Stony Brook University

ROC Random Forest and Its Application

A Dissertation Presented

by

Bowen Song

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

January 2015

Stony Brook University

The Graduate School

Bowen Song

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Wei Zhu – Dissertation Advisor
Professor, Deputy Chair, Department of Applied Mathematics and Statistics,
Stony Brook University

Song Wu – Chairperson of Defense
Assistant Professor, Department of Applied Mathematics and Statistics,
Stony Brook University

Yi Gao – Inside Member
Adjunct Professor, Department of Applied Mathematics and Statistics;
Assistant Professor, Department of Biomedical Informatics,
Stony Brook University

Ellen Li – Outside Member
Professor, Department of Internal Medicine;
Devision Chief, Gastroenterology and Hepatology,
Stony Brook University

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

ROC Random Forest and Its Applications

by

Bowen Song

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2015

Classification algorithms that optimize the overall accuracy or class distribution purity often suffer from difficulties in classifying class imbalanced data, in which most cases in the testing set will be classified to the majority class. However for imbalanced data classification, one usually cares more about the accuracy for identifying the minority class (e.g. diseased samples), that is, the sensitivity, other than the overall accuracy and therefore low sensitivity is highly undesirable.

Receiver operating characteristic (ROC) is a 2 dimensional graph by plotting sensitivity versus specificity, i.e., accuracy in identifying the majority class (e.g. normal samples). A curve is formed by varying the decision threshold and the area under ROC (AUC) is employed as an accuracy measurement to evaluate the performance of classification. Random Forest, a modern ensemble classifier, is gaining increasing attention in the community because of its good classification capability. Each single learner is a decision tree, built on a bagging data with each node split based on a randomly selected feature subset. As a result, each base learner is relatively "independent" to the others and thus the ensemble's classification accuracy improves overall.

In this dissertation, we combine the ROC analysis and the Random Forest to establish the proposed ROC Random Forest algorithm. There are two goals to this algorithm: (1) improving the AUC value, and (2) producing balanced classification result. Verification was carried out using 18 public data sets from the UCI and the results show that the ROC Random Forest not only improves the classification accuracy in terms of higher AUC value but also delivers a more balanced classification result comparing to other Random Forest settings. One draw-back of the ROC Random Forest lies in its difficulty in processing categorical predictors. Given the importance of categorical predictors in many classification problems, we have further combined the ROC Random Forest with optimal node splitting algorithms other than ROC for categorical predictors. The resulting Hybrid ROC Random Forest is further evaluated on 8 UCI data sets.

Acknowledgments

First and foremost, I would like to sincerely thank my advisor Prof. Wei Zhu for her tremendous help, guidance, for the freedom and great support I was granted throughout these years.

I am also very grateful to all members of the committee, Prof. Ellen Li, Prof. Song Wu and Prof. Yi Gao, in this work and for taking the time to evaluate this dissertation and provide insightful advices.

I also want to extend my thanks to Prof. Jerome Liang for his support during my research. Part of the work was supported by the NIH/NCI under Grants #CA082402 and #CA143111.

I also want to take this opportunity to thank my friends and colleagues who provide me with fun, courage, discussion and pleasant and stimulating working environment during my PhD pursuing life: Muqi, Guoli, Tengjie, Shuai, Yan, Sam, Hongyan, Xuebing, Tian, Shaonan, Huan, Hao Z., Hao H., Yan L., Ming, Lin, Sarah.

Last but not least, Helen and my parents, I am forever grateful for your unconditional support and love.

Table of Contents

List of Tables

List of Figures

Chapter I: Introduction

1.1 Classification

Statistical learning is a framework for machine learning drawing from the fields of statistics and functional analysis (Mohri et al., 2012). It has been gaining increasing attention and playing a key role in many areas of science as well as finance and economics (Hastie et al., 2009). Machine learning algorithms mainly fall into the following three categories: supervised learning, unsupervised learning and reinforcement learning. In this dissertation, we mainly focus on the area of supervised learning.

The goal of learning is prediction. Supervised learning, as its name suggests, represents those learning algorithms whose learning processes are exposed and therefore guided with the presence of the target variable, which we wish to predict. By the nature of the target variable, supervised learning is further divided into (1) classification for discrete target variable, and (2) regression for continuous target variable. In this dissertation, we will focus on the classification algorithms.

Classification is the task of learning a target function which could be used to identify to which of a set of categories (sub-populations) a new observation belongs. To be specific, the learning process involves learning a mapping function or model between the input observations, denote as $\boldsymbol{X}$ and its corresponding output categories, denote as $\boldsymbol{\omega}$. The output model optimizes a predefined goal function of the predicted and true value of target variable, e.g., error rate, on the basis of a training set of data containing observations with known category membership (the reason why we call it supervised

learning). Such systematic approaches employed to build classification models are called classifiers or classification techniques. Classifier is most suited for predicting or describing data sets with binary or nominal categories. It is less effective for ordinal categories, e.g., to predict tomorrow's temperature to be hot, mild or cool, because they do not consider the implicit order among the categories (Frank and Hall, 2001). Other forms of relationships, such as the subclass-superclass relationships among categories, for example, humans and apes are primates, which in turn, is a subclass of mammals, are not considered.

Various tools and algorithms are employed to perform the classification task. Based on whether or not assumptions are made on the data, these algorithms could be further divided to two major sub-groups, i.e., parametric classification algorithms and nonparametric classification algorithms. Due to the complexity of a problem, there is no superior approach that always performs the best. In parametric methods, linear discriminant analysis and logistic regression are classical and standard methods, they assume Gaussian distribution and binomial distribution of the data point respectively. Moreover, some modern techniques have been introduced, e.g., Naive Bayesian method, in which conditional independence assumption is made on the attributes variables. Nonparametric methods have no such assumptions which imply the decision boundaries could be of any arbitrary geometry (Hubert et al., 2001). Nearest neighbor based algorithms belongs to this categories. Decision tree algorithms are also included in this group as well as the neural network based algorithms and the well-known support vector machine.

If we want to summarize classification algorithms in terms of their algorithm structure, classification algorithms fall into the following groups, namely, single classifiers and ensemble classifiers. As the name suggests, a single classifier represents a standalone classification algorithm while an ensemble classifier is a combination of single classifiers. Ensemble classifier could be better considered as a higher level classifier combination strategy rather than a classification algorithm and its goal is to improve the ensemble classification performance by properly combining the single learners, i.e., single classifiers. Follow this categorization we review several fundamental single classifiers and ensemble classifiers in the following sections.

1.2 Single classifier

1.2.1 K-nearest neighbor classifier

The k-nearest neighbor algorithm (kNN) is a very intuitive method that classifies unlabeled instances based on their similarity to examples in the training set. In simple word, for a given unlabeled example $X^* \in \Re^p$, we first find the k "closest" labeled examples in the training data set and assign X^* to the class that appears most frequently within the k closest neighbors.

kNN classifier is closely related to kNN density estimation. And we briefly introduce its model setting here. Assume we have a dataset of N samples, of which N_i are from class ω_i, and we are interested in predict the label of an unknown sample X^*. First we draw a hyper-sphere of volume V around X^*. Let's further assume the volume contains a total of k examples, of which k_i are from class ω_i.

The likelihood functions using kNN (Bishop, 2006) could be approximated by

$$p(X|\omega_i) \cong \frac{k_i}{N_i V}$$

Similarly, the unconditional density is estimated by

$$p(X) \cong \frac{k}{NV}$$

And the priors are approximated by

$$p(\omega_i) \cong \frac{N_i}{N}$$

If we put everything together, the Bayes classifier becomes

$$p(\omega_i|X) = \frac{p(X|\omega_i)}{p(X)} = \frac{\frac{k_i}{N_i V} \cdot \frac{N_i}{N}}{\frac{k}{NV}} = \frac{k_i}{k}$$

And the outputting ω_i should maximize $p(\omega_i|X)$. Just as our introduction in the very beginning, the right-most term in the above equation indicates that any unknown instance will be classified to the most frequently appeared class in its k nearest neighbors. An example could be shown in **Figure 1**.

"Nearest" is defined by any similarity measures, like Euclidean distance and the Mahalanobis distance. The difference between the two types of distances is that the Mahalanobis distance considers the correlations of the dataset and is scale-invariant. Nearest neighbor problem has been extensively studied in the field of computational geometry under the name closest pair of points problem, which is one of the oldest problems in computational geometry (Shamos et al., 1975).

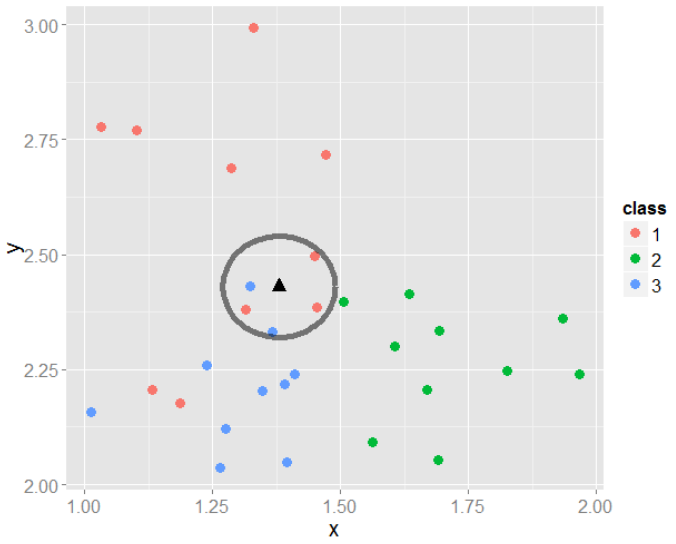


Figure 1 Illustration of classification of nearest neighbor method. The red, blue and green points in the 2 dimensional space belongs to class 1, 2 and 3 respectively. The newly input point's label, black solid triangle, will be classified based on 5 nearest neighbor. The gray circle indicates the 5 nearest neighbor of new input point, which includes 3 points of class 1 and two points of class 3. As a result, this new point will be classified as class 1.

kNN is considered as a lazy learning algorithm, where the function is only approximated locally and all computation is deferred until classification. Comparing to the eager learning algorithms, like Logistic Regression and Random Forest, which compiles data into a compressed description/model and classifies incoming patterns using the induced model, lazy learning algorithms have fewer computational costs in the training phase, however, they require greater storage space of data points and have higher computational costs on recall in the testing phase.

1.2.2 Decision tree classifier

Decision tree algorithm is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input

variables. There are two main types: classification tree which predict discrete outcomes and regression tree which predict continuous outcomes. The term Classification and Regression Tree (CART) analysis is an umbrella term used to refer to both procedures (Breiman et al., 1984). In this dissertation, we only focus on the classification part.

Decision tree is a tree-like structure which contains three components: internal (non-leaf) node, branch and leaf (terminal) node. Decision tree learning is the process of constructing a decision tree from class-labeled training tuples. Each internal node denotes a test on an attribute, or feature. Each branch represents the outcome of the test and each leaf node holds a class label.

Before we talk about the algorithm for building decision tree, let us introduce the node splitting criteria first. The basic idea here is very simple, for each node, by optimizing a cost function, we identify a feature and a threshold corresponding to the feature. In the testing phase, any observation with a value of that feature larger than the threshold falls into the right child node, and if its value is smaller than the threshold the observation will fall into the left child node. Traditionally, Gini impurity and Entropy are employed here to choose the "best splitting" feature and corresponding threshold. Their definitions are given below:

$$I_G(f) = -\sum_{i=1}^{m} f_i(1 - f_i) = 1 - \sum_{i=1}^{m} f_i^2$$

$$I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i$$

For binary class, there corresponding plot is shown in **Figure 2**, both methods provide a measure of the homogeneity of the target variable. In Figure 5, f stands for the faction of class "1" observations. As f approaches 0 or 1, both Gini impurity and Entropy will approaches 0, which indicates that observations are homogeneous and tends to from same class. And therefore, the feature and threshold which will generate the largest drop of these values between parent node and child nodes will be selected to splitting this node. The difference is called information gain, and we are trying to maximize this information gain.

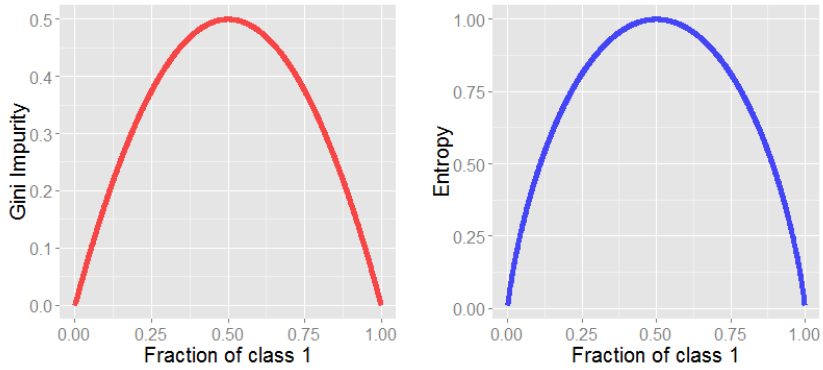


Figure 2 Curves of Gini Impurity, left figure and Entropy, right figure. Note that the domain of Gini Impurity and Entropy are different.

Here comes the Decision tree pseudo-code, which includes two parts: splitting_attribute_threshold and decision_tree.

Algorithm 1 splitting_attribute_threshold

Inputs(s): The matrix of training examples $\boldsymbol{X}$ and the corresponding label vector $\boldsymbol{\omega}$
Output(s): Selected attribute $\mathcal{A}$ and its corresponding splitting threshold θ
Require: Homogeneity measure H

set $\theta \leftarrow -\infty$
set $\mathcal{A} = \mathcal{A}_1$
set $max\_gain \leftarrow 0$
for each attribute $\mathcal{A}_i \in X$ **do**
    for each possible threshold $\theta_j^{\mathcal{A}_i} \in \mathcal{A}_i$ **do**
        set $temp\_gain \leftarrow IG(root_{node}, left_{node}, right_{node}, \boldsymbol{\omega})$
        if $temp\_gain >$ max_gain **then**
            set $\theta \leftarrow \theta_j^{\mathcal{A}_i}$
            set $\mathcal{A} \leftarrow \mathcal{A}_i$
            set $max\_gain \leftarrow temp\_gain$
        end if
    end for
end for
return θ and $\mathcal{A}$
end

Algorithm 2 decision_tree

Input(s): The matrix of training examples $\boldsymbol{X}$ and the corresponding label vector $\boldsymbol{\omega}$
Output(s): Decision tree T

if $\boldsymbol{X} == \boldsymbol{\phi}$ **then**
    return a single node with $\boldsymbol{\phi}$
end if
if $\boldsymbol{\omega}$ consists records all with the same value for the class label **then**
    return a single leaf node with that value
end if
set $(\theta, \mathcal{A}) \leftarrow$ splitting_attribute_threshold$(\boldsymbol{X}, \boldsymbol{\omega})$
set $(\boldsymbol{X}_{left}, \boldsymbol{\omega}_{left})$ and $(\boldsymbol{X}_{right}, \boldsymbol{\omega}_{right})$ as the subsets of $(\boldsymbol{X}, \boldsymbol{\omega})$ consisting of observations respectively with value greater than or equal to and less than θ for attribute $\mathcal{A}$
recursively apply decision_tree to subset $(\boldsymbol{X}_{left}, \boldsymbol{\omega}_{left})$ and $(\boldsymbol{X}_{right}, \boldsymbol{\omega}_{right})$ until they are empty or the stopping criteria are met
return a tree with root or node labelled $(\theta, \mathcal{A})$ and child node decision_tree$(\boldsymbol{X}_{left}, \boldsymbol{\omega}_{left})$ and decision_tree$(\boldsymbol{X}_{right}, \boldsymbol{\omega}_{right})$
end

Algorithm 1 and **Algorithm 2** are two very basic component to build a tree, and based on them, variation decision tree structures have been introduced. Hunt's algorithm (Hunt et al., 1966) is one of the earliest decision tree algorithms. Quinlan (1986) invented ID3 (Iterative Dichotomiser 3) which is the precursor to the C4.5 algorithm, which was also developed by Quinlan (1996). It made a number of improvements to ID3. C4.5 handles both continuous and discrete attributes. It could also handle training data with missing attribute values. Both ID3 and C4.5 employ entropy method to calculate the gain. CART (Breiman et al., 1984) is also widely used tree algorithm. Unlike ID3 and C4.5, it uses Gini Impurity to measure the homogeneity.

To prevent over fitting, a pruning process, called minimal cost-complexity pruning, is performed in many methods, like CART. The purpose of this step is to build a right sized tree by estimating the true misclassification cost. Take CART for example, first, CART builds a full grown tree and then cuts the pair of leaves sequentially. In each sub-tree, misclassification cost and cost-complexity value are calculated using 10-fold cross-validation. Finally, the CART algorithm chooses the final optimal tree using these values.

Decision tree has several advantages over other classification methods and that is why it is so common and popular. First it is simple to understand and interpret. Secondly, it requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Thirdly, it uses a white box model. It has clear and explicit classification model. We can see how the variables are associated with the response in the tree structure. Lastly, as non-parametric classifier, it is robust. Not only serve as a single classifier, decision tree

is also a common choose of base classifier of popular ensemble classifiers, e.g., Random Forest, which we will introduce in §**1.3**.

1.2.3 Support vector machine

Support vector machine (SVM) was first introduced by Vladimir Vapnik in 1995. A detailed introduction could be found in Burges's paper (Burges, 1998). And here we briefly introduce its math model of linearly separable setting:

Give training data $\mathcal{D}$, a set of n points of the form

$$\mathcal{D} = \left\{(\boldsymbol{x_i}, \omega_i)\middle|\boldsymbol{x_i} \in \mathbb{R}^p, \omega_i \in \{-1,1\}\right\}, i = 1, \dots, n$$

where the ω_i is either 1 or -1, indicating the label of observation $\boldsymbol{x_i}$ belongs. Each $\boldsymbol{x_i}$ is a *p*-dimensional feature vector.

We want to find a hyperplane which maximize the margin between the points having $\omega_i = 1$ and those having $\omega_i = -1$. Any hyperplane can be written as the set of points $\boldsymbol{x}$ satisfying

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = 0$$

where $\cdot$ denotes the dot product and $\boldsymbol{w}$ the normal vector to the hyperplane. Then $\frac{b}{\|\boldsymbol{w}\|}$ determines the distance from the hyperplane to origin.

Since the data are linearly separable (our assumption), we could select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. Margin is defined as the region bounded by them. These hyperplanes could be described by the equations:

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = 1 \text{ and } \boldsymbol{w} \cdot \boldsymbol{x} - b = -1$$

It could be easily shown that the distance between these two hyperplanes is $\frac{2}{\|\boldsymbol{w}\|}$, so we want to minimize $\|\boldsymbol{w}\|$ (**Figure 3**).

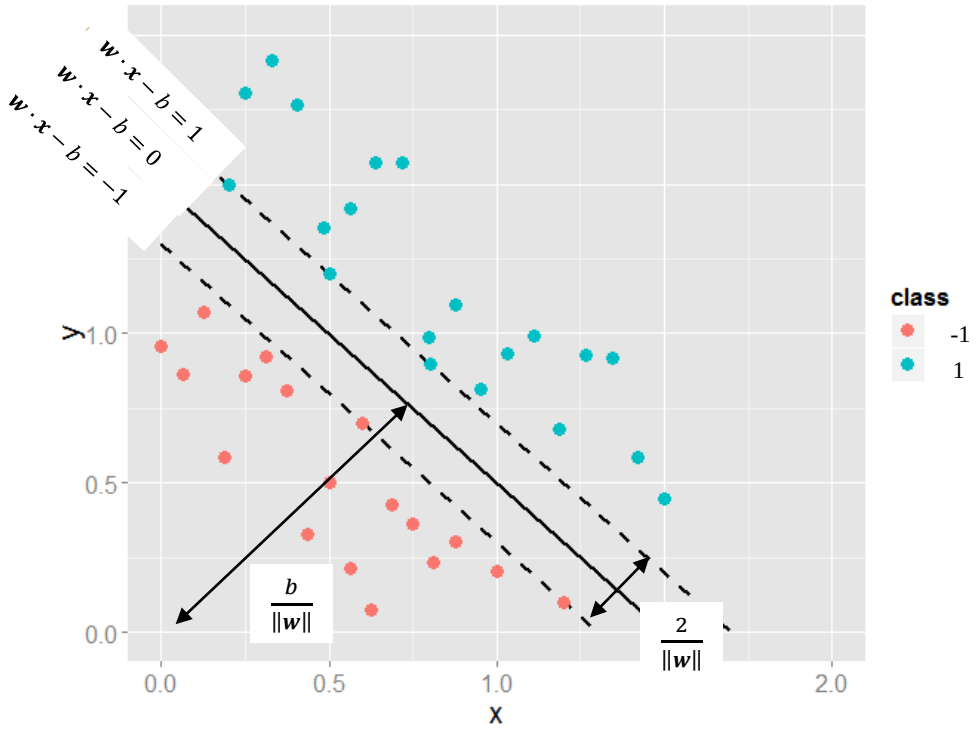


Figure 3 Illustration of SVM classification boundary. Red and blue solid circle points represent training point belonging to different classes. The two dashed line indicate the boundary of maximum margin, while the solid line indicates the classification boundary.

To prevent data point from falling into the margin, we add the following constraint:

$$\boldsymbol{w} \cdot \boldsymbol{x_i} - b \geq 1 \text{ for } \boldsymbol{x_i} \text{ having label "1"}$$

$$\boldsymbol{w} \cdot \boldsymbol{x_i} - b \leq -1 \text{ for } \boldsymbol{x_i} \text{ having label "-1"}$$

The above inequations could be further reduced to

$$\omega_i(\boldsymbol{w} \cdot \boldsymbol{x_i} - b) \geq 1 \text{ for all } 1 \leq i \leq n$$

To summarize, our optimization problems become (note that minimizing $\|\boldsymbol{w}\|$ mathematically equals minimizing $\frac{1}{2}\|\boldsymbol{w}\|^2$)

$$\arg\min_{\mathbf{w},\mathrm{b}} \max_{\alpha \geq 0} \{\frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=0}^{n} \alpha_i[\omega_i(\boldsymbol{w} \cdot \boldsymbol{x_i} - b) - 1]\}$$

The above problem can be solved by standard quadratic programming techniques and programs, e.g., Karush-Kuhn-Tucker condition (Fletcher, 1987).

For linearly un-separable problems, a slack variable is introduced to the optimization function, and it becomes

$$\arg\min_{\mathbf{w},\xi,\mathrm{b}} \max_{\alpha \geq 0, \beta \geq 0} \{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=0}^{n} \xi_i - \sum_{i=0}^{n} \alpha_i[\omega_i(\boldsymbol{w} \cdot \boldsymbol{x_i} - b) - 1 + \xi_i] - \sum_{i=0}^{n} \beta_i \xi_i\}$$

The slack variable ξ here measures our tolerance of the misclassification of the model.

For non-linearly classification problem, kernel trick (Aizerman, 1964) is applied, the basic idea of which is to map the input data to higher dimension or infinite dimension feature space, in which the problem could be considered as linearly separable and solved. Popular kernel functions include linear kernel, Gaussian radial basis function kernel (RBF) and polynomial kernel.

SVM has been widely applied in sciences nowadays and its robust and well performance make it the status quo classifier and therefore it usually serves as a benchmark in many classification comparison papers.

1.3 Ensemble Classifier

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms (Opitz,1999 and Polikar, 2006). A motivation for ensembles is that a combination of the outputs of many weak classifiers produces a powerful committee (Hastie et al., 2009).

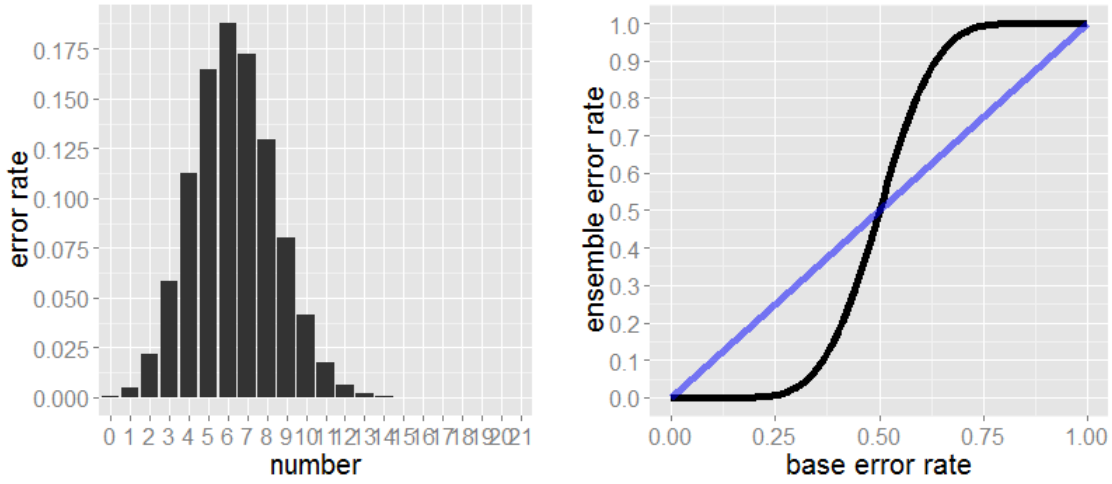


Figure 4 Ensemble classifier error rate. The left figure indicates how the error rate changes with the number of base classifiers, which make correct prediction, when the error rate of base classifier is 0.3. The right figure indicates how the ensemble error rate changes with the base classifier's error rate when majority of the base classifier make correct prediction.

We borrow the example from Dietterich's paper (Dietterich, 2000) to describe how this ensemble structure works. Consider an ensemble of 21 binary classifiers, each of which has an error rate of $\epsilon = 0.3$. The ensemble classifier predicts the class label of a test example by taking a majority vote on the predictions made by the base classifier. If the base classifiers are identical, the resulting ensemble classifier will also have an error rate of 0.3. On the other hand, if the base classifier are independent, say they are uncorrelated, then then ensemble makes a wrong decision only if more than half of the base classifiers predict incorrectly, in which case the error rate could be calculated using the following equation and its correspond plot is shown in **Figure 7**

$$Ensemble\_Error\_Rate = \sum_{i=11}^{21} \binom{21}{i} \epsilon^i (1-\epsilon)^{21-i} = 0.0264$$

The probability of the ensemble making and decision error is 0.0264. With the 21 base classifiers fixed, we could also show how the ensemble prediction errors change with the base classifier error rate changing. The diagonal line (blue curve) indicates when all the base classifiers are identical while the black curve shows all the base classifier are uncorrelated. The ensemble classifier performs worse than the base classifiers when ϵ is larger than 0.5.

The preceding example illustrates two conditions for an ensemble classifier in order to perform better than the base classifier: (1) The base classifiers need to have low correlations from each other, detailed discussions could be found in these papers (Williams, 1975; Beriman, 2001; Ahn et al., 2007), and (2) the base classifiers should do

better than a classifier that performs random guess, in other word, the error rate of the base classifier should be less than 0.5 (Hansen et al., 1990).

As introduced in Duin's paper (Duin et al., 2000), common combining classifiers could be distinguished into the following 3 groups: 1. Parallel combining of classifiers computed for different feature sets; 2. Stacked combining of different classifiers computed for the same feature space; 3. Combining weak classifiers, in which case large sets of simple classifiers are trained on modified version of the original dataset. We focus on the third part and introduce two popular ensemble classifiers, i.e., adaptive boosting and random forest in the following sections.

1.3.1 Random forest

Bootstrap aggregating (Breiman, 1996), often abbreviated as bagging, involves having each model in the ensemble vote with equal weight. Random forest (Breiman, 2001) is a popular bagging algorithm which combines random decision trees with bagging to achieve very high classification accuracy. We briefly introduce the math setting of random forest in this session.

Firstly, let's introduce its precursor, tree bagging. Considering a training set (x_i, ω_i) for $i = 1,2, \dots, N$, where x_i is a p dimensional vector and ω_i indicates the target label of x_i. Tree bagging repeatedly selects a bootstrap sample of the training set and then fits trees to these samples. Detailed algorithms is shown below

Algorithm 3 tree_bagging

Input(s): Training set (x_i, ω_i) for $i = 1,2, \dots, N$; number of trees B
Output: Tree classifier $T_1, T_2, \dots, T_B$
1. **for** $b = 1$ **to** B **do**
2. Build a dataset S_b, by sampling N items randomly with replacement from the original data pool (x_i, ω_i) for $i = 1,2, \dots, N$.
3. Train decision tree T_b using S_b and save it.
4. **end for**

For any new testing points, the final classification result will be the majority vote of these B decision trees. Random forest algorithm differs in only one way from tree bagging algorithm that a random subset of the features are selected to split the node in the tree building process instead of searching the best splitting feature in the whole feature set. The reason for doing so is to reduce the correlation of the trees in an ordinary bootstrap sample. For example, if one or a few features are very strong predictors for the target variable, they will be selected in many of the decision trees causing them highly correlated and reducing the power of the ensemble. Following is the algorithm for random forest

Algorithm 4 random_forest

Input(s): Training set (x_i, ω_i) for $i = 1,2, \dots, N$; number of trees B; number of features F
Output: Tree classifier $T_1, T_2, \dots, T_B$
1. **for** $b = 1$ **to** B **do**
2. Build a dataset S_b, by sampling N items randomly with replacement from the original data pool (x_i, ω_i) for $i = 1,2, \dots, N$
3. Train decision tree T_b without pruning using S_b. In each node splitting process, randomly select F features without replacement from the whole feature set and search the best splitting threshold inside the selected feature subset
4. **end for**

The same as tree_bagging, new testing point classification is done base on the majority vote of the B built decision trees.

For both tree_bagging and random_forest algorithms, we also need to specify the number of trees B we want to build beforehand. Out of bag (OOB) error could be used to do the parameter specification. The definition of OOB is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in the bootstrap sample.

Random forest is one of the most popular ensembles because of several advantages. Firstly, the performance of random forest is usually better than bagging and comparable with boosting. Yet, it is not too vulnerable to the outliers or noises unlike Boosting. Secondly, it can handle a very large number of input attributes and the running time is short (Skurichina, 2002). Moreover, random forest could also be used to calculate proximities between pairs of instances and the importance of each variable (Breiman, 2001 and Archer, 2008).

1.3.2 Random subspace

Random subspace is considered as the generalization of the random forest algorithm (Ho, 1998). Random forest are composed of decision trees, while a random subspace classifier can be composed from any underlying classifiers, e.g., linear classifiers (Skurichina, 2002), support vector machine (Tao, 2006), nearest neighbors (Tremblay, 2004) and other types of classifiers. The algorithm is shown below:

Algorithm 5 random_subspace

Input(s): Training set (x_i, ω_i) for $i = 1,2, \dots, N$; number of classifiers B; number of features F

Output: Classifiers $C_1, C_2, \dots, C_B$

1. **for** $b = 1$ **to** B **do**
2. Randomly select F features without replacement from the whole feature set to feature subset F_b
3. Train classifier C_b using the original data with feature subset F_b
4. **end for**

Similar to random forest, new testing point classification is done base on the majority vote of the B trained classifiers.

The algorithm is an attractive choice for classification problems where the number of features is much larger than the number of training objects, such as fMRI (Kuncheva, 2010) data or gene expression data (Bertoni, 2005).

Chapter II: Class Imbalanced Data

2.1 The problem of class imbalanced data

Recent developments in science and technology enabled the explosion of data in both quantity and diversity, which created immense opportunity for knowledge discovery and data engineering research to play an increasingly important role in a wide range of applications (He, 2009). In recent years, imbalanced data learning problem has drawn a significant amount of attention from the machine learning society. As the name suggests, class-imbalanced data stand for dataset where the number of observations belonging to each class is different. Usually, one of the classes contributes only a very small minority of the data and makes the data significantly imbalanced, as introduced in many real problems, e.g., fraud detection (Fawcett et al., 1997), medicine (Mac Namee et al., 2002), language (Cardie et al., 1997), etc.

