A MACHINE LEARNING ENSEMBLE APPROACH FOR BINARY CLASSIFICATION PROBLEM IN IMBALANCED DATASET

by

SAHIDTIASADASON CHINDU

Submitted in partial requirement for
The degree of Master of Data Analytics

London Metropolitan University
Faculty of Computing
School of Computing and Digital Media

Supervisor: Professor Karim Ouazzane

ACKNOWLEDGEMENTS

I want to take this opportunity to thank my family who has always been encouraging and supporting me throughout this MSc. Without their continuous support, this would not have been possible.

I would also like to thank Professor Karim Ouazzane for his continuous guidance and advice. Despite the numerous difficulties faced, I managed to keep going due to his continuous support. His insightful suggestions, constant encouragements and guidance have assisted me in the preparation of this report.

ABSTRACT

Data is crucial for training models in the domain of machine leaning. Standard machine learning algorithms are unable to deal with the data in the presence of an imbalance distribution. The imbalanced distribution of the two classes is ubiquitous in the real world. This paper concentrates on the binary classification problem of imbalanced distribution in the response variable. This paper starts by exploring current methods that are commonly undertaken to deal with the data imbalance problems. After obtaining a good knowledge of the different methods used to tackle data imbalance, several experiments were carried out using logistic regression and decision tree classifiers. In the presence of an imbalanced dataset with a ratio of 2:23, Classical data balancing methods such as random over-sampling, random under-sampling and SMOTE were explored. From the experiments carried out it was understood that SMOTE based balancing was ineffective with logistic regression, but showed good potential when used with a decision tree model. Another finding from this study is that a data balancing method is not necessary unless a model is unable to deal with the data imbalance.

Experiments were then carried out using an ensemble-based method, particularly bagging to deal with the data imbalance problems. Bagging is performed by resampling a dataset many times, customising a prediction model for each resample dataset and then combining prediction models for these datasets into a new prediction. In this paper, we examine the effectiveness of bagging on classification trees and logistic regression models. From this study, it was found that bagging was ineffective with a logistic regression model, but was able to improve the overall model performance of a decision tree. One interesting finding was that a data balancing method was not required for a bagged decision tree even though a single decision tree failed to perform when modelled without a data balancing method.

Finally, based on the framework of an ensemble method, a new method was proposed. This method eliminates the bootstrap process in bagging and replaces it by a variable level subset selection method. Furthermore, the model fusion method in bagging was replaced by an averaging method instead of majority voting. This method was ineffective when used alongside a logistic regression model, but displayed impressive results when used with a decision tree model. The proposed method integrated with a random over-sampling method has the best performance among all the models covered in this paper.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii
ABSTRACT iii
TABLE OF CONTENTS iv
LIST OF FIGURES vi
LIST OF TABLES vii
NOMENCLATURE viii
CHAPTER 1 INTRODUCTION 1
1.1 Background 1
1.2 Motivation 1
1.3 Problem Definition 2
1.4 Aim and Objectives 3
1.5 Thesis Outline 4
CHAPTER 2 Literature Review 6
2.1 Predictive analysis 6
2.2 Class Imbalance problems 7
2.2.1 Resampling methods 11
2.2.2 Ensemble methods 12
2.2.2.1 Bagging method 15
2.2.2.2 Boosting method 17
2.3 Comparison of Ensemble method 18
2.4 Performance evaluation for imbalanced data models 20
2.5 Conclusion 21
CHAPTER 3 Methodology 22
3.1 Introduction 22
3.2 Data preparation 23
3.2.1 Exploratory Data Analysis (EDA) 23
3.2.2 Data Processing 24
3.2.4 Data Partition 25
3.3 Bagging 25
3.4 Proposed method 27
3.4.1 Data subsetting and Balancing 28
3.4.2 Modelling 30

3.4.3 Model Fusion 31
3.5 Model evaluation 31
3.6 Computing Environment 33
CHAPTER 4 Algorithm 34
4.1 Terminology and Notation 34
4.2 Logistic Regression 34
4.2 Decision Tree (CART) 35
4.3 Ensemble Method 36
4.3.1 Bootstrap 36
4.3.2 Bagging 37
4.4 Sampling methods 39
4.4.1 Random Over-sampling and bagging 39
4.4.2 Random Under-sampling and bagging 40
4.4.3 SMOTEBagging 40
4.5 Bagging (imbalanced data) 41
4.6 Proposed method (imbalanced data) 42
CHAPTER 5 Experiments and Implementations 44
5.1 Overview 44
5.2 Data preparation 46
5.2.1 Exploratory Data Analysis/ Data Pre-processing 46
5.2.2 Data preparation evaluation on model 53
5.3 Data partition 54
5.4 Logistic regression based classifier 54
5.4.1 Bagged logistic regression 57
5.4.2 Proposed Method (Logistic regression) 59
5.5 Decision tree (CART) based classifier 60
5.5.1 Bagged decision tree (CART) 63
5.5.2 Proposed decision tree (CART) 65
CHAPTER 6 Conclusion and Recommendation for Future work 68
6.1 Conclusion 68
6.2 Future works 71
References 72
APPENDIX 77

LIST OF FIGURES

Figure 3. 1 Process flow diagram for a single base classifier 23
Figure 3. 2 Process flow diagram of bagging 27
Figure 3. 3 Process flow diagram of the proposed method 28

Figure 4. 1 The Bootstrap. The original sample is shown at the top. To create a bootstrap sample one draws from the original sample with replacement (Debik, 2017). 37

Figure 5. 1 Scatterplot (‘Adult’, ‘Children’, ‘Infants’) 48
Figure 5. 2 Scatterplot (‘Duration’, ‘Lead Time’) 49
Figure 5. 3 Scatterplot (‘TempSent’, ‘ConversationRCD’) 49
Figure 5. 4 Frequency Distribution of ‘Destination’ 50
Figure 5. 5 Frequency Distribution of ‘DepAirport’ 51
Figure 5. 6 Frequency Distribution of ‘AccomType’ 51
Figure 5. 7 Frequency Distribution of ‘Transport.Type’ 52
Figure 5. 8 Logistic regression (ROC) 54
Figure 5. 9 Logistic regression (ROC) Sampling methods and base model 55
Figure 5. 10 Logistic regression (PR) Sampling methods and base model 56
Figure 5. 11 Logistic regression (ROC) Sampling methods and bagging 58
Figure 5. 12 Logistic regression (PR) Sampling methods and bagging 58
Figure 5. 13 Logistic regression (ROC) Sampling methods and Proposed method 60
Figure 5. 14 Logistic regression (PR) Sampling methods and proposed method 60
Figure 5. 15 Decision tree (ROC) Sampling methods and base model 62
Figure 5. 16 Decision tree (PR) Sampling methods and base model 62
Figure 5. 17 Decision tree (ROC) Sampling methods and bagging 64
Figure 5. 18 Decision tree (PR) Sampling methods and bagging 65
Figure 5. 19 Decision tree (ROC) Sampling methods and proposed method 66
Figure 5. 20 Decision tree (PR) Sampling methods and proposed method 67

LIST OF TABLES

Table 3. 1 Sample Dataset 29
Table 3. 2 Sample test data (1 observation) 29
Table 3. 3 Sample test data (Prediction) 30
Table 3. 4 Standard Confusion Matrix 32

Table 5. 1 Summary of agent allocation speed 45
Table 5. 2 Data Structure 47
Table 5. 3 Statistical Analysis 48
Table 5. 4 Summary of feature evaluation on models 53
Table 5. 5 Model evaluation summary (sampling and training data) Logistic regression 56
Table 5. 6 Model evaluation summary (Bagging logistic regression) 57
Table 5. 7 Model evaluation summary (Proposed method and sampling) logistic regression 59
Table 5. 8 Model evaluation summary (sampling and training data) Decision tree 61
Table 5. 9 Model evaluation summary (Bagging decision tree) 63
Table 5. 10 Model evaluation summary (Proposed method and sampling) decision tree 65

NOMENCLATURE

Abbreviations

AUPR	Area under precision-recall curve
AUROC	Area under Receiver Operating Characteristic
CART	Classification and Regression Tree
CRM	Customer Relation Management
DM	Data Mining
EDA	Exploratory Data Analysis
ERP	Enterprise Resource Planning
FN	False Negative
FP	False Positive
*k*NN	*K*-Nearest Neighbours
ML	Machine Learning
ROC	Receiver Operating Characteristic
ROS	Random Over-Sampling
RUS	Random Under-Sampling
SMOTE	Synthetic Minority Over-sampling Technique
TN	True Negative
TP	True Positive

CHAPTER 1
INTRODUCTION

1.1 Background

Machine learning can be categorised into two broad categories (Supervised and un-supervised) based on the availability of the dependent variable. Supervised learning involves discovering the relationship between the explanatory variable and the target variable; in this type of learning the dependent class is known in advance. In unsupervised learning, no dependent target class is provided by the data and involves discovering relations among data instances and features. Supervised learning can be further categorised into two groups based on the property of the target variable. If the target variable contains discrete values, then the learning task is called classification, if the target variable contains continuous numerical values, then it is called regression. This paper focuses on the classification aspect of supervised learning.

Classification is the mapping of explanatory variables to a specific target variable. A classifier can be used to understand how a target variable is related to the explanatory variable. Another application is predictive modelling where a model is used to predict an outcome based on previous outcomes. Classification is a popular and important research area in machine learning due to its importance in real-world applications. Many algorithms have been introduced over the years and applied successfully in many real-world classification problems. A classifier's performance is usually judged by its ability in predicting unseen data. This paper focuses on predictive modelling with two discrete levels (binary)

1.2 Motivation

Most classification algorithms implicitly assume that classification errors coming from different classes have the same cost and treat them equally during training. The training procedure is driven by maximising the overall accuracy. However, this is not always the case in the real-world application as rare cases in specific classes are usually more costly and vital. Failure in identifying these classes can result in huge losses. Solving this problem in real life cases is essential. This situation is commonly referred to as the class imbalance problem. The proportion of observations belonging to each class in the data-set plays a crucial role in classification.

There have been many methods proposed over the years to deal with class imbalance problems, and all these methods seem to be adequate to deal with the class imbalance problem in concrete frameworks, but there are no exhaustive comparisons of their performance among them (Galar, et al., 2011). It is still not well defined when exactly to use a data balancing method or even which data balancing method works best with a classification algorithm. Hence, it would be interesting to carry out a study to understand the effects of different data balancing methods on different classifiers in order to understand which methods work better for a particular type of classifier.

1.3 Problem Definition

A classifier algorithm is usually faced with the problem of imbalanced data-set. Imbalanced dataset problems occur when one class is underrepresented in the dataset. Example the number of the ‘No’ cases outnumber the cases ‘Yes’ cases. In such cases, the model could output a reasonable accuracy rate, but in reality, the model has terrible performance. For example, when a data is imbalanced with a ratio of 1:9, a model can obtain an accuracy of 90% just by classifying all the instances as majority cases. In most real-world application, predicting the minority class is a primary concern.

Class imbalance problem in binary classification problem has emerged as one of the challenges in the data mining community. Considering the importance of this issue, numerous methods were proposed to address this issue. These proposals can be categorised into three broad categories; algorithm level, data level and cost-sensitive methods. At the algorithm level, solutions adapt algorithms to enforce the learning with regard to the small classes. The problem with the algorithm level approach is that it cannot be used for general problems. Data-level methods include different forms of resampling techniques, which rebalance skewed class distribution by manipulating training data directly, such as random over-sampling and random under-sampling. This paper will focus on the data level approach due to its flexibility.

In extension to these approaches, another group of techniques emerges when the use of an ensemble of classifiers is considered. The objective of ensemble learning is to build a set of models on the same task and then combine them to form a better one. Particularly bagging and boosting are the most popular ensemble learning methods. They have been modified and widely used in class imbalance learning for their flexibility and effectiveness. From the individual level, an ensemble can be integrated with other class imbalance learning techniques easily. From the ensemble level, combining multiple classifiers can stabilise the prediction and achieve better generalisation. In this paper, the focus will be on data variation-based ensemble, particularly bagging. This method works by manipulating the training data in such a way that each model is trained with a slightly different training set. Because of the accuracy-oriented design, ensemble algorithm directly applied to imbalanced datasets do not solve the problem that underlay in the base classifier by themselves (Galar, et al., 2011).

1.4 Aim and Objectives

Classification algorithms are designed to maximise the accuracy; hence they tend to perform poorly in the presence of data imbalance. This paper addresses three different data pre-processing methods along with bagging methods to understand the need for data pre-processing and the effectiveness of these methods. The paper aims to review state of the art on ensemble techniques to address a binary classification problem with three different data balancing methods on two different classification algorithms to determine which data balancing method is suitable to be used along with the two different classification algorithm.

Considering the great efforts of ensemble learning methods in class imbalance learning, this thesis gives a thorough study of three different data balancing methods integrated with bagging for class imbalance problems and inspires a new ensemble algorithm for classification problems.

We use real-world CRM data which suffers from the class imbalance problem. The ratio of imbalance used in this paper is approximately 1:10. We consider logistic regression as the first base classifier as it is commonly used in CRM predictions. Another base classifier that was considered is a Decision tree since it has been widely used in imbalanced domains. According to the imbalance framework, we use the AUROC curve as the evaluation criterion.

The focus of this thesis can be described merely as exploring resampling methods integrated with ensemble-based methods for class imbalance learning. Several factors such as model type, data balancing techniques, model-combining techniques, and model evaluation techniques should be taken into account when selecting a suitable model for implementation. The primary objective here is to identify suitable data balancing methods that work well with a decision tree and logistic regression and to gain a good understanding of the effectiveness of the bagging approach to deal with the data imbalance problem. Modelling is done in R-Studio using the R language and several available packages, including, but not limited to, ‘CRAN’ and ‘pROC’.

In addition to understanding data balancing methods and ensemble methods, we will address the following questions,

- When should data balancing method be used?
- Will data balancing improve a model’s AUC even if the algorithm is able to deal with the data imbalance?
- Is bagging useful for linear classifiers such as logistic regression?
- Can a standard bagging algorithm solve the class imbalance problem?

1.5 Thesis Outline

In Chapter 2, we give an introduction to the binary classification problem. We identify conventional approaches to deal with data imbalance. We continue by exploring ensemble-based methods and their application in data imbalance. Finally, we cover the evaluation metrics that are used for performance measurement of a model designed using imbalance data.

Chapter 3 starts by introducing the steps that should be carried out in the data preparation phase. We proceed by introducing bagging and explain how it can be applied to classification tree and logistic regression models. Furthermore, we introduce a new method and present a detailed explanation of this method. Finally, we introduce the model evaluation methods used in this paper and the computing environment used.

Chapter 4 focuses on the algorithm of the different methods used in this paper. We begin by introducing the algorithm of a basic logistic regression tree and a classification tree (CART). We proceed by introducing the bootstrap and bagging algorithm. Furthermore, we introduce the algorithms of the data balancing methods used in this paper. After the introduction of the data balancing algorithm, we introduce the algorithm for bagging integrated with the data imbalance methods. Finally, we introduce the algorithm for the proposed method.
In Chapter 5, we perform a simulation study using the methods discussed in Chapter 3. We start by preparing the data and partitioning the dataset into training and testing datasets. The training sets were used to create multiple models as listed below,

- Logistic regression without data balancing
- Logistic regression with three different data balancing methods
- Bagged logistic regression without data balancing
- Bagged logistic regression with three different data balancing methods
- Proposed method with three different data balancing methods.

Similarly, the same list of models was created for decision tree classifiers. The testing dataset was used to evaluate the model performance using the evaluation metrics discussed in Chapter 3.

Chapter 6 summarises the results. Conclusions about the different models and all the findings from this paper are documented. The R scripts used in this paper are included in the Appendix.

CHAPTER 2
LITERATURE REVIEW

2.1 Predictive analysis

Predicting the future using available data can help companies have a competitive edge (Morelli, et al., 2010). Companies can predict the future by integrating predictive analysis methods with their CRM database. Predictive analysis plays a significant role in extracting useful information and utilising it to model sales pattern, customer behaviour and other trends for the future.

Predictive analysis is closely linked to ML, as it utilises patterns in data to make predictions, where machines use current data and apply them to a model to predict trends in the future. Over the years, many research papers on different ML techniques for predictive analysis were released. (Sharma, et al., 2016) proposed a regression model for sales prediction using various customer related attributes. The application of this model resulted in improved business quality and profit. (Signorini, et al., 1999) carried out predictions on the survival of patients using multiple logistic regression, (King & Zeng, 2001) examined rare events data with a binary dependent variable using logistic regression, (Raghupathi & Raju, 2010) used neural network application for bankruptcy predictions and concluded that the configuration with two hidden layers has a superior classification accuracy.

Some of the factors that should be considered for predictive analysis are the quality and size of the training dataset, a clear definition of the concept that is to be predicted and the training dataset must be representative of the test dataset. Generally, the training data come from the past, but the test data arise in the future. If the concept that is to be predicted is not stable over time, then predictions are likely not to be useful (Elkan, 2013). Predictive analysis is divided into two broad categories, classification and regression (Gutierrez, 2014). Regression is used for responses that are quantitative while classification is used for responses that have a few known values. In this paper, the focus would be on classification, specifically binary classification. The task of classifying elements to one of two available groups is known as binary classification. In its simplest form, it reduces to the question: given a pattern x drawn from a domain X, estimate, which value an associated binary random variable, $\mathcal{Y} \in \{\pm 1\}$ will assume (Smola & Vishwanathan, 2008).

In the real world, there are many binary classification problems such as medical testing, risk assessment and fraud detection. However, different cases have distinct characteristics that may lead to different difficulties of the problem. One critical characteristic is the degree of imbalance of two classes in datasets.

2.2 Class Imbalance problems

In the real world, the imbalanced distribution of the two classes is ubiquitous. Neglecting the class imbalance distribution and modelling traditional algorithms for binary classifications leads to poor performance on the datasets (Han, et al., 2005), which leads to the unsatisfied suboptimal result (Chawla, et al., 2004) that only the majority class was identified well, and the minority class was ignored. In binary classification problems, the priority is usually to identify the minority class rather than the majority (Han, et al., 2005), especially when the cost is expensive for the failure in recognising the minority ones.

