Random Projection Optimal Trees Ensemble

Nosheen Faiz

A Thesis Submitted for the Degree of

Doctor of Philosophy

Department of Mathematical Sciences

University of Essex

July 2021

Dedicated to

My beloved and affectionate parents for their endless love, care and support.

Abstract

Ensemble classifiers, formed by the combination of multiple weak learners, have been shown to outperform ordinary classification methods in that the former decrease bias, variance and/or improve predictions. These classifiers, however, can still result in low prediction performance when used with the wrong choice of their hyper-parameters values and/or when there are noisy features in the data. Thus, feature selection and fine tuning hyper-parameter could improve predictive accuracy of ensemble classifiers. This thesis first investigates the effect of feature selection on three methods: Random Forest (RF), Optimal Trees Ensemble (OTE) and Random Projection Ensembles (RP) in high dimensional settings. To this end, LASSO has been considered for selecting the most important features based on training data for dimension reduction. Additionally, the influence of various hyper-parameters regulating the three methods has also been assessed.

Secondly, this thesis proposes a novel idea to use random projection method in conjunction with optimal tree selection to get an improved trees ensemble. This is achieved by randomly projecting the training data into lower dimension and classification trees are grown on bootstrap samples taken from the newly projected datasets. The best performing trees are selected based on out-of-bag error rate and combined to get the final Optimal Random Projection Trees Ensemble (ORPTE). The results of ORPTE are compared with those of Tree, RF, OTE, RP, *k*-NN, XGBoost and SVM.

Analysis on several benchmark datasets is given to illustrate the effect of feature selection and hyper-parameter tuning on the methods and the efficiency of the proposed method. The results reveal that feature selection improves the predictive performance of the RP method in addition to reducing the computational burden on benchmark and

example datasets. The performance of OTE and RF is less influenced by feature selection. Moreover, ORPTE has outperformed in terms of prediction accuracy in majority of the cases.

Acknowledgements

All praises and thanks to God Almighty for every thing!

It is a pleasure to thank many people who made this research possible. I gratefully acknowledge the support and generosity of my supervisor Professor Berthold Lausen and co-supervisor Dr Metodi Metodiev, their patience, time and most important their guidance, support and immense knowledge in this research. I could not have imagined having a better advisor and mentor for my Ph.D study. It was a great previllege to work with them.

To Abdul Wali Khan University, Mardan, Pakistan for funding this research. At the outset, I wish to thank and express my profound gratitude, indebtedness and appreciation to my esteemed mentor, and the founder Vice Chancellor, Professor Dr Ihsan Ali (Sitara-e-Imtiaz) for his endless efforts to develop the University's faculty and for encouraging me.

For many valuable suggestions, constructive feedback and discussions on my thesis and papers, I am extremely grateful and indebted to Dr Adi Lausen, who has also been an indefatigable advisor and proof-reader.

I would like to express my appreciation and thanks to all the staff at the Department of Mathematical Sciences, University of Essex for their kindness and assistance.

I want to acknowledge a special debt to Dr. Zardad Khan who reviewed several drafts

of chapters of my thesis and provided me assistance and invaluable comments on my writing.

I acknowledge the people who mean a lot to me in my life. To my sisters, Hina, Sumaira, Faryal, Kiran and the brother Waqas who are continually praying for my fortune. Thank you doesn't seem sufficient, but it is said with appreciation and respect for your kindness, affection and love.

To all my brilliant friends for helping me get through the difficult times, and for all the emotional support, comraderie, entertainment, and caring they provided. Particularly, Ms Almas who has been my companion throughout my journey. I am forever thankful to my friends, Ms Seema, Ms Safia, Ms Rabia, and Ms Neelam who took a very good care of my family in Pakistan in my absence. Lastly, and most importantly, I wish to thank my parents. They bore me, raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

Declaration

This study is based on research carried out at the Applied Statistics Group, Department of Mathematical Sciences, University of Essex, United Kingdom. No part of this thesis has been submitted elsewhere (except my own publications) for any other degree, and it is based on the results of my own investigations and findings, unless referenced, to the contrary, in the text.

Notations

Notations	Description
$\mathcal{L} = (\mathbf{X}, Y)$	Training dataset
$\mathbf{X}$	$n \times q$-dimensional matrix of n observations
Y	$n\times$ 1 vector of responses
n	Size of training sample.
n_{test}	Number of test observations
n_{oob}	Number of out of bag observations
q'	Dimension of projected space with $q' < q$
q_{mtry}	Number of features chosen randomly at each node for splitting where $q_{mtry} < q'$
PE	Prediction error
$\mathbf{b_1}$	Number of blocks
$\mathbf{b_2}$	Block size
$\mathcal{T}$	Number of trees

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ensemble methods have acquired great importance and drawn attention of the researchers in the field of Machine Learning over the years. These methods basically build a set of classifiers using learning algorithms and then classify new instances by applying a weighted vote of their predictions. These techniques focus on strengthening a weak classifier by integrating the decision of several weak learners for the classification and regression tasks [1]. The basic aim of such ensemble methods is to enhance generalizability/robustness over an individual classifier. Generally, improvement in the predictive accuracy is attained for these methods but sometimes the results are complicated and less interpretable. The use of tree structured ensemble techniques is common in classification and regression problems and have been extensively explored. Random Forest which is an improvement over Bagging and known for higher predictive performance is an example of ensemble methods.

The predictive performance and efficiency of the overall forest depend on the individual strength of trees in the ensemble and the amount of correlation between the error rates of two independent trees. Optimal Trees Ensemble is introduced for more refinement of the

Random Forest idea by selecting optimal trees from a large group of trees [2]. Diversity in the base models could also be introduced by using other techniques like Random Projection method and then selecting a percent of accurate and diverse trees or limiting the number of trees might improve the predictive performance of an ensemble. Following this notion, the idea of developing the ensemble of trees that are formed on the projected data into a lower dimension and accuracy has explored in this thesis. The aim here is to obtain improved predictive performance by using an ensemble of small number of classification trees from an initial ensemble grown on projected data.

The rest of the work is divided into six further chapters and three appendices in this thesis and are explained in the following.

In Chapter 2, we provide an in-depth discussion of some of the tree-based ensemble methods and other machine learning ensemble methods from literature and its general framework. We also discuss importance of ensemble methods. A detailed explanation of the basic Classification and Regression Tree is given prior to the description of ensemble methods such as Random Forest, Optimal Trees Ensemble, Extreme Gradient Boosting and dimensionality reduction technique Random Projection Ensembles as a background for this thesis. These ensemble methods are used for classification problems. Support Vector Machines and k-Nearest Neighbours are also introduced for comparison purposes.

Chapter 3 gives an overview of assessing how hyper-parameters influence Random Forest, Optimal Trees Ensemble and Random Projection Ensembles. Several hyper-parameters regulating these methods have been assessed such as node size, number of trees, number of features for splitting the nodes, number of blocks etc. This chapter also presents benchmarking for evaluating Random Forest, Optimal Trees Ensemble and Random Projection

Ensembles on several benchmark datasets. LASSO has been considered for selecting the most important features based on training data for evaluating the influence of feature selection on these ensemble methods.

Chapters 4 gives an overview of extension of the idea of Optimal Trees Ensemble to Optimal Random Projection Trees Ensemble (ORPTE) and to select the percent of best trees from an ensemble formed on projected data which gives minimum error on out-of-bag observations and combine them together to form a new forest ensemble. This ensemble is called as Optimal Random Projection Trees Ensemble (ORPTE). Using a total of 33 benchmark datasets, the results from ORPTE are compared with those of Random Forest, Optimal Trees Ensemble, Random Projection with QDA, Tree, k-Nearest Neighbours, Extreme Gradient Boosting and Support Vector Machine (Linear and Radial). Simulation models are also described to further interpret the behaviour of the proposed method. The class membership probabilities are also estimated for ORPTE in terms of Brier Score and compared with others on the same benchmark and simulated datasets.

Chapter 5 illustrates influence of hyper-parameters and feature selection for the newly proposed method ORPTE in comparison with Random Forest, Optimal Trees Ensemble and Random Projection Ensembles (with QDA, LDA). To this end, several hyper-parameters regulating ORPTE have been assessed such as, number of trees initially grown, lower dimension, percent of the best trees etc. Also, LASSO has been considered for selecting the most important features for investigating the effect of feature selection on ORPTE using benchmark datasets.

Chapter 6 consists of analysis of next generation sequencing data for binary classification task. The aim is to select differentially expressed genes and then apply ensemble models

such as Random Forest, Optimal Trees Ensemble, Random Projection Ensembles and Optimal Random Projection Trees Ensemble to investigate it's utility under high dimensional setting. As a motivational example this chapter in particular uses the pre-processed next generation sequencing gene expression dataset i.e., colon cancer, stage II vs stage III only (non-invasive vs invasive). The analysis done in the chapter also flag out genomic markers that are responsible for regulating the outcome and better explain the relationship.

Chapter 7 briefly describes overall conclusion drawn on the basis of findings and discussions from the previous chapters. This chapter also provides possible feature plans, merits and demerits of the work done in the research study.

Appendix *A* gives a comprehensive sketch of the benchmark datasets utilized throughout this thesis. Appendix *B* displays tables, plots and figures, referred to in different chapters. Appendix *C* provides manual pdf of the ORPTE, R package, executed in R software [3].

Chapter 2

Ensemble methods and dimensionality reduction

2.1 Introduction

In today's modern world the development of artificial intelligence has got attention for technological advancement. Machine learning consists of computational methods which make use of previous information to enhance the predictive performance and to make valid prediction. Supervised learning methods deal with finding the relationship between the predictor variables and the response variable which is then expressed in the form of a model. The final model is then used to assign class label to new instances whose class labels are unknown. In literature, several supervised classification algorithms have been proposed such as Random Forest, Support Vector Machines, Decision Trees, Neural Networks etc.

Combining multiple classifiers, known as ensemble methods, has proved to acheive

higher rate of correct classification as compared to the single classifier [4]. Ensemble learning basically works by executing a base learning algorithm multiple times and let them vote for deciding about the class of test/new observation. The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability, accuracy and efficiency over a single estimator [5,6]. Ensemble methods are widely used in modern classification and regression problems. It might be difficult to know the beginning of the history of ensemble methods since the basic idea of using multiple base models have been in use for a long period of time, yet it is obvious that the research on ensemble methods done since 1990's contributes much to the work in this field. This can be divided into two types of works. The first work relates to applied research conducted by Hansen and Salamon [7]. In this work they found that predictions made by combining a set of classifiers are usually more accurate as compared to the predictions by a single classifier that might be individually very strong. The second type of work relates to theoretical research where Schapire [8] proved that weak learners can be boosted to make strong learners. This resulted in a famous algorithm called Boosting [9,10] that is one of the most influential ensemble methods in modern research done in the field. Research that resulted in similar arguments could be found in [11–13].

This chapter provides a detailed discussion on ensemble methods and dimensionality reduction for classification task. Ensemble techniques such as Decision Trees, Random Forest, Optimal Trees Ensemble and Random Projection Ensembles are discussed in detail. Support Vector Machine and k-Nearest Neighbours are also introduced in this chapter for comparison purposes.

2.1.1 Importance of Ensemble Methods

Ensemble methods can be used for the purpose of classification, regression and modelling data related to real world problems which can be described as follows;

1. In case of building more regular/conventional models, if the prediction error is large than an ensemble method, it shows that the conventional model is not suitable. It suggests that one should be careful while selecting a model. Similarly, over-fitting can be avoided by using cross-validation for a conventional model [14].

2. Ensemble methods can be applied to evaluate inter-correlation/association between predictor and response variable which may have low contribution to the model and choose those predictor variables which are ignored in model building and could play an important role.

3. As compared to logistic regression, ensemble methods could construct more reliable estimates for the intervention effect through propensity scores [15].

2.2 Classification and Regression Tree

A prediction model, which is used in the construction of Random Forest is called Classification and Regression Tree (CART) [16]. To predict the class label of new instances, the training dataset is utilized to build a decision tree. Suppose training data is presented as $\mathcal{L} = (\mathbf{X}, Y) = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{X}$ is an $n\times$ q-dimensional matrix of n observations each with q features and Y is an n $\times$ 1 vector of responses. The root node of tree constitutes of the complete dataset and the terminal nodes comprise of the subsets

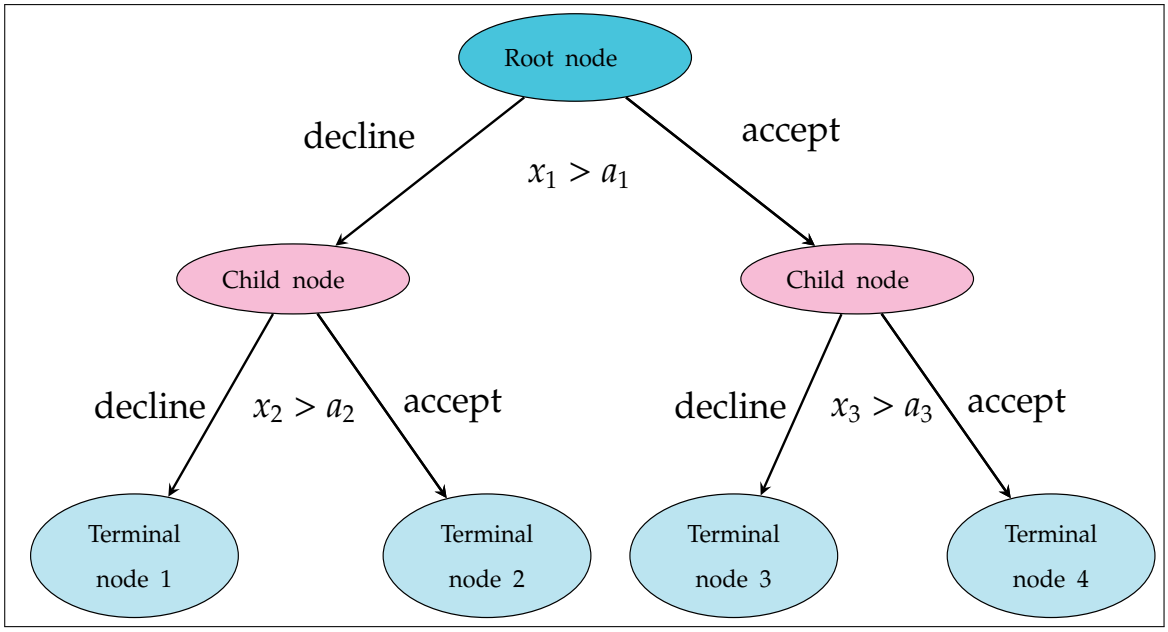


Figure 2.1: *Basic classification tree structure*

of the training data obtained from the final partitions [17]. Suppose, x_1, x_2 and x_3 are any three variables and a_1, a_2 and a_3 are three given constants then a basic classification tree structure is given in Figure 2.1.

The formation of leaf nodes, and the outcome of CART, depend on three principle criteria: splitting; variable selection; and size of the tree. To avoid over-fitting, it is needed a quality assessment of the tree [18]. CART consider best splits for the predictor variables in partitioning of the root nodes of the tree to create child nodes. Let suppose S_c is any node where $c = 1, 2, ..., t$, if the node S_c contains all the observations having all values either 0 or 1 for the binary response variable then the node is considered as least impure. For splitting the nodes of the tree, selection of the best split depends on the degree of impurity. Generally, the Gini Index and Entropy are the criteria used as impurity functions [19] that are given in equations (2.5) and (2.6). To calculate Gini Index, let say ϕ_{GI}, for a set of observations with K classes, the proportion Θ_{iS_c} of observations belonging to class i among

all observations at a given node S_c is utilized. The Gini Index, ϕ_{GI} is then given by

$$\phi_{GI}(S_c) = \sum_{i=1}^{K}(\Theta_{iS_c} \sum_{j \neq i} \Theta_{jS_c}), \tag{2.1}$$

$$= \sum_{i=1}^{K} \Theta_{iS_c}(1 - \Theta_{iS_c}), \tag{2.2}$$

$$= \sum_{i=1}^{K}(\Theta_{iS_c} - \Theta_{iS_c}^2), \tag{2.3}$$

$$= \sum_{i=1}^{K} \Theta_{iS_c} - \sum_{i=1}^{K} \Theta_{iS_c}^2, \tag{2.4}$$

$$= 1 - \sum_{i=1}^{K} \Theta_{iS_c}^2. \tag{2.5}$$

For binary class problems, the value of K in the above equation is set equal to 2 i.e., $K = 2$. Similarly, for using Entropy as the impurity measure for $K = 2$ classes, the following expression is used.

$$\phi_{Entropy}(S_c) = -\sum_{i=1}^{K} \Theta_{iS_c} log \Theta_{iS_c}. \tag{2.6}$$

Throughout this thesis, Gini Index has been used as impurity measure for splitting the nodes of trees. The ”randomForest” R package also uses Gini Index as the default choice.

Moreover, the standard Random Forest algorithm uses mid point on a variable as the default cut-off point. In literature, there are sevral alternative splitting rules used for partitioning the data. Maximally selected rank statistic has been suggested for estimating an exact p-value for variable selection under the null hypothesis of no influence of the selected variable and to evaluate the effect of chosen cut-off points [20]. Lausen et al., [21] have proposed a method to adjust for variables measured on different scales for classification and regression tree. The adjusted p-value of a maximally selected log-rank statistic is utilized for the adjustment of variable selection bias that arises due to features measured on different scales. A detailed discussion on further improving the split criteria for classification trees and ensemble methods and assessing optimal features is given in [21–23]. Generalized maximally selected rank statistics is proposed by [24] to assess a large number of cut-points and to examine the asymptotic distributions of these maximally selected statistics. Furthermore, for the exact distribution of a linear rank statistics a lower bound is calculated to get the exact distribution of maximally selected rank statistics [25].

Tree based methods suffer from instability and variable selection bias because small changes in training samples may cause large changes in the results while variable selection bias arises due to features measured on different scales. Tree based methods suffer from variable selection bias in that the algorithm involved might favors the predictor variables with more values for node splitting of tree. These methods are instable in that they are based on a greedy search approach and might fail in the search of finding a global optimum. Instability in learning algorithms was first investigated by Breiman and reconsidered different paths to get rid of these instabilities. Bootstrap Aggregation/Bagging is one of the effective procedures, used to improve on unstable classifiers which reduces the variance of

the base models. An idea of Subagging based on subsampling as an alternative aggregation technique is motivated by [26] and explored the effect of variance reduction of Bagging. The basic tree algorithm pseudo code is given in *Algorithm 1*.

Algorithm 1 ***Pseudo code of a basic Tree***

1. *Start with root node consisting of all the observations*
2. *The root node is partitioned on the basis of a variable which minimizes the sum of the node impurities between the two child nodes and choose the partition with minimum impurity*
3. *Repeat step 2 on the resultant child nodes in turn until a stopping criterion reached*

2.3 Tree structured ensemble methods

2.3.1 Random Forest

Bagging uses regression trees as base classifier grown on bootstrap samples taken from the training data and the results are then averaged to obtain a single estimate [27]. Random Forests is modified version of Bagging. For classification tasks, a forest of trees are grown on the bootstrap samples drawn from the training data and a subset of features are selected randomly for splitting each node. A decision is made by majority voting about the predicted class. Random Forest was introduced by Breiman [28]. This technique basically tries to reduce the variance by decorrelating the trees which is attained by choosing a set of variables randomly at each node. The main steps of the Random Forest algorithm are given as:

- Divide the data into training and testing parts.
- Bootstrap samples are taken from the training dataset.

- The classification/regression tree is grown on each of the bootstrap sample from training data. The tree is grown in a manner such that a random sample of features is selected at each node rather than all the features for splitting the nodes and continues until it reaches the terminal nodes.

- New observations are processed through the grown trees and reach the terminal nodes. The decision about the new data predictions is made by the majority votes (defined in Section 4.3.1).

Increasing the number of trees as the number of features becomes larger can be helpful for the better predictive performance of Random Forest. One possible reason could be that if a small number of trees is grown while there is a large number of features then some predictor might not be selected in all subspaces thus leading to a low predictive performance [29]. For evaluating the predictive performance of Random Forest, one can compare predictions of a forest to predictions made by subsets of trees from a forest. If the predictions of subsets of trees are well as given by the full forest then a forest contains sufficient number of trees. The pseudo-code of a Random Forest algorithm is given in *Algorithm 2*.

Algorithm 2 Pseudo code of a Random Forest

Generate $\mathcal{T}$ trees

1: **for** $i = 1 \rightarrow \mathcal{T}$ **do**
   Select a random sample from the training data of size n, with replacement for producing ith bootstrap sample.
   Generate a root node, S_{R_i} containing ith bootstrap sample.
   Call BuildTree(i).
2: **end for**

BuildTree(i)

3: **if** S_{R_i} contains all the instances from the same class **then**
4: **return**
5: **else**
   Randomly select q_{mtry} features from all q features in S_{R_i}.
   Select the feature with the highest information gain, that best split S_{R_i}.
   Create child nodes of S_{R_i}
6: **end if**
7: **for** $j = 1 \rightarrow |childnodes|$ **do**
   Call BuildTree(j)
8: **end for**

2.3.2 Optimal Trees Ensemble

Expanding on Breiman's [28] idea of growing the number of trees for classification and regression purposes, [2,30] proposed Optimal Trees Ensemble (OTE). Basically, this method selects the best trees from a number of trees initially grown by Random Forest on the basis of their individual and collective performance. A percentage of trees are then chosen on their individual performance on out-of-bag (OOB) observations and assessed on independent training data for their collective performance using the Brier Score [31]. A tree is called optimal for the final ensemble if its addition to the previously added trees increase its predictive performance. The OTE method selects (1) trees for the final ensemble one by one starting from the tree having the highest prediction accuracy rate and (2) the best trees from an initial ensemble on the basis of their individual predictive accuracy. The selected trees are then combined together to develop a new forest ensemble. To construct

the ensemble of the optimal trees from the data points following steps are performed:

- The training dataset is divided into disjoint groups randomly say, Set A and Set B. Trees are grown on bootstrap samples drawn from Set A.

- For growing the trees, a random sample of q_{mtry} features is selected from the set of q features at the nodes of each tree which inculcates additional randomness in the trees. As the trees are grown on bootstrap samples, there are some observations that are left out of the samples. These observations are called out-of-bag (OOB) observations and play no role in the training of the base learner.

- Trees with the smallest individual error rate on out-of-bag observations (defined in Section 4.3) are selected and arranged in ascending order with respect to their error rates.

- Trees are selected one by one and the Brier Score (defined in Section 4.6.2) is calculated to see if a tree is improving the performance on validation data. Those trees are added in the final ensemble.

- For the prediction of new instances, the predictions of the selected trees are aggregated.

The pseudo-code of OTE is given in *Algorithm 3*.

Algorithm 3 Pseudo code of OTE

Grow $\mathcal{T}$ trees

for $i = 1 \rightarrow \mathcal{T}$ **do**

- Select a bootstrap sample from the training data (Set A) for producing τ_i, where τ_i is the set of observations used for growing the ith tree.
- Store the remaining observation from the i^{th} sample as $OOB(i)$.
- Make a root node S_{R_i} comprising τ_i.
- Call on $GrowTree(\tau_i)$ to grow tree M_i.
- Estimate unexplained variance $V_i(M_i)$ or misclassification error $MC_i(M_i)$ for M_i using $OOB(i)$

end for

Select the best trees based on individual performance

for $j = 1 \rightarrow \mathcal{T}$ **do**

  if Classification & $MC_j(M_j) < Q$ **then**
    Select M_j, where Q is a quintile point of misclassification errors of $\mathcal{T}$ trees
  else
    Drop M_j
  end if
  if Regression & $V_j(M_j) < Q$ **then**
    Select M_j, where Q is a quintile point of the unexplained variances of $\mathcal{T}$ trees
  else
    Drop M_j
  end if

end for

GrowTree(i)

if Classification and S_{R_i} contain observations of the same class **then**
  return
else
  Select q_{mtry} features out of q features for splitting S_{R_i} and repeat the process for the child nodes until the leaf nodes are reached.
  Store the tree for voting or averaging.
end if

Select the best trees based on ensemble performance

Arrange the trees selected above, say M, in ascending order with respect to errors

Initialize $\ell = 1$ and select the first best tree with the smallest misclassification error or unexplained variance as the initial ensemble.

for $\ell = 2 \rightarrow M$ **do**

  if Classification & $\widehat{BS}^{\langle \ell \rangle} < \widehat{BS}^{\langle \ell - 1 \rangle}$ **then**
    Select the ℓ^{th} tree, where $\widehat{BS}^{\langle \ell \rangle}$ is the Brier Score having ℓ^{th} tree and $\widehat{BS}^{\langle \ell - 1 \rangle}$ is the Brier Score without ℓ^{th} tree after using validation set (Set B).
  else
    Do not select the ℓth tree
  end if
  if Regression & $\mathcal{V}^{\langle \ell \rangle} < \mathcal{V}^{\langle \ell - 1 \rangle}$ **then**
    Select the ℓth tree, where $\mathcal{V}^{\langle \ell \rangle}$ is the unexplained variance of the ensemble having the ℓ^{th} tree and $\mathcal{V}^{\langle \ell - 1 \rangle}$ is the unexplained variance of the ensemble without ℓ^{th} tree after using some independent training data
  else
    Do not select the ℓ^{th} tree
  end if

end for

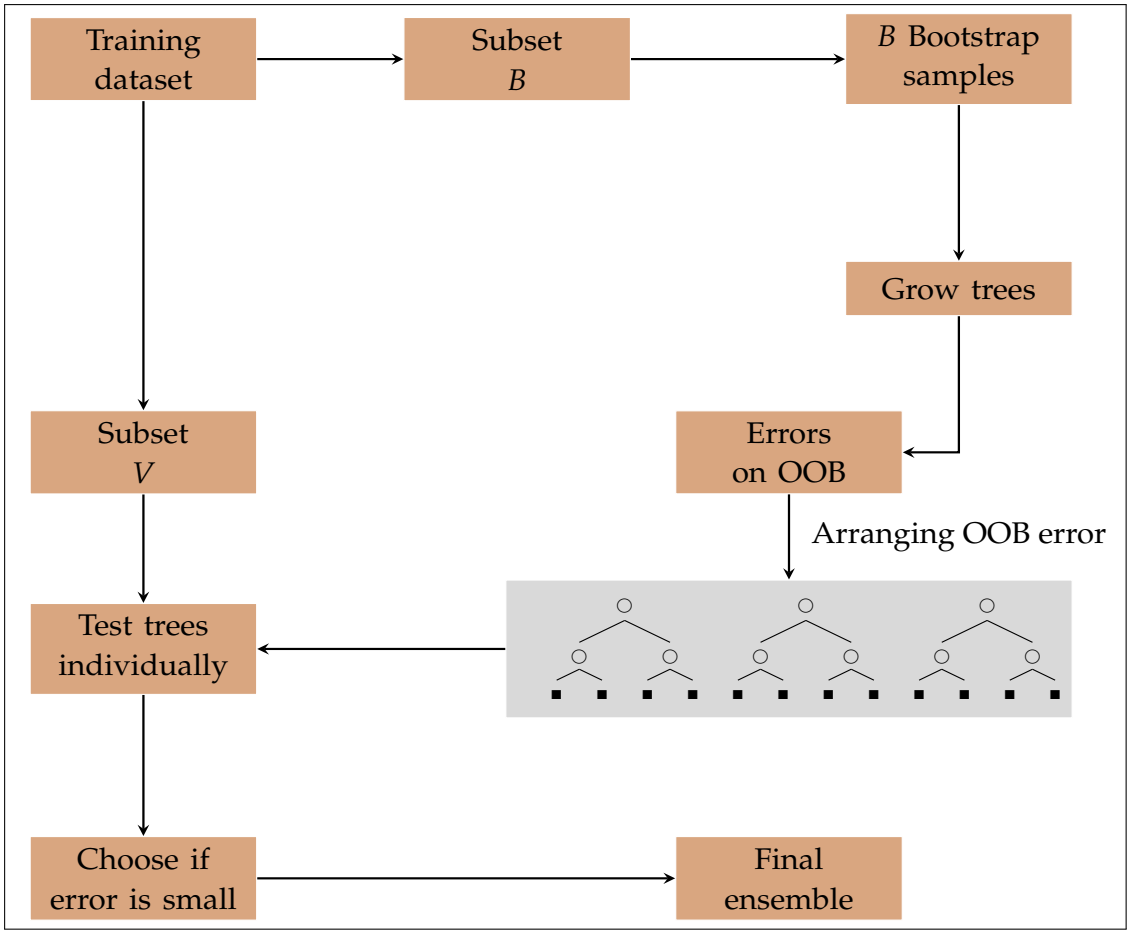


Figure 2.2: *OTE flow chart for regression and classification.*

2.3.3 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a machine learning algorithm. This method is basically a decision-tree-based ensemble that utilises a Gradient Boosting scheme. Extreme Gradient Boosting is a modified version of the Gradient Boosting technique known for it's efficiency, high computational speed and predictive performance [45]. This technique can be used for regression and classification problems, which develops a prediction model in a way to combine predictions of weak models, typically decision trees. It constructs the model in a stage-wise manner similar to other boosting methods and generalizes them by enabling optimization of differentiable loss function. For making the best split, XGBoost

emloys pre-sorted and histogram-based algorithm. The data points are split for each feature into discrete bins by using histogram-based algorithm which are then used to compute the split value of the histogram. Also, the number of terminal nodes in the trees can vary in XGBoost algorithm.

XGBoost makes use of a more regularized model formalization which helps to reduce over-fitting and makes it unique. Therefore, XGBoost algorithm gives better performance. Moreover, XGboost is available in multiple programming languages including R. XGBoost can be applied through the "xgboost" [46] R package.

2.4 Support Vector Machines and *k*-Nearest Neighbours

For the comparison purpose, we also consider Support Vector Machine and *k*-Nearest Neighbours in this work. In the following sections, we describe the Support Vector Machine and *k*-Nearest Neighbours based on nearest neighbour rule.

2.4.1 Support Vector Machines

Vladimir Vapnik [32] introduced Support Vector Machines (SVM) in 1995. SVM is a supervised learning method and has gained importance in statistical theory. This method can be utilized for classification [33] and regression [34] purposes. SVM was initially built for binary class problem by using the concept of large margin where classification is performed by spotting the hyperplane which maximizes the margin between the two classes. This technique holds a distinguish property i.e., it considers only those data points which are closest to the hyperplane. These data points are called support vectors.

For the demonstration consider the following Figure 2.3 which shows how SVM works in general. Suppose that there are two classes (class 1 and class -1) where the data points can be separated in the given classes. Now the point is that there can be many possible hyperplanes as can be seen in the Figure 2.3 to divide the data into two classes but which one is to be chosen although hyperplane 1 and 2 are very close to data points. SVM can tackle this issue by finding the hyperplane which maximizes the margin and imposes equal penalty on both classes.

