWKSVC (RV-LSTWKSVC), random vector robust energy based least square TWSVM (RV-RELSTSVM) and random vector LSTSVM (RV-LSTSVM). The modified formulation of the proposed models are given as

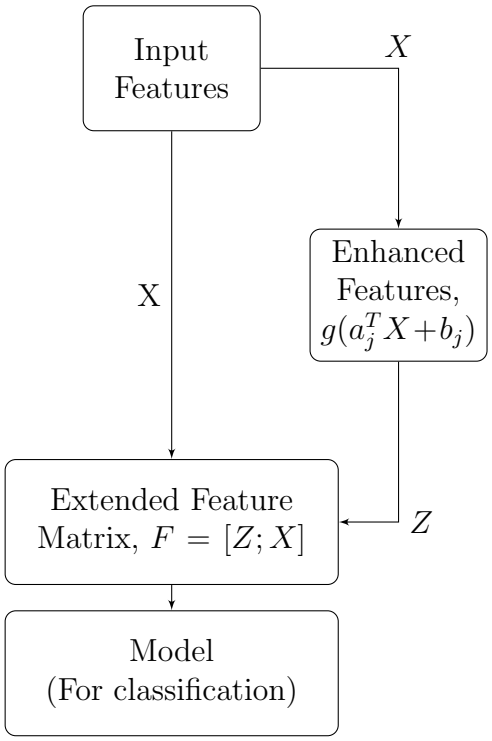


Figure 8.1: Proposed architecture of random weighted models.

follows:

8.1.1 RV-TBSVM

The primal formulation of RV-TBSVM is given as follows:

$$\begin{aligned} \min_{w_1,b_1,\xi,\xi^*} \quad & \frac{1}{2}c_3(||w_1||^2 + b_1^2) + \frac{1}{2}\|\xi^*\|^2 + c_1 e^T \xi \\ s.t. \quad & ([A \;\; g([A \;\; e][H \;\; s]')])w_1 + eb_1 = \xi^*, \\ & -(([B \;\; g([B \;\; e][H \;\; s]')])w_1 + eb_1) + \xi \geq e, \; \xi \geq 0 \end{aligned} \tag{8.1}$$

and

$$\begin{aligned} \min_{w_2,b_2,\eta,\eta^*} \quad & \frac{1}{2}c_4(||w_2||^2 + b_2^2) + \frac{1}{2}\|\eta^*\|^2 + c_2 e^T \eta \\ s.t. \quad & ([B \;\; g([B \;\; e][H \;\; s]')])w_2 + eb_2 = \eta^*, \\ & (([A \;\; g([A \;\; e][H \;\; s]')])w_2 + eb_2) + \eta \geq e, \; \eta \geq 0. \end{aligned} \tag{8.2}$$

8.1.2 RV-LSTSVM

The primal formulation of RV-LSTSVM is given as follows:

$$\begin{aligned} \underset{w_1,b_1,\xi^*}{min} \quad & \frac{1}{2}\left\|([A \;\; g([A \;\; e][H \;\; s]')])w_1 + eb_1\right\|^2 + \frac{c_1}{2}\left\|\xi^*\right\|^2 \\ s.t. \quad & -(([B \;\; g([B \;\; e][H \;\; s]')])w_1 + eb_1) + \xi^* = e \end{aligned} \tag{8.3}$$

and

$$\begin{aligned} \underset{w_2,b_2,\eta^*}{min} \quad & \frac{1}{2}\left\|([B \;\; g([B \;\; e][H \;\; s]')])w_2 + eb_2\right\|^2 + \frac{c_2}{2}\left\|\eta^*\right\|^2 \\ s.t. \quad & (([A \;\; g([A \;\; e][H \;\; s]')])w_2 + eb_2) + \eta^* = e. \end{aligned} \tag{8.4}$$

8.1.3 RV-RELSTSVM

The primal formulation of RV-RELSTSVM is given as follows:

$$\begin{aligned} \underset{w_1,b_1,\xi^*}{min} \quad & \frac{1}{2}\left\|([A \;\; g([A \;\; e][H \;\; s]')])w_1 + eb_1\right\|^2 + \frac{c_1}{2}\left\|\xi^*\right\|^2 + \frac{c_3}{2}\left\|\begin{bmatrix} w_1 \\ b_1 \end{bmatrix}\right\|^2 \\ s.t. \quad & -(([B \;\; g([B \;\; e][H \;\; s]')])w_1 + eb_1) + \xi^* = e \end{aligned} \tag{8.5}$$

and

$$\begin{aligned} \underset{w_2,b_2,\eta^*}{min} \quad & \frac{1}{2}\left\|([B \;\; g([B \;\; e][H \;\; s]')])w_2 + eb_2\right\|^2 + \frac{c_2}{2}\left\|\eta^*\right\|^2 + \frac{c_4}{2}\left\|\begin{bmatrix} w_2 \\ b_2 \end{bmatrix}\right\|^2 \\ s.t. \quad & (([A \;\; g([A \;\; e][H \;\; s]')])w_2 + eb_2) + \eta^* = E_2. \end{aligned} \tag{8.6}$$

8.1.4 RV-TWKSVC

The primal formulation of RV-TWKSVC is given as follows:

$$\begin{aligned} \underset{w_1,b_1,\xi,\xi^*}{min} \quad & \frac{1}{2}\left\|([A \;\; g([A \;\; e][H \;\; s]')])w_1 + eb_1\right\|^2 + c_1 e^t\xi^* + c_2 e_3^t\xi \\ s.t. \quad & -(([B \;\; g([B \;\; e][H \;\; s]')])w_1 + eb_1) + \xi^* \geq e, \\ & -(([C \;\; g([C \;\; e_3][H \;\; s]')])w_1 + e_3 b_1) + \xi \geq e_3(1-\epsilon), \\ & \xi^* \geq 0, \; \xi \geq 0 \end{aligned} \tag{8.7}$$

and

$$\begin{aligned} \underset{w_2,b_2,\eta,\eta^*}{min} \quad & \frac{1}{2}\left\|([B \;\; g([B \;\; e][H \;\; s]')])w_2 + eb_2\right\|^2 + c_3 e^t \eta^* + c_4 e_3^t \eta \\ s.t. \quad & (([A \;\; g([A \;\; e][H \;\; s]')])w_2 + eb_2) + \eta^* \geq e, \\ & (([C \;\; g([C \;\; e_3][H \;\; s]')])w_2 + e_3 b_2) + \eta \geq e_3(1-\epsilon), \\ & \eta^* \geq 0, \; \eta \geq 0. \end{aligned} \tag{8.8}$$

8.1.5 RV-LSTWKSVC

The primal problems of RV-LSTWKSVC is given as follows:

$$\begin{aligned} \underset{w_1,b_1,\xi,\xi^*}{min} \quad & \frac{1}{2}\left\|([A \;\; g([A \;\; e][H \;\; s]')])w_1 + eb_1\right\|^2 + \frac{c_1}{2}\left\|\xi^*\right\|^2 + \frac{c_2}{2}\left\|\xi\right\|^2 \\ s.t. \quad & -(([B \;\; g([B \;\; e][H \;\; s]')])w_1 + eb_1) + \xi^* = e, \\ & -(([C \;\; g([C \;\; e_3][H \;\; s]')])w_1 + e_3 b_1) + \xi = e_3(1-\epsilon) \end{aligned} \tag{8.9}$$

and

$$\begin{aligned} \underset{w_2,b_2,\eta,\eta^*}{min} \quad & \frac{1}{2}\left\|([B \;\; g([B \;\; e][H \;\; s]')])w_2 + eb_2\right\|^2 + \frac{c_3}{2}\left\|\eta^*\right\|^2 + \frac{c_4}{2}\left\|\eta\right\|^2 \\ s.t. \quad & (([A \;\; g([A \;\; e][H \;\; s]')])w_2 + eb_2) + \eta^* = e, \\ & (([C \;\; g([C \;\; e_3][H \;\; s]')])w_2 + e_3 b_2) + \eta = e_3(1-\epsilon). \end{aligned} \tag{8.10}$$

8.2 Proposed twin weighted models

In this section, we will elaborate the parameter learning process of the proposed models. The learning process of the proposed approach using different models like twin bounded SVM (TBSVM), twin k-class support vector classification (TWKSVC), least square TWKSVC (LSTWKSVC), robust energy based LSTSVM (RELSTSVM) and LSTSVM models is a three step process. The main three steps involved are

- Weight generation phase.
- Training of models.
- Output value prediction.

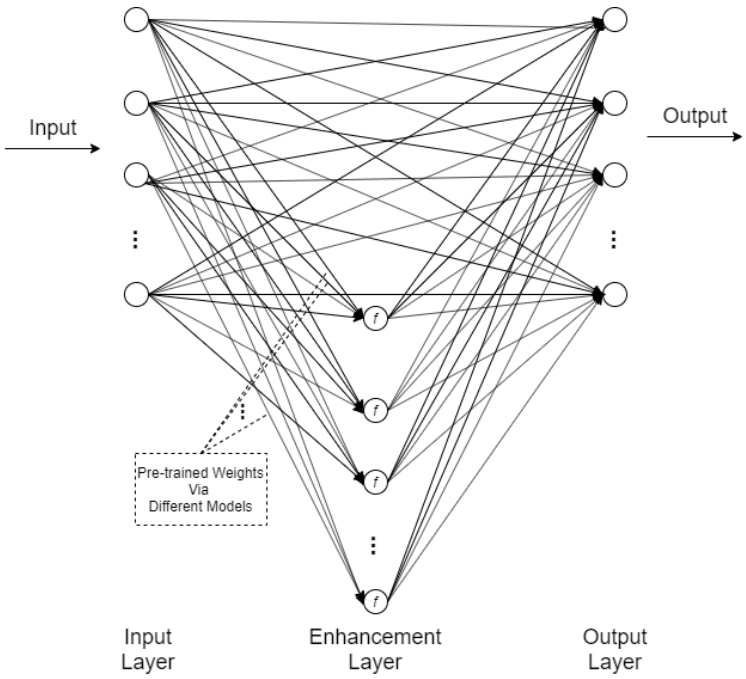


Figure 8.2: Proposed architecture of twin weighted models.

8.2.1 Weight generation phase

The performance of the random vector functional link model is sensitive to the parameter selection. Hence, the efficient selection of model parameters can improve the performance of the model. We use TBSVM, TWKSVC, LSTWKSVC, RELSTSVM and LSTSVM models to generate the weights by solving their corresponding optimization problems. The weights generated by solving the objective functions of the TBSVM, TWKSVC, LSTWKSVC, RELSTSVM and LSTSVM are known as pre-trained weights. These weights are used to initialize the hidden layer of the proposed architecture. Based on the different models, we name these classifiers as TBSVM based functional link (TBSVM-FL), TWKSVC based functional link (TWKSVC-FL), LSTWKSVC based functional link (LSTWKSVC-FL), RELSTSVM based functional link (RELSTSVM-FL) and LSTSVM based functional link (LSTSVM-FL) as they solve the optimization problems of TBSVM, TWKSVC, LSTWKSVC, RELSTSVM and LSTSVM for initializing the hidden layer. The optimization problems solved by TBSVM-FL, TWKSVC-FL, LSTWKSVC-FL, RELSTSVM-FL and LSTSVM-FL are the QPPs (2.22, 2.23), (2.38, 2.39), (2.44, 2.45), (2.33, 2.34) and (2.29, 2.30) respectively.

8.2.2 Training of models

In this subsection, we focus on learning the efficient model via optimal parameter tuning. The learning of parameters via different models gives the appropriate weights for initialization of the hidden layer of the network. For multiclass problems, binary class models are extended to multiclass via 'one-vs-all' approach. We randomly selected the first class as the class 'one' and the rest classes in 'all' class, and generate optimal weight hyperplanes corresponding to these classes. This optimal weight matrix can be represented as $\hat{W} = [w_1, b_1; w_2, b_2]$. With the weight matrix $\bar{W}$ and biases $\bar{b}$ chosen randomly from the weight matrix $\hat{W}$ for the initialization of the hidden layer parameters. The objective function of the proposed architecture with L enhancement nodes (i.e., hidden nodes) is given as:

$$f^*(x) = \sum_j (\bar{\beta}_j . g(\bar{W}_j^T x + \bar{b}_j)),$$

where $\bar{\beta}_j$ are the output layer weights that need to be optimized.
We use ridge regression to optimize the weights of the output layer as:

$$\sum_i (y_i - d_i^T \bar{\beta})^2 + \lambda ||\bar{\beta}||^2, \quad i = 1, 2, \cdots, M. \tag{8.11}$$

Now, $\bar{\beta} = (D^T D + \lambda I)^{-1} D^T Y$ gives the solution, here λ, D and Y represent the regularization parameter, sample features and target responses, respectively.

8.2.3 Output value prediction

Here, the testing data samples are firstly projected into the hidden layer to get the enhanced feature representation. After learning the parameters $\bar{W}, \bar{b}$ and $\bar{\beta}$, the labels of the samples can be predicted as:

$$y_i^{test} = [\bar{\beta} . g(\bar{W}^t x_i^{test} + \bar{b})], \tag{8.12}$$

where y_i^{test} is the predicted target value for testing sample x_i^{test}.

The proposed twin weighted architecture is depicted in Figure (8.2) and the detailed steps are given in Algorithm 8.1.

Algorithm 8.1 Proposed twin weighted models.

Input:
$X = M \times n$ is the dataset with M samples each with feature length n.
$Y = M \times 1$ are data labels corresponding to the training dataset.
N is number of hidden neurons.

[1] Training with different classification models: Solve the optimization problems corresponding to TBSVM, TWKSVC, LSTWKSVC, RELSTSVM and LSTSVM.

[2] For each model, choose the weights randomly from the weights of the corresponding hyperplane generated in $step - 1$, to assign as weights of input layer weights of the proposed architecture.

[3] For each model, apply the non-linear function (radbas function)

$$y = exp(-s^2)$$

on the weighted sum generated from the input layer to the hidden layer. Here, s and y are the input and output variables respectively.

[4] For each model, optimize the weights $\bar{\beta}$ of the output layer. For obtaining the weights of the output layer, we used regularized least square (ridge regression) method as:

$$\sum_i (y_i - d_i^T \bar{\beta})^2 + \lambda ||\bar{\beta}||^2, \quad i = 1, 2, \cdots, M. \tag{8.13}$$

$\bar{\beta} = (D^T D + \lambda I)^{-1} D^T Y$ gives the optimal output layer weights, here λ, D and Y represents the regularization parameter, sample features and target responses, respectively.