The fundamental issue with imbalanced data learning problem is the ability of imbalanced data to significantly compromise the performance of most standard learning algorithms. Breiman et al. (1984) discussed the connection between the prior probability of a class and its error cost. Classes with fewer observations in the training set have a lower prior probability and a lower error cost. This is problematic when the true error cost of the minority class is higher than is implied by the distribution of observations in the training set. And this is usually the case, considering credit card fraud problem, among all the transactions less than 0.1% are fraud, which will translate into billions of dollars in losses (Hassibi, 2000). How to accurately predict or differentiate these minority events, is of fundamental importance to classification task, however, most

available classification algorithms assume or expect balanced class distributions and equal misclassification costs, so when applied to imbalanced dataset, they prefer to classify most cases to the majority class and often fail to correctly classify the minority class, so called compromised performance by Breiman. Before we go into the reason causing these problems, let us introduce the accuracy measure first.

2.2 Measures of classifier performance

2.2.1 Overall accuracy

It is one of the most widely used measurement to evaluate the performance of a classifier. The ideal of overall accuracy is very simple, and it is the fraction with number of correctly classified examples ($N_{correct}$) as numerator and the number of total examples (N_{total}) as the denominator.

$$\text{overall\_accuracy} = \frac{N_{correct}}{N_{total}}$$

From the above equation, the overall classification error is defined as $1 -$ overall_accuracy. And usually this error measurement serves directly as the cost function or as a core elements of the cost function to be minimized/maximized in the classification algorithm training process. Overall classification error assigns equal misclassification cost, which is $\frac{1}{N_{total}}$, to every data point. For balanced data, the measure works well since the population of each class is very close. However, as the class imbalance degree increases, a systematic bias is introduced to this measurement which will assign examples from class with larger population more weights those examples in

minority class. And as a result, for those algorithm including this error measure in the loss function, will incline to classify most points to the majority class. Take the credit card fraud problem as an example, if all the transactions are classified as non-fraud transaction, the classification overall accuracy will be 99.9% which is still very high. However, the resulting model would be of no use since it failed to predict any of the fraud event we have more interest in.

Though accuracy provides a single simple number for diagnostic performance, it is often too simple and must be interpreted with considerable caution, just as we described above. Meanwhile, more accuracy measures are needed to supplement its limitation.

2.2.2 Alternative accuracy metrics

Overall accuracy calculate the correction prediction rate across all classes, a simple improvement would be to further partition the hits (correct predictions) and misses (wrong predictions), which leads to the following confusion matrix **Table 1** (in this dissertation, we only focus on binary classification problems)

Table 1 Confusion matrix

		Predicted label	
		Positive	Negative
Actual label	Positive	TP	FN
	Negative	FP	TN

The above table is named as confusion matrix, and all the possible binary classification outputs are included. True positive (TP) corresponds to the number of positive examples correctly classified by the classifier and true negative (TN)

corresponds to the number of correctly classified negative cases. False positive (FP) indicates the number of those negative cases that were classified as positive cases and false negative (FN) represents the number of positive cases that were wrongly predicted as positives cases. Based on these four numbers, the overall accuracy could be redefined as

$$\text{overall\_accuracy} = \frac{N_{correct}}{N_{total}} = \frac{TP + TN}{TP + FN + TN + FP}$$

Besides this, we could also define the within class accuracy as well, i.e., sensitivity, positive class classification accuracy and specificity, negative class classification accuracy.

$$\text{sensitvity} = \frac{TP}{TP + FN}$$

$$\text{specificity} = \frac{TN}{TN + FP}$$

Using these three definitions, we could rethink the problems in §2.2.1 more quantitatively (Song, 2014). Suppose the positive cases number in the training set is $N_+ = TP + FN$ while the number of negative cases is $N_- = TN + FP$. $k = \frac{N_+}{N_-}$ is defined as the ratio of the number of the two classes.

$$\begin{aligned}\text{overall\_accuracy} &= \frac{TP + TN}{TP + FN + TN + FP} \\ &= \frac{TP}{TP + FN + TN + FP} + \frac{TN}{TP + FN + TN + FP} \\ &= \frac{TP}{(TP + FN) + \frac{1}{k}(TP + FN)} + \frac{TN}{k(TN + FP) + (TN + FP)}\end{aligned}$$

$$= \frac{k}{1+k} * \text{sensitivity} + \frac{1}{1+k} * \text{specificity}$$

Now if we define $\beta = \frac{N_+}{N_+ + N_-} = \frac{k}{1+k}$ as the imbalanced rate of the dataset, the above equation could be reformulated as

$$\text{overall\_accuracy} = \beta * \text{sensitivity} + (1-\beta) * \text{specificity}$$

For balanced data, β is near or equal to 0.5, in which case maximizing the overall accuracy is equivalent to maximizing the sensitivity and specificity with the same weight. However, for imbalanced data with β approaching 0 (positive class minority), maximizing the overall accuracy will bias toward maximizing the specificity more than the sensitivity. We give an example here, suppose for one dataset $\beta = 0.01$, an unit increase of specificity will actually contribute 100 times more than the contribution made by 1 unit increasing in sensitivity. This is probably the reason why most examples in positive class minority data are classified as negative, because the increase in specificity contribute more to the increase of overall accuracy comparing to specificity. The opposite situation could be observed as β approaches 1 (negative class minority).

Sometimes, sensitivity is also refer to as recall. With similar definition, precision determines the rate of number of correct classified positive cases over all the cases that were classified as positive. The higher the precision, the lower the number of FP errors committed by the classifier. Recall and precision are two widely used metrics employed in applications where successful detection of one of the classes is considered more significant than detection of the other classes. For the convenience of description, the formal definition of these metrics is given below.

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

$$\text{Recall} = \frac{TP}{TP + FN}$$

It is often possible to construct baseline models that maximize one metric but not the other. For example, a model that predict every record as positive class will have a perfect recall but very poor precision because of the high FP cases. Conversely, the model assigns a positive class to every test example that matches one of the positive records in the training set have very high precision, but low recall because of the low TP rate. As a result, a preferred model would be capable to maximize precision and recall simultaneously. And the F measure serves for this purpose.

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} = \frac{2 * TP}{2 * TP + FP + FN}$$

In math, F measure is the harmonic mean of precision and recall. F measure tends to be closer to the smaller one of the two elements. Hence a high value of F ensures that both precision and recall are reasonably high.

In **Chapter 4**, we borrow the idea of harmonic mean and propose to use the harmonic mean of sensitivity and specificity for imbalanced data classification. So we save the discussion to **Chapter 4** and introduce some traditional strategies to deal with imbalanced data in the following sections.

2.3 Solutions for imbalanced learning

Provost (2000) gave a good summary about the related issue for imbalance dataset classification. He pointed out first that why the normal classifier would cause problems

for the imbalanced data set: (1) Maximizing accuracy is the goal. (2) In use, the classifier will operate on data drawn from the same distribution as the training data. It also pointed out that when studying problems with imbalanced data, using the classifiers produced by standard machine learning algorithms without adjusting the output threshold may well be a critical mistake. In this dissertation, we actually also follow the problem list to provide solutions. For the accuracy maximization problem, we have already discussed earlier this chapter and for the problem of threshold adjusting problem in **Chapter 3** we proposed ROC based operating point selection method which could preserve the minority class example detection rate.

In this section, we review the common strategies solutions proposed to overcome the effects of imbalanced data. We establish some notations first. Considering an imbalanced dataset $\mathcal{D}$ with binary labels $\{+1, -1\}$. $\mathcal{D}_{maj}$ stands for set of examples with majority class while $\mathcal{D}_{min}$ represents the set of minority class cases, and $|\mathcal{D}_{maj}|$ and $|\mathcal{D}_{min}|$ are the number of observations in the two set respectively.

2.3.1 Sampling methods for imbalanced data

Provost (2001) also asked a question if the natural class distribution would be the best training data class distribution. Weiss's answer is no based on studies carried out in 2001. In this section, we review the most common sampling method to deal with imbalanced data. Typically, the use of sampling methods in dealing with imbalanced data learning problems consists of modification of the degree of imbalance in order to provide a balanced distribution (He, 2009). Studies has shown that for several base classifiers, a balanced data set provides improved (Weiss et al., 2001, Estabrooks et al.,

2004 and Song et al., 2014). These results justify the use of sampling methods for imbalanced learning.

2.3.1.1 Random oversampling and undersampling

The methods of random oversampling follow naturally from its description by adding a set E randomly sampled from the minority class set $\mathcal{D}_{min}$ with replacement. In this way the number of total examples in $\mathcal{D}_{min}$ is increased by $|E|$ and the class distribution is therefore adjusted. While oversampling appends data to the original data set, random undersampling (also refer to as downsampling) removes data from the dataset. Opposite to oversampling, in undersampling we randomly select a set E of majority class examples $|\mathcal{D}_{maj}|$ and remove them so that $|\mathcal{D}_{maj}| - |E| = |\mathcal{D}_{min}|$. The undersampling also provides us a simple method for adjusting the balance of the original data set $\mathcal{D}$.

At first glance the oversampling and undersampling methods appear to be functionally equivalent since they both alter the size of the original data set and can actually provide the same proportion of balance. However, this commonality is superficial and each method has their own limitation which can potentially hinder the learning process (Holte et al., 1989, Drummond et al., 2003 and Mease et al., 2007). In the case of undersampling, the problem is relative obvious: by removing examples from the majority class the information contained in $\mathcal{D}$ is reduced, which will cause the classifier to miss important concept pertaining to the majority class. With respect to oversampling, the problem is a little more opaque: since oversampling simply appends replicated data to the original data set, multiple instances of certain examples become "tied" leading to overfitting (Mease et al., 2007). In particular, overfitting in

oversampling occurs when classifiers produce multiple clauses in a rule for multiple copies of the same observation which causes the rule to become too specificity and lace generalization capability. The training accuracy will be high in this scenario while the classification performance in the testing set is worse.

2.3.1.2 Informed undersampling

In this section we introduce several algorithms for informed undersampling that have shown good results.

2.3.1.2.1 Tomek links

The definition of Tomek links (Tomek, 1976) is very similar to the single linkage definition used in clustering algorithms. Considering two examples $E_i \in \mathcal{D}_{maj}$ and $E_j \in \mathcal{D}_{min}$, if there is not an example $E_l \in \mathcal{D}_{min}$ such that the Euclidean distance $d(E_i, E_l) < d(E_i, E_j)$ or $E_l \in \mathcal{D}_{maj}$ such that $d(E_j, E_l) < d(E_i, E_j)$, then we cay that E_i and E_j form a Tomek link. If two examples form a Tomek link, then either one of these examples is noise or both examples are borderline. Tomek links can be used as an under-sampling method, in which only examples belonging to the majority class are eliminated, or as a data cleaning method, in which examples of both classes are removed.

2.3.1.2.2 Edited nearest neighbor rule (ENN)

Edited nearest neighbor rule apply the modified KNN rule introduced by D. Wilson (1972). Usually the k is set to 3. For each point $E_i \in \mathcal{D}_{maj}$, if minority class data dominate its k nearest neighbor, then this point will be removed from $\mathcal{D}_{maj}$.

2.3.1.2.3 Condensed nearest neighbor rule (CNN)

Hart's condensed nearest neighbor rule (Hart, 1968) is used to find a consistent subset of examples. A subset $\widehat{\mathcal{D}} \in \mathcal{D}$ is consistent with $\mathcal{D}$ if and only if $\widehat{\mathcal{D}}$ could correctly classify the example in $\mathcal{D}$ when using a 1-nearest neighbor. $\widehat{\mathcal{D}}$ could be created in following steps: first, randomly draw one majority class example and all examples from the minority class and put these examples in $\widehat{\mathcal{D}}$. Afterwards, use a 1NN over the examples in $\widehat{\mathcal{D}}$ to classify the examples in $\mathcal{D}$. Every misclassified example from $\mathcal{D}$ is moved to $\widehat{\mathcal{D}}$. It is important to note that this procedure does not find the smallest consistent subset from $\mathcal{D}$. The idea behind this implementation of a consistent subset is to eliminate the examples from the majority class that are distant from the decision border, since these sorts of examples might be considered less relevant for learning.

2.3.1.2.4 Neighborhood cleaning rule (NCL)

Neighborhood cleaning rule (NCL, Laurikkala, 2001) uses Wilson's ENN to remove majority class examples. ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors. NCL modifies the ENN in order to increase the data cleaning process. For a binary class problem, the algorithm can be described in the following way: for each example E_i in the training set, its three nearest neighbors are found. If E_i belongs the majority ($E_i \in \mathcal{D}_{maj}$) class and the classification given by its three nearest neighbors contradicts the original class of E_i, then E_i is removed. If E_i belongs to the minority ($E_i \in \mathcal{D}_{min}$) class and its three nearest neighbors misclassify E_i, then the nearest neighbors that belong to the majority class are removed.

2.3.1.2.5 One side selection (OSS)

One side selection (Kubat et al., 1997) is an undersampling method resulting from the application of Tomek links followed by the application of CNN. Tomek links are used as an undersampling method and removes noisy and borderline majority class examples. Borderline examples can be considered "unsafe" since a small amount of noise can make them fall on the wrong side of the decision border. CNN aims to remove examples from the majority class that are distant from the decision border. The remainder examples, i.e., "safe" majority class examples and all minority class examples are used for learning.

2.3.1.2.6 Balance Cascade

BalanceCascade (Liu et al., 2006) algorithm takes a supervised learning approach that develops an ensemble of classifier to systematically selection which majority cases to be included in the undersampled set E. Consider a sampled set E_1 of majority class $\mathcal{D}_{maj}$, and the fact that $|E_1| = |\mathcal{D}_{min}|$. Subject to set $\mathcal{D}_1 = \{E_1 \cup \mathcal{D}_{min}\}$ we induce ensemble C_1. Then based on the classification of C_1on $\mathcal{D}_1$, we identify all cases that are correctly classified as belongings to $\mathcal{D}_{maj}$, we call them as N_{maj}^1. Since we know C_1 and it is reasonable to assume that N_{maj}^1 is somewhat redundant in $\mathcal{D}_{maj}$ since C_1 is already trained, so we remove them from the $\mathcal{D}_{maj}$ and generate a new sampled set from the resulting majority class samples, E_1, with $|E_1| = |\mathcal{D}_{min}|$. Again, subject to $\mathcal{D}_2 = \{|E_2| \cup |\mathcal{D}_{min}|\}$ we derive ensemble C_2. This procedure is iterated to a stopping criteria at which point a cascading combination scheme is used to form a final classifier.

2.3.1.2.7 kNN undersampling

kNN undersampling (Zhang et al., 2003) employs the structure of kNN classifier to guide the undersampling process. Based on the characteristics of the given data distribution, four kNN undersampling methods were proposed, namely NearMiss-1, NearMiss-2, NearMiss-3, and the "most distant" method. The NearMiss-1 method selects those majority examples whose average distance to the three closest minority class examples is the smallest, while the NearMiss-2 method selects the majority class examples whose average distance to the three farthest minority class examples is the smallest. NearMiss-3 selects a given number of the closest majority examples for each minority example to guarantee that every minority example is surrounded by some majority examples. Finally, the "most distance" method selects the majority class examples whose average distance to the three closest minority class cases is the largest. Experimental results (Zhang et al., 2003) shows NearMiss-2 method can provide competitive results for imbalanced learning.

2.3.1.3 Synthetic minority oversampling technique (SMOTE)

Chawla et al. (2002) combined over-sampling and down-sampling to achieve better classification performance than simply down-sampling the majority class. Rather than over-sampling with replacement, they create synthetic minority class examples to boost the minority class, which is called the synthetic minority oversampling technique (SMOTE). To be specific, for subset $\mathcal{D}_{min} \in \mathcal{D}$, consider the K-nearest neighbors (with Euclidean distance setting) for each point $x_i \in \mathcal{D}_{min}$, for some specified integer K. To create a synthetic sample, randomly select one of the K-nearest neighbors, then multiply

the corresponding feature vector difference with a random number in [0,1], and finally, add this vector to x_i which leads to

$$x_{new} = x_i + (\hat{x}_i - x_i) * \delta$$

where $\hat{x}_i$ is one of its K-nearest neighbors and $\delta \in [0,1]$ is a random number. Therefore, the resulting synthetic instance according to the above definition is a point along the line segment joining x_i and one of its K-nearest neighbors $\hat{x}_i$. These synthetic samples help break the ties introduced by random oversampling. Favorable improvement is shown in Chawla's paper by comparing SMOTE to random undersampling and random oversampling.

2.3.1.4 Combination of oversampling with undersampling

Although oversampling minority class examples can balance class distributions, some other problems usually present in data sets with skewed class distributions are not solved. Frequently, class clusters are not well defined since some majority class examples might be invading the minority class space. The opposite can also be true, since interpolating minority class examples can expand the minority class clusters, introducing artificial minority class examples too deeply in the majority class space. Including a classifier under such a situation can lead to overfitting.

2.3.1.4.1 SMOTE + Tomek links

In order to create better defined class clusters, a method applying Tomek links to the oversampled SMOTE set was proposed (Batista et al., 2004). Instead of removing only the majority class examples that form Tomek links, examples from both classes are removed, as the data cleaning method we introduced in §**2.3.1.2.1**. The application could

be implemented in the following steps: firstly using SMOTE to oversample the dataset, secondly identify Tomek links and remove the examples forming the link from both classes, which produces a balanced data set with well-defined class clusters.

2.3.1.4.2 SMOTE + ENN

The motivation behind this method is similar to SMOTE + Tomek links. ENN tends to remove more examples than the Tomek links does, so it is expected that it will provide a more in depth data cleaning. Differently from NCL, which is an undersampling method, ENN is used to remove examples from both classes. Thus, any example that is misclassified by its three nearest neighbors is removed from the training set. SMOTE+ENN method was proposed by Batista et al. in 2004.

2.3.1.5 Simulation on sampling method

In this section, we simulate a 2 dimensional binary class dataset to show how these sampling strategies perform on the dataset. Class "0" is the minority class and its 20 data points satisfy $\{(x,y)|x\sim N(0,0.8), y\sim N(x-2,0.8)\}$. Class "1" is the majority class and its 100 points satisfy $\{(x,y)|x\sim N(0,2), y\sim N(x+1,2)\}$. The original data plot is compared with 10 other sampling method, please refer to the following **Figure** 5.

Randomly down sampling and over sampling are very easy to understand. For Tomek links sampling, we can observe that the majority example which is very close to minority class are removed. Similar to Tomek links, ENN also removes majority examples which are close to minority cases, however, it is more conservative. CNN remove most of the majority example which are distant from class-border with minority class. OSS is similar to Tomek links. However, comparing to CNN, less distant

examples are removed. NCL extends ENN strategies and more majority examples close to class-border are removed. SMOTE generate more new minority examples to balance the two class. SMOTE + Tomek links and SMOTE + ENN are application of Tomek and ENN methods on SMOTE dataset.

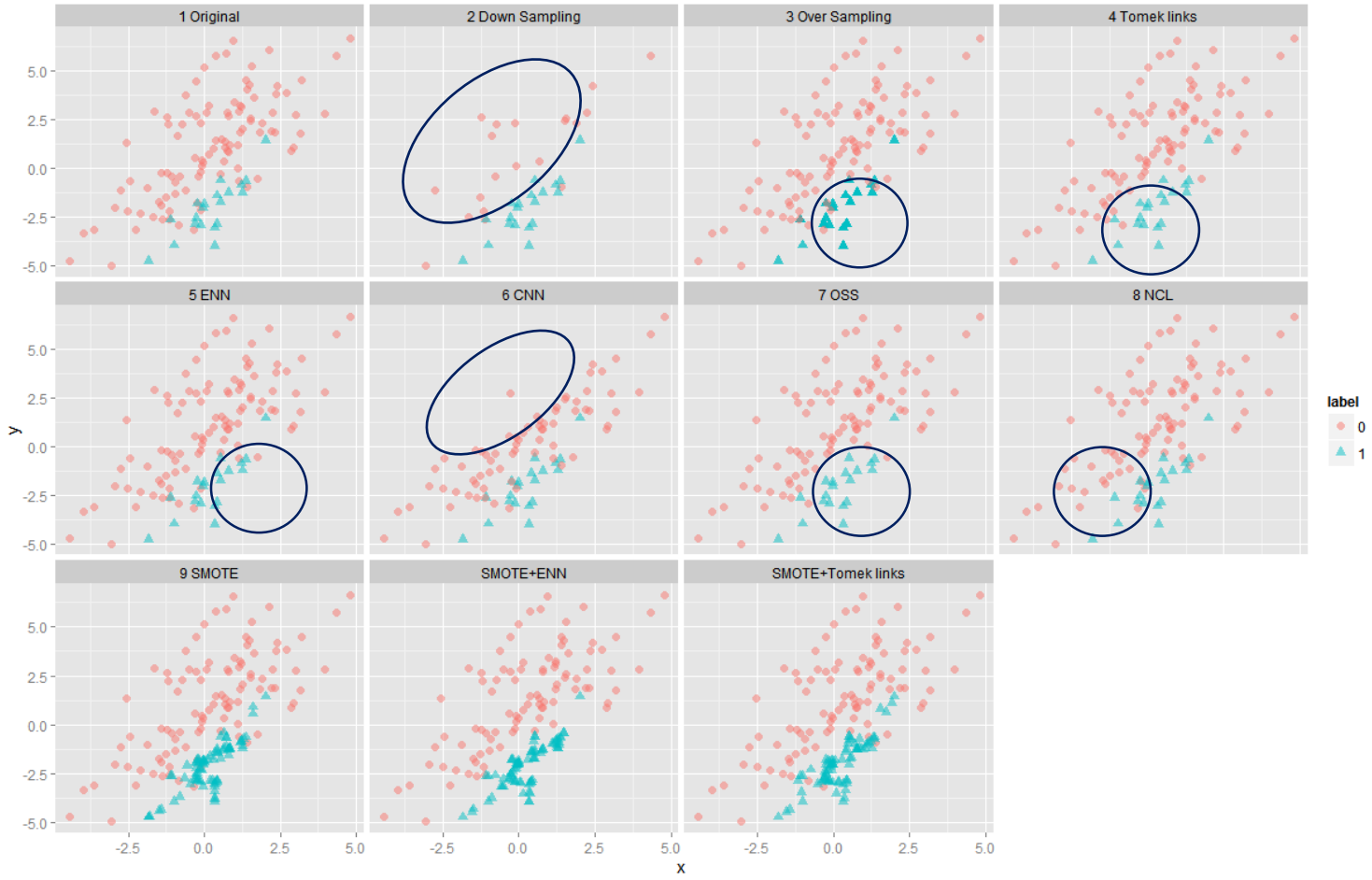


Figure 5 Comparison of different sampling methods. The red dots indicates majority class examples and blue triangle indicate minority class examples. The dark blue circle region marks the area where majority example are removed.

Basically, it could be observed from the dataset that informed sampling method mainly focus on removing two aspects of the dataset, i.e., examples distant from class-border and examples close to the class-border. Additionally, informed sampling usually do not generate a balanced dataset, instead, they remove the hard examples and leave the classification problems to the classification algorithms.

2.3.2 Cost sensitive learning for imbalanced data

While sampling methods attempt to balance distributions by considering the representative proportions of class examples in the distribution, cost-sensitive learning methods consider the costs associated with misclassifying examples (Elkan, 2001 and Ting, 2002). Cost sensitive learning targets the imbalanced learning problem by using different cost matrices that describe the costs for misclassifying any particular data example. It provides a viable alternative to sampling methods for imbalanced learning domain.

Fundamental to the cost-sensitive learning methodology is the concept of cost matrix, which can be considered as a numerical representation of the penalty of classifying examples to the wrong class. As shown in **Table 2**, C(+,-) denotes the cost of misclassifying a positive case to negative class. Typically, there is no cost for correct classification of either class and the misclassification cost of minority examples is higher than the positive cases, i.e., $C(-,+)>C(+,-)$. Recalling the Adaboost algorithm, it assumes that $C(-,+)=C(+,-)$, and this probably is the reason why Adaboost is not a good choice for dealing with imbalanced data.

Table 2 Cost matrix

		Predicted label	
		Positive	Negative
Actual label	Positive	C(+,+)	C(+,-)
	Negative	C(-,+)	C(-,-)

The objective of cost sensitive learning then is to develop a hypothesis that minimizes the overall cost on the training data set. In general, there are three ways of implementing cost-sensitive learning (He et al., 2009). The first class of techniques apply misclassification costs to the data set as a form of dataspace weighting; these techniques are essentially cost-sensitive bootstrap sampling approaches where the misclassification costs are used to select the best training distribution for induction. The second class applies cost-minimizing techniques to the combination schemes of ensemble methods, this class consists of various metatechniques where standard learning algorithms are integrated with ensemble methods to develop cost-sensitive classifiers. The last class of techniques incorporates cost-sensitive functions or features directly into classification paradigms to essentially "fit" the cost sensitive framework into these classifiers. Since different classification algorithms have different structures, there is no unifying framework for this cost sensitive learning method.

Chapter III: Receiver Operating Characteristic Analysis

In Chapter II, we introduce the class imbalanced learning problem and the common solutions to deal with it, that is, sampling strategies and cost sensitive learning techniques. However, they do not imply that classifiers cannot learn from imbalanced data sets and studies have also shown that classifiers induced from certain imbalanced data sets are comparable to classifiers induced from the same data either by employing sampling techniques or by using cost sensitive learning (Batista et al., 2004; Japkowicz et al., 2002; Chen et al., 2004; Maloof, 2003 and Song et al., 2014). It has also been noticed that for certain studies by simply tuning the decision threshold of the classifier induced from imbalanced data, the classification results, in terms of sensitivity and specificity, are comparable with those employing sampling and cost sensitive learning techniques (Chen et al., 2004 and Maloof, 2003). Receiver operating characteristics (ROC) analysis (Metz, 1978) consider relative ranking of the cases by varying the classification threshold from the smallest to the largest and therefore is an alternative measure of classification accuracy which is not affected by selection of the threshold (Fawcett, 2006). In our previous study (Song, 2014), we conducted experiments on medical imaging data by comparing our proposed method of moving decision threshold along ROC curve with the other two methods, and the results showed the advantages of our proposed methods. In this section, we review the basic setting of ROC analysis.

3.1 Receiver operating characteristics (ROC) space

ROC was first introduced in signal detection theory to depict the tradeoff between hit rates and false alarm rates of classifiers (Metz, 1978). ROC analysis was then

extended for use in visualizing and analyzing the behavior diagnostic systems (Swets, 1988). Spackman's study (1989) in evaluating and comparing algorithms using ROC curves is one of the earliest adopters of ROC graphs in machine learning area. An increase in the use of ROC graphs has been seen in the machine learning community, due in part to the realization that simple classification accuracy is often a poor metric for measuring performance (Provost et al., 1997, 1998). Besides its capability of being a generally useful performance graphing method, it also have properties that make them especially useful for domains with skew class distribution and unequal classification error costs. And because of this, ROC analysis is gaining increasing attentions from researchers of areas of imbalanced data learning.

ROC graph are two-dimensional graph by plotting sensitivity on the Y axis and (1-specificity) on the X axis. Recalling the definition of sensitivity and specificity from Chapter 2, and here we redefine them as true positive rates (TPR) and false positive rates (FPR)

$$\text{TPR} = \text{sensitivity} = \frac{TP}{TP + FN}$$

$$\text{FPR} = 1 - \text{specificity} = \frac{FP}{FP + TN}$$

Every classification decision, the label outputs of a classifier applied on a dataset, corresponds to a single point in the ROC graph. Several points in ROC space are important to note. As shown in Figure 8. The lower left point E (0, 0) represents that all examples are classified to the negative class since its sensitivity is 0 and specificity is 1. The upper right point G (1, 1) indicates the opposite strategy which predicts all the cases

as positive class. If all the examples are correctly identified by the classifier, the corresponding point will locate in the upper left F (0, 1), i.e., both sensitivity and specificity are 1. And informally, one point is better, in terms of classification performance, than another in ROC graph if it is closer to the upper left corner (higher TPR, lower FPR, or both) of the other one. Fawcett in his paper (2006) give a conservative vs liberal description on the locations of the points generated by classifiers "Classifiers appearing on the left-hand side of an ROC graph, near the X axis, may be thought of as 'conservative': they make positive classifications only with strong evidence so they make few false positive errors, but this also leads to a lower TPR; on the other hand, classifiers on the upper right-hand side of the ROC graph may be thought of as 'liberal': they make positive classifications will weak evidence so they classify nearly all positives correctly, but often with high FPR". As shown in **Figure 6**, A is more conservative than B.

The diagonal line (dashed line in **Figure 6**) represents the strategy of randomly guessing the class. For example, if a classifier randomly guesses positive class with probability of 0.5 then it can be expected to get half the positives and half the negatives correct, which yields the point (0.5, 0.5) on the ROC graph. If the corresponding positive class guessing probability is 0.7, then it will yield the point (0.7, 0.7) in the ROC space, point C on **Figure 6**. As we vary this probability from 0 to 1 and the points yielding will form a line in the ROC graph, i.e., the dashed diagonal line. Any points locates in the lower right triangle, e.g., point D in Figure 8 indicate the classifier makes a worse than random guessing. Actually, this is a "worse" but useful classification. If we simply reverse its classification decision on every instance, the resulting point will locates in the

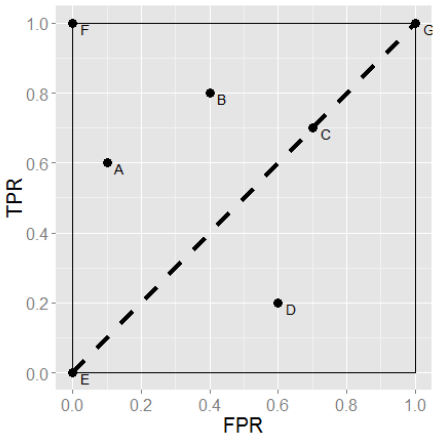


Figure 6 Points in ROC space. Points A and B locate above the dashed diagonal line, which indicates random guess performance. Points C locates on the dashed line while point D is below the dashed line. Point E, point F and point G indicate all negative classification, perfect classification and all positive classification respectively.

upper left triangle with its TPR and FPR being the previous (1-TPR) and (1-FPR) respectively. For example, if we reverse the decision made on D (0.6, 0.2), the new decision will yield point at (0.8, 0.4). The process is shown below:

Table 3 Confusion matrices with changing of classification direction.

		Prediction$_1$		Prediction$_2$	
		P	N	N	P
Truth	P	TP_1	FN_1	FN_2=TP_1	TP_2=FN_1
	N	FP_1	TN_1	TN_2=FP_1	FP_2=TN_1

$$\text{TPR}_2 = \frac{TP_2}{TP_2 + FN_2} = \frac{FN_1}{FN_1 + TP_1} = 1 - \text{TPR}_1$$

$$\text{FPR}_2 = \frac{FP_2}{FP_2 + TN_2} = \frac{TN_1}{TN_1 + FP_1} = 1 - \text{FPR}_1$$

The above reversion was employed in the ROC Random Forest construction which will be introduced in Chapter 4. And we start to review the mathematical setting of ROC curve in the following section.

3.2 ROC curve

Many classifiers, like decision tree and KNN will only output the class label, e.g., P or N of the input examples, and therefore it yields a single confusion matrix, which in turn corresponds to one single point in ROC graph. Some strategies could be used to convert such discrete classifier to continuous scoring classifier, e.g., looking inside method (Provost, 2001), aggregating method (Domingos, 1999) and weighted voting method (Fawcett, 2001).

Some other classifiers, such as SVM and Random Forest, will generate an instance probability or score instead, which is a continuous numeric value measuring the degree to which an instance is a member of a class. Techniques, like sigmoid conversion for SVM score, exist for converting an uncalibrated score into a proper probability, and without loss of generality, we call both a probabilistic classifier.