The fundamental issue with imbalanced data is the ability of imbalanced data to significantly compromise the performance of most standard learning algorithms (He & Garcia, 2009). Majority of the standard algorithms assume or expect balanced class distribution or equal misclassification costs. Therefore, when presented with complex imbalanced datasets, these algorithms fail to represent the distributive characteristics of the data adequately and resultantly provide unfavourable accuracies across the classes of the data (Wang, 2011).

(Zhou & Lai, 2009) investigated the effect of commonly used binary classification methods on datasets with different degree of imbalance and was found that Support Vector Machine could maintain proper balance and classification accuracy on datasets with various degree of imbalance. Methods such as linear regression and K Nearest Neighbour are sensitive to the imbalance of the datasets. Another research carried out by (He & Garcia, 2009) using datasets with various degree of imbalanced proved that in the presence of the imbalance datasets, most of the standard classifier learning algorithms, such as *k*-nearest neighbour, decision tree, back-propagation neural networks, failed to give optimal results. He also stated that ensemble learning algorithms are useful and powerful methods to deal with the imbalance class problem and stressed the importance of balancing data with practical techniques. (Alirezaee, et al., 2012) proved that when a neural network is trained with an imbalanced dataset, there is a significant difference between the prediction accuracy of majority class and minority class. When tested with different sampling methods, the over-sampling method works well in identifying the minority class in the neural network.

The imbalanced distribution pervasively exists in real-world applications. Although it is often and should be the focus of research efforts, some studies have shown that the minority class can be learnt quite well for specific imbalanced datasets with simple class distributions, such as linear separable problem (Batista, et al., 2005). These studies suggest that the imbalance degree is not the only factor responsible for performance reduction of learning algorithms. A learner's sensitivity to the class imbalance was found to depend on the data complexity and the overall size of the training set.

Data complexity comprises issues such as overlapping (Batista, et al., 2005) and small disjuncts (Jo & Japkowicz, 2005). The degree of overlapping between classes and how the minority class examples distribute in data space aggravate the adverse effect of class imbalance. The small disjuncts problem is also associated with the within-class imbalance (Jo & Japkowicz, 2005). In terms of the size of the training data, a vast domain has a good chance that the minority class is represented by a reasonable number of examples and thus may be less affected by imbalance than a small domain containing very few minority class examples.

Conventional learning algorithms are usually not suitable in the presence of imbalanced class distributions. Most algorithms are focused on maximising overall accuracy, and this usually leads to a high probability of the classifier producing a meagre recognition rate of the minority class (Wang, 2011). The adverse effects of class imbalance on classifiers such as neural networks (Visa & Ralescu, 2005) and decision tree (He & Garcia, 2009) has been reported in the literature.

The decision tree model is a recursive and top-down greedy search procedure (Wang, 2011). It functions by selecting the variables that can best split the classes of examples at individual nodes. As a result, the training dataset is successively divided into smaller subsets, corresponding to disjoint rules for each class. However, the majority class dominates the leaves, and successive partitioning results in precise rules for the minority class. Some papers specialised in the performance analysis of decision trees in the presence of imbalanced data (Chawla, et al., 2004).

To deal with the class imbalance problem, different types of methods were introduced. These methods can be categories into three broad groups; algorithm-level approaches, data-level approaches and cost-sensitive learning (Galar, et al., 2011).

Data-level methods involve resampling techniques to rectify the skewed class distributions. They are efficient and straightforward, but their effectiveness depends on the solving problem considerably (Wang, 2011). Generally, the use of sampling methods in imbalanced learning applications consists of the modification of an imbalanced dataset by some method in order to provide a balanced distribution in the target variable. Studies were carried out of several classifiers, and it was found that a balanced dataset provides improved overall classification performance compared to an imbalanced dataset (Laurikkala, 2001), (Błaszczynski & Stefanowski, 2015) and (Zhou & Lai, 2009). Their results justify the use of sampling methods for imbalanced learning.

Algorithm level methods involve modification of the training mechanism to improve the accuracy of the minority class. Various algorithm-level methods address the class imbalance by modifying their training mechanism with the most direct goal of better accuracy on the minority class such as one-class learning and cost-sensitive learning algorithms. Algorithm-level solutions require different treatments for specific kinds of learning algorithms and this restricts their use in many applications since it is not known which algorithm would be the best choice beforehand in most cases. A good solution is expected to be able to deal with most given problems effectively (Visa & Ralescu, 2005). Therefore, it is more meaningful to explore solutions that are applicable to a broader scope.

With respect to cost-sensitive methods, decision trees and neural networks have been made capable of processing different misclassification costs of classes, such as adjusting the decision threshold (Japkowicz, 2000), making the tree split criteria cost sensitive (Ting, 2002), and applying cost-sensitive modifications to the error minimization function or outputs of the neural network (Zhou & Liu, 2006). A cost-sensitive learning method is preferred only when the exact misclassification costs of classes are available. In many situations, however, this information is unknown (Wang, 2011). Given misclassification costs for each class, a cost-sensitive method considers the cost information and treats misclassifications differently during the training procedure. It is natural to apply cost-sensitive learning methods to class imbalance problems by assigning a more significant cost to the minority class. (Maloof, 2003) showed that learning from imbalanced datasets and learning with unequal costs can be handled similarly. The cost matrix plays an essential role in cost-sensitive methods, a numerical representation of the penalty when an example is mislabelled from one class to another (Zadrozny & Elkan, 2001). In many real-world problems, unfortunately, such cost information is not available.

Due to the lack of guidance for the choice of resampling techniques, the idea of combining multiple classifiers arose, in order to make up each other's weaknesses (Seiffert, 2008). Ensemble learning thus has become one of the major techniques in class imbalance learning. It allows individual classifiers to emphasise the minority class regions differently and exploits their combination to lower down the risk of overfitting for better generalisation. The basic idea of the ensemble method is to use many classifiers each of which is specialised for a subset of the data in the problem space and then use their consensus vote as the classification, leading to better performance due to the reinforcement of the classifier in error-prone problem spaces (Parvin, et al., 2013).

Many ensemble methods were proposed with varying degrees of success to deal with imbalanced data. Existing ensemble solutions so far mainly focused on how to rebalance the training subset for each base classifier from the data level and how to make the ensemble cost-sensitive. From the individual level, it can be easily adapted for emphasising the minority class by integrating different resampling techniques (Liu, et al., 2009)) or by applying different misclassification costs (Chawla & Sylvester, 2007). From the ensemble level, the idea of combining multiple classifiers can make up every single classifier's weaknesses and lower down the risk of overfitting by reducing the error variance (Brown, et al., 2005). Therefore, an ensemble is believed to outperform a single classifier with a more stable performance by learning from different views (Wang, 2011).

In order to establish a less specific, simple and effective way to deal with class imbalance problems, data-level approach integrated with ensemble learning methods should be used for their excellent flexibility and effectiveness. The next two sub-section explores some of the most effective data-level methods for dealing with imbalanced data.

2.2.1 Resampling methods

Resampling is a group of data-level techniques that adjust the distribution of training data directly. It includes over-sampling, under-sampling and hybrid methods. Over-sampling replicates or creates new instances until the imbalance is eliminated. Likewise, under-sampling eliminates some instances from the majority class until the dataset is relatively balanced. The hybrid method combines both the sampling methods. Previous researches have shown that resampling method has drawbacks, under-sampling may throw out potentially useful data (He & Garcia, 2009), while over-sampling artificially increase the size of the dataset and consequently, worsens the computational burden of the learning algorithm and could also lead to overfitting (Ganganwar, 2012). Even though there are drawbacks, several papers about resampling techniques that study the effect of changing class distribution to deal with imbalanced datasets empirically prove that the application of a pre-processing step in order to balance the class distribution usually leads to a positive solution (Batista, et al., 2004). In the three different techniques of resampling, there exist several different proposals. The following are some of the sampling techniques that have been used in conjunction with ensemble methods.

- Random over-sampling: This method balances the class distribution by randomly replicating minority class instances. Several studies are proving that this method increases the likelihood of overfitting (Chawla, et al., 2002).

- Random under-sampling: This method aims to balance the class distribution by randomly eliminating the majority class instances. Several authors claim that this method discards potentially useful data that could be important for the induction process (He & Garcia, 2009).

- Synthetic minority over-sampling technique (SMOTE): This is an over-sampling method that creates new minority class instances by interpolating several minority class instances that lie together. SMOTE (Chawla, et al., 2002) creates instances by randomly selecting one of the *k* nearest neighbour (*k*NN) of a minority class instance and the generation of the new instance values from a random interpolation of both instances. Thus, the overfitting problem is eliminated and causes the decision boundaries for the minority class to be spread further into the majority class space. SMOTE has shown great success in many applications but has been reported to have an over-generalisation problem (Hu, et al., 2009).

- Modified synthetic minority over-sampling technique (MSMOTE): This method is a modified version of SMOTE. The MSMOTE algorithm divides the instances of the minority class into three groups, safe, border and latent noise instances by the calculation of the distances among all instances. For safe instances, the algorithm randomly selects a data point from the kNN for border instances, and it only selects the nearest neighbour; finally, for the latent noise instances, it does nothing.

Several papers have discussed the effectiveness of resampling techniques and their combination with ensemble methods. Resampling techniques have shown to improve classification performance for most imbalanced datasets (Ganganwar, 2012). They work independently with learning algorithms and are thus more versatile that algorithm-level methods. However, it is always an issue of tuning them effectively. It is unclear which resampling method performs better and which sampling rate should be used. Some studies found that resampling to full balance is not necessarily optimal, and the best resampling rate varies with problem domains and resampling techniques (EstaBrooks, et al., 2004). Their performance also depends on different classifiers and evaluated measures (Hulse, et al., 2007).

2.2.2 Ensemble methods

Ensemble methods combine the output of multiple models to achieve a better predictive performance than any of the constituent learning algorithm alone. Most ensemble methods use a single base learning algorithm to produce homogeneous base leaners, but there are also methods that use complex multiple learning algorithms to produce homogeneous leaners (Zhou, 2012). Ever since the discovery of ensemble methods, they have been used as a universal solution to improve regression and classification models in terms of accuracy and stability (Zhou, 2012). (Geman, et al., 1992) proved that a combination of several Neural Networks could reduce the variance of the average regression model. The extension to classification problems was straightforward after the formulation of a bias-variance decomposition for zero-one loss functions (Domingos, 2000).

The basic idea of classifier ensemble learning is to construct multiple classifiers from the original data and then aggregate their predictions by voting (for classification) or averaging (for regression) when classifying unknown samples to yield high performance. The primary motivation for combining classifiers in redundant ensembles is to improve their generalisation ability. Each component classifier is known to make errors with the assumption that it has been trained on a limited set of data; however, the patterns that are misclassified by the different classifiers are not necessarily the same. Compared to an individual classifier, ensembles have shown to have superior performance in most cases (Wan & Yang, 2013). Two of the most common ensemble algorithms are Bagging (Breiman, 1996) and AdaBoost (Freund & Schapire, 1996). The application of these two methods has led to significant improvements in several classification problems (Oza & Tumer, 2008).

Ideally, ensembles should be made of classifiers with high accuracy which differ as much as possible (Polikar, 2007). This follows the idea that if the classifiers make different mistakes, the total error would be reduced when the results are derived using voting for classification problems. (Wan & Yang, 2013) claims that it is only worth combining multiple models which are significantly different from each other in terms of the level of disagreement between them. It is commonly agreed that the success of ensemble is attributed to diversity- the degree of disagreement with the ensemble (Brown, et al., 2005). In the regression context, it has already been quantified and measured in terms of covariance between individual learners by decomposing the ensemble error (Brown, et al., 2005). In the classification context, the effect of diversity has not been clearly defined, but (Galar, et al., 2011) states that diversity among classifiers is crucial to forming an ensemble.

For example, bagging creates diversity by bootstrapping different training subsets; AdaBoost achieves diversity by altering the distribution of training examples for each classifier to avoid making the same errors. Both algorithms have shown great success in ensemble learning area (Chawla & Sylvester, 2007). The underlying principle of creating a classification ensemble can be generally described as "individual classifiers make errors on different examples" (Polikar, 2007). There is no neat theory to rigorously define how differences between individual classifiers contribute to overall classification accuracy, depending on the type of error function and choice of combing methods. As such, it is hard to achieve an agreed definition of diversity for classification ensembles.

Despite the arguments, many researchers believe the central concept that enables ensemble methods is diversity. It is pointless in combining models with the same output (Polikar, 2007). Diversity among classifiers can be achieved in many ways, such as training classifiers with different subsets of the features (Díez-Pastor, et al., 2015) and training multiple different classifiers on the same dataset and combining their predicitions. One conventional method to achieve diversity is to use different training data subsets obtained by resampling of the original training data and constitutes the link between ensemble systems and bootstrap techniques. The partial understanding of diversity in classification ensembles leads to various metrics proposed to quantify diversity and continuous research efforts in the role of diversity in classification accuracy from different perspectives. Despite the resulting benefits of diversity in dealing with classification problems, some studies show counterintuitive empirical results due to its vague link to overall accuracy. For example, (Kuncheva & Whitaker, 2003) found a weak relationship between diversity and overall accuracy through experimental discussions and raised the doubt of the usefulness of diversity measures in training classification ensembles. The similar observation also happened in a study done by (Garcia-Pedrajas, et al., 2005). They proposed an evolutionary multi-objective method for designing ensemble. Their results showed that considering diversity as one of the objectives does not bring definite performance improvement. (Chen, 2008) stated that diversity highly correlates with the classification error only when diversity is small, and its effect is reduced out of a specific range.

In recent years, ensemble learning methods have become a popular way of advancing the classification of imbalanced data. They can be easily adapted for emphasising the minority class by using resampling techniques from the data level ((Chawla, et al., 2004) & (Liu, et al., 2009)) or by applying different misclassification costs from the algorithm level (Chawla & Sylvester, 2007). Besides, the idea of combining multiple classifiers itself can lower down the risk of overfitting and reduce the error variance (Zhou, 2012). Particular techniques, such as over-sampling and under-sampling, are often used with ensembles to improve the generalisation of predicting the minority class. Ensemble classifiers combined with data-level sampling methods were proved to be more effective than data sampling techniques to enhance the classification performance of imbalanced data (Khoshgoftaar, et al., 2015).

Bagging (Breiman, 1996)) and Boosting (Freund & Schapire, 1996) are the two most popular ensemble learning methods. In class imbalance learning, both of these classical methods become less effective in recognising rare cases (Liu, et al., 2009). Hence, these methods have been modified and widely used in class imbalance learning for their flexibility and effectiveness.

2.2.2.1 Bagging method

The idea of bagging is to inject model diversity at the data level, by training the ensemble classifiers on bootstrap replicates of the training dataset. Then the outputs of the classifiers are combined using the majority vote for classification. Bagging is most effective when it is used with unstable non-linear learning algorithms such as decision trees and neural networks, where small changes in the training dataset will lead to significant changes in the classifier predictions (Kuncheva, 2014). The random forests (Breiman, 2001) learning algorithm has built-in bagging algorithms and often produces high performance, but as the size and dimension of the training data increases, powerful computing resources and long training time are required.

The bagging algorithm (Breiman, 1996) constructs multiple base classifiers by sampling the training examples at random with replacement from data space, known as bootstrapping. Those sub-classifiers uniformly vote the final prediction. Applying the original bagging algorithm to classifying an imbalanced dataset is not a wise idea. Every training subset bootstrapped from the original data still has an imbalanced distribution. It could be biased even more towards the majority class (Zhou & Lai, 2009). Either ignoring or overfitting the minority class examples is very likely to happen to each base classifier (He & Garcia, 2009). Consequently, the final performance will favour the majority class. Specific techniques are necessary to compensate for this situation. A straight forward way is to correct this skewness within each subset by resampling, and this can build component classifiers form data with equal class distributions (Wang, 2011). A common approach to balance the skewness in the class distribution is to combine sampling methods with ensembles. Bagging methods that are combined with resampling methods can be categorised into three broad groups; UnderBagging, OverBagging and UnderOverBagging.

The underbagging method is a combination of under-sampling and bagging which was first introduced by (Barandela, et al., 2003). The algorithm of UnderBagging is almost similar to the bagging ensemble algorithm that builds several models from the bootstrapped dataset and then aggregates the predictions from all the models. Each dataset contains all minority classes, and the majority class is taken at random with or without replacement (Hakim, et al., 2017).

Underbagging has been proven to improve the prediction accuracy of the minority class (Hakim, et al., 2017). One variation of underbagging method is Bagging Ensemble Variation (BEV) introduced by (Li, 2007). This method functions by maximising the use of minority class data without creating synthetic observations. Example if in a case there is an imbalanced class with a ratio of 20:80, the majority class is divided into four sections, and each section is inserted into a minority class so that the ratio of the majority and minority class is equal. The downside of this method is that it is weak with a high-class ratio (Li, 2007), but has been proven to improve prediction accuracy for low imbalance datasets (Hakim, et al., 2017). It has been argued that these Bagging variations inherit the disadvantage of sacrificing too much performance on the majority class from under-sampling. Besides, since the same minority class examples are taken as input by every classifier, the resulting ensemble still has a high possibility of suffering from overfitting (Liu, et al., 2009).

The OverBagging method is a combination method between Over-sampling technique and bagging; this method was first introduced by (Wang & Yao, 2009). Unlike UnderBagging, OverBagging adds minority class using the bootstrap process. SMOTEBagging is a variation of the OverBagging method which combines algorithms from SMOTE and bagging that involves synthesis generation data (Wang & Yao, 2009). SMOTEBagging is weak with high-class ratio (Krawczyk, et al., 2014) and with Outliers (BÃlaszczy´nski, et al., 2013), but this method has been proven to improve the prediction accuracy of the minority class (Hakim, et al., 2017). SMOTEbagging works by balancing each bootstrapped dataset by SMOTE before modelling. When using SMOTE, two parameters have to be pre-decided: k-nearest neighbours and the total number of over-sampling from minority class (Hanifah, et al., 2015). (Hanifah, et al., 2015) carried out a study using SMOTE bagging algorithm for an imbalanced dataset with a logistic regression base classifier and concluded that SMOTEbagging could increase the accuracy of the minority class compared to a logistic regression model.