[5] For each model, predict the output label:

$$y_i^{test} = [\bar{\beta}.g(\bar{W}^t x_i^{test} + \bar{b})], \tag{8.14}$$

where y_i^{test} is the predicted target value for testing sample x_i^{test}.

8.3 Experiments

In this section, we discuss the experimental setup and analyse the performance of the classification models.

8.3.1 Experimental setup

Experimental study was done to evaluate the performances of different classifiers. The different classification models evaluated are RVFL, TBSVM, RV-TBSVM, TWKSVC, RV-TWKSVC, LSTWKSVC, RV-LSTWKSVC, RELSTSVM, RV-RELSTSVM, LSTSVM and RV-LSTSVM.

We evaluate the proposed and baseline methods based on the benchmark datasets from the UCI repository [60, 62] and real world non-UCI datasets which are about fecundity estimation of fisheries: namely, oocMerl2F (3-class classification according to the stage of development of the oocyte) for fish species Merluccius; and oocTris5B (stages) for fish species Trisopterus [81]. All the experiments were performed in MATLAB R2017a on the machine Intel(R) core(TM) $i7-6700$ processor with 8GB RAM and windows-10 platform. The experimental setting of parameters used in the chapter are given as follows

- Enhanced number of patterns (L) was chosen from the range $[2:20:300]$.
- *Radbas* activation function is used in all the methods. The performance of *radbas* function is comparatively better than other activation functions [304].
- For the given methods, we performed grid search over the parameters of c_1 and c_3 with the parameters varying from $\{2^{-5}, 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3, 2^4, 2^5\}$. To reduce the computation, we set the optimal parameters as $c_1 = c_2$ and $c_3 = c_4$.

8.3.2 Computational complexity analysis

Consider a binary class problem with dataset size $M \times n$, where M is the number of samples and n is the number of features corresponding to each sample. Let L be the number of enhanced features. Following the standard mathematical approach, the inversion of the $n \times n$ matrix requires $O(n^3)$ complexity.

In the first proposed approach, wherein the enhanced features are generated via random mapping, inverting the matrix of size $(n + L + 1) \times (n + L + 1)$ are involved while as in baseline models (TBSVM, TWKSVC, LSTWKSVC, RELSTSVM, LSTSVM) inverting the matrix of size $(n + 1) \times (n + 1)$ are involved. Thus, in the proposed random based feature generation approach, matrix inversions of large size are involved.

In linear twin weighted models, hidden layer feature generation involves the inversion of matrix size $(n+1) \times (n+1)$ and the output layer involves the inversion of matrix of size $(n+L+1) \times (n+L+1)$. Hence, in twin weighted models additional matrix inversions are calculated while as in random weighted feature generation approach no such matrix inversion is required as it involves random mapping.

From the above analysis, it is clear that the complexity of the proposed models is a function of n and L. Hence, L provided better control over complexity compared to the complexity involved by the kernel matrix.

8.3.3 Performance analysis

The classification performance of the models are reported in Table 8.2 and Table 8.3. Table 8.2 corresponds to the classification of the proposed random weighted and the baseline models while the Table 8.3 reports the classification accuracies corresponding to the second proposed method and the classification models.

From Table 8.2, one can see that the classification accuracy of the proposed RV-RELSTSVM model is highest among the given classification models. Also, the average rank of the proposed RV-RELSTSVM model is lowest (2.2727) among the given classification models.

From Table 8.3, one can see that the TBSVM-FL is highest among the given classification models. Also, the average rank of the proposed TBSVM-FL is 3.8788 which is lowest among the given classification models. The second highest accuracy corresponds to the proposed RELSTSVM-FL model with 75.6584. Hence, the proposed TBSVM-FL and RELSTSVM-FL models showed better generalization performance among the given classification models.

From the above analysis, it is clear that the performance of the classification models is improved with the extended feature space comprising of both the enhanced features and the original features. Hence, enhanced features lead to improved generalization performance of the classification models.

Dataset	Patterns	Features	Classes
balance-scale	625	4	3
breast-tissue	106	9	6
contrac	1473	9	3
dermatology	366	34	6
ecoli	336	7	8
energy-y1	768	8	3
energy-y2	768	8	3
flags	194	28	8
glass	214	9	6
heart-cleveland	303	13	5
heart-switzerland	123	12	2
heart-va	200	12	5
iris	150	4	3
led-display	1000	7	10
lenses	24	4	3
libras	360	90	15
low-res-spect	531	100	9
lung-cancer	32	56	3
lymphography	148	18	4
oocytes-merluccius-states-2f	1022	25	3
oocytes-trisopterus-states-5b	912	32	3
pittsburg-bridges-MATERIAL	106	4	3
pittsburg-bridges-REL-L	103	4	3
pittsburg-bridges-SPAN	92	4	3
pittsburg-bridges-TYPE	105	4	6
primary-tumor	330	17	15
seeds	210	7	3
statlog-vehicle	846	18	4
synthetic-control	600	60	6
teaching	151	5	3
vertebral-column-3clases	310	6	3
wine	179	13	3
zoo	101	16	7

Table 8.1: Summary of the datasets used for evaluation.

Table 8.2: Classification accuracy of RVFL [202], TBSVM [233], TWKSVC [294], LSTWKSVC [191], RELSTSVM [260], LSTSVM [144] and proposed classification models.

Dataset	RVFL	TBSVM	RV-TBSVM*	TWKSVC	RV-TWKSVC*	LSTWKSVC	RV-LSTWKSVC*	RELSTSVM	RV-RELSTSVM*	LSTSVM	RV-LSTSVM*
balance-scale	91.5191	87.5196	95.2634	87.4875	**95.7415**	87.806	91.2629	87.232	91.9027	87.0396	91.072
breast-tissue	60.2335	62.0879	64.9038	68.6813	**72.5275**	63.9423	70.6731	64.011	67.7885	64.011	67.7885
contrac	55.1931	51.7328	56.2809	52.0021	55.6023	50.5777	55.9403	51.2563	**56.4172**	51.1882	55.8728
dermatology	96.9839	97.2646	97.8081	93.7286	93.7109	97.2764	98.0769	98.0769	**98.6323**	98.0769	98.0769
ecoli	81.8452	84.8214	87.7976	87.5	86.0119	85.7143	87.7976	85.119	**88.6905**	84.8214	88.3929
energy-y1	89.0625	85.8073	93.75	88.5417	**96.4844**	88.6719	95.5729	83.9844	94.6615	83.9844	95.8333
energy-y2	88.8021	87.7604	92.3177	88.9323	92.4479	87.5	**93.3594**	86.7187	92.9688	86.0677	93.2292
flags	50.0208	47.9167	**53.125**	35.0417	37.6875	46.875	51.4792	51	53.0417	49.4583	47.8333
glass	65.3345	57.9074	68.6364	63.5678	67.8302	61.5437	68.2161	62.5214	69.1424	61.1407	**69.1595**
heart-cleveland	57.7179	59.7436	59.0769	59.0641	58.3846	59.0513	59.3846	**60.7308**	59.7436	59.4231	57.1154
heart-switzerland	46.0606	43.7879	**48.7121**	41.0606	33.9394	41.0606	47.0455	42.7273	47.803	41.1364	40.6061
heart-va	32	30	**36.5**	29	31	31	**36.5**	28.5	**36.5**	28.5	34

Continued on next page

Table 8.2 – continued from previous page

Dataset	RVFL	TBSVM	RV-TBSVM*	TWKSVC	RV-TWKSVC*	LSTWKSVC	RV-LSTWKSVC*	RELSTSVM	RV-RELSTSVM*	LSTSVM	RV-LSTSVM*
iris	95.9806	93.3818	98.0076	98.0423	97.3666	96.6909	97.3666	90.6791	**98.6833**	90.6791	97.3666
led-display	72.8	72.5	73.4	72.8	72.3	71.6	72.7	72.7	**73.6**	72.5	72.9
lenses	87.5	87.5	87.5	83.3333	83.3333	87.5	87.5	**91.6667**	**91.6667**	**91.6667**	**91.6667**
libras	73.8889	56.3889	82.7778	45.5556	77.2222	62.7778	85.5556	64.1667	**87.5**	56.6667	82.5
low-res-spect	83.6237	85.7155	88.1524	73.2407	81.5446	82.6936	87.9588	87.3864	**89.0783**	82.1086	86.8182
lung-cancer	43.75	34.375	43.75	37.5	37.5	37.5	**50**	**50**	46.875	43.75	43.75
lymphography	**85.8108**	64.8649	66.2162	83.1081	82.4324	83.7838	82.4324	66.2162	66.2162	66.2162	65.5405
oocytes-merluccius-states-2f	92.1767	92.6646	93.3501	92.6646	**94.0364**	90.6134	92.7619	92.1759	93.74	92.1752	93.7407
oocytes-trisopterus-states-5b	84.6491	93.0921	94.0789	92.9825	94.1886	91.6667	**94.8465**	92.2149	94.7368	92.1053	94.6272

Continued on next page

Table 8.2 – continued from previous page

Dataset	RVFL	TBSVM	RV-TBSVM*	TWKSVC	RV-TWKSVC*	LSTWKSVC	RV-LSTWKSVC*	RELSTSVM	RV-RELSTSVM*	LSTSVM	RV-LSTSVM*
pittsburg-bridges-MATERIAL	84.0659	86.8132	86.8132	84.1346	85.0962	85.8516	**86.8819**	85.0275	85.9203	85.0275	85.9203
pittsburg-bridges-REL-L	69.75	69.75	**73.75**	70.75	68.0714	71.9643	**73.75**	69.75	73.5357	68.8571	71.4286
pittsburg-bridges-SPAN	68.4783	67.3913	**72.8261**	63.0435	65.2174	63.0435	64.1304	65.2174	71.7391	65.2174	64.1304
pittsburg-bridges-TYPE	58.084	57.0869	59.0456	57.0869	55.1638	60.8974	59.01	58.0484	**60.933**	58.0484	58.0484
primary-tumor	45.6809	43.2491	44.4977	42.378	42.0877	46.3124	47.2198	46.9077	**47.5174**	46.0003	46.9077
seeds	92.3967	96.1895	96.2429	96.2251	**97.6496**	94.3198	95.7621	97.1688	97.1866	97.1688	96.688
statlog-vehicle	72.924	77.783	85.9383	79.2037	84.635	77.4364	85.2318	78.372	**86.0601**	78.2536	85.5895
synthetic-control	95.5	88.1667	**97.6667**	89.5	91.8333	87.5	94.6667	85.6667	97.3333	82.8333	95.6667

Continued on next page

Table 8.2 – continued from previous page

Dataset	RVFL	TBSVM	RV-TBSVM*	TWKSVC	RV-TWKSVC*	LSTWKSVC	RV-LSTWKSVC*	RELSTSVM	RV-RELSTSVM*	LSTSVM	RV-LSTSVM*
teaching	55.0676	55.1689	55.0676	53.0912	**62.8716**	53.8176	59.6453	54.4932	58.8176	54.4932	57.0946
vertebral-column-3clases	81.0414	75.2137	85.854	85.2293	85.5047	81.0332	**86.4952**	83.9388	86.4869	83.9306	86.8116
wine	98.3202	97.752	97.1838	97.2085	98.3202	97.2332	96.1215	97.752	**98.8883**	97.752	97.1591
zoo	94.1154	93.1154	95.1154	91.1154	93.0769	95.1154	**96.0385**	93.1154	**96.0385**	91.1923	90.1923
Average-Accuracy	74.2539	72.2579	76.7093	72.2061	74.8733	73.3446	77.3147	73.471	77.5711	72.4694	75.8645
Average-Rank	6.8182	7.7121	3.9242	7.7576	6.1818	7.8182	3.7879	6.8182	2.2727	7.9545	4.9545
Overall-Win-Tie-Loss	1-0-6	0-0-4	4-0-0	0-0-4	6-0-5	0-0-3	3-0-1	1-0-0	11-0-0	0-0-3	2-0-1

Here, * denote the proposed methods.

Table 8.3: Classification accuracy of RVFL [202], RVFL-AE [311], TBSVM [233], TWKSVC [294], LSTWKSVC [191], RELSTSVM [260], LSTSVM [144] and proposed classification models.