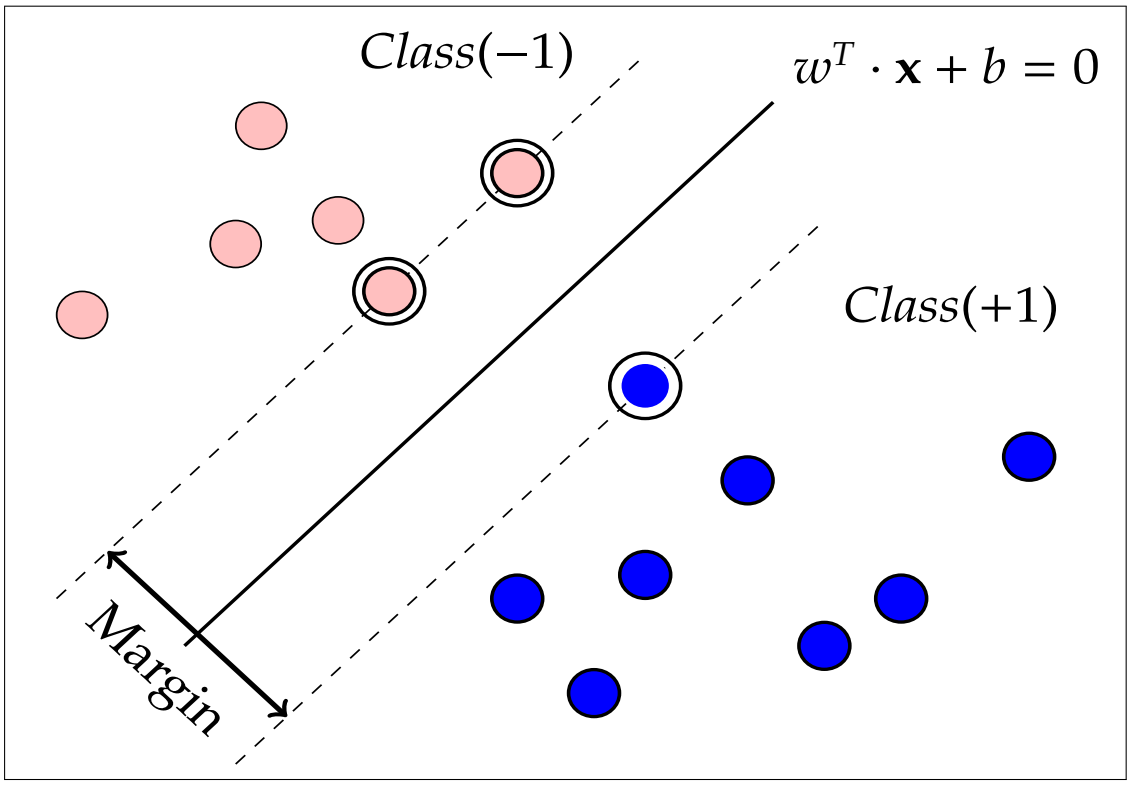


Figure 2.3: *Graph shows possible hyperplanes separating the data into two classes.*

In the case of linear separable classes shown in figure 2.3, the hyperplane can be presented by the equation

$$z(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + \mathbf{b}, \tag{2.7}$$

where w is the weight vector and b is the bias. The classification function, g(z), is then

defined as,

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases} \tag{2.8}$$

A complete SVM model can be written as,

$$f_{SVM}(\mathbf{x}) = g(z(\mathbf{x})) = g(\mathbf{w^T x_i} + \mathbf{b}). \tag{2.9}$$

The best hyperplane is the one which maximizes the margin between training observations from different classes.

It is often experienced that linear separators and boundaries can not be applied to data when there are non linear interactions and the non linear dependence between the features in the data. This problem is solved through a non linear transformation on the features and converting them to a higher dimentional space by using kernal tricks for example, Radial Basis Function kernal. This transformation helps in separating non linear data using a non linear decision boundary. Radial Basis Function kernal is defined as,

$$R_{SVM}(\mathbf{x}, \mathbf{x'}) = exp\left(-\frac{||\mathbf{x} - \mathbf{x'}||^2}{2\sigma^2}\right). \tag{2.10}$$

$||\mathbf{x} - \mathbf{x'}||^2$ presents the squared Euclidean distance between the two feature vectors.

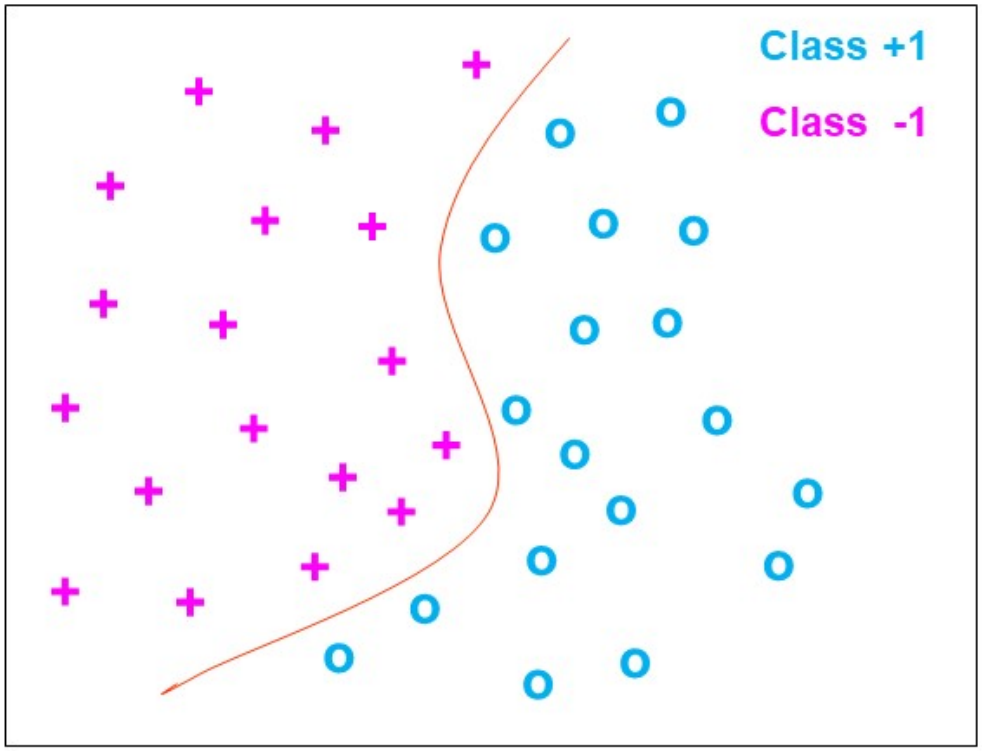


Figure 2.4: *Radial Basis Function Support Vector Machine (RBF-SVM).*

2.4.2 *k*-Nearest Neighbour Classifier

Fix and Hodges [35] in 1951 developed the method of *k*-Nearest Neighbour (*k*-NN) which is known for it's simplicity and easy implementation. It is a non-parametric supervised learning algorithm and was later employed by Cover and Hart [36] to the pattern recognition purpose. Due to the high predictive performance of *k*-NN in real applications, this method was recognized as one of the top 10 methods in data mining [37].

For the classification of data points to a particular class, it makes use of nearest neighbour rule. In *k*-NN technique, a group of *k* observations from the training data which are nearest

to the test object are selected and the new observation is then assigned the class label of majority in selected *k*-Nearest Neighbours. This method uses Euclidean distance as a default measure of distance for locating the nearest neighbour(s) shown in Figure 2.5. The test observation is denoted by the symbol '?' is assigned the class label of the majority in neighbourhood. Euclidian distances is calculated for the test/new observation from all given training observations. For instance, if $k = 1$ the test object, say $\mathbf{x}$, is then allocated the class of the closest neighbour observation.

Euclidean is a good distance measure to use if the input variables are similar in type and is a default choice. Other choices of distance measure are also available. Manhattan distance is a good distance measure to use if the input variables are dissimilar in type. Hamming distance is used for categorical data. The choice of *k* and distance measure may affect the prediction accuracy of *k*-NN [38,39]. The odd value of *k* is usually taken in order to avoid ties between two classes.

Despite its simplicity, *k*-NN algorithm produces comparable results and even outperforms in some situations, as compared to other complicated learning models. However, the presence of non-informative features in the data may affect the performance of *k*-NN usually in high dimensional data [40]. General pseudo-code of *k*-NN is given in *Algorithm 4*.

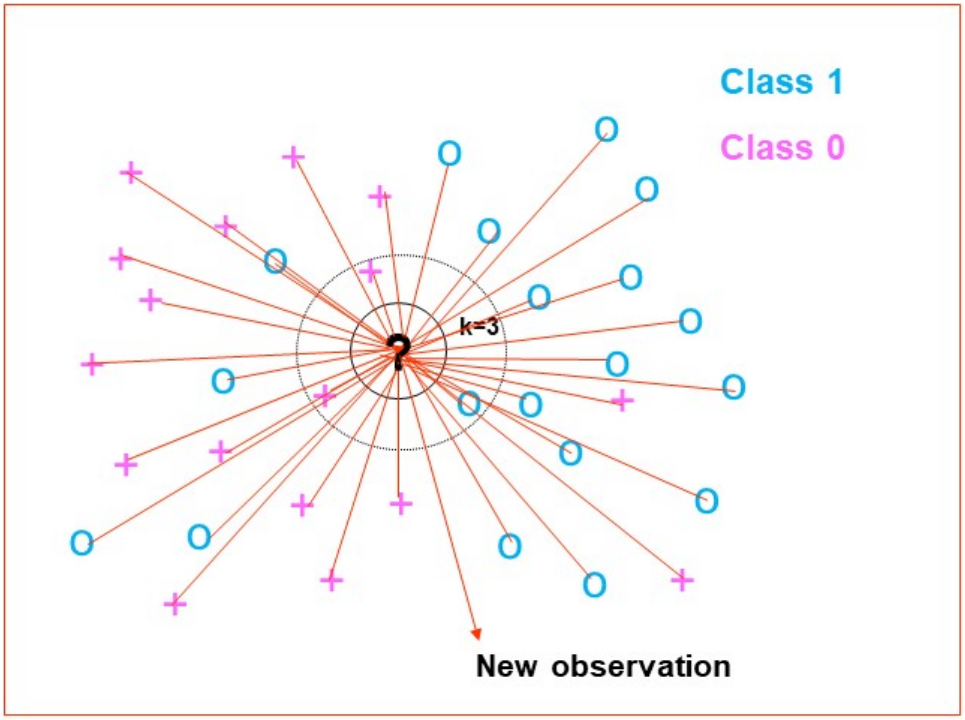


Figure 2.5: *k-Nearest Neighbour classification in a 2-dimensional feature space for k = 3. The lines represent the Euclidian distances from the test observation (denoted by '?') to all the training observations belonging to two classes. For k = 3, the nearest neighbours are identified within the circle and the test observation is assigned class 0 as the majority of selected nearest observations belong to class 0.*

Algorithm 4 pseudo code of k-Nearest Neighbours

n is the number of observations in the training set and k is the number of nearest neighbours.

1: **for** $i = 1 \rightarrow n$ **do**
   Find distance D_i between test observation and the ith training observation.
   Store the distance results to a distance vector.
2: **end for**
   Put the distance vector in ascending order.
   Select top k nearest data point to the test point based on the arranged distances.
   Predict class label for the test observations on the basis of class label of majority in k nearest observations.

2.5 Dimensionality Reduction

Dimensionality reduction is the procedure of bringing down the number of variables in high dimensional data and only a set of essential variables is chosen. In the recent age dealing with the big data is a challenging issue for the data science researchers. High dimensional data like images, gene expression, and next generation sequencing data often makes the implementation of classification methods difficult when the number of features are greater than the number of observations because large amount of data is required for learning algorithm. It is also known as curse of dimensionality. In this situation, one of the remedies is to reduce the dimensionality of large datasets. There are several techniques available for dimensionality reduction traditional techniques such as Principal Component Analysis (PCA) etc. Random Projection techniques are also used for dimensionality reduction, which project high-dimensional data onto a low-dimension. Therefore, several techniques based on Random Projection are developed. However, we have considered Random Projection Ensemble classifier in this work and focussed on LDA and QDA as base classifiers.

2.5.1 Importance of Dimensionality Reduction

Dimensionality reduction can be helpful in many scenarios due to it's vast applications. Usually the computational complexities arise while training the classifiers on high dimensional data, the running time becomes longer to process the data [41]. In terms of performance when the database is large enough then it leads to poor predictive performance in classification because it is hard to find an accurate classifier in a wide search space. There is also the probability of selecting a classifier which is over-fitted. Dimension-

ality reduction can reduce both computational cost and over-fitting by preserving most of the relevant information in the data to train the models accurately which potentially leads to better performance. High dimensional data requires a large storage capacity as the training data of size n. Reduced dimension minimizes storage complexity. Dimensionality reduction allows one to precisely visualize and plot the data by projecting them into 2 or 3 dimensions.

2.5.2 Random Projection Ensembles

Random Projection is a general technique used for dimensionality reduction [42]. The key idea of Random Projection is motivated by the Johnson Lindenstrauss Lemma [43] which states that a set of n points $\in R_q$ can be projected into a random space of q'-dimension without changing the Euclidian distance between any pair of points more than a factor of $1\pm\xi$ where $0 < \xi < 1$. The lower bound for q' is

$$q' > \frac{24}{3\xi^2 - 2\xi^3} \log(n), \tag{2.11}$$

There exists a mapping f: $R_q \rightarrow R_{q'}$ such that

$$(1 - \xi) \parallel \mathbf{x_i} - \mathbf{x_j} \parallel^2 \leq \parallel f(\mathbf{x_i}) - f(\mathbf{x_j}) \parallel^2 \leq (1 + \xi) \parallel \mathbf{x_i} - \mathbf{x_j} \parallel^2 . \tag{2.12}$$

In Random Projection, the original q-dimensional data is projected to a q'-dimensional subspace by using a random $q' \times q$- dimensional matrix R (random matrix) whose rows have unit lengths. Let $\mathbf{X}'_{q \times n}$ (data matrix) be the original set of n q-dimensional observations,

then

$$A_{q' \times n} = R_{q' \times q} \mathbf{X}'_{q \times n}, \tag{2.13}$$

is the projection of original data into a q'-dimensional subspace, assuming $q' < q$. A base classifier C_n^A on the projected training set is then applied. A general ensemble classifier based on Random Projections is then defined. By setting

$$W_n(\mathbf{x}) = W_n^{\mathbf{b_1}}(\mathbf{x}) := \frac{1}{\mathbf{b_1}} \sum_{b=1}^{\mathbf{b_1}} 1_{\{C_n^{A_b}(\mathbf{x})=1\},} \tag{2.14}$$

we define the random projection ensemble classifier for $\alpha \in (0, 1)$ (α is a voting threshold) as follows,

$$C_n^{RP}(\mathbf{x}) = \begin{cases} 1 & \text{if } W_n(\mathbf{x})) \geq \alpha, \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

Cannings and Samworth [42] argued, however, that the aggregation of all Random Projections Classifications is not always sensible in the sense that they may affect the class structure. Therefore, they suggested to divide the projection into dissimilar groups and retain the projections producing the smallest test error. In their approach, the Random Projections are divided into $\mathbf{b_1}$ non-overlapping blocks or groups by using "Haar" or "Gaussian" measure, and within each block (of size $\mathbf{b_2}$) the projections yielding the smallest error estimate on the testing data are then selected. Random Projection classifier then combines the predictions of applying a base classifier on the chosen projections to decide the final assignment by using a data driven threshold. Possible choices of base classifier are for example, Linear Discriminant Analysis (LDA) or Quadratic Discriminant Analysis

(QDA) discussed in the next section. The pseudo-code of Random Projection Ensembles algorithm is given in *Algorithm 5*.

Algorithm 5 Pseudo code of Random Projection Ensembles

- Start with the given training data consisting of all features. Decide on number of blocks in which the random projections are to be divided. Also decide on the size of each block.
- Generate the required $b_1 \times b_2$ number of random projections.
- Apply a base classifier "k-NN" for example, to each projection in a block.
- Estimate the error for example, training error or a leave-one-out estimate of the base classifier applied on each projections in each block.
- Choose the best projection from each block that yields the smallest estimate of the error. This will give us a total of b_1 selected projections.
- Use the selected projections in the generic classifier.
- Average over the selected projections in parallel to classify a new observation.

2.5.3 Linear Discreminant Analysis

Ronald A.Fisher developed Linear Discriminant Analysis (LDA) first time in 1936 [44]. Initially, LDA was described for a binary class problem and later it was generalized as "Multi-class Linear Discriminant Analysis" by C.R.Rao in 1948. Linear Discriminant Analysis is a dimensionality reduction technique used for the supervised classification problems. LDA projects the higher dimension space into a lower dimension space and maximizes the separation between groups or classes. LDA assumes that covariance across all response classes is the same and observation in each of the response classes is normally distributed with a class-specific mean and common covariance.

2.5.4 Quadratic Discreminant Analysis

Quadratic Discreminant Analysis (QDA) is an extension of Linear Discreminant Analysis (LDA) as a classifer that performs a non-linear separation of the data. As mentioned earlier, LDA is based on the assumption that the class contidtional distributions are normal with common covariance matrix. QDA does not make the homoskedasticity assumption. QDA assumes that covariance across all response classes is not the same i.e., different covariance for each of the target classes and observation in each of the response classes is normally distributed with a class-specific mean and covariance. QDA attempts to estimate the covariance of all reponse classes. There are larger number of parameters to be estimated. When the number of classes increases, the number of parameters grows quadratically.

Chapter 3

Assessing hyper-parameters and feature selection for Random Forest, Optimal Trees Ensemble and Random Projection Ensembles

3.1 Introduction

Supervised machine learning (ML) techniques for regression and classification depend on various hyper-parameters that have to be adjust before executing ML algorithms. Tuning strategy for determining a bunch of tunable hyper-parameters and their respective ranges and scales can be decided by the user. However, wrong choice in some scenarios can obstruct the quality, efficiency and quick convergence of the learning algorithm while tuning. The predictive performance of Random Forest can be enhanced by tuning the

hyper-parameters [47]. The hyper-parameters should be fine-tuned carefully. One of the most important issues associated to hyper-parameter tuning is over-fitting. It is very simple to tuned the model without model validation to the point where it starts overfitting without realizing it. Training algorithm is assumed to tune parameters in order to minimize a loss function, which sometimes goes too far then the model overfits and may not perform well with independent data. The best way to avoid such sub-optimal parameter values, is to make use of test dataset or cross-validation strategies while tuning the hyper-parameters. For Random Forest, one can use the out-of-bag (OOB) observations.

The predictive performance of various ML algorithms could be improved by selecting appropriate values of the hyper-parameters particularly in classification problems which could be done by evaluating a set of values during the tuning process. The tuning process enables us to get the best possible combination of hyper-parameters. In addition, feature selection empowers machine learning algorithm to train the classifiers faster, minimizes the cost and complexity of a model and improves the accuracy and performance of a model.

This chapter focuses on assessing hyper-parameters and feature selection for the ensemble methods such as: Random Forest, Optimal Tree Ensemble and Random Projection Ensembles discussed in detail in Chapter 2. The main aim is to investigate the influence of hyper-parameters and feature selection on these methods in particular.

3.2 Hyper-parameters of Random Forest, Optimal Trees Ensemble and Random Projection Ensembles

Hyper-parameters are usually fixed before the actual training process begins. There are various hyper-parameters that might be fine-tuned for improving the performance of the ensemble methods. Hyper-parameters of Random Forest (RF), Optimal Trees Ensemble (OTE) and Random Projection Ensembles (RP) are:

3.2.1 Hyper-parameters of Random Forest

Mainly, three parameters in the algorithm are considered for tuning a Random Forest (RF).

1. **ntrees**: number of trees grown in RF.
2. **nf**: number of variables for splitting in the node.
3. **ns**: number of observations in terminal nodes.

Further possibilities are tuning on choices of the split criterium, using pruning parameters etc.

3.2.2 Hyper-parameters of Optimal Trees Ensemble

Hyper-parameters regulating Optimal Trees Ensemble (OTE) are given as follows,

1. **t.initial**: number of trees initially grown in OTE.
2. **nf**: number of features for splitting the nodes of the trees.
3. **ns**: minimal number of samples in the nodes.

4. **p**: percent of the best t.initial trees to be chosen on their performance on OOB observations.

3.2.3 Hyper-parameters of Random Projection Ensembles

There are several hyper-parameters responsible for managing Random Projection Ensembles (RP), however, we have consider three main hyper-parameters described below,

1. $\mathbf{b_1}$: number of blocks.

2. $\mathbf{b_2}$: block size.

3. **base**: the base classifier can be one of "Linear Discriminant Analysis (LDA)","Quadratic Discriminant Analysis (QDA)" or any other base classifier.

3.3 Background of the methods

A fundamental problem of machine learning (ML) is to estimate the functional relationship between input features and response variable, based on a set of data points. Breiman [27] proposed the idea of Bagging (bootstrap aggregation), that builds each classifier in the ensemble on a randomly drawn sample (called bootstrap samples) of the training data. To return the final decision of the ensemble, a voting scheme called the majority rule is used. Ensemble methods tend to yield improved results when the base classifiers are accurate and diverse [48, 49]. An accurate classifier is the one that can correctly guess the class lable of observations. Classifiers are diverse if they don't repeat the same errors on new data points. Breiman [28] extended the idea of Bagging by combining it with the random

selection of features in order to construct a collection of decision trees with controlled variation. The study shows that incorporating the right amount of randomness can be effective in prediction by strengthen the individual predictors. Further researches have been done to decide on the number of trees in random forests by various authors, (for example, see [50–54]). To exploit both accuracy and diversity of the base classification and regression tree models, Khan et al. [2,30] proposed the idea of optimal trees selection based on their individual and collective performance in the ensemble which is modification over Random Forest. The best performing trees are selected based on their out-of-bag accuracy. Starting from the best performing tree, if the subsequent selected tree is contributing to the ensemble then the tree is added to the final ensemble otherwise it is discarded. The method is named as Optimal Trees Ensemble (OTE) and used both for regression and classification. The performance of the proposed method is evaluated on 34 standard benchmark datasets consist of 20 classification problems and 14 regression problems in comparison with k-NN, Tree, Random Forest, Support Vector Machine (with Linear, Radial, Laplacian and Bessel) and Node Harvest (NH). The method has performed better than the others in terms of prediction accuracy.

Principal Component Analysis (PCA) is a popular technique available for dimensionality reduction. However, when the number of features is more than the number of observations these methods may perform very poor. To avoid the curse of dimensionality, one way is to project the data into a lower dimensional space. The idea of Random Projections for dimensionality reduction was stimulated by Johnson–Lindenstrauss Lemma [43]. Durrant [55] in his work trained a linear classifier on a single projection and calculated Vapnik-Chervonenkis bound on generalization error. Other studies showed that aggre-

gation of many Random Projection is beneficial, see [56–58] for instance. Canning [42] proposed Random Projection Ensemble Classifier for the classification of high dimensional data where the projections are divided into non-overlapping blocks because sometimes the aggregation of all Random Projections in classifications are not useful in the sense that they may affect the class structure in the data. The classification performance of Random Projection Ensemble Classifier is compared with several machine learning methods such as Random Forest, Optimal Trees Ensemble, k-nearest neighbours, ESknn, Radial SVM, Linear SVM, and the results reveal that the lowest error rate is achieved in 23 out of 36 simulated and real data. Although the authors have applied the methods (Random Forest, Optimal Trees Ensemble and Random Projection Ensembles) on a number of benchmark datasets, it's utility under big data situations and feature selection has not been explored.

3.4 Benchmark datasets

To cover the spectrum of materials used in machine learning experiments and to assess the influence of various hyper-parameters and feature selection in classification tasks, we have selected a set of standard benchmark datasets from Machine Learning Repository [59] and other sources (for a brief description of the selected datasets see Appendix A). These datasets are selected because of their flexible nature, various types and number of features and observations.

3.5 Experimental design

We evaluate RF, OTE and RP on benchmark datasets for the classification problem by comparing their predictive performance in a series of experiments. We compare classification error rates for RF, OTE, RP_{LDA_5}, and RP_{QDA_5} on 19 benchmark datasets. Experiments performed on these datasets are designed as follows:

3.5.1 Experimental setup for benchmark datasets

For assessing the influence of the hyper-parameters of RF and OTE we have tuned various hyper-parameters such as, **ns** (node size), **nf** (number of features), **t.initial** (number of trees initially grown in OTE), and **ntrees** (number of trees grown in RF). Each dataset is divided into training part consists of 90% while the testing/validation part contains of 10% of the data. For tuning the **ns** we have taken the values (1, 5, 10, 15), for tuning **t.initial** and **ntrees** we have chosen the values (500, 1000, 1500) and for tuning **nf**, we have tried ($\sqrt{(q)}$, $\sqrt{(q/2)}$, $\sqrt{(q/3)}$, $\sqrt{(q/4)}$, $\sqrt{(q/5)}$). The purpose of trying the different values is to assess their effect on the predictive performance of the methods. Each experiment is repeated 100 times and average all the runs for the final result.

The effect of main hyper-parameters of OTE, i.e., proportion of trees **p** is evaluated for 19 benchmark datasets. We have computed misclassification rate of OTE by considering **t.initial** equal to 1000 and choose different values of **p** (10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%) for the benchmark datasets in comparison with RF. While RF has been computed by fixing **ntrees**=1000. We also calculate the mean number of trees corresponding to each error rate for the same datasets by taking different training data sizes (n) i.e., 30%,

60% and 90%.

Furthermore, the error rates for RP are calculated by considering different values of the hyper-parameters i.e., $\mathbf{b_1}$ (number of blocks) and $\mathbf{b_2}$ (block size). In each scenario, the data is split to create a training set of size n and the remaining data is treated as a test set and various values of $\mathbf{b_1}$ and $\mathbf{b_2}$ are considered. In the first scenario we fix $\mathbf{b_1} = 100$ and $\mathbf{b_2} = 20$, in the second scenario we take $\mathbf{b_1} = 300$ and $\mathbf{b_2} = 30$ and in the third senario $\mathbf{b_1} = 500$ and $\mathbf{b_2} = 50$, all using "Haar" distributed projections while each experiment is repeated 100 times. Classification error rates are calculated for RP, using base=LDA (Linear Discriminant Analysis) and base=QDA (Quadratic Discriminant Analysis).

We also investigate RF, OTE, and RP to feature representation of benchmark datasets. We have compared classification error rates for RF, OTE, RP_{LDA_5}, and RP_{QDA_5} by applying the feature selection method using LASSO. In the case of RF and OTE, the error rates are calculated by considering an ensemble of size 1000, and each experiment is repeated 100 times. In case of RP, the results of RP_{LDA_5} and RP_{QDA_5} are calculated for $q' = 5$ using "Haar" as the projection method [42]. We have taken $\mathbf{b_1} = 500$ and $\mathbf{b_2} = 50$, and each experiment is repeated 100 times. Testing part of the data is used on the final ensemble. Final result is the average of all 100 runs.

3.5.2 Softwares and packages

All the datasets are analysed using the R programming language [60] for carrying out the results and comparison of the methods in this study. We have implemented Optimal Trees Ensemble using R package "OTE" [61]. Similarly, Random Forest is implemented through the R package "randomForest" [62]. While for Random Projection Ensembles we

have used R package "RPEnsemble" [63]. LASSO has been applied through the R package "glmnet" [64].

3.5.3 Assessment of hyper-parameters for Optimal Trees Ensemble and Random Projection

Table 3.1 displays the misclassification rate of OTE for three datasets (Spambase, Musk, Leukaemia) in comparison with RF. It can be observed that the mean number of trees increases by increasing the percent of best trees (**p**) for all the datasets. It has been noticed that increasing the value of **p**, improves the predictive performance of OTE for the datasets up to some extent for most of the datasets. However, once the improvement is gained then increasing the value of **p** adversely affect the predictive performance of the method. For instance, the error rate for the Spambase dataset increases as **p** increases. Although increasing the value of **p**, the error rates for Musk and Leukaemia datasets show a slight increase or decrease. It can also be observed that for each dataset the error rate minimizes with increase in training set size (n) for RF and OTE. The best results are highlighted in bold. For an overview on all other datasets see Tables in Appendix B.

Table 3.1 *Error rates and mean number of trees for* `OTE`.

	Spambase						Musk						Leukaemia					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
p	error rate	mean trees	error rate	mean trees	error rate	mean trees	error rate	mean trees	error rate	mean trees	error rate	mean trees	error rate	mean trees	error rate	mean trees	error rate	mean trees
0.1	0.0589	51.98	0.0516	52.69	0.0472	53.55	0.1991	50.97	0.1313	51.64	**0.1006**	51.64	0.1254	29.28	0.0344	37.62	0.0242	34.07
0.2	0.0587	101.78	**0.0512**	103.13	**0.0468**	104.3	0.1945	101.39	0.1288	101.88	0.1031	102.56	**0.1244**	58.62	**0.0317**	85.34	0.0228	81.74
0.3	0.0585	151.61	0.0514	152.59	0.0472	154.28	0.1942	152.98	0.1288	151.5	0.1016	152.87	0.1246	87.81	0.0344	133.52	0.0228	133.59
0.4	0.0584	201.77	0.0512	202.71	0.0470	204.92	0.1945	203.31	**0.1281**	201.27	0.1027	202.06	0.1268	116.85	0.0386	179.74	0.0228	187.92
0.5	0.0585	251.69	0.0514	253.14	0.0472	254.28	0.1938	253.35	0.1287	251.48	0.1047	251.5	0.1286	145.57	0.0406	226.39	**0.0214**	241.79
0.6	0.0584	302.21	0.0514	302.46	0.0473	305.06	0.1939	303.4	0.1283	301.77	0.1029	301.39	0.1284	173.6	0.0431	273.32	0.0242	295.81
0.7	**0.0583**	352	0.0515	352.41	0.0475	355.18	**0.1937**	353.5	0.1293	351.78	0.1058	349.92	0.1312	201.50	0.0441	318.51	0.0214	349.10
0.8	0.0584	401.61	0.0516	402.17	0.0475	405.03	0.1943	403.47	0.1299	401.66	0.1058	398.34	0.1338	228.35	0.0458	362.65	0.0214	403.19
0.9	0.0585	451.88	0.0516	451.30	0.0476	454.74	0.1939	454.37	0.1296	452.13	0.1070	448.32	0.1428	253.61	0.0479	406.59	0.0214	455.78
1	0.0585	501.53	0.0518	501.47	0.0477	504.83	0.1939	504.22	0.1301	501.61	0.1079	496.91	0.2060	278.60	0.0489	446.28	0.0228	505.42
RF	0.0576		0.0511		0.0476		0.1873		0.1267		0.1087		0.1160		0.0513		0.0428	

Note: We have considered different values of **p** in comparison with the error rates of RF for different training set sizes (30%, 60%, 90%) on Spambase, Musk and Leukaemia datasets. Here n is the training set size. OTE is computed by considering **t.initial**=1000 and different values of **p**. RF is computed by fixing **ntrees**=1000. Each experiment is repeated 100 times. The best results are highlighted in bold.

Table 3.2 presents the misclassification rate for RP_{LDA_5} and RP_{QDA_5} for the various values of number of blocks ($\mathbf{b_1}$) and size of the block ($\mathbf{b_2}$). The results illustrate that increasing $\mathbf{b_1}$ and $\mathbf{b_2}$ minimize the misclassification rate. As it can be observed that RP with LDA and QDA provides the best results on 11 datasets out of 19 datasets for $\mathbf{b_1}$=500 and $\mathbf{b_2}$=50. For $\mathbf{b_1}$=300 and $\mathbf{b_2}$=30, RP with LDA and QDA produces the better results on 6 datasets while for $\mathbf{b_1}$=100 and $\mathbf{b_2}$=20, RP with LDA and QDA gives the better results on 3 datasets out of 19 datasets. However, it is notable that increase in $\mathbf{b_1}$ and $\mathbf{b_2}$ slightly improves the results and there is not much gain in the predictive performance of the method.