UnderOverBagging is a combination of algorithms from under-sampling, over-sampling and bagging, but the data generation process is not like the underbagging or overbagging algorithm. The data generation process is similar to SMOTEBagging (Wang & Yao, 2009) but this method uses two sampling techniques over-sampling and under-sampling technique, the resampling rate is set in each iteration (Hakim, et al., 2017). This method has been reported to be weak with outliers (Duque, et al., 2016) and works well for a small number of trees (Galar, et al., 2011).

2.2.2.2 Boosting method

The boosting algorithms have been ranked among the top ensemble methods, due to their accuracy, robustness and broad applicability (Kuncheva, 2014). AdaBoost (Freund & Schapire, 1996) is one of the top boosting algorithms. There are two implementations of the AdaBoost algorithm, namely with resampling and with reweighting. In the AdaBoost algorithm with reweighting, no sampling is required. The base classifiers directly use the probabilities on the training dataset as weights (Kuncheva, 2014).

AdaBoost is observed to be capable of bias reduction. The AdaBoost algorithm weighs each sample reflecting its importance and places the most weights on those examples which are most often misclassified by the preceding classifiers. Such a focus may cause the learner to produce an ensemble unction that differs significantly from the single learning algorithm. With an imbalanced dataset, AdaBoost attempts to reduce the bias error as it focuses on misclassified examples. The sample weighting strategy of AdaBoost is equivalent to resampling the data space combining both over-sampling and under-sampling. It hence belongs to data-level solutions, which are applicable to most classification systems without changing their learning methods. It is observed to be capable of both bias and variance reduction but does not deal well with noise (Galar, et al., 2011). The emphasis on misclassified examples makes AdaBoost an accuracy-oriented algorithm. In the presence of imbalanced data, its learning strategy tends to bias towards the majority class as rare examples contribute less to the overall classification accuracy (Chawla, et al., 2004). Therefore, the original AdaBoost is not good at recognising rare cases. Besides, it may suffer from the overfitting problem, for the reason that rare examples tend to be weighted more than common ones, and thus replicated and boosted more often (Polikar, 2007). Nevertheless, AdaBoost is still popular due to several reasons including its flexibility, efficiency, little information loss compared to sampling methods, less overfitting risk and ability to reduce bias error (Galar, et al., 2011)

These attractive properties encourage AdaBoost to have evolved in two ways – the integration with resampling techniques and cost-sensitive Boosting (He & Garcia, 2009). When it is integrated with resampling techniques, data generation methods are often used to broaden the decision region for the minority class. For instance, SMOTEBoost (Chawla, et al., 2004) introduces SMOTE (Chawla, et al., 2002) in each round of Boosting, to enhance the probability of selecting the difficult rare cases that are dominated by the majority class examples. It utilises SMOTE for improving the prediction of the minority class and utilises modified boosting to not sacrifice accuracy over the entire dataset (Hu, et al., 2009). Meanwhile, Boosting prevents from sacrificing accuracy over the entire dataset. It is believed to produce higher diversity within the ensemble. It shows improved classification performance compared to conventional Boosting, but the parameter setting in SMOTE is crucial, which could cause over-generalisation (Wang, 2011).

MSMOTE is a modified version of SMOTE and was introduced by (Hu, et al., 2009). The combination of MSMOTE and AdaBoost was tested on several highly and moderately imbalanced datasets, and it was concluded that the prediction performance of MSMOTEBoost is better than SMOTEboost in on the minority class and F-Values was also improved (Hu, et al., 2009). Another method that is similar to SMOTEboost is RUSBoost (Seiffert, et al., 2010). Unlike SMOTEboost, this method removes instances from the majority class by random under-sampling the data-set in each iteration. The main advantages of this method are simplicity, speed and performance (Seiffert, et al., 2010). The main drawback of the random under-sampling method can be overcome by combining it with boosting, while specific information may be absent during a given iteration of boosting, it will likely be included when training models during other iterations. (Seiffert, et al., 2010) stated that RUSBoost algorithm performs favourably when compared to SMOTEboost, often resulting in significantly better classification performance despite its simplicity.

2.3 Comparison of Ensemble method

(Wan & Yang, 2013) explored popular ensemble methods for classification and regression problems and stated that the bagging method does not improve the predictive power of the model, but it reduces variance which in turn would narrowly tune the prediction to expected outcome (Wan & Yang, 2013). In another research by (Opitz & Maclin, 2006) it was found that bagging is almost always more accurate than a single classifier but is sometimes much less accurate than boosting. Bagging works best with high variance models. High variance models refer to models which produce differing generalisation behaviour with small changes to the training data. Examples of high variance models are decision trees and neural networks.

Bagging (Breiman, 1996) therefore tends not to work well with elementary models (Wang, 2011).

(Wan & Yang, 2013) used the Boosting (Freund & Schapire, 1996) approach and declared that by varying the weighting formula, a model with a high predictive force could be obtained but (Opitz & Maclin, 2006) claimed that boosting could create ensembles that are less accurate than a single classifier, especially when using neural networks. This holds for noisy datasets, as boosting ensembles may over fit and decrease the performance (Krieger, et al., 2002). (Wan & Yang, 2013) also carried out a comparison between random forest with boosting on a balanced dataset and concluded that random forest is more robust and faster to train. In another research carried out by (Pandey & Taruna, 2014) it was concluded that the random forest algorithm has the lowest prediction performance for an imbalanced dataset, compared to other methods such as boosting and bagging. (Opitz & Maclin, 2006) carried out a study on churn classification and based on the finding of this study, random forest and Boosting models gave the best accuracy.

Boosting combines weak learners to obtain a strong learner. A particular limitation of boosting is that it applies only to binary classification (Wan & Yang, 2013). (Pramanik, et al., 2010) investigated different ensemble models using a C4.5 algorithm(decision tree) on biomedical data to detect heart disease and concluded that boosting is more effective than bagging. (Joshi & Srivastava, 2014) investigated the effect of bagging on classification accuracy by using different decision trees as the base classifiers on seven different datasets and concluded that the classification accuracy increases when ensemble learning is used. (Opitz & Maclin, 2006) stated that the majority of the performance gains in an ensemble comes in the first few classifiers combined, but relatively large gains can be expected up to 25 classifiers when boosting decision trees.

(Galar, et al., 2011) carried out different ensemble methods that deal with the class imbalance on a C4.5 decision tree as the base model. The methods with the best performance are SMOTEBagging, RUSBoost and UnderBagging. Among these models, SMOTEBagging obtained slightly better performance. (Chujai & Kerdprasop, 2015) presented different method for dealing with imbalanced data classification problem by applying the decision tree ensemble learning using both bagging and boosting techniques. The finding was that RUSBoost is the best model that can classify an imbalanced data which have a high imbalance ratio. (Chawla, et al., 2004) explored the use of SMOTEBoost algorithm on imbalanced datasets and concluded that this method could handle imbalanced dataset well compared to methods such as AdaCost and AdaBoost techniques. (Wang, et al., 2015) carried out a Lasso-Logistic Regression bagging ensemble method for prediction of credit scoring on an unbalanced dataset and concluded that this method outperforms popular credit scoring models such as decision tree and random forest.

2.4 Performance evaluation for imbalanced data models

When measuring the overall accuracy of an ensemble model, taking into consideration only the accuracy of a model is meaningless and insufficient when dealing with the imbalanced dataset (Joshi, 2002) as it is highly sensitive to changes in data. For performance measure of a single class, some single-class measurements were defined by (Riksbergen, 1979), including F-measure, Recall(sensitivity) and Precision.

Recall (R), precision (P) and F-measure (F) come from information retrieval area (Riksbergen, 1979). For the positive class, they are defined as,

$$R = \frac{TP}{TP + FN}, \tag{2.1}$$

$$P = \frac{TP}{TP + FP}, \tag{2.2}$$

$$F = \frac{(1 + \rho^2) \times R \times P}{\rho^2 \times R + P}, \tag{2.3}$$

where ρ controls the relative importance of precision and recall. It is usually set to one (Wang, 2011). The recall is a measure of completeness; the proportion of positive class examples that are classified correctly to all positive class examples (Wang, 2011). Precision is a measure of exactness, the proportion of positive class examples that are classified correctly to the examples predicted as positive by the classifier (He & Garcia, 2009)). F-measure incorporates both recall and precision to express the trade-off between them. It was shown to be a more favourable measure (Powers, 2011) and has been used as a criterion for classifier selection by (Chawla & Sylvester, 2007). Another standard evaluation method is specificity. This method is a measure of accuracy of the majority class.

Receiver Operating Characteristic (Fawcett, 2003) and the associated use of the area under the ROC curve (AUROC) (Ling, et al., 2003) are the conventional technique to evaluate overall classification performance in this area. They measure the separating ability of a classifier between two classes. ROC curve depicts all possible trade-offs between TP rate and FP rate.

Closely related to ROC, AUC represents a ROC curve as a single scalar value by estimating the area under the curve, varying from zero to one. AUROC is proved to be independent of the selected decision threshold and class distributions (Fawcett, 2003). It should be noted that the ROC analysis and AUROC estimation are only able to describe the discriminability of a pair of classes (Wang, 2011).

2.5 Conclusion

This chapter focused on existing studies related to imbalanced learning. Current research on imbalance learning problems, including data-level and algorithm-level ensemble-based methods, were explored. The data level approach would be a suitable approach in many real-world problems due to its flexibility and effectiveness. Algorithm level solutions require different treatments for specific kinds of learning algorithms, and it restricts their use in many applications since it is not known which algorithm would be the best choice beforehand in most cases.

It is well established that the ensemble models generally yield better results compares to a single model. Bagging and Boosting decrease the variance of a single model as they combine several estimates from different models and hence the result may be a model with higher stability. When the challenge faced by a single model is over-fitting, then the best option will be bagging. Bagging gets around the overfitting problem by creating its own variance amongst the data. Boosting does not help to avoid over-fitting; in fact, this technique is faced with this problem itself. For this reason, Bagging is useful more often than Boosting.

Various resampling methods have been used together with ensemble methods but, it is unclear which resampling method performs better and which sampling rate should be used. Some studies found that resampling to full balance is not necessarily optimal, and the best resampling rate varies with problem domains and resampling techniques (EstaBrooks, et al., 2004). Their performance also depends on different classifiers and evaluated measures (Hulse, et al., 2007).

Past studies have not reached any conclusive results with regards to whether under-sampling or over-sampling is best with regards to classification performance. Most likely, conflicting results are due to the combination of different datasets and classification algorithms. Also, the choice of resampling method is probably both domain and problem specific.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explains the methods that are carried out in chapter 5 of this paper. This chapter begins by discussing the data preparation techniques and how to carry out this process effectively. The next part of this chapter explains the steps involved in bagging. Finally, a proposed method will be introduced. The proposed method is similar to bagging, but the bootstrap process in bagging is eliminated. The data will be sampled by subsetting different parts of the data manually which might allow the models to learn better. The final prediction for this proposed method will be carried out by averaging the probabilities. The process flow diagram for the entire process that will be carried out in chapter 5 is shown in the flow diagram Figure 3.1.

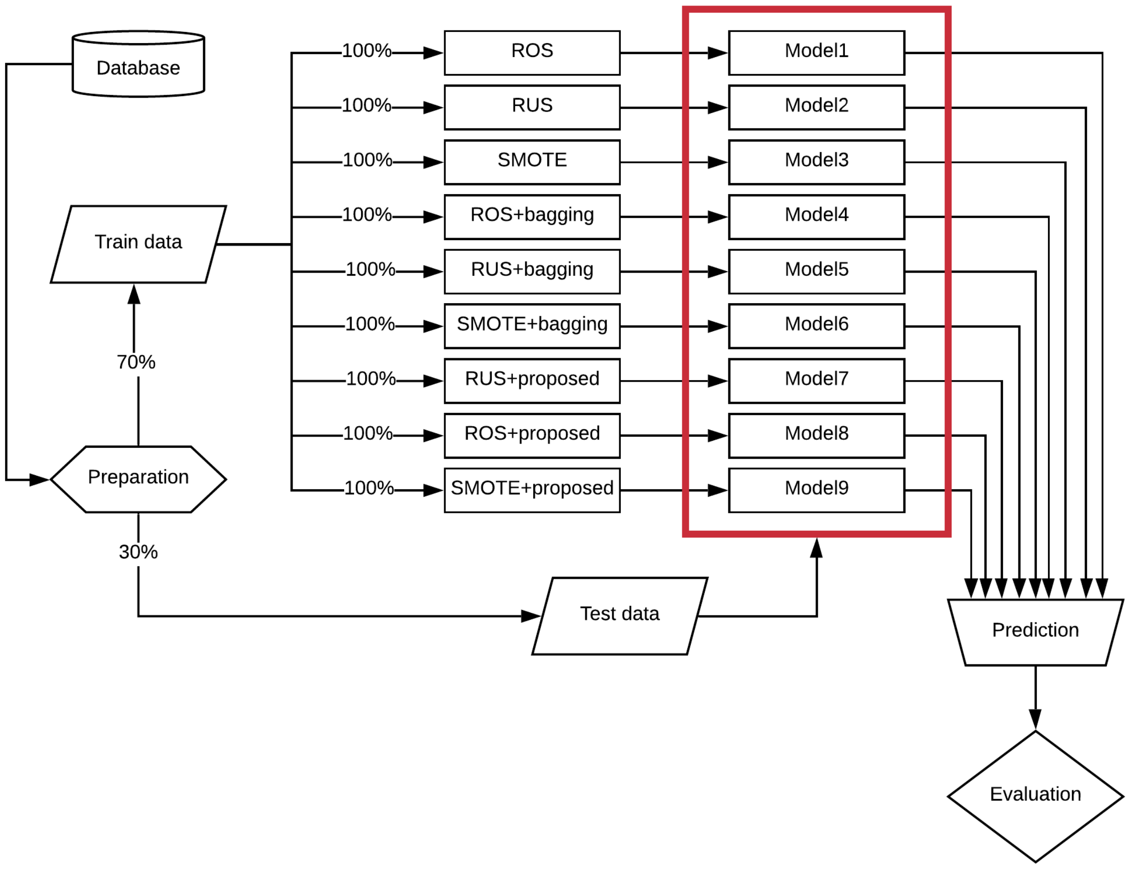


Figure 3. 1 Process flow diagram for a single base classifier

3.2 Data preparation

Data preparation is the process of collecting, cleaning and consolidating data into one file or table, primarily for use in the analysis. Data is often created with missing values, inaccuracies or other errors. When using a logistic regression model, it is critical to deal with missing values as this model eliminates an entire row even if only one variable has a missing value. Data preparation has a significant effect on the predictive ability of a model. This section explains the steps performed to pre-process the dataset before the modelling procedure.

3.2.1 Exploratory Data Analysis (EDA)

EDA is typically applied before any formal modelling commences. Several statistical summaries and visualisations can be used to understand the data at hand and eliminate or sharpening potential hypotheses. Some of the standard statistical summaries and plots include histograms and predictor’s statistics. The histogram can be used for class distribution while predictor’s statistics can be used for missing values, minimum value, the maximum value and standard deviation. The box-and-whisker plot is typically used to represent most predictor's statistics graphically. EDA also identifies what data preparation steps are needed, example data pre-processing to resolve the skewness of predictors. The EDA methods that should be used varies with the variable type and the data type. EDA serves as an essential guide to carry out data processing.

3.2.2 Data Processing

In many real-world datasets, some of the predictors have no values for a given sample. It is vital to understand why these values are missing and identify if the pattern of missing data is correlated with the outcome, before any strategy to handle the missing value is applied. A good understanding of the dataset is required before dealing with missing values. For example, a missing value could merely mean that field is not applicable (NA) for that particular case, trying to replace this missing value with the mean value would directly cause incorrect predictions.

Data processing is the most critical art in machine learning which creates a vast difference between a good model and a terrible model. Data processing is about creating new input variables from existing ones. Data processing also involves encoding of predictors and is considered an essential step in the data preparation phase as it often has a significant impact on the performance of models. Some of the data processing tasks, such as adding/removing predictors, are informed by the modeller's knowledge of the problem domain at hand. Others, such as binning predictors and converting from categorical to dummy predictors, are standard procedures.

Combining rare classes in categorical variables is an essential step in data processing. Sparse classes are those that have very few total observations. They can be problematic for specific machine learning algorithms, causing models to overfit. In this paper, classes with a frequency of less than 5% will be combined. There is no specific rule on how many observations should be in each class. Sometimes, removing the unwanted variable is also data processing as the variable that is not related degrades the performance of the model.

The steps for data processing are as follows:

- Brainstorm features
- Create features
- Check how the features work with the model
- Repeat the process until the features work perfectly

3.2.4 Data Partition

The proper allocation of data to different tasks is one of the essential aspects of predictive modelling. A crucial element of data partition is maintaining a similar population across all samples. In a classification problem, a similar class distribution must be preserved across all datasets by using stratified sampling. The number of observations/samples allocated for each modelling task depends on the size of the available data. If the data is small, data partition can have a critical impact on the quality of the final models. For example, if the training dataset is small, the learning algorithms, used to build models, may not be able to capture the underlying patterns. On the other hand, a small testing dataset would have a limited capacity to evaluate the performance of models. The conventional approach undertaken in many pieces of literature is to carry out a split of 70% for the training data and 30% for the testing data. This paper will adopt the same proportion of split. In most cases, the training and test datasets are expected to be as homogeneous as possible. This thesis will adopt a simple splitting based on the outcome variable. The ‘createDataPartition’ function from the ‘Caret’ package is used to carry out the data partition procedure. This function is used to create balanced splits of the data. If the ‘y’ argument to this function is a factor, the random sampling occurs within each class and will preserve the overall class distribution.

3.3 Bagging

Bagging meaning Bootstrap Aggregating. Bootstrap is a process of selecting samples from the original sample and using these samples to create multiple models. Bootstrapping is a process of creating random samples with replacement for estimating sample statistics. One of the ways to select samples or bootstrap samples is to select n items with replacement from an original sample. A bootstrap sample may have a few duplicate observations or records, as the sampling is done with replacement. In this paper, these bootstrapped samples will undergo a data balancing method, and each of these balanced datasets is used to create models. These models are then used to carry out predictions using a majority vote method.

When bagging a logistic regression model one separately fits logistic GLMS on bootstrapped datasets which are resampled to deal with class imbalance problem. The testing data will then be used on each of these models to carry out predictions. The final prediction is then carried out using a majority vote from all the models.