Dataset	RVFL	RVFL-AE	TBSVM	TBSVM-FL*	TWKSVC	TWKSVC-FL*	LSTWKSVC	LSTWKSVC-FL*	RELSTSVM	RELSTSVM-FL*	LSTSVM	LSTSVM-FL*
balance-scale	91.5191	91.8103	87.5196	91.9716	87.4875	91.4998	87.806	91.8223	87.232	91.649	87.0396	91.3264
breast-tissue	60.2335	62.25	62.0879	61.875	68.6813	55.625	63.9423	65.25	64.011	66.25	64.011	59.625
contrac	55.1931	54.9224	51.7328	57.0231	52.0021	56.6218	50.5777	56.8245	51.2563	55.9456	51.1882	56.415
dermatology	96.9839	98.0556	97.2646	98.3333	93.7286	96.6667	97.2764	97.7778	98.0769	98.0556	98.0769	97.7778
ecoli	81.8452	84.2657	84.8214	86.5967	87.5	81.0256	85.7143	87.0396	85.119	85.7343	84.8214	83.007
energy-y1	89.0625	88.2331	85.8073	89.5363	88.5417	89.0226	88.6719	88.4962	83.9844	88.8659	83.9844	89.1541
energy-y2	88.8021	88.891	87.7604	89.4173	88.9323	88.7594	87.5	88.7594	86.7187	89.1541	86.0677	89.1541
flags	50.0208	49.3364	47.9167	51.5332	35.0417	50.2975	46.875	51.968	51	51.8764	49.4583	51.8764
glass	65.3345	66.8762	57.9074	65.9238	63.5678	60.2095	61.5437	62.5905	62.5214	63.9429	61.1407	64.5714
heart-cleveland	57.7179	60.4848	59.7436	59.4242	59.0641	60.1212	59.0513	61.4545	60.7308	60.1212	59.4231	61.7576
heart-switzerland	46.0606	44.5	43.7879	42.8333	41.0606	42	41.0606	46.1667	42.7273	45.3333	41.1364	44.5
heart-va	32	34	30	32.5	29	32.5	31	33.5	28.5	34.5	28.5	37
iris	95.9806	97.3333	93.3818	96	98.0423	97.3333	96.6909	96	90.6791	96.6667	90.6791	97.3333
led-display	72.8	73.4	72.5	73.1	72.8	73.6	71.6	74.1	72.7	74.1	72.5	74.3
lenses	87.5	88.3333	87.5	83.3333	83.3333	71.6667	87.5	88.3333	91.6667	73.3333	91.6667	78.3333
libras	73.8889	75.2778	56.3889	80.8333	45.5556	74.7222	62.7778	79.4444	64.1667	81.3889	56.6667	79.4444
low-res-spect	83.6237	87.5856	85.7155	88.5255	73.2407	85.7163	82.6936	88.1481	87.3864	88.3368	82.1086	88.1481
lung-cancer	43.75	48.6667	34.375	54	37.5	35.3333	37.5	35.3333	50	38.6667	43.75	45.3333

Continued on next page

Table 8.3 – continued from previous page

Dataset	RVFL	RVFL-AE	TBSVM	TBSVM-FL*	TWKSVC	TWKSVC-FL*	LSTWKSVC	LSTWKSVC-FL*	RELSTSVM	RELSTSVM-FL*	LSTSVM	LSTSVM-FL*
lymphogra-phy	85.8108	82.987	64.8649	84.8701	83.1081	88.1818	83.7838	86.4935	66.2162	83.4416	66.2162	86.2987
oocytes-merluccius-states-2f	92.1767	91.8722	92.6646	92.2662	92.6646	91.8722	90.6134	91.776	92.1759	92.3624	92.1752	91.5818
oocytes-trisopterus-states-5b	84.6491	91.9875	93.0921	92.5393	92.9825	88.0385	91.6667	91.8799	92.2149	92.6492	92.1053	90.1241
pittsburg-bridges-MATERIAL	84.0659	86.125	86.8132	86.125	84.1346	85.125	85.8516	87.125	85.0275	87.125	85.0275	84.125
pittsburg-bridges-REL-L	69.75	67.9231	69.75	68.3846	70.75	67.1538	71.9643	65.3846	69.75	69.1538	68.8571	70.6923
pittsburg-bridges-SPAN	68.4783	73.9394	67.3913	72.8283	63.0435	74.8485	63.0435	72.8283	65.2174	72.8283	65.2174	73.9394
pittsburg-bridges-TYPE	58.084	58	57.0869	60.6667	57.0869	61.6667	60.8974	56.3333	58.0484	62.3333	58.0484	61
primary-tumor	45.6809	46.0606	43.2491	47.5758	42.378	44.5455	46.3124	46.0606	46.9077	46.6667	46.0003	46.3636
seeds	92.3967	95.7143	96.1895	96.6667	96.2251	93.8095	94.3198	96.6667	97.1688	96.1905	97.1688	97.619
statlog-vehicle	72.924	77.2698	77.783	79.4127	79.2037	74.8333	77.4364	77.8571	78.372	77.6111	78.2536	76.6746
synthetic-control	95.5	94.8333	88.1667	96.8333	89.5	96.1667	87.5	97.1667	85.6667	96.3333	82.8333	96.5
teaching	55.0676	55.7083	55.1689	58.3333	53.0912	55.0417	53.8176	58.375	54.4932	57	54.4932	55.0417

Continued on next page

Table 8.3 – continued from previous page

Dataset	RVFL	RVFL-AE	TBSVM	TBSVM-FL*	TWKSVC	TWKSVC-FL*	LSTWKSVC	LSTWKSVC-FL*	RELSTSVM	RELSTSVM-FL*	LSTSVM	LSTSVM-FL*
vertebral-column-3clases	81.0414	83.5484	75.2137	82.9032	85.2293	81.2903	81.0332	83.871	83.9388	84.5161	83.9306	82.2581
wine	98.3202	100	97.752	99.0118	97.2085	98.8235	97.2332	98.4235	97.752	99.4118	97.752	98.2353
zoo	94.1154	95.1818	93.1154	96.0909	91.1154	94.0909	95.1154	96.0909	93.1154	95.1818	91.1923	94.0909
Average-Accuracy	74.2539	75.6174	72.2579	76.2808	72.2061	73.8851	73.3446	75.7376	73.471	75.6584	72.4694	75.5637
Average Rank	7.3939	5.6667	8.2879	3.8788	7.6515	7.3182	8.3788	4.7576	7.0152	4.0455	8.3788	5.2273
Overall-Win-Tie-Loss	0-0-5	2-0-0	1-0-4	9-0-0	4-0-7	2-0-3	1-0-3	4-0-2	0-0-0	2-0-0	0 -0-3	4-0-0

Here, * denotes the proposed methods.

8.3.4 Statistical analysis based on the Friedman test

The objective here is to perform the statistical analysis of the proposed models and the baseline models. It ensures whether the improvement observed in different models is statistically significant or not.

8.3.4.1 Random weighted models

The performance of the models corresponding to different datasets is given in Table 8.2. On each dataset, the classification models are ranked based on their performance with the lower performing model being assigned higher rank and the higher performing model being assigned lower rank. Average rank of a classifier across all the datasets is given as the rank of the classifier.

With simple calculations, the average ranks of RVFL, TB-SVM, RV-TBSVM, TWKSVC, RV-TWKSVC, LSTWKSVC, RV-LSTWKSVC, RELSTSVM, RV-RELSTSVM, LSTSVM and RV-LSTSVM are 6.8182, 7.7121, 3.9242, 7.7576, 6.1818, 7.8182, 3.7879, 6.8182, 2.2727, 7.9545 and 4.9545, respectively. Then based on simple calculations, we get $\chi_F^2 = 116.1159$, $F_F = 17.3725$ while evaluating the 11 algorithms on 33 datasets, F_F is distributed according to the F-distribution with $11 - 1 = 10$ and $(11 - 1)(33 - 1) = 320$ degrees of freedom. Since $F_{(10,320)}$ for $\alpha = 0.05$ is 1.8550. So, we reject the null hypothesis. With $\alpha = 0.05$, the critical value based on Nemenyi test is given as:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 3.219 \times \sqrt{\frac{11 \times 12}{6 \times 33}} = 2.6283.$$

Table 8.4 shows the statistical significance of different classification models. From the given table, one can see that the proposed approach based models are significantly better compared to other models. The proposed approach based models achieved better accuracy and lower average rank in comparison to the given baseline models.

Table 8.4: Statistical comparison of RVFL [202], TBSVM [233], TWKSVC [294], LSTWKSVC [191], RELSTSVM [260], LSTSVM [144] and proposed classification models.

Method	RVFL (6.8182)	TBSVM (7.7121)	RV-TBSVM* (3.9242)	TWKSVC (7.7576)	RV-TWKSVC* (6.1818)	LSTWKSVC (7.8182)	RV-LSTWKSVC* (3.7879)	RELSTSVM (6.8182)	RV-RELSTSVM* (2.2727)	LSTSVM (7.9545)	RV-LSTSVM* (4.9545)
RVFL (6.8182)											
TBSVM (7.7121)											
RV-TBSVM* (3.9242)	✔	✔		✔		✔		✔		✔	
TWKSVC (7.7576)											
RV-TWKSVC* (6.1818)											
LSTWKSVC (7.8182)											
RV-LSTWKSVC*											

Continued on next page

Table 8.4 – continued from previous page

Method	RVFL (6.8182)	TBSVM (7.7121)	RV-TBSVM* (3.9242)	TWKSVC (7.7576)	RV-TWKSVC* (6.1818)	LSTWKSVC (7.8182)	RV-LSTWKSVC* (3.7879)	RELSTSVM (6.8182)	RV-RELSTSVM* (2.2727)	LSTSVM (7.9545)	RV-LSTSVM* (4.9545)
(3.7879)	✔	✔		✔		✔		✔		✔	
RELSTSVM (6.8182)											
RV-RELSTSVM* (2.2727)	✔	✔		✔	✔	✔		✔		✔	✔
LSTSVM (7.9545)											
RV-LSTSVM* (4.9545)		✔		✔		✔				✔	

Here, * denote the proposed methods.

8.3.4.2 Twin weighted models

The performance of the models corresponding to different datasets is given in Table 8.3. We proceed in the same manner to rank the classifiers. The average ranks of RVFL, RVFL-AE , TBSVM, TBSVM-FL, TWKSVC, TWKSVC-FL, LSTWKSVC, LSTWKSVC-FL, RELSTSVM, RELSTSVM-FL, LSTSVM and LSTSVM-FL are 7.3939, 5.6667, 8.2879, 3.8788, 7.6515, 7.3182, 8.3788, 4.7576, 7.0152, 4.0455, 8.3788 and 5.2273, respectively. The Friedman statistics are $\chi_F^2 = 80.1246$ and $F_F = 9.0640$ with 12 algorithms evaluated on 33 datasets. F_F is distributed according to the F-distribution with $(12-1) = 11$ and $(12-1)(33-1) = 352$ degrees of freedom. For $\alpha = 0.05$, $F_{(11\ ,352)}$ is 1.78. Hence, we reject the null hypothesis. Using simple calculations with $\alpha = 0.05$, critical difference is given as $CD = 2.9008$.

Table 8.5: Statistical comparison of RVFL [202], RVFL-AE [311], TBSVM [233], TWKSVC [294], LSTWKSVC [191], RELSTSVM [260], LSTSVM [144] and proposed classification models.

Method	RVFL (7.3939)	RVFL-AE (5.6667)	TBSVM (8.2879)	TBSVM-FL* (3.8788)	TWKSVC (7.6515)	TWKSVC-FL* (7.3182)	LSTWKSVC (8.3788)	LSTWKSVC-FL* (4.7576)	RELSTSVM (7.0152)	RELSTSVM-FL* (4.0455)	LSTSVM (8.3788)	LSTSVM-FL* (5.2273)
RVFL 7.3939												
RVFL-AE 5.6667												
TBSVM 8.2879												
TBSVM-FL* 3.8788	✔		✔		✔	✔	✔		✔		✔	
TWKSVC 7.6515												
TWKSVC-FL* 7.3182												
LSTWKSVC 8.3788												
LSTWKSVC-FL* 4.7576			✔				✔				✔	
RELSTSVM												

Continued on next page

Table 8.5 – continued from previous page

Method	RVFL (7.3939)	RVFL-AE (5.6667)	TBSVM (8.2879)	TBSVM-FL* (3.8788)	TWKSVC (7.6515)	TWKSVC-FL* (7.3182)	LSTWKSVC (8.3788)	LSTWKSVC-FL* (4.7576)	RELSTSVM (7.0152)	RELSTSVM-FL* (4.0455)	LSTSVM (8.3788)	LSTSVM-FL* (5.2273)
7.0152												
RELSTSVM-FL* 4.0455	✔		✔		✔	✔	✔		✔		✔	
LSTSVM 8.3788												
LSTSVM-FL* 5.2273			✔				✔				✔	

Here, * denote the proposed methods.

Table 8.6: Significant difference between TBSVM, RV-TBSVM, TWKSVC, RV-TWKSVC, LSTWKSVC, RV-LSTWKSVC, RELSTSVM, RV-RELSTSVM, LSTSVM and RV-LSTSVM based on pair-wise sign test.

Method	Significance
(RV-TBSVM, TBSVM) (28,4)	✔
(RV-TWKSVC, TWKSVC) (21,11)	
(RV-LSTWKSVC, LSTWKSVC) (28,3)	✔
(RV-RELSTSVM, RELSTSVM) (29,3)	✔
(RV-LSTSVM, LSTSVM) (23,9)	✔

The meaning of two pairs of brackets $(A, B)(m, n)$ in each row means that while evaluating the methods (A, B) pairwise, method A performs better in m number of datasets while as method B performs better in n number of datasets. Corresponding to each pairwise test, ✔ means significant difference exists between the given methods. Blank entries denote that no significant difference exists among the given models.

Table 8.5 shows the statistical significance of the classification algorithms. From the given table, it is evident that the proposed approach based models show better performance and are significantly better compared to other models. The proposed approach based models achieved better average accuracy and lower average rank as compared to other classifier models.

8.3.5 Win-tie-loss: sign test

Sign test or the pairwise comparison of algorithms is used here to access the overall performance of classifiers. Here, the number of datasets in which a given algorithm is absolute winner among the given algorithms is counted.

8.3.5.1 Random weighted models

From the Table 8.6, it is evident that significant difference exist between each pairs of classifiers (except RV-TWKSVC,TWKSVC), with the proposed models emerging as the overall winner in most of the datasets.

8.3.5.2 Twin weighted models

From the Table 8.7, it is evident that significant difference exist between each pairs of classifiers (except TWKSVC-FL, TWSKVC), with the proposed models emerging as the

Table 8.7: Significant difference between RVFL, RVFL-AE, TBSVM, TBSVM-FL, TWKSVC, TWKSVC-FL, LSTWKSVC, LSTWKSVC-FL, RELSTSVM, RELSTSVM-FL, LSTSVM and LSTSVM-FL based on pair-wise sign test.

Method	Significance
(RVFL-AE, RVFL)(23, 9)	✔
(TBSVM-FL, TBSVM)(24, 8)	✔
(TWKSVC-FL, TWSKVC)(19, 13)	
(LSTWKSVC-FL, LSTWKSVC)(26, 6)	✔
(RELSTSVM-FL, RELSTSVM)(24, 8)	✔
(LSTSVM-FL, LSTSVM)(23, 9)	✔

The meaning of two pairs of brackets $(A, B)(m, n)$ in each row means that while evaluating the methods (A, B) pairwise, method A performs better in m number of datasets while as method B performs better in n number of datasets. Corresponding to each pairwise test, ✔ means significant difference exists between the given methods. Blank entries denote that no significant difference exists among the given models.

overall winner in most of the datasets.

8.3.6 Analyzing the effect of enhanced patterns

In this subsection, we will analyze the effect of enhanced number of features on the performance of the classification models.

8.3.6.1 Random-weighted models

Figure (8.3(a)) to Figure (8.3(h)) signify the effect of enhanced features on the performance of given classification models.