Combining with a pre-specified threshold, e.g., 0.5 for a probabilistic score, such scoring classifier could produce a binary classification label: if the score is greater than the threshold, the prediction of corresponding example will be P, otherwise it will be N. Every threshold will generate a different point in the ROC graph. Here we consider efficient operating threshold only: if two thresholds produce the same (TPR, FPR), we consider them as non-efficient operating thresholds. Theoretically, if we vary the threshold from the possibly smallest value to the largest value and then trace the corresponding points in the ROC space, we could get a curve. Since the efficient operating points only depends on the scoring list generated by the classifier, a common way of generating ROC curve is to use theses scores as threshold and a naive implementation is given below:

Algorithm 6. ROC_generating

Inputs: Y the set of test examples; $p(i)$ the probabilistic score that example i is positive; P and N, the number of positive and negative examples.

Output: R, a list of ROC points (TPR, FPR) with decreasing order of TPR, and T, splitting corresponding threshold of R

1. $Y_{sort} \leftarrow Y$ sorted increasing by p score
2. **insert** $-$Inf **to** Y_{sort}
3. $i \leftarrow 1$
4. **for** $i = 1$ **to** $|Y_{sort}|$ **do**
5. $thres \leftarrow Y_{sort}(i)$
6. $TP \leftarrow FP \leftarrow 0$
7. $j \leftarrow 1$
8. **for** $j = 1$ **to** $|Y|$
9. **if** $p(j) > thres$ **then**
10. **if** $Y(i)$ is positive **then**
11. $TP \leftarrow TP + 1$
12. **end if**
13. **if** $Y(i)$ is negative **then**
14. $FP \leftarrow FP + 1$
15. **end if**
16. **end if**
17. **end for**
18. **push** $(\frac{TP}{P}, \frac{FP}{N}, thres)$ onto R
19. **end for**
20. **return** R

Table 4 gives an example of classification results with the instances sorted by the probabilistic score. The corresponding ROC curve is plotted in Figure 7.

One very important idea ROC curve bring to us is that it only cares about the relative ranking of the score of the examples instead of the accurate calibrated probability estimates. For the above example, if we change the probability score to 20, 19, …, 1 for instance 1, 2, …, 20 respectively, we will get the exact same ROC curve as we use probabilistic score. As long as we rank the positives higher than negatives, the model would be robust for the differentiation task, no matter how large the "probability" of the

negatives. This property was another reason why we employ ROC in our designing of ROC Random Forest.

Table 4 Dataset with information of label and probabilistic score.

Instance	Label	Score	Instance	Label	Score
1	**p**	0.95	11	**p**	0.43
2	**p**	0.85	12	**n**	0.41
3	**p**	0.80	13	**n**	0.37
4	**n**	0.65	14	**p**	0.35
5	**p**	0.64	15	**n**	0.29
6	**p**	0.60	16	**n**	0.25
7	**n**	0.52	17	**n**	0.19
8	**p**	0.51	18	**p**	0.13
9	**n**	0.49	19	**n**	0.10
10	**n**	0.47	20	**n**	0.05

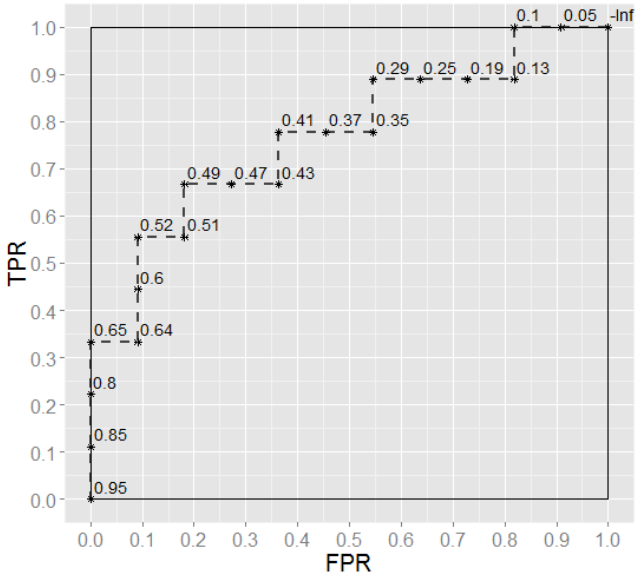


Figure 7 ROC curve corresponding to dataset in Table 4.

A very important fact we need to point out here is that the classifier scores should not be compared across model classes. One model class may be designed to produce scores in the range [0, 1], e.g., logistic regression model, while another produces scores in $[-\infty, +\infty]$, e.g., SVM. In such case, comparing model performance at a common threshold will be meaningless, especially for class imbalanced data.

Another property of ROC curve is that the curve is monotonically increasing. As we can see from **Figure 7**, starting from (0, 0) the discrete curve could only move upward or right till it reaches point (1, 1).

3.3 The area under ROC curve (AUC)

ROC curve is a two-dimensional depiction of the classifier performance and in order to compare the performance of different classifiers we may want to reduce it to a single scalar value representing the expected performance level. A common method is to calculate he area under the ROC curve (AUC) (Bradley 1997 and Hanley et al., 1982). There are several methods could be used to calculate AUC and the most common way is to calculate using trapezoidal integration shown in the following equation:

$$AUC = \int_a^b f(\alpha) d\alpha \approx \sum_{i=1}^{n} \frac{h}{2} \left(f\left(\alpha + \frac{i-1}{h}\right) + f(a + ih) \right)$$

where $f(\cdot)$ is the sensitivity and it is a function of α, $\alpha = (1 - specificity)$, $a = 0, b = 1$, $n =$ size to increase, and $h = \frac{b-a}{n}$

Because AUC is only a portion of the area of a unit square, its value rangers from 0 to 1. So it could happen that the AUC will be less than 0.5. Since AUC of the random

guess line is 0.5, an AUC less than 0.5 means worse performance of a classifier than random guessing. As we introduced before, this only means we do not use the information appropriately and with simple operation the AUC would be larger than 0.5. Previously we assume the probabilistic scores measure the degree of belonging to positive class and now we assume the opposite, therefore we classify an example to a positive if and only if its score is lower than the threshold. The resulting ROC curve will be above the random guess line and its corresponding AUC becomes $(1 - \mathrm{AUC}_{\mathrm{previous}})$.

The AUC has an important statistical property: the AUC of a classifier depicts the probability that a randomly chosen positive example is ranked/scored higher than a randomly chosen negative example. In other words, the AUC measure is equivalent to the Mann-Whitney U statistics normalized by the number of possible pairings of positive and negative values, also known as the two sample Wilcoxon rank-sum statistic (Hanley, 1982). Meanwhile, the AUC is also closely related to the Gini coefficient (Breiman et al., 1984), which is twice the area between the diagonal and the ROC curve, and actually $\mathrm{Gini} + 1 = 2 * \mathrm{AUC}$ (Hanley et al., 2001).

Algorithm 7. AUC_calculation

Input(s): (TPR, FPR) pair: output generated from **Algorithm 7**
Output: $area$, the area under the ROC curve
1. $area \leftarrow 0$
2. **for** $i = 2$ **to** $|TPR|$
3. $\quad area \leftarrow area + \frac{1}{2} * (TPR(i) + TPR(i-1)) * |FPR(i) - FPR(i-1)|$
4. **end for**
5. **return** $area$

Generally speaking, high AUC value reflects good differentiation capability of a classifier. Even though certain discussions, e.g., how to explain ROC curves with intersection, exist (Metz, 1978), in practice, AUC performs very well and is often used as

general measure of the prediction power of a classifier. And therefore maximizing the AUC is often employed to direct sequential parameter or model searching to achieve better classification performance (Rakotomamonjy, 2004; Zhao et al., 2011). Meanwhile, AUC could also be used to select relevant features for classification have also been reported (Calle et al., 2011 and Wu et al., 2014).

An efficient AUC calculation algorithm is given in **Algorithm 7**. **Figure 8** shows ROC curves generated by two classifiers 1 and 2. From the figure we can observe that the curve generated by classifier 1 is closer to the upper left corner than the one of classifier 2. And for a given FPR value, the corresponding TPR of classifier 1 is higher than the TPR of classifier 2. All these facts indicate that than classifier 1 outperforms classifier 1 and as we expected, the higher AUC of classifier 1 well reflects this.

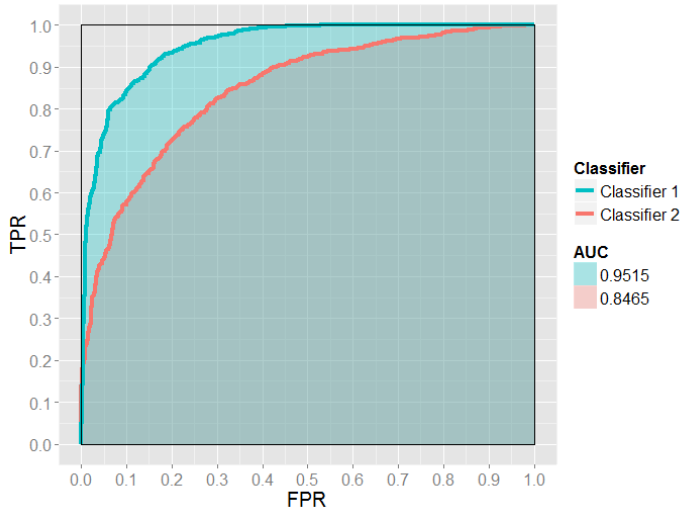


Figure 8 ROC curves of two different classification algorithms on the same dataset. The ROC curves of classifier 1 and classifier 2 are marked with blue and red, respectively. The shaded area indicates the corresponding area under curve for each ROC. And the AUC value of classifier 1 and classifier 2 are 0.9515 and 0.8465.

3.4 Averaging ROC curve

When doing comparison of classifiers, in order to reduce the bias of one time running, several strategies, e.g., random partition, bootstrapping (Efron, 1979), k-fold cross-validation (Geisser, 1993), were introduced to the process in order to take consideration of the variance of the AUC values. The basic idea of above methods are very similar: first train a classifier on selected examples, called training data, and then evaluate the model on the data, called testing set, that was not included in the training process, set of examples that are not included in the training process. Random partition, as its name suggests, simply randomly partition the dataset to two part, one serves as training set and the other serves as testing set. The size of these two set is pre-specified. For bootstrapping method, it repeatedly selects N, the size of the whole dataset, examples with replacement into the training set and uses the rest as the testing set. Theoretically, $(1 - \frac{1}{e})$ of the examples are expected to be included in the training set. The process of random partition and bootstrapping is usually repeated many times and a series of (training, testing) pairs are generated and then used to evaluate candidate classifiers. For k-fold cross validation, it first partition the dataset into k sub-groups with the same size, and then, in turn, each of the k sub-group serves as testing set and the rest (k-1) groups serve as the training set. Then k (training, testing) dataset pairs are generated and will be used for comparison purpose.

For each training-testing pair, one ROC curve could be plotted based on performance of the built model. Even though, based on these curves, we could easily get their AUC as well as the variance, one would also want to know whether we could get an averaged ROC curve, since AUC is the abstracted representation of the ROC and it

disregards the shape information of the curve. We could do this in two ways, vertical averaging and threshold averaging. The latter one is relatively easy to understand, it choose a universal threshold to threshold all the classifier scores. However, it suffers from problems that when the ground truth score distribution is different among different dataset and classifiers, this method may introduce bias. Meanwhile, the advantage of this method is that it keeps control of the thresholds, which could be extracted for explicitly. The vertical method is image based method which average the TPR with a universal fixed FPR for all the ROC curves. Comparing to threshold method, vertical method has no problem with the score distribution problem, however, it loss control of the threshold, which means for the resulting averaged ROC we could not tract the thresholds used to generate it. And in this dissertation, we stick on the vertical method to the comparison purpose. The corresponding algorithm is given in **Algorithm 8**.

Algorithm 8. vertical_ROC_averaging

Input(s): $list\_ROC$: a list of ROC curves generated from **Algorithm 7**; N: number of points to be kept on the x axis.

Output: $average\_ROC$

1. $num \leftarrow$ number of elements in $list\_ROC$
2. $tpr_{global} \leftarrow$ length (N + 1) zero vector
3. $fpr_{global} \leftarrow$ length (N + 1) vector of elements from 0 to 1 increase by $\frac{1}{\mathrm{N}}$
4. **for** $i = 1$ **to** num
5. $t \leftarrow$ tpr in ob[i]
6. $f \leftarrow$ fpr in ob[i]
7. $tpr_{local} \leftarrow$ length $(N + 1)$ zero vector
8. $idx \leftarrow 2$
9. **for** $j = 1$ **to** $|t|$
10. **while** $fpr_{global}(idx) < f(j)\ and\ idx < (N + 1)$
11. $tpr_{local}(idx) \leftarrow t(j - 1)$
12. $idx \leftarrow idx + 1$
13. **end while**
14. $tpr_{local}(idx) \leftarrow t(j)$
15. **end for**
16. **for** $j = 1$ **to** $N + 1$

17. $tpr_{global}(i) \leftarrow tpr_{global}(i) + tpr_{local}(i)$
18. **end for**
19. **end for**
20. **for** $i = 1$ **to** $N + 1$
21. $tpr_{global}(i) \leftarrow tpr_{global}(i)/num$
22. **end for**
23. **return** $(tpr_{global}, fpr_{global})$

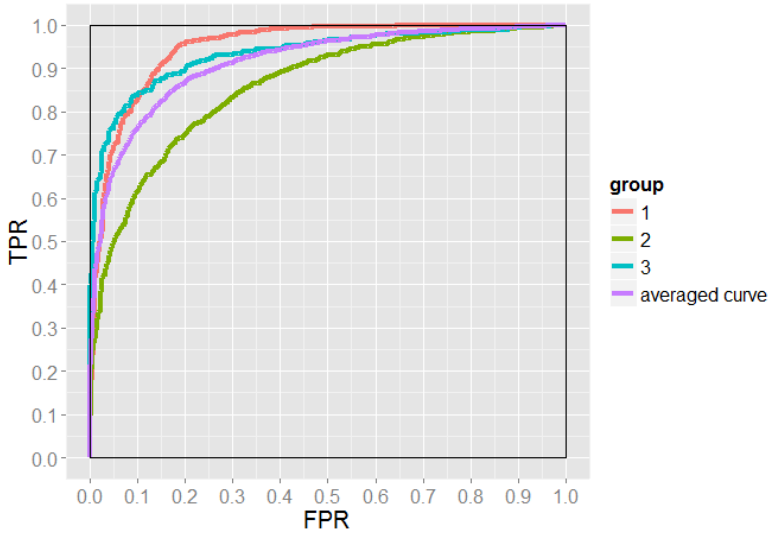


Figure 9 ROC curves and their averaged curve. The red, green and blue curve indicates 3 different ROC curves and the purple curve is the averaged ROC of the three generated by algorithm 9.

Figure 9 shows the averaged curve of three ROC curves, from which we can see that the averaged curve well reflects the main trends shown in the three curves.

4.5 ROC for imbalance data

One important assumption, in this dissertation, is we believe the class differentiation capability of the classifier we use here is better than random guess. This is a reasonable assumption, otherwise the algorithm could not be named as "classifier". To be specific, in terms of decision values of classifiers, it means positive cases will have

similar values, negative cases will have similar values and these two groups will be totally separated (ideal separation) or partly overlapped. Without loss of generality, let's say higher classification value indicates higher probability a case belongs to positive class. High differentiation capability indicates a classifier will assign higher value to positive cases than negative cases. Coincidently, this is the statistical meaning of the AUC value (see §**4.2**).

Considering probability classification output, in which case the decision value ranges from 0 to 1, an ideal classifier will assign positive cases with value near to 1 and negative cases with value near to 0. However, as long as the relative ranking of cases do not change, the AUC value will stay unchanged. Therefore, under certain circumstance, both positive and negative cases classification value are near to 1, it is possible that it has same AUC value as the ideal classifier. Let us take a look at the following example, considering dataset $\mathcal{D}$ of size 10, 3 of which belongs to the positive class and the rest belong to negative class. Based on certain features, classifier 1 and classifier 2 give their decision value of each case (**Table 5**). Classifier 1 gives a perfect separation and using 0.5 as threshold will deliver accuracy 1 classification. For classifier 2, all decision values shift to 0 and the classification accuracy would be 0.7 if 0.5 is employed as the threshold. However the AUC of both methods are same and classifier 2 also delivers a perfect separation of the two classes. The corresponding ROC curve is shown in the following **Figure 10**.

Table 5 Dataset with information of class label, probability score from classifier 1 and probability score from classifier 2.

Instance	1	2	3	4	5	6	7	8	9	10
Class	**p**	**p**	**p**	**p**	**p**	**n**	**n**	**n**	**n**	**n**
Classifier1	1.000	0.995	0.990	0.035	0.030	0.025	0.020	0.015	0.010	0.005
Classifier2	0.010	0.009	0.008	0.007	0.006	0.005	0.004	0.003	0.002	0.001

This is a typical problem, i.e., the decision value shifting problem, when we try to address imbalanced data problem. However, the relative ranking of these instances stay unchanged. In ROC analysis, the decision value is represented in terms of relative ranking and therefore AUC is invariant to such value shifting problems. And this is also the main reason why we choose ROC to deal with imbalance data problem in this dissertation. In next chapter, we will propose the ROC based Random Forest structure.

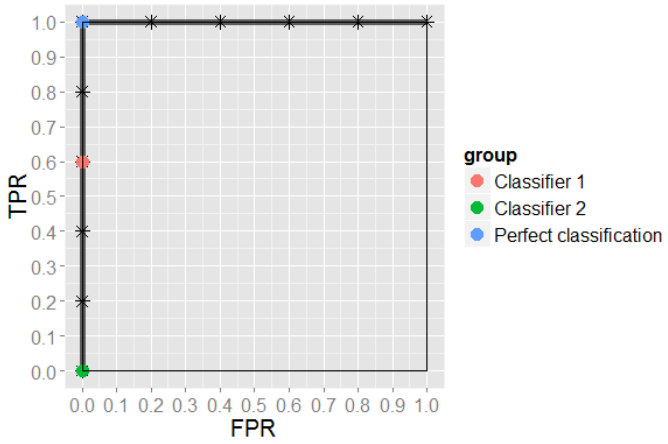


Figure 10 Classification results corresponding to Table 5. The solid curve is the ROC curve of both classifier. Red dot indicates classification point of classifier 1 and green dot indicates classification point of classifier 2. The blue dot is the ideal operating point which perfectly separate the dataset.

Chapter IV: ROC Random Forest

Random forest (Breiman, 1996, 2001) is a special case under the random subspace framework. A standard random forest algorithm consists of two main components: construction of base classifiers and the aggregating strategy, as we introduced in Chapter 1. By using the bagging sample and the random feature subset to build each base tree learner, the correlation between each single learner is well reduced, and therefore random forest performs well for many problems and serves as gold standard tool in many studies. However, since tree base learners used to build the forest are vulnerable to imbalanced data, it random forest performs poorly for imbalanced data. In this chapter, we first review the disadvantages of conventional tree learner and then propose ROC Random Forest algorithm which employs the merits of ROC analysis and ensemble classifier structure.

4.1 Node splitting criteria

Since, the traditional decision tree methods, e.g., CART, ID3 and C4.5, chooses the splitting feature and threshold simultaneously, we introduce splitting criteria of our proposed methods first in this section and the feature selection in next section.

More details about decision tree node feature selecting and splitting could be found in **Algorithm 1** and **Algorithm 2**, and in brief, for each feature in the subset, the algorithm split dataset using each possible splitting value and choose the one which maximize the impurity difference between parent node and the resulting children nodes. Entropy and Gini impurity (definition could be found in §**1.2.2**) are calculated for each node.

We introduce ROC and its advantages in **Chapter 3**. Here we proposed a new splitting strategy based on ROC analysis. In ROC space, points in lower left corner of the ROC space represent a too conservative decision. Too little positive decision is made and as a result the sensitivity is only 0.1. If the positive case are important ones, such result is unacceptable. Point on the upper right corner of the ROC space represent a too aggressive decision. Too many positive decision is made and therefore the false positive cases are inflated. We introduce ROC based measure, which combines sensitivity and specificity, to locate a more reasonable operating point.

In mathematics, the harmonic mean is one kind of averages. As it tends strongly toward the smaller element of the pair, it may mitigate the influence of the larger value and increase the influence of the small value. In other words, the larger the difference of the elements in the pair, the smaller the harmonic mean is. It pays more attention to the balance of the pair compared to simple arithmetic mean. Its definition is given below:

$$\text{Harmonic\_Mean}(\text{TPR}, \text{FPR}) = \frac{2 * \text{TPR} * (1 - \text{FPR})}{\text{TPR} + (1 - \text{FPR})} = \frac{2 * \text{Sensitivity} * \text{Specificity}}{\text{Sensitivity} + \text{Specificity}}$$

The 3D plot of Harmonic_Mean surface on ROC space together with its contour figure are shown in **Figure 11**. Comparing to arithmetic mean, the harmonic mean tends to give lower value to unbalanced pair and therefore, in dealing with imbalanced data, lower sensitivity – high specificity pair will be ranked lower and the problem of classifying most cases to majority class is well avoided.

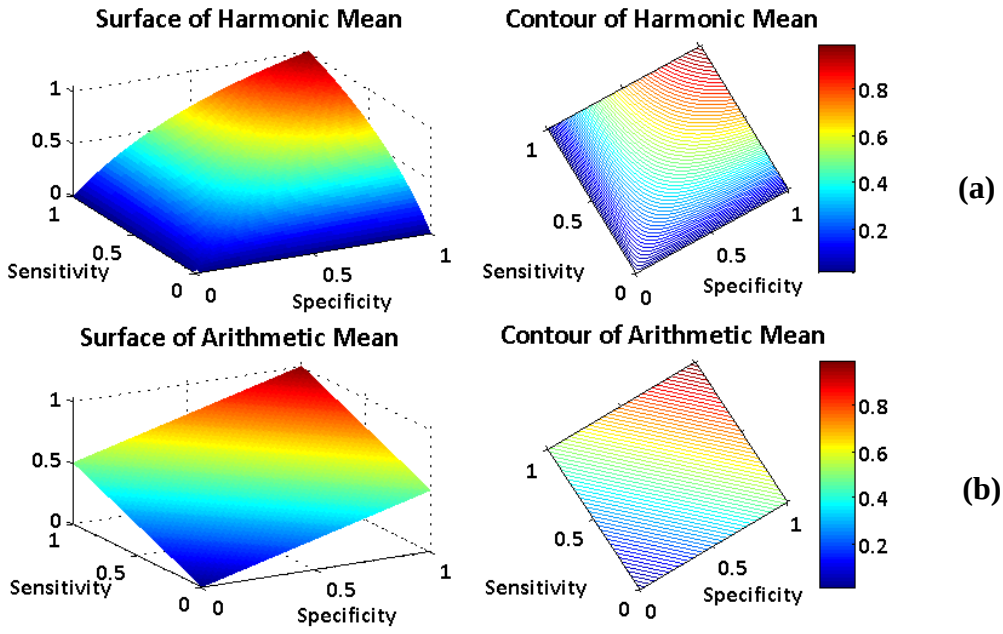


Figure 11 Figure (a) shows the 3D surface plot (left) of harmonic mean over sensitivity-specificity space. The corresponding contour plot is on the right as well as the heat bar. Figure (b) shows the 3D surface plots of arithmetic mean over sensitivity-specificity space, as well as the corresponding contour plot and the heat bar.

4.1.1 The property of proposed method for continuous ROC setting

Following the definition of §**3.3**, we define

$$\alpha = FPR = 1 - specificity$$

$$f(\alpha) = TPR = sensitivity$$

where $\alpha \in (0,1)$, $f(\alpha) \in (0,1)$ and $f(\alpha)$ is an monotonic increasing function in its domain. And therefore:

$$H\big(\alpha, f(\alpha)\big) = \frac{2f(\alpha)(1-\alpha)}{f(\alpha)+(1-\alpha)} = Harmonic\_Mean(TPR, FPR)$$

To find the maximum of $H\big(\alpha, f(\alpha)\big)$ we need to solve the following equation:

$$\frac{\partial H\big(\alpha, f(\alpha)\big)}{\partial \alpha} = 0$$

$$\Rightarrow \frac{\big(2f'(\alpha)(1-\alpha) - 2f(\alpha)\big)\big(f(\alpha) + (1-\alpha)\big) - (f'(\alpha) - 1)(2f'(\alpha)(1-\alpha))}{(f(\alpha) + (1-\alpha))^2} = 0$$

Since the denominator is always positive, the above equation holds if and only if the numerator equals 0 and with simple algebra it could be reduce to

$$f'(\alpha) = \frac{f^2(\alpha)}{(1-\alpha)^2}$$

Empirical ROC curve is mostly modeled and fitted by parametric method and the most common assumption is binormal setting (Hanley 1988, Mets et al., 1998, and Pepe, 2003, Robin, 2011). A normal quantile function ϕ values of sensitivities and specificities is used and the coefficients of the following equation are estimated:

$$\phi^{-1}(sensitivity) = a + b\phi^{-1}(specificity)$$

Mets and Robin apply maximum likelihood estimation and least square estimation for the coefficient respectively.

Under binormal assumption, let us move back to the solution equation, for ideal convex continuous ROC curve, which means $f'(\alpha) > 0$ and $f''(\alpha) < 0$, $f'(\alpha)$ is monotonically decreasing function. For the right hand terms, $f(\alpha)$ is increasing and $(1-\alpha)$ is decreasing and therefore as a whole function the right term is increasing, which guarantee there would be a unique global maximization would be reached. For certain circumstance (discussed in study conducted by Pesce et al., 2010) when the curve is not convex, local maximums could be found and in which case, a unique global maximum

could also be found by comparing the harmonic mean of points which satisfy $f'(\alpha) = \frac{f^2(\alpha)}{(1-\alpha)^2}$.

For non-parametric density method fitting method (Zou et al., 1997), a unique solution may not be existed and therefore, we will choose the left most point which maximize the harmonic mean. And in our realistic data, this usually happens and therefore we employ the same strategy to select the left most local optimal point.

4.2 Node attribute selection criteria

Previous studies have established the use of ROC curve for feature selection, to identify the discriminative attributes in the dataset. Ferri et al. (2002) used AUC as a quality metric to choose nodes in a decision tree. This method selects a feature and split point based on the AUC corresponding to a classifier for every potential class labelling for the induced child nodes. This is not really a ROC curve generated by varying the threshold but by exhausting every possible labelling result to form a convex hull on ROC space and choosing the edge as the ROC curve. For binary classification problem and two child leaves tree structure, this method would be inaccurate. Hossain et al. (2008) conducted a study which also used AUC measure to select a node based on classification performance and then uses the misclassification rate to choose a split point. The classification rate here is the overall accuracy and therefore is not suitable for our imbalanced data. However, we adapt their idea of the first part, using AUC to select splitting attribute.

ROC curve is plotted for each of the pairs formed by each of the genes and the class label. It means treating a single attribute as a classifier and calculating the

classification in term of the sensitivity and specificity by varying the operating point. The idea of treating single attribute as classifier has been already introduced in decision tree algorithm in **Chapter 1**. This is a common procedure in tree based learning algorithms. For each attribute or feature, the AUC is calculated and the one with highest AUC is selected to split the node.

Note that it is possible the AUC could be less than 0.5, and as we introduced in Chapter 3, by simply changing the splitting direction, the corresponding AUC will be larger than 0.5. For calculation convenience, we bring in a new variable to store the splitting direction. And this variable has another function in our algorithm, it will be used to labelling the leaf nodes. Conventional decision tree algorithms simply use the majority classes' label in the node. Hossain's method labels the node positive class if the child node's attribute value is larger than the threshold, otherwise the node would be labeled as negative. By introducing the splitting direction variable, our leaf node labelling strategy is very similar to Hossain's method. If the splitting direction is positive, the labelling process is the same as Hossain's method. If it is negative, we will label the child node negative if the attribute value of the node is larger than the threshold and positive if the attribute value is less than splitting threshold.

Recording splitting direction is of fundamental importance in our study since the positive classification direction contains most information of the positive class. However, in traditional decision tree algorithm, there is no classification information carried to child node. And the node is not labeled until it is determined as a leaf node, whom will be labeled as the majority cases label it contains. We use the following example to show the difference of ROC based method and the traditional information gain or Gini impurity

based method. The dataset information is shown in Table 6. The information gain, Gini impurity and harmonic mean curves by varying splitting threshold are shown in **Figure 12**, in which the dashed line indicate the optimal splitting threshold. For all the method, the dataset will be separated into two subset with threshold to be feature value 0.705. If we decide the induced nodes are leaf node, then for information gain Gini impurity, the two child nodes will be labeled as negative. For ROC method, the corresponding AUC is 0.7778 and therefore cases with feature value larger than 0.705 will be classified as positive class, and we put them in the right child node. The result would be left node will be negative node and right child node will be positive node.

Table 6 Dataset with information of class label and feature score. Instances from number 47 to number 94 are omitted, and they all belongs to negative class and their corresponding feature score decrease from 0.53 to 0.07 with difference 0.01.

Instance	Class	Feature	Instance	Class	Feature	Instance	Class	Feature
1	**n**	1.00	21	**p**	.80	41	**n**	.60
2	**n**	.99	22	**p**	.79	42	**n**	.59
3	**n**	.98	23	**p**	.78	43	**n**	.58
4	**n**	.97	24	**p**	.77	44	**n**	.57
5	**n**	.96	25	**p**	.76	45	**n**	.56
6	**n**	.95	26	**p**	.75	46	**n**	.55
7	**n**	.94	27	**p**	.74	47	**n**	.54
8	**n**	.93	28	**p**	.73	•	•	•
9	**n**	.92	29	**p**	.72	•	•	•
10	**n**	.91	30	**p**	.71	•	•	•
11	**n**	.90	31	**n**	.70	•	•	•
12	**n**	.89	32	**n**	.69	•	•	•
13	**n**	.88	33	**n**	.68	•	•	•
14	**n**	.87	34	**n**	.67	94	**n**	.07
15	**n**	.86	35	**n**	.66	95	**n**	.06
16	**n**	.85	36	**n**	.65	96	**n**	.05
17	**n**	.84	37	**n**	.64	97	**n**	.04
18	**n**	.83	38	**n**	.63	98	**n**	.03
19	**n**	.82	39	**n**	.62	99	**n**	.02
20	**n**	.81	40	**n**	.61	100	**n**	.01

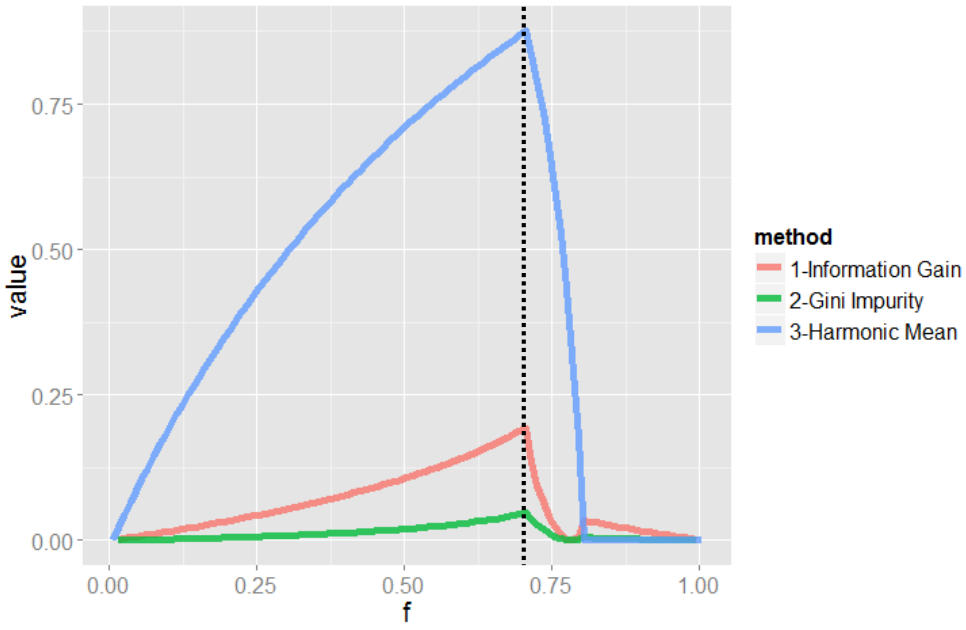


Figure 12 Splitting threshold selection curve. The red curve indicates the information gain with different threshold. The green and blue curves represent the Gini impurity curve and harmonic mean curve. The black dashed line indicate the operating point which maximize the value on these curves and the corresponding feature value is 0.70.