When bagging decision trees, the variance is improved by majority voting of outcome from multiple fully grown trees on variants of the training set. It uses bootstrap with replacement to generate multiple training sets, and these training sets will undergo a data balancing method. For decision trees, bagging has a single parameter, which is the number of trees. All trees are fully grown binary tree (unpruned), and at each node in the tree, one searches all features to find the feature that best splits the data at that node.

The process flow diagram of a bagging method is shown in Figure 3.2.

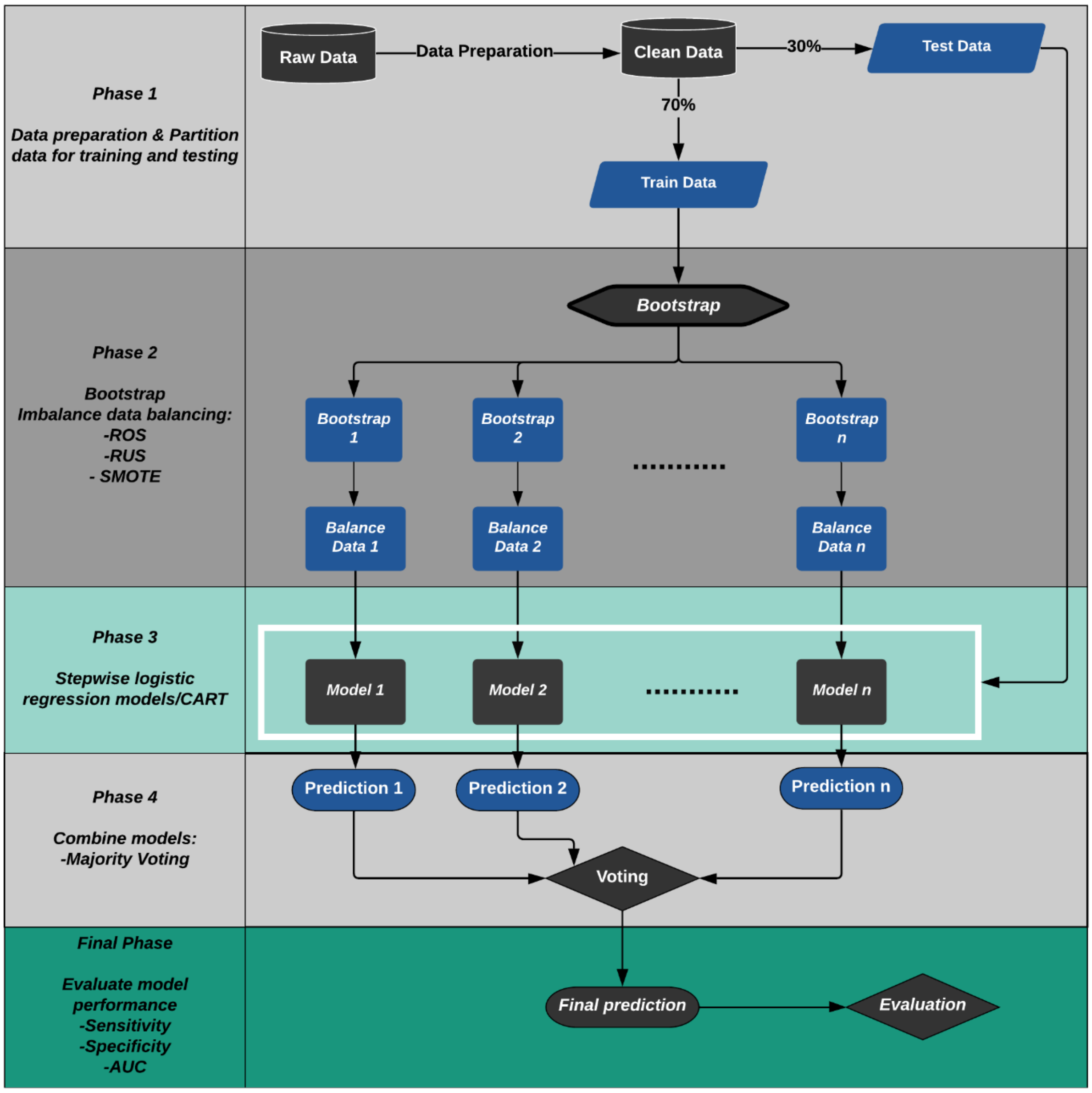


Figure 3. 2 Process flow diagram of bagging

In this paper, the bagging method will be explored using two different classifiers and with three different sampling methods.

3.4 Proposed method

Bagging framework entails three critical phases; bootstrap, modelling and combining. In the proposed method, modification will be done in phase 2 and phase 4. The bootstrap process in bagging will be eliminated; instead, data will be subset using a manual process. Furthermore, instead of using majority voting this method will use averaging to determine the outcome. It would not make much sense to average the outcome of each model; instead, the probability of each of these outcomes will be averaged. The process flow diagram of the modified method is shown in Figure 3.3.

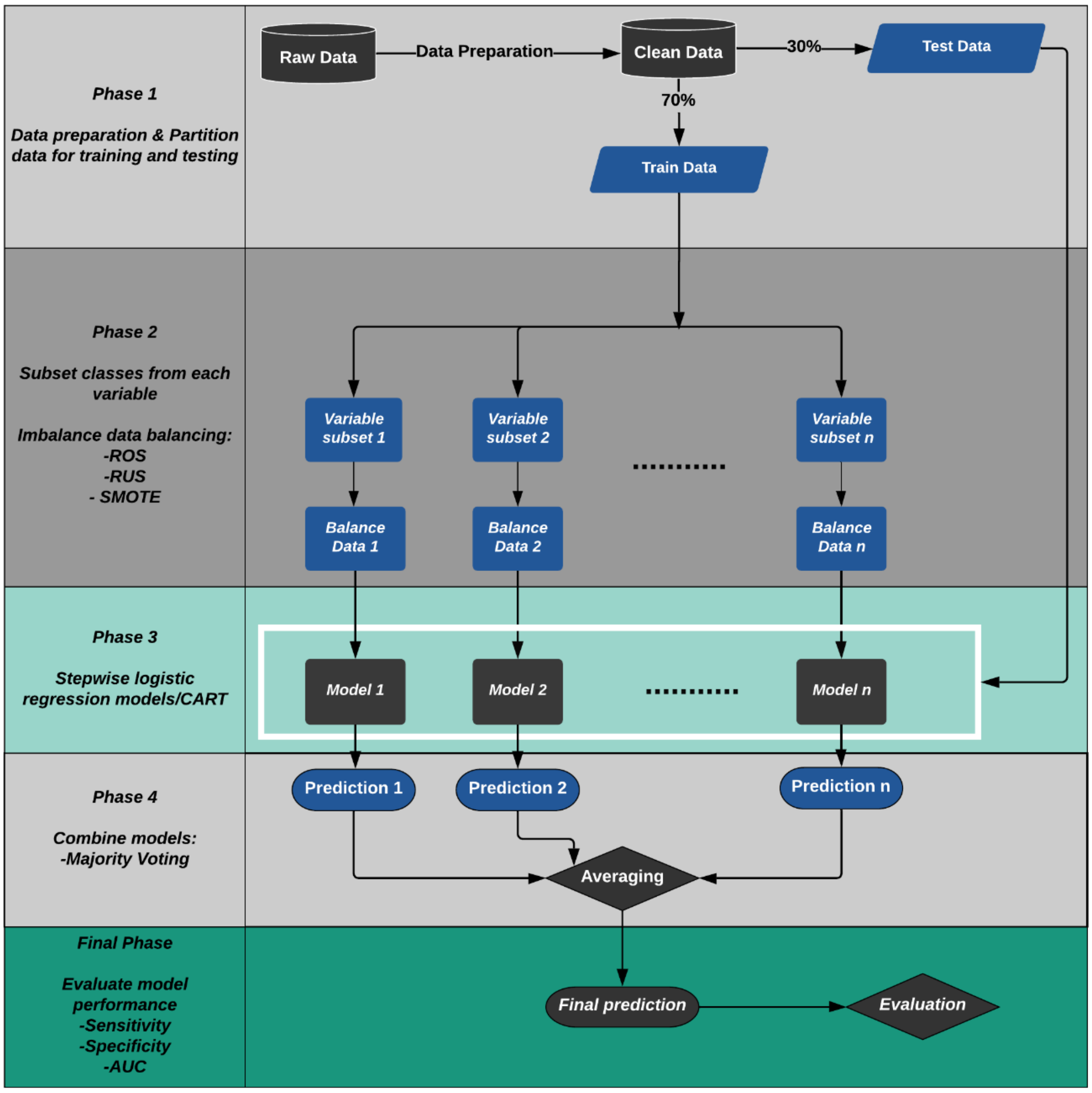


Figure 3. 3 Process flow diagram of the proposed method

3.4.1 Data subsetting and Balancing

In this proposed method, data will be subset based on each level of a categorical variable. To deal with the imbalanced problem, each of the subsets will be combined with sampling methods that deal with class imbalance problems before entering the modelling phase.

The proposed method of data level subset will be explained with the aid of an example. Consider a dataset (Table 3.1) with four explanatory variables X1, X2, X3 and X4.

X1	X2	X3	X4	Target variable
A	R	K	H	Yes
A	T	K	H	Yes
A	T	K	F	Yes
B	R	K	F	Yes
B	R	L	F	No
A	T	L	F	No
A	T	L	F	Yes
B	T	K	H	No
B	R	L	F	No
B	R	L	F	No

Table 3. 1 Sample Dataset

In this proposed method, data subsets will be created from all levels of each categorical explanatory variable. Example variable X1 has two levels; A and B. Data will be subset for each of these levels; separate datasets which contain only subset of data with level A and another dataset which contains only subset of data with level B will be formed. This process will be repeated for all the other categorical variables. In the dataset shown in Table 3.1, 8 data subsets will be created (4 categorical variables with 2 levels each). Based on each of the subsets from different variables, multiple models will be created. Data balancing methods will be applied to each of these subsets before building a model. Predictions will be carried out on each of these models, and the output (probability) from each of these models will be averaged to obtain the final prediction.

The basic idea behind this method is that if a model learns from a specific subset of the data, it could be more reliable in predicting when that specific case occurs. For example, when a model is created with a subset of A, this model might be able to give a better prediction when A occurs compared to the other models. For example in Table 5.2;

Given an observation in the test dataset:

X1	X2	X3	X4	Target variable
A	R	L	F	Prediction

Table 3. 2 Sample test data (1 observation)

In the observation, the variable X1 contains A, so model 1 which leaned from a subset of dataset A might be able to determine better if the outcome is Yes or NO. The idea is that models learning from subset of A, R, L and F will be able to determine better the probability of this observation as the observation contains A, R, L and F. Example if the model containing the observation outputs a high probability while other models output a low probability as shown in Table 3.3.

Model	Probability
Model 1 with subset of A	0.8
Model 1 with subset of B	0.3
Model 1 with subset of R	0.6
Model 1 with subset of T	0.35
Model 1 with subset of L	0.7
Model 1 with subset of K	0.25
Model 1 with subset of F	0.65
Model 1 with subset of H	0.35

Table 3. 3 Sample test data (Prediction)

Combining the probability of the eight models and averaging yields a result of 0.5. Therefore, the basic concept behind this method is that, if we can guide some model to predict a case with high accuracy for specific observations while other models fail, the prediction could still be classified accurately. The idea is to allow models to focus better on specific cases to improve the predictive power of the model.

Decision tree models are sensitive to small changes in data, by applying the proposed method, the resulting decision trees can be entirely different, and in turn, the predictions can be quite different, hence combing the predictions from the highly diverse trees could result in a model with high AUROC. For a logistic regression which is less sensitive to changes in data, this method might not be useful.

3.4.2 Modelling

The purpose of the modelling phase is to create models from the data subsets. Models are created by training these data subsets with a machine learning algorithm that creates a homogeneous ensemble of classifiers. Multiple machine learning algorithm may be employed to create a heterogeneous ensemble, but this paper focuses on a homogeneous ensemble. Two different base classifiers are tested with this method; logistic regression (linear) and CART (non-linear).

It is highly probable that the fusion of several models would outperform a single model. However, the fusion of different models might only add to the complexity of the system without any performance improvement. Thus, in order for an ensemble to be successful, the errors made by its base models should not be highly correlated, and each base model has to have acceptable performance. Accordingly, model diversity is considered one of the key design feature within a successful ensemble. However, in this method, diversity if not of the primary concern, but if the majority of the models are correlated, then this method would not be effective. If a few models are highly correlated, it could be an added advantage when combining using the averaging method. For example, if two models predict an observation with 0.80 probability and the other models performed below 0.5, averaging these predictions could still lead to a probability of greater than 0.5 and could lead to a better prediction.

3.4.3 Model Fusion

The final phase in any ensemble method usually involves combining the predictions of all the individual models created in the modelling phase. A majority voting method is the most common method used for the classification problem in bagging. However, in this proposed method, averaging of probabilities will be used as it would be more effective in dealing with the diversity issue of models.

Since the predictions are either ‘1’ or ‘0’, averaging does not make much sense for binary outputs. Instead, the probability of each observation from each model will be added and then averaged by the total number of models. The final prediction will then be carried out by using these average predictions.

3.5 Model evaluation

Various evaluation metrics are available to determine which model would perform better for a task. The evaluation metrics that should be used depends entirely on the case at hand. For example, when building a classifier to predict the presence of disease, the aim is high sensitivity no matter the specificity. Identifying all subjects with the disease is more important than correctly predicting healthy as healthy. Even if the healthy is predicted as subjects with the disease, it would not have a significant adverse effect. Example if a subject is predicted as having a disease, he could go through some medical checks and be declared healthy. On the other hand, if a person with the disease is misclassified as healthy, he could potentially end up dead without receiving the proper treatment. In such a situation, the aim would be to maximise sensitivity. A suitable measure of model performance in such a situation would be using the AUROC. A high AUROC value indicates a model with high sensitivity as well as high specificity. Plotting the ROC curve is also necessary to determine how well the model performs in different threshold levels.

Another set of evaluation metrics are recall and precision. Recall is just another name for sensitivity. Precision is defined as the fraction of relevant true positive along all the instances predicted as positive. This measure should be used in a situation where predicting the negative class as positive is an issue. Using the example mentioned earlier, precision represents the percentage of people who were classified as having disease actually had a disease. It is ideal to have high precision, but not always necessary. A PR curve can represent the trade-off between precision and recall.

A confusion matrix can be used to measure the performance of a classification algorithm. Confusion matrix contains information about the actual instance and the predicted instances by the model. Based on the values obtained in the confusion matrix, evaluation metrics such as sensitivity, specificity and precision can be calculated. A standard confusion matrix table is shown in Table 3.1

	Predicted positive	Predicted positive
Actual positive	True Positive (TP)	False Negative (FN)
Actual negative	False Positive (FP)	True Negative (TN)

Table 3. 4 Standard Confusion Matrix

This paper uses sensitivity, specificity, ROC, PR and AUC of PR and ROC curves as model evaluation methods. Accuracy is not an ideal method to gauge model performance for imbalanced data. When dealing with imbalanced datasets, it may be desirable to select a model with a lower accuracy because it has a higher predictive power on the problem. For example, in a problem where there is a significant class imbalance, a model can predict the value of the majority class for all predictions and achieve a high classification accuracy. Usually, in the domain of imbalanced class distribution, the main concern will be to maximise sensitivity while maintaining a high level of specificity.

Sensitivity is defined as,

$$Sensitivity = \frac{TP}{TP + FN}, \tag{3.1}$$

Specificity is defined as,

$$Specificity = \frac{TN}{TN + FP}, \tag{3.2}$$

One crucial part of predictive modelling in classification is to determine the threshold level for the case at hand. There are many approaches to determining thresholds, which fall into two categories: subjective and objective. Sometimes, a specific level example 95% of sensitivity or specificity is desired or deemed acceptable and is predetermined. This approach is subjective because a specific level for some attribute is predetermined. This paper will adopt a subjective approach to setting the threshold level. Models will have different threshold levels to meet the specific, predetermined level. Example, the threshold level of each model, will be changed to achieve a sensitivity of 95% and models will then be compared based on that threshold. It will be a comparison of how well the model performs when 95% sensitivity is obtained.

3.6 Computing Environment

All the processes discussed in this chapter are carried out using the R language. R is a language used for statistical computation, data analysis and graphical representation of data. R was designed as a statistical platform for data cleaning, analysis and representation.

R provides ample tools for developers to train and evaluate an algorithm and predict future events. Thus, R makes machine learning a lot more natural and approachable. An extensive amount of machine learning packages are available in R. The package that is mainly used in this paper is ‘CARET’. This package is used for classification and regression training.

CHAPTER 4

ALGORITHM

4.1 Terminology and Notation

A set of variables which can be measured or selected are known as explanatory variables. These explanatory variables may be quantitative or qualitative and can influence one or more response variables. When training a statistical model, one aims at fitting it in such a way that it describes the relationship between the explanatory and response variables.

Qualitative variables are variables that are assumed to come from a finite set $\boldsymbol{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots\ldots, \boldsymbol{C_k}\}$ of distinct categories or groups, where C_i denotes category i. Quantitative variables are often referred to as categorical or discrete variables. If the qualitative variable consists of only two categories, these are often coded by the numbers 0 and 1, with 1 representing the minority class in a class imbalance problem.

Input variables that explain the output will be referred to as explanatory variables and the variable that is predicted as response variable. This thesis will use uppercase letters when referring to random variables. Observed values will be written in lowercase. A hat will represent fitted models. Example $\boldsymbol{\hat{y}} = \boldsymbol{\hat{f}}(\boldsymbol{X})$ is a model fitted to the set of input variable $\boldsymbol{X}$, where $\boldsymbol{\hat{y}_i} = \boldsymbol{\hat{f}}(\boldsymbol{x_i})$ is the prediction for individual **i**. **M** will be used to represent the number of models, **B** will represent the bootstrapped datasets.

4.2 Logistic Regression

A variant of a generalised linear model is the logistic regression model. Logistic regression is one of the most critical methods for analysing categorical data and plays a prominent role is CRM predictions. The binary variable can be coded as 0/1 response $\boldsymbol{Y_i}$, where $\boldsymbol{Y_i = 1}$ corresponds to the minority class(positive) and $\boldsymbol{Y_i = 0}$ corresponds to the majority class (negative).