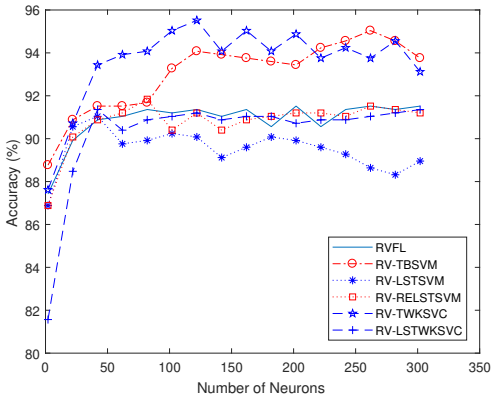
From Figures (8.3(a)), (8.3(c)) and (8.3(d)), one can see that increasing the number of enhanced patterns leads to better generalization of the proposed models. Thus, adding the number of enhanced features upto a certain level increases the generalization ability of the proposed models. Also, one can see that RVFL model in Figures (8.3(c)) and (8.3(d)) achieved lower performance as compared to the proposed models.

Figure (8.3(b)) shows the consistent performance of the proposed models with the varying number of enhanced features. The reason could be that the hidden features doesn't increase the generalization ability of the model.

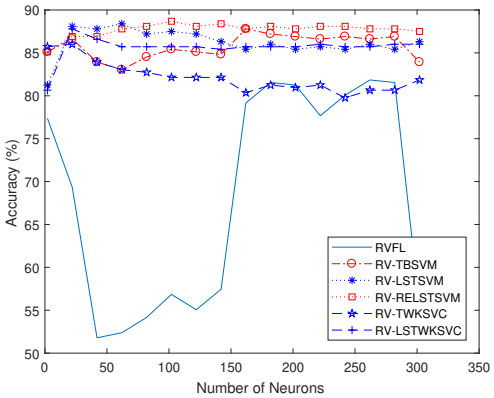
Figure (8.3(e)) indicates that the RVFL model outperforms the proposed models across all the number of enhanced patterns.

Figure 8.3: Effect of enhanced features on the performance of proposed random weighted classification models and baseline models.

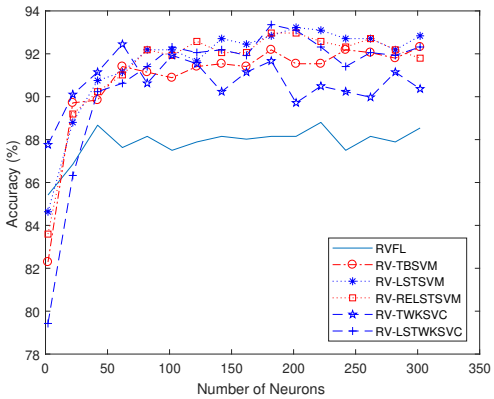
(a) Balance-scale.



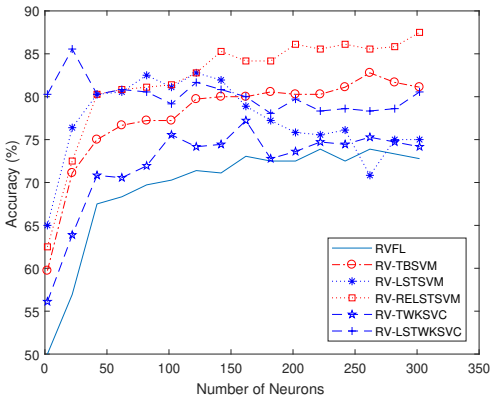
(b) Ecoli.

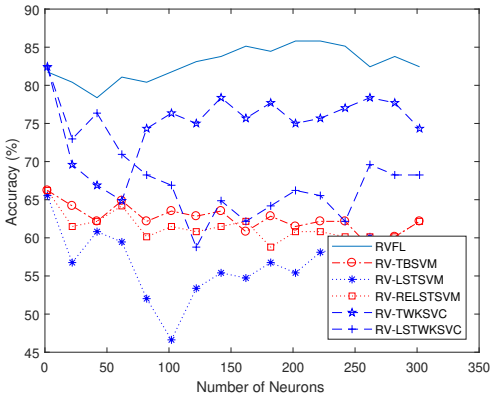


(c) Energy-y2.

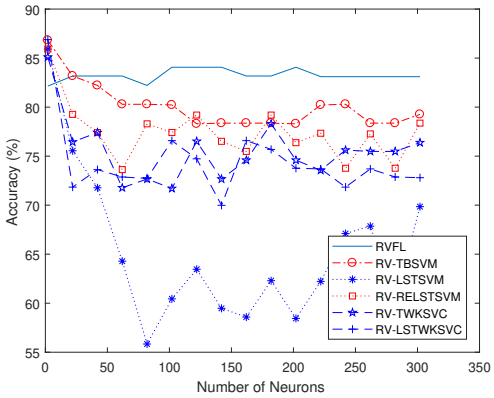


(d) Libras.

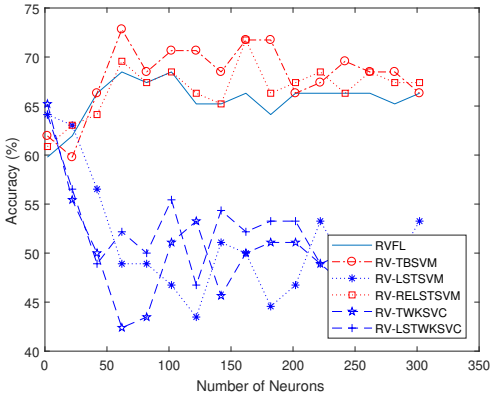




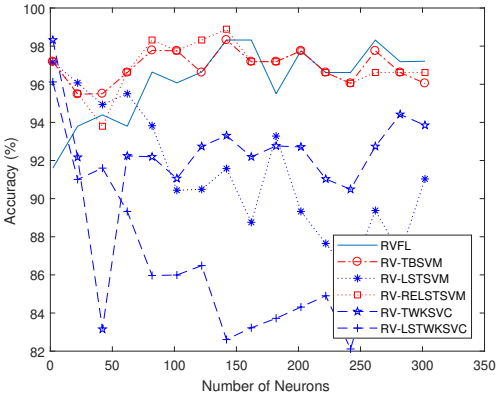
(e) Lymphography.



(f) Pittsburg-bridges-MATERIAL.



(g) Pittsburg-bridges-SPAN.



(h) wine.

In Figure (8.3(f)), it is evident that initially proposed approach shows higher performance compared to the RVFL model. However, as enhanced features increase the performance of RVFL model remains consistent while as that of proposed models decreases as the number of hidden features increase.

Figure (8.3(g)) shows that the RVFL model performs lower than the RV-TBSVM and RV-RELSTSVM when the number of hidden neurons is large. Other models (RV-LSTSVM, RV-TWKSVC, and RV-LSTWKSVC) have lower performance as compared to the RVFL model. Initially, the performance of RVFL model is lower in comparison with other models.

In Figure (8.3(h)), one can see the that the proposed RV-TBSVM and RV-RELSTSVM achieves better performance as compared to the RVFL across all the number of hidden features. However, the RVFL model performs better as compared to the RV-LSTSVM, RV-TWKSVC and RV-LSTWKSVC models.

8.3.6.2 Twin-weighted models

Figures (8.4(a)), (8.4(c)) and (8.4(d)) show that the performance is initially lower for lower number of enhanced features and increases with the increase in number of enhanced features. However, in Figure (8.4(b)) the performance of the models is almost consistent across all the number of enhanced features except TWKSVC-FL, where performance is initially higher at lower number of hidden neurons and then decreases as enhanced features increase and again increases with increase of enhanced features.

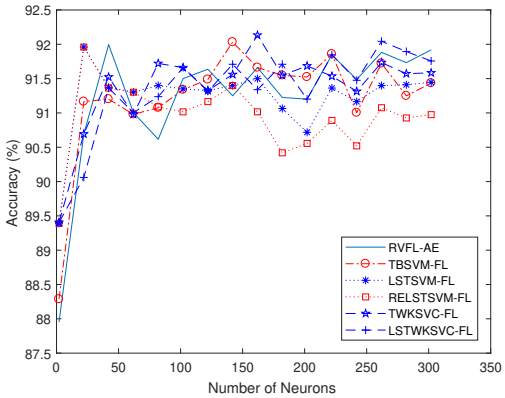
Figure (8.4(e)) shows that increasing the number of enhanced features improves the performance of the models while as Figure (8.4(g)) shows that increasing the number of enhanced features decreases the performance and the different models show varying performance across varying number of enhanced features.

In Figure (8.4(f)), the performance is initially lower and the performance increases initially with the increase in number of enhanced features and remains almost consistent after 100 number of hidden features.

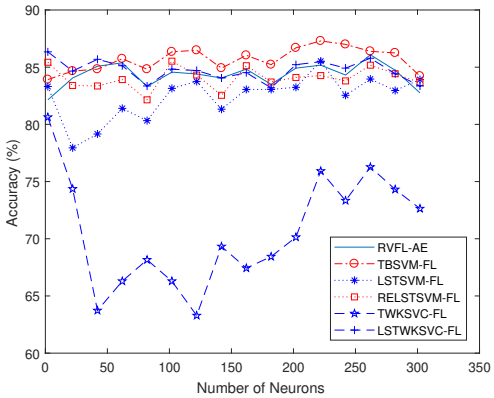
In Figure (8.4(h)), TBSVM-FL and RELSTSVM-FL show almost consistent performance across all the number of enhanced features while as the performance of RVFL-AE is low initially and increases as the enhanced features increase. One can observe that the performance of LSTSVM-FL and LSTWKSVC-FL is initially higher and decreases as the number of enhanced features increase while as the performance of TWKSVC-FL initially decreases and

Figure 8.4: Effect of enhanced features on the performance of proposed twin weighted classification models and baseline models.

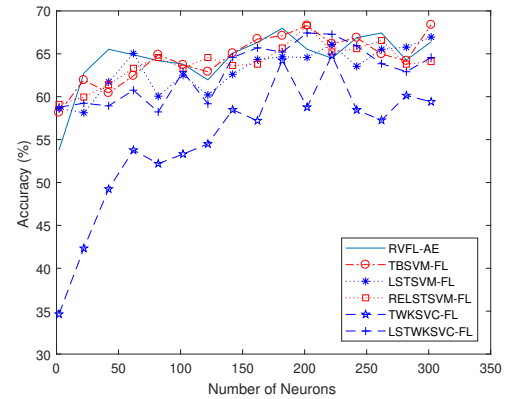
(a) Balance-scale.



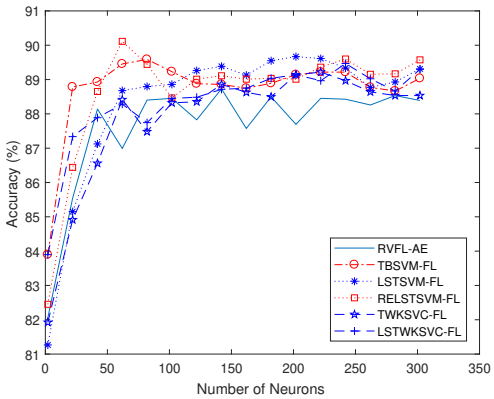
(b) Ecoli.

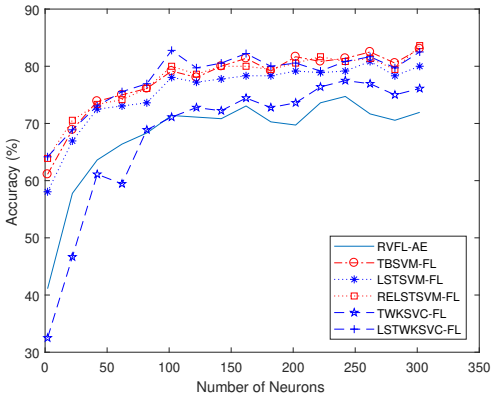


(c) Glass.

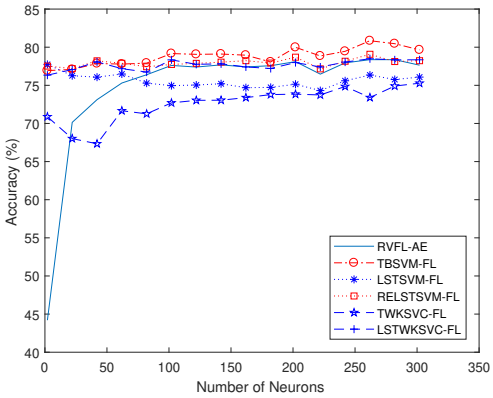


(d) Energy-y1.

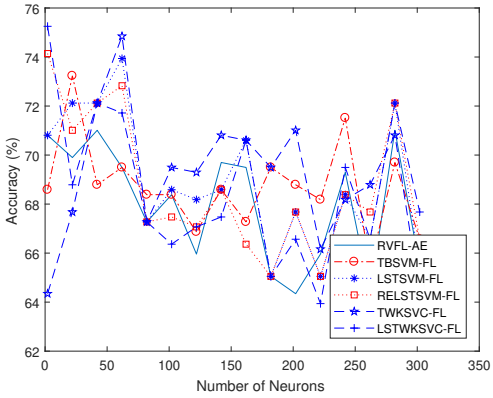




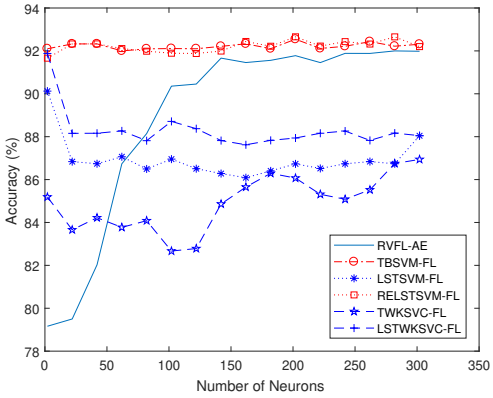
(e) Libras.



(f) Statlog-vehicle.



(g) Pittsburg-bridges-SPAN.



(h) Oocytes-trisopterus-states-5b.

then increases again with the increase in number of enhanced features.

Hence, to get the better generalization performance the number of hidden neurons need to be selected optimally.