4.3 Stopping Criterion

To decide when to stop growing the tree, AUC of the selected attribute is tested. The value of AUC equals 1 represents that the selected attribute can classify the node perfectly. However, to avoid overfitting problem, we choose **0.98** for the stopping threshold, because we do not want to grow the tree for a smaller subset of the training set.

As we are trying to construct a ROC based Random Forest, only a limit number of attributes will be used to build each tree. Following traditional setting of Random Forest algorithm, attributes those used in parent node will not be used for the child node any more. And therefore, when we ran out of attribute for current node, this node will stop grow.

4.4 ROC Random Forest Algorithm

Now we have all the four main components which are necessary for building a forest, i.e., node attribute selection criteria, node splitting criteria and stopping criteria. We have introduced ROC generating algorithm (**Algorithm 6**) and AUC calculation algorithm (**Algorithm 7**) in Chapter 3. These two algorithms will be frequently referred in the following algorithms.

Algorithm 9 search through all the possible splitting attributes for current node and output the one which has the largest AUC value as well as the splitting direction, ROC curve points and threshold vector.

Algorithm 9 Node_Attribute_Selection

Input(s): $\boldsymbol{X}$, the matrix of training examples; $\boldsymbol{\omega}$, the corresponding label vector; P and N, the number of positive and negative examples; $\boldsymbol{\mathcal{A}}$, the attribute set

Output(s): $\mathcal{A}$, attribute with the highest AUC; α, (TPR, FPR) and $thres$, corresponding splitting direction, ROC points and threshold vector

$max_{\mathcal{A}} \leftarrow 0$
$max_{\alpha} \leftarrow 1$
$max_{\mathrm{ROC}} \leftarrow NULL$
$max_{\mathrm{thres}} \leftarrow -\mathrm{Inf}$
for each attribute $\mathcal{A}_i \in \boldsymbol{\mathcal{A}}$ **do**
    $temp_{ROC}, temp_{thres} \leftarrow \mathrm{ROC\_generating}(\mathcal{A}_i, \boldsymbol{\omega}, P, N)$
    $temp_{AUC} \leftarrow \mathrm{AUC\_calculation}(temp_{ROC})$
    $temp_{\alpha} \leftarrow 1$
    if $temp_{AUC} < 0.5$ **then**
        $temp_{AUC} = 1 - temp_{AUC}$
        $temp_{\alpha} = -1$
    end if
    if $temp_{AUC} > \max\_\mathcal{A}$ **then**
        $max_{\mathcal{A}} = temp_{AUC}$
        $max_{\alpha} = temp_{\alpha}$
        $max_{\mathrm{ROC}} = temp_{ROC}$
        $max_{\mathrm{thres}} = temp_{thres}$
    end if
end for
return $max_{\mathcal{A}}, max_{\alpha}, max_{\mathrm{ROC}}, max_{\mathrm{thres}}$
end

Considering the ROC curve with AUC value less than 0.5, when we change the splitting direction, all the previous "positive" decision will become "negative" and previous "negative" decision will become "positive". Considering a single point on the ROC space, the following changes will occur:

Table 7 Previous classification

		Predicted label	
		Positive	Negative
Actual label	Positive	a	b
	Negative	c	d

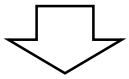


Table 8 Classification after changing splitting direction

		Predicted label	
		Positive	Negative
Actual label	Positive	b	a
	Negative	d	c

Therefore, we can simply get the ROC curve after changing splitting direction by using information from the previous curve.

$$new\_TPR = \frac{b}{a+b} = 1 - previous\_TPR$$

$$new\_FPR = \frac{d}{c+d} = 1 - previous\_FPR$$

Actually, the new point is symmetric to the previous one with respect to the center point (0.5, 0.5) in the ROC space. With this fact, it could be easy to verify our previous corollary, the AUC of the new ROC equals $(1 - AUC\_previous)$. Based on this result, **Algorithm 10** implements the calculation of the splitting threshold.

Algorithm 10 Node_Splitting_threshold

Input(s): $\mathcal{A}_\alpha$, $\mathcal{A}_{(TPR,FPR)}$ and $\mathcal{A}_{thres}$, the splitting direction, ROC points and corresponding threshold associating with attribute $\mathcal{A}$, which has the largest AUC

Output(s): $thres$, final splitting threshold for attribute $\mathcal{A}$

1. $cur_{hmean} \leftarrow -\text{Inf}$
2. $thres \leftarrow -\text{Inf}$
3. **for** each ROC points $(TPR_i, FPR_i) \in \mathcal{A}_{(TPR,FPR)}$ **do**
4. **if** $\mathcal{A}_\alpha == 1$ **then**
5. $temp_{hmean} = \text{Harmonic\_Mean}(TPR_i, FPR_i)$
6. **else then**
7. $temp_{hmean} = \text{Harmonic\_Mean}(1 - TPR_i, 1 - FPR_i)$
8. **end if**
9. **if** $temp_{hmean} > cur_{hmean}$ **then**
10. $cur_{hmean} = temp_{hmean}$
11. $thres = \mathcal{A}_{thres_i}$
12. **end if**
13. **end for**
14. **return** $thres$
15. **end**

The following **Algorithm 11** will be employed to generate the ROC trees. For each node in ROC tree, we have the splitting threshold and the splitting direction and therefore in the tree building process it could be easy to fix the left node to be negative labelling node and the right node to be positive labelling node. By doing so, it could be easy to retrieve the labelling information of current node if the stopping criteria was met in current node.

Algorithm 11 ROC_tree

Input(s): $\boldsymbol{X}$, the matrix of training examples; $\boldsymbol{\omega}$, the corresponding label vector; P and N, the number of positive and negative examples; $\boldsymbol{\mathcal{A}}$, the attribute set

Output(s): $\mathcal{T}$, the final ROC tree

1. **if** $(\boldsymbol{X}, \boldsymbol{\omega}, P, N)$ contains no record **then**
2. **return** a single NULL node
3. **end if**
4. **if** $(\boldsymbol{X}, \boldsymbol{\omega}, P, N)$ consists of records all with the same label value **then**
5. **return** a single leaf node labelling this value
6. **end if**
7. $(\mathcal{A}_\alpha, \mathcal{A}_{(TPR,FPR)}, \mathcal{A}_{thres}) \leftarrow \text{Node\_Attribute\_Selection}(\boldsymbol{X}, \boldsymbol{\omega}, P, N, \boldsymbol{\mathcal{A}})$

8. $thres \leftarrow$ Node_Splitting_threshold($\mathcal{A}_\alpha$, $\mathcal{A}_{(TPR,FPR)}$, $\mathcal{A}_{thres}$)
9. **set** $(\boldsymbol{X}, \boldsymbol{\omega}, P, N)_{left}$ as the negative child node and $(\boldsymbol{X}, \boldsymbol{\omega}, P, N)_{right}$ as the positive child node based on $\mathcal{A}_\alpha$ and $thres$
10. Recursively apply ROC_tree to subsets $\left(\boldsymbol{X}, \boldsymbol{\omega}, P, N, \boldsymbol{\mathcal{A}_{left}}\right)_{left}$ and $\left(\boldsymbol{X}, \boldsymbol{\omega}, P, N, \boldsymbol{\mathcal{A}_{right}}\right)_{right}$ until they are empty or the stopping criteria are met. Here $\boldsymbol{\mathcal{A}_{left}}$ and $\boldsymbol{\mathcal{A}_{right}}$ are the randomly selected attributes subset for the left tree and right tree.
11. **return** ROC tree $\mathcal{T}$
12. **end**

After we get all these basic ROC tree learners, the class prediction of a test point could be induced by aggregating all the decision from these learners, i.e., majority vote. A probabilistic value, the percentage of positive class decision among these learners, will be reported (**Algorithm 12**).

Algorithm 12 ROC_Random_Forest

Input(s): $\boldsymbol{X}$, the matrix of training examples; $\boldsymbol{\omega}$, the corresponding label vector; $\boldsymbol{N_t}$, the number of trees to be generated; $\boldsymbol{N_a}$, the number of attributed needed for each node

Output(s): $\boldsymbol{\mathcal{F}}$, the final forest

1. **set** $\boldsymbol{\mathcal{F}}$ to NULL
2. **for** $i = 1$ **to** $\boldsymbol{N_t}$
3. generate bagging sample $(X_i, \omega_i, P_i, N_i)$
4. $tree_i \leftarrow$ROC_tree(X_i, ω_i, P_i, N_i), $\boldsymbol{N_a}$ is used in this step
5. append $tree_i$ to $\boldsymbol{\mathcal{F}}$
6. **end**
7. **return** $\boldsymbol{\mathcal{F}}$

4.5 Complexity of algorithm: ROC Random Forest

Let us consider a dataset $\mathcal{D}$ of N examples, where each example comprises M attributes. In implementation, for each tree, after a bootstrapping sample is generated, the dataset was first sorted by each attribute with the sorted index stored. Then for the node splitting phase, we do not need to sort each attribute again. This will take $O(NlogN)$ for each attribute and $O(m_{try}NlogN)$ for the attribute. For attribute selection and splitting

threshold selection phase, we only need linear time $O(N)$ scan to decide the attribute and threshold. Comparing to $O(m_{try}NlogN)$, $O(N)$ is negligible. And therefore, for building each tree node, ROC Random Forest has the same complexity with regular Random Forest algorithm. If we build n_{tree} trees, it has the same time complexity with regular Random Forest algorithm which is $O(n_{tree}m_{try}Nlog^2N)$ (Louppe, 2014).

4.6 Hybrid ROC Random Forest for categorical data

The innovation part of the ROC Random Forest lies in the node splitting algorithms. It first looks for the attribute that maximizes the AUC in the current feature subset and then uses the harmonic mean method to search for the splitting threshold for the current node. However, the ROC Random Forest cannot deal with nominal categorical features. The essence of the ROC analysis is to consider the ranking of the instances, and there is no ranking information in nominal data. This drawback greatly limits the application of our proposed ROC Random Forest method.

Ferri et al. (2003) argued that by exhausting all the possible partitions of the categorical data, the corresponding decision point in ROC space will form a convex hull. And they treat the induced curve by connecting the outermost decision point as the "ROC" curve. However, this would lead to exponential computational complexity ($2^{N-1}-1$ for N-level attribute), which would quickly become highly prohibitive to explore if N is too large. Similar problem is encountered in decision tree growing with categorical variable. Fortunately, for decision tree problem, this exponential complexity can be reduced from $2^{N-1}-1$ partitions to $N-1$ partitions with theoretical support (Fisher, 1958 and Breiman et al. 1984). This method is introduced below.

Let us reorder categories of variable $\mathcal{A}$ ($\mathcal{A} \in \{l_1, \dots, l_N\}$) such that:

$$p(\omega_1|\mathcal{A} = l_{i_1}) \leq p(\omega_1|\mathcal{A} = l_{i_2}) \leq \cdots \leq p(\omega_1|\mathcal{A} = l_{i_N})$$

where ω_1 indicates the negative class (for binary classification problem). Then one of the $N-1$ subsets $B = \{l_{i_1}, l_{i_2}, \dots, l_{i_h}\}$, $h = 1, \dots, N-1$ defines a binary partition of the node samples into:

$$Node_{left}^{B} = \{(X, \omega)|(X, \omega) \in Node_{parent}, X_{\mathcal{A}} \in B\}$$

$$Node_{right}^{B} = \{(X, \omega)|(X, \omega) \in Node_{parent}, X_{\mathcal{A}} \in \bar{B}\}$$

where $B \cup \bar{B} = \{l_1, \dots, l_N\}$, and such partition guarantee a maximum of impurity drop. The basic idea of this theory is using the category frequency in a certain class to rank different categorical levels, and the induced nodes following such partition also guarantee a maximum impurity drop, which satisfies the impurity measure requirement given by Breiman et al. (1984).

Even though this theory greatly reduce the computational complexity, we cannot directly apply it to either harmonic mean. Because harmonic mean is not an impurity measure and does not satisfy the impurity measure requirement, therefore trying to maximize the harmonic mean in these partitions does not guarantee an optimal splitting. Simply combining such categorical splitting with the proposed continuous ROC splitting could be a solution, however, such combination will introduce imbalanced bias into the framework, as we shown before, the decision tree method does not have splitting direction. Here we propose a method which employ a simple strategy to help decide the splitting direction:

$$if\ \frac{sensitivity + specificity}{2} > 0.5\ then\ \begin{cases} Node_{left}: negative \\ Node_{right}: positive \end{cases}$$

$$if\ \frac{sensitivity + specificity}{2} < 0.5\ then\ \begin{cases} Node_{left}: positive \\ Node_{right}: negative \end{cases}$$

Simply put, $\frac{sensitivity+specificity}{2} > 0.5$ indicates that the decision point is above the random guess line, and thus we will change the splitting direction. For each categorical attribute in the node, we use this method to obtain the (sensitivity, specificity) pair of the threshold that generates the maximum impurity drop. For each categorical variable, we compare the harmonic mean of its corresponding decision point and choose the attribute with the highest harmonic mean as the potential splitting candidate. For the continuous attributes, following the ROC splitting method introduced before, we can also choose a splitting attribute. We compare their harmonic mean and choose the one with the higher value as the final splitting threshold for the current node. If either continuous or categorical attribute is absent, then the candidate is used. This hybrid ROC RF combining the continuous ROC splitting and original categorical splitting with splitting direction we propose the following node splitting algorithm:

Algorithm 13 Combination_splitting

Input(s): $\boldsymbol{X}$, the matrix of training examples; $\boldsymbol{\omega}$, the corresponding label vector; $\boldsymbol{\mathcal{A}_{con}}$, the categorical attribute set; $\boldsymbol{\mathcal{A}_{cat}}$, the continuous attribute set; P and N, the number of positive and negative examples;

Output(s): $\mathcal{A}_{thres}^{con}$, final splitting threshold for attribute $\mathcal{A}_{high}^{con}$ or $(B, \bar{B})$, final splitting set for attribute $\mathcal{A}_{high}^{cat}$

1. $\mathcal{A}_{high}^{con} = NULL$
2. $\mathcal{A}_{high}^{cat} = NULL$
3. **if** $\boldsymbol{\mathcal{A}_{con}}$ is not empty **then**
4. $(\mathcal{A}_{high}^{con}, \mathcal{A}_{(TPR,FPR)}, \mathcal{A}_{thres}) \leftarrow$ Node_Attribute_Selection$(\boldsymbol{X}, \boldsymbol{\omega}, P, N, \boldsymbol{\mathcal{A}_{con}})$
5. $\mathcal{A}_{thres}^{con} \leftarrow$ Node_Splitting_threshold$(\mathcal{A}_{high}^{con}, \mathcal{A}_{(TPR,FPR)}, \mathcal{A}_{thres})$
6. calculate Harmonic_Mean$_{\text{con}}$ based on $\mathcal{A}_{thres}^{con}$

7. **end if**
8. **if** $\boldsymbol{\mathcal{A}_{cat}}$ is not empty **then**
9. $impurity\_drop = -\infty$
10. $B = NULL$
11. $\bar{B} = NULL$
12. **for** each $\mathcal{A}_i$ in $\boldsymbol{\mathcal{A}_{cat}}$ **do**
13. calculate the partition $(B_{temp}, \bar{B}_{temp})$ which yields the largest impurity
14. drop $temp\_drop$, record current $(sensitivity, specificity)$
15. **if** $\frac{sensitivity+specificity}{2} < 0.5$ **then**
16. swap$(B_{temp}, \bar{B}_{temp})$
17. **end if**
18. **if** $temp\_drop > impurity\_drop$ **then**
19. $(B, \bar{B}) = (B_{temp}, \bar{B}_{temp})$
20. $impurity\_drop = temp\_drop$
21. $\mathcal{A}_{high}^{cat} = \mathcal{A}_i$
22. **end if**
23. **end for**
24. calculate Harmonic_Mean$_{\text{cat}}$ based on $\mathcal{A}_{high}^{cat}$ and $(B, \bar{B})$
25. **end if**
26. **if** $\mathcal{A}_{high}^{con} == NULL$ or Harmonic_Mean$_{\text{cat}}$ > Harmonic_Mean$_{\text{con}}$**then**
27. **return** $(B, \bar{B})$ and $\mathcal{A}_{high}^{cat}$
28. **else then**
29. **return** $\mathcal{A}_{thres}^{con}$ and $\mathcal{A}_{high}^{con}$
30. **end if**
31. **end**

Then **Algorithm 11** and **Algorithm 12** will call this **Algorithm 13** to build the ROC Random Forest. To differentiate it from ROC Random Forest, we name it simply as the Hybrid ROC Random Forest. In Chapter 5, we compare it with regular Random Forest algorithm using datasets with categorical predicting variables.

4.7 Complexity of algorithm: Hybrid ROC Random Forest

For each node, the difference between ROC Random Forest and Hybrid ROC Random Forest is that it considers both continuous variable splitting and categorical variable splitting, and then compare the maximum available harmonic mean of the two. For the continuous variable, it takes $O(m_{con}NlogN)$. For categorical part, it takes linear time

$O(N)$ to get the categorical conditional probability and $O(logN_{level})$ for ranking, in total with m_cat variables, it takes $O(m_{cat}NlogN_{level})$ which is negligible comparing to the continuous case. After this, it only take constant time to calculate the harmonic mean and do the comparison. Therefore, the total time complexity of Hybrid ROC Random Forest is also similar to regular Random Forest.

Chapter V: Validation: Results and Discussion

In this chapter, we introduce how we conduct the validation experiment and how the algorithms are applied to deal with practical imbalanced data problem.

5.1 ROC analysis for imbalanced data classification threshold correction

In **Chapter 3**, we give detailed introduction of ROC analysis and also explain why it could be used to tackle class imbalanced data. In §**4.1**, we introduce how to use self-defined harmonic mean of sensitive and specificity to pick the "optimized" operating point from the ROC space. For this purpose, in this section, we conduct experiments to show the feasibility of combining the two methods to choose a better cut-off threshold.

5.1.1 Dataset

The experiment was conducted on a Computed Tomography (CT) Colonography database of 49 scans from 25 patients with polyps size from 6-22mm. The data acquisition process is also introduced in details in paper (Song et al., 2014). 786 initial polyp candidates (IPCs) were obtained, among which 64 are true polyps (TPs). Twenty-one geometric features and density features were calculated on each of the extracted IPCs' volume. To be specific, density features include statistical information, i.e., mean, variance, entropy, etc., of the CT values, and geometric features include statistical information of the shape index and curvedness. Volume-based features, e.g., number of region growing seeds, axis ratio, disk-likeness and highlighting ratio, are also included. In summary, we have 786 observations, each of which have 21 explanatory features. Among the 786 observations, 64 are true polyps, which makes the dataset very

unbalanced. The corresponding class imbalanced ratio of the dataset is 0.0815. All the following experiments were based on this imbalanced database.

5.1.2 Basic classifiers

We choose the original Random Forest algorithm, which employs Gini index for node splitting, and SVM as the basic classifiers to verify the role of ROC analysis in helping find optimal classification threshold. In implementation, the widely used randomForest function in the well-known package random forest (Liaw and Wiener, 2002), was employed with the number of tree set to be 5000, and the number of attributes used for splitting each node was determined by out of bag error rate. For SVM, the widely used SVM package LIBSVM (Chang and Lin, 2011) with radial basis function (RBF) kernel was employed in this study. Following the guideline of LIBSVM, the two parameters (cost and gamma) in the RBF kernel were determined by a grid search process (fivefold cross-validation) in the training step.

5.1.3 Traditional data correcting technologies

For comparison purpose, two common approaches introduced in **Chapter 2** to tackle the imbalanced data classification problem, i.e., the cost-sensitive learning (weighted random forest and weighted SVM) and down-sampling technique, were implemented as references or baseline and compared with our proposed strategies. In the cost-sensitive learning, for both weighted Random Forest and weighted SVM, the best fitting weight was automatically searched from 0 to 1 by stepwise 0.02 in the training phase. For the down-sampling technique, we down sampled the majority (non-TPs) class to make the dataset more balanced with different levels, i.e., 75%, 50% and 25% of the non-TPs cases were sampled in the training step.

In details, for each classifier, taking Random Forest for example, we evaluate it with 5 different settings, i.e., un-weighted Random Forest with imbalanced data, weighted Random Forest with imbalanced data, weighted Random Forest with 75% down sampling, weighted Random Forest with 50% down sampling and weighted Random Forest with 25% down sampling.

5.1.4 Three-way cross-validation

For each the experiments we described above, the original dataset was randomly partitioned into three subsets with the same class imbalance ratio as the original dataset, i.e., 0.081425 as the original dataset. One subset (subset 1) was reserved for testing purpose, and the other two subsets (subset 2 and subset 3) were treated as the training set. For the original threshold method, we treated both subset 2 and subset 3 as training set to train the classifier. Regular threshold (0.5 for the probability output) was applied on subset 1 to draw results. For out proposed operating point chosen strategies, we used subset 2 as classifier training set and subset 3 as cut-off optimization training set and applied the selected threshold on the testing set. Then we rotated the role of the three subsets (6 permutations in total) and outputted the average results, which we call three-way cross validation. To minimize the bias of one time running, the random partition process was repeated 100 times and the average results were outputted.

5.1.5 Results and discussion

Table 9 shows the averaged AUC information, i.e., mean and standard deviation of Random Forest and SVM with different settings which were introduced in **§5.1.3**. **Figure 14** and **Figure 15** show the averaged ROC curves of Random Forest and SVM with different settings. We can see from the figures and tables that Random Forest with

different settings all achieved high AUC values (0.96 for Random Forest and 0.995 for SVM) and the standard deviation for both classifiers are relatively small, around 0.02, which indicates that these two classifiers are capable to classify the imbalanced data. However, the classification accuracy with the original threshold is not favorable – very low prediction accuracy in the minority TP class, as shown as the green marker in **Figure 13** and **Figure 14**. Taking the weighted original imbalanced data for example, when using the original threshold, Random Forest only detected half polyps (0.52 sensitivity), in testing set while SVM detected around 60% of polyps (0.62 sensitivity).

Table 9 Averaged Random Forest and SVM AUC information of the 100 runs

	Averaged AUC of Random Forest		Averaged AUC of SVM	
	Mean	Std.	Mean	Std.
Imbalanced data, un-weighted	0.9614	0.0201	0.9493	0.0247
Imbalanced data, weighted	0.9608	0.0197	0.9523	0.0240
75% down-sampling, weighted	0.9613	0.0194	0.9547	0.0215
50% down-sampling, weighted	0.9600	0.0201	0.9561	0.0208
25% down-sampling, weighted	0.9600	0.0204	0.9543	0.0208

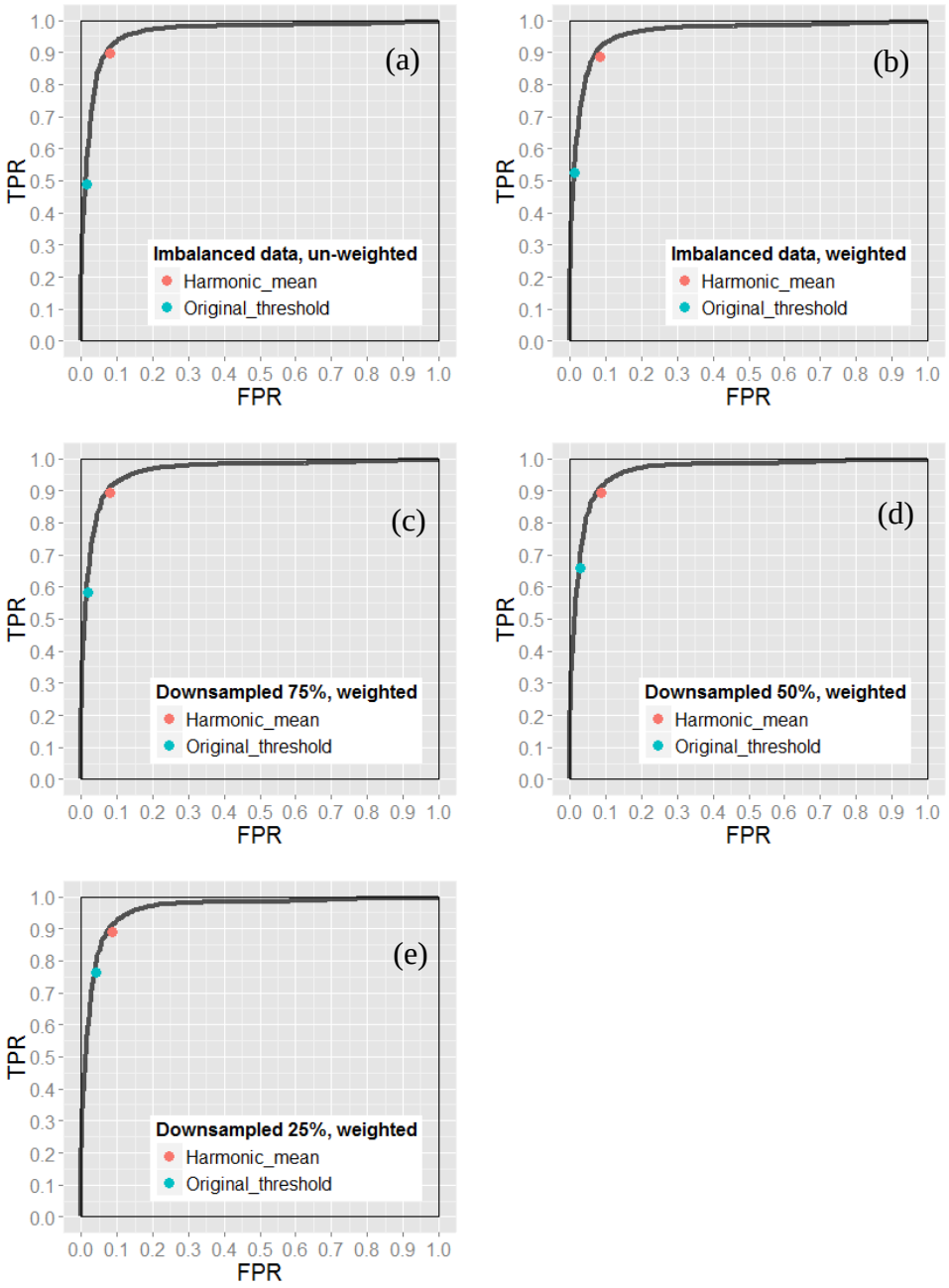


Figure 13 Averaged ROC curve of Random Forest results. (a)-(e) show averaged ROC curves of un-weighted Random Forest with imbalanced data, weighted RFs with imbalanced data, weighted Random Forest with 75% down-sampling data, weighted Random Forest with 50% down-sampling data and weighted RFs with 25% down-sampling data. The red, blue circle marker represent results of regular 0.5 threshold, and harmonic mean respectively. The averaged ROC curves was conducted according to the horizontal axis, where the linear interpolation was employed when needed.

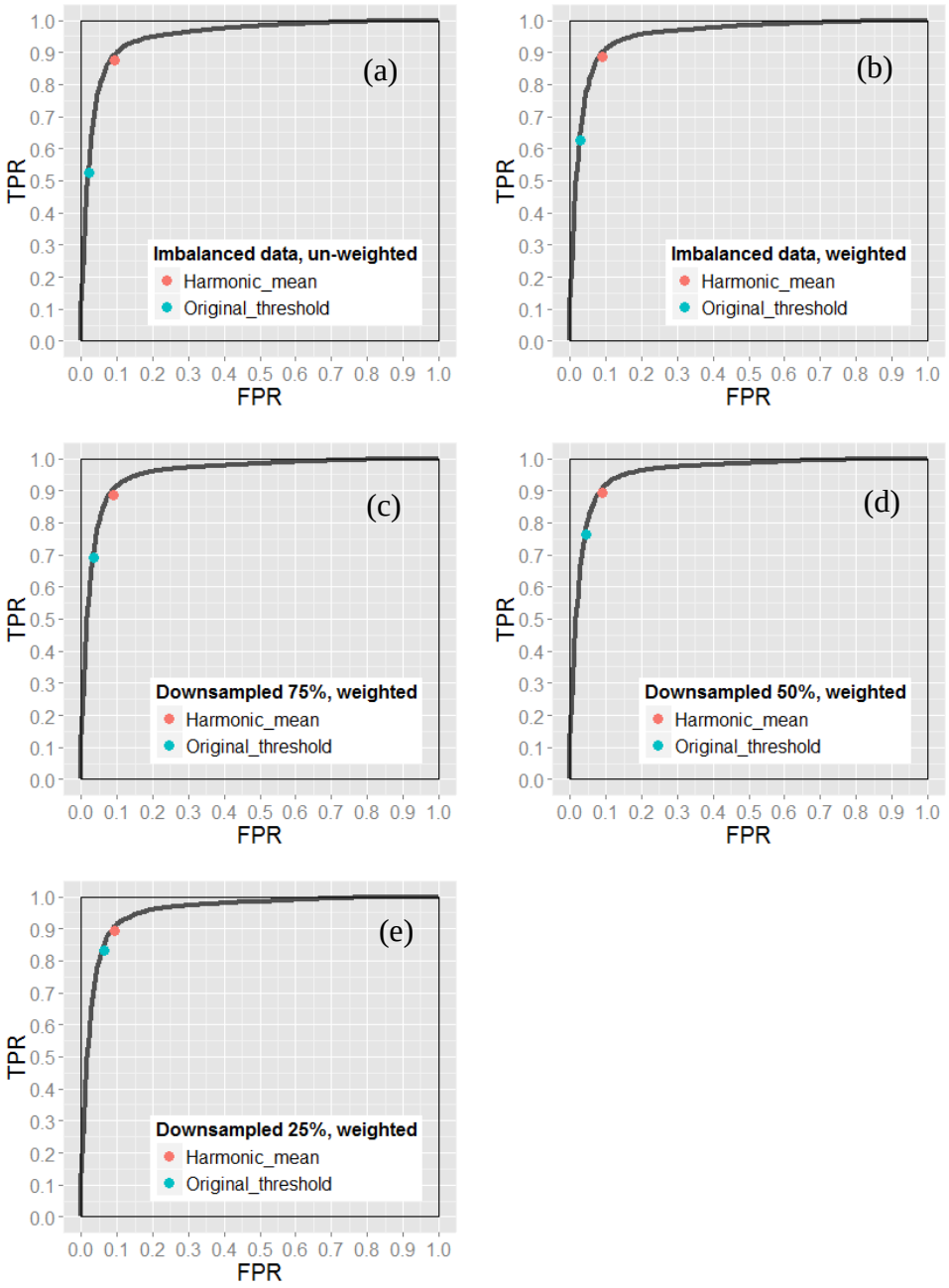


Figure 14 Averaged ROC curve of SVM results. (a)-(e) show averaged ROC curves of un-weighted SVM with imbalanced data, weighted SVM with imbalanced data, weighted SVM with 75% down-sampling data, weighted SVM with 50% down-sampling data and weighted SVM with 25% down-sampling data. The red, blue circle marker represent results of regular 0.5 threshold, and harmonic mean respectively. The averaged ROC curves was conducted according to the horizontal axis, where the linear interpolation was employed when needed.