Table 3.2 *Classification error rates for RP_{LDA}, and RP_{QDA}.*

Data sets	q'=5, $\mathbf{b_1}$=100, $\mathbf{b_2}$=20		q'=5, $\mathbf{b_1}$=300, $\mathbf{b_2}$=30		q'=5, $\mathbf{b_1}$=500, $\mathbf{b_2}$=50	
	$\mathbf{RP}_{LDA_5}$	$\mathbf{RP}_{QDA_5}$	$\mathbf{RP}_{LDA_5}$	$\mathbf{RP}_{QDA_5}$	$\mathbf{RP}_{LDA_5}$	$\mathbf{RP}_{QDA_5}$
Eyestate	0.3736	0.4683	0.3705	0.4624	**0.3684**	0.4562
Hepatitis	0.1875	0.1687	0.1650	0.1625	0.1700	**0.1537**
Parkinson	0.1710	0.1536	0.1794	**0.1463**	0.1715	0.1468
Body	0.0180	0.0166	**0.0145**	0.0170	0.0150	0.0154
Thyroid	0.0488	**0.0422**	0.0484	0.0425	0.0482	0.0425
WDBC	0.0549	0.0571	0.0596	0.0568	**0.0531**	0.0557
WPBC	0.2189	0.2236	**0.2173**	0.2226	0.2210	0.2189
Ionosphere	0.0937	0.0540	0.0902	0.0554	0.0894	**0.0528**
Oil-spill	0.0445	0.0420	0.0442	0.0411	0.0429	**0.0398**
Spambase	0.1785	0.3159	0.1752	0.3077	0.1670	**0.3051**
Glaucoma	0.1040	0.1320	**0.1**	0.124	0.1020	0.1285
Nki 70	0.1764	0.1850	**0.1728**	0.1878	0.1814	0.1885
Musk	0.2666	0.1656	0.2641	0.1643	0.2606	**0.1635**
Breast	**0.3600**	0.3987	0.3625	0.3987	0.3712	0.3837
Prostate	0.1260	**0.1050**	0.1210	**0.1050**	0.1160	0.1330
Leukaemia	0.0971	0.1328	0.1042	0.0985	**0.0885**	0.14
Adenocarcinoma	0.1712	0.1537	0.1600	0.1475	0.1525	**0.1162**
Lung	0.0361	0.0322	0.0205	0.0216	**0.0155**	0.0161
Ovarian	0.0236	0.0260	0.0192	0.0232	0.0212	**0.0180**

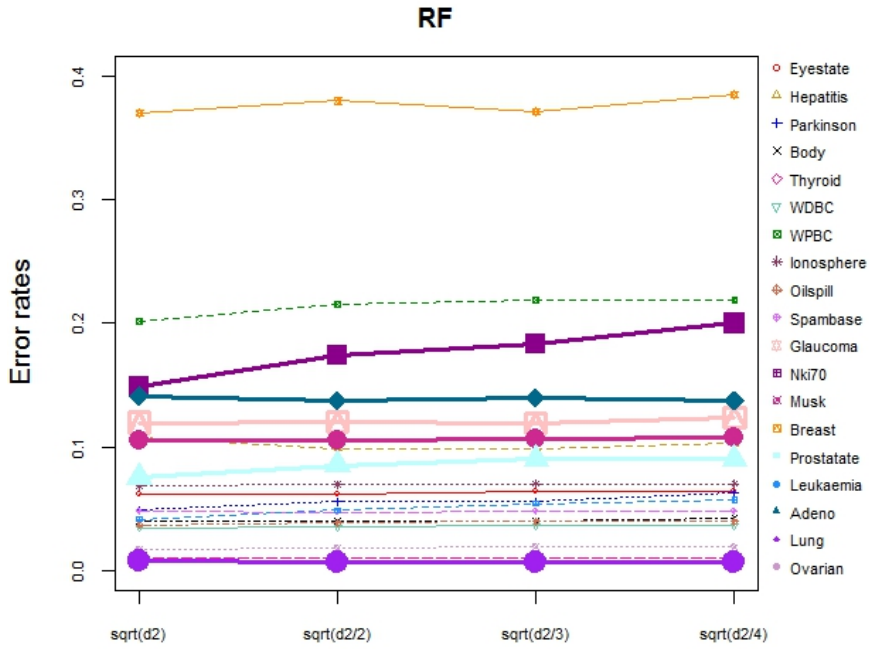
Note: Taking different values of $\mathbf{b_1}$=100, $\mathbf{b_2}$=20, $\mathbf{b_1}$=300, $\mathbf{b_2}$=30 and $\mathbf{b_1}$=500, $\mathbf{b_2}$=50 respectively. The best results are highlighted in bold.

The plots shown in Figure 3.1 present the effect of number of features (`nf`) choosen for splitting the node at random in RF and OTE for classification. Both the plots show fluctuations for various values of **nf** = ($\sqrt{(q)}$, $\sqrt{(q/2)}$, $\sqrt{(q/3)}$, $\sqrt{(q/4)}$, $\sqrt{(q/5)}$) for some datasets, Nki70, Breast, Adeno for example. Here, the result shows that this parameter is data dependent and hence shall be fine tuned in order to achieve better performance for both the methods.

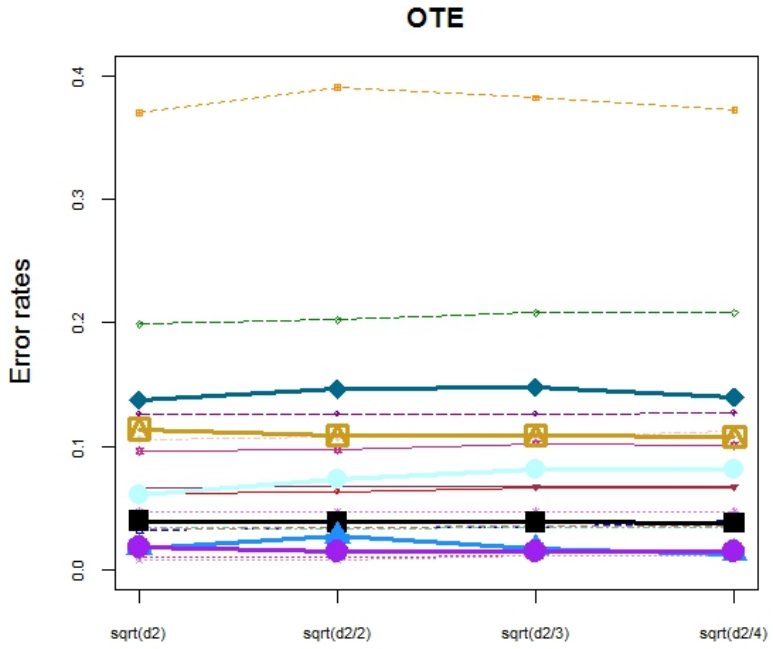
Figure 3.2 shows the effect of node size (`ns`) and misclassification rate on datasets for RF and OTE. Our benchmark study reveals that we should stick to default value of this hyper-parameter which is 1 for both the methods. It is obvious from the plots that the error rates for larger datasets such as Prostate, Leukaemia and Ovarian are not much affected by increasing the node size. In the case of datasets with smaller number of features WPBC, Nki70, Parkinson for example, increasing node size decreases the predictive accuracy of the classifiers.

The effect of number of trees grown on bootstrap sample from training data to develop an ensemble and the error rate are shown in Figure 3.3. The results illustrate that by increasing this number, better results can be achieved in term of prediction accuracy for most of the datasets. The benchmark experimental study suggests that an ensemble of size 1000 or 1500 is suitable for RF and OTE.

Figure 3.4 shows the effect of the main hyper-parameter (**p**) of OTE, is the number of best trees selected to develop the final ensemble. Various values of p demonstrate different behaviour of the method. We consider **p** = (10%, 20%, 30%, 40%, ... , 100%) to see the effect of percent of best trees for classification. It is clear from plot that p can be fine-tuned depends on the nature of the data. As discussed earlier, our plot shows that after achieving the gain in predictive accuracy increasing the percent of best trees can negatively affect the ensemble e.g., this pattern can be seen for musk, leukaemia datasets.

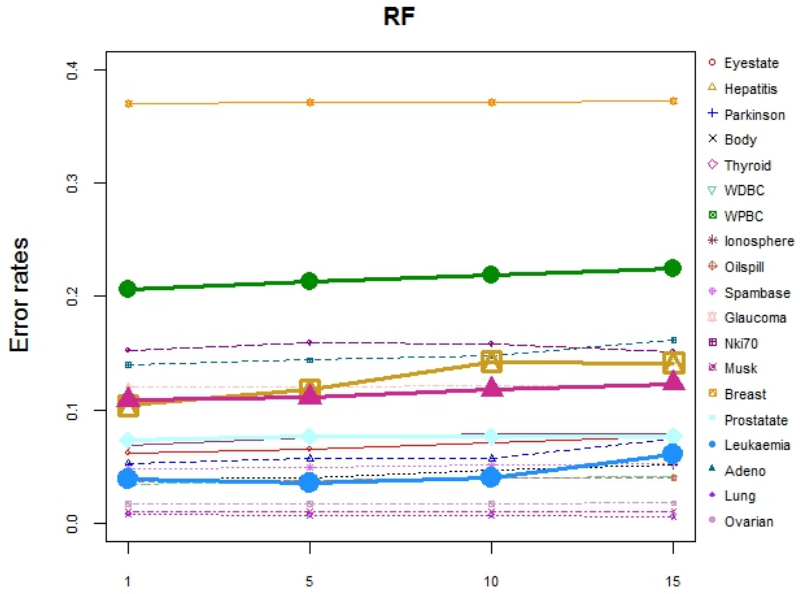


(Number of features)



(Number of features)

Figure 3.1: *The effect of the number of features selected randomly for node splitting and misclassification error rate on datasets. The figures show that this parameter can be fine-tuned.*



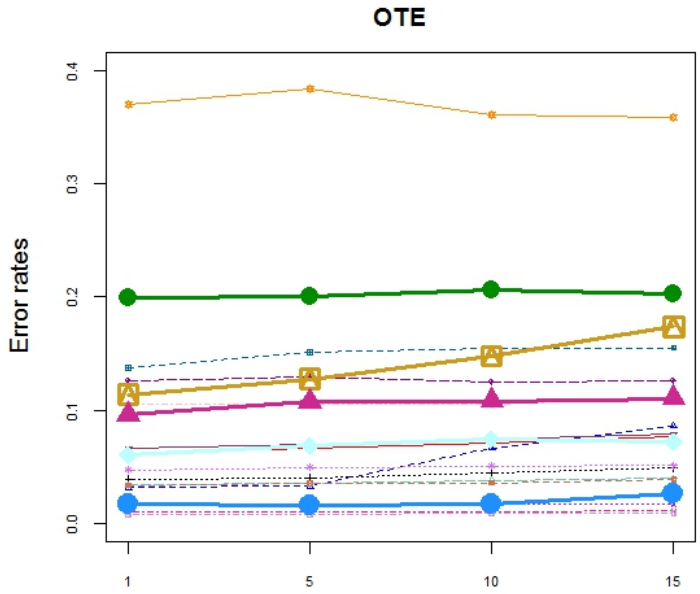


Figure 3.2: *The effect of nodesize and misclassification error rate on datasets. The figures show that the default value of node size is appropriate.*

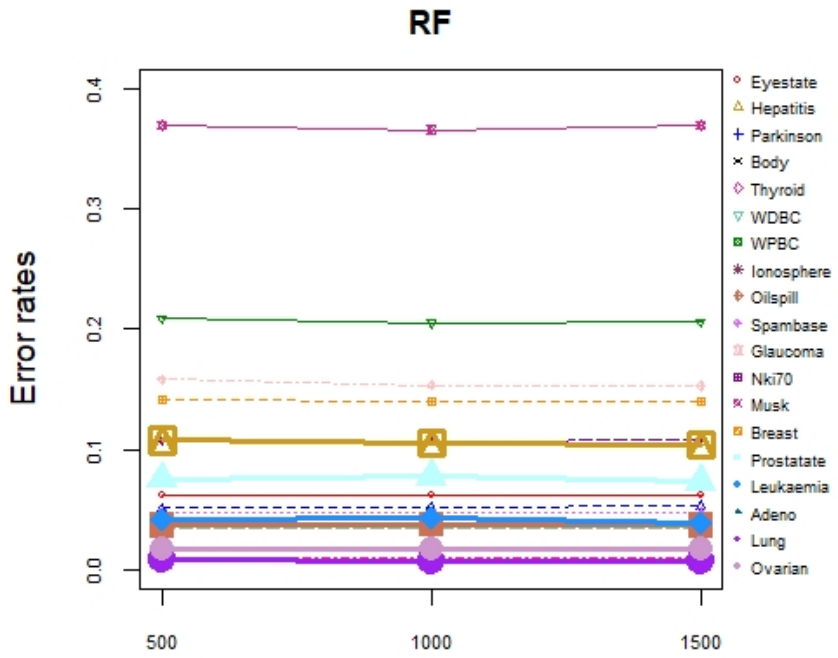


Figure 3.3: *The effect of number of trees and misclassification error rate on datasets. The figures show that an ensemble of size of 1000 or 1500 is appropriate for both the methods.*

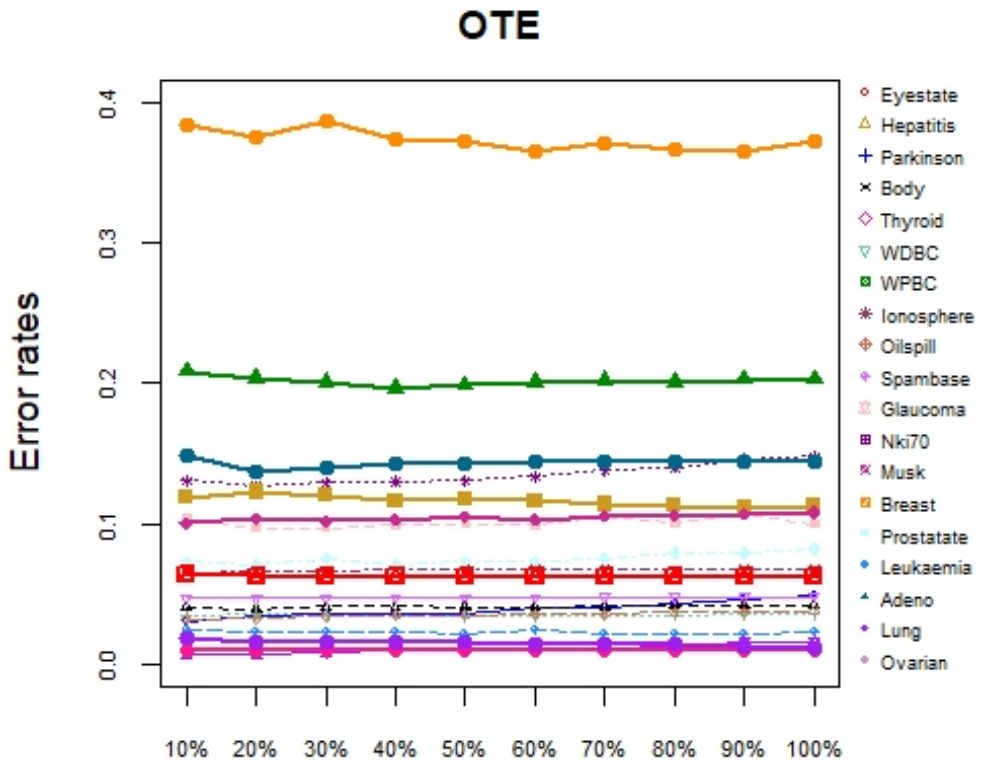


Figure 3.4: *The effect of percent of trees in the initial set on and misclassification error for the datasets using* OTE. *The figure shows fluctuations therefore,* **p** *can be fine-tuned.*

3.6 Feature selection for the methods

There are several types of feature selection methods however, we have considered LASSO as a feature selection method which can be explained briefly as follows:

3.6.1 Least Absolute Shrinkage and Selection Operator (LASSO)

The LASSO is a shrinkage and selection method for linear regression [65]. It's objective is the minimization of the usual sum of squared errors, with a constrain on the sum of the absolute values of the coefficients. LASSO basically shrinks the co-efficients of non-important variables to zero. The significant variable are assigned non-zero coefficients.

We assume that the dataset is $(\mathbf{x}_i, y_i)$, where y_i $(i = 1, ..., n)$ is a continuous variable. The LASSO solution is the minimization of α:

$$\alpha = \sum_{i=1}^{n} (y_i - \beta^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^{q} |\beta_j| \tag{3.1}$$

where $\beta = (\beta_1, ..., \beta_q)$ and $\lambda \geq 0$ is a penalty term. A L_1 regularization/constraint is used, which adds an L_1 penalty equal to the absolute value of the magnitude of coefficients and as a result some coefficients may become zero and eliminated [66]. In the linear regression setting, LASSO estimation has been taken into consideration by [65]. LASSO was considered originally for estimating the coefficients in regression models. However, our interest is in using LASSO for binary classification problem through `glmnet` R package [67]. For the benchmark datasets the minimum value of λ has been chosen via cross validation because the number of features in each dataset is different.

3.6.2 Feature selection via LASSO on benchmark datasets

Table 3.3 represents the misclassification rates without feature selection and with feature selection using LASSO for RF, OTE, RP_{LDA_5} and RP_{QDA_5}. The results reveal that feature selection, using LASSO improves predictive performance of RF for 11 datasets out of 19 datasets. Similarly, LASSO minimizes the error rate for RP_{LDA_5} and RP_{QDA_5} on 16 and 13 datasets respectively, in addition to reducing the computational burden. However, it can be seen that LASSO is not improving the predictive performance of OTE as it has minimized the error rate for only two of the datasets i.e., Leukaemia and Adenocarcinoma. It is interesting to observe that for the datasets with small number of features, LASSO is slightly improving the results e.g., Eyestate, Body, Thyroid, Oil-Spill. On the other hand, for the datasets with relatively a large number of features such as; Breast, Prostate, Leukaemia, Ovarian, LASSO provides better results in terms of predictive accuracy of the methods.

Table 3.3 *Classification error rates for RF, OTE, RP_{LDA_5}, and RP_{QDA_5}.*

	Without feature selection				Feature selection via LASSO			
Data sets	**RF**	**OTE**	**RP_{LDA_5}**	**RP_{QDA_5}**	**RF**	**OTE**	**RP_{LDA_5}**	**RP_{QDA_5}**
Eyestate	0.0616	0.0628	0.3684	0.4562	0.3305	0.3439	0.3973	**0.4538**
Hepatitis	0.1050	0.1225	0.1700	0.1537	0.1235	0.1661	**0.1647**	0.1789
Parkinson	0.0521	0.0336	0.1715	0.1468	**0.0494**	0.0489	**0.1631**	**0.1357**
Body	0.0384	0.0390	0.0150	0.0154	**0.0339**	0.0494	**0.0145**	**0.0131**
Thyroid	0.0098	0.0099	0.0482	0.0425	**0.0094**	0.0113	**0.0481**	**0.0423**
WDBC	0.0345	0.0356	0.0531	0.0557	0.0398	0.0412	**0.0371**	**0.0396**
WPBC	0.2052	0.2036	0.2210	0.2189	**0.2036**	0.2189	**0.2194**	0.2300
Ionosphere	0.0680	0.0800	0.0894	0.0528	0.0700	0.0840	0.0922	**0.0505**
Oil-spill	0.0368	0.0325	0.0429	0.0398	**0.0338**	0.0335	**0.0422**	0.0404
Spambase	0.0476	0.0473	0.1670	0.3051	0.0488	0.0553	**0.1608**	**0.2807**
Glaucoma	0.1100	0.0970	0.1020	0.1285	**0.1015**	0.1235	0.1275	0.1495
Nki 70	0.1535	0.1264	0.1814	0.1885	0.1664	0.1842	**0.1462**	**0.1607**
Musk	0.1070	0.1022	0.2606	0.1635	**0.1010**	0.1214	**0.2489**	0.1643
Breast	0.3662	0.3750	0.3712	0.3837	**0.3425**	0.3850	**0.3262**	0.3312
Prostate	0.0770	0.0710	0.1160	0.1330	0.0833	0.1233	**0.1044**	**0.1055**
Leukaemia	0.0428	0.0228	0.0885	0.14	**0.0071**	**0.0085**	**0.0471**	**0.0242**
Adenocarcinoma	0.1400	0.1362	0.1525	0.1162	**0.0800**	**0.1062**	**0.0787**	**0.1037**
Lung	0.0066	0.0161	0.0155	0.0161	0.0140	0.0300	**0.0088**	**0.0077**
Ovarian	0.0168	0.0072	0.0212	0.0180	**0**	0.0160	**0**	**0.0012**

Note: Taking ensemble of size 1000 for RF, OTE and $\mathbf{b_1}$=500, $\mathbf{b_2}$=50 and q'=5 for RP. The minimum value of λ has been chosen via cross validation in LASSO. The improved results are highlighted in bold in the table.

3.7 Discussion

Based on the analysis of 19 benchmark datasets, two aspects of RF, OTE and RP have been investigated. The first aspect is the effect of hyper-parameters on the predictive performance of the methods. In the second phase, feature selection has been considered to evaluate the performance of the three selected ensemble methods.

For RF and OTE the hyper-parameters such as trees grown in RF (**ntrees**), node size (**ns**), number of trees initially grown in OTE (**t.initial**), and number of features (**nf**) have been investigated and misclassification rates are calculated based on the training sets of all the 19 benchmark datasets. The size of an ensemble in RF and OTE depends on the number of trees taken i.e., by increasing this number better results are achieved in terms of prediction accuracy. The results have revealed that ensemble of size or 1000 or 1500 is suitable for both the methods. Selecting optimal trees for accuracy and diversity, Khan et al., [30] has suggested the value of **p** as 20%. Our results in Fig 3.4 show that value of selecting optimal trees, for accuracy and diversity can be fine-tuned depends on the nature of the data. However, once the improvement is achieved adding further trees adversely effect the ensembles. For instance, the OTE unnecessarily selects a large number of trees in the final ensemble for large **p** values, which is not required as shown in Table 3.1. The next hyper-parameter is the number of features (**nf**) taken randomly at each node while growing the trees. Although a generally accepted value for this hyper-parameter is square root of number of features ($\sqrt{q}$), the benchmark experimental results reveal that this value is also data dependent and hence shall be fine-tuned for both RF and OTE see Fig 3.1. Another parameter which actually determines the size of a tree is **ns** (node size), is less influential

when it is tuned. Our benchmark experimental study reveals that the default value of **ns** for RF and OTE is appropriate. While tuning the parameters such as number of features, node size and number of trees for RF and OTE both the methods behave alike or show the same pattern because OTE uses RF methodology for growing the trees. Additionally, the training set size (**n**) also affects the predictive performance of these methods because the classifiers can be well trained by providing training data with large number of observations.

In RP, the projections are divided into blocks and projections with the smallest error rates are chosen, therefore, $\mathbf{b_1}$ (number of blocks) and $\mathbf{b_2}$ (block size) need to be selected carefully. As claimed by Canning et al. [42] $\mathbf{b_1}$ should be chosen larger. However, it is notable that as the values of $\mathbf{b_1}$ and $\mathbf{b_2}$ increase, the misclassification rates become smaller but this can be observed from the experiments that improvement is not very obvious when $\mathbf{b_1}$ is sufficiently large. However, small values of $\mathbf{b_1}$ and $\mathbf{b_2}$ reduce the computational cost. Selecting the best projections in RP, involves a huge amount of computational costs when the values of $\mathbf{b_1}$ and $\mathbf{b_2}$ are very large.

In the second part of the analysis the misclassification rates are calculated without using feature selection and with feature selection. For the benchmark datasets, feature selection via LASSO has improved the predictive performance of RF for 11 out of 19 datasets. While for RP_{LDA} it has improved the predictive performance for 16 out of 19 benchmark datasets and for RP_{QDA} it has minimised the error rates of 13 datasets, in addition, to reducing the computational cost. It is notable that for the datasets with lower number of features, LASSO has slightly improved the error rates for RF and RP for instance, Eyestate, Parkinson, Thyroid, Ionosphere and Oill-spill. However, for the datasets with relatively larger number of features LASSO has considerably minimised the error rates for RF and RP

e.g Breast, Leukaemia, and Ovarian. Furthermore, feature selection has only improved the performance of OTE for 2 out of 19 datasets that means OTE is less influenced by feature selection.

The analysis in this chapter revealed that some of the hyper-parameters can be fine tuned for Random Forest and OTE such as, number of features for splitting the nodes, percent of best trees (in OTE), number of trees grown in the forest (in Random Forest), number of trees initially grown (in OTE). However, in the case of Random Projection Ensembles, large values of $\mathbf{b_1}$ and $\mathbf{b_2}$ slightly improve the results but increase the computational cost. Similarly, the results showed that feature selection improved the predictive performance of Random Forest and Random Projection Ensembles. However, OTE is less influenced by feature selection.

Chapter 4

Optimal Random Projection Trees Ensemble

4.1 Introduction

In most of the tree-structured ensemble methods such as Bagging and Random Forest, the trees are grown on bootstrap samples from the training data and the base models are integrated to form the final ensemble. The number of base models remain constant for the final ensemble. The prediction error of an ensemble of trees highly depends on the individual strength of trees and the correlation among the trees in the ensemble. Trees will be considered as strong if their individual error rate is low, and the trees are uncorrelated. It means that the predictive accuracy and efficiency of the overall forest depend on the individual strength of trees in the ensemble and the amount of correlation between the error rates of two independent trees. The overall prediction error of a Random Forest can be found using Breiman's [28] upper bound:

$$PE^* \leq \bar{\rho} PE_j \tag{4.1}$$

where $j = (1, ..., \mathcal{T})$, where $\mathcal{T}$ is the number of trees grown in the forest, PE^* gives the overall prediction error of the Random Forest, $\bar{\rho}$ denotes the weighted correlation between residuals from two independent trees and the prediction error of j^{th} tree is denoted by PE_j in the forest. Secondly, limiting the number of trees in the ensemble by eliminating trees yielding higher error rates also improves the prediction accuracy of an ensemble.

Selecting the accurate and diverse trees on the grounds of their individual and collective performance in an ensemble has recently been focused on for classification and regression purposes [2,30]. Diversity in the base models could be introduced by using other techniques like random projection method. Our proposed method Optimal Random Projection Trees Ensemble (ORPTE) has considered using tree selection in conjunction with the idea of randomly projecting the feature space into a smaller dimension. This proposed method is different from the Random Projection Ensembles in the sense that Random Projection Ensembles method is based on selecting the best projections from non-overlapping blocks producing the smallest estimate of error in a block by applying a base classifier such as LDA, QDA and *k*-NN. However, ORPTE is a tree-structured ensemble method consisting of optimal trees based on out-of-bag classification accuracy, choosen from an initial pool of trees grown on random projections of the training data without any arrangment of the projections into blocks. Using a total of 33 benchmark datasets, the results from ORPTE are compared with those of Random Forest, Optimal Trees Ensemble, Random Projection with QDA, Extreme Gradient Boosting (XGBoost), *k*-Nearest Neighbours (*k*-NN) and Support Vector Machine (with Linear, Radial kernels).

4.2 Objectives of the work

1. Extending the idea of Optimal Trees Ensemble to Optimal Random Projection Trees Ensemble.

2. Projecting the data into lower feature subspace.

3. Reducing size of an ensemble by selecting a portion of trees from the forest initially grown.

4. Attaining better or comparable predictive performance by using projections and reducing size of ensemble.

4.3 Optimal Random Projection Trees Ensemble (ORPTE)

Intuitively, an ensemble of accurate and diverse classification/regression trees may perform better than a forest comprising of simple trees. This intuition also holds for the Random Forest [28] and led to the idea of optimal trees ensemble (OTE) for regression and classification [2, 30]. OTE selects accurate and diverse trees based on individual and collective performance of the trees to construct the final ensemble. Further improvement could be brought by projecting the data into lower dimension for growing each tree and selecting a proportion of best trees (yielding the smallest out-of-bag error) to form the final forest. The intuition behind the proposed idea is that trees grown are random as they are built on different random projections of the training data along with the randomness caused by bootstrapping. Furthermore, subsets of the projected features are considered for splitting at each node for constructing trees in Random Forest style for more randomness. Accuracy

is maintained in the final ensemble by discarding trees that perform weakly on the training data. For constructing an ensemble of the best trees, the training data $\mathcal{L} = (\mathbf{X}, Y)$ is projected into $\mathcal{T}$ random projections each with $q' < q$ dimension where q is original and q' is the lower dimension. Random Projection uses random matrix in order to project the original dimension into a lower dimensional space [68,69]. We assume that $A^r_{q' \times n}$ is given by

$$A^r_{q' \times n} = R^r_{q' \times q} \mathbf{X'}^r_{q \times n} \tag{4.2}$$

is the rth, $r = 1, 2, 3, \ldots, \mathcal{T}$, projection of original data into a q'-dimensional subspace for each. Where $\mathbf{X}'_{q \times n}$ is the original data matrix. For dimensionality reduction an efficient technique using Johnson-Lindenstrauss Lemma [43] discussed in Chapter 2, involves a random matrix $R^r_{q' \times q}$ is the rth matrix whose elements are identically distributed (i.i.d) and independent. The matrix $R^r_{q' \times q}$ could be produced randomly by using any of the several well known methods like "Haar", "Gaussian", "axis", etc. In this thesis, we have produced a random matrix with elements generated by using "Gaussian distribution", where $r_{i,j} \sim$ N(0, 1) is one of the commonly used distribution in terms of simplest analysis.

Classification tree is grown on a bootstrap sample from the randomly projected data $A^r_{q' \times n}$ using the Random Forest style algorithm. That is, a random subset of $q_{mtry} < q'$ features are considered at each node of the tree for splitting. Each of the trees is assessed by out-of-bag (OOB) error rate. The OOB error rate is calculated from observations that do not play any role in growing the individual classification tree.

Let n^r_{oob} be the total number of OOB observations from the *rth* bootstrap sample and $\hat{y}$ be the estimated class label for the actual label y of observation in the bootstrap sample.

Then the error rate $\widehat{err}_r, r = 1, 2, 3, \ldots, \mathcal{T}$ for the *rth* tree grown on the r^{th} bootstrap sample is

$$\widehat{\mathbf{err}}_\mathbf{r} = \frac{\sum_{i=1}^{n^r_{oob}} \mathbf{I}(\hat{y}_i \neq y_i)}{n^r_{oob}}, \tag{4.3}$$

where $\mathbf{I}(.)$ is an indicator function with values 0 or 1 such that

$$\mathbf{I}(\hat{y}_i \neq y_i) = \begin{cases} 1 & \text{if } \hat{y}_i \neq y_i \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

The trees are then arranged in ascending order with respect to their error rates $\widehat{err}_r, r = 1, 2, 3, \ldots, \mathcal{T}$ calculated by using Equation 4.3. Trees with high error rate are discarded from the forest and the remaining are integrated together for the ultimate ensemble.

4.3.1 Majority voting

Majority voting (plurality voting) is a decision rule widely used in ensemble methods where the class lable is estimated by majority votes. In the proposed method, each model makes a prediction or vote for each test observation and the final prediction about the class label is made on the basis of majority votes received. In other words, the estimated class label for a particular sample is the mode of the class labels individually predicted by each classifier.

4.3.2 ORPTE algorithm

The proposed ORPTE algorithm consists of the following steps:

- Project the training $\mathcal{L} = (\mathbf{X}, Y)$ into $\mathcal{T}$ number of random projections to reduce the dimension with $q' < q$ features i.e. $A^r_{q' \times n} = R^r_{q' \times q} \mathbf{X'}^r_{q \times n}, r = 1, 2, 3, \ldots, \mathcal{T}$.

- Grow classification tree on a bootstrap sample from each of the $A^r_{q' \times n}, r = 1, 2, 3, \ldots, \mathcal{T}$, using the Random Forest algorithm, i.e. select randomly $q_{mtry} < q'$ projected feature while splitting the nodes in each tree.

- Estimate out-of-bag (OOB) error rate of the trees.

- Arrange the trees in ascending order with respect to their OOB error rates.

- A proportion of trees are selected with the smallest error rates to develop a new forest ensemble.

- For the prediction of a new observation, filter it through trees in the new forest and estimate its class label by using majority voting rule.

A flow chart illustrating the proposed ORPTE is given in Figure 4.1. The pseudo code of the Optimal Random Projection Trees Ensemble algorithm is given in *Algorithm* 6.

Algorithm 6 Pseudo code of ORPTE

Generate $\mathcal{T}$ projections and grow $\mathcal{T}$ trees

1: **for** $i = 1 \rightarrow \mathcal{T}$ **do**
- Generate a random projection from the training dataset $\mathcal{L} = (\mathbf{X}, Y)$.
- Select a bootstrap sample from the projected data.
- Construct tree on the bootstrap sample.
- Estimate OOB error rate for the tree.
- Store the results.

2: **end for**

Select the best trees for the final ensemble

3: Rank the trees with respect to their out of bag error rate in ascending order.
4: Select the highest ranked p% of the trees.
5: Combine them to construct ORPTE.