8.4 Summary

In this chapter, we presented two approaches for classification problems. The first architecture is based on the random weighted extended feature space and solves different optimization problems corresponding to the models based on these extended feature spaces. Based on the extended features obtained via random weights, one can see that the performance of the proposed models is increased. The performance improvement of the RV-TBSVM and RV-TWKSVC models achieved approximately 8% increase in balance-scale, energy-y1 dataset and 6% increase by RV-LSTWKSVC in energy-y2 dataset as compared to their corresponding base models. The proposed RV-RELSTSVM achieved lowest average rank and emerged as the overall winner in most of the datasets (11 datasets). In the second proposed method, different models are used to generate the hidden layer weights of the architecture and weights in the output layer are optimized via closed form solution. The proposed TBSVM-FL achieved highest accuracy and lower rank in comparison with other models. All the baseline models and the proposed models are evaluated on 33 benchmark datasets from the UCI repository and fecundity estimation of fisheries datasets. The experimental results show that the average accuracy of the proposed classification models is better as compared to the baseline models. Moreover, both the models avoid the computation and memory issues of kernel trick as the presented models are efficient compared to the kernel trick based projections. We analyzed the performance of the classifiers based on varying number of enhanced features and conclude that the number of hidden neurons need to be chosen optimally for better performance of the classification models.

The model presented in this chapter and the previous chapter provided an alternate for the non-linear projections, which resulted in better control over complexity and memory issues.

Chapter 9

Conclusions and future work

In this thesis, we focused on random forest (RaF), random vector functional link network (RVFL) and their ensembles. All these algorithms have been successfully employed in diverse domain of applications. Random forest is an ensemble of decision trees. Each decision tree is a sequence of If-Then rules which are intuitive to humans. It is composed of terminal and non-terminal nodes, here terminal nodes give the classes of the samples while as each non-terminal evaluates the best split. Based on the split functions used in the non-terminal nodes of decision trees, there are two categories: axis-parallel and oblique decision trees. In this thesis, we have explored axis parallel, oblique decision trees and their ensembles. In neural networks, we have focused on randomized neural networks, more specifically RVFL which is a shallow neural network architecture composed of input layer, hidden layer and the output layer. The weights of the hidden layer are initialized randomly and kept fixed while as the output layer weights are optimized via closed form method. In the context of support vector machine (SVM), we have focused on the improvement of the twin SVM based models. We proposed the variants of twin SVM to overcome the issues of memory and time. The summary of this thesis is as follows:

9.1 Conclusions

[1] Literature review: We presented the comprehensive reviews of ensemble deep learning strategies and twin SVM based models. We explored the different strategies followed in the literature for ensembling the models. Furthermore, we also presented the review of twin SVM based models.

[2] Oblique ensembles of decision tree: We presented a novel approach to generate the decision tree ensembles with different base models like RaF, rotation forest (RoF) and random sub-rotation forest (RRoF). Here, the splitting plane is decided based on multivariate features in each non-leaf node. This splitting is based on the hyperplanes generated by twin bounded support vector machine (TBSVM). Unlike multi-surface proximal support vector machine (MPSVM) based oblique decision trees, the proposed twin bounded random forest (TBRaF), twin bounded rotation forest (TBRoF) and twin bounded random sub-rotation forest (TBRRoF) require no explicit regularization techniques. This is due to the reason that in TBSVM based oblique decision trees, the matrices appearing in the dual formulation are positive-definite. Also, the structural risk minimization principle is implemented in the proposed models. The proposed models show consistent performance with all the three baseline methods (RaF, RoF, RRoF). Among all the models, the proposed TBRaF and TBRRoF models emerged as the overall winner in 21 datasets in terms of accuracy while as other models show lower number of overall wins (less than or equal to 9 datasets). Also, average accuracy of the proposed TBRaF and TBRRoF is 84% while as all the other variants of RaF, RoF and RRoF have less than 84% accuracy. The proposed TBRaF, TBRoF and TBRRoF show consistent performance with different base classifiers.

[3] Oblique and rotation based ensembles of double random forest: We presented two approaches for generating the double random forest models. In the first approach, we presented oblique double random forest and in the second approach, we presented rotation based double random forest. In oblique double random forest models, the splitting hyperplane at each non-leaf node is generated via MPSVM. This leads to the incorporation of geometric structure and hence, leads to better generalization performance. As the decision tree grows, the problem of sample size may arise. Hence, we use Tikhonov regularization and axis parallel split regularization for generating decision trees to full depth. In rotation based double random forest models, we used two transformations - principal component analysis and linear discriminant analysis on randomly chosen feature subspace at each non-leaf node. Rotations on different random features subspace leads to more diverse decision tree ensemble and hence, results in better generalization performance. Unlike standard random forest where

the bootstrap aggregation is used at root node only, the proposed oblique and rotation double random forest use bootstrap aggregation at each non-terminal node for choosing the best split and then the original samples are sent down the decision tree. Experimental results and the statistical analysis show the efficacy of the proposed oblique and rotation double random forest over standard baseline classifiers.

[4] RVFL, ensemble of RVFL: We presented variance embedded RVFL and co-trained RVFL model. We propose total variance minimization based RVFL (Total-Var-RVFL) and intraclass variance minimization based RVFL (Class-Var-RVFL). The proposed methods exploit the training data dispersion in the original feature space as well as the randomized feature space while optimizing the output layer weights. Experimental analysis revealed that the incorporation of total variance and class variance improved the generalization performance of the proposed RVFL based models. In comparison to given baseline models, the proposed Total-Var-RVFL and Class-Var-RVFL models achieved better average accuracy. Also, the average rank of the proposed Class-Var-RVFL is better than other baseline models except minimum class variance extreme learning machine (MCVELM).

In co-trained RVFL (coRVFL), two RVFL models are trained jointly such that each RVFL model is predicting the target label as closely as possible. Since, two RVFL models are trained on different feature representations and hence, it is unlikely for the outcome of two models to agree on a particular data sample resulting in forcing the less accurate model to be as close as possible to the more accurate model. Experiments conducted on publicly available datasets show that the proposed coRVFL is better as compared to the baseline models. Furthermore, statistical analysis show that the proposed coRVFL-avg is statistically significantly better compared to the baseline models.

[5] Learning using privileged information (LUPI) framework for deep RVFL and its ensemble: We proposed deep RVFL using privileged information (dRVFL+) and ensemble deep RVFL using privileged information (edRVFL+) for the diagnosis of Alzheimer's disease. Standard RVFL, deep architectures of RVFL and its ensemble versions have shown better generalization performance, however, these models use only normal information for optimizing the network parameters. The proposed dRVFL+ and edRVFL+

are enabled to incorporate the privileged information, which is sidelined by the standard RVFL and its deep models. Both dRVFL+ and edRVFL+ efficiently utilize the privileged information in combination with the original features to get better generalization performance. Unlike the standard LUPI based models (like RVFL+) which normally utilize the half of the available features as normal features and rest as the privileged features. We proposed a novel approach for the generation of privileged information. We utilize different activation functions while processing the normal and privileged information in the proposed deep architectures. To the best of our knowledge, this is first time that all the features are utilized in the LUPI framework and a separate information is generated as the privileged information. The proposed dRVFL+ and edRVFL+ models are employed for the diagnosis of Alzheimer's disease. Experiments demonstrate the superiority of the proposed dRVFL+ and edRVFL+ models over baseline models.

[6] Improvements of twin SVM based algorithms: We proposed two approaches for twin SVM: pre-trained functional link network based twin SVM and ensemble of classification models with weighted functional link network. Pre-trained functional link network based twin SVM is an improved model for the classification problems using the least squares twin support vector machine (LSTSVM) and enhanced features from the pre-trained functional link network. Inspired by the direct link benefits and impact of randomization range on the performance of the RVFL. We proposed a model with LSTSVM as the basic unit for classification. The proposed enhanced feature based least squares twin support vector machine (ELSTSVM) provides the advantages of implicit feature representation. The original input data is fed into the input layer. At hidden layer, LSTSVM generates the weights and a non-linear function is applied to get the enhanced feature space. Finally, the optimal decision boundary is generated by LSTSVM based on the extended features (input features + enhanced features) to get the final labels. In oocytes-merluccius-nucleus-4d dataset, the proposed ELSTSVM achieved approximately 12% and 4% increase in accuracy with respect to RVFL and LSTSVM model, respectively. In statlog-image dataset, the proposed ELSTSVM achieved approximately 10% increase in accuracy with respect to both the baseline models. Furthermore, the proposed ELSTSVM uses lesser number of

hidden neurons and achieves better or comparable performance as that of RVFL network. The numerical experiments and the statistical tests show that the proposed ELSTSVM gives improved performance as compared to RVFL and LSTSVM models. From experiments, one can see that the enhanced features in combination with the original features improved the classification performance.

In ensemble of classification models with weighted functional link network, we presented two architectures for classification problems. The first architecture is based on the random weighted extended feature space and solves different optimization problems corresponding to the models based on these extended feature spaces. Based on the extended features obtained via random weights, one can see that the performance of the proposed models is increased. The performance improvement of the random vector TBSVM (RV-TBSVM) and random vector TWKSVC (RV-TWKSVC) models achieved approximately 8% increase in balance-scale, energy-y1 dataset and 6% increase by random vector least square TWKSVC (RV-LSTWKSVC) in energy-y2 dataset as compared to their corresponding base models. The proposed RV-RELSTSVM achieved lowest average rank and emerged as the overall winner in most of the datasets (11 datasets). In the second proposed method, different models are used to generate the hidden layer weights of the architecture and weights in the output layer are optimized via closed form solution. The proposed TBSVM-FL achieved highest accuracy and lower rank in comparison with existing models. All the baseline models and the proposed models are evaluated on 33 benchmark datasets from the UCI repository and fecundity estimation of fisheries datasets. The experimental results show that the average accuracy of the proposed classification models is better as compared to the baseline models.

9.2 Future directions

In this section, we present the possible future directions emanating from this thesis.

[1] Efficient oblique random forest: The oblique random forest via TBSVM presented in this thesis solves QPPs at each node, however, one can explore the efficient variants to generate the oblique hyperplane and hence, improve the generalization process of the models.

[2] Robust and diverse oblique double random forest: We presented oblique double random forest with MPSVM based hyperplane being used for splitting the nodes. However, limiting the splitting planes to one single classifier may results in suboptimal performance. Hence, one can explore the use of multiple linear classifiers like SVM, LDA so that more diverse splits results in better generalization performance.

[3] Robust RVFL: The RVFL uses l_2 norm which may suffer in presence of noise and outliers. Hence, one can explore the other loss functions like l_1 loss or pinball loss based models which are robust to noise and outliers.

[4] Diverse deep RVFL: In this thesis, we proposed deep RVFL and ensemble deep RVFL model with LUPI framework. One can explore the negative correlation framework based RVFL and its ensembles for both shallow and deep models. For improving the generalization performance of the models, diversity is of utmost importance, hence, one can explore diversity inducing approaches like bagging, boosting etc.

[5] Applications of classification models in other domains: The classification models presented in this thesis have been evaluated on the benchmark UCI [60] datasets. Moreover, the deep RVFL with LUPI framework has shown better efficiency in diagnosing the AD. One can explore the application of the presented models in regression, forecasting and other biomedical applications.

Bibliography

[1] Euijoon Ahn, Ashnil Kumar, Dagan Feng, Michael Fulham, and Jinman Kim. Unsupervised feature learning with k-means and an ensemble of deep convolutional neural networks for medical image classification. *arXiv preprint arXiv:1906.03359*, 2019.

[2] Nabil Alami, Mohammed Meknassi, and Noureddine En-nahnahi. Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning. *Expert Systems with Applications*, 123:195–211, 2019.

[3] Monther Alhamdoosh and Dianhui Wang. Fast decorrelated neural network ensembles with random weights. *Information Sciences*, 264:104–117, 2014.

[4] Ricardo F Alvear-Sandoval and Aníbal R Figueiras-Vidal. On building ensembles of stacked denoising auto-encoding classifiers and their further improvement. *Information Fusion*, 39:41–52, 2018.

[5] Terry Anderson. *The theory and practice of online learning*. Athabasca University Press, 2008.

[6] Robert E Banfield, Lawrence O Hall, Kevin W Bowyer, and W Philip Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2006.

[7] Iñigo Barandiaran. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8), 1998.

[8] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9368–9377, 2018.

[9] Yoshua Bengio. *Learning deep architectures for AI.* Now Publishers Inc, 2009.

[10] Kristin P Bennett, Ayhan Demiriz, and Richard Maclin. Exploiting unlabeled data in ensemble methods. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 289–296, 2002.

[11] Hugues Berry and Mathias Quoy. Structure and dynamics of random recurrent neural networks. *Adaptive Behavior*, 14(2):129–137, 2006.

[12] Alina Beygelzimer, Elad Hazan, Satyen Kale, and Haipeng Luo. Online Gradient Boosting. Technical report.

[13] Anil Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[14] Shun Bian and Wenjia Wang. On diversity and accuracy of homogeneous and heterogeneous ensembles. *International Journal of Hybrid Intelligent Systems*, 4(2):103–128, 2007.

[15] Jerzy Błaszczyński and Jerzy Stefanowski. Neighbourhood sampling in bagging for imbalanced data. *Neurocomputing*, 150:529–542, 2015.

[16] Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, Larry D Jackel, Yann LeCun, Urs A Muller, Edward Sackinger, Patrice Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, volume 2, pages 77–82. IEEE, 1994.

[17] Anne-Laure Boulesteix, Silke Janitza, Jochen Kruppa, and Inke R König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):493–507, 2012.

[18] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[19] Leo Breiman. Bias, variance, and arcing classifiers. 1996.

[20] Leo Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996.

[21] Leo Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801–849, 1998.

[22] Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242, 2000.

[23] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[24] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[25] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[26] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.

[27] Gavin Brown, Jeremy L Wyatt, and Peter Tiňo. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6(Sep):1621–1650, 2005.

[28] Peter Bühlmann, Bin Yu, et al. Analyzing bagging. *The Annals of Statistics*, 30(4): 927–961, 2002.

[29] Andreas Buja and Werner Stuetzle. Smoothing effects of bagging. *Preprint. AT&T Labs-Research*, 2000.

[30] Sebastian Buschjäger, Lukas Pfahler, and Katharina Morik. Generalized negative correlation learning for deep ensembling. *arXiv preprint arXiv:2011.02952*, 2020.

[31] Yue Cao, Thomas Andrew Geddes, Jean Yee Hwa Yang, and Pengyi Yang. Ensemble deep learning in bioinformatics. *Nature Machine Intelligence*, 2(9):500–508, 2020.

[32] Zhen Cao, Xiaoyong Pan, Yang Yang, Yan Huang, and Hong-Bin Shen. The lncLocator: a subcellular localization predictor for long non-coding RNAs based on a stacked ensemble classifier. *Bioinformatics*, 34(13):2185–2194, 2018.