Table 10 and **Table 11** show the classification results of Random Forest and SVM with different threshold selection strategies, where the classification results, i.e., sensitivity, specificity and overall accuracy, with the original threshold, i.e., 0.5 for the probability score of both Random Forest and SVM, and the proposed harmonic mean method. The thresholds (averaged over 100 runs) chosen by the harmonic mean method is also listed in the tables. As we expected, it is observed from the last column of **Table 10** and **Table 11** that the operating point did shift due to the imbalance of the data, and the original threshold was no longer suitable to draw the decision. From both tables, it was also observed that both the weighted classifier (i.e., the cost sensitivity learning) and the down-sampling technique did help to some degree in improving the sensitivity (compared to the original threshold). For example, for the weighted classifier, only moderate improvement can be observed, about 0.04 increasing for Random Forest and 0.10 for SVM, for the outcomes of 0.52 for Random Forest and 0.62 for SVM (see the first two rows arranged along the left column). The down-sampling technique showed improvements as the down-sampling level went up, and the classification result with 25% down sampling (see the last row arranged along the left column) is very close to our proposed strategies (similar results can be observed in study of Maloof et al. (2003)). However, there are concerns with the down-sampling technique: the first one is the concern about the information loss because it only uses part of the training data; the second one is the concern that there is not any rule to determine how much we should down sample the majority class.

The gain by our proposed strategies is noticeable as demonstrated by **Table 10** and **Table 11**. Taking the weighted original imbalanced data for example, when using our proposed harmonic mean thresholding method, the sensitivity improved from 0.52 to

nearly 0.885 for Random Forest and for SVM it was from 0.625 to nearly 0.88; meanwhile, a relatively high specificity level was retained, i.e., above 0.90 sensitivity for both classifiers. The corresponding ROC points (the blue dot markers) shown in **Figure 13** and **Figure 14** indicate that our proposed method choose the operating point very close to the upper-left corner.

In all the experiments, we always used the original class ratio in all the testing sets, which is 0.0814, and with different levels of class ration in the training sets, i.e., the original imbalanced data (ratio = 0.0814), 75% down sampling (ratio=0.1057), 50% down sampling (ratio=0.1506) and 25% down sampling (ratio=0.2618). The outcomes showed that our proposed strategies deliver consistent good performances, i.e., around 0.88 sensitivity with 0.90 specificity for both classifiers. This indicates that the performance of harmonic mean method is invariant to different class ratios. Meanwhile, the result also indicates that the method has the ability to be combined with other imbalanced data correcting techniques, such as the cost-sensitive learning, based on the fact that the outcome from the un-weighted data also showed improvement.

Table 10 Averaged Random Forest classification results with the original threshold and cut-off chosen by the proposed harmonic mean method

		Averaged results			
		Sensitivity	Specificity	Accuracy	Threshold
Imbalanced data, un-weighted	Original threshold	0.4884 ±0.1177	0.9855 ±0.0081	0.9450 ±0.0104	0.5
	Harmonic mean	0.8957 ±0.0802	0.9186 ±0.0425	0.9167 ±0.0370	0.1198 ±0.0601
Imbalanced data, weighted	Original threshold	0.5243 ±0.1226	0.9862 ±0.0079	0.9486 ±0.0106	0.5
	Harmonic mean	0.8876 ±0.0796	0.9174 ±0.0384	0.9150 ±0.0328	0.1115 ±0.0627
75% down-sampling, weighted	Original threshold	0.5835 ±0.1151	0.9825 ±0.0090	0.9500 ±0.0110	0.5
	Harmonic mean	0.8927 ±0.0756	0.9187 ±0.0374	0.9165 ±0.0321	0.1484 ±0.0749
50% down-sampling, weighted	Original threshold	0.6591 ±0.1198	0.9720 ±0.0126	0.9465 ±0.0126	0.5
	Harmonic mean	0.8933 ±0.0818	0.9136 ±0.0392	0.9120 ±0.0334	0.2184 ±0.0894
25% down-sampling, weighted	Original threshold	0.7649 ±0.1026	0.9592 ±0.0164	0.9434 ±0.0147	0.5
	Harmonic mean	0.8907 ±0.0805	0.9139 ±0.0387	0.9120 ±0.0334	0.3245 ±0.1066

Table 11 Averaged SVM classification results with the original threshold and cut-off chosen by the proposed harmonic mean method.

		Averaged results			
		Sensitivity	Specificity	Accuracy	Threshold
Imbalanced data, un-weighted	Original threshold	0.5257 ±0.1645	0.9772 ±0.0111	0.9404 ±0.0117	0.5
	Harmonic mean	0.8756 ±0.0860	0.9083 ±0.0388	0.9056 ±0.0337	0.3028 ±0.0450
Imbalanced data, weighted	Original threshold	0.6252 ±0.1559	0.9700 ±0.0136	0.9419 ±0.0121	0.5
	Harmonic mean	0.8848 ±0.0872	0.9088 ±0.0398	0.9068 ±0.0347	0.3414 ±0.0617
75% down-sampling, weighted	Original threshold	0.6908 ±0.1390	0.9647 ±0.0136	0.9424 ±0.0124	0.5
	Harmonic mean	0.8854 ±0.0918	0.9109 ±0.0392	0.9088 ±0.0339	0.3690 ±0.0632
50% down-sampling, weighted	Original threshold	0.7642 ±0.1252	0.9547 ±0.0168	0.9392 ±0.0139	0.5
	Harmonic mean	0.8928 ±0.0880	0.9087 ±0.0406	0.9066 ±0.0352	0.4069 ±0.0286
25% down-sampling, weighted	Original threshold	0.8307 ±0.1086	0.9367 ±0.0249	0.9281 ±0.0211	0.5
	Harmonic mean	0.8918 ±0.0860	0.9077 ±0.0420	0.9064 ±0.0369	0.4513 ±0.0698

5.2. Evaluation of ROC Random Forest

In this section, we use 18 imbalanced dataset, downloaded from UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/index.html), to evaluate the performance of our proposed ROC Random Forest. We also evaluated the performance of original Random Forest based on different settings, i.e., cost-sensitive setting, Tomek links sampling, CNN sampling, ENN sampling, NCL sampling, OSS sampling, SMOTE sampling, SMOTE+Temok sampling and SMOTE+ENN sampling. Extensive experiments are designed and averaged results are reported.

5.2.1 Dataset

The datasets used in this evaluation are downloaded from UCI repository. All of them are binary classification data. With proper adjustment, e.g., proper binary labelling, all the dataset are imbalanced and binary labeled data. Detailed information is shown in **Table 12**.

The UCI Machine Learning Repository is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine analysis of machine learning algorithms (Bache et al., 2013). It has been widely used by students, educators, and researchers all over the world as a version of the web site was designed in 2007. Since our ROC Random Forest in current stage is only suitable for continuous feature variable and binary class variable problem, only limited dataset could be used. We select 18 popularly used dataset, which contains real value features and binary label class. Certain data preparation was done to the dataset before we send them to the classifiers. The first one is the minority class suppression, we randomly down sampled the minority class to keep an around 0.1 class-

imbalance ratio. The second one is label binarilization. Take the wine dataset for example, the quality is measured by ordinal categorical variable, taking value from 1 to 10. We label level 8-10 as "good" quality, i.e., positive class, and the rest as "bad" quality, i.e., negative class. It does not necessarily indicates the quality is really bad, but relatively bad. The third one is missing value substitution. There are many missing values in ozone level data, and we substitute the missing value with column mean.

Table 12 Data information of the 18 dataset from UCI repository

Dataset	**Positive number**	**Negative number**	**Imbalance ratio**	**Feature number**	**Source**
red_wine_quality	217	1382	0.1357	11	Cortez et al., 2009
white_wine_quality	180	4718	0.0367	11	Cortez et al., 2009
breast_cancer	80	357	0.1831	30	Wolberg et al., 1990
connectionist_bench	16	97	0.1416	60	Bache et al., 2013 (UCI)
ozone_level	73	2463	0.0288	72	Bache et al., 2013 (UCI)
ionosphere	30	225	0.1176	34	Sigillito et al., 1989
pima_diabetes	60	500	0.1071	8	Bache et al., 2013 (UCI)
spect	30	212	0.1240	44	Bache et al., 2013 (UCI)
vertebral	30	210	0.1250	6	Bache et al., 2013 (UCI)
breast tissue	21	95	0.1810	9	Bache et al., 2013 (UCI)
haberman	40	225	0.1509	3	Bache et al., 2013 (UCI)
banknote	100	762	0.1161	4	Bache et al., 2013 (UCI)
magic	1000	12332	0.0750	10	Bache et al., 2013 (UCI)
page block	560	4913	0.1023	10	Bache et al., 2013 (UCI)
parkinsons	20	147	0.1198	19	Bache et al., 2013 (UCI)
seismic bump	170	2414	0.0658	14	Bache et al., 2013 (UCI)
secom	104	1463	0.0709	590	Bache et al., 2013 (UCI)
seeds	20	140	0.1250	7	Bache et al., 2013 (UCI)

5.2.2 Learning methods

Purpose of the experiment in this section is to compare performance of the proposed ROC Random Forest with the original Random Forest in terms of dealing with

class imbalanced data. At the same time, we also compare the classification performance of ROC Random Forest with original Random Forest with cost sensitive setting and sampling strategy. The cost sensitive setting is similar to the one we use in §**5.1**. For the down sampling setting, we randomly down sampled the majority class to artificially balance the data, i.e., dataset with imbalanced ratio of 0.5. Meanwhile, as we introduced in **Chapter 2**, we employ various kind of informed sampling algorithms, i.e,. CNN, ENN, Tomek links, NCL, OSS, SMOTE+Tomek and SMOTE+ENN, in order to generate clean or balanced data. In summary, for each dataset, we record the classification performance of the 11 classifier settings, ROC Random Forest with original data, weighted Random Forest with original data, Random Forest with randomly down sampling, Random Forest with CNN under sampling, Random Forest with ENN under sampling, Random Forest with Tomek links under sampling, Random Forest with NCL under sampling, Random Forest with OSS under sampling, Random Forest . 2000 tree is built for each forest, and for each node square root of attribute numbers is randomly selected for splitting.

5.2.3 Two-way random splitting

Comparing to the three-way cross validation §5.1, we do not need an additional run to selection the splitting threshold, which is also an advantage of ROC Random Forest, thus we use a two-way random splitting strategy. For each experiment, the dataset is randomly partitioned into two subsets, i.e., training set and testing set. And both sets have the same class imbalanced ratio. We used training set to build the classification model and use the testing set, considered as holdout data, to validate the model. We record the ROC curve for each classification as well as the accuracy (all with

0.5 decision threshold). To minimize the bias of one time running, the random partition process was repeated 100 times and the average results were outputted.

5.2.4 Results and discussion

Table 13 shows the classification results. For each dataset, the averaged AUC, accuracy, sensitivity and specificity are listed for each of the four classification setting. Accuracy, sensitivity and specificity are obtained using 0.5 as splitting threshold. **Table 14** gives summary result of **Table 13**. From the table we can see that for 12 of the 18 imbalanced dataset, ROC Random Forest produce the highest averaged AUC value. Such results indicate that ROC RF is very accurate classifier and performs robustly for various kind of imbalanced data. In 3 of the 12 dataset, i.e., ozone level, magic and secom, ROC RF significantly outperform other classifiers. In 6 of the 12 dataset, ROC RF outperform most the classifiers (>=5). Within the rest 3 dataset, i.e., harberman, page block and seeds, we could observe that the performance of all classifiers are really close. For the rest 6 dataset which the averaged AUC of ROC RF are not the best, in 4 of the 6, its averaged AUC is not significant different with the best performer and only for 2 of them it losses the competition, in vertebral data it losses to OSS and in the seismic bump data it losses to smoteRF. For the rest Random Forest classification settings, smoteRF method and ossRF show up in the first 2 places more than 4 times. It first implies that smote is very useful to construct balanced distribution and this did not reduce the classifier power in terms of AUC. For ossRF, which removes majority class examples either close to the minority class or distant from the decision border, and such process do help RF improve its performance. Since Tomek links, ENN, CNN, NCL and OSS not really generate balanced data, they either remove data distant from the decision boundary,

close to the border or intrude in the minority class, for most dataset, after such examples are removed the class distribution may still imbalanced. There is no strong trend in these sampling settings (except OSS). For weighted RF, it actually very similar to random over sampling data, this is also discussed in study conducted by He and colleagues (He et al., 2009). In Random over sampling, the example are uniformly sampled with replacement. This results in the final number of each example is expected to be the same. For weighted Random Forest, assuming the weight to be (0.1, 0.9) for class (0, 1), the classifier will treat 1 minority example as nine identical minority examples. This equals to resampling each minority example 9 times. Actually no additional information is introduced for weighted RF and therefore it only delivers intermediate performance.

Table 13 Averaged classification results with different classifier settings, i.e., ROC Random Forest(ROC RF), weighted Random Forest(wRF), Random Forest with SMOTE data(smoteRF), Random Forest with down sampling data(downRF), Random Forest with Tomek links(tomekRF), Random Forest with ENN(ennRF), Random Forest with CNN(cnnRF), Random Forest with NCL(nclRF), Random Forest with OSS(ossRF), Random Forest with SMOTE+Tomek(smote+tRF) and Random Forest with SMOTE+ENN(smote+eRF). The mean and standard deviation value of AUC, accuracy, sensitivity and specificity are reported. Cell with values marked red indicates the highest averaged AUC value. Paired Z test is applied to compare the averaged mean of method with highest value and the rest methods. Cell marked orange indicates a significant difference with $\alpha = 0.01$ and cells marked blue indicate a non-significant difference.

		Averaged Results			
Dataset	Classifier setting	AUC	accuracy	sensitivity	specificity
red wine quality	ROC RF	0.8956±0.0149	0.8320±0.0262	0.8210±0.0391	0.8338±0.0075
	wRF	0.8945±0.0151	0.8981±0.0066	0.4292±0.0445	0.9721±0.0331
	smoteRF	0.8920±0.0141	0.8528±0.0110	0.7043±0.0529	0.8762±0.0160
	downRF	0.8836±0.0147	0.7855±0.0186	0.8318±0.0430	0.7782±0.0242
	tomekRF	0.8947±0.0144	0.8963±0.0076	0.4817±0.0475	0.9617±0.0084
	ennRF	0.8950±0.0150	0.8976±0.0064	0.4436±0.0447	0.9692±0.0079
	cnnRF	0.8949±0.0147	0.8975±0.0065	0.4284±0.0419	0.9715±0.0078
	nclRF	0.8940±0.0144	0.8935±0.0077	0.5058±0.0535	0.9547±0.0108
	ossRF	0.8945±0.0150	0.8966±0.0074	0.4836±0.0491	0.9618±0.0084
	smote+tRF	0.8911±0.0138	0.8393±0.0132	0.7396±0.0518	0.8551±0.0195
	smote+eRF	0.8907±0.0146	0.8393±0.0139	0.7392±0.0540	0.8550±0.0192
white wine quality	ROC RF	0.8753±0.0183	0.7970±0.0396	0.7749±0.0434	0.7978±0.0419
	wRF	0.8713±0.0197	0.9727±0.0015	0.2687±0.0398	0.9995±0.0005
	smoteRF	0.8690±0.0164	0.8698±0.0143	0.6017±0.0568	0.8800±0.0158
	downRF	0.8563±0.0189	0.7286±0.0306	0.8224±0.0544	0.7250±0.0332
	tomekRF	0.8753±0.0184	0.9726±0.0016	0.2693±0.0407	0.9994±0.0006
	ennRF	0.8744±0.0186	0.9727±0.0015	0.2691±0.0401	0.9995±0.0005
	cnnRF	0.8734±0.0186	0.9726±0.0015	0.2688±0.0399	0.9995±0.0005
	nclRF	0.8744±0.0194	0.9726±0.0016	0.2714±0.0397	0.9993±0.0006
	ossRF	0.8750±0.0183	0.9726±0.0016	0.2701±0.0406	0.9994±0.0006
	smote+tRF	0.8641±0.0174	0.8368±0.0179	0.6623±0.0584	0.8435±0.0199
	smote+eRF	0.8655±0.0170	0.8486±0.0160	0.6458±0.0555	0.8563±0.0176
breast cancer	ROC RF	0.9791±0.0122	0.9651±0.0165	0.9247±0.0283	0.9582±0.0213
	wRF	0.9786±0.0123	0.9521±0.0093	0.8682±0.0516	0.9868±0.0095
	smoteRF	0.9803±0.0112	0.9532±0.0180	0.9110±0.0450	0.9626±0.0235
	downRF	0.9778±0.0112	0.9326±0.0256	0.9300±0.0400	0.9331±0.0340
	tomekRF	0.9791±0.0123	0.9643±0.0095	0.8710±0.0511	0.9852±0.0099
	ennRF	0.9794±0.0116	0.9634±0.0093	0.8718±0.0506	0.9839±0.0112
	cnnRF	0.9781±0.0128	0.9633±0.0102	0.8778±0.0465	0.9824±0.0120
	nclRF	0.9796±0.0116	0.9621±0.0096	0.8752±0.0511	0.9815±0.0121

	ossRF	0.9791 ±0.0128	0.9640 ±0.0092	0.8698 ±0.0508	0.9851 ±0.0098
	smote+tRF	0.9791 ±0.0120	0.9507 ±0.0188	0.9135 ±0.0482	0.9590 ±0.0254
	smote+eRF	0.9791 ±0.0127	0.9496 ±0.0193	0.9120 ±0.0449	0.9580 ±0.0253
connection bench	ROC RF	0.8825 ±0.0637	0.7798 ±0.1043	0.8025 ±0.1205	0.7750 ±0.1273
	wRF	0.8222 ±0.0778	0.8847 ±0.0168	0.1889 ±0.1185	0.9984 ±0.0075
	smoteRF	0.8249 ±0.0759	0.8756 ±0.0542	0.5375 ±0.1901	0.9308 ±0.0661
	downRF	0.8377 ±0.0790	0.6988 ±0.0908	0.8100 ±0.1781	0.6806 ±0.1108
	tomekRF	0.8178 ±0.0754	0.8846 ±0.0193	0.1925 ±0.1311	0.9976 ±0.0088
	ennRF	0.8250 ±0.0757	0.8854 ±0.0182	0.1963 ±0.1208	0.9980 ±0.0074
	cnnRF	0.8256 ±0.0829	0.8861 ±0.0182	0.2062 ±0.1184	0.9971 ±0.0092
	nclRF	0.8160 ±0.0785	0.8863 ±0.0200	0.2087 ±0.1282	0.9969 ±0.0106
	ossRF	0.8172 ±0.0754	0.8851 ±0.0197	0.1963 ±0.1309	0.9976 ±0.0083
	smote+tRF	0.8825 ±0.0680	0.8812 ±0.0565	0.5513 ±0.1947	0.9351 ±0.0594
	smote+eRF	0.8716 ±0.0742	0.8563 ±0.0613	0.5787 ±0.1927	0.9016 ±0.0775
ozone level	ROC RF	0.8920 ±0.0214	0.8359 ±0.0364	0.8016 ±0.0450	0.8369 ±0.0382
	wRF	0.8822 ±0.0231	0.9702 ±0.0010	0.0081 ±0.0146	0.9991 ±0.0010
	smoteRF	0.8859 ±0.0221	0.9016 ±0.0178	0.6624 ±0.0842	0.9088 ±0.0198
	downRF	0.8829 ±0.0203	0.7820 ±0.0344	0.8151 ±0.0509	0.7810 ±0.0361
	tomekRF	0.8868 ±0.0214	0.9701 ±0.0010	0.0111 ±0.0172	0.9989 ±0.0010
	ennRF	0.8856 ±0.0218	0.9702 ±0.0009	0.0076 ±0.0144	0.9991 ±0.0009
	cnnRF	0.8865 ±0.0217	0.9702 ±0.0010	0.0084 ±0.0137	0.9991 ±0.0010
	nclRF	0.8866 ±0.0209	0.9701 ±0.0011	0.0195 ±0.0252	0.9986 ±0.0010
	ossRF	0.8873 ±0.0206	0.9702 ±0.0010	0.0114 ±0.0181	0.9990 ±0.0010
	smote+tRF	0.8882 ±0.0196	0.8929 ±0.0190	0.6670 ±0.0745	0.8997 ±0.0207
	smote+eRF	0.8850 ±0.0194	0.8736 ±0.0231	0.7062 ±0.0666	0.8786 ±0.0249
ionosphere	ROC RF	0.9760 ±0.0120	0.9253 ±0.0319	0.8840 ±0.0451	0.9308 ±0.0378
	wRF	0.9757 ±0.0127	0.9513 ±0.0160	0.6620 ±0.1150	0.9896 ±0.0113
	smoteRF	0.9577 ±0.0226	0.9373 ±0.0237	0.7567 ±0.0991	0.9613 ±0.0256
	downRF	0.9165 ±0.0405	0.8620 ±0.0606	0.8113 ±0.1089	0.8688 ±0.0725
	tomekRF	0.9749 ±0.0120	0.9512 ±0.0150	0.6747 ±0.1073	0.9879 ±0.0118
	ennRF	0.9764 ±0.0121	0.9502 ±0.0160	0.6613 ±0.1186	0.9886 ±0.0113
	cnnRF	0.9213 ±0.0842	0.8838 ±0.1238	0.6940 ±0.1357	0.9089 ±0.1462
	nclRF	0.9733 ±0.0134	0.9514 ±0.0137	0.6840 ±0.1103	0.9869 ±0.0113
	ossRF	0.9751 ±0.0123	0.9511 ±0.0140	0.6733 ±0.1057	0.9880 ±0.0112
	smote+tRF	0.9543 ±0.0244	0.9341 ±0.0324	0.7500 ±0.1031	0.9585 ±0.0354
	smote+eRF	0.9556 ±0.0232	0.9368 ±0.0259	0.7540 ±0.1050	0.9611 ±0.0315
pima diabetes	ROC RF	0.8040 ±0.0359	0.7654 ±0.0523	0.7097 ±0.0706	0.7721 ±0.0624
	wRF	0.7970 ±0.0356	0.8896 ±0.0082	0.1040 ±0.0528	0.9839 ±0.0093
	smoteRF	0.7944 ±0.0281	0.8136 ±0.0231	0.5120 ±0.0976	0.8498 ±0.0309
	downRF	0.7956 ±0.0295	0.7200 ±0.0429	0.7440 ±0.0917	0.7172 ±0.0539
	tomekRF	0.7983 ±0.0355	0.8876 ±0.0110	0.1990 ±0.0778	0.9702 ±0.0138
	ennRF	0.7990 ±0.0351	0.8900 ±0.0079	0.1170 ±0.0602	0.9828 ±0.0095
	cnnRF	0.7972 ±0.0355	0.8894 ±0.0082	0.1057 ±0.0547	0.9834 ±0.0094
	nclRF	0.8044 ±0.0347	0.8853 ±0.0126	0.2243 ±0.0983	0.9646 ±0.0171
	ossRF	0.8030 ±0.0354	0.8881 ±0.0104	0.1983 ±0.0800	0.9709 ±0.0131

	smote+tRF	0.7949 ±0.0289	0.8003 ±0.0249	0.5693 ±0.1124	0.8280 ±0.0343
	smote+eRF	0.7967 ±0.0314	0.7967 ±0.0283	0.5810 ±0.1088	0.8226 ±0.0388
spect	ROC RF	0.8401 ±0.0370	0.7620 ±0.0721	0.8187 ±0.1019	0.7150 ±0.0882
	wRF	0.7884 ±0.0520	0.8765 ±0.0055	0.0200 ±0.0373	0.9977 ±0.0052
	smoteRF	0.8385 ±0.0357	0.8273 ±0.0341	0.5007 ±0.1426	0.8735 ±0.0437
	downRF	0.8365 ±0.0427	0.7112 ±0.0440	0.8340 ±0.1230	0.6938 ±0.0570
	tomekRF	0.8054 ±0.0453	0.8773 ±0.0086	0.0627 ±0.0620	0.9926 ±0.0081
	ennRF	0.7908 ±0.0508	0.8774 ±0.0051	0.0247 ±0.0375	0.9980 ±0.0039
	cnnRF	0.7719 ±0.0653	0.8766 ±0.0053	0.0233 ±0.0372	0.9974 ±0.0047
	nclRF	0.8161 ±0.0451	0.8769 ±0.0114	0.0847 ±0.0937	0.9890 ±0.0107
	ossRF	0.8060 ±0.0470	0.8770 ±0.0089	0.0620 ±0.0609	0.9924 ±0.0084
	smote+tRF	0.8329 ±0.0395	0.8279 ±0.0313	0.4853 ±0.1409	0.8764 ±0.0425
	smote+eRF	0.8397 ±0.0364	0.7863 ±0.0422	0.6740 ±0.1599	0.8022 ±0.0574
vertebral	ROC RF	0.8949 ±0.0282	0.8125 ±0.0162	0.8247 ±0.0779	0.8108 ±0.0633
	wRF	0.8941 ±0.0294	0.8908 ±0.0502	0.2507 ±0.1034	0.9822 ±0.0193
	smoteRF	0.8935 ±0.0281	0.8390 ±0.0371	0.6800 ±0.1268	0.8617 ±0.0461
	downRF	0.8689 ±0.0325	0.7745 ±0.0459	0.8387 ±0.1161	0.7653 ±0.0577
	tomekRF	0.9004 ±0.0258	0.8853 ±0.0232	0.3707 ±0.1094	0.9589 ±0.0296
	ennRF	0.8946 ±0.0285	0.8902 ±0.0166	0.2620 ±0.1122	0.9799 ±0.0210
	cnnRF	0.8522 ±0.0739	0.8872 ±0.0220	0.2693 ±0.1156	0.9755 ±0.0255
	nclRF	0.8997 ±0.0249	0.8761 ±0.0271	0.4387 ±0.1377	0.9386 ±0.0379
	ossRF	0.9006 ±0.0254	0.8848 ±0.0228	0.3667 ±0.1083	0.9588 ±0.0285
	smote+tRF	0.8928 ±0.0287	0.8407 ±0.0341	0.6987 ±0.1358	0.8610 ±0.0417
	smote+eRF	0.8847 ±0.0303	0.8189 ±0.0458	0.7380 ±0.1317	0.8305 ±0.0559
breast tissue	ROC RF	0.9617 ±0.0337	0.9257 ±0.0208	0.8527 ±0.0630	0.9444 ±0.0288
	wRF	0.9579 ±0.0339	0.9367 ±0.0252	0.8236 ±0.1147	0.9656 ±0.0240
	smoteRF	0.9622 ±0.0295	0.9374 ±0.0258	0.8873 ±0.1105	0.9502 ±0.0359
	downRF	0.9630 ±0.0343	0.9015 ±0.0801	0.9245 ±0.0976	0.8956 ±0.1045
	tomekRF	0.9609 ±0.0311	0.9396 ±0.0198	0.8773 ±0.1045	0.9556 ±0.0283
	ennRF	0.9591 ±0.0331	0.9409 ±0.0207	0.8545 ±0.1034	0.9630 ±0.0227
	cnnRF	0.9625 ±0.0278	0.9283 ±0.0356	0.8418 ±0.1124	0.9505 ±0.0489
	nclRF	0.9612 ±0.0316	0.9404 ±0.0215	0.8864 ±0.1092	0.9542 ±0.0278
	ossRF	0.9605 ±0.0314	0.9393 ±0.0201	0.8736 ±0.1033	0.9560 ±0.0282
	smote+tRF	0.9604 ±0.0324	0.9396 ±0.0260	0.9100 ±0.0927	0.9472 ±0.0353
	smote+eRF	0.9610 ±0.0317	0.9400 ±0.0222	0.9045 ±0.0935	0.9491 ±0.0290
haberman	ROC RF	0.6580 ±0.0476	0.6416 ±0.0564	0.6195 ±0.0762	0.6455 ±0.0706
	wRF	0.6506 ±0.0494	0.8289 ±0.0174	0.0555 ±0.0502	0.9658 ±0.0218
	smoteRF	0.6540 ±0.0530	0.7416 ±0.0386	0.3635 ±0.1117	0.8085 ±0.0520
	downRF	0.6209 ±0.0626	0.6067 ±0.0623	0.5660 ±0.1143	0.6139 ±0.0776
	tomekRF	0.6514 ±0.0484	0.8104 ±0.0228	0.1180 ±0.0754	0.9329 ±0.0303
	ennRF	0.6514 ±0.0469	0.8255 ±0.0178	0.0585 ±0.0560	0.9612 ±0.0237
	cnnRF	0.6482 ±0.0501	0.8281 ±0.0165	0.0590 ±0.0548	0.9642 ±0.0216
	nclRF	0.6513 ±0.0485	0.7994 ±0.0277	0.1605 ±0.0922	0.9125 ±0.0340
	ossRF	0.6513 ±0.0486	0.8089 ±0.0222	0.1135 ±0.0735	0.9320 ±0.0293
	smote+tRF	0.6526 ±0.0545	0.7214 ±0.0430	0.4135 ±0.1121	0.7758 ±0.0531

	smote+eRF	0.6573±0.0561	0.7229±0.0454	0.4150±0.1024	0.7774±0.0576
banknote	ROC RF	0.9976±0.0019	0.9802±0.0090	0.9726±0.0105	0.9812±0.0119
	wRF	0.9976±0.0019	0.9816±0.0107	0.8842±0.0679	0.9944±0.0035
	smoteRF	0.9968±0.0028	0.9793±0.0096	0.9502±0.0444	0.9831±0.0091
	downRF	0.9924±0.0058	0.9500±0.0255	0.9542±0.0462	0.9495±0.0291
	tomekRF	0.9976±0.0019	0.9821±0.0089	0.8866±0.0660	0.9946±0.0036
	ennRF	0.9976±0.0019	0.9816±0.0090	0.8836±0.0682	0.9945±0.0036
	cnnRF	0.9951±0.0050	0.9782±0.0184	0.9106±0.0596	0.9871±0.0193
	nclRF	0.9976±0.0019	0.9819±0.0088	0.8870±0.0670	0.9944±0.0035
	ossRF	0.9976±0.0019	0.9818±0.0088	0.8844±0.0663	0.9946±0.0034
	smote+tRF	0.9969±0.0027	0.9774±0.0105	0.9460±0.0460	0.9815±0.0110
	smote+eRF	0.9968±0.0026	0.9774±0.0107	0.9458±0.0429	0.9815±0.0109
magic	ROC RF	0.9217±0.0069	0.8830±0.0146	0.8216±0.0181	0.8880±0.0168
	wRF	0.9165±0.0062	0.9538±0.0015	0.4762±0.0185	0.9925±0.0011
	smoteRF	0.9196±0.0055	0.9167±0.0044	0.7320±0.0185	0.9317±0.0053
	downRF	0.9140±0.0060	0.8544±0.0101	0.8130±0.0211	0.8577±0.0118
	tomekRF	0.9165±0.0057	0.9531±0.0017	0.5109±0.0202	0.9889±0.0015
	ennRF	0.9162±0.0071	0.9534±0.0016	0.4810±0.0187	0.9917±0.0012
	cnnRF	0.9158±0.0065	0.9536±0.0016	0.4761±0.0187	0.9924±0.0012
	nclRF	0.9176±0.0063	0.9528±0.0017	0.5189±0.0206	0.9880±0.0015
	ossRF	0.9161±0.0068	0.9532±0.0018	0.5109±0.0207	0.9891±0.0014
	smote+tRF	0.9177±0.0056	0.9092±0.0050	0.7415±0.0186	0.9228±0.0060
	smote+eRF	0.9175±0.0054	0.9056±0.0048	0.7496±0.0186	0.9183±0.0057
page block	ROC RF	0.9904±0.0029	0.9590±0.0079	0.9713±0.0101	0.9576±0.0093
	wRF	0.9897±0.0033	0.9746±0.0023	0.8565±0.0233	0.9881±0.0024
	smoteRF	0.9904±0.0021	0.9614±0.0045	0.9502±0.0152	0.9626±0.0054
	downRF	0.9888±0.0022	0.9387±0.0099	0.9723±0.0116	0.9349±0.0115
	tomekRF	0.9901±0.0029	0.9745±0.0025	0.8846±0.0216	0.9847±0.0029
	ennRF	0.9900±0.0030	0.9748±0.0024	0.8693±0.0224	0.9869±0.0027
	cnnRF	0.9904±0.0027	0.9745±0.0024	0.8574±0.0242	0.9879±0.0024
	nclRF	0.9902±0.0026	0.9737±0.0028	0.8898±0.0224	0.9833±0.0035
	ossRF	0.9902±0.0028	0.9744±0.0025	0.8844±0.0249	0.9846±0.0029
	smote+tRF	0.9898±0.0022	0.9561±0.0055	0.9567±0.0160	0.9560±0.0068
	smote+eRF	0.9898±0.0021	0.9562±0.0058	0.9605±0.0129	0.9557±0.0070
parkinsons	ROC RF	0.9140±0.0457	0.8644±0.0723	0.7590±0.0842	0.8786±0.0832
	wRF	0.9019±0.0495	0.9188±0.0307	0.4240±0.1584	0.9857±0.0239
	smoteRF	0.8867±0.0527	0.8512±0.0449	0.6730±0.1814	0.8753±0.0503
	downRF	0.8829±0.0498	0.7738±0.0658	0.8080±0.1555	0.7692±0.0820
	tomekRF	0.9139±0.0445	0.9226±0.0302	0.4760±0.1621	0.9830±0.0255
	ennRF	0.9110±0.0455	0.9194±0.0282	0.4290±0.1572	0.9857±0.0217
	cnnRF	0.9106±0.0483	0.9169±0.0322	0.4470±0.1666	0.9804±0.0274
	nclRF	0.9109±0.0477	0.9204±0.0307	0.4750±0.1654	0.9805±0.0264
	ossRF	0.9126±0.0465	0.9210±0.0320	0.4650±0.1654	0.9826±0.0261
	smote+tRF	0.8791±0.0538	0.8308±0.0581	0.7220±0.1889	0.8455±0.0684
	smote+eRF	0.8851±0.0524	0.8468±0.0479	0.6930±0.2006	0.8676±0.0594