Taking the probability for an observation to come from the positive class is $\boldsymbol{\pi_i(x_i)}$ and the probability for an observation to come from the negative class is $\boldsymbol{1-\pi_i(x_i)}$. A logistic regression model can then be expressed as,

$$\boldsymbol{Y_i} = \begin{cases} \boldsymbol{1}, & \boldsymbol{with\,probability\; P(Y_i = 1|X_i = x_i) = \pi_i(x_i)} \\ \boldsymbol{0}, & \boldsymbol{with\,probability\; P(Y_i = 1|X_i = x_i) = 1 - \pi_i(x_i)}, \end{cases} \tag{4.1}$$

A logistic regression model is represented by:

$$\boldsymbol{\pi_i = logit^{-1}(\beta^T x_i) = \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}}, \tag{4.2}$$

This formulation of the logistic regression thus refers directly to the probability of success. The logistic regression curve is S-shaped. The parameter $\boldsymbol{\beta_i}$ determines the rate of increase or decrease of the S-Shaped curve. The sign of $\boldsymbol{\beta_i}$ indicated whether the curve ascends or descends and if a variable has a positive effect or negative effect on the outcome variable.

4.3 Decision Tree (CART)

Classification and Regression Trees (CART) algorithm is a classification algorithm for building a decision tree based on Gini's impurity index as the splitting criterion. CART is a binary tree build by splitting nodes into two child nodes repeatedly. The algorithm works repeatedly in three steps:

- Find each feature's best split. For each feature with K different values, there exist K-1 possible splits. Find the split, which maximises the splitting criterion. The resulting set of splits contains best splits (one for each feature)

- Find the node's best split. Among the best splits from the previous step find the one, which maximises the splitting criterion

- Split the node using the best node split from the 2nd step and repeat step 1 until a stopping criterion is satisfied.

The Gini impurity measure at a node t is defined as,

$$i(t) = \sum C(i|j)p(i|t)p(j|t), \tag{4.3}$$

Where C(i|j) is the cost of misclassifying a class j case as class i case. P(i|t) is the probability of case in class i given that falls into node t.

The Gini splitting criterion is the decrease of impurity defined as,

$$\Delta i(s,t) = i(t) - pLi(tL) - pRi(tR), \tag{4.4}$$

where *pL* and *pR* are probabilities of sending a case to the left child node *tL* and to the right child node *tR* respectively. They are estimated as *pL*=*p*(*tL*)/*p*(*t*) and *pR*=*p*(*tR*)/*p*(*t*).

4.4 Ensemble Method

Bootstrap AGGregation or Bagging was proposed by (Breiman, 1996) to improve the classification by combining classifications of randomly generated training sets. This method uses bootstrap to create multiple versions of a training set with replacement. Each of these datasets is used to train a different model, increasing diversity among individuals. The outputs of the individual models are combined by voting the individual outputs or averaging the probability of the output to create a final ensemble output. Given a standard training set **D**, the bagging technique generates L training sets; $\boldsymbol{D_i = 1, 2, \ldots, L}$, by sampling examples from D uniformly and with replacement. In so doing, it is likely that some samples will be repeated for any training set $\boldsymbol{D_i}$. If $|\boldsymbol{D}| = |\boldsymbol{D_i}|$ and if the training set is large enough, the $\boldsymbol{D_i}$ is expected to have about 63.2% of the unique examples of $\boldsymbol{D}$, the rest being repeated. Each sampled dataset $\boldsymbol{D_i}$ is then used to train a base classifier. The output of the final ensemble is calculated by combining the individual's outputs (model fusion).

4.4.1 Bootstrap

The main idea behind bagging is to reduce the variance of a prediction by considering predictions made by different models using techniques such as majority vote and averaging.

Bagging requires multiple datasets. Bagging utilises a bootstrap method to split one dataset into multiple datasets by using a random sampling approach.

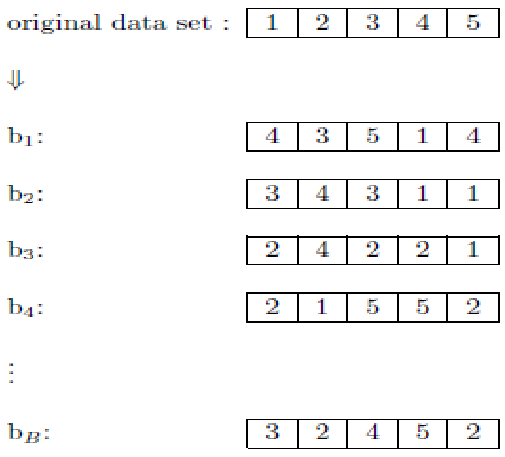


Figure 4. 1 The Bootstrap. The original sample is shown at the top. To create a bootstrap sample one draws from the original sample with replacement (Debik, 2017).

This method is commonly used to quantify the uncertainty associated with a given estimator or a statistical model. By drawing a large number of samples with replacement from a single dataset, datasets which differ from one another can be obtained. This is called resampling with replacement as shown in Figure 4.1. The samples are denoted by $\boldsymbol{b_1, b_2, \dots \dots, b_B}$ and are called the bootstrap datasets. $\boldsymbol{B}$ is the total number of resampled dataset. The probability of drawing an observation x_i from the sample is $\boldsymbol{Pr(X = x_i) = \frac{1}{n}, i = 1, \dots, n}$. Hence the probability that observation $\boldsymbol{x_i}$ has not been drawn is $\boldsymbol{1 - \frac{1}{n}}$.

When the bootstrap algorithm is applied, n observations from the sample will be selected with replacement. The probability for choosing a specific observation is always the same for all observations.

4.4.2 Bagging

In a situation with n observations of random variable $\boldsymbol{X}$ each with variance $\boldsymbol{\sigma^2}$. These random variables are thus independently and identically distributed. The mean can be calculated by,

$$\boldsymbol{\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i} \tag{4.5}$$

and the variance of the mean is defined as,

$$\boldsymbol{Var(\bar{X}) = Var\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) = \frac{\sigma^2}{n}} \tag{4.6}$$

Thus by averaging, the variance can be reduced. In the bagging process, several training sets $\boldsymbol{B}$ are obtained by bootstrapping, and separate models are built for each of these training sets. In the case of binary classification, the most standard process to obtain the final prediction is to carry out a majority vote.

Majority vote: Let each of the GLMs make a prediction $\hat{\boldsymbol{f}}_{\boldsymbol{i}}$ for $\boldsymbol{y_i} = \{\boldsymbol{0}, \boldsymbol{1}\}$, given the input values $\boldsymbol{x_i}$. This gives a total of $\boldsymbol{B}$ predictions. The predictions for new observations $\boldsymbol{x}$ are obtained by taking a majority vote over all these $\boldsymbol{B}$ predictions and is defined by,

$$\hat{\boldsymbol{f}}^{\boldsymbol{B}}_{\boldsymbol{bag}}(\boldsymbol{x}) = \boldsymbol{majority\ vote}\ \{\hat{\boldsymbol{f}}^{\boldsymbol{b}}(\boldsymbol{x})\}_{\boldsymbol{1}}^{\boldsymbol{B}} \tag{4.7}$$

In the event of a tie, the typical conversion is that these are split at random. In a binary classification problem, a tie can be avoided by selecting an odd number of bootstrap samples.

The bagging algorithm is described as,

```
    • Input training set D
    • Number of bootstrap samples: B
for b= 1 to B
    • Build a  dataset $D^{*b}$ , by sampling N items, randomly with replacement from D
    • Train a model, m using $D^{*b}$and add it to the ensemble
    • end for
for m = 1 to M
    • input test data
    • predict outcome
Combine model outputs by majority vote
End
```

4.5 Data balancing methods

Traditional ensemble bagging method does not deal with the problem of imbalanced datasets. Several methods of ensemble-based methods for dealing with imbalanced data were discussed in the literature. In this paper, sampling methods will be applied to each bootstrapped samples before the modelling process. Three different sampling methods will be compared for compatibility on two base classifiers. These methods are RUS, SMOTE and ROS. The following sections explore the algorithms for these methods. The full algorithm for only SMOTE and bagging is described as the overall process is similar for the three methods described below, except that the sampling method is different.

4.5.1 Random Over-sampling

This method works with minority class. It replicates the observations from minority class to balance the data. Random sampling uses simple randomness to create desired distributions between the two classes. Random over-sampling will randomly replicate the instances from minority class and add them back to the training set. A replication percentage of $\boldsymbol{\delta}$ can be used to control the amount of duplicated instances. Example, $\boldsymbol{\delta = 100\%}$ means additional $\boldsymbol{N^+}$ number of replications will be performed resulting in $\boldsymbol{2 \cdot N^+}$ instances for the minority class. Therefore, the overall imbalance ratio $\boldsymbol{\beta = N^- : (1 + \delta) \cdot N^+}$ can be adjusted by $\boldsymbol{\delta}$ accordingly to become a balanced value.

An advantage of using this method is that it leads to no information loss. The disadvantage of using this method is that since over-sampling simply adds replicated observations in the original dataset. It ends up adding multiple observations of several types which could lead to overfitting.

The ROS algorithm can be described as,

- ***Identify the minority $\boldsymbol{C_1}$ and majority $\boldsymbol{C_2}$ class***
- ***Calculate the number of instances to be added based on the percentage $\boldsymbol{\delta}$ of over-sampling***
- ***Identify a random instance from the minority class $\boldsymbol{C_1}$, replicate it and add to new resampled dataset***

- ***Repeat the previous step till the number of instances replicated as per the given percentage $\boldsymbol{\delta}$***

4.5.2 Random Under-sampling

This method works with the majority class. It reduces the number of observations from the majority class to make the dataset balanced. This method is best to use when the dataset is huge and reducing the number of training samples helps to improve run time and storage troubles. Random under-sampling randomly selects a subset of instances from majority class and then combines them with the minority class to form a new training dataset $\boldsymbol{D'}$, which is then applied with standard classification algorithms. Assume only $\boldsymbol{\gamma}$ percent of majority instances are selected, the new imbalance ratio is $\boldsymbol{\beta = \gamma \cdot N^{-} : N^{+}}$.

Theoretically, one of the problems with this method is that one cannot control what information about the majority class is thrown away. In particular, crucial information about the decision boundary between the minority and majority class may be eliminated. Despite its simplicity, random under-sampling has empirically been shown to be one of the effective methods to deal with imbalanced data.

The RUS algorithm can be described as,

- ***Identify the minority $\boldsymbol{C_1}$ and majority $\boldsymbol{C_2}$ class***
- ***Calculate the number of instances to be removed based on the percentage $\boldsymbol{\gamma}$ of under-sampling***
- ***Identify a random instance from majority class and remove it from the dataset $\boldsymbol{b_i}$***
- ***Repeat steps until the number of instances removed as per the given percentage $\boldsymbol{\gamma}$***

4.5.3 SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) overcomes imbalance by generating artificial data. It is also a type of over-sampling technique. SMOTE algorithm creates artificial data based on feature space similarities from minority samples. Some advantages of using SMOTE is that it alleviates overfitting problems and causes the decision boundaries for the minority class to spread. The downside of this method is that it can make the process of learning fine features difficult.

Two parameters need to be decided in SMOTE: *k*-nearest neighbours and the total number of over-sampling from minority class. In this paper, the k value is set to 2 for all experiments carried out using SMOTE in chapter 5.

The SMOTE algorithm can be described as,

```
Given: the SMOTE rate H; number of nearest neighbour k
• Let : $S_{min} = Z_{min}$

• For each minority example $x_i (i = 1 \dots\dots, |Z_{min}|)$:

    - Compute the k-NN set $\{\hat{x}_i\}$ for $x_i$ in $Z_{min}$
    - For j = 1 to H:
        o Choose a random neighbour $\hat{x}_i \epsilon \{\hat{x}_i\}$
        o Compute difference vector $dif = \hat{x}_i - x_i$
        o Multiply the value on each dimension of $dif$ by a random number $\varphi \epsilon [0, 1]$
        o $x_{new} = x_i + dif$
        o Add $x_{new}$ to $S_{min}$
• Return: $S_{min}$
```

4.6 Bagging (imbalanced data)

When bagging a model, one separately fits ***B*** models on bootstrapped datasets. The final prediction is carried out using a majority vote.

The Algorithm for bagging with sampling method is defined as,

```
Input: Training data D
        Number of bootstrap samples: B
for b= 1 to B: repeat
```

```
        Draw a bootstrap sample $D^{*b}$ from the training data
        Apply the sampling method to each bootstrapped data
        Fit a model to the balanced data $\hat{f}^b$
        Predict the probability of success $\hat{\pi}^b(x)$ for the observations in the
Input: Testing data
    • Generate outputs from each model
Output: Final prediction~ Majority vote

        $\hat{f}^B_{bag}(x) = majority\ vote\ \{\hat{f}^b(x)\}^B_1$
end
```

4.7 Proposed method (imbalanced data)

In the proposed method, data is subset based on each level of all categorical variables. Each of these subsets will undergo a data balancing method (RUS/ROS/SMOTE). These balanced datasets will then be used to create models. These models will then undergo a model fusion process where the testing data is used to carry out predictions from all the models and averaged to obtain the final prediction.

The Algorithm for the proposed method is defined as,

```
Input: Training data D
        Let M a given set of variables
        Select N categorical variables
For s= 1 to N
        Subset levels
For c= 1 to s: repeat
        Apply the sampling method to subset
        Fit a model to the subset data
Input Test data
For p=1 to c
    • Predict the probability of success $\hat{\pi}^b(x)$ for the observations
    • Generate outputs from each model
```

Output: Final prediction: Average the predicted probabilities:

$\bar{\hat{\boldsymbol{\pi}}}_{bag}^{B}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{\boldsymbol{\pi}}^{b}(\boldsymbol{x})$***. Classify*** $\hat{\boldsymbol{f}}_{bag}^{B}(\boldsymbol{x})$ ***according to a threshold value.***

End

Averaging probabilities: Let $\hat{\boldsymbol{\pi}}_i$ denote the estimated probability for $\boldsymbol{y_i = 1}$, given the input values $\boldsymbol{x_i}$. This gives a total of $\boldsymbol{B}$ probabilities. The prediction for new observations $\boldsymbol{x}$ are obtained by taking an average over all of these $\boldsymbol{B}$ probabilities, and then classify according to a threshold value $\boldsymbol{c}$, which is by default set as 0.5. Averaging is defined as,

$$\bar{\hat{\boldsymbol{\pi}}}_{bag}^{B}(\boldsymbol{x}) = \frac{\boldsymbol{1}}{\boldsymbol{B}}\sum_{b=1}^{B} \hat{\boldsymbol{\pi}}^{b}(\boldsymbol{x}) \tag{4.8}$$

$$\hat{\boldsymbol{f}}_{bag}^{B}(\boldsymbol{x}) = \begin{cases} \boldsymbol{1}\ \boldsymbol{if}\ \bar{\hat{\boldsymbol{\pi}}}_{bag}^{B}(\boldsymbol{x}) \geq \boldsymbol{c} \\ \boldsymbol{0}\ \boldsymbol{if}\ \bar{\hat{\boldsymbol{\pi}}}_{bag}^{B}(\boldsymbol{x}) < \boldsymbol{c} \end{cases} \tag{4.9}$$

CHAPTER 5

EXPERIMENTS AND IMPLEMENTATIONS

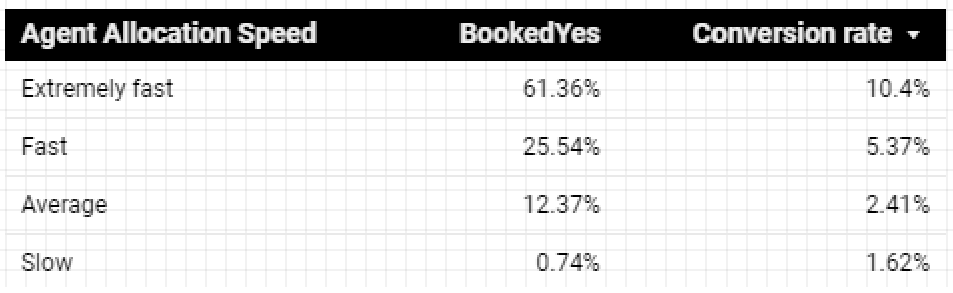
5.1 Overview

Customer Relationship Management (CRM) specialists have access to a massive amount of customer behavioural data. Understanding this data could lead to significant improvements in businesses. Over the years, many methods were introduced to analyse and understand data. Two common challenges faced by companies today are the selection of suitable data analytics techniques and effective utilisation of these data in CRM processes (Gončarovs, 2017). Many companies consider themselves market driven but are still organised around their products. In the era of a rapidly changing, globalised economies, and highly competitive markets, the transformation from a product-centric focus to a more customer-centric view is required (Gončarovs, 2017).

To gain a competitive edge, companies should take advantage of past and current data to predict what might happen in the future (Morelli, et al., 2010). Predicting the future is referred to as predictive analysis in the context of machine learning.

CRM is part of an Enterprise Resource Planning (ERP) constituent that deals with managing the organisations current and the prospects of the customers (Sharma, et al., 2016). In CRM, the analysis is carried out on historical customer data to identify the elements that could aid in customer retention and thereby improving the business. Besides customer retention, CRM also plays a part in bringing new customers to the organisation with a technique called 'Lead management' (Kim, 2004). Leads are potential customers who are likely to do business with the organisation.

The travel agency Ocean Holiday is a company that is focused on improving its leads by applying machine-learning techniques. Ocean Holiday receives approximately 80000 enquires a year. Out of these enquiries, about 5% are valuable enquires. Being able to identify the valuable enquires could significantly improve the business, as resources can be allocated accordingly. Ocean Holiday carried out Exploratory Data Analysis (EDA) as an initial step to determine the cases that are affecting potential customers. It was identified that one of the main factors affecting leads depends on the speed of agent allocation. Table 5.1 shows a summary of the percentage of booked enquiries and the conversion rate for the four different 'agent allocation speed' categories. Conversion rate refers to the percentage of booked enquiries divided by the total enquiries in that category. It is clear from the table that as the allocation speed decreases, the percentage of booked enquires as well as the conversion rate decreases.