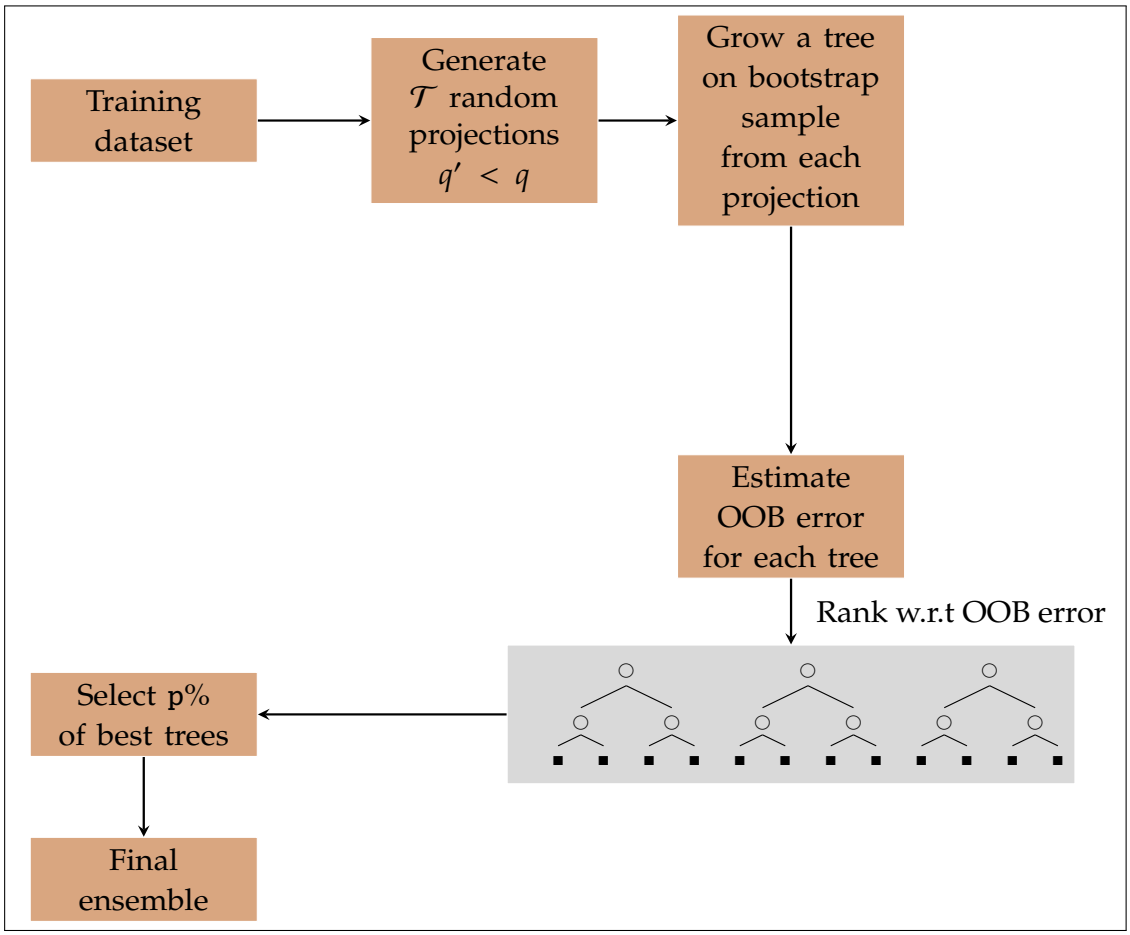


Figure 4.1: *Optimal Random Projection Trees Ensemble (ORPTE) flow chart for classification.*

4.4 Benchmark datasets

To assess the performance of proposed method in comparison with other machine learning methods, we have used 33 benchmark datasets from publicly available sources like Machine Learning Repository (UCI) [59], KEEL data bases `http://sci2s.ugr.es/keel/category.php?cat=clas#sub2` openml data bases `https://www.openml.org` etc. These datasets used are diverse in nature in terms of their types, features and observations. Details for the benchmark datasets are given in Appendix A.

4.4.1 Experimental setup and results for benchmark datasets

Experiments are carried out on the 33 benchmark datasets. Each dataset is split into 90% training and 10% testing parts. A total of 1000 random projections are generated from the training part and $\mathcal{T}$ independent classification trees are grown. The testing part (10%) is used for external validation check. While growing the trees, the number of features in the projected space is kept constant at ($q' = 10$) for benchmark datasets. For the microarray datasets the value of q' is fixed at $q' = 50$. The number of features randomly selected to decide a split on the nodes is fixed at $q_{mtry} = \sqrt{q'}$. The proportion ($\mathbf{p}$) of the highest ranked trees based on OOB error is fixed at 20% of the total $\mathcal{T}$=1000 trees initially grown. Testing part is used to assess the method in terms of the classification error rate. The above process is repeated 100 times for each dataset and the average error rate over all the 100 runs is reported for each dataset. For a fair comparison, the same training and testing parts have been used over all the runs for all the other methods considered in this chapter.

The hyper-parameters of the rest of the methods have been fine-tuned so, as to select their best possible combination. We have used the function "tune.rpart" for tuning various hyper-parameters of classification tree, using the R package "e1071" available in [70]. For fixing the minimal number of splits and the maximal depth of the tree, values $(5, 10, 15, 20, 25, 30)$ are tried. The number of nearest neighbours (k) in k-NN, is tuned by applying the R function "tune.knn" from the R library "e1071" [70] by using several values of k i.e., (3, 5, 7, 9). In Random Forest, various hyper-parameters such as number of feature for splitting, node size and number of trees are tuned by utilizing the function "tune.randomForest" available in the R package "e1071" [70]. We have tuned "number of feature for splitting" by considering various values like ($\sqrt{(q)}$, $q/2$, $q/3$, $q/4$, $q/5$). Similarly,

for tuning the node size we have taken a number of values (1, 5, 10, 15, 20, 25, 30) and for "**ntree**" we have tried values (500, 1000, 1500). In case of microarray datasets, we have considered the default values of the hyper-parameters of Random Forest while fixing **ntree** = 1000 due to computational reasons. We have implemented Optimal Trees Ensemble using the default setting while fixing **t.initial**=1000 in the R package "OTE" [61]. In case of Random Projection, the results of RP_{QDA_5} are calculated for $q' = 5$ using "Haar" as the projection method [63] and $\mathbf{b_1} = 50$ and $\mathbf{b_2} = 20$. For the implementation of SVM, the R package "kernlab" is used [71]. While Extreme Gradient Boosting (XGBoost) is implemented using R package "xgboost" [46]. The classification error rates on 25 datasets are given in Tables 4.1 and the results on remaining datasets are given in Appendix B. The best results are highlighted in bold.

Table 4.1 *Classification error rates for k-NN, SVM, RP (with QDA), Tree, RF, OTE and ORPTE.*

Data Set	*k*NN	SVM (Radial)	SVM (Linear)	RP (QDA)	Tree	RF	OTE	XGBoost	ORPTE
Vowel	0.0005	0.0403	0.0587	0.0078	0.0166	0.0051	0.0056	0.0288	**0**
Wind	0.1579	0.1618	0.1452	0.1475	0.1690	**0.1336**	0.1371	0.1690	0.1446
Eyestate	**0.0241**	0.3211	0.4303	0.4805	0.2977	0.0628	0.0646	0.3326	0.0505
Thoracic	0.1578	0.3834	0.3321	0.1738	0.1648	0.1565	0.1578	0.1559	**0.1462**
Ring	0.2860	0.0224	0.2321	0.2384	0.1498	0.0479	0.0493	0.2350	**0.0215**
Two norms	0.0272	0.0211	**0.0208**	0.0229	0.2342	0.0265	0.0284	0.2415	0.0230
Parkinson	0.0410	0.1084	0.1252	0.1610	0.0636	0.0380	0.0368	0.1005	**0.0205**
Body	0.0329	0.0127	**0.0121**	0.0156	0.0790	0.0394	0.0380	0.0901	0.0236
Collins	0.0018	0.0216	0.0664	0.0090	0.0034	**0**	0.0016	0.0032	**0**
White-clover	0.2883	0.2783	0.3033	0.3266	0.3216	0.2350	**0.1966**	0.3900	0.2272
Backache	0.1561	0.2338	0.3033	0.2027	0.1916	0.1505	0.1594	0.1588	**0.1389**
Wisconsin	0.4331	0.4273	0.4421	0.4194	0.4505	0.4226	0.4426	**0.3852**	0.4260
Ionoshpere	0.1631	0.0520	0.1274	0.0582	0.1405	0.0680	0.0602	0.1108	**0.0409**
Sonar	0.1871	0.1685	0.2500	0.1790	0.2904	0.1814	0.1704	0.2614	**0.1434**
Coil	0.0606	0.3549	0.3144	0.0605	0.0607	**0.0597**	0.0664	0.0600	0.0659
Hill-valley	0.4680	0.5252	0.4514	0.4008	0.4872	0.4293	0.4800	0.5234	**0.2807**
USPS	**0.0133**	0.0287	0.0259	0.0183	0.0738	0.0207	0.0180	0.0753	0.0191
Isolet	0.0038	0.0035	**0.0023**	0.0063	0.0285	0.0175	0.0161	0.0218	0.0109
SRBCT	0.0560	0.0360	0.0080	0.0160	0.1080	**0.0060**	0.0200	0.1300	0.0134
Colon	0.2483	0.1866	0.1866	0.3383	0.2333	0.1850	0.1933	0.2850	**0.1829**
Breast	0.4050	0.3800	**0.3625**	0.4137	0.4212	0.3662	0.3750	0.4225	0.4063
Prostate	0.1480	0.0850	0.1040	0.1640	0.1410	0.0770	**0.0710**	0.1210	0.1346
Leukaemia	0.1428	0.0671	0.0142	0.1600	**0.0071**	0.0428	0.0228	0.0185	0.1250
Tumors C	0.3716	0.3216	0.3933	0.3616	0.4050	0.3966	0.4233	0.3600	**0.3067**
Lung	**0**	0.0193	0.0060	0.0194	0.0594	0.0066	0.0161	0.0588	0.0100

The results given in Tables 4.1 illustrate that our proposed method (ORPTE) is giving better performance than the other methods (k-NN, SVM Radial, SVM Linear, RP (QDA), Tree, XGBoost, RF, OTE) on majority of the datasets. ORPTE out performs on 11 datasets out of 25 datasets and gives better or comparable results to RF in some cases.

4.5 Simulation study

To assess our proposed method (ORPTE) simulation models are taken into consideration along with benchmark datasets. We present two simulation models to evaluate the predictive performance of ORPTE. The proposed simulation models in the study take into account of several variations to get insight into performance of ORPTE under different circumstances.

The proposed ensemble method could work for data with various kind of structures. Random projection only reduces the dimensionality of the original data. A randomly projected data retains the intrinsic nature of the data that leads to similar model interpretations as with data in their original shape. Moreover, tree selection discards the adverse effects of un-useful patterns in the data on the base model thus improving the final ensemble predictive performance. This section introduces generating data of various structures to see the efficiency of the proposed ensemble in varying situations. Two data simulating models are given as follows.

4.5.1 Simulation Model 1

For Model 1, a total of $q = 500$ variables is generated. Out of these variables, 200 variables are generated from normal distribution with mean zero and standard deviation one i.e., $N(0, 1)$. 200 variables are generated from uniform distribution. Similarly, 100 binary variables are also generated. As the proposed method deals with classification problems currently, the following logistic type model is used to estimate class membership probabilities.

$$p(y|\mathbf{X}) = \frac{exp\left(c_2 \times (\hat{y} - c_1)\right)}{1 + exp\left(c_2 \times (\hat{y} - c_1)\right)}, \tag{4.5}$$

where,

$$\hat{y} = \beta\mathbf{X}, \tag{4.6}$$

and c_1 and c_2 are arbitrary constants.

The β in the above predictor are the variable coefficients generated from uniform distribution over the interval $[0, 40]$.

4.5.2 Simulation Model 2

For Model 2, a total of $q = 500$ variables are generated. All these variables are generated from normal distribution with mean zero and standard deviation one i.e., $N(0, 1)$. As the proposed method deals with classification problems, the logistic type model given in equations 4.5 and 4.6, is used to estimate class membership probabilities. The β in equation 4.6 are the variable coefficients generated from uniform distribution over the interval $[0, 70]$.

We have fixed the value of c_1 and c_2 at 0.05 and 0.5, respectively, in all the models. In each model q=500 features are generated and n=100, n=250, n=700 observations, respectively.

Machine learning methods such as *k*-NN, SVM (Radial and Linear), RP (QDA), Tree, Extreme Gradient Boosting (XGBoost), RF and OTE are trained on 90% of training data and the remaining 10% testing data for validation. Each experiment is repeated 100 times and the final results are then averaged. We have taken *k*=3 in *k*-NN. While Tree algorithm is implemented using the default values of hyper parameters. For Random Projection, the results of RP_{QDA_5} are calculated for q'=5 using "Haar" as the projection method [63] and $\mathbf{b_1}$=50 and $\mathbf{b_2}$=20. A total of 1000 trees are grown in RF and OTE. Similarly, SVM (with Linear and Radial) is implemented using R package "kernlab" is used [71]. The results from all models are presented in Table 4.2. Unsurprisingly, the performance of *k*-NN and Tree algorithms is poor in most of the cases for the simulated models. RF, OTE and ORPTE is giving almost similar performance quite with small variations in some cases.

Table 4.2 *Classification error rates for k-NN, SVM, RP (with QDA), Tree, RF, OTE and ORPTE for model 1 and model 2 taking number of features q=500.*

Methods	Model 1			Model 2		
	n=100	n=250	n=700	n=100	n=250	n=700
k-NN	0.1690	0.2792	0.2350	0.4555	**0.4056**	0.4653
SVM (Radial)	0.1580	0.2016	0.5025	0.3977	0.4865	0.5371
SVM (Linear)	0.1570	0.2008	0.2060	**0.3955**	0.4460	**0.2621**
RP (QDA)	0.1545	0.2592	0.2474	0.4730	0.4545	0.4004
Tree	0.1650	0.3080	0.2860	0.5011	0.5069	0.5016
RF	0.1580	0.2053	**0.1994**	0.4422	0.4273	0.4037
OTE	0.1540	0.1984	**0.1994**	0.4633	0.4547	0.4340
XGBoost	0.1600	0.2744	0.2331	0.5122	0.5143	0.4931
ORPTE	**0.1410**	**0.1952**	0.2012	0.4320	0.4696	0.4653

4.6 Probability Machine

The term probability machines was introduces by Malley et al. [72] to estimate the probability of group membership in a binary class problem. In other words, probability machines is a learning algorithm for estimating group membership probabilities. Machine learning methods are known as the non-parametric or assumptions free methods, can be used alternative to the classical models based on underlying assumptions and to avoid the issue of misspecification. Machine learning methods aim to predict the class label of individuals also known classification problems. These classification techniques are widely used in various biomedical fields and can be seen in [73–75]. Machine learning methods could also be utilised for the task of probability estimation. Probability estimation for the group membership presents more information about the degree of association rather than only assigning a class for an individual.

In this section, we use our proposed method, the ORPTE, for class membership probability estimation. The results of ORPTE are compared with k-NN, SVM (with Radial, Linear), Random Projection with QDA, Tree, Extreme Gradient Boosting (XGBoost), Random Forest, and Optimal Trees Ensemble by calculating Brier Score on the same benchmark and simulated datasets.

4.6.1 ORPTE as Probability Machine

The proposed method ORPTE is a tree structured method based on trees grown by using Random Forest on random projections and selects a percent of trees from an ensemble. We estimate the conditional probability, of an individual belonging to a particular class given

features. The proposed method is given in the following steps.

- Draw random projections from the training set to reduce the dimension.
- Grow probability estimation tree on a bootstrap sample from each of the projection, using the Random Forest algorithm.
- Estimate out-of-bag (OOB) prediction error of the trees.
- Rank the trees with respect to their OOB prediction error.
- Select a percent of trees with the smallest error rates to develop a new forest ensemble.
- For estimating the class membership probability of a new observation, filter it through trees in the new forest and record the class membership probability given by the overall forest.

4.6.2 Performance Measure of Class Membership Probability Estimation

There are several techniques for assessing the performance of a prediction model. However, we use the Brier Score [31] which is a widely used measure for evaluating probability estimates in the literature. A desirable property of Brier Score is that it is strictly proper. Brier Score can be used as performance measure of the class membership probability estimation on test data. It is considered an appropriate assessment criterion where the outcome variable is binary, and the true probabilities are unknown. A small value of Brier Score indicates that the computed probabilities are captured exactly the same as the true probabilities that are not available. A machine learning method with the smallest Brier

Score have the best probability estimate among others. The class label of test observations can be either 1 or 0 in a binary class problem. The Brier Score can then be defined as:

$$\mathbf{BS} = E((Y - p(Y/\mathbf{X}))^2, \tag{4.7}$$

Y is the discrete response variable i.e., $Y \in \{0, 1\}$ and $p(Y/\mathbf{X})$ is the true unknown probability for the binary response given the features. The range of Brier Score is between and 0 and 1. An estimator of Brier Score can be written as:

$$\widehat{\mathbf{BS}} = \frac{\sum_{i=1}^{n_{test}} (y_i - p(y_i = 1/\mathbf{X}))^2}{total\ n_{test}}. \tag{4.8}$$

4.6.3 Experimental setup for Brier Score and results

We implement our proposed method ORPTE as a probability machine for estimation of class membership probability on the same benchmark datasets used for calculating the misclassification rate in table 4.1. Each of the datasets is divided into two parts i.e., testing and training. We consider 90% of the data as a training and the remaining 10% for validation. Each experiment is repeated 100 times and the proposed method is evaluated on testing observations in each run. The results of ORPTE are compared with k-NN, SVM (Radial and Linear), Tree, Extreme Gradient Boosting (XGBoost), Random Forest and Optimal Trees Ensemble using the experimental setup given in Section 4.4.1. We have not considered Random Projection Ensembles because class membership probabbility is not supported in package. The method with the smallest Brier Score is considered best. Table 4.3 presents Brier Score on 25 datasets and the remaining results are given in Appendix B.

Table 4.3 *Brier Scores of k-NN, SVM (Linear and Radial), Tree, RF, OTE and ORPTE. The best result is shown in bold.*

Data Set	*k*NN	SVM (Radial)	SVM (Linear)	Tree	RF	OTE	XGBoost	ORPTE
Vowel	**0.0007**	0.0020	0.0256	0.0162	0.0071	0.0062	0.0205	0.0037
Wind	0.1123	0.1118	0.1016	0.1338	**0.0980**	0.0984	0.1221	0.1038
Eyestate	**0.0218**	0.2090	0.2315	0.2050	0.0668	0.0646	0.2146	0.0505
Thoracic	0.1398	0.1293	0.1276	0.1337	0.1289	0.1282	0.1321	**0.1239**
Ring	0.2238	**0.0154**	0.1692	0.1253	0.0444	0.0439	0.1734	0.0442
Two norms	0.0234	**0.0164**	0.0165	0.1743	0.0471	0.0463	0.1622	0.0300
Parkinson	0.0312	0.0589	0.0864	0.0601	0.0450	0.0337	0.0690	**0.0245**
Body	0.0248	**0.0111**	0.0115	0.0739	0.0353	0.0287	0.0697	0.0198
Collins	0.0018	0.0033	0.0354	0.0034	0.0022	0.0018	0.0050	**0.0014**
White-clover	0.1951	0.1864	0.2127	0.2921	0.1703	0.1726	0.2470	**0.1649**
Backache	0.1348	0.1268	0.1253	0.1676	0.1140	0.1247	0.1315	**0.1129**
Wisconsin	0.2604	0.2458	0.2493	0.3050	0.2546	0.2585	**0.2447**	0.2651
Ionoshpere	0.1245	**0.0388**	0.1040	0.1021	0.0475	0.0467	0.0869	0.0549
Sonar	0.1458	**0.1158**	0.1723	0.2479	0.1247	0.1225	0.1774	0.1300
Coil	0.0585	0.0539	0.0537	0.0570	0.0556	0.0580	**0.0543**	0.0578
Hill-valley	0.3114	0.2517	0.2297	0.2961	0.2627	0.2712	0.2625	**0.1949**
USPS	**0.0107**	0.0167	0.0210	0.0653	0.0231	0.0203	0.0546	0.0330
Isolet	0.0061	0.0034	**0.0032**	0.0267	0.0128	0.0106	0.0177	0.0313
SRBCT	0.0674	0.0300	**0.0068**	0.1043	0.0607	0.0370	0.0695	0.0967
Colon	0.1529	0.1482	0.1543	0.2191	0.1506	0.1383	0.1533	**0.1366**
Breast	0.2394	0.2289	0.2376	0.3356	0.2234	**0.2158**	0.2958	0.2323
Prostate	0.1411	0.0905	**0.0781**	0.1042	0.1164	0.0927	0.1035	0.1385
Leukaemia	0.1115	0.0386	0.0162	**0.0071**	0.0562	0.0187	0.0204	0.0868
Tumors C	0.2690	0.2308	0.2422	0.3848	0.2278	0.2396	0.2782	**0.2222**
Lung	0.0059	0.0186	**0.0031**	0.0561	0.0198	0.0137	0.0415	0.0228

The best results on the datasets are highlighted in bold. It can be observed from Table 4.3 that ORPTE outperforms all the other methods and gives the smallest value of Brier Scores on 8 datasets. On one datasets RF gives the smallest Brier Scores. OTE and Tree provide the best result on one dataset. It is observed that SVM with Radial outperform on 5 datasets and SVM with Linear shows the best results on 4 datasets while k-NN presents the best result on 3 datasets. While XGBoost performs better on 2 datasets.

4.6.4 Brier Score on simulated data

The same simulation models defined in Section 4.5 are used for evaluating the performance of ORPTE in estimating class membership probabilities. The results are then compared with k-NN, SVM (with Radial, Linear), Random Projection with QDA, Tree, Random Forest, XGBoost and OTE. The results given below show that the performance of Tree, XGBoost and k-NN are consistently poor as compared to SVM (Linear and Radial), Random Forest, OTE and ORPTE. The ORPTE is showing better or comparable results to SVM, Random Forest and OTE on simulated datasets.

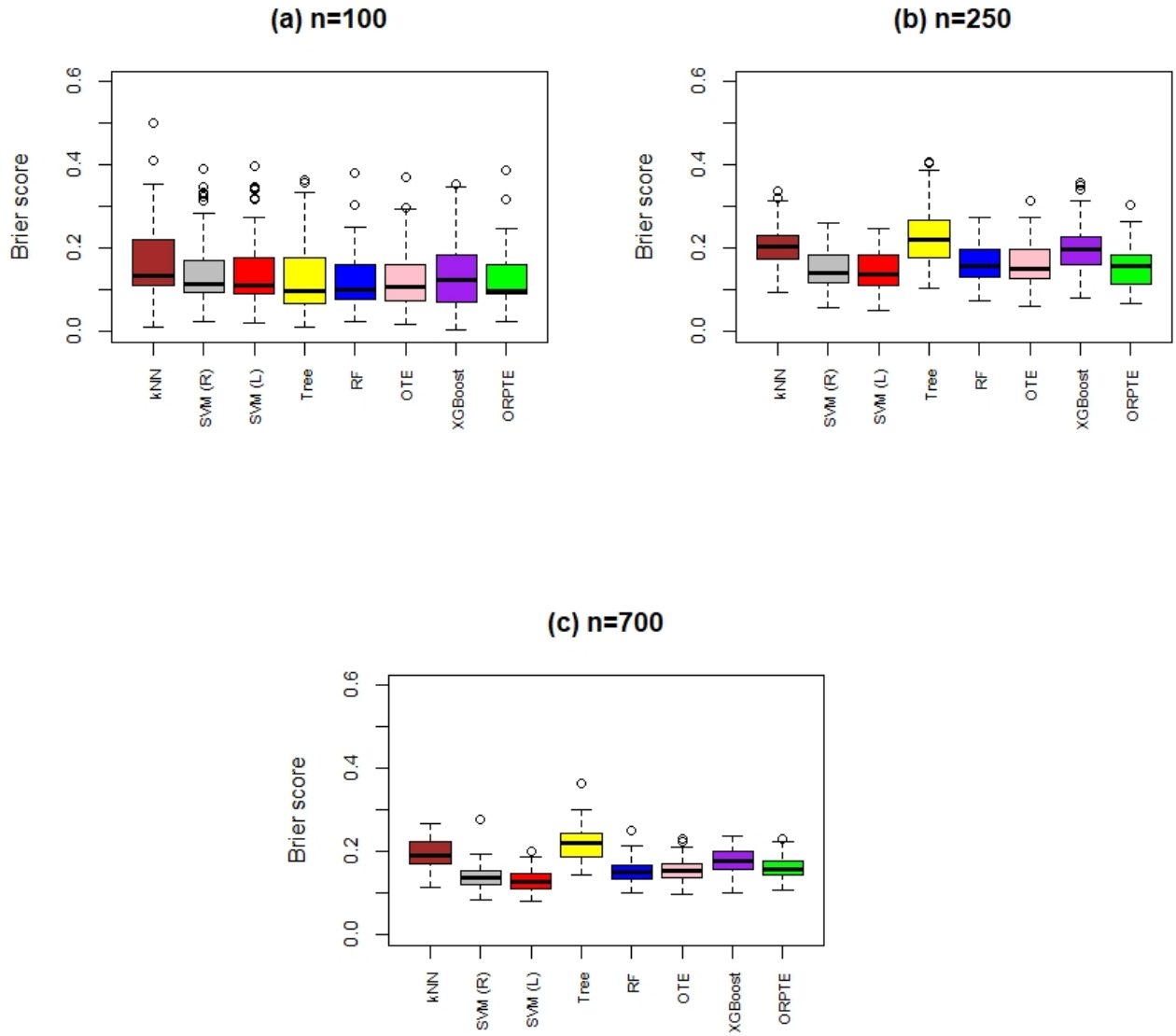


Figure 4.2: *Brier Score, of simulated data generated from model 1 in 100 runs for the seven classifiers k-NN, SVM (Radial and Linear), Tree, Random Forest, OTE, XGBoost, and ORPTE. ORPTE is performing better than k-NN, XGBoost and Tree while showing better/comparable results to SVM (Linear and Radial), Random Forest and OTE.*

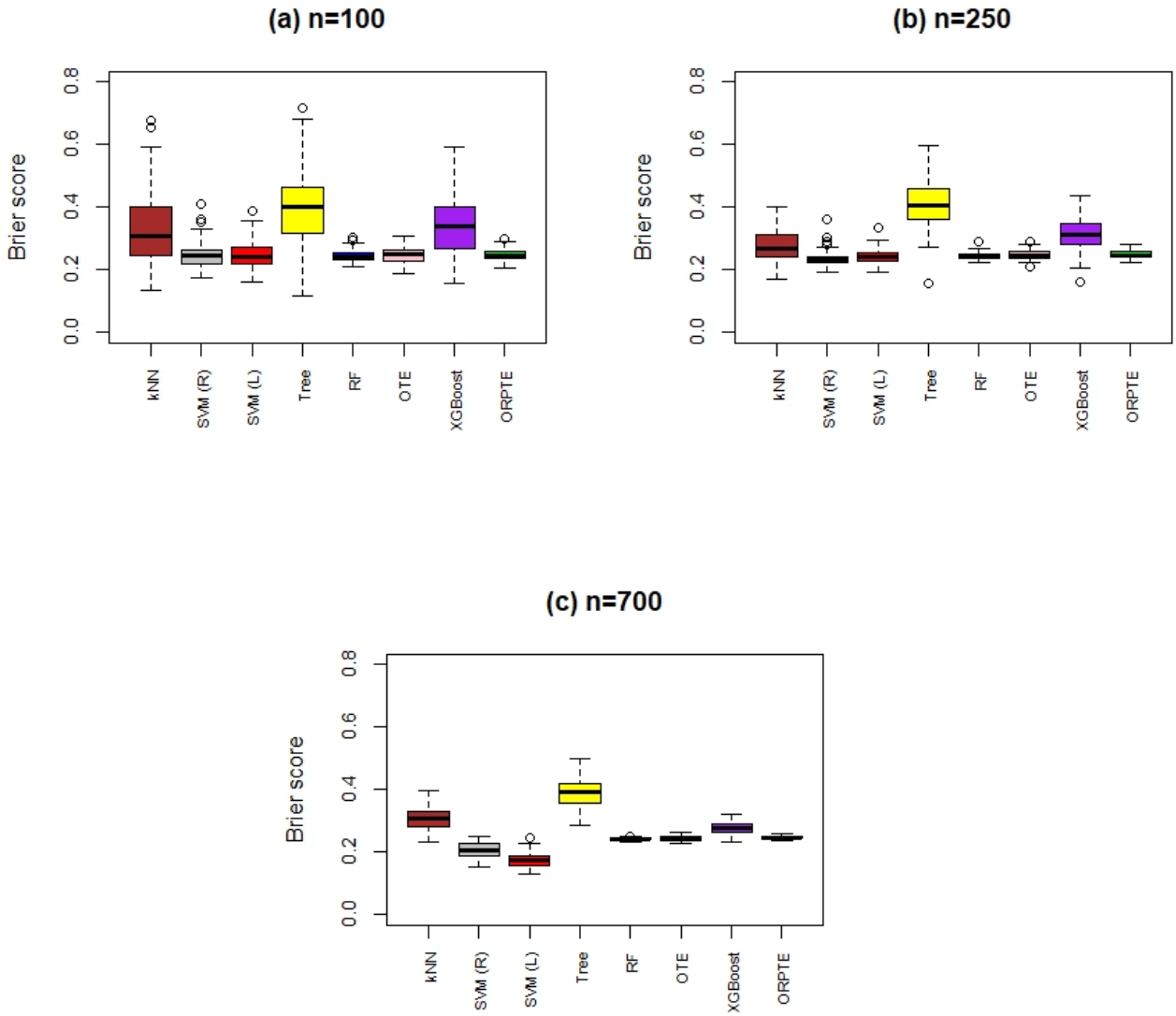


Figure 4.3: *Brier Score, of simulated data generated from model 2 in 100 runs for the seven classifiers k-NN, SVM (Radial and Linear), Tree, Random Forest, OTE, XGBoost and ORPTE. ORPTE is performing better than k-NN, XGBoost and Tree while showing better/comparable results to Random Forest and OTE.*

4.7 Discussion

We have empirically evaluated the predictive performance of our proposed method on a number of benchmark datasets. The results given in Tables 4.1 illustrate that our proposed method (ORPTE) is giving better performance than the other methods (*k*-NN, SVM Radial, SVM Linear, RP (QDA), Tree, RF, OTE, XGBoost) on most of the datasets. ORPTE out performs on 11 datasets out of 25 datasets and comparable results to RF in some cases i.e., Collins, USPS, White-clover, Two norms, for example. *k*-NN, gives better performance on three datasets. SVM (Linear) produces better result on four datasets. XGBoost is performing well on only one dataset. RF is performing best on four datasets while OTE is giving better result on two dataset. In the case microarray datasets, where the number of features is relatively large, the proposed method ORPTE provides improved prediction accuracy than the other methods on 2 datasets. On 2 datasets (Prostate and Leukaemia) it gives better results than *k*-NN and RP (QDA). ORPTE gives comparable result on Lung and SRBCT datasets as compared to *k*-NN, SVM (Linear), RF and better results than RP (QDA), Tree, SVM (Radial) and OTE.

The proposed method has used two simulation models to see when does the proposed method perform better. Simulation model 1 uses features from different distributions i.e. Normal, Uniform and Binomial. The coefficients of these variables are generated in a sparse manner over the range [0,40] which imply that some variables are more important from the others. ORPTE can capture meaningful structures in such sparse datasets with the help of random projection and tree selection. On the other hand, when features are generated from a single distribution (normal distribution given in the thesis), random

projection (along with bootstrapping) and tree selection in the proposed ORPTE does not bring improvement as compared to the other method. This, nevertheless, is not restricted to the above mentioned situations. As given in the analysis on a number of benchmark datasets, ORPTE can effectively learn in a variety of situations.