seismic bump	ROC RF	0.7510 ±0.0185	0.7001 ±0.0377	0.6894 ±0.0395	0.7050 ±0.0421
	wRF	0.7345 ±0.0218	0.9324 ±0.0016	0.0134 ±0.0132	0.9971 ±0.0019
	smoteRF	0.7550 ±0.0204	0.8541 ±0.0154	0.4171 ±0.0628	0.8849 ±0.0184
	downRF	0.7506 ±0.0210	0.7286 ±0.0325	0.6459 ±0.0534	0.7345 ±0.0367
	tomekRF	0.7438 ±0.0186	0.9311 ±0.0027	0.0373 ±0.0238	0.9940 ±0.0035
	ennRF	0.7409 ±0.0201	0.9324 ±0.0018	0.0145 ±0.0138	0.9971 ±0.0021
	cnnRF	0.7400 ±0.0209	0.9324 ±0.0016	0.0128 ±0.0130	0.9972 ±0.0020
	nclRF	0.7460 ±0.0194	0.9285 ±0.0044	0.0724 ±0.0351	0.9888 ±0.0057
	ossRF	0.7429 ±0.0201	0.9312 ±0.0027	0.0380 ±0.0243	0.9941 ±0.0034
	smote+tRF	0.7534 ±0.0195	0.8109 ±0.0198	0.5331 ±0.0680	0.8305 ±0.0237
	smote+eRF	0.7536 ±0.0199	0.8320 ±0.0167	0.4836 ±0.0631	0.8565 ±0.0205
secom	ROC RF	0.7082 ±0.0259	0.6996 ±0.0487	0.6375 ±0.0556	0.7040 ±0.0543
	wRF	0.6523 ±0.0319	0.9336 ±0.0002	0.0000 ±0.0000	1.0000 ±0.0002
	smoteRF	0.6857 ±0.0309	0.9276 ±0.0054	0.0429 ±0.0322	0.9905 ±0.0065
	downRF	0.6625 ±0.0334	0.6215 ±0.0411	0.6227 ±0.0884	0.6214 ±0.0480
	tomekRF	0.6544 ±0.0314	0.9335 ±0.0006	0.0000 ±0.0000	0.9998 ±0.0006
	ennRF	0.6542 ±0.0298	0.9336 ±0.0002	0.0000 ±0.0000	1.0000 ±0.0002
	cnnRF	0.6540 ±0.0327	0.9336 ±0.0002	0.0000 ±0.0000	1.0000 ±0.0002
	nclRF	0.6593 ±0.0292	0.9334 ±0.0007	0.0002 ±0.0019	0.9997 ±0.0007
	ossRF	0.6575 ±0.0317	0.9335 ±0.0005	0.0000 ±0.0000	0.9998 ±0.0005
	smote+tRF	0.6578 ±0.0329	0.9184 ±0.0083	0.0804 ±0.0439	0.9779 ±0.0103
	smote+eRF	0.6619 ±0.0364	0.9167 ±0.0092	0.0779 ±0.0447	0.9763 ±0.0111
seeds	ROC RF	0.9870 ±0.0103	0.9333 ±0.0286	0.9000 ±0.0001	0.9380 ±0.0327
	wRF	0.9863 ±0.0102	0.9437 ±0.0253	0.8320 ±0.1463	0.9597 ±0.0258
	smoteRF	0.9833 ±0.0123	0.9253 ±0.0211	0.9100 ±0.1147	0.9274 ±0.0294
	downRF	0.9818 ±0.0124	0.9055 ±0.0361	0.9740 ±0.0694	0.8957 ±0.0444
	tomekRF	0.9865 ±0.0110	0.9375 ±0.0253	0.8820 ±0.1304	0.9454 ±0.0301
	ennRF	0.9857 ±0.0106	0.9425 ±0.0267	0.8340 ±0.1507	0.9580 ±0.0257
	cnnRF	0.9850 ±0.0150	0.9480 ±0.0251	0.8400 ±0.1485	0.9634 ±0.0247
	nclRF	0.9837 ±0.0109	0.9335 ±0.0236	0.8660 ±0.1350	0.9431 ±0.0262
	ossRF	0.9867 ±0.0102	0.9373 ±0.0253	0.8800 ±0.1309	0.9454 ±0.0300
	smote+tRF	0.9855 ±0.0103	0.9207 ±0.0268	0.9180 ±0.1101	0.9211 ±0.0351
	smote+eRF	0.9843 ±0.0101	0.9213 ±0.0240	0.9180 ±0.1207	0.9217 ±0.0333

Format: mean ± standard deviation

Table 14 Summary table of the result. Rows are the 18 dataset and columns are the Random Forest with different setting. The first and second (or tied first) method which produce highest averaged AUC are marked red. A paired Z-test is performed to compare method with highest AUC with the rest. Result with significant level 0.01 are shown. Cell marked orange indicates a significant difference and cell marked light blue indicate a non-significant result.

	ROC RF	weightd RF	smote RF	down RF	tomek RF	enn RF	cnn RF	ncl RF	oss RF	smote+t RF	smote+e RF
red wine quality	1st					2nd					
white wine quality	1st				1st						
breast cancer	4th		1st								
connection bench	1st									1st	
ozone level	1st									2nd	
ionosphere	2nd					1st					
pima diabetes	2nd							1st			
spect	1st										2nd
vertebral	4th								1st		
breast tissue	3rd			1st			2nd				
Haberman	1st										2nd
banknote	1st	1st			1st	1st		1st	1st		
magic	1st		2nd								
page block	1st		1st				1st				
parkinsons	1st								2nd		
seismic bump	4th		1st								
secom	1st		2nd								
seeds	1st								2nd		

Figures corresponding to each data set and each Random Forest setting could be found in the **Appendix**. For all the datasets, we plot the averaged sensitivity and specificity (sen-spe) pair on the averaged ROC curve (**Algorithm 8**). Since averaged ROC curve is a graph average and so the average sen-spe pair may not locates on the averaged curve, but they should be very close. From these figures we could observe that ROC RF always deliver the balanced sen-spe pair. Surprisingly, the randomly down sampled RF also deliver balanced sen-spe pair. However, comparing to ROC RF, since it loss data information, its averaged AUC is nearly always significantly less than ROC RF. For SMOTE sampling method, i.e., smoteRF, smote+tRF and smote+eRF they show more balanced sen-spe pairs than other methods and their AUC are also correlated to each other. However, observed from some of the dataset, comparing to ROC RF, they prefer a higher specificity choice on the pair, which leads to operating point close to the lower left corner. As we introduced before, Tomek links, ENN, CNN, OSS, NCL methods may not generate balanced data in general and therefore, for most process dataset the class distribution is still imbalanced. For most dataset, the induced classification point locates on the lower left part of the ROC space, which indicates they still suffer from the imbalanced learning problem. Since they removed some majority data, comparing to weighted RF result, the operating point shift towards the upper right corner a little bit.

The running time of ROC Random Forest and base Random Forest are also compared. **Table 15** gives the averaged single forest build time. ROC Random Forest always take more time to build a model. However, it never exceed too much. Possibly this is because in tree node splitting phase, we use an additional scan to select the

splitting threshold while in original Random Forest algorithm, they finish attribute selection and node splitting in the same scan. The running time of ROC Random Forest is comparable with the original Random Forest algorithm.

Table 15 Averaged running time to build a single base Random Forest and ROC Random Forest model using the 18 dataset. The unit of time is second.

	red wine quality	white wine quality	breast cancer	connection bench	ozone level
base RF	0.8080	2.4675	0.2350	0.1070	4.0950
ROC RF	0.9485	2.8880	0.2785	0.1125	5.2600
	ionosphere	pima diabetes	spect	vertebral	breast tissue
base RF	0.2105	0.2880	0.2845	0.0790	0.0340
ROC RF	0.2425	0.3345	0.3335	0.0945	0.0370
	haberman	banknote	magic	page block	parkinsons
base RF	0.0780	0.2300	11.891	3.0815	0.0710
ROC RF	0.0900	0.2680	13.5360	3.9060	0.0895
	seismic bump	secom	seeds		
base RF	1.3930	25.5915	0.0420		
ROF RF	1.6650	34.5245	0.0525		

To summarize this section, based on the classification result on the 18 dataset, ROC Random Forest performs among the best classifiers and the decision point well balanced the minority class and majority class.

5.3 Evaluation of the Hybrid ROC Random Forest

In this section, we use 8 imbalanced dataset, which contains categorical variables, downloaded from UCI Machine Learning Repository (Bache et al., 2013), to evaluate the performance of the extension of ROC Random Forest – Hybrid ROC Random Forest. We also evaluate the performance of original Random Forest with cost-sensitive setting,

randomly downsampling setting and SMOTE sampling setting. Average results are reported.

5.3.1 Dataset

The detailed information is shown in **Table 16**. The dataset we choose are among the most popular dataset in UCI repository, and preprocessing, e.g., dealing with missing data, class binarilization. All of the dataset set are imbalanced with imbalanced ratio close to 15%.

Table 16 Data information of the 8 dataset from UCI repository with categorical variable

Dataset	Positive Instance#	Negative Instance#	Imbalance Ratio	Continuous Feature#	Categorical Feature#	Source
abalone	62	4115	0.0148	8	1	UCI
acute	16	61	0.2078	1	5	UCI
credit AUS	60	383	0.1354	6	8	UCI
credit GER	100	700	0.1250	7	13	UCI
credit APP	60	367	0.1405	6	9	UCI
band of cylinder	40	224	0.1515	19	16	UCI
contraceptive	100	844	0.1059	4	5	UCI
animals in zoo	20	81	0.1980	1	15	UCI

5.3.2 Learning methods

Experiments in this section is to evaluate the performance of the proposed Hybrid ROC Random Forest and at the same time compare it with original Random Forest with cost sensitive setting and sampling strategies. Since for categorical data, the distance is difficult to define, and therefore, Tomek links is not available. In Chawla's study in 2002, an approximation of continuous-categorical and categorical only distance measure is defined and therefore we adapt their method to generate SMOTE sample. So in this

section, we only choose cost-sensitive Random Forest, down sampling Random Forest and SMOTE Random Forest for comparison. The cost sensitive setting, downsampling setting and SMOTE setting are similar to the previous sections. Two thousand trees are built for each forest, and for each node square root of attribute numbers is randomly selected for splitting.

5.3.3 Two-way random splitting

We use the same partition strategy as we used in §5.2. We record the ROC curve for each classification as well as the accuracy (all with 0.5 decision threshold). To minimize the bias of one time running, the random partition process was repeated 100 times and the averaged results were outputted.

5.3.4 Results and discussion

Table 17 shows the averaged AUC value, classification accuracy, sensitivity and specificity value and their corresponding standard deviation of the four different Random Forest algorithms setting applying on the 8 datasets. The corresponding significance test result is shown in **Table 18**. The averaged ROC curve and classification points are listed in **Appendix**. In terms of AUC value, Hybrid ROC RF performs best in 5 of 8 datasets, and in 1 of them it significantly outperform the second place classifier. Weighted Random Forest deliver the best AUC for 4 of the dataset and in one it significantly outperforms others. Only for the abalone dataset, Random Forest with SMOTE setting outperform other classifiers. For Random Forest with down sampling, its AUC is inferior comparing with others. With respect to the classification points, the Hybrid ROC Random Forest and the Random Forest with downsampling setting deliver balanced classification results, i.e., balanced sensitivity specificity pair. For Random Forest with

SMOTE sampling, it still have the problems that it prefer specificity more than sensitivity, possible reason is that the new generated point will always locates in the minority area and this add no information to the classification. Weighted Random Forest fails to generate balanced classification results and it indicates that randomly oversampling is not less useful for imbalanced data learning and the prior weight strategy have weak capability to detect the minority data. In all these dataset, Hybrid Random Forest performs very similar to Random Forest with weighted Random Forest, their AUC value are nearly the same (no significant difference for most dataset). For dataset where categorical variable dominates the feature space, its performance in choosing splitting point is biased, see the averaged ROC curve on acute data. And therefore, we do not recommend to use this method in dealing with categorical data which dominant the feature space. It is also interesting to observe that 1. Also from acute data, the AUC of down sampling method and SMOTE method are not one, which indicates that even though sometime such strategies improve the classification or ROC curve, for some time to do loss information or introduce system noise.

Table 19 shows the averaged running time of base Random Forest and the Hybrid ROC Random Forest on the 8 datasets. Comparing to base Random Forest, we could see that the running time is still comparable. The more comparison steps in node splitting phase also reflects on the running time. The Hybrid ROC Random Forest uses more time than that based on the Random Forest, however, there is only a constant factor difference.

Table 17 Averaged classification results with different classifier settings, i.e., Hybrid ROC Random Forest (HROC RF), weighted Random Forest(wRF), Random Forest with SMOTE data(smoteRF), Random Forest with down sampling data(downRF). The mean and standard deviation value of AUC, accuracy, sensitivity and specificity are reported. Cell with values marked red indicates the highest averaged AUC value. Paired Z test is applied to compare the averaged mean of method with highest value and the rest methods. Cell marked orange indicates a significant difference with $\alpha = 0.01$ and cells marked blue indicate a non-significant difference.

		Averaged Results			
Dataset	Classifier setting	AUC	accuracy	sensitivity	specificity
abalone	HROC RF	0.8560 ±0.0190	0.7918 ±0.0368	0.7763 ±0.0465	0.7924 ±0.0388
	wRF	0.8508 ±0.0199	0.9664 ±0.0012	0.0103 ±0.0126	0.9985 ±0.0014
	smoteRF	0.8612 ±0.0179	0.8614 ±0.0184	0.6369 ±0.0715	0.8690 ±0.0204
	downRF	0.8442 ±0.0197	0.7447 ±0.0317	0.8690 ±0.0572	0.7429 ±0.0337
acute	HROC RF	1.0000 ±0.0000	0.9726 ±0.0075	0.8663 ±0.0366	1.0000 ±0.0000
	wRF	1.0000 ±0.0000	0.9687 ±0.0356	0.8475 ±0.1737	1.0000 ±0.0000
	smoteRF	0.9992 ±0.0053	0.9828 ±0.0370	0.9325 ±0.1512	0.9958 ±0.0281
	downRF	0.9920 ±0.0232	0.9400 ±0.0767	0.9513 ±0.1328	0.9371 ±0.0939
credit AUS	HROC RF	0.9231 ±0.0280	0.8563 ±0.0308	0.8537 ±0.0572	0.8567 ±0.0387
	wRF	0.9212 ±0.0284	0.9208 ±0.0109	0.5260 ±0.0844	0.9825 ±0.0081
	smoteRF	0.9073 ±0.0311	0.8901 ±0.0215	0.7557 ±0.0763	0.9111 ±0.0285
	downRF	0.9139 ±0.0283	0.8373 ±0.0369	0.8520 ±0.0754	0.8349 ±0.0478
credit GER	HROC RF	0.7456 ±0.0278	0.6917 ±0.0428	0.6924 ±0.0680	0.6916 ±0.0542
	wRF	0.7475 ±0.0283	0.8737 ±0.0033	0.0184 ±0.0192	0.9959 ±0.0034
	smoteRF	0.7226 ±0.0312	0.8105 ±0.0210	0.3470 ±0.0762	0.8767 ±0.0278
	downRF	0.7290 ±0.0289	0.6814 ±0.0338	0.6478 ±0.0685	0.6862 ±0.0430
credit APP	HROC RF	0.9315 ±0.0204	0.8719 ±0.0323	0.8680 ±0.0474	0.8725 ±0.0397
	wRF	0.9303 ±0.0193	0.9070 ±0.0130	0.4823 ±0.0974	0.9762 ±0.0122
	smoteRF	0.9232 ±0.0198	0.8832 ±0.0203	0.7750 ±0.0925	0.9001 ±0.0294
	downRF	0.9203 ±0.0229	0.8421 ±0.0272	0.8880 ±0.0679	0.8347 ±0.0362
band of cylinder	HROC RF	0.7690 ±0.0480	0.7494 ±0.0688	0.6426 ±0.0908	0.7676 ±0.0877
	wRF	0.7648 ±0.0513	0.8681 ±0.0097	0.1016 ±0.0600	0.9981 ±0.0050
	smoteRF	0.7548 ±0.0504	0.8295 ±0.0361	0.4037 ±0.1176	0.9017 ±0.0424
	downRF	0.7314 ±0.0625	0.6495 ±0.0579	0.6779 ±0.1303	0.6447 ±0.0768
contraceptive	HROC RF	0.7140 ±0.0311	0.7140 ±0.0472	0.6130 ±0.0564	0.7259 ±0.0549
	wRF	0.7158 ±0.0301	0.9033 ±0.0052	0.1824 ±0.0386	0.9888 ±0.0055
	smoteRF	0.7154 ±0.0313	0.8303 ±0.0209	0.4128 ±0.0682	0.8798 ±0.0259
	downRF	0.6994 ±0.0341	0.6779 ±0.0378	0.6136 ±0.0748	0.6855 ±0.0461
animals in zoo	HROC RF	1.0000 ±0.0000	1.0000 ±0.0000	1.0000 ±0.0000	1.0000 ±0.0000
	wRF	1.0000 ±0.0000	1.0000 ±0.0000	1.0000 ±0.0000	1.0000 ±0.0000
	smoteRF	1.0000 ±0.0000	1.0000 ±0.0000	1.0000 ±0.0000	1.0000 ±0.0000
	downRF	0.9999 ±0.0012	0.9896 ±0.0389	1.0000 ±0.0000	0.9871 ±0.0483

Table 18 Summary table of the result. Rows are the 8 dataset and columns are the Random Forest with different setting. The first and second (or tied first) method which produce highest averaged AUC are marked red. A paired Z-test is performed to compare method with highest AUC with the rest. Result with significant level 0.01 are shown. Cell marked orange indicates a significant difference and cell marked light blue indicate a non-significant result.

	HROC RF	weighted RF	smote RF	down RF
abalong	2nd		1st	
acute	1st	1st		
credit AUS	1st	2nd		
credit GER	2nd	1st		
credit APP	1st	2nd		
band of cylinder	1st	2nd		
contraceptive	2nd	1st		
animals in zoo	1st	1st	1st	

Table 19 Average running time to build a single base Random Forest and a Hybrid ROC Random Forest model using the 8 datasets. The unit of time is second.

	abalone	acute	credit AUS	credit GER
base RF	1.15	0.024	0.214	0.934
HROC_C RF	1.466	0.0285	0.2965	1.1405
	credit APP	band of cylinder	contraceptive	animals in zoo
base RF	0.4025	0.1985	0.2385	0.028
HROC_C RF	0.53	0.2745	0.306	0.037

Chapter VI: Conclusion and Future Work

In this dissertation, we proposed a novel binary classification algorithm, the ROC Random Forest, by combining the ROC analysis with the Random Forest based ensemble classifier. The main idea of this algorithm is to substitute the information gain based tree node splitting method with the ROC based splitting method. For each node, we first select the feature that produces the highest AUC and then a splitting threshold, which maximizes the harmonic mean of sensitivity and specificity, to split the node. And then, as what regular Random Forest does, a majority vote strategy is employed. Furthermore, to extend the model to deal with nominal categorical variable, we proposed the Hybrid ROC Random Forest, that combines the harmonic mean method with the original impurity drop method to accommodate categorical predictors.

Three validation experiments were conducted to evaluate the proposed algorithm. Experiment one is a proof of concept experiment to validate the benefit of ROC based splitting threshold selection strategy (Song et al., 2014). For both Random Forest and SVM, it well corrected the problem caused by imbalanced data. In experiment two, we evaluated the proposed ROC Random Forest in 18 imbalanced dataset downloaded from UCI (Bache et al., 2013). In terms of AUC value, ROC Random Forest outperform other classification setting in 12 of the 18 dataset. In terms of classification sensitivity and specificity, points with well-balanced sensitivity-specificity value were produced. As shown in the averaged ROC curve, the point always locates on the upper left corner. We also evaluate the Hybrid ROC Random Forest on 8 UCI dataset with categorical predictors. The Hybrid ROC Random Forest produced more balanced classification result for imbalanced data in comparison to the classic Random Forest.

In summary, our contributions are as follows:

1. We proposed harmonic mean based ROC operating selection method. Experiment results well validate its using in find a balanced classification point for class-imbalanced data.
2. We proposed a Random Forest algorithm based on ROC analysis. The innovation of the algorithm is listed below:
    a. ROC based splitting node selection.
    b. Harmonic mean based node splitting.
    c. Directed child node labeling.

    Based on the experiments of ROC Random Forest on the 18 UCI realistic dataset, ROC Random Forest performs among the best classifiers and the decision point well balanced the minority class and majority class.
3. We extend the ROC Random Forest to deal with nominal categorical variables, which we name as the Hybrid ROC Random Forest. The innovation of the algorithm is that we combine the maximum AUC continues feature selection with maximum impurity drop categorical feature selection. Based on validation experiments of the Hybrid ROC Random Forest on 8 UCI dataset with categorical variable, the Hybrid ROC Random Forest could also produce a more balanced classification decision with relatively high AUC value.

Meanwhile, we also need to mention that there is also some disadvantages with the proposed methods. For example, the ROC curve is only plausible for binary classification problems while Information Gain/ Gini Impurity could be used for multiple

class classification problems. What's more, in its current state, the ROC Random Forest could only handle continuous data or ordinal categorical data. Even though we have subsequently extended the ROC Random Forest to the Hybrid ROC Forest method, to incorporate all types of categorical variables, the new method is dependent upon the original impurity drop method and therefore for categorical variable dominated dataset, the classification result is not ideal yet. However, in study conducted by Ferri and colleagues (Ferri et al., 2002), they proposed an exhaust labeling strategy to plot the ROC convex hull and part the boundary could be considered as the ROC curve. The computation time of this method is excessive and impractical when the feature contains many categorical levels. Methods have been proposed to deal with such problem and we will focus on this and extend the usage of our proposed model in the near future.

In summary, the ROC Random Forest proves to be a better alternative in general for binary class classification with imbalanced classes. The prediction result is more optimal on the average and its hybrid version can also absorb categorical predictors.

References

Ahn, H., Moon, H., Fazzari, M. J., Lin, N., Chen, J. J., and Kodell, R. L., Classification by ensembles from random partitions of high-dimentional data. Computational statistics and data analysis, 51, 6166-6179, 2007.

Aizerman, A., Braverman, E. M., Rozoner. L. I., Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control, 25, 821-837, 1964.

Archer, K., Kimes, R., Empirical characterization of random forest variable importance measures. Computational Statistics and Data Analysis, 52(4), 2249-2260.

Bache, K., Lichman, M., UCI Machine Learning Repository. [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science. 2013

Batista, A., Prati, C., Monard, C., A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. ACM SIGKDD Explorations Newsletter, 6(1), 20-29, 2004.

Bertoni, A., Raffaella, F., Giorgio, V., Bio-molecular cancer prediction with random subspace ensembles of support vector machines. Neurocomputing, 63, 535-539, 2005.

Bishop, C. M., Pattern Recognition and Machine Learning. Springer 2006

Bradley, A. P., The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30(7), 1145-1159, 1997.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., Classification and regression trees. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey CA, 1984.

Breiman, L., Bagging predictors. Machine Learning, 24, 123-140, 1996.

Breiman, L., Random forest. Machine Learning, 45, 5-32, 2001.

Burges, C. J., A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2, 121-167, 1998.

Calle, M.L., Urrea, V., Boulesteix, A., Malats, N., AUC-RF: a new strategy for genomic profiling with Random Forest. Hum Hered, 72,121-132, 2011.

Cardie, C., Howe, N., Improving minority class prediction using case-specific feature weights. Proceedings of the Fourteenth International Conference on Machine Learning, San Francisco, CA: Morgan Kaufmann, 57-65, 1997.

Caruana, R., Niculescu-Mizil, A., An empirical comparison of supervised learning algorithms. Proceedings of the 23rd international conference on Machine Learning, 2006

Chawla, N. V., Bowyer, K. W., Hall, L. O., Kegelmeyer, W. P., SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357, 2002.

Chen, C., Liaw, A., Breiman, L., Using Random Forest to Learn Imbalanced Data. Technical Report of Department of Statistics, UC, Berkeley, 2004.

Chang, C., Lin, C., LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2(27), 1-27, 2011.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J., Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems. Elsevier, 47(4): 547-553, 2009.

Dietterich, T., Ensemble Methods in Machine Learning, Mutiple Classifier Systems, Lecture Notes in Computer Science, 1857, 1-15, 2000.

Domingos, P., MetaCost: A general method for making classifiers cost-sensitive. Proceedings of Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 155-164, 1999.

Drummond, C., Holte, R. C., C4.5, Class Imbalance, and Cost Sensitivity: Why Under Sampling Beats Over-Sampling. Proceedings of International Conference of Machine learning, Workshop Learning from Imbalanced Data Sets II, 2003.

Duin, R. P. W., Tax, D. M. J., Experiments with classifier combining rules. Mutiple Classifier Systems, Lecture Notes in Computer Science, 1857, 16-29, 2000.

Efron, B., Bootstrap methods: Another look at the jackknife. The Annals of Statistics, 7(1), 1-26, 1979.

Elkan, C., The foundations of Cost-Sensitive Learning. Proceedings of International Joint Conference of Artificial Intelligence, 973-978, 2001.

Estabrooks, A., Jo, T., Japkowicz, N., A Multiple Resampling Method for Learning from Imbalanced Data Sets. Computational Intelligence, 20, 18-36, 2004.

Fawcett, T., Provost, F., Adaptive fraud detection. Data Mining and Knowledge Discovery, 1, 291-316, 1997.

Fawcett, T., Using the rule sets to maximize ROC performance. Proceedings of IEEE International Conference on Data Mining, 131-138, 2001.

Fawcett, T., An introduction to ROC analysis. Pattern Recognition Letters, 27, 861-874, 2006.

Ferri, C., Flach, P., Hernandez-Orallo, J., Learning decision trees using the area under the ROC curve. Proceedings of ICML 2002, 139-146, 2002.

Fletcher, R., Practical Methods of Optimization. John Wiley and Sons, Inc. 2nd edition, 1987.

Frank, E. and Hall, M., A simple approach to ordinal classification. Machine Learning:ECML 2001, 145-156.

Geisser, S., Predictive Inference. Chapman and Hall, New York, NY. ISBN 0-412-03471-9, 1993.

Hand, D. J., Till, R. J., A simple generalization of the area under the ROC curve to multiple class classification problems. Machine Learning, 45(2), 171-186, 2001.

Hanley, J. A., McNeil, B. J., The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 143, 29-36, 1982.

Hanley, J. A., The robustness of the "binormal" assumptions used in fitting ROC curves. Md Decis Making, 8,197-203, 1988.

Hansen, L. K., and Salamon, P., Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12, 993-1001, 1990.

Hart, P.E., The condensed nearest neighbor rule. IEEE Transactions on Information Theory IT-14, 515-516, 1968.

Hastie, T., Tibshirani, R., Friedman, J., The Elements of Statistical Learning: Data Mining, Inference, and Prediction. (Second Edition) Springer 2009.

Hassibi, K., Business Applications of Neural Networks, Chapter 9. Singapore-New jersey-London-Hon Kong: World Scientific, 141-158, 2000.

He, H., Garcia, E. A., Learning from Imbalanced Data, IEEE Transcations on Knowledge and Data Engineering, 21(9), 1263-1284, 2009.

Ho, T. K., The Random Subspace Method for Constructing Decision Forests, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), 832-844, 1998.

Holte, R. C., Acker, L., Porter, B. W., Concept Learning and the Problem of Small Disjuncts. Proceeding of International Joint Conference on Artificial Intelligence, 813-818, 1989.

Hossain, M. M., Hassan, M. R., Bailey, J., ROC-tree: A Novel Decision Tree Induction Algorithm Based on Receiver Operating Characteristics to Classify Gene Expression Data. Proceedings of the 2008 SIAM international conference on data mining (SDM), 455-465, 2008.

Hubert-Moy, L., Cotonnec, A., Le Du, L., Chardin, A., Perez, P., A Comparison of Parametric Classification Procedures of Remotely Sensed Data Applied on Different Landscape Units. Remote Sensing of Environment 2001, 75(2), 174-187

Hunt, E., Martin, J., and Stone, P., Experiments in induction. Adacemic Press, NY, 1966.

James, G., Witten, D., Hastie, T., Tibshirani, R., An Introduction to Statistical Learning. Springer, 2013.

Japkowicz, N., Stephen, S., The Class Imbalance Problem: A Systematic Study. Intelligent Data Analysis, 6(5), 429-449, 2002.

Kubat, M., Matwin, S., Adding the course of imbalanced training sets: one-sided selection. ICML, 179-186, 1997.

Kuncheva, L., Rodriguez, J., Plumpton C., Linden, D., Johnston, J., Random Subspace Ensembles for fMRI Classification, IEEE Transactions on Medical Imaging, 29(2), 531-542, 2010.

Laurikkala, J., Improving identification of difficult small classes by balancing class distribution. Technique Report. University of Tampere, A-2001-2, 2001.

Pesce, L., Metz, C., Berbaum, K., On the convexity of ROC curves estimated from radiological test results. Acad Radiol, 17(8), 960-968, 2010.

Liu, X. Y., Wu, J., Zhou, Z. H., Exploratory Under Sampling for Class Imbalance Learning. Proceedings o International Conference of Data Mining, 965-969, 2006.

Liaw, A., Wiener, M., Classification and Regression by randomForest. R News 2(3), 18-22, 2002.

Louppe, G., Understanding random forests: from theory to practice. PhD thesis, University of Li ège, 2014.

Mac Namee, B., Cunningham, P., Byrne, S., Corrigan, O., The problem of bias in training data in regression problems in medical decision support. Artificial Intelligence in Medicine, 24, 51-70, 2002.

Maloof, A., Learning When Data Sets are Imbalanced and When Costs are Unequal and Unknown. Proceedings of International Conference Machine Learning, Workshop Learning from Imbalanced Data Sets, 2003.

Mease, D., Wyner, A. J., Buja, A., Boosted Classification Trees and Class Probability/Quantile Estimation. Journal of Machine Learning Research, 8, 409-439, 2007.

Metz, C. E., Basic principles of ROC analysis. Seminars in Nuclear Medicine. 8(4), 283-298, 1978.

Metz, C.E., Herman, B., Shen, J., Maximum likelihood estimation of receiver operating characteristic (ROC) curves from continuously-distributed data. Stat Med, 17,1033-1053, 1998.