[33] Miguel A Carreira-Perpinán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in Neural Information Processing Systems*, 31:1211–1221, 2018.

[34] Salvatore Carta, Andrea Corriga, Anselmo Ferreira, Alessandro Sebastian Podda, and Diego Reforgiato Recupero. A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Applied Intelligence*, pages 1–17, 2020.

[35] Sung-Hyuk Cha and Charles C Tappert. A genetic algorithm for constructing compact binary decision trees. *Journal of Pattern Recognition Research*, 4(1):1–13, 2009.

[36] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[37] Chang Chen, Zhiwei Xiong, Xinmei Tian, and Feng Wu. Deep boosting for image denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[38] Chang Chen, Zhiwei Xiong, Xinmei Tian, Zheng-Jun Zha, and Feng Wu. Real-world image denoising with deep boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[39] Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2377–2383. IEEE, 2017.

[40] Li-Fen Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, and Gwo-Jong Yu. A new LDA-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33(10):1713–1726, 2000.

[41] Xi-liang Chen, Lei Cao, Chen-xi Li, Zhi-xiong Xu, and Jun Lai. Ensemble network architecture for deep reinforcement learning. *Mathematical Problems in Engineering*, 2018, 2018.

[42] Hoi-Ming Chi and Okan K Ersoy. A statistical self-organizing learning system for remote sensing classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43(8):1890–1900, 2005.

[43] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[44] Cheng Chu, Sang Kyun Kim, Yian Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. *Advances in Neural Information Processing Systems*, 19:281, 2007.

[45] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE, 2012.

[46] Marquis de Condorcet. Essay on the application of analysis to the probability of majority decisions. *Paris: Imprimerie Royale*, 1785.

[47] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[48] Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In *31st International Conference on Machine Learning, ICML 2014*, 2014. ISBN 9781634393973.

[49] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. AdaNet: Adaptive Structural Learning of Artificial Neural Networks. Technical report, 2017.

[50] Heriberto Cuayáhuitl, Donghyeon Lee, Seonghan Ryu, Yongjin Cho, Sungja Choi, Satish Indurthi, Seunghak Yu, Hyungtak Choi, Inchul Hwang, and Jihie Kim. Ensemble-based deep reinforcement learning for chatbots. *Neurocomputing*, 366:118–130, 2019.

[51] Ron Tor Das, Kai Keng Ang, and Chai Quek. ierspop: A novel incremental rough set-based pseudo outer-product with ensemble learning. *Applied Soft Computing*, 46: 170 – 186, 2016. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2016.04.015. URL http://www.sciencedirect.com/science/article/pii/S1568494616301636.

[52] Yajnaseni Dash, Saroj Kanta Mishra, Sandeep Sahany, and Bijaya Ketan Panigrahi. Indian summer monsoon rainfall prediction: a comparison of iterative and non-iterative approaches. *Applied Soft Computing*, 70:1122–1134, 2018.

[53] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(Jan):1–30, 2006.

[54] L. Li Deng, Dong Yu, and John Platt. Scalable stacking and learning for building deep architectures. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, (February 2015):2133–2136, 2012. ISSN 15206149. doi: 10.1109/ICASSP.2012.6288333.

[55] Li Deng and Dong Yu. Deep convex net: A scalable architecture for speech pattern classification. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, (August):2285–2288, 2011. ISSN 19909772.

[56] Li Deng, Gokhan Tur, Xiaodong He, and Dilek Hakkani-Tur. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *2012 IEEE Workshop on Spoken Language Technology, SLT 2012 - Proceedings*, pages 210–215. IEEE, 12 2012. ISBN 9781467351263. doi: 10.1109/SLT.2012.6424224. URL http://ieeexplore.ieee.org/document/6424224/.

[57] Thomas G Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, pages 1–15. Springer, 2000.

[58] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1994.

[59] Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1994.

[60] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[61] F. Duan and L. Dai. Recognizing the gradual changes in sEMG characteristics based on incremental learning of wavelet neural network ensemble. *IEEE Transactions on Industrial Electronics*, 64(5):4276–4286, 2017. doi: 10.1109/TIE.2016.2593693.

[62] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[63] Elizabeth A Freeman, Gretchen G Moisen, John W Coulston, and Barry T Wilson. Random forests and stochastic gradient boosting for predicting tree canopy cover: comparing tuning processes and model performance. *Canadian Journal of Forest Research*, 46(3):323–339, 2016.

[64] Benoît Frénay and Michel Verleysen. Using SVMs with randomised feature spaces: an extreme learning approach. In *ESANN*, 2010.

[65] Yoav Freund and Robert E Schapire. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer, 1996.

[66] Yoav Freund and Robert E Schapire. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[67] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[68] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics New York, NY, USA:, 2001.

[69] Jerome H Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, 1997.

[70] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[71] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200): 675–701, 1937.

[72] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[73] M A Ganaie, M Tanveer, and P N Suganthan. Regularized robust fuzzy least squares twin support vector machine for class imbalance learning. In *2020 International Joint Conference on Neural Networks, IJCNN*, pages 1–8. IEEE, 2020.

[74] MA Ganaie, Saptarshi Ghosh, Naveen Mendola, M Tanveer, and Sarika Jalan. Identification of chimera using machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(6):063128, 2020.

[75] MA Ganaie, M Tanveer, and Alzheimer's Disease Neuroimaging Initiative. Fuzzy least squares projection twin support vector machines for class imbalance learning. *Applied Soft Computing*, 113:107933, 2021.

[76] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[77] Ramazan Gençay and Min Qi. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks*, 12(4):726–734, 2001.

[78] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[79] James S Goerss. Tropical cyclone track forecasts using an ensemble of dynamical models. *Monthly Weather Review*, 128(4):1187–1193, 2000.

[80] Sergio González, Salvador García, Javier Del Ser, Lior Rokach, and Francisco Herrera. A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. *Information Fusion*, 64:205–237, 2020.

[81] Encarnación González-Rufino, Pilar Carrión, Eva Cernadas, Manuel Fernández-Delgado, and Rosario Domínguez-Petit. Exhaustive comparison of colour texture features and classification methods to discriminate cells categories in histological images of fish ovary. *Pattern Recognition*, 46(9):2391–2407, 2013.

[82] Felix Grassmann, Judith Mengelkamp, Caroline Brandl, Sebastian Harsch, Martina E Zimmermann, Birgit Linkohr, Annette Peters, Iris M Heid, Christoph Palm, and Bernhard HF Weber. A deep learning algorithm for prediction of age-related eye disease study severity scale for age-related macular degeneration from color fundus photography. *Ophthalmology*, 125(9):1410–1420, 2018.

[83] Gabriela Grmanová, Peter Laurinec, Viera Rozinajová, Anna Bou Ezzeddine, M. Lucká, P. Lacko, Petra Vrablecová, and P. Návrat. Incremental ensemble learning for electricity load forecasting. 2016.

[84] Ping Guo. A vest of the pseudoinverse learning algorithm. *arXiv preprint arXiv:1805.07828*, 2018.

[85] Ping Guo, CL Philip Chen, and Yinguan Sun. An exact supervised learning for a three-layer supervised neural network. In *Proceedings of 1995 International Conference on Neural Information Processing*, pages 1041–1044, 1995.

[86] Xiaotong Guo, Fulin Liu, Ying Ju, Zhen Wang, and Chunyu Wang. Human protein subcellular localization with integrated source and multi-label ensemble classifier. *Scientific Reports*, 6:28087, 2016.

[87] Kyoungnam Ha, Sungzoon Cho, and Douglas MacLachlan. Response models based on bagging neural networks. *Journal of Interactive Marketing*, 19(1):17–30, 2005.

[88] Bohyung Han, Jack Sim, and Hartwig Adam. Branchout: Regularization for online ensemble tracking with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3356–3365, 2017.

[89] Shizhong Han, Zibo Meng, Ahmed Shehab Khan, and Yan Tong. Incremental boosting convolutional neural network for facial action unit recognition. *arXiv preprint arXiv:1707.05395*, 2017.

[90] Sunwoo Han and Hyunjoong Kim. On the optimal size of candidate feature set in random forest. *Applied Sciences*, 9(5):898, 2019.

[91] Sunwoo Han, Hyunjoong Kim, and Yung-Seop Lee. Double random forest. *Machine Learning*, 109(8):1569–1586, 2020.

[92] Alexander Hans and Steffen Udluft. Ensembles of neural networks for robust reinforcement learning. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 401–406. IEEE, 2010.

[93] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[94] Basma Hassan, Samir E Abdelrahman, Reem Bahgat, and Ibrahim Farag. Uests: An unsupervised ensemble semantic textual similarity method. *IEEE Access*, 7:85462–85482, 2019.

[95] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In *Advances in Neural Information Processing Systems*, pages 507–513, 1998.

[96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[97] Pablo A Henríquez and Gonzalo A Ruz. A non-iterative method for pruning hidden neurons in neural networks with random weights. *Applied Soft Computing*, 70:1109–1121, 2018.

[98] Daniel Hernández-Lobato, Gonzalo MartíNez-MuñOz, and Alberto Suárez. How large should ensembles of classifiers be? *Pattern Recognition*, 46(5):1323–1336, 2013.

[99] Shohei Hido, Hisashi Kashima, and Yutaka Takahashi. Roughly balanced bagging for imbalanced data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2(5-6):412–426, 2009.

[100] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[101] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[102] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[103] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[104] Torsten Hothorn, Berthold Lausen, Axel Benner, and Martin Radespiel-Tröger. Bagging survival trees. *Statistics in Medicine*, 23(1):77–91, 2004.

[105] Torsten Hothorn, Friedrich Leisch, Achim Zeileis, and Kurt Hornik. The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*, 14(3):675–699, 2005.

[106] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[107] Barbara FF Huang and Paul C Boutros. The parameter sensitivity of random forests. *BMC Bioinformatics*, 17(1):331, 2016.

[108] Dong Huang, Jianhuang Lai, and Chang-Dong Wang. Ensemble clustering using factor graph. *Pattern Recognition*, 50:131–142, 2016.

[109] Dong Huang, Chang-Dong Wang, and Jian-Huang Lai. Locally weighted ensemble clustering. *IEEE Transactions on Cybernetics*, 48(5):1460–1473, 2017.

[110] Furong Huang, Jordan T Ash, John Langford, and Robert E Schapire. Learning Deep ResNet Blocks Sequentially using Boosting Theory. Technical report, 2018. URL `https://arxiv.org/pdf/1706.04964.pdf`.

[111] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

[112] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get M for free. *arXiv preprint arXiv:1704.00109*, 2017.

[113] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2011.

[114] Po-Sen Huang, Li Deng, Mark Hasegawa-Johnson, and Xiaodong He. Random features for Kernel Deep Convex Network. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, number 2, pages 3143–3147. IEEE, 5 2013. ISBN 978-1-4799-0356-6. doi: 10.1109/ICASSP.2013.6638237. URL http://ieeexplore.ieee.org/document/6638237/.

[115] Dirk Husmeier and John G Taylor. Neural networks for predicting conditional probability densities: Improved training scheme combining EM and RVFL. *Neural Networks*, 11(1):89–116, 1998.

[116] Brian Hutchinson, Li Deng, and Dong Yu. A deep architecture with bilinear modeling of hidden representations: Applications to phonetic recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4805–4808. IEEE, 3 2012. ISBN 978-1-4673-0046-9. doi: 10.1109/ICASSP.2012.6288994. URL http://ieeexplore.ieee.org/document/6288994/.

[117] Brian Hutchinson, L. Li Deng, and Dong Yu. Tensor deep stacking networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1944–1957, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.268.

[118] Boris Igelnik and Yoh-Han Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995.

[119] Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas. Minimum class variance extreme learning machine for human action recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(11):1968–1979, 2013.

[120] Alexandros Iosifidis, Anastasios Tefas, and Ioannis Pitas. Minimum variance extreme learning machine for human action recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5427–5431. IEEE, 2014.

[121] Gareth M James. Variance and bias for general loss functions. *Machine Learning*, 51 (2):115–135, 2003.

[122] Jayadeva, R. Khemchandani, and S. Chandra. Twin support vector machines for pattern classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):905–910, 2007.

[123] Hongying Jiang, Youping Deng, Huann-Sheng Chen, Lin Tao, Qiuying Sha, Jun Chen, Chung-Jui Tsai, and Shuanglin Zhang. Joint analysis of two microarray gene-expression data sets to select lung adenocarcinoma marker genes. *BMC Bioinformatics*, 5(1):81, 2004.

[124] Xudong Jiang. Linear subspace learning-based dimensionality reduction. *IEEE Signal Processing Magazine*, 28(2):16–26, 2011.

[125] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, pages 137–142. Springer, 1998.

[126] Cheng Ju, Aurélien Bibaut, and Mark van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818, 2018.

[127] Cheng Ju, Mary Combs, Samuel D Lendle, Jessica M Franklin, Richard Wyss, Sebastian Schneeweiss, and Mark J van der Laan. Propensity score prediction for electronic healthcare databases using super learner and high-dimensional propensity score methods. *Journal of Applied Statistics*, 46(12):2216–2236, 2019.

[128] Tianyu Kang, Ping Chen, John Quackenbush, and Wei Ding. A novel deep learning model by stacking conditional restricted boltzmann machine and deep neural network. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1316–1324, 2020.

[129] Rakesh Katuwal and Ponnuthurai N Suganthan. Enhancing multi-class classification of random forest using random vector functional neural network and oblique decision surfaces. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[130] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[131] AS Khwaja, M Naeem, A Anpalagan, A Venetsanopoulos, and B Venkatesh. Improved short-term load forecasting using bagged neural networks. *Electric Power Systems Research*, 125:109–115, 2015.

[132] Zeynep H Kilimci and Selim Akyokus. Deep learning-and word embedding-based heterogeneous classifier ensembles for text classification. *Complexity*, 2018, 2018.

[133] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung-Yang Bang. Support vector machine ensemble with bagging. In *International Workshop on Support Vector Machines*, pages 397–408. Springer, 2002.

[134] Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757–2767, 2003.

[135] Keigo Kimura, Mineichi Kudo, Lu Sun, and Sadamori Koujaku. Fast random k-labelsets for large-scale multi-label classification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 438–443. IEEE, 2016.

[136] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 972–981, 2017.

[137] EM Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial intelligence*, 1(1):207–239, 1990.