We have also used our proposed method for estimating group membership probabilities in a binary class problem and compared the results with k-NN, SVM (Radial and Linear kernels), RP (QDA), Tree, RF, XGBoost and OTE on the same benchmark datasets. ORPTE is showing better performance with respect to the smallest Brier Scores on 8 out of 25 datasets as compared to other methods. For microarray datasets ORPTE gives smallest Brier Score for 2 datasets. We have estimated the class membership probability on the simulated datasets and computed the Brier Score. The new method ORPTE is performing better than k-NN, XGBoost and Tree while showing better or comparable results to SVM (with Linear, Radial), RF and OTE on simulated datasets.

Chapter 5

Assessing hyper-parameters and feature selection for ORPTE

5.1 Introduction

In Chapter 5, we have proposed Optimal Random Projection Trees Ensemble (ORPTE) and compared prediction accuracy with some other machine learning methods. The prediction accuracy also depends on the choice of best possible combinations of hyper-parameters. There are several hyper-parameters regulating ORPTE which need to be tuned prior to the implementation of ORPTE algorithm. In addition, we also explore the influence of feature selection on newly proposed method.

This chapter focuses on assessing hyper-parameters and feature selection for ORPTE. To this end, we have considered LASSO as a feature selection method. The main aim is to investigate the influence of hyper-parameters and feature selection on ORPTE in comparison with Random Projection (with QDA, LDA), Random Forest and Optimal Trees

Ensemble in particular.

5.2 Hyper-parameters assessment

For assessing the effect of various hyper-parameters regulating ORPTE, we have tuned various hyper-parameters such as, initial size of the ensemble (**initial**.**size**), number of features in the lower dimension (q') and the percent of best trees (**p**) selected from the initial ensemble on the basis of their performance on OOB observations. Each dataset is divided into training part consists of 90% while the testing/validation part contains of 10% of the data. For tuning the q' we have taken the values (3, 5, 7, 10, 12), for tuning **initial**.**size**, we have chosen the values (100, 500, 1000, 1500, 2000, 2500, 3000) and for tuning **p**, we have tried (5%, 10%, 20%, 30%, 40%, 50%, 60%). Each experiment is repeated 100 times and average all the runs for the final result.

Figure 5.1 shows the effect of an important hyper-parameter (**p**) of ORPTE, is the proportion of best trees selected on the basis of their performance (yielding the smallest error rate on OOB) to develop the final ensemble. Various values of **p** demonstrate different behaviour of the method for classification purpose. It is clear from plot that **p** can be fine-tuned depends on the nature of the data. Our plot also shows that after achieving the gain in predictive accuracy increasing the percent of best trees can negatively affect the ensemble e.g., this pattern can be seen for Hillvalley, Parkinson, Body, and Wisconsin datasets. However, in the case of datasets with larger number of features Collins, Coil, USPS for example, increasing the value of **p** has no or very slight effect on the predictive accuracy of the ORPTE.

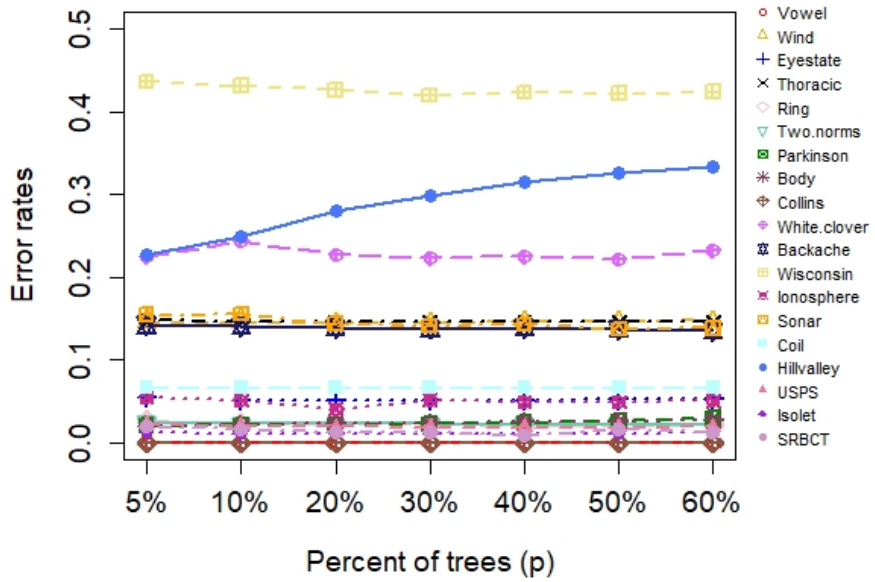


Figure 5.1: *The effect of percent of best trees and misclassification error for the datasets using* ORPTE. *The figure shows fluctuations therefore,* **p** *can be fine-tuned.*

The plot shown in Figure 5.2 displays the effect of the lower dimension of the image space of the projections (q') and classification error rate for classification. The plot shows decreasing pattern as the value of q' increases for most of the datasets. Since on the basis of the results, we recommend $q' = 10$, because it gives better result on 11 datasets out of 19 datasets such as Wind, Thoracic, Parkinson, Collins, Coil etc.

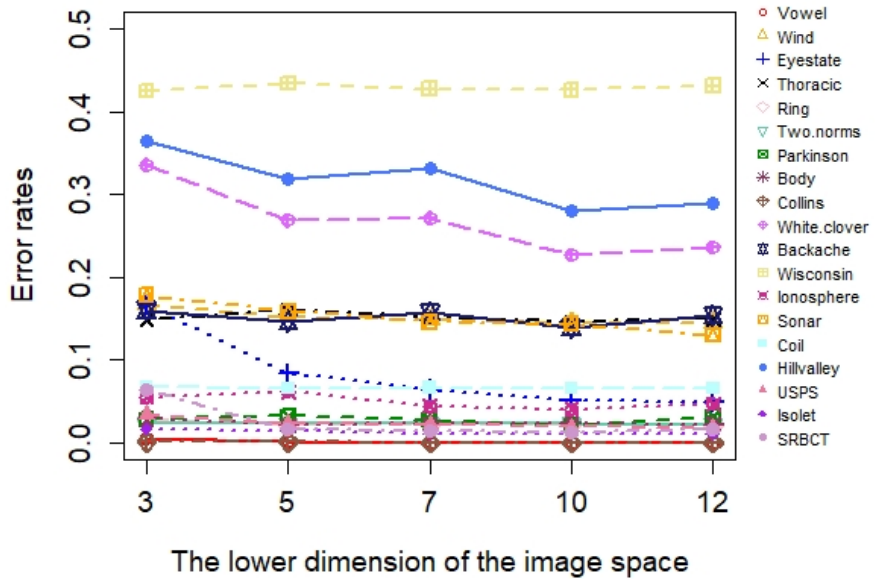


Figure 5.2: *The effect of lower dimension (q') and misclassification rate on datasets. The figure shows that q'=10 is appropriate.*

Figure 5.3 presents the misclassification rate for ORPTE for the various values of number of trees initially grown in the ensemble. An increase in the number of trees, minimizes the misclassification rate. However, it is notable that in some cases increase in number of trees slightly improves the results and there is not much gain in the predictive performance of the method. It is also observed from plot that for some of the datasets after gain in predictive accuracy increasing the number of trees can adversely affect the ensemble e.g., Wisconsin, SRBCT, and Thoracic datasets. Here, we recommend the number of trees initially grown between 1000 to 2000.

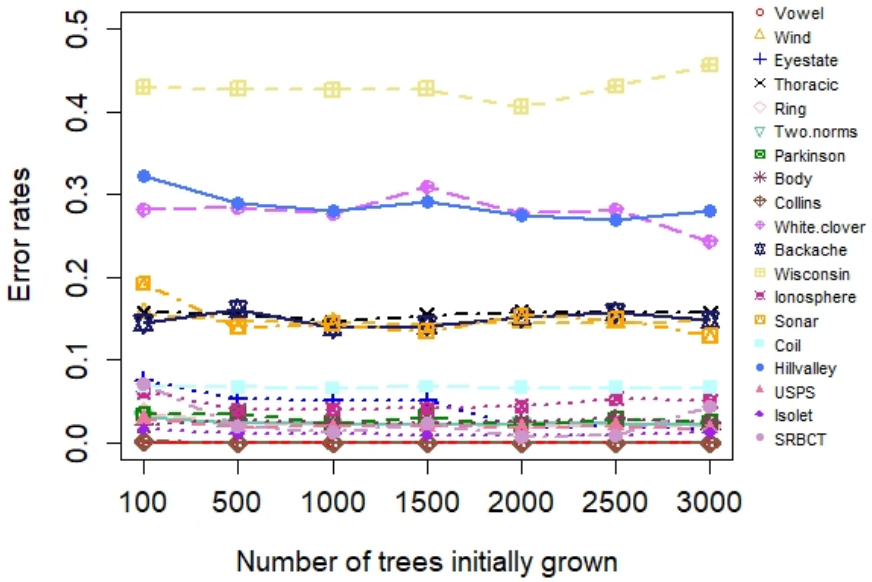


Figure 5.3: *The effect of number of trees initially grown and misclassification error rate on datasets. The figure shows that the number of trees between 1000 to 2000 is recommended.*

5.3 Feature selection using LASSO

Table 5.1 shows the classification error rate without feature selection and with feature selection using LASSO for RF, OTE, RP_{LDA_5}, RP_{QDA_5} and ORPTE. The experimental setup used for LASSO, RF, OTE, RP_{LDA_5} and RP_{QDA_5} is given in Chapter 3. The results of ORPTE are computed accordingly. Our benchmark study reveals that feature selection via LASSO improves the predictive performance of RP (with QDA, LDA) and RF while OTE and ORPTE are less influenced by feature selection. It can be seen that LASSO has minimized the classification error rate for only 7 out of 19 datasets for ORPTE. It is interesting to note that LASSO improves the result of ORPTE on microarray datasets such as Brest, Prostate, Leukaemia and Ovarian. Similarly, in the case of OTE, LASSO improves the results on only two datasets (Leukaemia, Adenocarcinoma).

Table 5.1 *Classification error rates for RF, OTE, RP_{LDA_5}, RP_{QDA_5} and ORPTE.*

	Without feature selection					Feature selection via LASSO				
Data sets	**RF**	**OTE**	**RP_{LDA_5}**	**RP_{QDA_5}**	**ORPTE**	**RF**	**OTE**	**RP_{LDA_5}**	**RP_{QDA_5}**	**ORPTE**
Eyestate	0.0616	0.0628	0.3684	0.4562	0.0505	0.3305	0.3439	0.3973	**0.4538**	0.1720
Hepatitis	0.1050	0.1225	0.1700	0.1537	0.5638	0.1235	0.1661	**0.1647**	0.1789	0.6188
Parkinson	0.0521	0.0336	0.1715	0.1468	0.0205	**0.0494**	0.0489	**0.1631**	**0.1357**	0.0290
Body	0.0384	0.0390	0.0150	0.0154	0.0236	**0.0339**	0.0494	**0.0145**	**0.0131**	**0.0202**
Thyroid	0.0098	0.0099	0.0482	0.0425	0.0340	**0.0094**	0.0113	**0.0481**	**0.0423**	0.0397
WDBC	0.0345	0.0356	0.0531	0.0557	0.0653	0.0398	0.0412	**0.0371**	**0.0396**	0.0653
WPBC	0.2052	0.2036	0.2210	0.2189	0.2465	**0.2036**	0.2189	**0.2194**	0.2300	**0.2385**
Ionosphere	0.0680	0.0800	0.0894	0.0528	0.0409	0.0700	0.0840	0.0922	**0.0505**	0.0614
Oil-spill	0.0368	0.0325	0.0429	0.0398	0.2273	**0.0338**	0.0335	**0.0422**	0.0404	0.2877
Spambase	0.0476	0.0473	0.1670	0.3051	0.1250	0.0488	0.0553	**0.1608**	**0.2807**	0.1552
Glaucoma	0.1100	0.0970	0.1020	0.1285	0.1920	**0.1015**	0.1235	0.1275	0.1495	0.2255
Nki 70	0.1535	0.1264	0.1814	0.1885	0.1360	0.1664	0.1842	**0.1462**	**0.1607**	**0.1287**
Musk	0.1070	0.1022	0.2606	0.1635	0.1142	**0.1010**	0.1214	**0.2489**	0.1643	0.2086
Breast	0.3662	0.3750	0.3712	0.3837	0.4063	**0.3425**	0.3850	**0.3262**	0.3312	**0.3330**
Prostate	0.0770	0.0710	0.1160	0.1330	0.1346	0.0833	0.1233	**0.1044**	**0.1055**	**0.0846**
Leukaemia	0.0428	0.0228	0.0885	0.14	0.1250	**0.0071**	**0.0085**	**0.0471**	**0.0242**	**0.0338**
Adenocarcinoma	0.1400	0.1362	0.1525	0.1162	0.1413	**0.0800**	**0.1062**	**0.0787**	**0.1037**	0.2295
Lung	0.0066	0.0161	0.0155	0.0161	0.0100	0.0140	0.0300	**0.0088**	**0.0077**	0.0100
Ovarian	0.0168	0.0072	0.0212	0.0180	0.0316	**0**	0.0160	**0**	**0.0012**	**0**

Note: Taking ensemble of size 1000 for RF, `OTE`, ORPTE and $\mathbf{b_1}$=500, $\mathbf{b_2}$=50 for RP. The minimum value of λ has been chosen via cross validation in LASSO. The improved results are highlighted in bold in the table.

5.4 Discussion

The main hyper-parameters of ORPTE are the number of trees initially grown (**initial.size**), the proportion of best trees (**p**) producing the smallest error rate on out-of-bag observations to form the final ensemble and the third hyper-parameter q' that is the lower dimension in the feature space. We have tried various values for these hyper-parameters in the assessment phase. Different values of (**p**) show different structures as can be seen in Figure 5.1. It is obvious from the results that the highest accuracy can be achieved by using a portion, 10% - %30, of the total classification trees grown on each projection that are accurate and diverse. Choosing a portion of best and accurate trees may lead us to an ensemble which is not very complex because only those trees are added to the final ensemble that contribute to the final ensemble. If the size of an ensemble is very large and there are some weak trees then, it might affect the prediction accuracy of the model in general.

The other hyper-parameter of ORPTE i.e., **initial.size** is also assessed using various values shown in Figure 5.3 and various behaviours are observed. The benchmark study reveals that as **initial.size** increases, the misclassification rate decreases. On the other side, it also is observed that in some cases increasing **initial.size** slightly improves the results up to some extent and there is not much gain in the predictive performance of the method. It is also revealed from plot that for some of the datasets after gain in predictive accuracy increasing **initial.size** can adversely affect the ensemble when **initial.size** is sufficiently large as discussed in Chapter 3. Similarly, from the perspective of computational complexity ORPTE may become computationally expensive when **initial.size** becomes very large.

Our benchmark experimental study also shows that after gaining the required accuracy, adding further trees unnecessarily can adversely affect the ensemble. This can also make the model more complex. Based on the results, we recommend the number of trees initially grown between 1000 to 2000.

Furthermore, the projection dimension q' is another factor in determining the lower dimension and the computational cost in ORPTE. The plot shown in Figure 5.2 displays that this parameter depends on the number of features need to be projected. The resulting plot shows decreasing pattern as the value of q' increases for most of the datasets. Since, we recommend $q' = 10$, because it gives better result on 11 datasets out of 19 datasets.

The results in Table 5.1 demonstrate that feature selection improves predictive performance of RP (with QDA, LDA) and RF on benchmark datasets in addition to reducing the computational cost already discussed in detail in Chapter 3. The benchmark study shows that OTE and ORPTE is less influenced by feature selection via LASSO as compared to the other methods. Our newly proposed methods is implemented using the idea of OTE which grows trees on the projected data. The trees are grown in Random Forest style which implicitly does variable selection at each node. A random subset of features is selected for splitting the nodes. This makes the model more robust as compared to Random Projection method. Thus, ORPTE is less influenced by feature selection via LASSO in the case of small datasets and significantly improves results in the case of high dimensional datasets. It gives an indication that the trees are more decorrelated in OTE and ORPTE. In addition, limiting the number of trees in an ensemble by discarding redundant or weak trees with high misclassification rates also increase prediction accuracy of the model.

Chapter 6

Data processing and analysis pipeline for next generation sequencing

6.1 Introduction

The technological advancement in the area of Molecular Biology specifically next generation sequencing analysis has brought a revolution and new discoveries in fundamentals of Biology. Quantitative assessment of differentially expressed genes can give clues about the structure of gene formation and interconnection about gene function and interactions. Machine learning methods have proved to be enough capable for molecular biological data because of its learning algorithm potential for building classifiers which can interpret the relationships in a better way. Analysing next generation sequencing data using ensemble methods has provided a better understanding for the effective diagnosis of different diseases [76].

Gene expression profiling by microarray technique or next generation sequencing allow the investigation of thousands of differentially expressed genes in parallel. It helps to generate a comprehensive sketch of cellular function. Basically, gene expression profiling aims to differentiate between cells which are dividing actively and to demonstrate cells

behaviour in response to a particular treatment. A whole genome is then studied simultaneously in this way to investigate each gene existing in a specific cell. There are many transcriptomics technologies designed to produce relevant data to be analysed.

Usually, statistical methods are applied to smaller datasets however, it may increase the probability of error when implemented on large datasets because there are a number of noisy features in the data. Hence, it is needed to remove noisy features from the data and to identify informative features [77].

The aim is to select differentially expressed genes and then apply ensemble models such as Random Forest, Optimal Trees Ensemble, Random Projection Ensembles and Optimal Random Projection Trees Ensemble to investigate it's implementation under high dimensional settings. As a motivational example, this thesis investigates the pre-processed next generation sequencing gene expression dataset i.e., colon cancer, stage II vs stage III only (non-invasive vs invasive). The analysis done in the chapter also flag out genomic markers that are responsible for regulating the outcome and better explain the relationship.

6.2 Colon Cancer

This cancer starts in the colon which is the large part of the bowel. Mostly colon cancers start as a growth, called a polyp on the internal lining of the colon. Many types of the polyps can convert into cancer during several years. It is, however, not necessary that all polyps turn into cancer. When malignant tumors are formed, the cancerous cells may pass on through the blood and lymph systems, spreading to other body parts as well. Lymphatic metastasis is considered as an important predictor for the recurrence of tumor and survival

in colon cancers [78, 79]. The primary tumor is removed during surgery including all draining lymph nodes. Adjuvant chemotherapy is done for stage III colon cancer to reduce the chances of recurrence of tumor and survival improvement in general [80]. Different stages of colon cancer is shown in Figure 6.1:

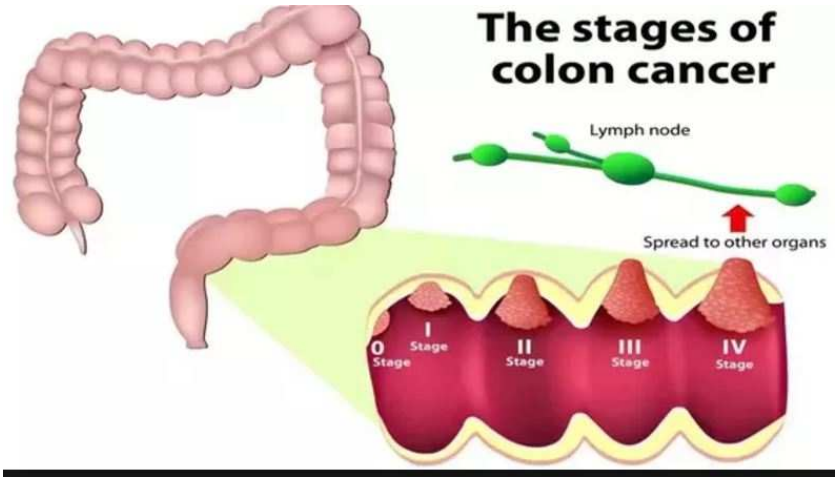


Figure 6.1: *Colon cancer [81]*

6.2.1 Types of Colon Cancer

Adenocarcinomas

Adenocarcinomas are the cancerous tumors, begin in secretory cells that releases substances called "mucus" to lubricate the internal of the colon and rectum. Initially, tumors begin to grow as a adenomatous polyps and ultimately become malignant tumors.

Gastrointestinal stromal tumors (GISTs)

The colon wall consists of some specific cells called interstitial cells of Cajal, in which Gastrointestinal stromal tumors begin to start. Some of these tumours are non-cancerous

and some are. Usually GISTs are found in the digestive track.

Carcinoid tumors

In our digestive system, there are tissues/cells which produce harmones. In these tissues a neuroendocrine tumour slowly grows and causes Carcinoid cancer.

Lymphomas

This type of cancers basically gets started in lymph nodes but it may also start in the rectum or other parts as well. Lymphomas are cancers of the lymphatic system/immune system cells.

Sarcomas

Malignant tumors of the connective tissues in the inner wall of the colon and rectum are called “sarcomas”. Sarcomas are cancers found in connective tissue like fat, blood vessels, nerves, bones, muscles and cartilage.

6.2.2 Invasive and non-invasive cancer

Non-invasive cancer is limited to the lobules and have not open out to the adjacent tissues and body other parts. Non-invasive cancers may turn into invasive tumors if not diagnosed or treated in time. As non-invasive tumours are confined to ducts thus, it does not include basement membrane break. Therefore, it has no connection with the mucous membrane and inner body cavity.

In contrast, invasive cancer cells escape out of the ducts where cancerous cells begin

started and probably can extend to the lymph nodes and other body parts. In this case, when colon cancer extends further side of the inner lining of the colon, is then called invasive adenocarcinoma. Thus, surgery is recommended for invasive cancers when it breaks out of the skin. Though open surgery is carried out for invasive cancers but invasive procedures may also consider perforation, incision, etc. There are four stages of colon cancer stageI, stageII, stage III and stage IV. The stage II colon cancer is referred to as non-invasive cancer and stage III as an invasive colon cancer.

6.3 Noisy features in the data

In classification tasks, where the number of features exceeds the number of observations has become the area of research itself. High dimensional data with small sample size cause problems for learning algorithms and is a challenge to classification methods because it increases the chances of over-fitting and affects the prediction accuracy of classifiers. The number of non-informative feature in the datasets usually increases with the increase in the number of features i.e., high-dimensional datasets generally have a higher number of redundant and non-informative features. Identifying the most relevant features not only overcome the curse of dimensionality but also reduce the computational complexities [82]. As discussed earlier, the next generation sequencing technologies produce data in parallel and the number of features is in thousands and millions even for relatively over few hundreds of patients. In these situations, the features which are more relevant to the outcome variable are very few in numbers. Therefore, it is essential either to remove these noisy features before training the classifiers on the dataset or to build a certain

mechanism that could handle the noisy features. The issue of non-informative features is commonly dealt by dimensionality reduction techniques such as feature extraction and feature selection techniques. Feature selection approaches aim to select the features/genes which carry the most of discriminative information about the classes in the data for better understanding [83] and have been well investigated in several studies [21, 84–86]. In general, feature selection can improve the predictive performance of classifiers by removing the irrelevant features from the training dataset [87, 88]. Redundant and non-informative features do not contribute to response regulation and, in most of the cases, their effect is adverse on classification algorithms. For some classification algorithms, that might be robust to such features, these features still do not have a positive impact on classification performance

6.4 Gene selection methods

There are three general methods of feature selection algorithms: filter method, wrapper method and embedded method. These methods can be used in order to identify and discard irrelevant and redundant variables/genes from data that do not contribute to the accuracy of a predictive model or sometimes may decrease the accuracy of the model.

6.4.1 Filter method

This method evaluates features/genes based on their relevant score in various statistical tests for their correlation with the outcome variable. Features with a low score are then rejected and the selected features with high relevant score may then be used to serve

classification through many types of classifiers. Filter-based methods for genes selection are very simple and fast that's why these methods can be implemented in high-dimensional settings. It should be kept in mind that filter methods do not deal with multicollinearity. Therefore, one must deal with the problem of multicollinearity of features prior to training of models [89].

The filter methods can further be divided into two sub-groups that are univariate and multivariate. For classification tasks, univariate techniques are deployed on statistical measures which individually take in to account each gene to find differences between two classes (Wilcoxon rank-sum, for example) or among three or more classes (Kruskal-Wallis test, for instance). Multivariate techniques, however, include gene dependencies within the feature selection process. Examples of multivariate techniques are Minimum Redundancy Maximum Relevance [90] and Fast Correlation Based Feature Selection [91].

6.4.2 Wrapper method

Wrapper methods divide the dataset into two parts i.e. training and testing sets, and assess subsets of features by using a predictive model. Thus, each features subset is utilized with a training dataset to train the model, which is then tested for the performance on the test set and a model prediction error is calculated, providing a score for that feature subset. Finally the feature subset with the highest score is chosen as the final set and the particular model is run over the chosen set. The wrapper methods are computationally costly as they require a new model to be fitted for each subset of features. A few common examples of wrapper methods are forward selection, backward selection, and recursive feature elimination. Genetic algorithm based feature selection techniques are related examples for wrapper

methods [89, 92].

6.4.3 Embedded method

Embedded methods are the combination of Filter and Wrapper methods in the sense that it combines the characteristics of these two methods. It makes use of algorithms which have own already built in feature selection methods. In other words, these methods perform feature selection as part of the model building process. They are computationally less costly as compared to Wrapper methods [89, 92]. Examples of this category is a classification tree [16], ridge regression, Least Absolute Shrinkage and Selection Operator' (LASSO) method [65].

6.5 Related work

The implementation of feature selection methods in medical data has highly increased because such datasets consist of large feature space having a high dimension particularly produced by NGS technologies. Subha et al. [93] carried out a comparative study on colon cancer classification by using t-statistic as feature selection for Support Vector Machine (SVM), Neural Nets and Logistic Regression. The results revealed that SVM outperform on the top 10 genes ranked through t-statistic [93]. It has been investigated that variations in gene expression can be effectively used in the classification of colon cancer into two classes either normal or malignant. The SVM-ensemble approach on the selected subset of genes through F-Score, PCA and mRMR, performs better than single SVM model [94]. Colon cancer type can be identified by utilizing non-clinical approach where gene expression

profiling is used instead. For this purpose, Bayesian model averaging techniques of Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) is suggested by [95]. Jose et al. demonstrated clustering analysis for clustering the colon data and picking up a subset of informative genes in his study [96]. In [97] Yong et al. showed that the Sparse Logistic Regression with the $L_{1/2}$ penalty attains greater class prediction accuracy than other regularization approaches based on ordinary L_1 and Elastic Net and select smaller number of important genes in cancer classification.

6.6 Non-invasive vs invasive colon cancer next generation sequencing data

This dataset is taken from the Gene Expression Datasets Collection available on [98]. The dataset originally consists of information for 434 cancer patients. The dataset contains two files, first file comprises of 81 variables which provide the clinical information about the patients such as gender, patient-barcode, tumor-status etc while the second file consists of gene ID and 20530 normalized RNA-Seq gene counts or genes. For analysing the data these two files are merged in one file using the patients barcode. We use the binary endpoint (response variable) cancer staging: non-invasive (stage II) and invasive cancer (stage III). Moreover, we aim to use pre-processed/engineered next generation sequencing features as predictors. The data is converted to a binary class problem by considering stage II patients against stage III and discarding the rest of the observations. This led to a data frame with 220 observations on 20530 features. More information about the dataset is available at `firebrowse.org`.

6.7 Methodology

The machine learning methods for next generation sequencing data classification require mining of the datasets in order to increase the chances of getting effective results. The identification of important features is one of the most essential steps in building or training of machine learning classifiers. In high dimensional settings, feature selection can reduce the noise and biases by picking up only those predictors which may increase accuracy of classification process. This chapter focuses on feature selection methods like Least Absolute Shrinkage and Selection Operator (LASSO) [65], Wilcoxon Two Sample Rank Test, and Proportional Overlapping Scores (POS) [99] for the ensemble methods such as: Optimal Tree Ensemble [30], Random Projection Ensembles [42] and Optimal Random Projection Trees Ensemble in comparison with Random Forest [28] in high dimensional settings discussed in Chapter 2. For the implementation of our methodological framework we have taken pre-processed next generation sequencing data on colon cancer as an example dataset. We represent subset of biological genes to be used for subsequent classification analysis of the pre-processed next generation sequencing gene expression colon cancer dataset, stage II and stage III only i.e., non-invasive vs invasive. These subsets are then used for training Random Forest, Optimal Trees Ensemble, Random Projection Ensembles and Optimal Random Projection Trees Ensemble classifiers for the classification purpose in a binary class problem. This chapter provides data processing and analysis pipeline for the next generation sequencing data and to see the influence of feature selection on Random Forest, Optimal Trees Ensemble, Random Projection Ensembles and Optimal Random Projection Trees Ensemble.

6.7.1 Experimental setup

Experiments performed on colon cancer dataset are designed as follows. The colon cancer dataset is split out into training part and testing part parts for training the algorithms. To train RF, OTE, ORPOTE a total number of 1000 classification trees are grown and calculate the classification error rate on the testing data. Similarly, RP classifier is trained on 90% training data and the testing part is used in the final ensemble to calculate classification error rate. Each experiment is repeated 100 times and averaging the all the runs for final results. For LASSO, we have fixed the value of lambda (λ) at 0.0001 and then selecting the most informative 100 features from the training data. In case of Wilcoxon Two Sample Test, the final subset of the top 100 ranked features are then selected according to predefined p-value or cut-off. Similarly, 100 discriminative gene are selected in Proportional Overlapping Scores approach (POS). Since the Proportional Overlapping Scores approach analyzes data based on the Inter Quartile Range (IQR), the proportional calculation for those genes for which the IQR equals to zero, should result in undefined values. Therefore, we exclude those genes from our feature space for the POS method because they are extremely non-informative in this context. Then we train RF, OTE, RP and ORPTE classifiers on the selected set of informative genes and calculate the misclassification rate on the testing data. The indices for selected genes are given in Table B.11 in Appendix B.

6.7.2 Softwares and packages

The R programming language is utilized [60] for carrying out the results and comparison of the methods in this study. We have implemented Optimal Trees Ensemble using

R package "OTE" [61]. While for Random Projection Ensembles, we have used R package "RPEnsemble" [63]. LASSO has been applied through the R package "glmnet" [64]. Proportional Overlapping Scores is applied by R package "propOverlap" [100]. Random Forest is implemented through the R package "randomForest" [62].

6.8 Results of classifying non-invasive vs invasive colon cancer using next generation sequencing data

The results given in Table 6.1 show misclassification rate for RF, OTE, ORPTE, RP_{LDA_5} and RP_{QDA_5} on example dataset using feature selection methods such as LASSO, Wilcoxon and POS. The results show that without feature selection RP_{LDA_5} gives better result than the other methods. In the case of feature selection, LASSO improves RP_{QDA_5} and ORPTE while Wilcoxon improves the results for ORPTE, RP_{LDA_5} and RP_{QDA_5} respectively. The best results are highlighted in bold.

Table 6.1 *Classification error rates for RF, OTE, ORPTE, RP_{LDA_5}, and RP_{QDA_5}.*

Methods	Without feature selection	Feature selection		
		LASSO	Wilcoxon	POS
RF	0.3645	0.3890	0.3654	0.4190
OTE	0.3804	0.3986	0.3854	0.4300
RP_{LDA_5}	0.3622	0.3836	**0.3500**	0.4390
RP_{QDA_5}	0.4095	**0.3845**	**0.3545**	0.4404
ORPTE	0.3932	**0.3796**	**0.3519**	0.4132

Note: Taking ensemble of size 1000 for RF, OTE, ORPTE and $\mathbf{b_1} = 500$ and $\mathbf{b_2} = 50$ for RP. The improved results are highlighted in the bold.