Mohri, M., Rostamizadeh, A., Talwalkar, A., Foundations of Machine Learning. The MIT Press, Cambridge USA, 2012.

Quinlan, J. R., Induction of Decision Trees. Machine Learning, 1986, 1(1), 81-106.

Quinlan, J. R., improved use of continuous attributes in C4.5. Journal of Artificial Intelligence Research, 1996, 4, 77-90.

Opitz, D., Maclin, R., Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11:169-198, 1999.

Pepe, M., The statistical evaluation of medical tets for classification and prediction. Oxford: Oxford University Press, 2003.

Polikar, R., Ensemble based systems in decision making. IEEE Circuits and Systems Magazine, 6(3), 21-45, 2006.

Provost, F., Fawcett, T., Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97). AAAI Press, Menlo Park, CA, 43-48, 1997.

Provost, F., Fawcett, T., Robust Classification for Imprecise Environments. Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), Madison, WI. 1998.

Provost, F., Machine learning from imbalanced data sets 101. Invited paper for the AAAI' 2000 Workshop on Imbalanced Data Sets (AAAI-2000).

Provost, F., Domingos, P., Well-trained PETs: Improving progability estimation trees. CeDER Working Paper #IS-00-04, Stern School of Business, New York University, NY, 10012, 2001.

Rakotomamonjy, A., Optimizing area under ROC curve with SVMs. ROC Analysis in Artificial Intelligence, 71-80, 2004.

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J., Muller, M., pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics, 17,12-77, 2011.

Raskutti, B., Kowalczyk, A., Extreme Re-Balancing for SVMs: A Case Study. ACM SIGKDD Explorations Newsletter, 6(1), 60-69, 2004.

Shamos, M. I., and Hoey, D., Closest-point problems. Proceedings in Annual Symposium on Foundations of Computer Science, 151-162, 1975

Sigillito, V., Wing, S., Hutton, L., Baker, K., Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest, 10, 262-266, 1989.

Skurichina, M., Duin, R. P. W., Bagging, boosting and the random subspace method for linear classifiers. Pattern Analysis and Applications, 5, 121-135, 2002.

Song, B., Zhang, G., Zhu, W., Liang, Z., ROC operating point selection for classification of imbalanced data with application to computer-aided polyp detection in CT colonography. International Journal of Computer Assisted Radiology and Surgery, 9(1), 79-89, 2014.

Sparckman, K. A., Signal detection theory: Valuable tools for evaluating inductive learning. Proceedings of Sixth International Workshop on Machine Learning. Morgan Kaufman, San Mateo, CA. 160-163, 1989.

Swets, J., Measuring the accuracy of diagnostic systems. Science, 240, 1285-1293, 1988.

Swets, J., Dawes, R., Monahan, J., Better decisions through science. Scientific American, 283, 82-87, 2000.

Tao, D., Tang, X., Li, X., Wu, X., Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(7), 1088-1099, 2006.

Tang, Y. C., Jin, B., Granular SVM with Repetitive Undersampling for Highly Imbalanced Protein Homology Prediction. Proceedings of International Conference of Granular Computing, 457-460, 2006.

Tang, Y. C., Jin, B., Zhang, Y. Q., Granular Support Vector Machines with Association Rules Mining for Protein Homology Prediction. Artificial Intelligence in Medicine, special issue on computational intelligence techniques in bioinformatics, 35, 121-134, 2005.

Tomek, I., Two modifications of CNN. IEEE Transactions on Systems Man and Communications SMC-6, 769-772, 1976.

Ting, K. M., An Instance-Weighting Method to Induce Cost-Sensitive Trees. IEEE Transactions on Knowledge and Data Engineering, 14(3), 659-665, 2002.

Tremblay, G., Optimizing Nearest Neighbour in Random Subspaces using a Multi-Objective Genetic Algorithm. 17th International Conference on Pattern Recognition, 208-211, 2004.

Vapnik, V., The Nature of Statistical Learning Theory. Springer-Verlag, NY, 1995.

Weiss, G. M., Provost, F., The Effect of Class Distribution on Classifier Learning: An Empirical Study. Technical Report MLRT-43, Department of Computer Science, Rutgers University, 2001.

Williams, D. A., The analysis of binary responses from toxicological experiments involving reproduction and teratogenicity. Biometrics, 31, 949-952, 1975

Wilson, D., Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems Man and Cybernetics, SMC-2, (3)408-421, 1972.

Wolberg, W., Mangasarian, O., Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Preceedings of the National Academy of Science, U.S.A. 87, 9193-9196, 1990.

Xu, J., Suzuki, K., Max-AUC feature selection in computer-aided detection of polyps in CT Colonography. IEEE Journal of Biomedical and Health Informatics, 18(2), 585-593, 2014.

Zhang, H., The Optimality of naive Bayes. FLAIRS2004 conference, 2004.

Zhang, J., Mani, I., KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. Proceedings of International Conference Machine Learning, Workshop Learning from Imbalanced Data Sets, 2003.

Zhao, P., Hoi, S., Jin, R., Yang, T., Online AUC maximization. Proceedings of International Conference of Machine Learning, 2011.

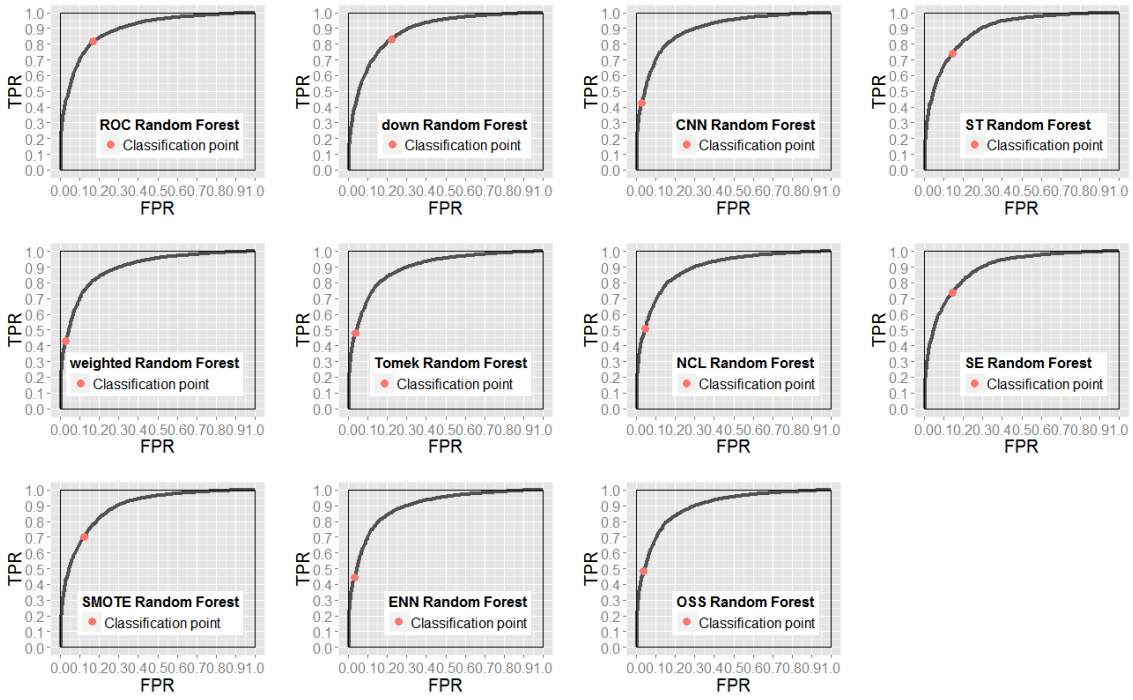
Zou, K., Hall, W., Shapiro, D., Smooth non-parametric receiver operating characteristic curves for continuous diagnostic tests. Stat Med, 16,2143-2156, 1997.

Appendix

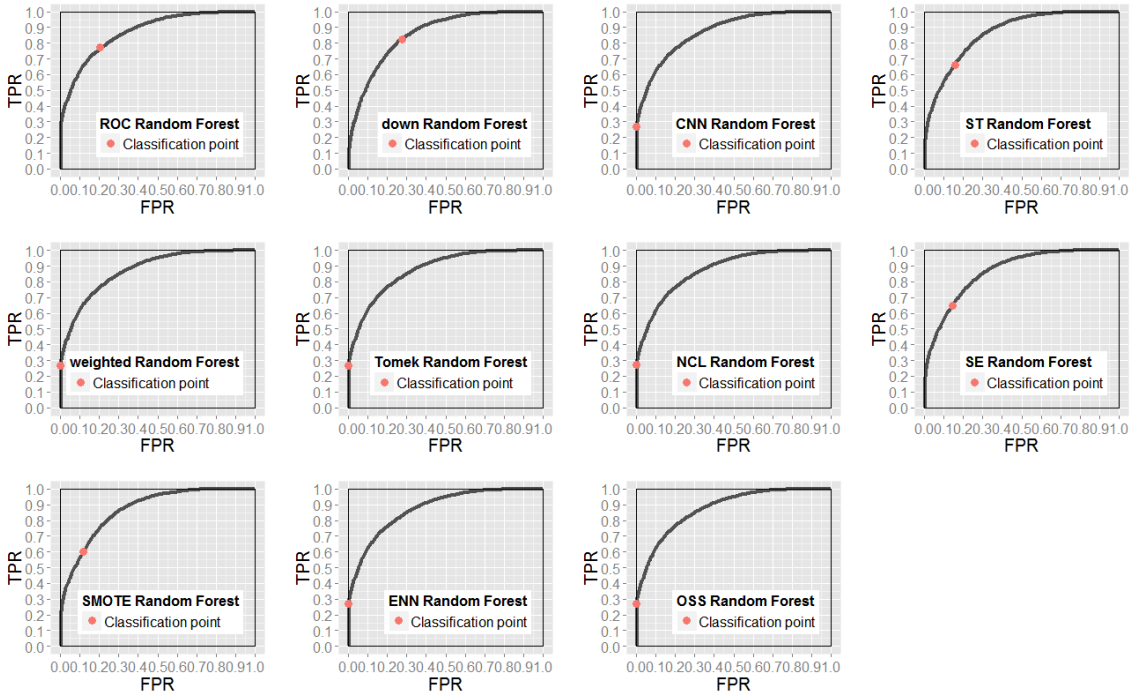
Averaged ROC curves and classification point for §5.2

For each data set, the averaged ROC curve of 11 classifiers are shown, from top to bottom, from left to right for each column, they are ROC Random Forest, weighted Random Forest, SMOTE Random Forest, down sampled Random Forest, Tomek links sampled Random Forest, ENN Random Forest, CNN Random Forest, NCL Random Forest, OSS Random Forest, SMOTE+Tomek Random Forest and SMOTE+ENN Random Forest respectively. For each curve, the red point indicates the operating point generated using 0.5 as splitting threshold.