Agent Allocation Speed	BookedYes	Conversion rate ▾
Extremely fast	61.36%	10.4%
Fast	25.54%	5.37%
Average	12.37%	2.41%
Slow	0.74%	1.62%

Table 5. 1 Summary of agent allocation speed

This company receives approximately 7000 enquires per month and attending to all these enquiries with optimal speed is close to impossible. Hence a strategy is needed to identify potential customers so that agents could be allocated to the more valuable enquires first, which could result in a higher probability of the customer purchasing the service.

As an initial step to tackle this problem, a logistic regression model with over-sampling was designed to identify potential customers so that agents could be allocated to these customers with optimal speed. This model was able to perform reasonably well, but a model with higher accuracy is desired which could focus on the minority class, as failure to identify the minority class could lead to loss of potential customers. Even though the minority class is the focus, it is also essential to identify the majority class with acceptable accuracy. The model should have a sensitivity of approximately 0.95; this was achieved by changing the threshold levels of the models in this paper. Model comparison in this chapter will be carried out with various threshold levels to meet the 95% sensitivity (recall).

An ideal model should be able to identify the majority of the enquiries that are likely to be booked while being able to identify at least 50% of the enquiries not likely to be booked. In other words, it is acceptable to have a high false positive rate, but ideal to have a specificity of greater than 50%. Considering the case at hand, it would be suitable to use sensitivity, specificity and precision for model evaluation. For general analysis purpose of measuring the overall performance of the models, AUPR, AUROC, ROC curve and PR curve would also be studied. These four methods are used for overall model performance because these measures are not affected when the threshold levels of the models are varied.

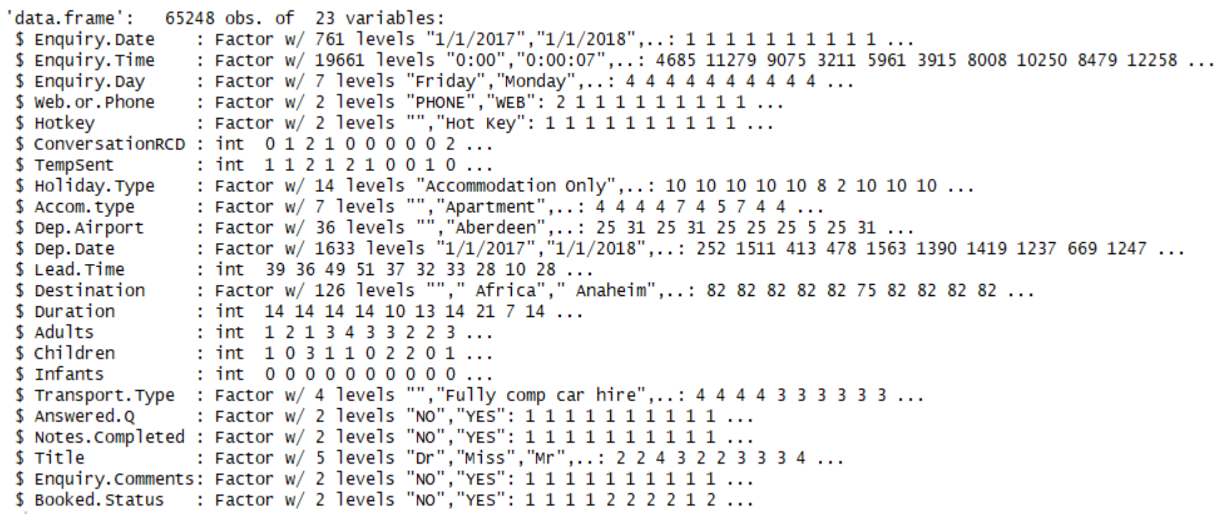
5.2 Data preparation

The dataset used in the paper consists of enquiries recorded by Ocean Holidays over a period of 24 months. The dataset contains 178347 observations with 24 variables of which seven are continuous variables, and the remaining are discrete variables. The target variable is 'BookedStatus'. Sensitive information such as customer details were excluded from this dataset.

As discussed earlier the idea is to build a model that can predict if a customer will book the company's service when an agent is allocated to an enquiry within 24 hours. Considering this, using the entire dataset will not yield acceptable results as the data consists of enquires where agents were allocated with various speeds. For a model to be useful in predicting if a person is likely to purchase the service if an agent is allocated to the individual with optimal speed, the model should only learn from observations where the agents were allocated 'Extremely Fast' (less than 1 hour) and 'Fast' (less than 24hour). Hence, the dataset was filtered from enquiries where agents responded to an enquiry after 24 hours. The resulting dataset consists of 116868 observations with an imbalance ratio of 2:23. The variable 'AgentAllocationSpeed' was eliminated from the dataset as this has no further functionality in this analysis. The minority class (YES) represents the enquires that are booked, and the majority class (NO) represents enquiries that are not booked.

5.2.1 Exploratory Data Analysis/ Data Pre-processing

Understanding the data is a crucial step before modelling. Once the data is loaded into R-Studio, it is vital to ensure that R-Studio has managed to identify the variable types correctly. In R-studio categorical variables are identified as 'factor' and numerical variables as 'int'. Hence, as an initial step, the structure of the dataset is explored. Table 5.2 shows the structure of all the available variables in the dataset. From this table, it is observed that some of the categorical variables have an incredibly high number of levels. Rare levels were grouped after carrying out some visualisations of the frequency distributions. This will be further explained in the next section.



```
'data.frame':	65248 obs. of  23 variables:
 $ Enquiry.Date    : Factor w/ 761 levels "1/1/2017","1/1/2018",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ Enquiry.Time    : Factor w/ 19661 levels "0:00","0:00:07",..: 4685 11279 9075 3211 5961 3915 8008 10250 8479 12258 ...
 $ Enquiry.Day     : Factor w/ 7 levels "Friday","Monday",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Web.or.Phone    : Factor w/ 2 levels "PHONE","WEB": 2 1 1 1 1 1 1 1 1 1 ...
 $ Hotkey          : Factor w/ 2 levels "","Hot Key": 1 1 1 1 1 1 1 1 1 1 ...
 $ ConversationRCD : int  0 1 2 1 0 0 0 0 0 2 ...
 $ TempSent        : int  1 1 2 1 2 1 0 0 1 0 ...
 $ Holiday.Type    : Factor w/ 14 levels "Accommodation Only",..: 10 10 10 10 10 8 2 10 10 10 ...
 $ Accom.type      : Factor w/ 7 levels "","Apartment",..: 4 4 4 4 7 4 5 7 4 4 ...
 $ Dep.Airport     : Factor w/ 36 levels "","Aberdeen",..: 25 31 25 31 25 25 25 5 25 31 ...
 $ Dep.Date        : Factor w/ 1633 levels "1/1/2017","1/1/2018",..: 252 1511 413 478 1563 1390 1419 1237 669 1247 ...
 $ Lead.Time       : int  39 36 49 51 37 32 33 28 10 28 ...
 $ Destination     : Factor w/ 126 levels ""," Africa"," Anaheim",..: 82 82 82 82 82 75 82 82 82 82 ...
 $ Duration        : int  14 14 14 14 10 13 14 21 7 14 ...
 $ Adults          : int  1 2 1 3 4 3 3 2 2 3 ...
 $ Children        : int  1 0 3 1 1 0 2 2 0 1 ...
 $ Infants         : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Transport.Type  : Factor w/ 4 levels "","Fully comp car hire",..: 4 4 4 4 3 3 3 3 3 3 ...
 $ Answered.Q      : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
 $ Notes.Completed : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
 $ Title           : Factor w/ 5 levels "Dr","Miss","Mr",..: 2 2 4 3 2 2 3 3 3 4 ...
 $ Enquiry.Comments: Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
 $ Booked.Status   : Factor w/ 2 levels "NO","YES": 1 1 1 1 2 2 2 2 1 2 ...
```

Table 5. 2 Data Structure

To get a better understanding of the data, a statistical analysis was carried out as shown in table 5.3. From this table, it is observed that the variable ‘Hotkey’ has many missing values/unlabelled cases. This is due to the reason that only in the occurrence of a ‘Hotkey’ the data was recorded, otherwise left blank. A logistic regression model ignores rows with missing values; hence, this variable was coded as ‘1’ for ‘Hotkey’ and ‘0’ for the missing values. The variable ‘ConversationRCD’ has a maximum value of 77; this could be a data entry error. This will be further explored using a scatterplot. It would not be useful to use dates directly in this analysis; hence it should be converted to months/seasons. The variable ‘Lead.Time’ (weeks before departure) has a minimum value of -44; this variable should not contain any negative values as ‘Lead.Time’ refers to the weeks between enquiry and departure date. Finally, the variable Duration has some missing values; these rows were omitted from the dataset as only a negligible number of rows had missing values.

```
   Enquiry.Date    Enquiry.Time       Enquiry.Day    Web.or.Phone       Hotkey      ConversationRCD      TempSent                   Holiday.Type
1/2/2019  :  210   13:00  :  104   Friday   : 8319   PHONE:23080          :35964   Min.   : 0.000   Min.   : 0.000   Package Holiday   :49678
3/30/2017 :  191   12:56  :  101   Monday   :10859   WEB  :42168    Hot Key:29284   1st Qu.: 0.000   1st Qu.: 1.000   Multi Centre      : 5511
1/21/2019 :  180   13:24  :  101   Saturday : 7762                                  Median : 0.000   Median : 1.000   Fly Drive         : 5244
12/30/2018:  177   14:30  :   99   Sunday   : 8621                                  Mean   : 1.595   Mean   : 1.597   Cruise + Stay     : 2322
1/9/2017  :  174   16:06  :   98   Thursday : 9058                                  3rd Qu.: 2.000   3rd Qu.: 2.000   Accommodation Only: 2067
12/27/2018:  172   12:44  :   97   Tuesday  : 9989                                  Max.   :77.000   Max.   :29.000   Flight Only       :  196
(Other)   :64144   (Other):64648   Wednesday:10640                                                                    (Other)           :  230
    Accom.type              Dep.Airport         Dep.Date       Lead.Time                       Destination       Duration         Adults
         : 4647   Manchester      :20697   4/6/2019 :  461   Min.   :-44.00    Orlando                 :20524   Min.   : 0.00   Min.   : 1.000
Apartment: 3553   London All      :15173   8/1/2018 :  428   1st Qu.: 28.00    FLORIDA                 :18812   1st Qu.:14.00   1st Qu.: 2.000
Combined :   35   London Gatwick  :11420   3/30/2018:  391   Median : 46.00    Walt Disney World Resort: 4844   Median :14.00   Median : 3.000
Hotel    :32921   Glasgow         : 5282   8/1/2019 :  391   Mean   : 47.73    Kissimmee Area          : 3779   Mean   :13.35   Mean   : 3.664
None     :  157   London Heathrow: 2602   7/28/2018:  329   3rd Qu.: 65.00    International Drive     : 3709   3rd Qu.:14.00   3rd Qu.: 4.000
TownHome :  114   Any Airport     : 2443   3/31/2018:  312   Max.   :216.00    Disney Area             : 2308   Max.   :63.00   Max.   :58.000
Villa    :23821   (Other)         : 7631   (Other)  :62936                   (Other)                 :11272   NA's   :17
   Children           Infants                        Transport.Type  Answered.Q  Notes.Completed  Title      Enquiry.Comments Booked.Status
Min.   : 0.000   Min.   : 0.00000                           : 5600   NO :45338   NO :55266        Dr  :  122   NO :54414        NO :58706
1st Qu.: 0.000   1st Qu.: 0.00000   Fully comp car hire:28195   YES:19910   YES: 9982        Miss: 8303   YES:10834        YES: 6542
Median : 1.000   Median : 0.00000   None Required      :16267                                Mr  :29399
Mean   : 1.065   Mean   : 0.05787   Return transfers   :15186                                Mrs :25568
3rd Qu.: 2.000   3rd Qu.: 0.00000                                                            Ms  : 1856
Max.   :16.000   Max.   :10.00000
```

Table 5. 3 Statistical Analysis

It is essential to remove outliers from the data as this reduces the problem of overfitting. A common approach to remove outlier is the Boxplot method, but in this dataset, using a box plot to eliminate outliers resulted in a considerable number of minority observations being removed. Hence, a manual method of removing outliers by analysing scatterplot is carried out to deal with extreme values.

Figure 5.1 shows the scatterplot of ‘Adults’, ‘Children’ and ‘Infants’. From these scatterplots, it was decided that,

- Adults >15 will be treated as outliers
- Children >8 will be treated as outliers
- Infants >2 will be treated as outliers

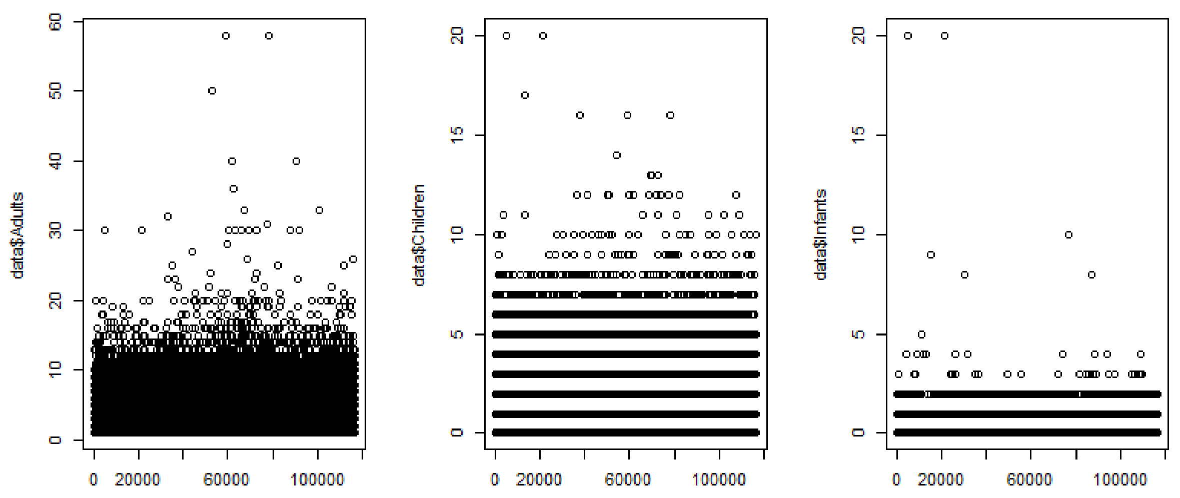


Figure 5. 1 Scatterplot (‘Adults’, ‘Children’, ‘Infants’)

Figure 5.2 shows the scatterplot of ‘Duration’ and ‘LeadTime’. From these scatterplots, it was decided that,

- Duration >30 will be treated as outliers
- LeadTime>140 will be treated as outliers and LeadTime <0 will be treated as outliers

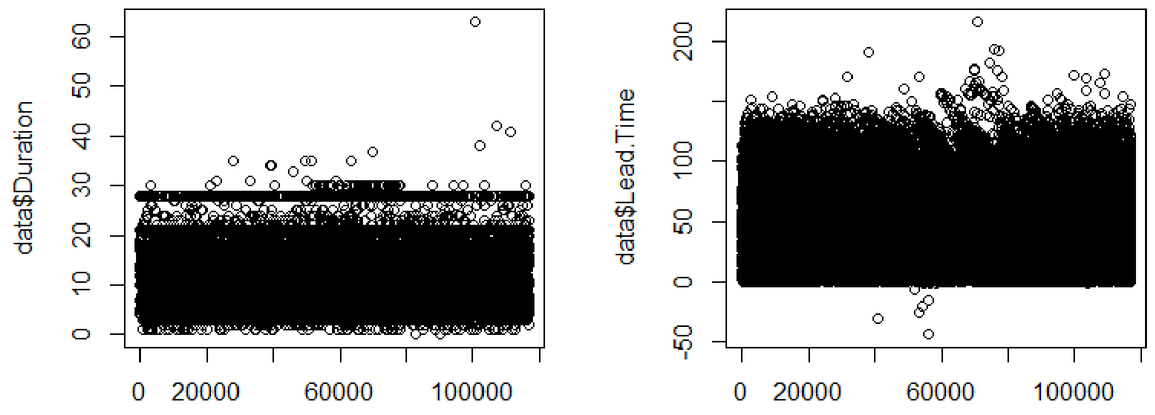


Figure 5. 2 Scatterplot (‘Duration’, ‘Lead Time’)

Figure 5.3 shows the scatterplot of ‘TempSent’ and ‘ConversationRCD’. From these scatterplots, it was decided that,

- TempSent >10 will be treated as outliers
- ConversationRCD >30 will be treated as outliers

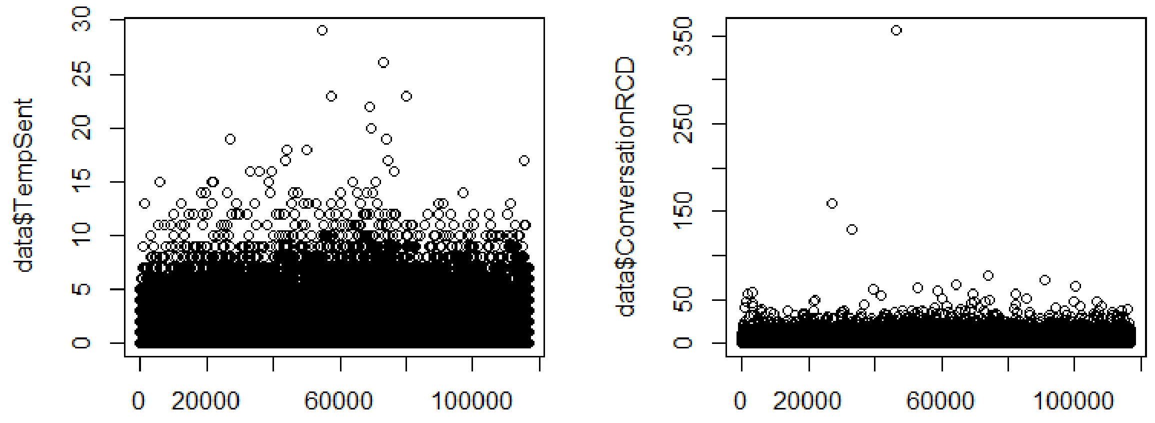


Figure 5. 3 Scatterplot (‘TempSent’, ‘ConversationRCD’)

Outlier removal was carried out by a function which replaces the specified value as ‘NA’. The dataset was then filtered from any rows containing ‘NA’ values. Refer to appendix for the outlier removal function.