6.9 Discussion

The pre-processed next generation sequencing data related to colon cancer i.e., non-invasive vs invasive cancer using the four ensemble methods RF, OTE, RP (with LDA, QDA) and ORPTE has been analysed in this chapter. The misclassification rates are calculated without using feature selection and with feature selection shown in Table 6.1. For selecting the most informative features we have applied feature selection methods, LASSO, Wilcoxon Two Sample Test, and Proportional Overlapping Score (POS) on the training data. The four classifiers i.e., RF, OTE, RP (with QDA, LDA) and ORPTE have applied on the reduced datasets. For colon cancer dataset, feature selection via LASSO has improved the predictive performance of RP_{QDA} and ORPTE. However, the lower misclassification rates for both

RP_{LDA}, RP_{QDA} and ORPTE have been achieved by selecting the most regulatory features via Wilcoxon method shown in Table 6.1 and gives almost the same or better results for RF and OTE. POS does not improve the predictive performance of the training classifiers, instead, it has increased the misclassification errors for all four classifiers.

The results conclude that RF and OTE are comparatively more stable as compared to RP (with LDA and QDA) and ORPTE when there are noisy features in the data. One reason could be that trees are decorrelated in RF because for splitting the nodes a subset of random features are taken thus, RF is doing implicit feature selection. In the case OTE, an ensemble of optimal and diverse trees are formed where initial set of trees are grown on Random Forest style that is why OTE and RF behave alike. As claimed by [42], RP selects the best projections of the feature space thus doing an indirect feature selection. On the contrary, RP and ORPTE however, improve with feature selection suggesting its vulnerability to noisy features in the data. In addition, RP and ORPTE need to be optimized between computational cost and classification accuracy and/or need to be accompanied with the pre-processing step of feature selection. Whereas, RF and OTE might be used without pre-processing the data with little loss in classification performance in most of the cases as compared to RP. As shown in Chapter 3, Random Projection Ensembles depend on the number of blocks and block size as the two main parameters. However, increasing the values of these two parameters in high dimensional datasets incur high computational cost. Therefore, there exist a trade-off between classification performance and the cost. The cost on classifier level could be minimized by using feature selection method to reduce the high dimension of the data. Similarly, in the case of ORPTE, for high dimensional datasets, it has been observed that increasing the number of trees improves the predictive performance.

However, growing large number of trees in the forest involves huge computational time implying the existence of trade-off between classification performance and computational cost.

In a nutshell the analyses lead to practical directions of how to use these methods for classifying high dimensional datasets and how to use feature selection for increasing the predictive accuracy of the classifiers. The practical directions derived from this are that feature selection should be considered as a pre-processing step while classifying high dimensional gene expression data via RP and ORPTE methods. Moreover, if the resultant improvement in classification performance by considering large values for the number of blocks, block size and number of trees is not significant, there is no harm in losing a small amount of classification performance by saving a huge amount of computational resources.

Chapter 7

Conclusion and future directions

7.1 Conclusion

Ensemble refers to the combination of several weak learners to develop a strong model. Tree based structured method such as Random Forest, Optimal Trees Ensemble are examples of ensemble methods. Ensemble classifiers formed by the combination of multiple weak learners, have been shown to outperform ordinary classification methods and usually produce more accurate solutions than a single model. The ultimate aim of machine learning problem is to find a single model over several models that can better predict the outcome. Generally, classification and regression trees are considered good in terms of interpretability but the prediction accuracy is low. However, tree structured methods, Random Forest, for instance, are known for higher predictive performance but normally have the issue of less interpretability.

There are several techniques given in the literature to enhance the interpretability and to reduce the complexity of Random Forest. Accuracy and diversity of classification and

regression trees are crucial for the prediction accuracy in a forest. Several attempts have been made by reducing the number of trees in the forest using different approaches see for instance, [50, 54, 101]. These methods have showed that reducing the size of an ensemble by discarding the unnecessary trees without loss of any prediction accuracy can increase interpretability. Optimal Trees Ensemble for example, is modified version of Random Forest which selects a proportion of best trees based on their individual and collective performance to develop a new forest.

The aim of supervised classification is to achieve maximum prediction accuracy. However, high dimensional data often makes the implementation of machine learning methods difficult. The curse of dimensionality may effect the predictive ability of classification algorithms [102]. Under such circumstances, projecting the high dimensional data into a lower space is a useful technique. The idea of random projection for dimensionality reduction was initiated from the well known Johnson-Lindenstrauss Lemma [43]. The literature shows work carried out by researchers in high dimensional settings, for instance, a linear classifier can be trained a on a single projection [55]. However, several studies show that aggregation of several projections might be helpful [56–58]. It is also suggested to divide the random projections into blocks as sometimes aggregating classifications of all random projections may not be useful [42] and this technique is named as Random Projection Ensembles.

In the first part of the thesis, we have investigated three ensemble methods: Random Forest, Optimal Trees Ensemble and Random Projection Ensembles in particular. These classifiers, however, can still result in low prediction performance when used with the wrong choice of their hyper-parameters values and/or when there are noisy features in the

data. We have also investigated the implementation of these methods in high dimensional settings, for example next generation sequencing colon cancer data. Based on the analysis of 19 benchmark datasets and colon cancer data as an example dataset, two aspects of Random Forest, Optimal Trees Ensemble and Random Projection Ensembles have been assessed. The first aspect is the effect of hyper-parameters on the predictive performance of these methods. In the second aspect, feature selection has been considered to evaluate the performance of the three selected ensemble methods. For Random Forest and Optimal Trees Ensemble the hyper-parameters such as node size, number of trees initially grown, number of features and percent of best trees (main hyper-parameter of OTE) have been assessed and misclassification rates are calculated based on the training sets of all the benchmark datasets. In the case of Random Projection Ensembles, the hyper-parameters such as number of blocks and block size have been assessed. In the second part of the analysis the misclassification rates are calculated without using feature selection and with feature selection. For the benchmark datasets, feature selection via LASSO has improved the predictive performance of Random Forest, Random Projection with LDA and Random Projection with QDA on most of the datasets while Optimal Trees Ensemble is less influenced by feature selection. In case of example dataset (colon cancer), feature selection improves the predictive performance of Random Projection Ensembles when there are noisy features in the data. It suggests that Random Projection needs to be optimised between computational cost and classification accuracy and/or need to be accompanied with the pre-processing step of feature selection. On the other side Random Forest and Optimal Trees Ensemble might be used without pre-processing the data with negligible loss in classification performance in most of the cases as compared to Random Projection

Ensembles.

Inspired from the ideas of Random Forest, Optimal Trees Ensemble and Random Projection Ensembles, we have proposed an ensemble of classification trees that is build on projected data and has small ensemble size with better or comparable results to some other methods. Our method has considered using tree selection in conjunction with the idea of randomly projecting the feature space into a smaller dimension. The proposed method is named as Random Projection Optimal trees ensemble (ORPTE). ORPTE is an extension of Optimal Trees Ensemble. Our proposed method projects the training data into a number of random projections to reduce the dimension. Classification trees are grown on a bootstrap sample taken from each of the projection, using the Random Forest algorithm and selects randomly projected features while splitting the nodes in each tree. The trees are then ranked in ascending order of their error rates. We select a proportion of trees with the smallest error rates to develop a new forest ensemble. Accuracy in the final ensemble is kept by discarding trees that perform weakly, whereas diversity is added by random projection in addition to bootstrapping and using feature subsets at nodes splitting.

The ORPTE is applied on 33 datasets including some microarray datasets. The proposed ensemble method has performed better on most of the datasets than k-NN, Support Vector Machine (Linear and Radial), Random Projection (with QDA), Trees, Random Forest, Extreme Gradient Boosting and Optimal Trees Ensemble. The better performance of ORPTE may be due the fact that if the data is projected into a lower dimensional subspace and then additional randomness is incorporated by drawing bootstrap samples and a random subset of features is selected at each node then the ensemble gives better or at least comparable results as compared to the one consisting of weak learners. It could

also be due to the reason that random projections preserve the pair wise distance between the data points when projected into a lower dimension. Therefore, ORPTE algorithm can capture different structures present in the data which may not be captured by an ordinary algorithm. Similarly, our proposed method attempts to select a proportion of meaningful trees which produce smallest error rate on out-of-bag observations since the selected trees contribute when added to the final ensemble and ignoring those that only increase the error rate.

In classification problems, machine learning methods simply assign class label to an individual. However, sometimes in biomedical data it becomes essential to compute class membership probability in order to get more information. Probability estimation for the group membership presents more information about the degree of association rather than only identifying a class label for an individual. Machine learning algorithms can also be used for estimating class membership probabilities of individuals. We have computed probability estimation of group membership of individuals on the same benchmark datasets for ORPTE and others, used in classification. In this study, the performance of probability machines is measured by using the Brier Score criterion proposed by Brier in 1950 [31]. Our benchmark study has revealed that the new method has performed better than the other methods for estimating the class membership probabilities in terms of providing the smallest Brier Scores on most of the datasets. SVM (Radial) has provided the minimum Brier Score, on five datasets while SVM (Linear) has given best results on four datasets. Random Forest, Tree and Optimal Trees Ensemble have given better results on one dataset respectively. We have also computed the Brier Score for all the methods on the simulated datasets. The new method ORPTE is performing better than *k*-NN, XGBoost

and Tree while showing better/comparable results to Random Forest and Optimal Trees Ensemble.

We have also assessed the influence of the hyper-parameters and feature selection for ORPTE. Various hyper-parameters regulating ORPTE are tuned, such as, initial size of the ensemble (**initial.size**), number of features in the lower dimension (q') and the percent of best trees (**p**) selected from the initial ensemble on the basis of their performance on out-of-bag observations. We have used LASSO as feature selection method. Our benchmark study reveals that ORPTE is less influenced by feature selection in the case of small datasets and significantly improves results in the case of high dimensional datasets. It shows that if there is a large number of non-informative or redundant features in the data then there might be the possibility that many base models of an ensemble are based upon those features having high errors which can affect the predictive performance of the model.

The main contribution of this reseach work is the establishement of an ensemble method for classification and class membership probability estimation based on accurate and diverse trees grown on random projections of the given training data. An attempt is made to reduce the size of tree ensemble without loosing accuracy. The proposed methods are implemented in an R-package, "ORPTE". Secondly, this research work provides full assessment of the hyper-parameters of Optimal Trees Ensemble, Random Forest and Random Projection Ensembles and the influence of feature selection on these algorithms.

7.2 Future directions

There are some ideas which provide directions for future research work. These proposals are briefly discussed below:

- This work is designed for binary classification problems. However, it can further be extended to Regression Analysis and multi-class problems.

- Aggregating the classifications of all random projections may not always be reasonable as several of these projections will destroy class structure of the original data sometimes. Therefore, it is essential to preserve the original structure of the data. A modified version of ORPTE can be established to improve the predictive performance of ORPTE algorithm by dividing the random projections into non-overlapping or homogeneous groups/blocks.

- A framework for the proposed method can be constructed in which diversity and accuracy of the classification and regression trees can be ensured after selecting a percent of best trees from the initial ensemble using diversity measures as given in [2, 30, 103]. This could make the model more robust and might enhance the predictive performance of ORPTE algorithm.

- The idea of the proposed ORPTE method can be applied to Optimal Survival Trees (OSTE) [104] and extended to Optimal Random Projection Survival Trees Ensemble (ORPSTE) where survival trees can be grown on projected data and a set of optimal trees can be chosen for the final ensemble.

Bibliography

[1] R. Berk, "An introduction to ensemble methods for data analysis," *Sociological Methods & Research*, vol. 34, no. 3, pp. 263–295, 2006.

[2] Z. Khan, A. Gul, A. Perperoglou, M. Miftahuddin, O. Mahmoud, W. Adler, and B. Lausen, "Ensemble of optimal trees, random forest and random projection ensemble classification," *Advances in Data Analysis and Classification*, pp. 1–20, 2019.

[3] R Core Team, *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[4] T. Hothorn and B. Lausen, "Bundling classifiers by bagging trees," *Computational Statistics & Data Analysis*, vol. 49, no. 4, pp. 1068–1078, 2005.

[5] N. Tabassum and T. Ahmed, "A theoretical study on classifier ensemble methods and its applications," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 374–378, IEEE, 2016.

[6] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, pp. 1–18, 2020.

[7] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001, 1990.

[8] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[9] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 1612, pp. 771–780, 1999.

[10] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proceeding of the Thirteenth International Conference on Machine Learning*, pp. 325–352, 1996.

[11] J. Quinlan, "Bagging, boosting, and c4. 5," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 725–730, 1996.

[12] R. Maclin and D. Opitz, "Popular ensemble methods: An empirical study," *Journal of Artificial Research*, vol. 11, pp. 169–189, 2011.

[13] T. Hothorn and B. Lausen, "Double-bagging: Combining classifiers by bootstrap aggregation," *Pattern Recognition*, vol. 36, no. 6, pp. 1303–1309, 2003.

[14] B. Efron and R. Tibshirani, *An introduction to the bootstrap*, vol. 57. Chapman & Hall CRC Press, 1993.

[15] R. A. Berk, Y. He, and S. B. Sorenson, "Developing a practical forecasting screener for domestic violence incidents," *Evaluation Review*, vol. 29, no. 4, pp. 358–383, 2005.

[16] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and regression trees*. Chapman & Hall/CRC, New York, 1984.

[17] D. Steinberg and P. Colla, "Cart: classification and regression trees," *The Top Ten Algorithms in Data Mining*, vol. 9, p. 179, 2009.

[18] R. L. Lawrence and A. Wright, "Rule-based classification systems using classification and regression tree (cart) analysis," *Photogrammetric Engineering and Remote Sensing*, vol. 67, no. 10, pp. 1137–1142, 2001.

[19] W. Y. Loh, "Fifty years of classification and regression trees," *International Statistical Review*, vol. 82, no. 3, pp. 329–348, 2014.

[20] B. Lausen and M. Schumacher, "Maximally selected rank statistics," *Biometrics*, pp. 73–85, 1992.

[21] B. Lausen, T. Hothorn, F. Bretz, and M. Schumacher, "Assessment of optimal selected prognostic factors," *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, vol. 46, no. 3, pp. 364–374, 2004.

[22] S. Potapov, *Improving the split criteria for classification trees and ensemble methods*. Phd dissertation, University of Erlangen-Nuremberg, Germany, 2012.

[23] B. Lausen, W. Sauerbrei, and M. Schumacher, "Classification and regression trees (cart) used for the exploration of prognostic factors measured on different scales," in *Physica-Verlag, a Springer-Verlag Company. Heidelberg. Germany*, pp. 483–496, Springer, 1994.

[24] T. Hothorn and A. Zeileis, "Generalized maximally selected statistics," *Biometrics*, vol. 64, no. 4, pp. 1263–1269, 2008.

[25] T. Hothorn and B. Lausen, "On the exact distribution of maximally selected rank statistics," *Computational Statistics & Data Analysis*, vol. 43, no. 2, pp. 121–137, 2003.

[26] P. Bühlmann, B. Yu, *et al.*, "Analyzing bagging," *The Annals of Statistics*, vol. 30, no. 4, pp. 927–961, 2002.

[27] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[28] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[29] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[30] Z. Khan, A. Gul, O. Mahmoud, M. Miftahuddin, A. Perperoglou, W. Adler, and B. Lausen, "An ensemble of optimal trees for class membership probability estimation," in *Analysis of Large and Complex Data*, pp. 395–409, Springer, 2016.

[31] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthey Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.

[32] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[33] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, no. Nov, pp. 45–66, 2001.

[34] R. Collobert and S. Bengio, "Svmtorch: Support vector machines for large-scale regression problems," *The Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[35] E. Fix and J. L. Hodges Jr, "Discriminatory analysis-nonparametric discrimination: consistency properties," tech. rep., DTIC Document, California University Berkeley, 1951.

[36] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[37] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, *et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[38] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for knn classification," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 3, pp. 1–19, 2017.

[39] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient knn classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2017.

[40] A. Gul, A. Perperoglou, Z. Khan, O. Mahmoud, M. Miftahuddin, W. Adler, and B. Lausen, "Ensemble of a subset of knn classifiers," *Advances in Data Analysis and Classification*, vol. 12, no. 4, pp. 827–840, 2018.

[41] A. Vrahatis, S. Tasoulis, S. Georgakopoulos, and V. Plagianakos, "Ensemble classification through random projections for single-cell rna-seq data," *BioRxiv*, 2020.

[42] T. I. Cannings and R. J. Samworth, "Random-projection ensemble classification," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 79, no. 4, pp. 1–38, 2017.

[43] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of johnson and lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[44] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[45] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[46] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, M. Li, J. Xie, M. Lin, Y. Geng, and Y. Li, *xgboost: Extreme Gradient Boosting*, 2021. R package version 1.4.1.1.

[47] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 3, p. e1301, 2019.

[48] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Information Fusion*, vol. 6, no. 1, pp. 5–20, 2005.

[49] T. T. Nguyen, A. V. Luong, M. T. Dang, A. W.-C. Liew, and J. McCall, "Ensemble selection based on classifier prediction confidence," *Pattern Recognition*, vol. 100, p. 107104, 2020.

[50] P. Latinne, O. Debeir, and C. Decaestecker, "Limiting the number of trees in random forests," in *International Workshop on Multiple Classifier Systems*, pp. 178–187, Springer, 2001.

[51] S. Bernard, L. Heutte, and S. Adam, "On the selection of decision trees in random forests," in *2009 International Joint Conference on Neural Networks*, pp. 302–307, IEEE, 2009.

[52] M. Wang and H. Zhang, "Search for the smallest random forest," *Statistics and its Interface*, vol. 2, no. 3, pp. 381–388, 2009.

[53] H. B. Li, W. Wang, H. W. Ding, and J. Dong, "Trees weighting random forest method for classifying high-dimensional noisy data," in *Proceedings of the IEEE 7th International Conference on E-Business Engineering (ICEBE)*, pp. 160–163, 2010.

[54] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?," *Machine Learning and Data Mining in Pattern Recognition*, pp. 154–168, 2012.

[55] R. Durrant and A. Kabán, "Sharp generalization error bounds for randomly-projected classifiers," in *International Conference on Machine Learning*, pp. 693–701, 2013.

[56] R. J. Durrant and A. Kabán, "Random projections as regularizers: learning a linear discriminant from fewer observations than dimensions," *Machine Learning*, vol. 99, no. 2, pp. 257–286, 2015.

[57] M. Lopes, L. Jacob, and M. J. Wainwright, "A more powerful two-sample test in high dimensions using random projection," in *Advances in Neural Information Processing Systems*, pp. 1206–1214, 2011.

[58] T. L. Marzetta, G. H. Tucci, and S. H. Simon, "A random matrix-theoretic approach to handling singular covariance estimates," *IEEE Transactions on Information Theory*, vol. 57, no. 9, pp. 6256–6271, 2011.

[59] A. Frank and A. Asuncion, "Uci machine learning repository, 2010," *URL http://archive. ics. uci. edu/ml*, vol. 15, p. 22, 2011.

[60] B. D. Ripley, "The r project in statistical computing," *MSOR Connections. The Newsletter of the LTSN Maths, Stats & OR Network*, vol. 1, no. 1, pp. 23–25, 2001.

[61] Z. Khan, A. Gul, A. Perperoglou, O. Mahmoud, W. Adler, Miftahuddin, and B. Lausen, *OTE: Optimal Trees Ensembles for Regression, Classification and Class Membership Probability Estimation*, 2015. R package version 1.0.

[62] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[63] T. I. Cannings and R. J. Samworth, *RPEnsemble: Random Projection Ensemble Classification*, 2016. R package version 0.3.

[64] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.

[65] R. Tibshirani, "Regression shrinkage and selection via the LASSO," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[66] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data Classification: Algorithms and Applications*, p. 37, 2014.

[67] J. Friedman, T. Hastie, N. Simon, and R. Tibshirani, "LASSO and elastic-net regularized generalized linear models," 2016. R-package version 2.0-5.

[68] D. Fradkin and D. Madigan, "Experiments with random projections for machine learning," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 517–522, 2003.

[69] V. Sulic, J. Perš, M. Kristan, and S. Kovacic, "Efficient dimensionality reduction using random projection," in *15th Computer Vision Winter Workshop*, pp. 29–36, 2010.

[70] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2019. R package version 1.7-3.

[71] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis, "kernlab-an s4 package for kernel methods in r," *Journal of Statistical Software*, vol. 11, no. 9, pp. 1–20, 2004.

[72] J. D. Malley, J. Kruppa, A. Dasgupta, K. G. Malley, and A. Ziegler, "Probability machines," *Methods of Information in Medicine*, vol. 51, no. 01, pp. 74–81, 2012.

[73] J. Kruppa, A. Ziegler, and I. R. König, "Risk estimation and risk prediction using machine-learning methods," *Human Genetics*, vol. 131, no. 10, pp. 1639–1654, 2012.

[74] A. Barla, G. Jurman, S. Riccadonna, S. Merler, M. Chierici, and C. Furlanello, "Machine learning methods for predictive proteomics," *Briefings in Bioinformatics*, vol. 9, no. 2, pp. 119–128, 2008.

[75] P. Sajda, "Machine learning for detection and diagnosis of disease," *Annu. Rev. Biomed. Eng.*, vol. 8, pp. 537–565, 2006.

[76] B. Yang and Y. Chen, "Somatic mutation detection using ensemble of flexible neural tree model," *Neurocomputing*, vol. 179, pp. 161–168, 2016.

[77] M. Ram, A. Najafi, and M. T. Shakeri, "Classification and biomarker genes selection for cancer gene expression data using random forest," *Iranian Journal of Pathology*, vol. 12, no. 4, p. 339, 2017.

[78] R. S. Croner, A. Peters, W. M. Brueckl, K. E. Matzel, L. Klein-Hitpass, T. Brabletz, T. Papadopoulos, W. Hohenberger, B. Reingruber, and B. Lausen, "Microarray versus conventional prediction of lymph node metastasis in colorectal carcinoma," *Cancer: Interdisciplinary International Journal of the American Cancer Society*, vol. 104, no. 2, pp. 395–404, 2005.

[79] R. S. Croner, T. Förtsch, W. M. Brückl, F. Rödel, C. Rödel, T. Papadopoulos, T. Brabletz, T. Kirchner, M. Sachs, J. Behrens, *et al.*, "Molecular signature for lymphatic metastasis in colorectal carcinomas," *Annals of Surgery*, vol. 247, no. 5, pp. 803–810, 2008.

[80] S. Gill, C. L. Loprinzi, D. J. Sargent, S. D. Thomé, S. R. Alberts, D. G. Haller, J. Benedetti, G. Francini, L. E. Shepherd, J. Francois Seitz, *et al.*, "Pooled analysis of fluorouracil-

based adjuvant therapy for stage ii and iii colon cancer: who benefits and by how much?," *Journal of Clinical Oncology*, vol. 22, no. 10, pp. 1797–1806, 2004.

[81] "Colon cancer alliance." `https://www.ccalliance.org/get-information/staging/`. Accessed: 2017-07-18.

[82] H. Rao, X. Shi, A. K. Rodrigue, J. Feng, Y. Xia, M. Elhoseny, X. Yuan, and L. Gu, "Feature selection based on artificial bee colony and gradient boosting decision tree," *Applied Soft Computing*, vol. 74, pp. 634–642, 2019.

[83] D. Dernoncourt, B. Hanczar, and J. D. Zucker, "Analysis of feature selection stability on high dimension and small sample data," *Computational Statistics & Data Analysis*, vol. 71, pp. 681–693, 2014.

[84] K. H. Chen, K. J. Wang, M. L. Tsai, K. M. Wang, A. M. Adrian, W. C. Cheng, T. S. Yang, N. C. Teng, K. P. Tan, and K. S. Chang, "Gene selection for cancer identification: a decision tree model empowered by particle swarm optimization algorithm," *BMC Bioinformatics*, vol. 15, no. 1, p. 49, 2014.

[85] C. Zou, J. Gong, and H. Li, "An improved sequence based prediction protocol for dna-binding proteins using svm and comprehensive feature analysis," *BMC Bioinformatics*, vol. 14, no. 1, p. 90, 2013.

[86] M. Marczyk, R. Jaksik, A. Polanski, and J. Polanska, "Adaptive filtering of microarray gene expression data based on gaussian mixture decomposition," *BMC Bioinformatics*, vol. 14, no. 1, p. 101, 2013.

[87] Y. Liu, F. Tang, and Z. Zeng, "Feature selection based on dependency margin," *IEEE Transactions on Cybernetics*, vol. 45, no. 6, pp. 1209–1221, 2014.

[88] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[89] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[90] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 02, pp. 185–205, 2005.

[91] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *Journal of Machine Learning Research*, vol. 5, no. Oct, pp. 1205–1224, 2004.

[92] M. Shardlow, "An analysis of feature selection techniques," *The University of Manchester*, vol. 1, pp. 1–7, 2016.

[93] S. M. Alladi, P. Shinde Santosh, V. Ravi, and U. S. Murthy, "Colon cancer prediction with genetic profiles using intelligent techniques," *Bioinformation*, vol. 3, no. 3, p. 130, 2008.

[94] S. Rathore, M. Hussain, and A. Khan, "Gecc: Gene expression based ensemble classification of colon samples," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 6, pp. 1131–1145, 2014.

[95] O. R. Olaniran and M. A. A. Abdullah, "Gene selection for colon cancer classification using bayesian model averaging of linear and quadratic discriminants," *Journal of Science and Technology*, vol. 9, no. 3, 2017.

[96] J. A. C. Garzón and J. R. González, "A gene selection approach based on clustering for classification tasks in colon cancer," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 4, no. 3, pp. 1–10, 2015.

[97] Y. Liang, C. Liu, X. Z. Luan, K. S. Leung, T. M. Chan, Z. B. Xu, and H. Zhang, "Sparse logistic regression with a l 1/2 penalty for gene selection in cancer classification," *BMC Bioinformatics*, vol. 14, no. 1, p. 198, 2013.

[98] C. G. A. Network *et al.*, "Comprehensive molecular characterization of human colon and rectal cancer," *Nature*, vol. 487, no. 7407, p. 330, 2012.

[99] O. Mahmoud, A. Harrison, A. Perperoglou, A. Gul, Z. Khan, M. V. Metodiev, and B. Lausen, "A feature selection method for classification within functional genomics experiments based on the proportional overlapping score," *BMC Bioinformatics*, vol. 15, no. 1, p. 274, 2014.

[100] O. Mahmoud, A. Harrison, A. Perperoglou, A. Gul, Z. Khan, and B. Lausen, *Propoverlap: Feature (gene) selection based on the Proportional Overlapping Overlapping scores*, 2014. R package version 1.0.

[101] W. Adler, O. Gefeller, A. Gul, F. K. Horn, Z. Khan, and B. Lausen, "Ensemble pruning for glaucoma detection in an unbalanced data set," *Methods of Information in Medicine*, vol. 55, no. 6, pp. 557–563, 2016.

[102] Y. Piao, H. W. Park, C. H. Jin, and K. H. Ryu, "Ensemble method for classification of high-dimensional data," in *International Conference on Big Data and Smart Computing (BIGCOMP)*, pp. 245–249, IEEE, 2014.

[103] X. Zeng, D. F. Wong, and L. S. Chao, "Constructing better classifier ensemble based on weighted accuracy and diversity measure," *The Scientific World Journal*, vol. 2014, 2014.

[104] N. Gul, N. Faiz, D. Brawn, R. Kulakowski, Z. Khan, and B. Lausen, "Optimal survival trees ensemble," *arXiv preprint arXiv:2005.09043*, 2020.

[105] C. Hurley, *gclus: Clustering Graphics*, 2012. R package version 1.3.1.

[106] W. Adler, A. Peters, and B. Lausen, "Comparison of classifiers applied to confocal scanning laser ophthalmoscopy data," *Methods of Information in Medicine*, vol. 47, no. 1, pp. 38–46, 2008.

[107] J. J. Goeman, *penalized: Penalized generalized linear models.*, 2012. Penalized R package, version 0.9-42.

[108] J. Khan, J. S. Wei, M. Ringner, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, *et al.*, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Medicine*, vol. 7, no. 6, pp. 673–679, 2001.

[109] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. Kim, L. C. Goumnerova, P. M. Black, C. Lau, *et al.*, "Prediction of central nervous

system embryonal tumour outcome based on gene expression," *Nature*, vol. 415, no. 6870, pp. 436–442, 2002.

[110] S. Michiels, S. Koscielny, and C. Hill, "Prediction of cancer outcome with microarrays: a multiple random validation strategy," *The Lancet*, vol. 365, no. 9458, pp. 488–492, 2005.

Appendix A

Benchmark Problems

A.1 Benchmark datasets

A brief description of benchmark datasets used throughout the study supporting the results, are given below.

A.1.1 Eyestate detection

The electroencephalogram Eyestate dataset is available on `http://archive.ics.uci.edu/ml/datasets/EEG+Eye+State`. This dataset consists of 14 features and 14,980 observations. The aim is to make use of the electroencephalogram reading to identify the state of the eye. There are 8,255 observations for which the eye is opened and 6,723 for which the eye is closed.

A.1.2 Hepatitis

Hepatitis is a disease where the liver begins to be reddened or swollen. The chronic stage starts when the disease continues for more than six months. This dataset has been taken from `https://archive.ics.uci.edu/ml/datasets/Hepatitis`. There are some missing observations but in the present study, cases with missing values on observations have not been considered. This dataset contains 19 features and 80 observations.

A.1.3 Parkinson

Parkinson's disease is a slowly progressive ailment of the nervous system that damages part of the brain with the passage of time. This disease commonly affects middle-aged and elderly people. These are a set of 195 observations on 23 features. Parkinson dataset is given on machine learning repository `https://archive.ics.uci.edu/ml/datasets/Parkinson`.

A.1.4 Body

The dataset is available within "gclus" [105] library of the *R* package. It consists of 507 observations on 24 features collected from 247 men and 260 women mostly in the age group of 20-30 years. The features involve height, age, weight and 21 other different body measurements. The class variable is gender having two categories, male and female.

A.1.5 Thyroid

Thyroid disease damages the function of the thyroid glands. Initially the problem was to distinguish a person as hypothyroid or not. The data was extracted from the Garvan Institute of Medical Research and the dataset can be found at `https://archive.ics.uci.edu/ml/datasets/Thyroid+Disease`. Two variables, T3 and TBG, have more than 20% missing observations. Therefore, these two variables were removed from the data. This dataset has 9172 observations and 27 features.

A.1.6 WDBC

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset is available at UCI Machine Learning Repository [59]. The problem is to identify breast tumors as malignant or benign. The features are the characteristics of the nuclei. It contains 569 observations for 30 different features.

A.1.7 WPBC

WPBC stands for Wisconsin Prognostic Breast Cancer. The dataset is taken from the UCI Machine Learning Repository [59]. The dataset consists of a total of 198 observations on 33 features that measure factors relevant to breast cancer. The class variable is the status of breast cancer to be either "R" (recurring) or "N" (nonrecurring).

A.1.8 Ionosphere

The Ionosphere is an area of the atmosphere, situated 85-600 km above the earth surface. The dataset is taken from `http://archive.ics.uci.edu/ml/datasets/Ionosphere` consists of 33 high frequency antenna measurements (features) on 351 observations. Observations are divided into two classes i.e. good or bad, depending on whether there is evidence for free electrons in the ionosphere or not. The class sizes contain 225 (good) and 126 (bad).

A.1.9 Oil-spill

An Oil-spill is a petroleum hydrocarbon in liquid form released into water either from underwater pipelines or from tankers. The Oil-spill dataset is available on `http://openml.org/`. This dataset holds 937 observations on 49 features. Classifying Oil-spill is a binary class problem of distinguishing between oil-spill and non-oil-spill regions in oil-spill images.

A.1.10 Spambase

The Spambase dataset is taken from the UCI Machine Learning Repository `https://archive.ics.uci.edu/ml/datasets/Spambase` and classifies emails into spam and non-spam. This dataset includes 4601 observations on 57 features in total. These variables indicate either a specific letter or word is taking place often in the email or not.

A.1.11 Glaucoma

Glaucoma is a disease of the eye considered as the second most common reason of blindness worldwide. The dataset consists of measurements on 62 different features on 196 observations to detect a "glaucomatous" or a "normal" eye [106]. All the feature variables are continuous and the response variable is binary.