[138] Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

[139] Ron Kohavi, David H Wolpert, et al. Bias plus variance decomposition for zero-one loss functions. In *ICML*, volume 96, pages 275–83, 1996.

[140] Eun Bae Kong and Thomas G Dietterich. Error-correcting output coding corrects bias and variance. In *Machine Learning Proceedings 1995*, pages 313–321. Elsevier, 1995.

[141] UH-G Krebel. Pairwise classification and support vector machines. *Advances in Kernel Methods: Support Vector Learning*, pages 255–268, 1999.

[142] Daniel Kreßner. Numerical methods and software for general and structured eigenvalue problems. 2004. doi: 10.14279/depositonce-983.

[143] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, pages 231–238, 1995.

[144] M Arun Kumar and Madan Gopal. Least squares twin support vector machines for pattern classification. *Expert Systems with Applications*, 36(4):7535–7543, 2009.

[145] Ludmila I Kuncheva and Juan J Rodriguez. Classifier ensembles with a random linear oracle. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):500–508, 2007.

[146] Ludmila I Kuncheva, Christopher J Whitaker, Catherine A Shipp, and Robert PW Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, 6(1):22–31, 2003.

[147] Vitaly Kuznetsov, Mehryar Mohri, and Umar Syed. Multi-class deep boosting. *Advances in Neural Information Processing Systems*, 3(January):2501–2509, 2014. ISSN 10495258.

[148] Avisek Lahiri, Abhijit Guha Roy, Debdoot Sheet, and Prabir Kumar Biswas. Deep neural ensemble for retinal vessel segmentation in fundus images towards achieving label-free angiography. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 1340–1343. IEEE, 2016.

[149] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[150] Michael Leblanc and Robert Tibshirani. Combining Estimates in Regression and Classification. *Journal of the American Statistical Association*, 91(436):1641–1650, 12 1996. ISSN 0162-1459. doi: 10.1080/01621459.1996.10476733. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.1996.10476733.

[151] Yuh-Jye Lee and Olvi L Mangasarian. RSVM: Reduced support vector machines. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–17. SIAM, 2001.

[152] Jichang Li, Si Wu, Cheng Liu, Zhiwen Yu, and Hau-San Wong. Semi-supervised deep coupled ensemble learning with classification landmark exploration. *IEEE Transactions on Image Processing*, 29:538–550, 2019.

[153] Jun Li, Heyou Chang, and Jian Yang. Sparse deep stacking network for image classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[154] Jun Li, Heyou Chang, Jian Yang, Wei Luo, and Yun Fu. Visual representation and classification by learning group sparse deep stacking network. *IEEE Transactions on Image Processing*, 27(1):464–476, 2017.

[155] Ping Li, Hong Li, and Min Wu. Multi-label ensemble based on variable pairwise constraint projection. *Information Sciences*, 222:269 – 281, 2013. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2012.07.066. URL http://www.sciencedirect.com/science/article/pii/S0020025512005385. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[156] Sheng Li, Xugang Lu, Shinsuke Sakai, Masato Mimura, and Tatsuya Kawahara. Semi-supervised ensemble DNN acoustic model training. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5270–5274. IEEE, 2017.

[157] Wei Li, Shuai Ding, Yi Chen, and Shanlin Yang. Heterogeneous ensemble for default prediction of peer-to-peer lending in china. *IEEE Access*, 6:54396–54406, 2018.

[158] Weitao Li, Dianhui Wang, and Tianyou Chai. Multisource data ensemble modeling for clinker free lime content estimate in rotary kiln sintering processes. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(2):303–314, 2014.

[159] Xin Li, Yu Yang, Haiyang Pan, Jian Cheng, and Junsheng Cheng. A novel deep stacking least squares support vector machine for rolling bearing fault diagnosis. *Computers in Industry*, 110:36–47, 9 2019. ISSN 01663615. doi: 10.1016/j.compind.2019.05.005. URL https://linkinghub.elsevier.com/retrieve/pii/S0166361519300533.

[160] Yuanqing Li and Cuntai Guan. Joint feature re-extraction and classification using an iterative semi-supervised support vector machine algorithm. *Machine Learning*, 71(1): 33–53, 2008.

[161] Andy Liaw and Matthew Wiener. Classification and regression by random forest. *R news*, 2(3):18–22, 2002.

[162] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of The American Statistical Association*, 101(474):578–590, 2006.

[163] Bo Liu, Lin Gu, and Feng Lu. Unsupervised ensemble strategy for retinal vessel segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 111–119. Springer, 2019.

[164] Hongfu Liu, Tongliang Liu, Junjie Wu, Dacheng Tao, and Yun Fu. Spectral ensemble clustering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 715–724, 2015.

[165] Hongfu Liu, Ming Shao, Sheng Li, and Yun Fu. Infinite ensemble for image clustering. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1745–1754, 2016.

[166] Jing Liu, Songzheng Zhao, and Gang Wang. SSEL-ADE: a semi-supervised ensemble learning framework for extracting adverse drug events from social media. *Artificial Intelligence in Medicine*, 84:34–49, 2018.

[167] Kun-Hong Liu and De-Shuang Huang. Cancer classification using rotation forest. *Computers in Biology and Medicine*, 38(5):601–610, 2008.

[168] Ping Liu, Shizhong Han, Zibo Meng, and Yan Tong. Facial expression recognition via a boosted deep belief network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1805–1812, 2014.

[169] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[170] Cheng-Yaw Low, Jaewoo Park, and Andrew Beng-Jin Teoh. Stacking-based deep neural network: Deep analytic network for pattern classification. *IEEE Transactions on Cybernetics*, 2019.

[171] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 681–688, 2009.

[172] Olvi L Mangasarian and Edward W Wild. Multisurface proximal support vector machine classification via generalized eigenvalues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):69–74, 2005.

[173] Naresh Manwani and PS Sastry. Geometric decision tree. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(1):181–192, 2011.

[174] Jianchang Mao. A case study on bagging, boosting and basic ensembles of neural networks for OCR. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 3, pages 1828–1833. IEEE, 1998.

[175] Dragos D Margineantu and Thomas G Dietterich. Pruning adaptive boosting. In *ICML*, volume 97, pages 211–218. Citeseer, 1997.

[176] Gonzalo Martínez-Muñoz and Alberto Suárez. Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition*, 43(1):143–152, 2010.

[177] Prem Melville and Raymond J Mooney. Constructing diverse classifier ensembles using artificial training examples. In *IJCAI*, volume 3, pages 505–510, 2003.

[178] Prem Melville and Raymond J Mooney. Diverse ensembles for active learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, page 74, 2004.

[179] Bjoern H Menze, B Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A Hamprecht. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics*, 10(1):213, 2009.

[180] Diego PP Mesquita, Joao Paulo P Gomes, Leonardo R Rodrigues, Saulo AF Oliveira, and Roberto KH Galvão. Building selective ensembles of randomization based neural

networks with the successive projections algorithm. *Applied Soft Computing*, 70:1135–1145, 2018.

[181] Leandro L Minku, Allan P White, and Xin Yao. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):730–742, 2009.

[182] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[183] Mohammad Moghimi, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. Boosted convolutional neural networks. In *BMVC*, pages 24–1, 2016.

[184] Alan Mosca and George D Magoulas. Deep incremental boosting. *arXiv preprint arXiv:1708.03704*, 2017.

[185] Jose M Moyano, Eva L Gibaja, Krzysztof J Cios, and Sebastián Ventura. An evolutionary approach to build ensembles of multi-label classifiers. *Information Fusion*, 50: 168–180, 2019.

[186] M. D. Muhlbaier, A. Topalis, and R. Polikar. Learn^{++} .NC: combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks*, 20(1):152–168, 2009. doi: 10.1109/TNN.2008.2008326.

[187] Michael D. Muhlbaier and Robi Polikar. An ensemble approach for incremental learning in nonstationary environments. In Michal Haindl, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 490–500, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72523-7.

[188] Kolluru Venkata Sreerama Murthy and Steven L Salzberg. *On growing better decision trees from data*. PhD thesis, Citeseer, 1995.

[189] Sreerama K Murthy, Simon Kasif, Steven Salzberg, and Richard Beigel. Oc1: A randomized algorithm for building oblique decision trees. In *Proceedings of AAAI*, volume 93, pages 322–327. Citeseer, 1993.

[190] Jalal A Nasiri, Nasrollah Moghadam Charkari, and Kourosh Mozafari. Energy-based model of least squares twin support vector machines for human action recognition. *Signal Processing*, 104:248–257, 2014.

[191] Jalal A Nasiri, Nasrollah Moghadam Charkari, and Saeed Jalili. Least squares twin multi-class classification support vector machine. *Pattern Recognition*, 48(3):984–992, 2015.

[192] Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

[193] Matthew Olson, Abraham J Wyner, and Richard Berk. Modern neural networks generalize on small data sets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3623–3632, 2018.

[194] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[195] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5189–5198, 2017.

[196] Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer, 2012.

[197] Edgar Osuna, Robert Freund, and Federico Girosit. Training support vector machines: an application to face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 130–136. IEEE, 1997.

[198] Nikunj C Oza. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345. Ieee, 2005.

[199] Akin Ozcift. SVM feature selection based rotation forest ensemble classifiers to improve computer-aided diagnosis of Parkinson disease. *Journal of Medical Systems*, 36(4): 2141–2147, 2012.

[200] Hamid Palangi, Li Deng, and Rabab K. Ward. Recurrent Deep-Stacking Networks for sequence classification. *2014 IEEE China Summit and International Conference on Signal and Information Processing, IEEE ChinaSIP 2014 - Proceedings*, (Xi):510–514, 2014. doi: 10.1109/ChinaSIP.2014.6889295.

[201] Y-H Pao and Yoshiyasu Takefuji. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5):76–79, 1992.

[202] Yoh-Han Pao, Stephen M Phillips, and Dejan J Sobajic. Neural-net computing and the intelligent control of systems. *International Journal of Control*, 56(2):263–289, 1992.

[203] Yoh-Han Pao, Gwang-Hoon Park, and Dejan J Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[204] D. Parikh and R. Polikar. An ensemble-based incremental learning approach to data fusion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):437–450, 2007. doi: 10.1109/TSMCB.2006.883873.

[205] Gwang Hoon Park and Yoh Han Pao. Unconstrained word-based approach for off-line script recognition using density-based random-vector functional-link net. *Neurocomputing*, 31(1-4):45–65, 2000.

[206] Gwang Hoon Park, Yoon Jin Lee, and Steven R LeClair. Intelligent rate control for MPEG-4 coders. *Engineering Applications of Artificial Intelligence*, 13(5):565–575, 2000.

[207] Ioannis Partalas, Grigorios Tsoumakas, and Ioannis Vlahavas. Pruning an ensemble of classifiers via reinforcement learning. *Neurocomputing*, 72(7-9):1900–1909, 2009.

[208] Domingos Pedro. A unified bias-variance decomposition and its applications. In *17th International Conference on Machine Learning*, pages 231–238, 2000.

[209] Witold Pedrycz and Zenon A Sosnowski. Genetically optimized fuzzy decision trees. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(3): 633–641, 2005.

[210] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[211] Gianvito Pio, Donato Malerba, Domenica D'Elia, and Michelangelo Ceci. Integrating microRNA target predictions for the discovery of gene regulatory networks: a semi-supervised ensemble learning approach. *BMC Bioinformatics*, 15(S1):S4, 2014.

[212] Vincent Pisetta. *New Insights into Decision Trees Ensembles*. PhD thesis, Lyon 2, 2012.

[213] John Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multi-class classification. *Advances in Neural Information Processing Systems*, 12:547–553, 1999.

[214] John C Platt, Nello Cristianini, and John Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553, 2000.

[215] R. Polikar, L. Upda, S. S. Upda, and V. Honavar. Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 31(4):497–508, 2001. doi: 10.1109/5326.983933.

[216] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *The Journal of Machine Learning Research*, 18(1):6673–6690, 2017.

[217] Xueheng Qiu, Ponnuthurai Nagaratnam Suganthan, and Gehan A.J. Amaratunga. Ensemble incremental learning random vector functional link network for short-term electric load forecasting. *Knowledge-Based Systems*, 145:182 – 196, 2018. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2018.01.015. URL http://www.sciencedirect.com/science/article/pii/S0950705118300236.

[218] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

[219] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333, 2011.

[220] Ye Ren, Le Zhang, and P N Suganthan. Ensemble Classification and Regression-Recent Developments, Applications and Future Directions. (February):41–53, 2016.

[221] Ye Ren, Le Zhang, and Ponnuthurai N Suganthan. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational Intelligence Magazine*, 11(1):41–53, 2016.

[222] B Richhariya, M Tanveer, AH Rashid, and Alzheimer's Disease Neuroimaging Initiative. Diagnosis of alzheimer's disease using universum support vector machine based recursive feature elimination (USVM-RFE). *Biomedical Signal Processing and Control*, 59:101903, 2020.

[223] Bharat Richhariya and M. Tanveer. EEG signal classification using universum support vector machine. *Expert Systems with Applications*, 106:169–182, 2018.

[224] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[225] Lior Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[226] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers-a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

[227] Louis B Rosenberg. Human swarms, a real-time method for collective intelligence. In *ECAL 2015: the 13th European Conference on Artificial Life*, pages 658–659. MIT Press, 2015.

[228] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[229] Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

[230] Leander Schietgat, Celine Vens, Jan Struyf, Hendrik Blockeel, Dragi Kocev, and Sašo Džeroski. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics*, 11(1):2, 2010.

[231] Wouter F Schmidt, Martin A Kraaijveld, Robert PW Duin, et al. Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, pages 1–1. IEEE Computer Society Press, 1992.

[232] Uri Shaham, Xiuyuan Cheng, Omer Dror, Ariel Jaffe, Boaz Nadler, Joseph Chang, and Yuval Kluger. A deep learning approach to unsupervised ensemble learning. In *International Conference on Machine Learning*, pages 30–39, 2016.

[233] Yuan-Hai Shao, Chun-Hua Zhang, Xiao-Bo Wang, and Nai-Yang Deng. Improvements on twin support vector machines. *IEEE Transactions on Neural Networks*, 22(6):962–968, 2011.

[234] Aman Sharma and Rinkle Rani. BE-DTI': ensemble framework for drug target interaction prediction using dimensionality reduction and active learning. *Computer Methods and Programs in Biomedicine*, 165:151–162, 2018.