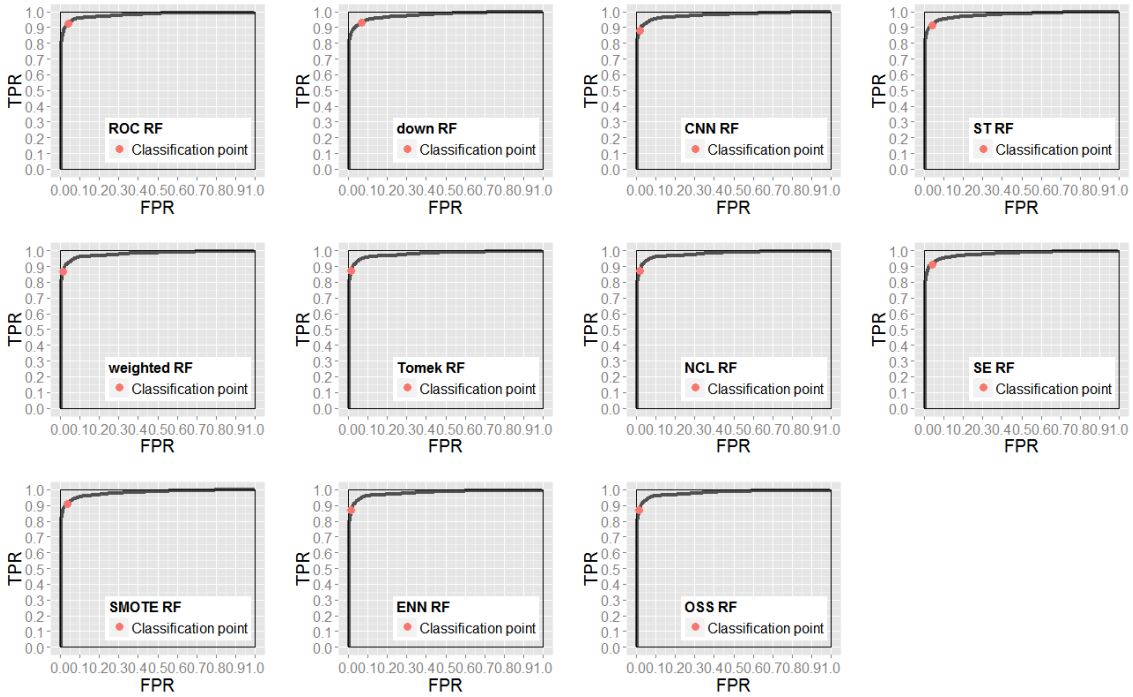
1. Red wine quality



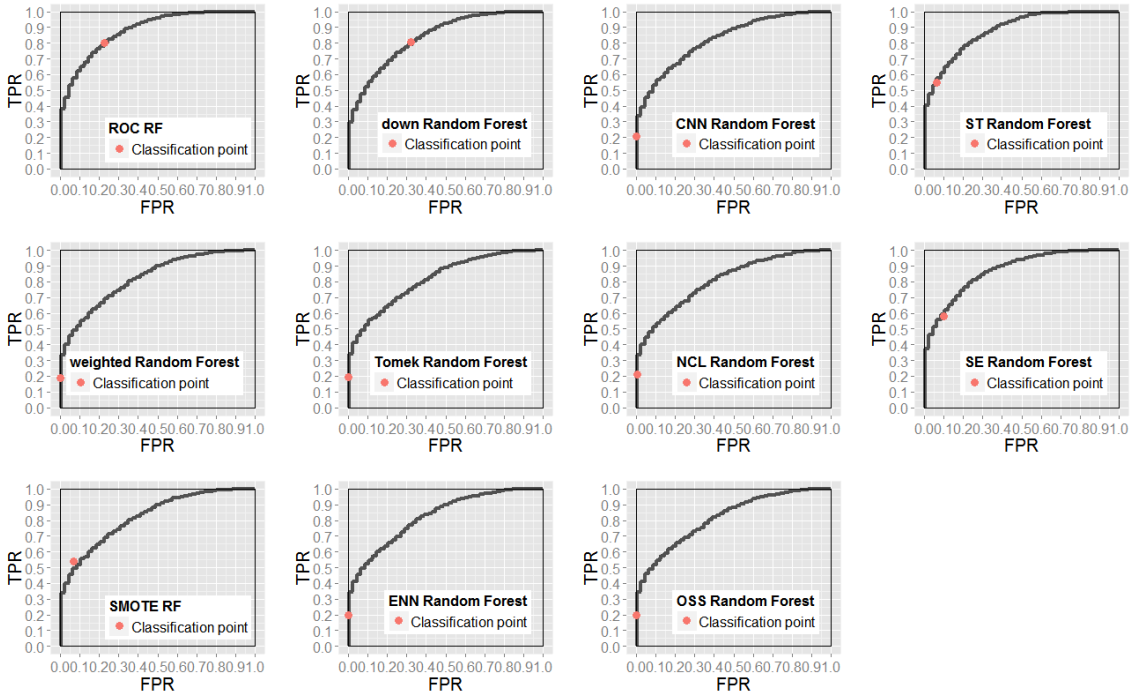
2. White wine quality



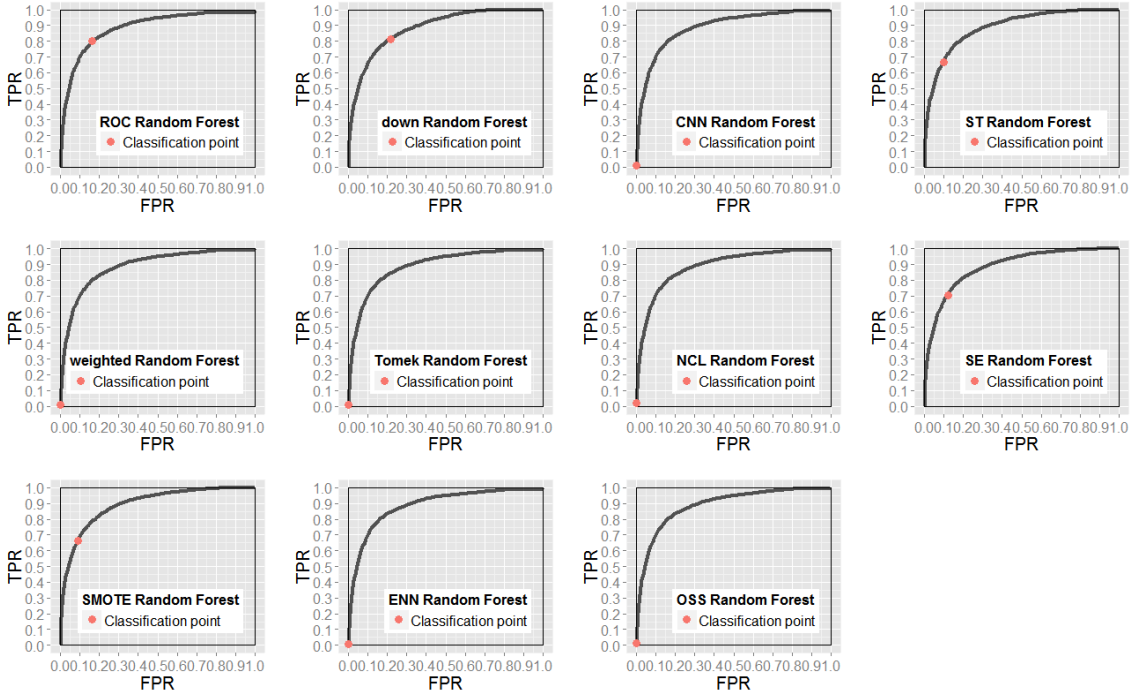
3. Breast cancer



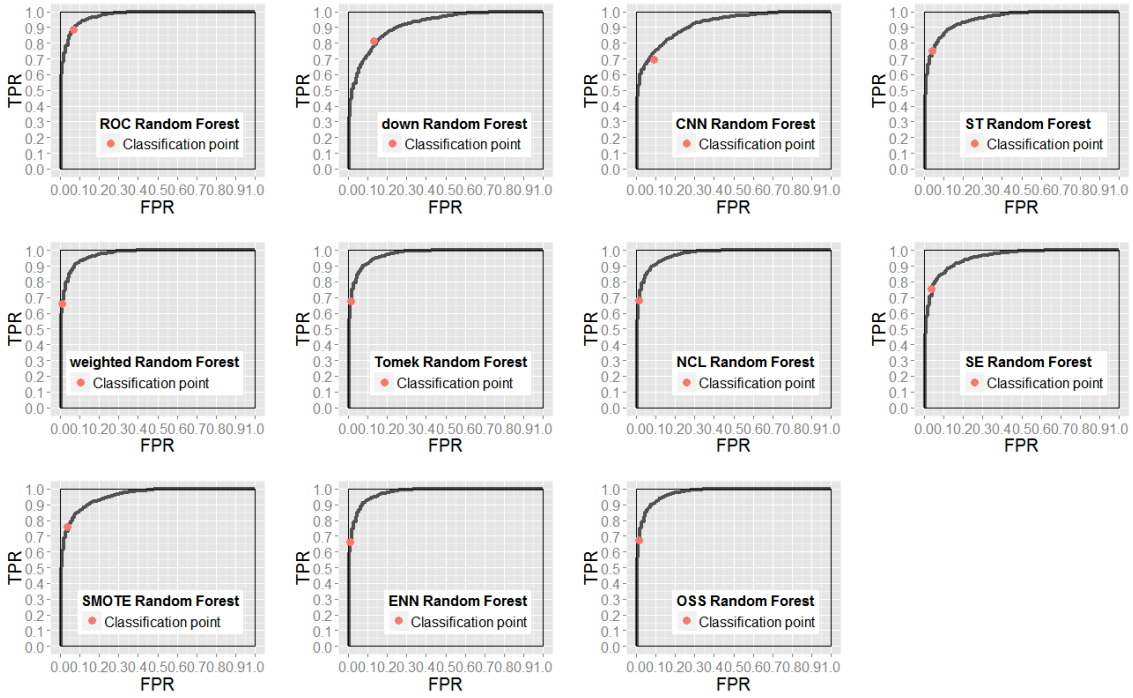
4. Connection bench



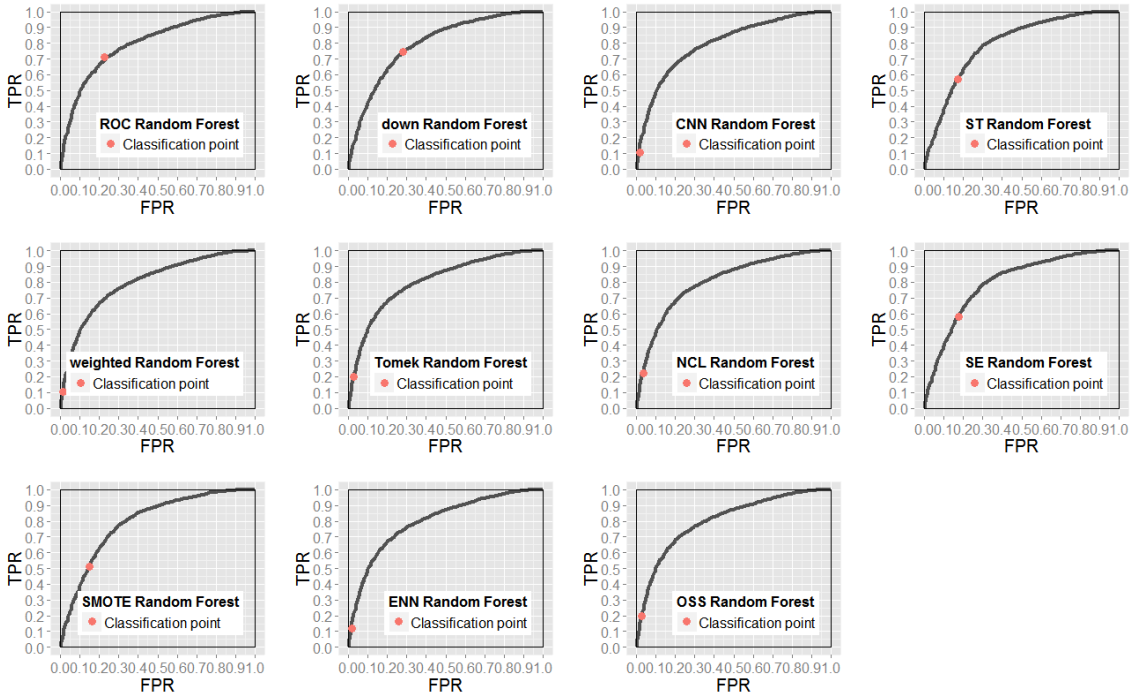
5. Ozone level



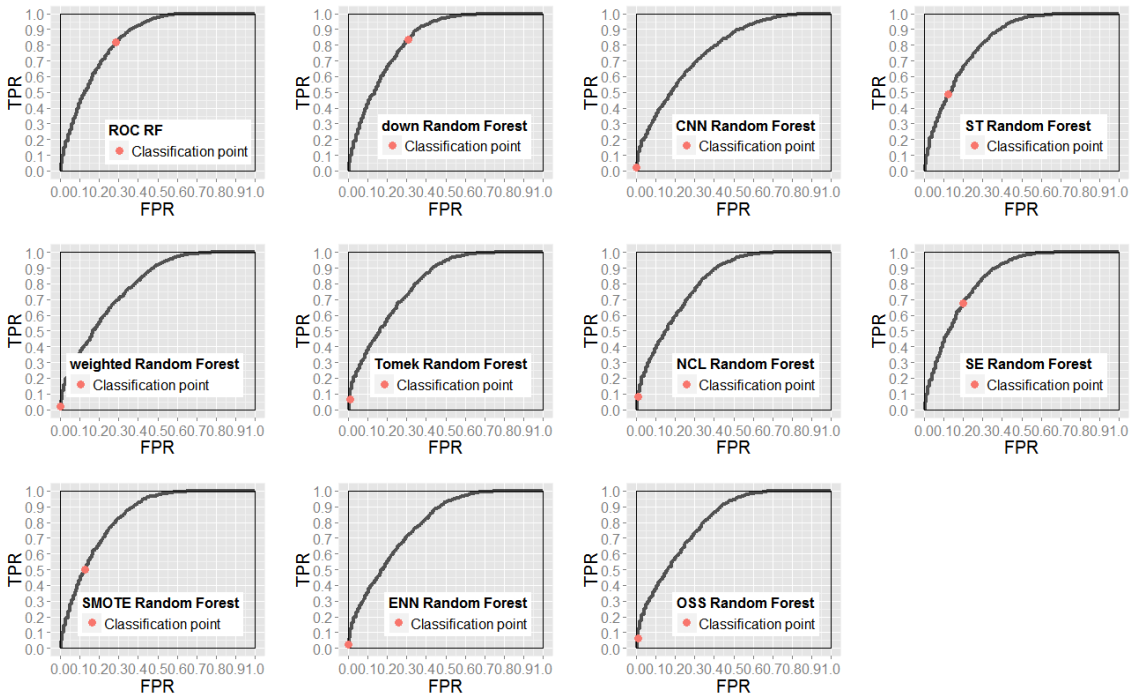
6. Ionosphere



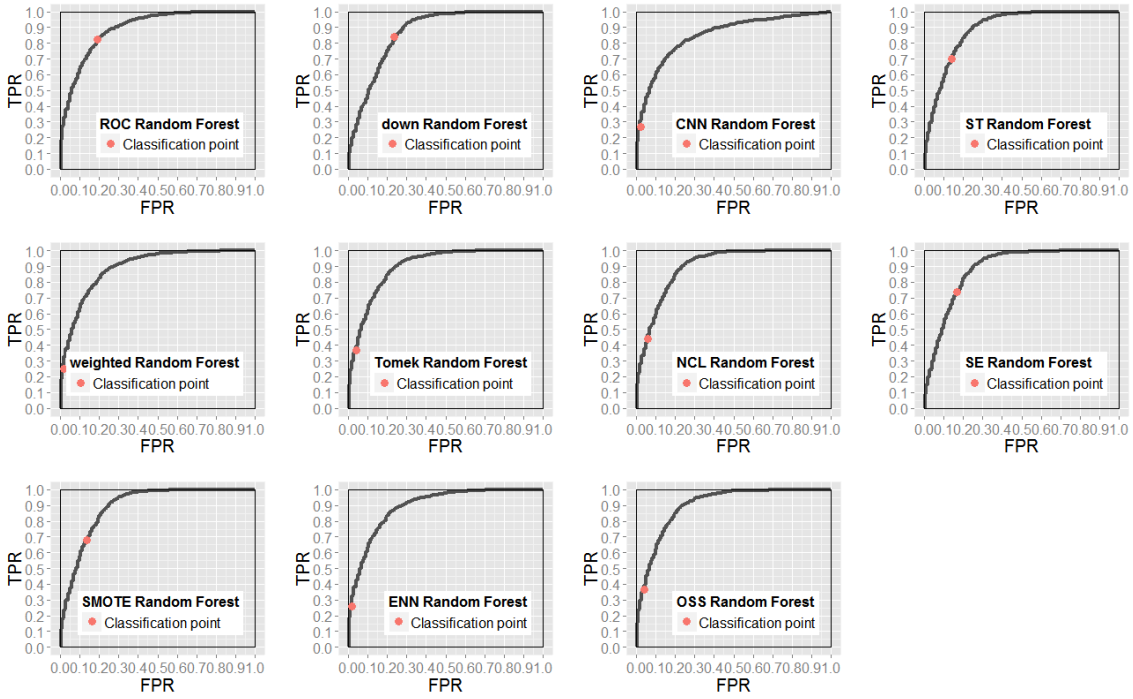
7. Pima diabetes



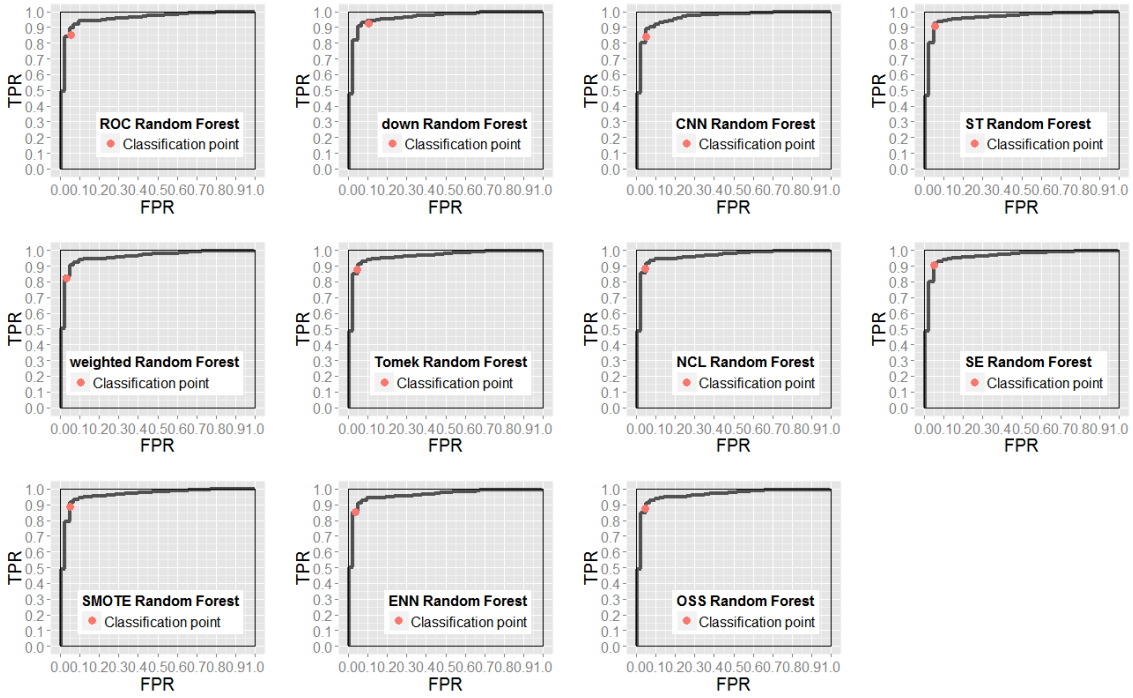
8. Spect



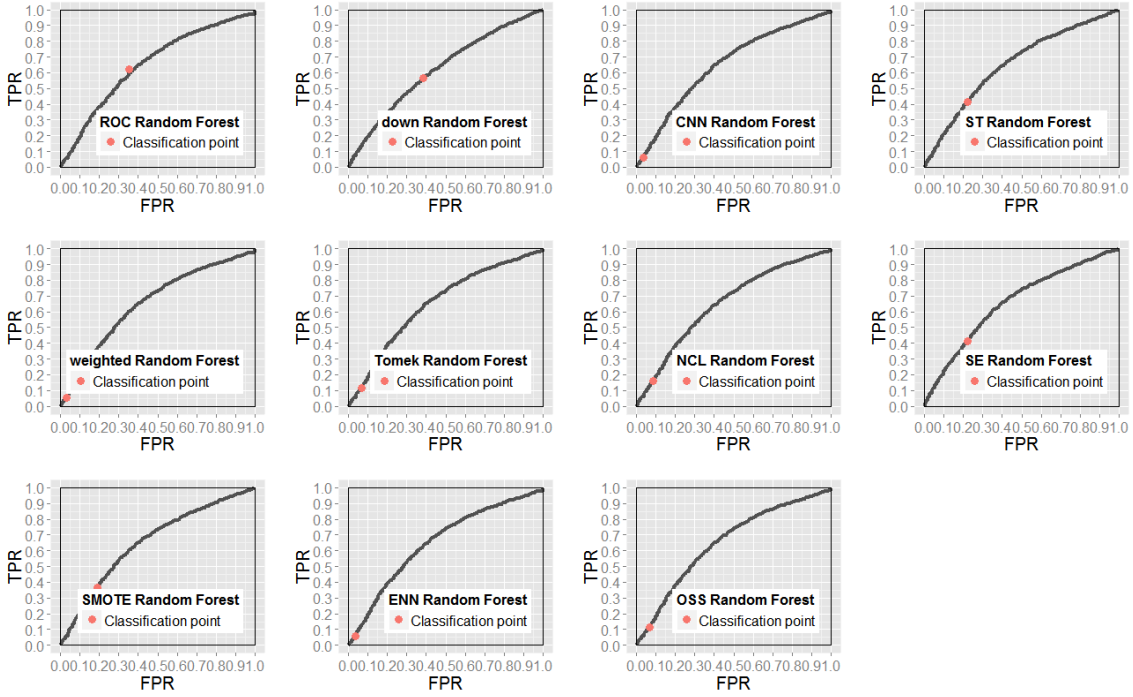
9. Vertebral



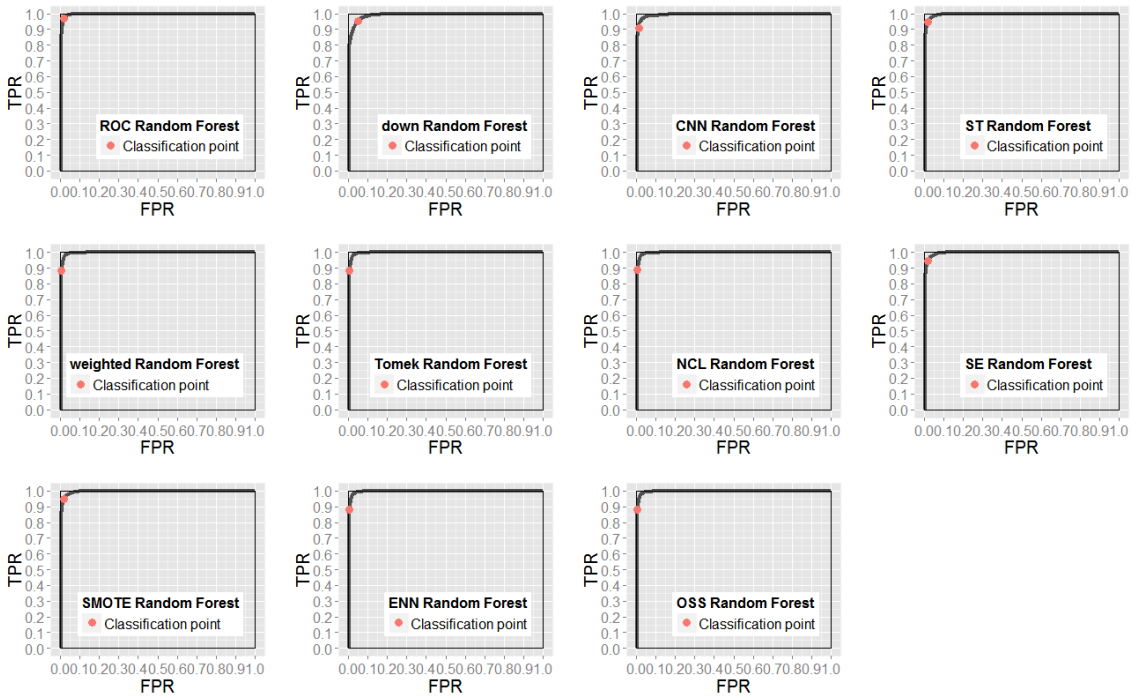
10. Breast tissue



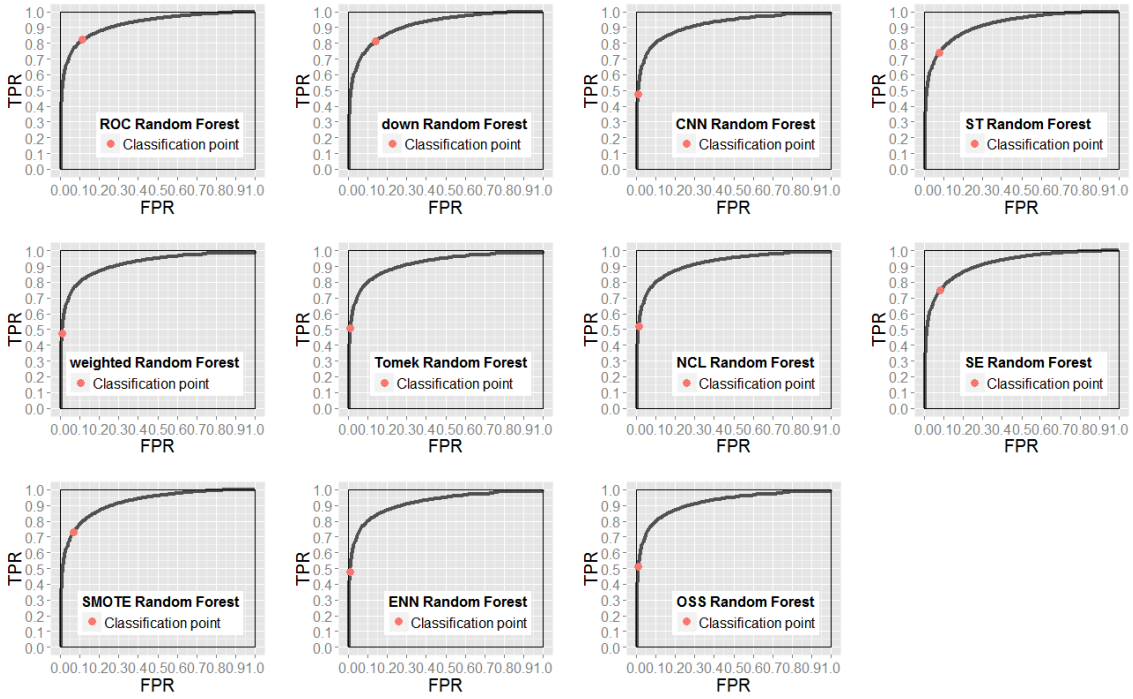
11. Haberman



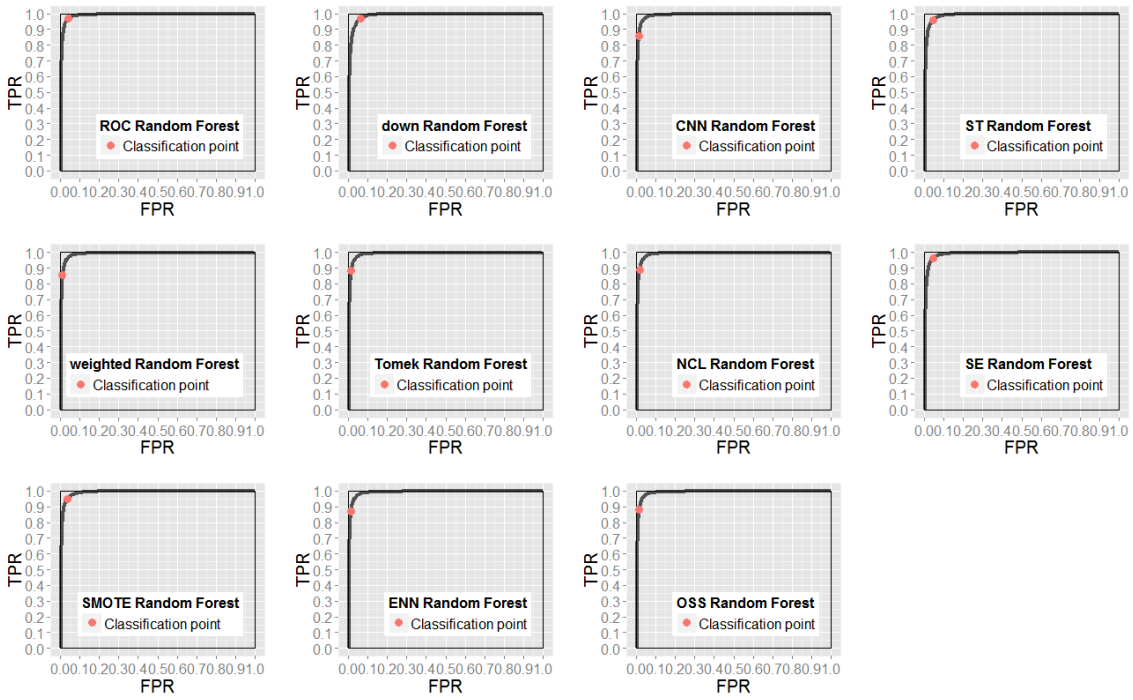
12. Banknote



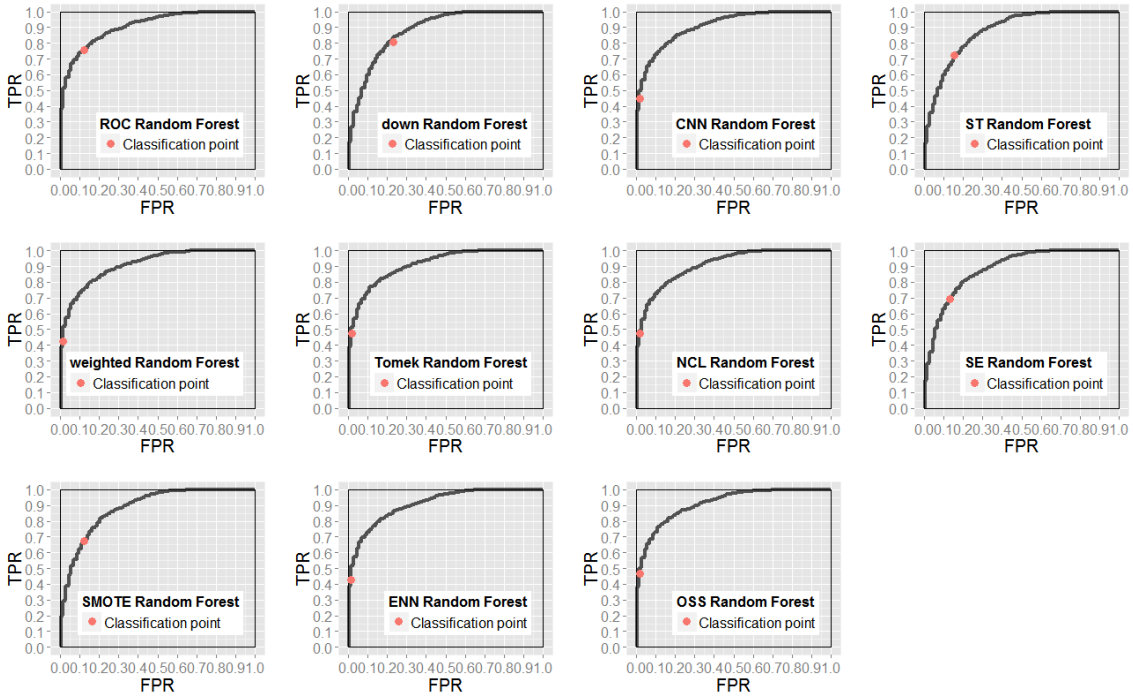
13. Magic



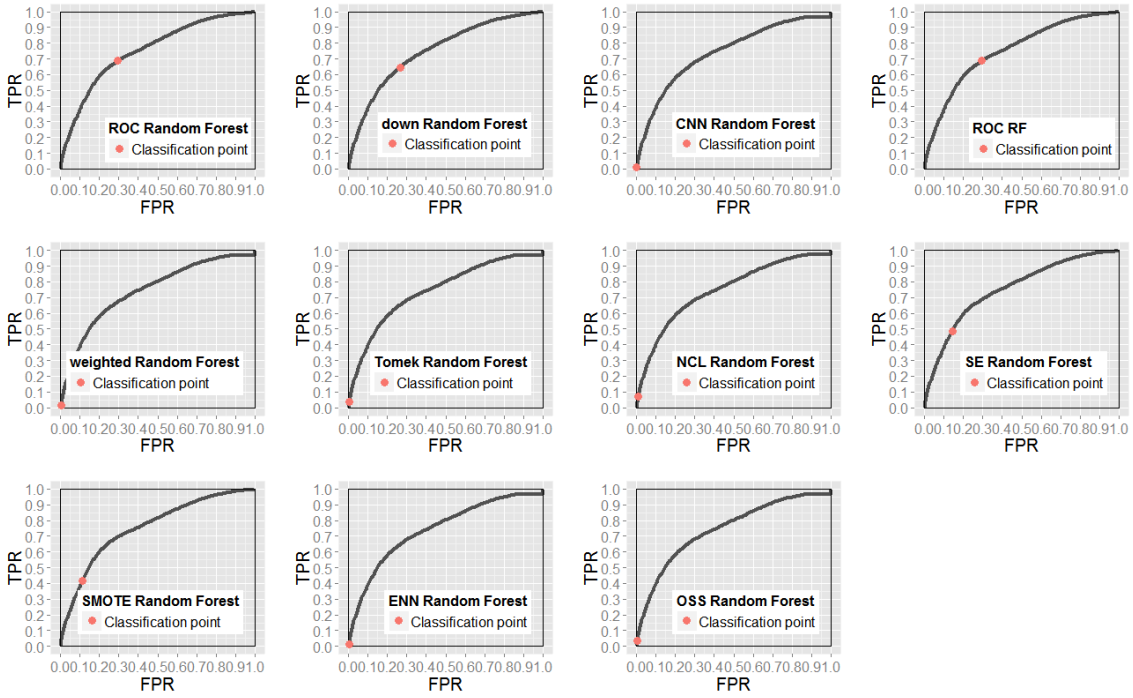
14. Page book



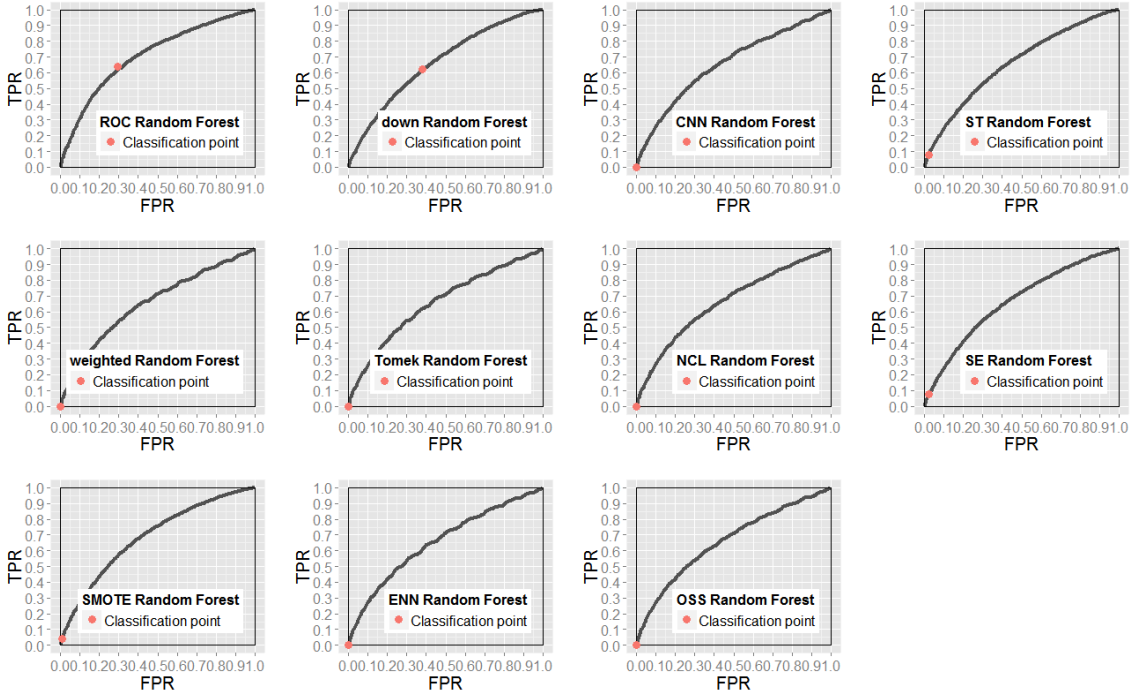
15. Parkinsons



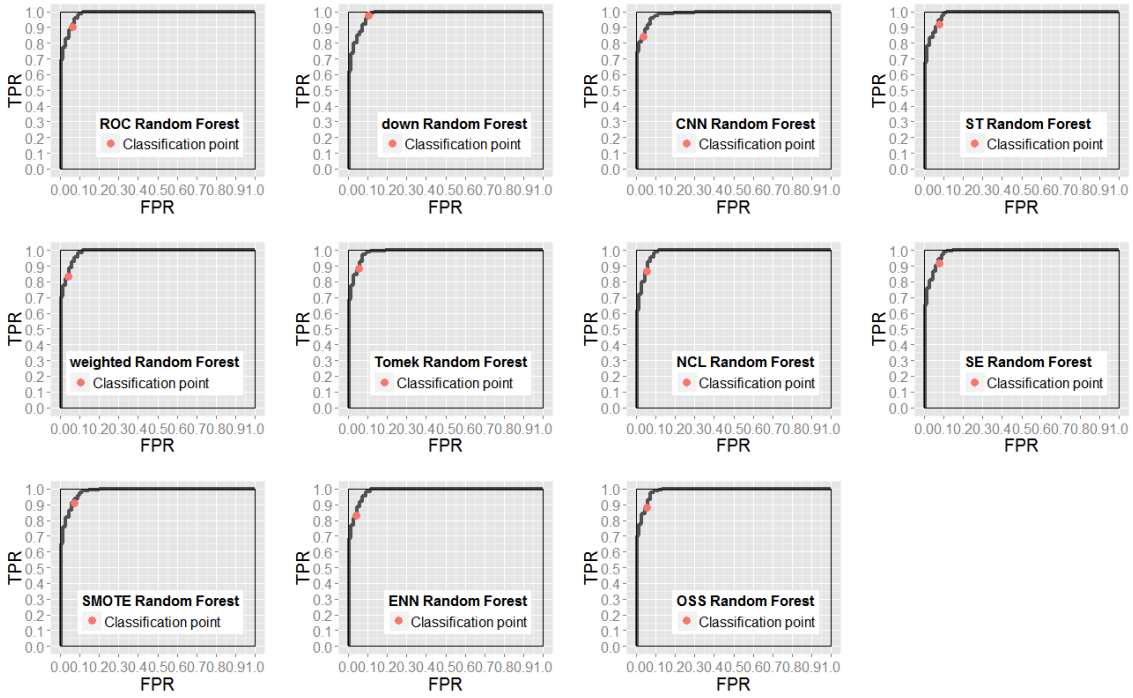
16. Seismic bump



17. Secom



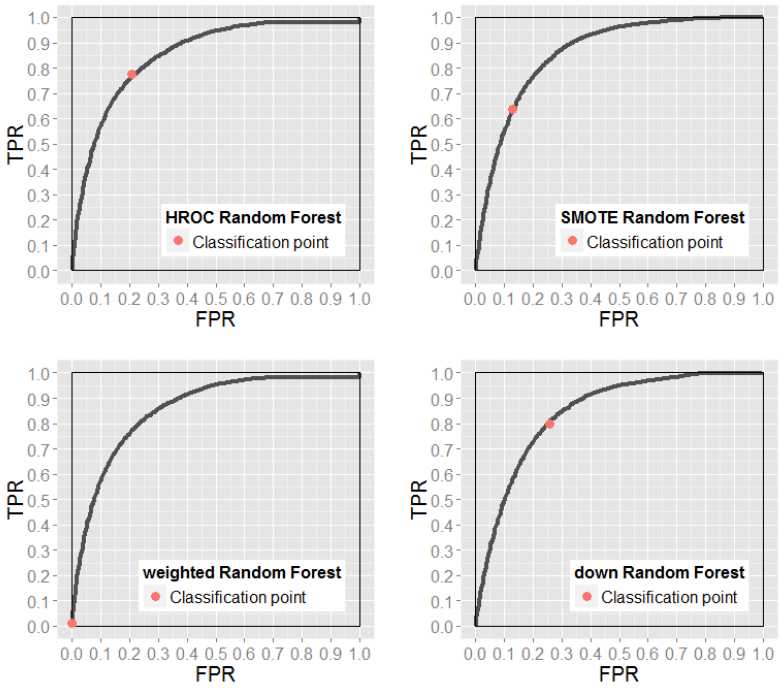
18. Seeds


ROC Random Forest
down Random Forest
CNN Random Forest
ST Random Forest
weighted Random Forest
Tomek Random Forest
NCL Random Forest
SE Random Forest
SMOTE Random Forest
ENN Random Forest
OSS Random Forest
Classification point
TPR
FPR

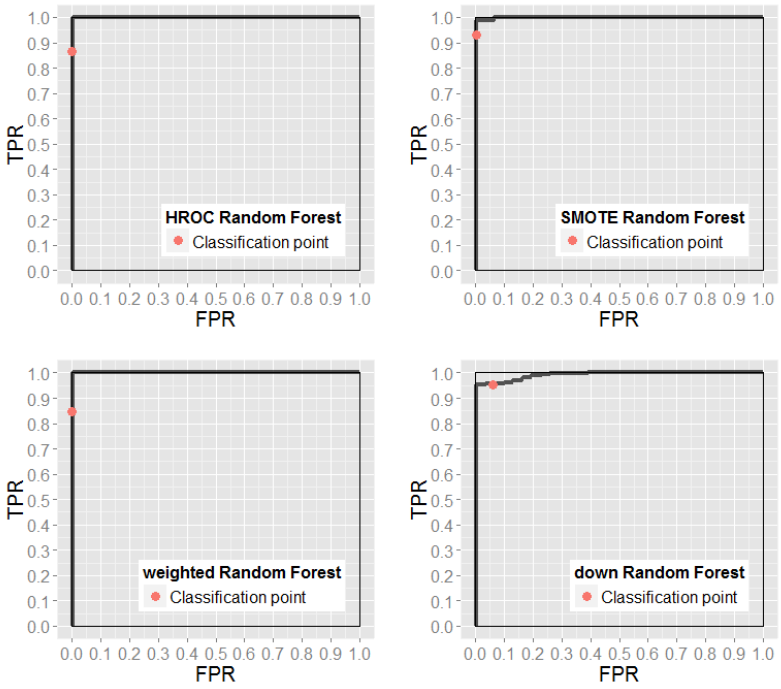
Averaged ROC curves and classification point for §5.3

For each data set, the averaged ROC curve of 4 classifiers are shown, from top to bottom, from left to right for each column, they are Hybrid ROC Random Forest, weighted Random Forest, SMOTE Random Forest, down sampled Random Forest respectively. For each curve, the red point indicates the operating point generated using 0.5 as splitting threshold.

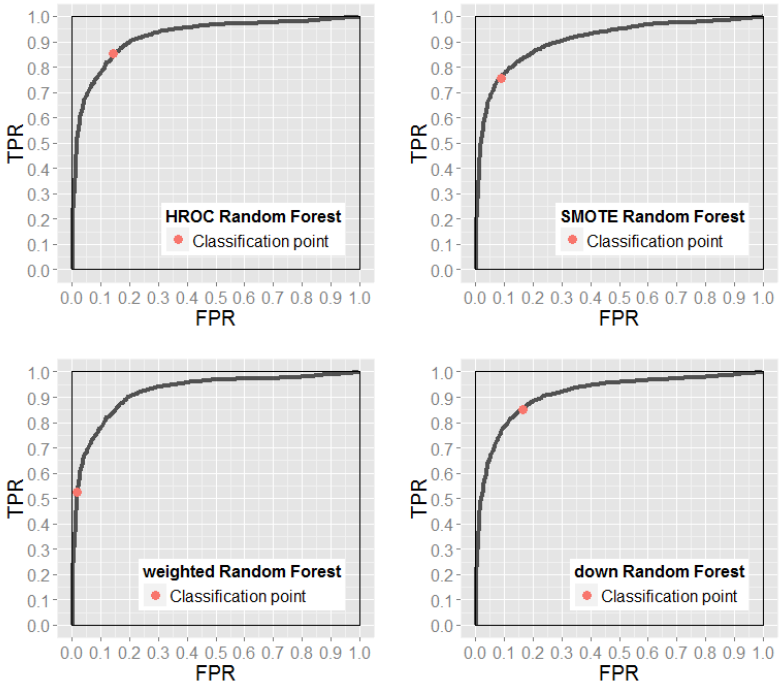
1. abalone



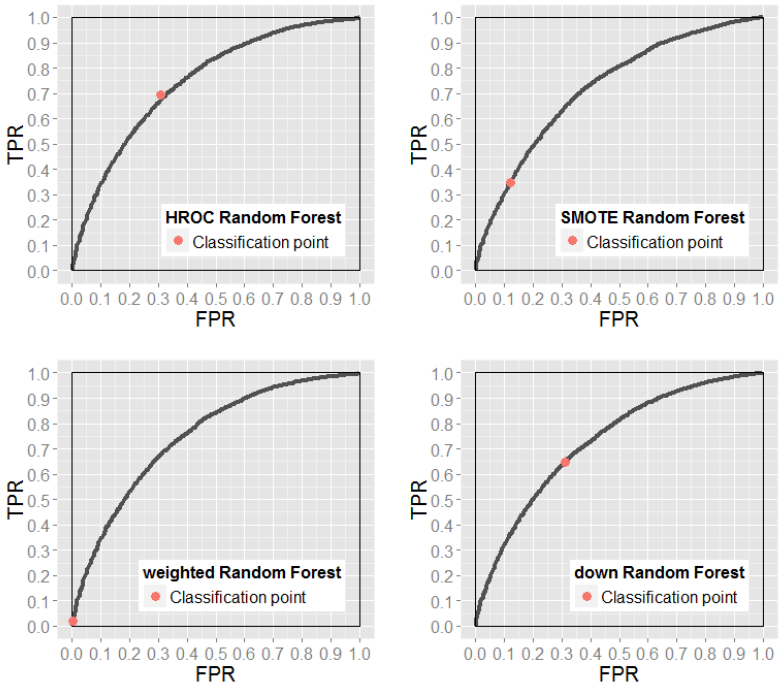
2. acute



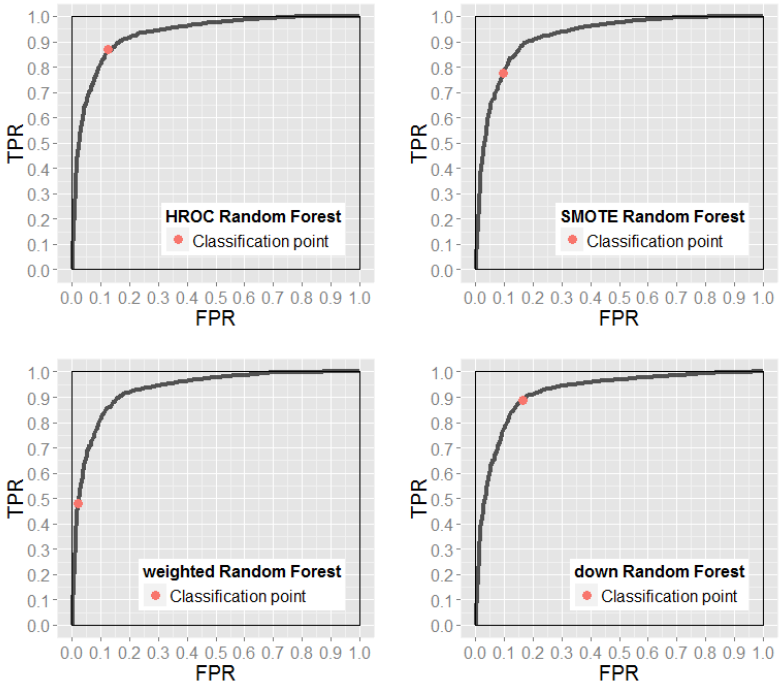
3. credit AUS



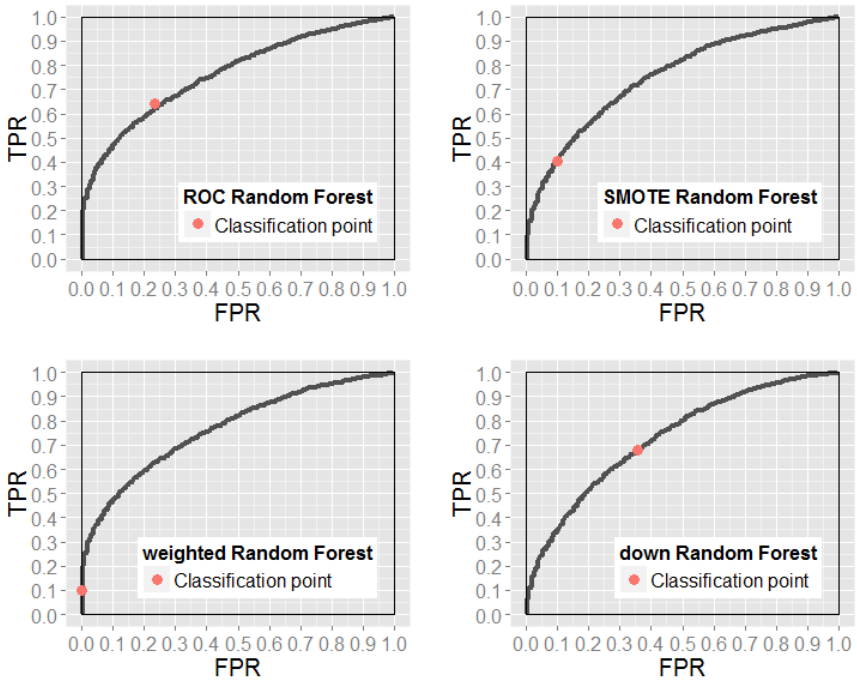
4. credit GER



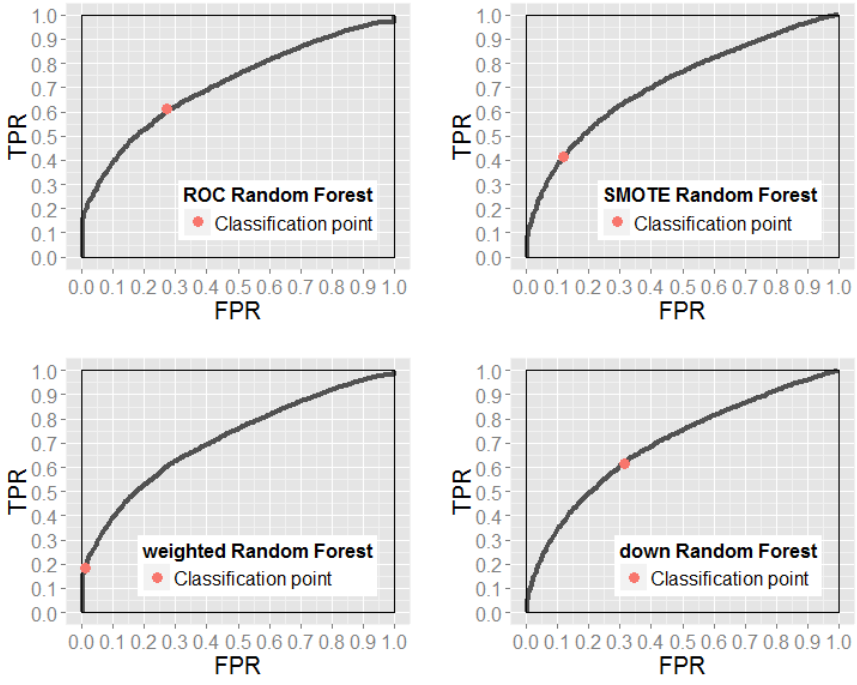
5. credit APP



6. band of cylinder



7. contraceptive



8. animals in zoo