A.1.12 Nki70

Nki70 is a data frame which contains 144 lymph node positive breast cancer patients for 76 features. There are 5 clinical risk factors in 76 features, one is age of patient and the remaining are gene expression measurements. This dataset can be obtained from "penalized" R package [107].

A.1.13 Musk

Musk is a sweet smelling and reddish-brown material employed in the making of perfumes. There are 92 molecules in the Musk dataset which can be broke down into two groups i.e., musk and non-musk. The task is to classify new instances. There are 476 observations on 166 features that describe the observations for the 92 molecules. This dataset is taken from `https://archive.ics.uci.edu/ml/datasets/Musk+(Version+1)`.

A.1.14 Sonar

This dataset is a composition of sonar signals. Each signal pattern consists of 60 numbers in an interval from 0.0 to 1.0, where the numbers show the energies within a frequency band over a specific time period. The classes are labelled as "R" for rock and "M", for mine (metal cylinder) object respectively. The aim is to distinguish between signals whether it is bounced off a rock or a metal cylinder. The data can be obtained from UCI machine learning repository `https://www.openml.org/d/40`. There are 208 observation and 60 features in the dataset.

A.1.15 Two norms

This is a binary class problem dataset with a total of 7400 observations and 20 features. The two classes of the dataset are generated using a multivariate normal distribution. Class 1 has mean = mean(a,a,..,a) and class 2 has mean = mean(-a,-a,..,-a) where a = $2/\sqrt{20}$. This dataset is approachable on `https://www.openml.org/d/1507`.

A.1.16 Vowel

This Vowel dataset provides details about speaker independent recognition of the eleven steady state vowels of British English. This dataset is taken from `https://sci2s.ugr.es/keel/dataset.php?cod=113` and comprising 13 features on 990 observations. This version is a combination of the original two datasets available at the UCI machine learning repository [59].

A.1.17 Wind

This dataset is Binarized version of the original Wind dataset. It gives daily average wind speeds for the years 1961 to 1978 at 12 stations in the Republic of Ireland. It comprises of 6574 observations and 14 features obtained from `https://www.openml.org/d/503`. It changes the numeric target variable to a binary class nominal variable by calculating the mean and the observations with a small target value is labelled as positive (P) and the remaining as negative (N).

A.1.18 Thoracic Surgery

The dataset is collected at Wroclaw Thoracic Surgery Centre. It is related to patients who went under major surgery for primary lung cancer in the years 2007 to 2011. The Wroclaw Thoracic Surgery Centre is connected with the Department of Thoracic Dataset from UCI repository. This dataset is accessible on `https://www.openml.org/d/4329` consisting of 16 features on 470 observations.

A.1.19 Ring

The Ring dataset is taken from `https://sci2s.ugr.es/keel/dataset.php?cod=106`. This is a binary classification problem, consisting of 20 features on 7400 observations. Classes are generated from a multivariate normal distribution. Class 1 has zero mean and class 2 has mean(a,a,..,a) where a = $2/\sqrt{20}$.

A.1.20 Collins

This dataset is generated from the text using Docuscope software. The multi-class response variable is turned to a binary class target variable by assigning the majority class as positive and others as negative. It has 23 features and 500 observations taken from `https://www.openml.org/search?q=collins&type=data`.

A.1.21 White-clover

The White-clovers are round heads of white flowers. This dataset contains information on the existence of white clover in summer dry hill. It consists of 31 features and 63 observations available at `https://www.openml.org/d/343`. It converts the multi-class

response variable into a binary class nominal target variable by considering the majority class as 'P' (possitive) and the remaining as 'N' (negative).

A.1.22 Backache

The dataset is taken from `https://www.openml.org/search?q=backache&type=data`. This dataset is related to "Problem-Solving" on "backache in pregnancy". It consists of 32 features on 180 observations.

A.1.23 Wisconsin

This dataset is taken from `https://www.openml.org/d/191`. It transforms the numeric target variable into a binary nominal target variable by calculating the mean. It classifies the observations as positive (P) having a lower target value otherwise as negative (N). There are 32 features on 194 observations.

A.1.24 Coil

The Coil dataset is available at `https://sci2s.ugr.es/keel/dataset.php?cod=164` and used in Coil 2000 Challenge. This dataset gives details of customers of an insurance company. It includes 85 features and 9822 observations including product usage and socio-demographic data.

A.1.25 Hill-valley

The Hill–valley dataset includes 606 instances of a terrain and 100 features available at `http://archive.ics.uci.edu/ml/datasets/Hill-Valley`. It represents either a hill or a valley when the points are plotted in a sequence. The main objective is to predict the class of terrain.

A.1.26 USPS

This dataset is the Binarized version of the original USPS dataset (version 2) about normalized handwritten digits. The U.S. Postal Service scans the binary digits of various sizes and positions from envelopes. The classes are labelled from 1 to 10. The observations in the original dataset with class labels 6 and 9 are assigned values as 0 and 1 respectively. This can be downloaded from `https://www.openml.org/d/41082`. There are 1424 observations and 256 features.

A.1.27 Isolet

ISOLET (Isolated Letter Speech Recognition) dataset is taken from `https://www.openml.org/d/41966`. It has 617 features on 600 observations. The aim of this dataset is to recognise which name or letter is spoken.

A.2 Microarray datasets

A detailed description of microarray datasets used throughout the study supporting the results, are given in the following section.

A.2.1 SRBCT

This dataset presents gene expression profiling related to small round blue-cell tumors of childhood (SRBCT) in [108] for classifying them into four classes. The dataset can be obtained from `https://cran.r-project.org/web/packages/plsgenomics/index.html`. It consists of 2,308 genes and 54 samples. All the 2,308 genes are standardized with zero mean and unit variance.

A.2.2 Colon

Colon is a microarray dataset measuring expression level for 6500 genes on 40 tumors and 22 normal tissues. A set of 2000 genes for 62 samples with minimum intensity are taken. This dataset is accessible on `http://microarray.princeton.edu/oncology/`. The response variable is binary which takes 0 and 1 values to represent tumor and normal tissues.

A.2.3 Tumors C

Tumors C gives information on embryonal tumours of the central nervous system which is a dissimilar group of tumors. This dataset is generated and used in [109] to predict the central nervous system embryonal tumour, consisting of 7129 features and 60 samples. It can be downloaded from `https://www.openml.org/d/1107`.

A.2.4 Breast

This kind of cancer is common in women. Lobural and Ductal carcinoma are the two types of breast cancer. Breast cancer can be classified as invasive and non-invasive. The dataset is chosen from a collection of breast cancer patient samples. This is a binary class problem and the dataset presents 78 observations and 4,948 genes/features [110].

A.2.5 Prostate

The Prostate gland is located under the bladder in males. Its function is to secrete a type of fluid that keep sperms cells safe. Cancer may begin in these cells which produce this fluid. Prostate is a microarry dataset containing of 6033 genes/features of 102 individuals. The problem is to diagnose prostate tumor and normal tissues. The data set is available on `http://www.gems-system.org`.

A.2.6 Leukemia

The leukemia dataset was taken from a collection of leukamia patient samples which is often used as a benchmark dataset in microarray analysis techniques. It presents gene expressions parallel to Acute Lymphoblast Leukaemia and Acute Myeloid Leukemia. The dataset is composed of 72 samples and 7,129 genes. This dataset can be downloaded from `https://www.openml.org/d/1104`.

A.2.7 Adenocarcinoma

Adenocarcinoma is a type of lung cancer. This dataset consists of 9868 genes/features and 76 samples in two classes (binary class problem). This data is given on `http://ligarto.org/rdiaz/Papers/rfVS/randomForestVarSel.html`.

A.2.8 Lung

The Lung dataset is a gene expression data for lung cancer classification between two classes, adenocarcinoma and malignant pleural mesothe-lioma. It can start on the outermost layer of the lungs. People who smoke or are former smokers are more at risk of getting lung cancer. The lung dataset contains 181 tissue samples, 150 adenocarcinomas and 31 malignant pleural mesothe-lioma. There are 12,533 genes/features and is taken from `https://cran.r-project.org/web/packages/propOverlap/index.html`.

A.2.9 Ovarian

Ovarian cancer is also common in women. One out of 73 women is diagnosed with ovarian cancer in their lifetime. It starts in the ovaries and can grow to the other parts in abdomen. The dataset is downloaded from the Clinical Proteomics Program Databank website, `http://clinicalproteomics.steem.com/download-ovar.php`. This dataset containing 15,154 genes/features for 253 observations. The dataset was prepared using the Ciphergen WCX2 ProteinChip array.

Appendix B

Tables and box plots

Table B.1 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Spambase, Musk and Leukaemia datasets.* **n** *is the training set size. OTE is computed by considering* `t.initial`*=1000 and different values of* **p**. *Random Forest is computed by fixing* `ntrees`*=1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	Spambase						Musk						Leukaemia					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees
0.1	0.0589	51.98	0.0516	52.69	0.0472	53.55	0.1991	50.97	0.1313	51.64	**0.1006**	51.64	0.1254	29.28	0.0344	37.62	0.0242	34.07
0.2	0.0587	101.78	**0.0512**	103.13	0.0468	104.3	0.1945	101.39	0.1288	101.88	0.1031	102.56	**0.1244**	58.62	**0.0317**	85.34	0.0228	81.74
0.3	0.0585	151.61	0.0514	152.59	0.0472	154.28	0.1942	152.98	0.1288	151.5	0.1016	152.87	0.1246	87.81	0.0344	133.52	0.0228	133.59
0.4	0.0584	201.77	0.0512	202.71	0.0470	204.92	0.1945	203.31	**0.1281**	201.27	0.1027	202.06	0.1268	116.85	0.0386	179.74	0.0228	187.92
0.5	0.0585	251.69	0.0514	253.14	0.0472	254.28	0.1938	253.35	0.1287	251.48	0.1047	251.5	0.1286	145.57	0.0406	226.39	**0.0214**	241.79
0.6	0.0584	302.21	0.0514	302.46	0.0473	305.06	0.1939	303.4	0.1283	301.77	0.1029	301.39	0.1284	173.6	0.0431	273.32	0.0242	295.81
0.7	**0.0583**	352	0.0515	352.41	0.0475	355.18	**0.1937**	353.5	0.1293	351.78	0.1058	349.92	0.1312	201.50	0.0441	318.51	0.0214	349.10
0.8	0.0584	401.61	0.0516	402.17	0.0475	405.03	0.1943	403.47	0.1299	401.66	0.1058	398.34	0.1338	228.35	0.0458	362.65	0.0214	403.19
0.9	0.0585	451.88	0.0516	451.30	0.0476	454.74	0.1939	454.37	0.1296	452.13	0.1070	448.32	0.1428	253.61	0.0479	406.59	0.0214	455.78
1	0.0585	501.53	0.0518	501.47	0.0477	504.83	0.1939	504.22	0.1301	501.61	0.1079	496.91	0.2060	278.60	0.0489	446.28	0.0228	505.42
RF	0.0576		0.0511		0.0476		0.1873		0.1267		0.1087		0.1160		0.0513		0.0428	

Table B.2 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Lung, Breast and Prostate datasets.* **n** *is the training set size. OTE is computed by considering* **t.initial**=*1000 and different values of* **p**. *Random Forest is computed by fixing* **ntrees**=*1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	Lung						Breast						Prostate					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees
0.1	0.0694	25.95	0.0283	38.35	0.0188	43.31	**0.4081**	55.80	0.3790	50.31	0.3837	51.40	**0.1830**	46.78	**0.0968**	52.11	0.0730	52.43
0.2	0.0645	51.03	0.0238	78.98	0.0161	84.64	0.4105	112.45	0.3763	100.20	0.3750	101.50	0.1838	95.52	0.0990	102.39	**0.0710**	104.50
0.3	0.0618	77.62	0.0215	118.25	0.0161	131.47	0.4111	170.23	**0.3726**	150.55	0.3875	152.91	0.1881	145.75	0.1046	154.26	0.0750	153.93
0.4	0.0586	103.45	0.0186	158.54	0.0155	176.36	0.4113	227.91	0.3783	202.64	0.3737	203.16	0.1916	195.37	0.1070	206.69	0.0710	205.06
0.5	0.0559	130.34	0.0177	199.64	0.0150	220.48	0.4141	284.92	0.3800	253.54	0.3725	253.99	0.1985	245.76	0.1095	259.6	0.0730	255.79
0.6	0.0517	157.79	0.0165	241.28	0.0133	263.86	0.4152	341.94	0.3803	304.16	**0.3650**	305.80	0.2007	296.24	0.1141	311.41	0.0730	306.75
0.7	0.0480	185.45	0.0151	283.39	0.0133	308.35	0.4167	398.89	0.3853	354.02	0.3712	357.34	0.2038	346.54	0.1148	363.72	0.0750	356.53
0.8	**0.0444**	213.37	0.0138	325.72	0.0133	350.99	0.4181	455.72	0.3840	403.93	0.3675	407.89	0.2069	397.06	0.1217	416.17	0.0790	406.05
0.9	0.0468	239.39	0.0141	454.95	0.0122	395.73	0.4194	512.02	0.3826	453062	0.3662	458.12	0.2119	447.7	0.1231	468.49	0.0790	455.84
1	0.0485	265.43	**0.0131**	395.50	**0.0111**	441.47	0.4145	567.01	0.3870	499.72	0.3725	499.23	0.2164	498.49	0.1265	518.51	0.0820	504.33
RF	0.0588		0.0104		0.0066		0.4139		0.3806		0.3750		0.1925		0.1248		0.0770	

Table B.3 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Body, WPBC and WDBC datasets.* **n** *is the training set size. OTE is computed by considering* **t.initial**=*1000 and different values of p. Random Forest is computed by fixing* **ntrees**=*1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	Body						WPBC						WDBC					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees
0.1	0.0627	50.55	0.0437	52.4	0.0401	53.21	0.2668	37.56	0.2312	47.43	0.2084	48.29	0.0553	46.88	0.0423	51.38	0.0340	50.77
0.2	**0.0614**	99.3	0.0455	103.44	**0.0390**	104.06	0.2602	74.95	0.2267	93.41	0.2036	96.68	0.0539	93.77	**0.0414**	101.09	0.0356	101.46
0.3	0.0619	148.67	0.0459	153.64	0.0409	154.24	0.2590	112.59	0.2252	139.63	0.2010	144.79	0.0543	140.04	0.0422	151.34	0.0345	150.16
0.4	0.0621	197.9	0.0472	203.51	0.0409	205.67	0.2577	151.31	0.2257	185.88	**0.1963**	192.36	0.0536	186.32	0.0417	201.05	0.0343	200.15
0.5	0.0623	246.96	0.0479	252.42	0.0401	256.04	0.2575	190.55	**0.2242**	232.22	0.1989	241.79	0.0537	232.74	0.0417	251.25	**0.0338**	249.11
0.6	0.0623	295.69	0.0476	302.12	0.0407	307.51	0.2551	230.03	0.2253	278.80	0.2005	291.22	0.0540	280.64	0.0418	302.9	0.0338	298.95
0.7	0.0628	344.71	0.0481	351	0.0409	358.08	0.2556	270.55	0.2271	325.91	0.2021	339.87	0.0536	329.05	0.0421	352.94	0.0345	348.59
0.8	0.0630	393.88	0.0488	400.73	0.0417	408.88	0.2547	311.15	0.2266	373.06	0.2005	390.27	0.0536	375.89	0.0422	402.3	0.0349	399.35
0.9	0.0632	442.96	0.0497	501.24	0.0415	459.75	0.2555	353.45	0.2269	420.56	0.2026	440.35	**0.0534**	423.62	0.0417	452.91	0.0354	449.68
1	0.0640	493.21	**0.0419**	510.34	0.0415	459.75	**0.2545**	399.35	0.2273	470.21	0.2031	492.22	0.0531	472.11	0.0421	504.08	0.0354	500.06
RF	0.0600		0.0471		0.0384		0.2444		0.2248		0.2052		0.0514		0.0429		0.0345	

Table B.4 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Hepatitis, Parkinson and Thyroid datasets.* **n** *is the training set size. OTE is computed by considering* **t.initial**=*1000 and different values of* **p**. *Random Forest is computed by fixing* **ntrees**=*1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	Hepatitis						Parkinson						Thyroid					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees
0.1	0.1646	49.06	0.1362	55.97	0.1187	55.33	0.1186	47.58	**0.0597**	53.48	**0.0305**	53.45	**0.0125**	52.92	0.0108	52.67	0.0101	51.18
0.2	0.1518	117.42	0.1331	125.48	0.1225	114.89	**0.1160**	106.31	0.0610	109.93	0.0336	107.07	0.0125	103.75	0.0108	101.61	0.0099	100.61
0.3	0.1466	191.58	0.1334	197.31	0.1200	175.59	0.1175	168.37	0.0616	164.95	0.0352	160.35	0.0125	152.93	0.0107	150.57	0.0099	150.12
0.4	0.1415	269.55	0.1325	270.35	0.1162	236.78	0.1197	231.40	0.0637	220.27	0.0363	213.36	0.0126	202.55	0.0107	199.26	0.0099	198.8
0.5	0.1399	350.06	0.1318	343.38	0.1175	299.03	0.1208	294.39	0.0652	275.68	0.0357	267.38	0.0126	251.52	0.0107	246.73	0.0098	246.99
0.6	0.1385	430.32	0.1300	416.37	0.1162	361.77	0.1237	358.06	0.0667	330.48	0.0394	320.52	0.0127	298.85	**0.0106**	294.44	0.0098	293.88
0.7	0.1399	510.39	0.1259	490.19	0.1137	424.93	0.1255	421.6	0.0680	383.8	0.0394	372.13	0.0128	345.85	0.0106	340.52	0.0097	339.36
0.8	0.1392	589.79	**0.1237**	562.86	0.1125	488.09	0.1242	485.34	0.0705	437.42	0.0431	425.61	0.0128	398	0.0107	384.5	0.0097	382.7
0.9	0.1385	668.78	0.1243	634.67	**0.1112**	551.01	0.1245	549.79	0.0729	489.24	0.0463	477.95	0.0129	432.22	0.0106	424.97	0.0098	424.4
1	**0.1372**	742.86	0.1240	704.48	0.1125	611.56	0.1264	611.85	0.0743	540.19	0.0484	528.93	0.0131	471.31	0.0107	461.62	**0.0095.**	472.31
RF	0.1329		0.1215		0.10		0.1188		0.0734		0.0521		0.0.0131		0.0105		0.0098	

Table B.5 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Nki70, Glaucoma and Oil-spill datasets.* **n** *is the training set size. OTE is computed by considering* **t.initial**=*1000 and different values of* **p**. *Random Forest is computed by fixing* **ntrees**=*1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	Nki70						Glaucoma						Oil-spill					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees	error	mean trees
0.1	**0.2154**	44.96	0.1563	46.04	0.1307	49.18	**0.1639**	50.12	0.1297	51.83	0.1035	51.69	**0.0414**	63.87	**0.0366**	54.98	**0.0320**	54.52
0.2	0.2202	91.59	**0.1550**	93.51	**0.1264**	96.83	0.1714	87.96	0.1283	103.27	**0.0970**	99.13	0.0414	128.98	0.0370	109.36	0.0325	108.08
0.3	0.2269	139.04	0.1550	140.71	0.1292	145.22	0.1722	135.19	0.1285	153.73	0.0970	148.11	0.0416	194.59	0.0370	162.59	0.0337	161
0.4	0.2369	187.22	0.1579	186.8	0.1300	192.91	0.1713	183.88	0.1261	203.5	0.0995	197.58	0.0416	259.8	0.0373	216.11	0.0347	214.16
0.5	0.2436	235.78	0.1637	232.15	0.1307	239.52	0.1697	232.62	0.1260	253.8	0.1000	246.3	0.0416	323.98	0.0378	269.25	0.0347	266.94
0.6	0.2488	284.76	0.1677	277.52	0.1335	285.01	0.1692	280.47	0.1255	303.79	0.0990	295.26	0.0417	388.30	0.0381	321.24	0.0352	318.77
0.7	0.2550	334.19	0.17.62	323.65	0.1378	332.03	0.1694	329.01	0.1269	354.32	0.1058	349.92	0.0419	451.85	0.0384	373.31	0.0355	370.72
0.8	0.2637	384.04	0.1829	370.31	0.1400	378.89	0.1676	377.27	0.1256	404.54	0.1010	393.27	0.0421	514.07	0.0386	424.41	0.0364	421.47
0.9	0.2672	434.52	0.1882	417.76	0.1457	425.83	0.1667	426.14	**0.1238**	454.28	0.1070	448.32	0.0422	574.11	0.0389	474.64	0.0368	472.34
1	0.2692	486.28	0.1924	465.4	0.1478	474.74	0.1670	474.02	0.1252	502.96	0.1000	492.91	0.0423	629.96	0.0395	522.55	0.0377	520.3
RF	0.2624		0.1994		0.1535		0.1592		0.1341		0.1100		0.0425		0.0386		0.0368	

Table B.6 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Adeno, Ovarian and Eyestate datasets.* **n** *is the training set size. OTE is computed by considering* **t.initial**=*1000 and different values of* **p**. *Random Forest is computed by fixing* **ntrees**=*1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	Adeno						Ovarian						Ionosphere					
	n=30%		n=60%		n=90%		n=30%		n=60%		n=90%		n=30%		n=60%		n=90%	
	error	**mean trees**	error	**mean trees**	**error**	**mean trees**	**error**	**mean trees**	**error**	**mean trees**	**error**	**mean trees**	**error**	**mean trees**	**error**	**mean trees**	**error**	**mean trees**
0.1	0.2070	17.85	0.1636	28.16	0.1487	37.16	0.0371	42.98	0.0157	50.05	**0.0072**	48.06	0.0815	56.89	0.0661	54.14	**0.0648**	52.51
0.2	0.1877	35.42	0.1650	55.54	**0.1362**	76.11	**0.0326**	87.07	**0.0135**	100.14	0.0072	97.08	0.0800	116.2	0.0654	107.24	0.0660	104.52
0.3	0.1806	50.27	0.1610	83.17	0.1399	105.02	0.0346	133.04	0.0142	147.52	0.0080	146.63	0.0788	174.07	**0.0652**	158.69	0.0654	156.58
0.4	0.1801	66.25	0.1613	110.5	0.1425	151.32	0.0366	180.50	0.0141	201.7	0.0096	195.10	0.0779	231.7	0.0655	210.77	0.0674	207.95
0.5	0.1792	82.47	0.1596	138.3	0.1425	190.49	0.0390	228.05	0.0156	252.66	0.0108	243.63	0.0784	289.31	0.0658	262.43	0.0682	259.44
0.6	0.1773	100.3	0.1616	167.23	0.1437	230.09	0.0420	275.15	0.0419	275.97	0.0104	295.97	0.0778	347.08	0.0654	313.66	0.0680	309.97
0.7	**0.1759**	118.37	0.1623	196.92	0.1437	269.26	0.0442	323.89	0.0170	352.32	0.0112	342.15	**0.0773**	403.80	0.0654	364.73	0.0674	360.61
0.8	0.1764	137.95	0.1626	227.85	0.1437	310.39	0.0420	375.15	0.0181	396.18	0.0132	394.18	0.0775	461.15	0.0652	417.11	0.0668	411.99
0.9	0.1798	162.02	0.1626	261.16	0.1437	353.26	0.0492	418.81	0.0192	447.08	0.0152	443.43	0.0773	518.73	0.0657	467.91	0.0677	462.71
1	0.1797	186.1	0.1630	299.66	0.1437	402.48	0.0515	467.69	0.0197	497.9	0.0152	492.53	0.0780	575.5	0.0656	519	0.0680	513.92
RF	0.1615		0.1543		0.1387		0.0432		0.0204		0.0168		0.0743		0.0640		0.0682	

Table B.7 *Error rate and mean number of trees for OTE by considering different values of* **p** *in comparison with the error rate of Random Forest (RF) for different training set sizes (30%, 60%, 90%) on Eyestate dataset.* **n** *is the training set size. OTE is computed by considering* **t.initial**=*1000 and different values of* **p**. *Random Forest is computed by fixing* **ntrees**=*1000. Each experiment is repeated 100 times. The best results are highlighted in bold.*

p	**Eyestate**					
	n=30%		**n=60%**		**n=90%**	
	error	**mean trees**	**error**	**mean trees**	**error**	**mean trees**
0.1	0.1094	54.01	0.0808	55.62	0.0644	55.6
0.2	0.1064	105.03	0.0782	106.55	0.0628	105.74
0.3	0.1054	154.4	0.0773	156.23	0.0623	155.52
0.4	0.1050	204.75	0.0773	205.52	0.0622	207.76
0.5	0.1048	254.43	0.0771	254.98	**0.0620**	257.26
0.6	**0.1047**	304.52	**0.0768**	304.36	0.0620	302.58
0.7	0.1048	354.09	0.0769	352.65	0.0621	351.5
0.8	0.1048	402.73	0.0774	401.43	0.0622	404.13
0.9	0.1048	452.76	0.0771	450.27	0.0621	446.56
1	0.1048	500.64	0.0774	498	0.0623	493.92
RF	0.1038		0.0763		0.0617	

Table B.8 *Error rate for OTE by considering different values of* **nf** *(number of features) in comparison with the error rate of RF for the training set sizes 90% on all the datasets.* **n** *is the training set size. RF and OTE are computed by considering* **nf**= $\sqrt{q}$, **nf**= $\sqrt{q/2}$, **nf**= $\sqrt{q/3}$ *and* **nf**= $\sqrt{q/4}$. *Each experiment is repeated 100 times. The best results are highlighted in bold.*

Datasets	RF				OTE			
	$\sqrt{q}$	$\sqrt{q/2}$	$\sqrt{q/3}$	$\sqrt{q/4}$	$\sqrt{q}$	$\sqrt{q/2}$	$\sqrt{q/3}$	$\sqrt{q/4}$
Eyestate	**0.0616**	**0.0616**	0.0635	0.0635	**0.0617**	0.0632	0.0659	0.0659
Hepatitis	0.1075	**0.0987**	**0.0987**	0.1025	0.1137	0.1087	0.1087	**0.1075**
Parkinson	**0.0489**	0.0557	0.0557	0.0626	**0.0315**	0.0342	0.0342	0.0394
Body	0.0401	**0.0396**	**0.0396**	0.0419	0.0392	0.0382	0.0382	**0.0378**
Thyriod	0.0099	**0.0097**	0.0103	0.0103	**0.0099**	0.0105	0.0113	0.0113
WDBC	**0.0347**	0.0349	0.0359	0.0359	0.0336	**0.0328**	0.034	0.034
WPBC	**0.2015**	0.2157	0.2189	0.2189	**0.1989**	0.2026	0.2084	0.2084
Ionosphere	**0.068**	0.0694	0.0702	0.0702	**0.0657**	0.0674	0.0674	0.0674
Oil-spill	**0.0365**	0.0384	0.0397	0.0397	**0.0335**	0.0344	0.0354	0.0354
Spambase	0.0477	**0.0471**	0.0474	0.0474	0.0472	**0.0466**	0.0471	0.0471
Glaucoma	0.1195	0.12	**0.119**	0.1235	**0.1055**	0.107	0.1075	0.1125
Nki70	**0.1485**	0.1742	0.1835	0.2	**0.1264**	**0.1264**	**0.1264**	0.1271
Musk	0.1056	**0.105**	0.1066	0.1081	**0.0958**	0.0968	0.1022	0.1008
Breast	**0.37**	0.38	0.3712	0.385	**0.37**	0.3912	0.3825	0.3725
Prostate	**0.075**	0.085	0.09	0.09	**0.06**	0.073	0.081	0.081
Leukaemia	**0.0414**	0.0485	0.0542	0.0571	0.0171	0.0271	0.0171	**0.0128**
Adenocarcinoma	0.1412	**0.1375**	0.14	**0.1375**	**0.1375**	0.1462	0.1475	0.14
Lung	0.0083	0.0066	0.0066	**0.0061**	0.0177	**0.0144**	**0.0144**	0.015
Ovarian	**0.0172**	0.0184	0.0192	0.0188	**0.0076**	0.008	0.0112	0.0116

Table B.9 *Error rate for OTE by considering different values of* **ns** *(node size) in comparison with the error rate of RF for the training set sizes 90% on all the datasets.* ***n*** *is the training set size. RF and OTE are computed by considering* **ns***=1,* **ns***=5,* **ns***=10 and* **ns***=15. Each experiment is repeated 100 times. The best results are highlighted in bold.*

Datasets	RF				OTE			
	ns=1	ns=5	ns=10	ns=15	ns=1	ns=5	ns=10	ns=15
Eyestate	**0.0613**	0.0656	0.0711	0.0766	**0.0617**	0.0664	0.0709	0.0763
Hepatitis	**0.1037**	0.1175	0.1425	0.1412	**0.1137**	0.1275	0.1475	0.1737
Parkinson	**0.0526**	0.0568	0.0568	0.0747	**0.0315**	0.0326	0.0668	0.0863
Body	**0.039**	0.0401	0.0462	0.0509	**0.0392**	0.0403	0.045	0.0486
Thyriod	**0.0098**	0.01	0.0102	0.0101	**0.0099**	0.0104	0.0103	0.0107
WDBC	**0.0345**	0.037	0.0398	0.0407	**0.0336**	0.0352	0.0377	0.0398
WPBC	**0.2063**	0.2136	0.2189	0.2242	**0.1989**	0.2	0.2063	0.2031
Ionosphere	**0.0682**	0.0751	0.0785	0.0788	**0.0657**	0.07	0.0745	0.0788
Oil-spill	**0.0364**	0.0375	0.0397	0.04	**0.0335**	0.0351	0.0358	0.0389
Spambase	**0.0476**	0.0493	0.0509	0.0521	**0.0472**	0.0495	0.0506	0.0509
Glaucoma	0.1205	**0.12**	0.1215	0.121	0.1055	0.1055	**0.105**	0.1115
Nki70	**0.1528**	0.1592	0.1578	0.1507	0.1264	0.1292	**0.125**	0.1257
Musk	**0.1085**	0.1106	0.1175	0.1229	**0.0958**	0.1073	0.1081	0.1102
Breast	**0.37**	0.3712	0.3712	0.3725	0.37	0.3837	0.3612	**0.3587**
Prostate	**0.073**	0.076	0.076	0.076	**0.06**	0.068	0.074	0.072
Leukaemia	0.0385	**0.0357**	0.04	0.06	0.0171	**0.0157**	0.0171	0.0257
Adenocarcinoma	**0.14**	0.1437	0.1475	0.1612	**0.1375**	0.1512	0.155	0.155
Lung	0.0072	0.0066	0.0061	**0.0055**	0.0177	**0.0155**	0.0166	0.0166
Ovarian	**0.0168**	**0.0168**	**0.0168**	0.0176	**0.0076**	**0.0076**	0.0084	0.0088

Table B.10 *Error rate for OTE by considering different values of* **t.initial** *in comparison with the error rate of RF for the training set sizes 90% on all the datasets.* **n** *is the training set size. RF and OTE are computed by considering* **ntrees** *and* **t.initial** *as 500, 1000, 1500 respectively. Each experiment is repeated 100 times. The best results are highlighted in bold.*