Some of the variable levels in the dataset were converted into groups which are more meaningful for this analysis. ‘Enquiry.Time’, ‘Enquiry.Date’ and ‘Dep.Date’ have multiple levels based on date and time. These variables will be more useful when grouped into more significant levels.

- ‘Enquiry.Time’ was converted into two categories, Business_hour and closed and labelled as ‘Enquiry.Timecat’. Another version of ‘Enquiry.Time’ was created as Morning, Afternoon and Night and was labelled as ‘Enquiry.Time_class’

- ‘Enquiry.Date’ and ‘Dep.Date’ were converted into monthly levels and was labelled as ‘EnquiryMonth’ and ‘DepartureMonth’ respectively. Another version of these two variables were created for season levels (Summer, Winter, Spring, Autumn) and was labelled as ‘EnquirySeason’ and ‘DepartureSeason’.

The variable ‘Destination’ had 126 levels. A frequency count of the levels was carried out, and it was found that this variable has many levels with few observations. It was decided that levels below a frequency of 5% be grouped as ‘OTHER’. Figure 5.4 shows the resulting levels.

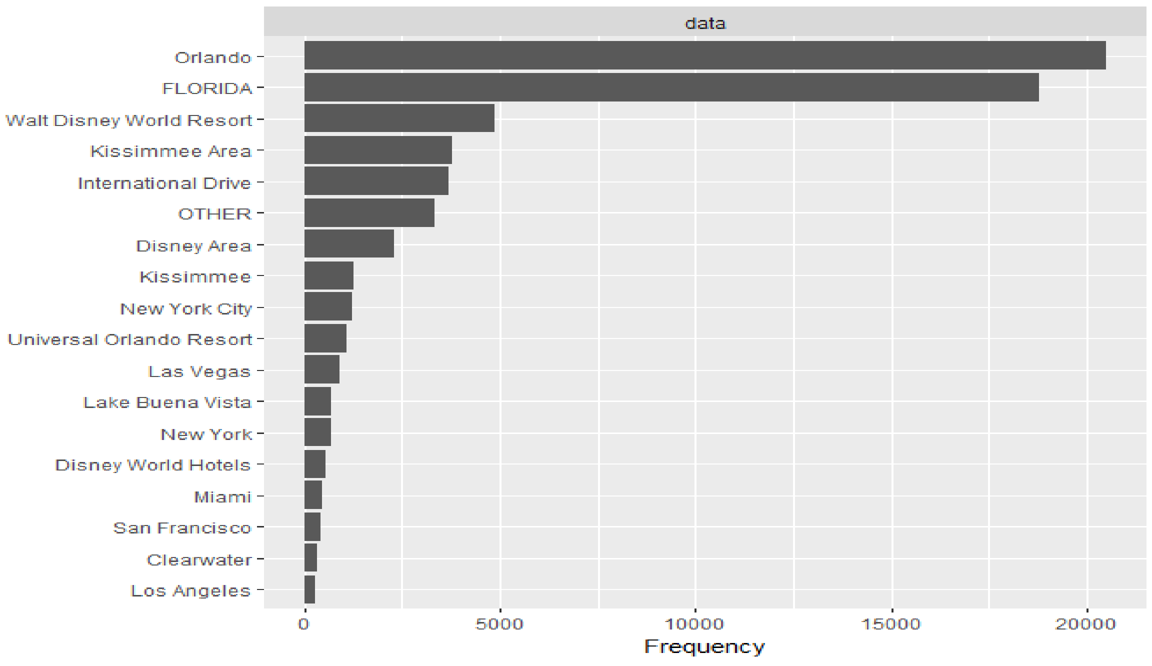


Figure 5. 4 Frequency Distribution of ‘Destination’

The variable ‘DepAirport’ had 36 levels. Similar to Destination, a threshold frequency of 5% was set, and the rare levels were grouped. Figure 5.5 shows the result after grouping the rare cases.

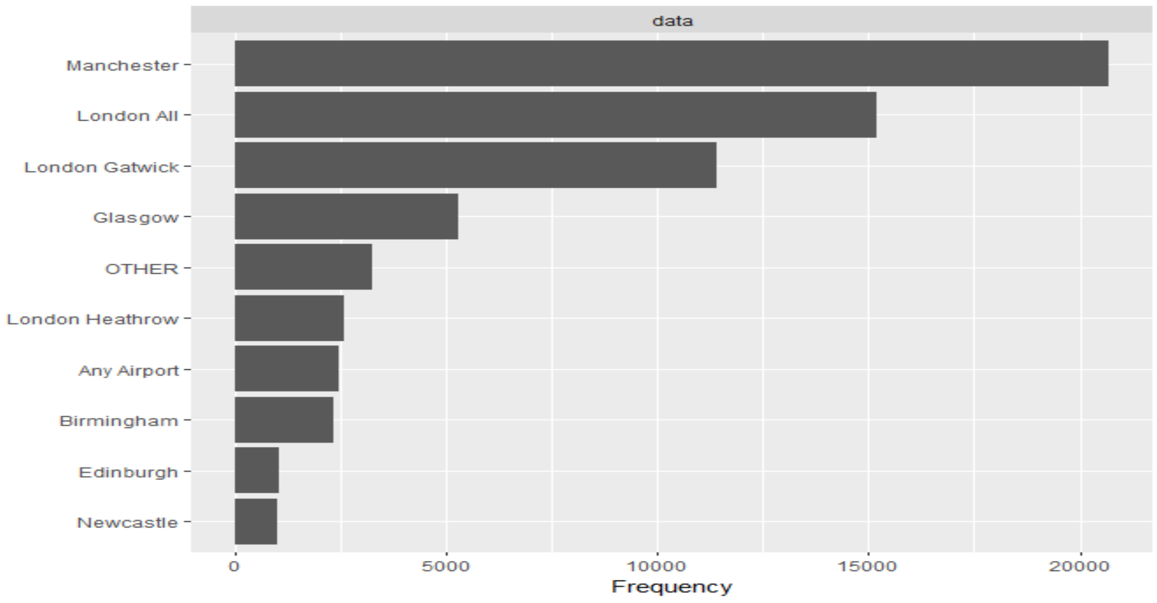


Figure 5. 5 Frequency Distribution of 'DepAirport'

The variable 'AccomType' contains seven levels, but three of these levels occurs rarely. Furthermore, there is a considerable number of unlabeled observations. In this case, the unlabelled variables were combined with the level 'None'. The other two rare levels were combined with apartments. Figure 5.6 shows the frequency distribution after grouping the rare levels.

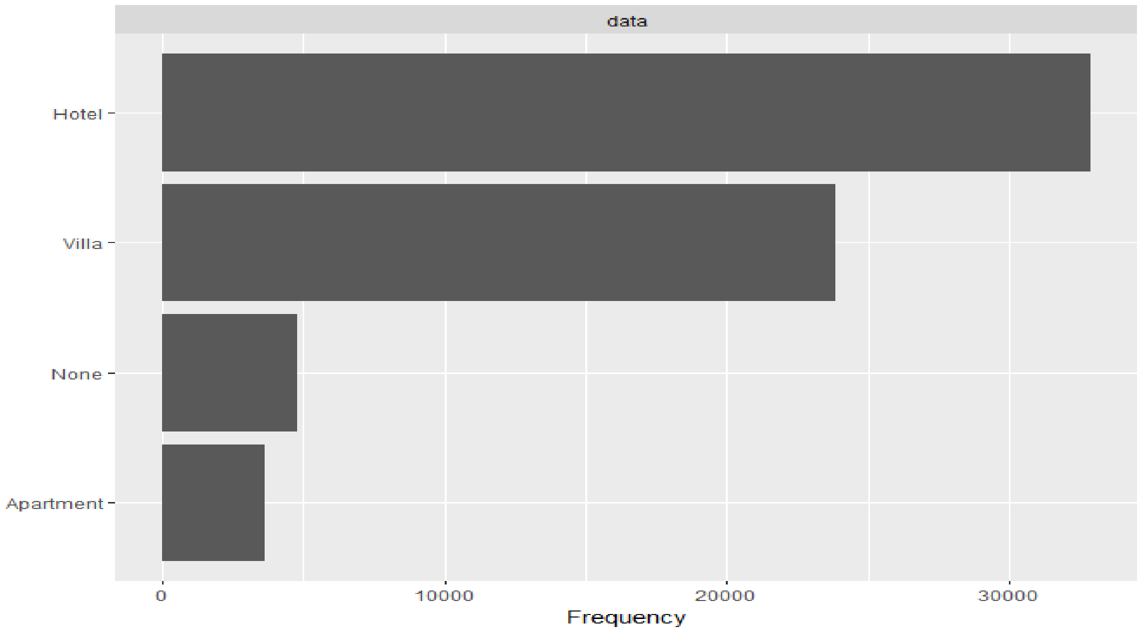


Figure 5. 6 Frequency Distribution of 'AccomType'

The variable ‘Transport.Type’ has four levels of which one was missing values. This level is combined with the level ‘None Required’. Figure 5.7 shows the resulting frequency distribution plot.

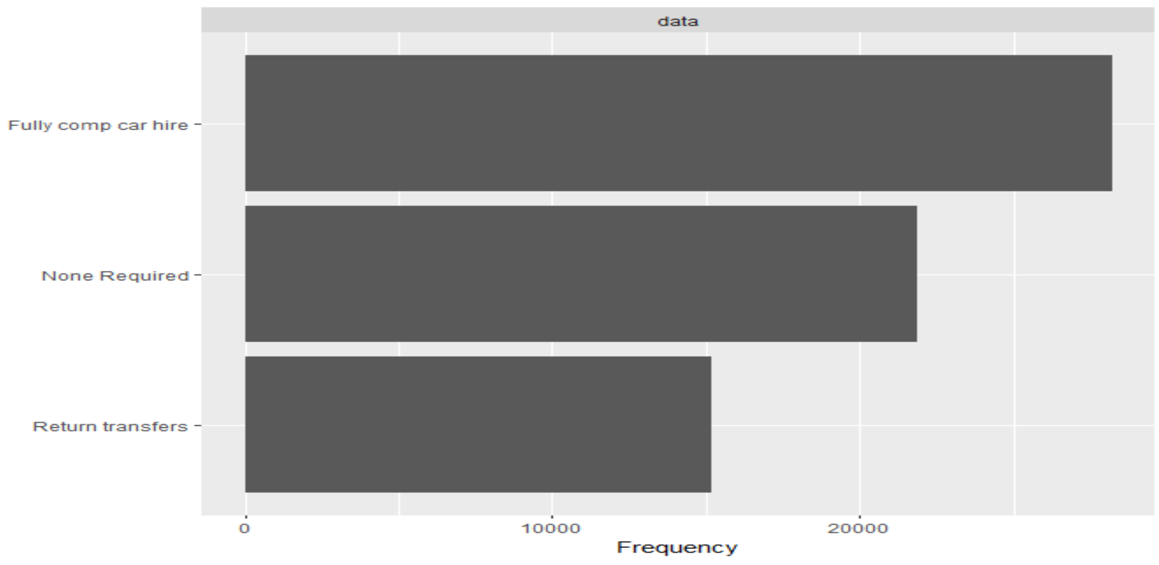
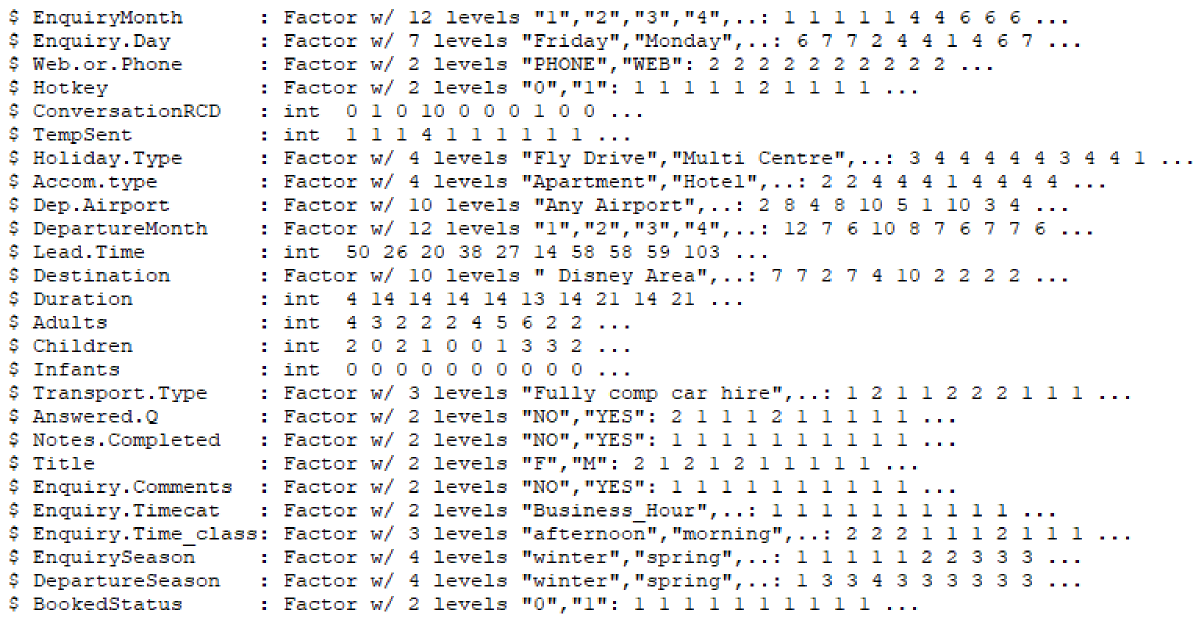


Figure 5. 7 Frequency Distribution of ‘Transport.Type’