[235] Rahul Sharma, Tripti Goel, M Tanveer, Shubham Dwivedi, and R Murugan. FAF-DRVFL: Fuzzy activation function based deep random vector functional links network for early diagnosis of Alzheimer disease. *Applied Soft Computing*, page 107371, 2021.

[236] Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. *arXiv preprint arXiv:1805.06440*, 2018.

[237] Kai-Quan Shen, Chong-Jin Ong, Xiao-Ping Li, Zheng Hui, and Einar PV Wilder-Smith. A feature selection method for multilevel mental fatigue EEG classification. *IEEE Transactions on Biomedical Engineering*, 54(7):1231–1237, 2007.

[238] Chuan Shi, Xiangnan Kong, Philip S. Yu, and Bai Wang. Multi-label ensemble learning. In Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba, and Michalis Vazirgiannis, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 223–239, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[239] Qiushi Shi, Rakesh Katuwal, P.N. Suganthan, and M. Tanveer. Random vector functional link neural network based ensemble deep learning. *Pattern Recognition*, page 107978, 2021. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.107978. URL https://www.sciencedirect.com/science/article/pii/S0031320321001655.

[240] Zenglin Shi, Le Zhang, Yun Liu, Xiaofeng Cao, Yangdong Ye, Ming-Ming Cheng, and Guoyan Zheng. Crowd counting with deep negative correlation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2018.

[241] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[242] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. Technical report, 2016. URL http://papers.nips.cc/paper/6205-swapout-learning-an-ensemble-of-deep-architectures.pdf.

[243] Chapman Siu. Residual Networks Behave Like Boosting Algorithms *. Technical report.

[244] Leslie N Smith, Emily M Hand, and Timothy Doster. Gradual dropin of layers to train very deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4763–4771, 2016.

[245] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Barbara Mele, and Guido Altarelli. Dropout: a simple way to prevent neural

networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. ISSN 03702693. doi: 10.1016/0370-2693(93)90272-J.

[246] Gregor Stiglic and Peter Kokol. Effectiveness of rotation forest in meta-learning based gene expression classification. In *Computer-Based Medical Systems, 2007. CBMS'07. Twentieth IEEE International Symposium on*, pages 243–250. IEEE, 2007.

[247] José L Subirats, José M Jerez, Iván Gómez, and Leonardo Franco. Multiclass pattern recognition extension for the new C-Mantec constructive neural network algorithm. *Cognitive Computation*, 2(4):285–290, 2010.

[248] Ponnuthurai Nagaratnam Suganthan. On non-iterative learning algorithms with closed-form solution. *Applied Soft Computing*, 70:1078–1082, 2018.

[249] Chuang Sun, Meng Ma, Zhibin Zhao, and Xuefeng Chen. Sparse deep stacking network for fault diagnosis of motor. *IEEE Transactions on Industrial Informatics*, 14(7):3261–3270, 2018.

[250] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.

[251] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, pages 1038–1044, 1996.

[252] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[253] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[254] Siham Tabik, Ricardo F Alvear-Sandoval, María M Ruiz, José-Luis Sancho-Gómez, Aníbal R Figueiras-Vidal, and Francisco Herrera. MNIST-NET10: a heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. ensembles overview and proposal. *Information Fusion*, 2020.

[255] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):809–821, 2015.

[256] Kai-Fu Tang, Hao-Cheng Kao, Chun-Nan Chou, and Edward Y Chang. Inquire and diagnose: Neural symptom checking ensemble using deep reinforcement learning. In *Proceedings of NIPS Workshop on Deep Reinforcement Learning*, 2016.

[257] Ke Tang, Minlong Lin, Fernanda L. Minku, and Xin Yao. Selective negative correlation learning approach to incremental learning. *Neurocomputing*, 72(13):2796 – 2805, 2009. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2008.09.022. URL http://www.sciencedirect.com/science/article/pii/S0925231209001192. Hybrid Learning Machines (HAIS 2007) / Recent Developments in Natural Computation (ICNC 2007).

[258] Ling Tang, Yao Wu, and Lean Yu. A non-iterative decomposition-ensemble learning paradigm using RVFL network for crude oil price forecasting. *Applied Soft Computing*, 70:1097–1108, 2018.

[259] M. Tanveer. Robust and sparse linear programming twin support vector machines. *Cognitive Computation*, 7(1):137–149, 2015.

[260] M. Tanveer, Mohammad Asif Khan, and Shen-Shyang Ho. Robust energy-based least squares twin support vector machines. *Applied Intelligence*, 45(1):174–186, 2016.

[261] M Tanveer, C Gautam, and Ponnuthurai N Suganthan. Comprehensive evaluation of twin SVM based classifiers on UCI datasets. *Applied Soft Computing*, page 105617, 2019.

[262] Dacheng Tao, Xiaoou Tang, Xuelong Li, and Xindong Wu. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7):1088–1099, 2006.

[263] Hubert AB Te Braake and Gerrit Van Straten. Random activation weight neural net (RAWN) for fast non-iterative training. *Engineering Applications of Artificial Intelligence*, 8(1):71–80, 1995.

[264] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[265] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *European Conference on Machine Learning*, pages 406–417. Springer, 2007.

[266] Giorgio Valentini and Francesco Masulli. Ensembles of learning machines. In *Italian Workshop on Neural Nets*, pages 3–20. Springer, 2002.

[267] Mark J Van der Laan, Eric C Polley, and Alan E Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[268] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[269] Sandro Vega-Pons and José Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25 (03):337–372, 2011.

[270] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Advances in Neural Information Processing Systems*, pages 550–558, 2016.

[271] Najdan Vuković, Milica Petrović, and Zoran Miljković. A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression. *Applied Soft Computing*, 70:1083–1096, 2018.

[272] Elad Walach and Lior Wolf. Learning to count with cnn boosting. In *European Conference on Computer Vision*, pages 660–676. Springer, 2016.

[273] Georg Waltner, Michael Opitz, Horst Possegger, and Horst Bischof. Hibster: Hierarchical boosted deep metric learning for image retrieval. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 599–608. IEEE, 2019.

[274] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of Neural Networks using DropConnect. In Sanjoy Dasgupta and David

McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 2013. PMLR. doi: 10.1109/TPAMI.2017.2703082. URL http://proceedings.mlr.press/v28/wan13.html.

[275] Bin Wang, Bing Xue, and Mengjie Zhang. Particle swarm optimisation for evolving deep neural networks for image classification by evolving and stacking transferable blocks. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[276] Guanjin Wang, Guangquan Zhang, Kup Sze Choi, and Jie Lu. Deep Additive Least Squares Support Vector Machines for Classification with Model Transfer. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(7):1527–1540, 2019. ISSN 21682232. doi: 10.1109/TSMC.2017.2759090.

[277] Jingyuan Wang, Kai Feng, and Junjie Wu. SVM-Based Deep Stacking Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:5273–5280, 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33015273.

[278] Ran Wang, Sam Kwong, Xu Wang, and Yuheng Jia. Active k-labelsets ensemble for multi-label classification. *Pattern Recognition*, 109:107583, 2021.

[279] Xiao Wang, Daisuke Kihara, Jiebo Luo, and Guo-Jun Qi. Enaet: Self-trained ensemble autoencoding transformations for semi-supervised learning. *arXiv preprint arXiv:1911.09265*, 2019.

[280] Zhihui Wang, Sook Yoon, Shan Juan Xie, Yu Lu, and Dong Sun Park. Random vector functional-link net based pedestrian detection using multi-feature combination. In *2013 6th International Congress on Image and Signal Processing (CISP)*, volume 2, pages 773–777. IEEE, 2013.

[281] Zhihui Wang, Sook Yoon, Shan Juan Xie, Yu Lu, and Dong Sun Park. A high accuracy pedestrian detection system combining a cascade AdaBoost detector and random vector functional-link net. *The Scientific World Journal*, 2014, 2014.

[282] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4): 279–292, 1992.

[283] Thomas Welchowski and Matthias Schmid. A framework for parameter estimation and model selection in kernel deep stacking networks. *Artificial Intelligence in Medicine*, 70:31–40, 6 2016. ISSN 09333657. doi: 10.1016/j.artmed.2016.04.002. URL https://linkinghub.elsevier.com/retrieve/pii/S0933365715300713.

[284] Halbert White. Approximate nonlinear forecasting methods. *Handbook of Economic Forecasting*, 1:459–512, 2006.

[285] Bernard Widrow, Aaron Greenblatt, Youngsik Kim, and Dookun Park. The no-prop algorithm: A new learning algorithm for multilayer neural networks. *Neural Networks*, 37:182–188, 2013.

[286] Marco A Wiering and Hado Van Hasselt. Two novel on-policy reinforcement learning algorithms based on td (λ)-methods. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 280–287. IEEE, 2007.

[287] Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38 (4):930–936, 2008.

[288] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[289] David H Wolpert. On bias plus variance. *Neural Computation*, 9(6):1211–1243, 1997.

[290] Jun-Feng Xia, Kyungsook Han, and De-Shuang Huang. Sequence-based prediction of protein-protein interactions by means of rotation forest and autocorrelation descriptor. *Protein and Peptide Letters*, 17(1):137–145, 2010.

[291] Yuelong Xia, Ke Chen, and Yun Yang. Multi-label classification with weighted classifier selection and stacked ensemble. *Information Sciences*, 557:421–442, 2021.

[292] Jingjing Xie, Bing Xu, and Zhang Chuang. Horizontal and vertical ensemble with deep representation for classification. *arXiv preprint arXiv:1306.2759*, 2013.

[293] Jie Xu, Lixing Chen, and Shaolei Ren. Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking*, 3(3):361–373, 2017.

[294] Yitian Xu, Rui Guo, and Laisheng Wang. A twin multi-class classification support vector machine. *Cognitive Computation*, 5(4):580–588, 2013.

[295] Jie Xue, Zhuo Wang, Deting Kong, Yuan Wang, Xiyu Liu, Wen Fan, Songtao Yuan, Sijie Niu, and Dengwang Li. Deep ensemble neural-like p systems for segmentation of central serous chorioretinopathy lesion. *Information Fusion*, 65:84–94, 2021.

[296] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Convolutional channel features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 82–90, 2015.

[297] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. *Available at SSRN*, 2020.

[298] Guoxian Yu, Carlotta Domeniconi, Huzefa Rangwala, Guoji Zhang, and Zhiwen Yu. Transductive multi-label ensemble classification for protein function prediction. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1077–1085, 2012.

[299] Chun-Xia Zhang and Jiang-She Zhang. RotBoost: a technique for combining rotation forest and AdaBoost. *Pattern Recognition Letters*, 29(10):1524–1536, 2008.

[300] H. Zhang, W. Liu, J. Shan, and Q. Liu. Online active learning paired ensemble for concept drift and class imbalance. *IEEE Access*, 6:73815–73828, 2018. doi: 10.1109/ACCESS.2018.2882872.

[301] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5978–5986, 2019.

[302] Junhao Zhang, Wei Zhang, Ran Song, Lin Ma, and Yibin Li. Grasp for stacking via deep reinforcement learning. pages 2543–2549, 2020.

[303] Le Zhang and Ponnuthurai N Suganthan. Oblique decision tree ensemble via multisurface proximal support vector machine. *IEEE Transactions on Cybernetics*, 45(10): 2165–2176, 2014.

[304] Le Zhang and Ponnuthurai N Suganthan. A comprehensive evaluation of random vector functional link networks. *Information Sciences*, 367:1094–1105, 2016.

[305] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Random forests with ensemble of feature spaces. *Pattern Recognition*, 47(10):3429–3437, 2014.

[306] Le Zhang and Ponnuthurai Nagaratnam Suganthan. Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier]. *IEEE Computational Intelligence Magazine*, 12(4):61–72, 2017.

[307] Le Zhang, Zenglin Shi, Ming-Ming Cheng, Yun Liu, Jia-Wang Bian, Joey Tianyi Zhou, Guoyan Zheng, and Zeng Zeng. Nonlinear Regression via Deep Negative Correlation Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(c): 1–1, 2019. ISSN 0162-8828. doi: 10.1109/tpami.2019.2943860.

[308] Li Zhang, Wei-Da Zhou, Tian-Tian Su, and Li-Cheng Jiao. Decision tree support vector machine. *International Journal on Artificial Intelligence Tools*, 16(01):1–15, 2007.

[309] Wen Zhang, Feng Liu, Longqiang Luo, and Jingxia Zhang. Predicting drug side effects by multi-label learning and ensemble learning. *BMC Bioinformatics*, 16(1):365, 2015.

[310] Wentao Zhang, Jiawei Jiang, Yingxia Shao, and Bin Cui. Snapshot boosting: a fast ensemble framework for deep neural networks. *Science China Information Sciences*, 63(1):112102, 2020.

[311] Yongshan Zhang, Jia Wu, Zhihua Cai, Bo Du, and S Yu Philip. An unsupervised parameter learning model for RVFL neural network. *Neural Networks*, 112:85–97, 2019.

[312] Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, pages 592–617, 2013.

[313] Qiang Li Zhao, Yan Huang Jiang, and Ming Xu. Incremental learning by heterogeneous bagging ensemble. In Longbing Cao, Jiang Zhong, and Yong Feng, editors, *Advanced*

Data Mining and Applications, pages 1–12, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-17313-4.

[314] Li Zheng, Tao Li, and Chris Ding. Hierarchical ensemble clustering. In *2010 IEEE International Conference on Data Mining*, pages 1199–1204. IEEE, 2010.

[315] Hongming Zhou, Guang Bin Huang, Zhiping Lin, Han Wang, and Yeng Chai Soh. Stacked extreme learning machines. *IEEE Transactions on Cybernetics*, 45(9):2013–2025, 9 2015. ISSN 21682267. doi: 10.1109/TCYB.2014.2363492. URL http://ieeexplore.ieee.org/document/6937189/.

[316] Zhi-Hua Zhou. When semi-supervised learning meets ensemble learning. In *International Workshop on Multiple Classifier Systems*, pages 529–538. Springer, 2009.

[317] Zhi-Hua Zhou and Ji Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.

[318] Zhi-Hua Zhou and Wei Tang. Clusterer ensemble. *Knowledge-Based Systems*, 19(1): 77–83, 2006.

[319] Zhi-Hua Zhou, Fabio Roli, Josef Kittler, et al. Multiple classifier systems. In *Proc. 2013 11th Int. Workshop Mult. Classifier Syst.(MCS)*, page 24. Springer, 2013.

[320] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.