Datasets	RF			OTE		
	500	1000	1500	500	1000	1500
Eyestate	0.0616	0.0616	**0.0613**	0.0653	0.0628	**0.0617**
Hepatitis	0.1075	0.105	**0.1037**	**0.1062**	0.1225	0.1137
Parkinson	**0.051**	0.0521	0.0526	0.0331	0.0336	**0.0315**
Body	**0.0378**	0.0384	0.039	0.0398	**0.039**	0.0392
Thyriod	0.0099	**0.0098**	**0.0098**	0.0101	**0.0099**	**0.0099**
WDBC	0.0347	**0.0345**	**0.0345**	**0.0333**	0.0356	0.0336
WPBC	0.2089	**0.2052**	0.2063	0.21	0.2036	**0.1989**
Ionosphere	**0.0665**	0.068	0.0682	0.0668	0.08	**0.0657**
Oil-spill	0.0365	0.0368	**0.0364**	0.0338	**0.0325**	0.0335
Spambase	**0.0474**	0.0476	0.0476	0.0475	0.0473	**0.0472**
Glaucoma	0.1165	**0.11**	0.1205	0.107	**0.097**	0.1055
Nki70	0.1585	0.1535	**0.1528**	0.1285	**0.1264**	**0.1264**
Musk	**0.1066**	0.107	0.1085	0.0993	0.1022	**0.0958**
Breast	0.37	**0.3662**	0.37	0.375	0.375	**0.37**
Prostate	0.075	0.077	**0.073**	0.076	0.071	**0.06**
Leukaemia	0.0414	0.0428	**0.0385**	0.0214	0.0228	**0.0171**
Adenocarcinoma	0.1412	**0.14**	**0.14**	**0.1362**	**0.1362**	0.1375
Lung	0.0077	**0.0066**	0.0072	0.0177	**0.0161**	0.0177
Ovarian	0.0172	**0.0168**	**0.0168**	**0.0072**	**0.0072**	0.0076

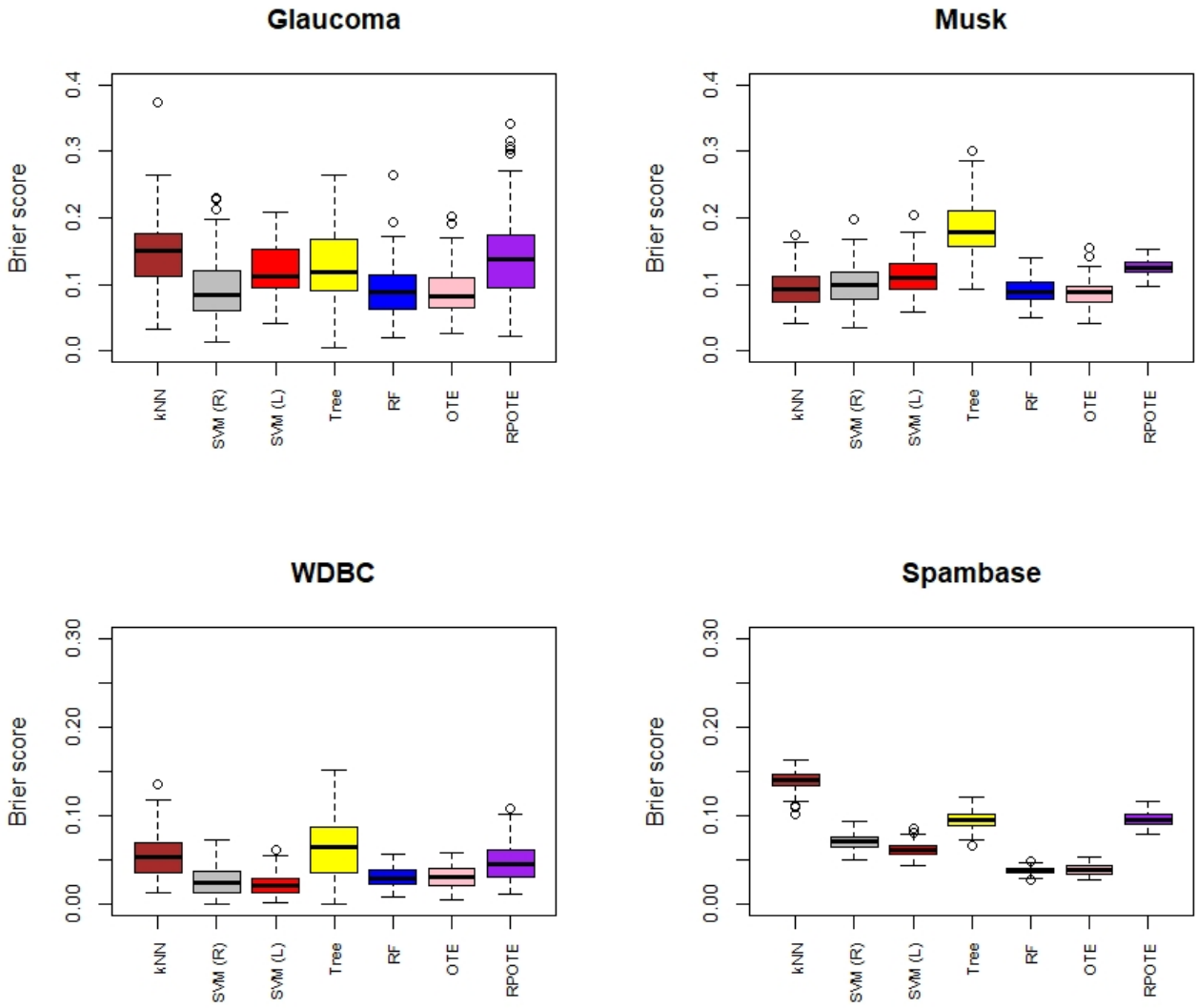


Figure B.1: *Brier Score, of Glaucoma, Musk, WDBC and Spambase for the six classifiers k-NN, SVM (Linear), SVM (Radial), RP (QDA), Tree, RF, OTE and ORPTE.*

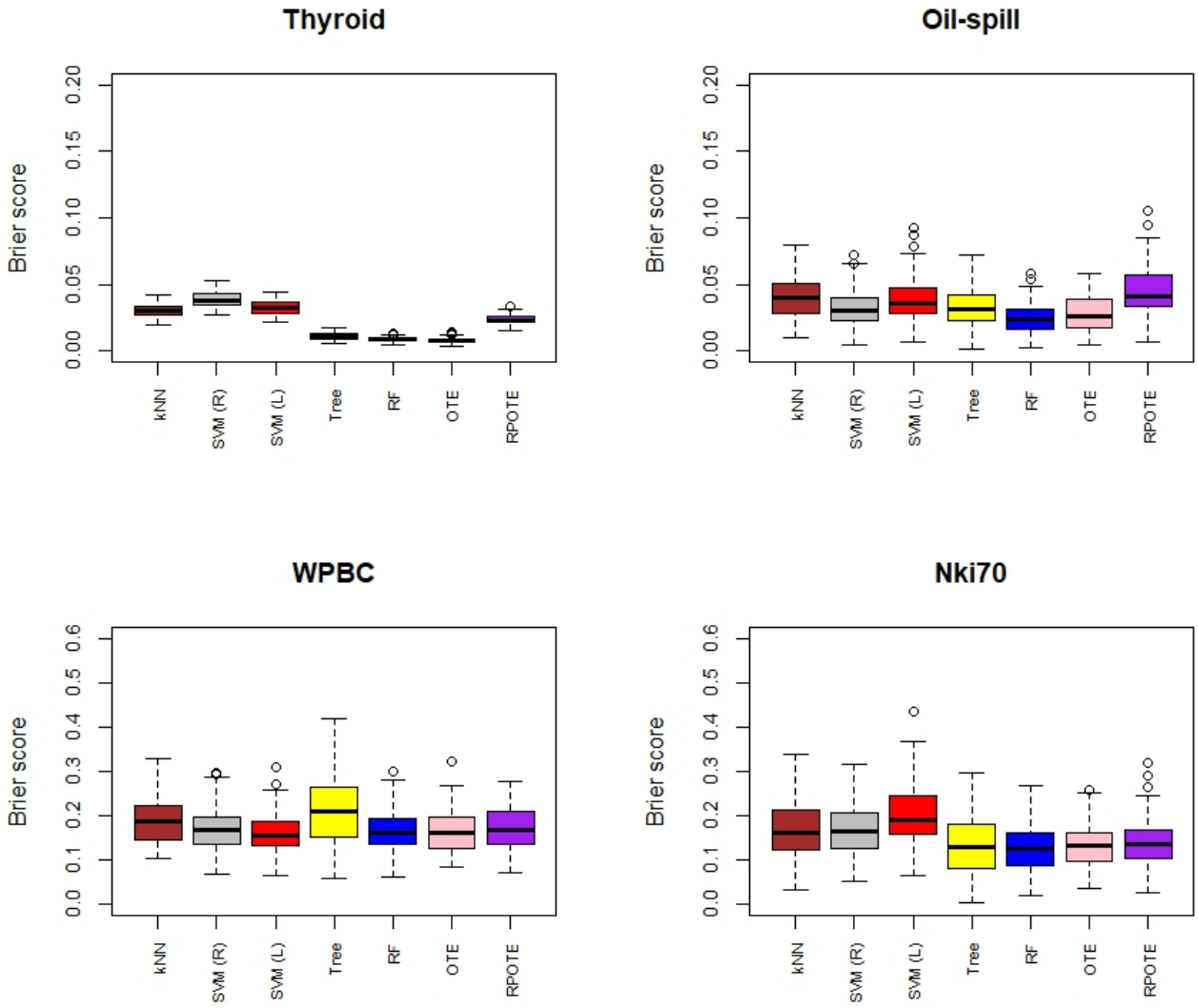


Figure B.2: *Brier Score, of Thyroid, Oil-spill, WPBC and Nki70 for the six classifiers k-NN, SVM (Linear), SVM (Radial), RP (QDA), Tree, RF, OTE and ORPTE.*

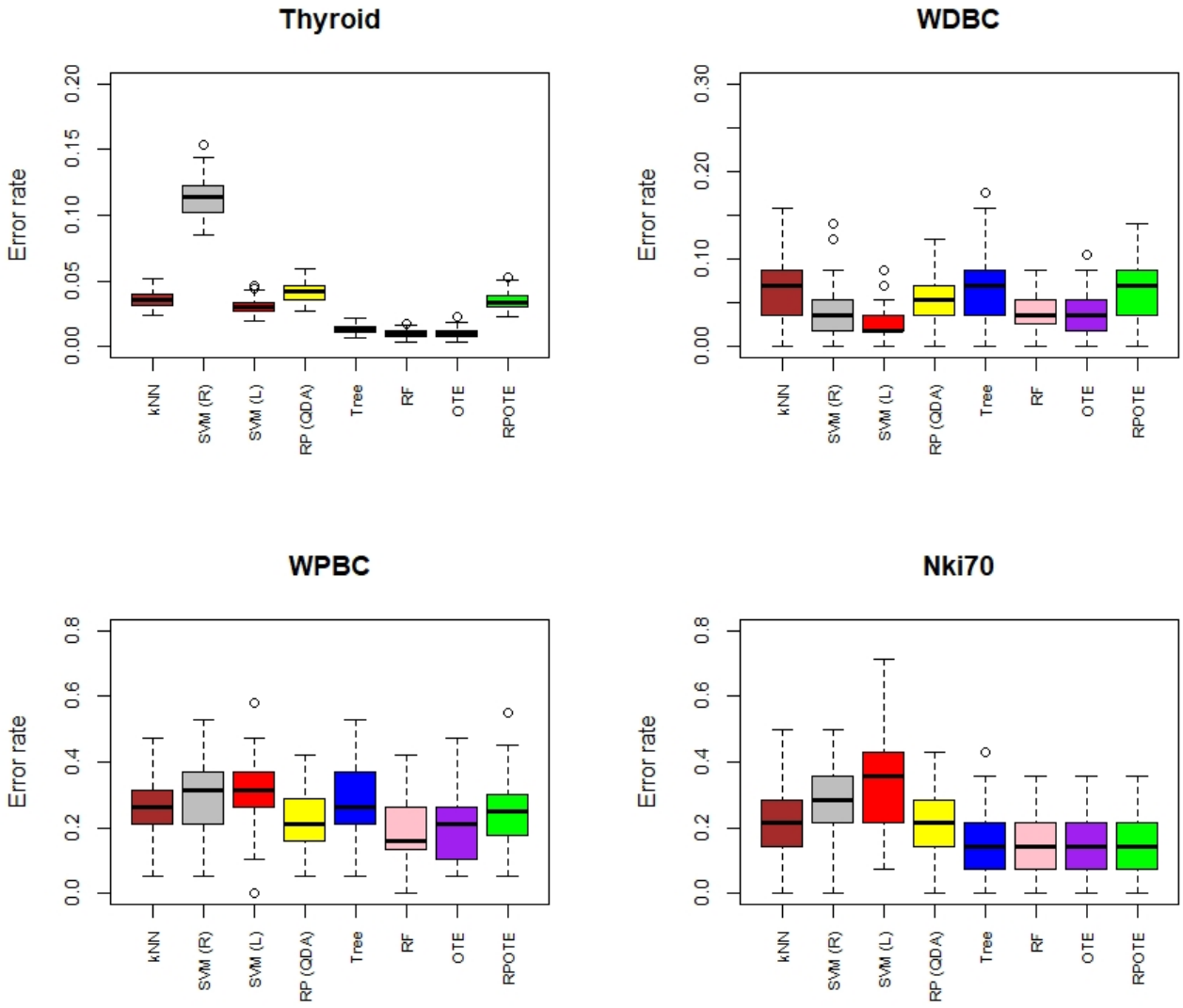


Figure B.3: *Misclassification rate, of Thyroid, WDBC, WPBC and Nki70 for the six classifiers k-NN, SVM (Linear), SVM (Radial), RP (QDA), Tree, RF, OTE and ORPTE.*

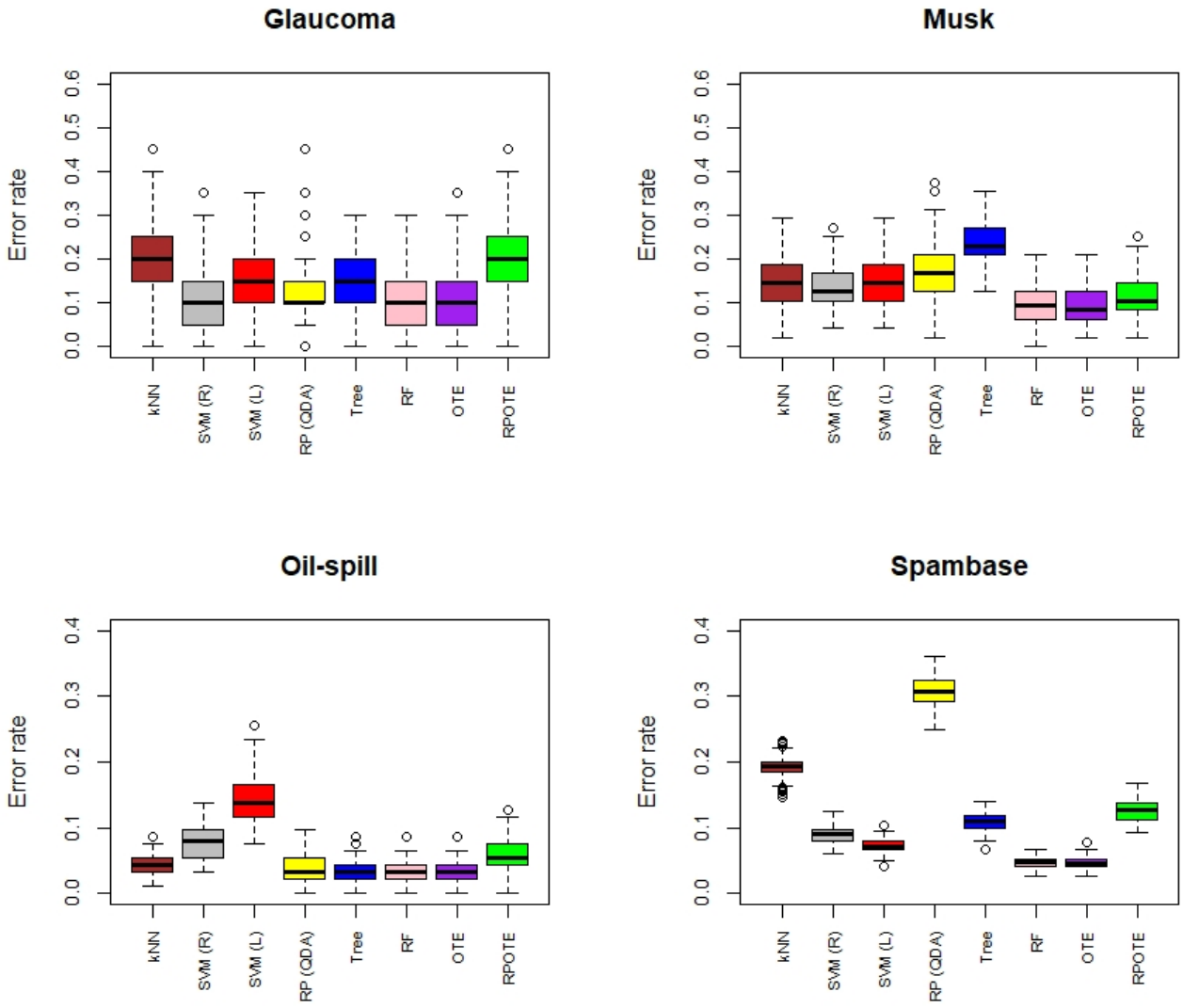


Figure B.4: *Misclassification rate, of Glaucoma, Musk, Oil-spill and Spambase for the six classifiers k-NN, SVM (Linear), SVM (Radial), RP (QDA), Tree, RF, OTE and ORPTE.*

Table B.11 *Selected genes indices by using feature selection methods such as POS, LASSO and Wilcoxon.*

No	POS	Wilcoxon	LASSO	No	POS	Wilcoxon	LASSO	No	POS	Wilcoxon	LASSO	No	POS	Wilcoxon	LASSO
1	X9087	X6459	X6701	26	X2480	X2346	X20112	51	X4238	X17394	X13968	76	X13813	X15752	X13762
2	X11905	X4139	X4518	27	X2262	X1227	X12862	52	X9300	X18081	X15896	77	X8590	X10489	X13554
3	X715	X16290	X4322	28	X2816	X10467	X5821	53	X4275	X20515	X577	78	X13973	X486	X19931
4	X5946	X11723	X5815	29	X2266	X11721	X5523	54	X9526	X15505	X8441	79	X8682	X2077	X16342
5	X4797	X18473	X5933	30	X2843	X13442	X15625	55	X4498	X19104	X8402	80	X14696	X14996	X11465
6	X2915	X14787	X12299	31	X2268	X15856	X8268	56	X10484	X16077	X13518	81	X8762	X19143	X4059
7	X17103	X19600	X17614	32	X2973	X19532	X18692	57	X6055	X1813	X7182	82	X16107	X19942	X4532
8	X845	X17867	X14444	33	X2572	X2942	X4344	58	X11029	X10809	X17867	83	X9243	X11630	X18086
9	X1884	X18660	X2346	34	X3365	X19224	X9543	59	X6412	X19227	X9183	84	X16332	X11974	X20192
10	X11420	X18930	X11435	35	X2599	X19890	X5219	60	X11277	X13328	X4595	85	X9244	X17465	X14407
11	X4698	X13968	X18382	36	X4015	X20101	X4437	61	X6524	X13762	X19756	86	X17302	X6639	X7087
12	X9497	X15666	X17688	37	X2662	X3599	X10621	62	X11416	X10155	X13401	87	X9250	X7468	X5527
13	X6668	X10446	X905	38	X4919	X7813	X7724	63	X6686	X8624	X15503	88	X17540	X9731	X195
14	X175	X10640	X11723	39	X2804	X3657	X10430	64	X11632	X21	X19546	89	X9471	X533	X2942
15	X447	X7087	X16211	40	X6056	X5294	X7010	65	X7064	X686	X14272	90	X17688	X4151	X3466
16	X284	X3420	X11834	41	X2963	X10272	X16284	66	X12414	X8946	X13378	91	X9557	X15361	X16401
17	X844	X1222	X10809	42	X6136	X1223	X15608	67	X7263	X17504	X14943	92	X17897	X5738	X1235
18	X1109	X12818	X3657	43	X3107	X18493	X17062	68	X12789	X19720	X15988	93	X9615	X9381	X9087
19	X1576	X17392	X789	44	X7117	X9104	X9822	69	X7659	X1538	X3296	94	X18614	X15013	X2616
20	X1624	X19818	X19969	45	X3118	X16260	X10467	70	X12980	X11180	X18650	95	X9698	X19223	X7260
21	X1719	X11415	X9104	46	X8072	X4754	X18930	71	X7877	X16005	X10286	96	X19149	X5074	X9909
22	X2052	X10430	X2885	47	X3779	X17495	X18642	72	X13286	X8615	X10649	97	X9762	X19045	X4541
23	X1781	X18133	X4155	48	X9237	X2616	X17807	73	X8037	X14346	X17052	98	X20497	X5354	X13972
24	X2094	X10443	X10694	49	X4045	X15896	X8233	74	X13429	X19205	X7468	99	X9957	X19003	X5983
25	X1882	X8922	X17051	50	X9279	X2965	X1368	75	X8518	X20367	X17216	100	X3668	X2224	X13618

Appendix C

R-Package

Package 'ORPTE'

October 9, 2020

Type Package

Title Optimal Random Projection Trees Ensembles for Classification and Class Membership Probability Estimation

Version 1.0

Date 2020-10-09

Author Nosheen Faiz, Naz Gul, Zardad Khan, Metodi Metodiev, Berthold Lausen

Maintainer Nosheen Faiz <ne16298@essex.ac.uk>

Description Optimal trees based on out-of-bag classification accuracy are selected from an initial pool of trees grown on random projections of the training data. ORPTE function takes training and test data for both predictors and response variables and train the model on the training data and returns predictions on the given test data. Predictions, confusion matrix, accuracy, class membership probability estimates and Brier score are returned for test data.

License GPL (>= 3.6.0)

Imports randomForest,RPEnsemble,stats

R topics documented:

`ORPTE-package` *Optimal Random Projection Trees Ensembles for Classification and Class Membership Probability Estimation*

Description

Optimal trees based on out-of-bag classification accuracy are selected from an initial pool of trees grown on random projections of the training data. ORPTE function takes training and test data for both predictors and response variables and train the model on the training data and returns predictions on the given test data. Predictions, confusion matrix, accuracy, class membership probability estimates and Brier score are returned for test data.

Details

Package:	ORPTE
Type:	Package
Version:	1.0.0
Date:	2020-10-09
License:	GPL-3

Author(s)

Nosheen Faiz, Naz Gul, Zardad Khan, Metodi Metodiev, Berthold Lausen Maintainer: Nosheen Faiz <ne16298@essex.ac.uk>

References

Z. Khan, A. Gul, A. Perperoglou, M. Miftahuddin, O. Mahmoud, W. Adler and B. Lausen, Ensemble of optimal trees, random forest and random projection ensemble classification. *Advances in Data Analysis and Classification*, 14(1), pp.97-116, 2020.

Z. Khan, A. Gul, O. Mahmoud, M. Miftahuddin, A. Perperoglou, W. Adler and B. Lausen, "An ensemble of optimal trees for class membership probability estimation", *In Analysis of large and complex data*, Springer, pp. 395-409, 2016.

T.I. Cannings and R.J. Samworth, Random-projection ensemble classification. *Journal of the Royal Statistical Society Series B*, 79(4), pp.959-1035, 2017.

T.I. Cannings and R.J. Samworth, RPEnsemble: Random projection ensemble classification. R package, v. 0.3. 2016.

Z. Khan, A. Gul, A. Perperoglou, O. Mahmoud, W. Adler, M. Miftahuddin and B. Lausen, OTE: Optimal Trees Ensembles for Regression, Classification and Class Membership Probability Estimation. R package version 1.0.1, 2020.

`Body` *Body Dimensions Relationships Exploration*

Description

The Body data set has a total of 507 instances on 24 features including age, hight, weight and 21 different body dimensions. All the 507 observations are on people, 247 men and 260 women, in the age of twenties and thirties with a relatively small number of old people. The response variable is sex having two categories female and male.

Usage

```
data(Body)
```

Format

A data frame with 507 instances recorded on the following 25 features.

`Biacrom` The diameter of Biacrom taken in centimeter.

`Biiliac` "Pelvic breadth" measured in centimeter.

`Bitro` Bitrochanteric whole diameter measured in centimeter.

`ChestDp` The depth of Chest of a person in centimeter between sternum and spine at nipple level.

`ChestD` The diameter of Chest of a person in centimeter at nipple level.

`ElbowD` The sum of diameters of two Elbows in centimeter.

`WristD` Sum of two Wrists diameters in centimeter.

`KneeD` The sum of the diameters of two Knees in centimeter.

`AnkleD` The sum of the diameters of two Ankles in centimeter.

`ShoulderG` The wideness of shoulder in centimeter.

`ChestG` The circumference of chest centimeter taken at nipple line for males and just above breast tissue for females.

`WaistG` The circumference of Waist in centimeter taken as the average of contracted and relaxed positions at the narrowest part.

`AbdG` Girth of Abdomin in centimeter at umbilicus and iliac crest, where iliac crest is taken as a landmark.

`HipG` Girth of Hip in centimeter at level of bitrochanteric diameter.

`ThighG` Average of left and right Thigh girths in centimeter below gluteal fold.

`BicepG` Average of left and right Bicep girths in centimeter.

`ForearmG` Average of left and right Forearm girths, extended, palm up.

`KneeG` Average of left and right Knees girths over patella, slightly flexed position.

`CalfG` Average of right and left Calf maximum girths.

`AnkleG` Average of right and left Ankle minimum girths.

`WristG` Average of left and right minimum circumferences of Wrists.

`Age` Age in years

`Weight` Weight in kilogram

`Height` Height in centimeter

`Gender` Binary response with two categories; 1 - male, 0 - female

Source

G. Heinz, L.J. Peterson, R.W. Johnson, and C.J. Kerk, “Exploring Relationships in Body Dimensions”, *Journal of Statistics Education*, 11, 2003.

References

C. Hurley, “ gclus: Clustering Graphics”, R package version 1.3.1, 2002, *https://CRAN.R-project.org/package=gclus*.

Examples

```
data(Body)
str(Body)
```

ORPTE *Optimal random projection trees ensemble*

Description

Optimal trees based on out-of-bag classification accuracy are selected from an initial pool of trees grown on random projections of the training data. ORPTE function takes training and test data for both predictors and response variables and train the model on the training data and returns predictions on the given test data.

Usage

```
ORPTE(xtrain, ytrain, xtest, ytest, d2 = 5, nodesize = NULL,
fset = NULL, initial.size = NULL, p = 0.2, proj.method = "Gaussian",
type = "prob", process.info = TRUE)
```

Arguments

`xtrain`	`xtrain` is the training part of the given data consisting of predictors values.
`ytrain`	`ytrain` is the training part of the given data consisting of response values. This is vector of binary values.
`xtest`	`xtest` is the testing part of the given data consisting predictors values only.
`ytest`	The observed test response values for assessment purposes. This is a vector of binary values.
`d2`	Dimension of projected data. This could be equal to or less than number of features in the original dataset.
`nodesize`	Number of samples in final nodes.
`fset`	Number of features selected randomly for splitting of nodes of the trees.
`initial.size`	Total number of trees to be grown initially for optimal trees ensemble.
`p`	The proportion of trees to be selected from the total `initial.size` trees. Top ranked proportion based on their out-of-bag accuracy will be selected. Its default value is `p=0.2` i.e. 20% of trees are selected by default. Tuning this value, by cross validation, for example, might improve the performance of the method.
`proj.method`	Method used for random projection generation. Currently three method are supported i.e. `"Gaussain"`,`"Haar"` and `"axis"`.
`type`	Type of the ensemble grown i.e. classification or probability estimation random projection optimal trees ensemble.
`process.info`	Boolean. Set to `True` for process informtion display.

Details

ORPTE reduces data dimesion by randomly projecting the original training data and growing classification and probability estimation trees on the projected data with reduced dimension. The method then discards those trees that proform poorly on the training data and combines the rest for final ensemble. The parameters of the method, i.e. `initial.size`, `p`, `d2`, `fset` and `nodesize` are data driven and need to be tuned using cross validation, for example.

Value

`Predictions`	Predictions made for `ytest` if `type="class"`
`Accuracy`	Classification accuracy for `ytest` if `type="class"`
`Confusion Matrix`	Confusion matrix constructed by cross tabulating `ytest` and `ytrain` if `type="class"`
`Probabilities`	Class membership probability estimates for `ytest` if `type="prob"`
`BS`	Brier score estimate based on `ytest` if `type="prob"`

Author(s)

Nosheen Faiz, Naz Gul, Zardad Khan, Metodi Metodiev, Berthold Lausen

References

Z. Khan, A. Gul, A. Perperoglou, M. Miftahuddin, O. Mahmoud, W. Adler and B. Lausen, Ensemble of optimal trees, random forest and random projection ensemble classification. *Advances in Data Analysis and Classification*, 14(1), pp.97-116, 2020.

Z. Khan, A. Gul, O. Mahmoud, M. Miftahuddin, A. Perperoglou, W. Adler and B. Lausen, "An ensemble of optimal trees for class membership probability estimation", *In Analysis of large and complex data*, Springer, pp. 395-409, 2016.

T.I. Cannings and R.J. Samworth, Random-projection ensemble classification. *Journal of the Royal Statistical Society Series B*, 79(4), pp.959-1035, 2017.

T.I. Cannings and R.J. Samworth, RPEnsemble: Random projection ensemble classification. R package, v. 0.3. 2016.

Z. Khan, A. Gul, A. Perperoglou, O. Mahmoud, W. Adler, M. Miftahuddin and B. Lausen, OTE: Optimal Trees Ensembles for Regression, Classification and Class Membership Probability Estimation. R package version 1.0.1, 2020.

See Also

`Body`

Examples

```
# Example

data(Body)
data <- Body
n <- nrow(data)
set.seed(999)
training <- sample(1:n,0.7*n)
testing <-(1:n)[-training]
xtrain <- as.matrix(data[training,-c(25)])
dim(xtrain)
class(xtrain)
ytrain <-  data[training,25]
xtest <- data[testing,-c(25)]
ytest <- data[testing,25]

orpte <- ORPTE(xtrain = xtrain,ytrain = ytrain,xtest = xtest,ytest = ytest,
               type="class",initial.size = 100)
orpte
```

```
### Output


# $Predictions
# [1] 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
# [40] 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
# [79] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
# [118] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0



# $Accuracy
# [1] 0.9477124

# $`Confusion Matrix`
# ytest
# Yhh  0  1
# 0 75  4
# 1  4 70

orpte <- ORPTE(xtrain = xtrain,ytrain = ytrain,xtest = xtest,ytest = ytest,
               type="prob",initial.size = 100)

orpte

### Output

# $Probabilities
# [1] 0.70 0.95 1.00 0.95 0.90 1.00 1.00 1.00 0.25 1.00 1.00 0.85 1.00 0.55 1.00
# [16] 0.95 0.85 1.00 0.90 0.70 0.90 1.00 1.00 0.95 1.00 1.00 1.00 0.35 0.75 1.00
# [31] 0.55 1.00 0.95 1.00 1.00 1.00 0.95 1.00 0.90 0.95 0.95 1.00 1.00 0.95 0.95
# [46] 0.60 1.00 0.70 1.00 1.00 0.40 1.00 1.00 1.00 1.00 0.85 0.95 0.95 0.95 1.00
# [61] 1.00 0.85 1.00 0.60 1.00 1.00 1.00 1.00 0.65 1.00 0.85 1.00 0.95 1.00 0.00
# [76] 0.00 0.00 0.00 0.00 0.00 0.00 0.15 0.25 0.00 0.70 0.05 0.35 0.00 0.05 0.00
# [91] 0.00 0.00 0.00 0.05 0.00 0.05 0.05 0.10 0.55 0.20 0.55 0.00 0.25 0.00 0.00
# [106] 0.00 0.10 0.15 0.75 0.05 0.10 0.20 0.20 0.05 0.00 0.05 0.05 0.05 0.00 0.05
# [121] 0.15 0.15 0.05 0.00 0.05 0.00 0.00 0.15 0.05 0.05 0.00 0.15 0.00 0.00 0.00
# [136] 0.05 0.05 0.00 0.00 0.00 0.00 0.30 0.05 0.00 0.00 0.00 0.00 0.00 0.00 0.10
# [151] 0.15 0.05 0.00


# $BS
# [1] 0.03315359
```

Index