The variable ‘Title’ has five levels. Miss, Ms and Mrs were grouped as ‘F’ the rest as ‘M’. Finally, the target variable ‘BookedStatus’ was recorded as ‘YES’ and ‘NO”, this was changed to dummy variables ‘1’ and ‘0’ respectively. Table 5.4 shows the data structure after carrying out the data preparations mentioned in this section.



```
$ EnquiryMonth      : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 4 4 6 6 6 ...
$ Enquiry.Day       : Factor w/ 7 levels "Friday","Monday",..: 6 7 7 2 4 4 1 4 6 7 ...
$ Web.or.Phone      : Factor w/ 2 levels "PHONE","WEB": 2 2 2 2 2 2 2 2 2 2 ...
$ Hotkey            : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
$ ConversationRCD   : int  0 1 0 10 0 0 0 1 0 0 ...
$ TempSent          : int  1 1 1 4 1 1 1 1 1 1 ...
$ Holiday.Type      : Factor w/ 4 levels "Fly Drive","Multi Centre",..: 3 4 4 4 4 4 3 4 4 1 ...
$ Accom.type        : Factor w/ 4 levels "Apartment","Hotel",..: 2 2 4 4 4 1 4 4 4 4 ...
$ Dep.Airport       : Factor w/ 10 levels "Any Airport",..: 2 8 4 8 10 5 1 10 3 4 ...
$ DepartureMonth    : Factor w/ 12 levels "1","2","3","4",..: 12 7 6 10 8 7 6 7 7 6 ...
$ Lead.Time         : int  50 26 20 38 27 14 58 58 59 103 ...
$ Destination       : Factor w/ 10 levels " Disney Area",..: 7 7 2 7 4 10 2 2 2 2 ...
$ Duration          : int  4 14 14 14 14 13 14 21 14 21 ...
$ Adults            : int  4 3 2 2 2 4 5 6 2 2 ...
$ Children          : int  2 0 2 1 0 0 1 3 3 2 ...
$ Infants           : int  0 0 0 0 0 0 0 0 0 0 ...
$ Transport.Type    : Factor w/ 3 levels "Fully comp car hire",..: 1 2 1 1 2 2 2 1 1 1 ...
$ Answered.Q        : Factor w/ 2 levels "NO","YES": 2 1 1 1 2 1 1 1 1 1 ...
$ Notes.Completed   : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
$ Title             : Factor w/ 2 levels "F","M": 2 1 2 1 2 1 1 1 1 1 ...
$ Enquiry.Comments  : Factor w/ 2 levels "NO","YES": 1 1 1 1 1 1 1 1 1 1 ...
$ Enquiry.Timecat   : Factor w/ 2 levels "Business_Hour",..: 1 1 1 1 1 1 1 1 1 1 ...
$ Enquiry.Time_class: Factor w/ 3 levels "afternoon","morning",..: 2 2 2 1 1 1 2 1 1 1 ...
$ EnquirySeason     : Factor w/ 4 levels "winter","spring",..: 1 1 1 1 1 2 2 3 3 3 ...
$ DepartureSeason   : Factor w/ 4 levels "winter","spring",..: 1 3 3 4 3 3 3 3 3 3 ...
$ BookedStatus      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Table 5. 4 Data structure after data preperation

5.2.2 Data preparation evaluation on model

As mentioned in chapter 3, the steps for adequate data preparation involves,

- Brainstorm features
- Create features
- Check how the features work with the model
- Repeat the process until the features work perfectly

After creating these features, a basic logistic regression model with under-sampling was used to gauge how these features perform by comparing sensitivity (the ability of the model to identify minority class) at a threshold of 0.5. Table 5.5 shows the summary of the results carried out using different data treatment methods.

	Original Data	BoxPlot method	Manual removal	Boxplot and data preparation	Manual removal and data preparation
Specificity	0.7640	0.6792	0.7625	0.6763	0.7533
Sensitivity	0.7106	0.7295	0.7325	0.7287	0.7491

Table 5. 5 Summary of feature evaluation on models

From the results, it is observed that the boxplot method causes a considerable drop in the specificity, compared to using the original dataset. Furthermore, applying data preparation methods together with boxplot method caused a further reduction in the model performance. The method to manually remove outliers based on a scatterplot was more successful in improving the model sensitivity while maintaining a reasonable specificity. This method together with data preparation methods resulted in a model that has a higher sensitivity by approximately 4% as compared to using the original dataset. The steps of data preparation should be carried out for each model and the steps repeated until the features work flawlessly for the model. In this paper, due to the time constrain an assumption is made that the manual removal and data preparation method works perfectly for all models. Hence, the dataset was refined with the manual removal of outliers and data preparation methods. This prepared dataset was used for creating all the models in this chapter.

5.3 Data partition

It is essential to test the performance of a model, and this should be carried out on unseen data or data that is not used to build the model. One way to test the model performance is not to use the entire dataset when training a model. A small percentage of the data should be randomly removed from the entire dataset before training a model, and this removed data can be used to test the performance of the trained model. In this paper, the basic cross-validation method, known as ‘holdout method’ will be adopted. The holdout method splits the data into testing and training dataset. In this paper, a 70/30 random split without replacement is carried out; 70% random observations for training the model and 30% of the remaining observations for testing the performance of the model.

5.4 Logistic regression based classifier

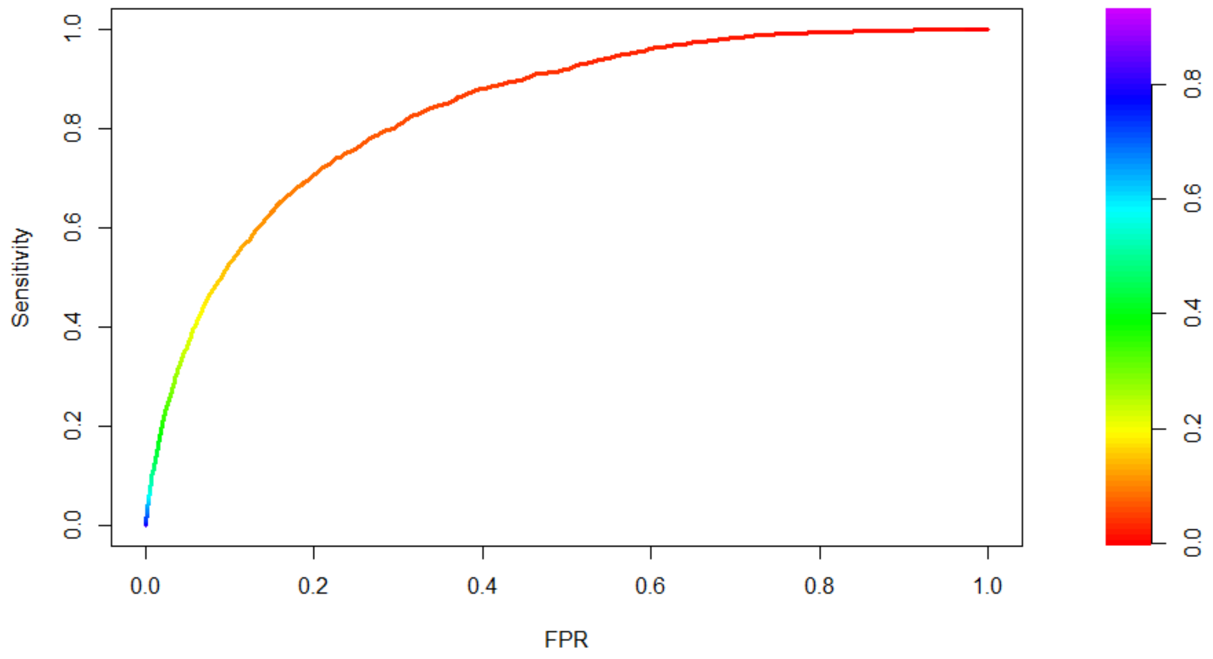


Figure 5. 8 Logistic regression (ROC)

Using the training dataset with no data balancing method, a logistic regression was modelled. As shown in Figure 5.8, this logistic regression model was able to handle the data imbalance in this dataset. This could be due to the size of the training data, a vast domain has a good chance that the majority class is represented by a reasonable number of examples and thus may be less affected by imbalance (Jo & Japkowicz, 2005).

The AUC (0.837) obtained proved that the overall performance of the model is above average. From the colour code on the right, which represents the threshold levels, it is observed that a low threshold level, less than 0.1 should be set in order to obtain high sensitivity. Overall, logistic regression was able to perform well despite the data imbalance for this particular dataset.

Next, to understand the effects of data balancing methods on a logistic regression model, experiments were carried out using three different sampling methods. Figure 5.9 shows the ROC plot for all the three models created using the three different sampling methods as well as the model without any sampling method. The ROC plot shows that all four models perform similarly. The SMOTE method seems to be performing marginally lower compared to the other methods. Overall sampling methods did not provide any significant improvements despite a longer computation time when using SMOTE and over-sampling.

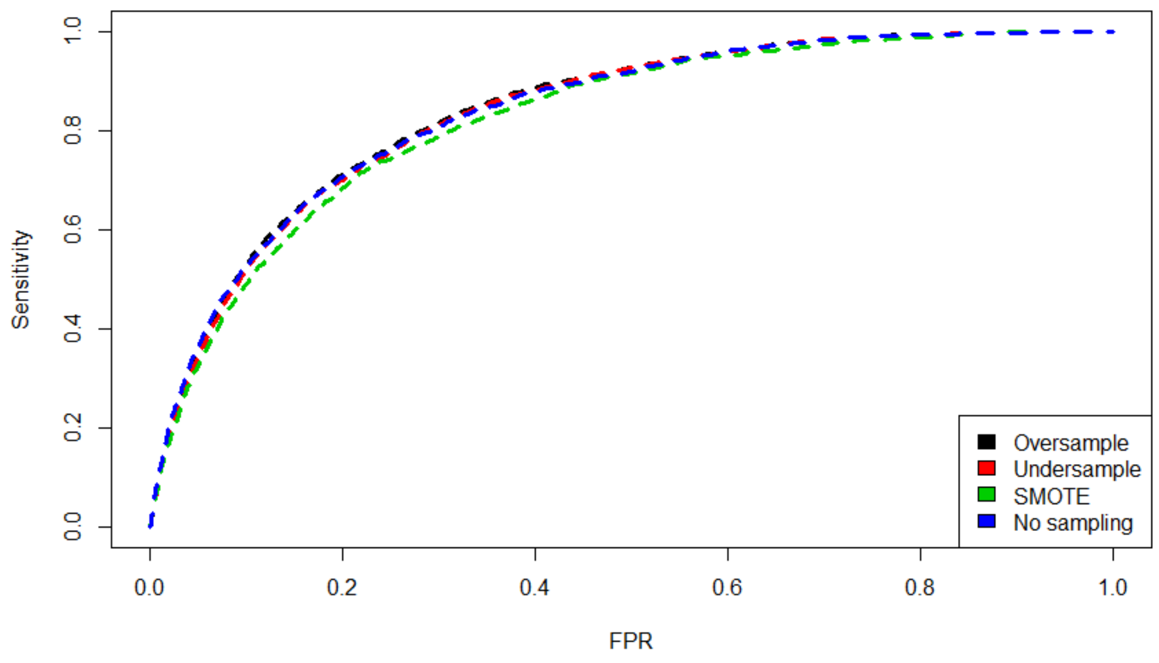


Figure 5. 9 Logistic regression (ROC) Sampling methods and basic model

From analysing the precision-recall curve shown in Figure 5.10, it was understood that balancing methods had reduced the overall AUC of a precision-recall curve. These plots from ROC and PR curves indicates that even if a data is imbalanced, a data balancing method is not necessary unless the model fails to perform well with the imbalanced dataset.

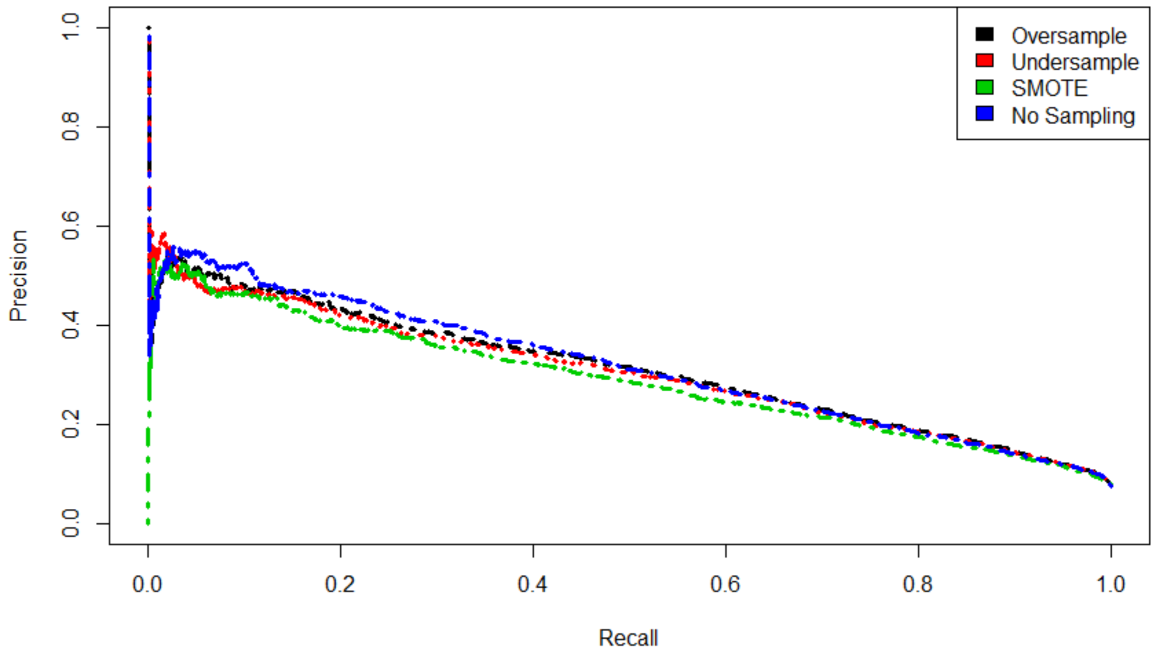


Figure 5. 10 Logistic regression (PR) Sampling methods and basic model

Table 5.6 shows the results of the four models. As mentioned earlier in the chapter, a fixed sensitivity of 0.95 is desired for this analysis, and hence all analysis is carried out by changing the threshold to obtain a sensitivity of 0.95. Based on the model requirements as stated in section 5.1, the best model is the model designed using no sampling method as this model has the highest specificity for the specified sensitivity. Furthermore, the overall performance of this model in terms of AUROC is the second best and has the highest AUPR as shown in Table 5.6. This experiment shows that using data balancing method when not necessary could result in models with weaker performance.

	Training data	Over-sample	Under-sample	SMOTE
Sensitivity	0.95	0.95	0.95	0.95
Specificity	0.432	0.431	0.429	0.409
AUROC	0.837	0.839	0.836	0.824
Precision	0.120	0.120	0.116	0.120
AUPR	0.317	0.311	0.305	0.290
Threshold	0.029	0.239	0.239	0.21

Table 5. 6 Model evaluation summary (sampling and training data) Logistic regression

As the basic model with no data balancing method is the best performing model among the four models, this model will be used as a standard to gauge the remaining models build using a logistic regression classifier.

5.4.1 Bagged logistic regression

Bagging was carried out by setting the number of bootstraps to 100; this means that 100 bootstrapped models are created. The output of each model was combined to obtain the final prediction by majority vote. As expected, there were no actual improvements to the AUC compared to using basic sampling methods. This is due to the reason that a logistic regression model is a linear stable classifier and learning from different subset do not result in considerable changes to the predictions (low diversity). That means to say that models created from bagging logistic regression are highly correlated and did not provide much improvements compared to the basic models. This proves the point mentioned is some of the literature, that bagging works well only with unstable classifiers.

	Bagged (Training data)	Bagged (Over-sample)	Bagged (Under-sample)	Bagged (SMOTE)
Sensitivity	0.95	0.95	0.95	0.95
Specificity	0.41	0.43	0.425	0.38
AUROC	0.826	0.830	0.830	0.813
Precision	0.12	0.12	0.12	0.11
AUPR	0.305	0.306	0.306	0.290

Table 5. 7 Model evaluation summary (Bagging logistic regression)

From Table 5.7 it is clear that there are no significant improvements by bagging a logistic regression model. The AUROC also shows that the overall model performance is approximately the same for bagged models with no sampling, over-sampling and under-sampling. Similar to a single model, SMOTE did not perform well even when integrated with a bagged logistic regression model. (Hanifah, et al., 2015) stated that SMOTE bagging algorithm with a logistic regression base classifier increases the accuracy, but in the experiment carried out SMOTE bagging performed worst than the other sampling methods with bagging as well as basic models without bagging. This could be due to the reason that we only used 100 bootstrapped samples and a fixed value of k=2 for SMOTE.

From the ROC curves in Figure 5.11 and the PR curves in Figure 5.12, it is observed that bagging did not provide any significant improvements. The basic model with no sampling method was still superior. SMOTE bagging performed worse than the other sampling methods.

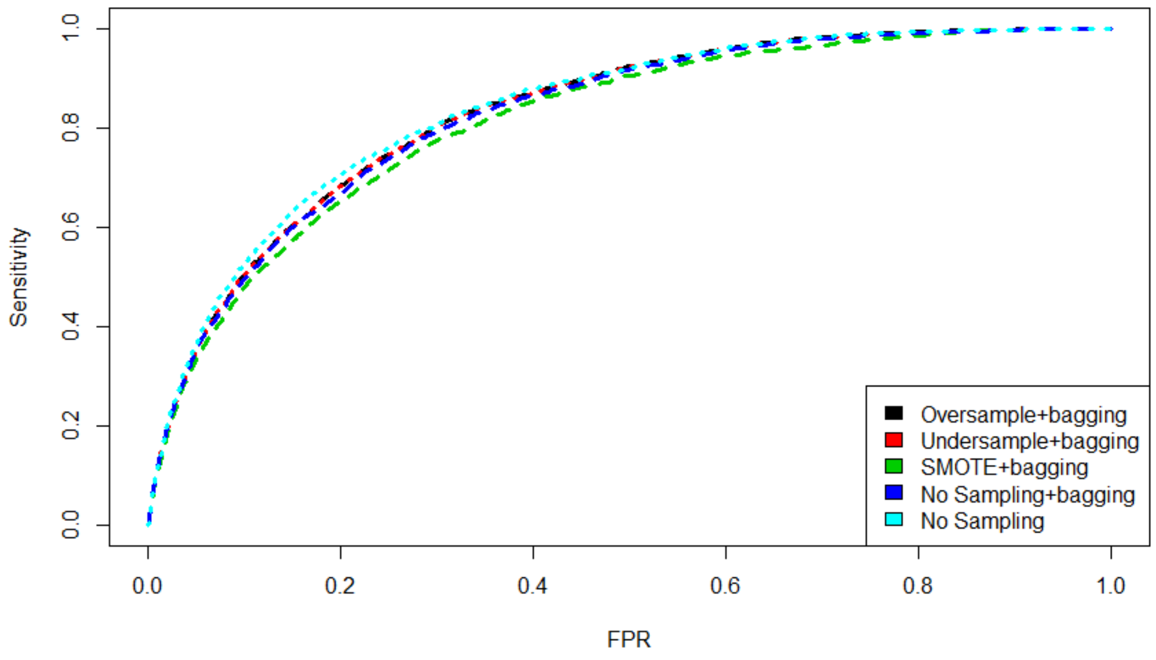


Figure 5. 11 Logistic regression (ROC) Sampling methods and bagging

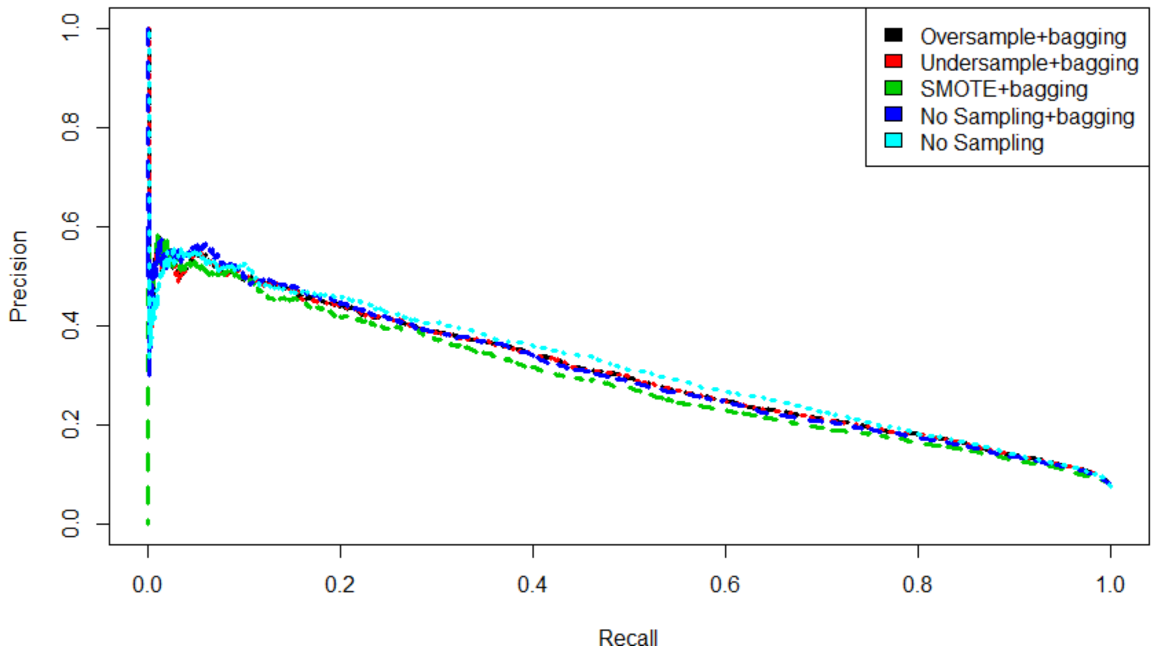


Figure 5. 12 Logistic regression (PR) Sampling methods and bagging

5.4.2 Proposed Method (Logistic regression)

The proposed method is based on a level subset of each categorical variable. In this dataset, there are some variables which have more than 10 levels. Creating a model for each of these levels would be time-consuming. Hence, in this paper, only categorical variables with five levels or less will be applied with the proposed method. 36 data subsets were created from 13 different variables.

	Proposed (No sampling)	Proposed (Over-sample)	Proposed (Under-sample)	Proposed (SMOTE)
Sensitivity	0.95	0.95	0.95	0.95
Specificity	0.42	0.43	0.44	0.39
AUROC	0.828	0.832	0.833	0.826
Precision	0.12	0.12	0.12	0.11
AUPR	0.308	0.305	0.308	0.304
Threshold	0.035	0.260	0.273	0.20

Table 5. 8 Model evaluation summary (Proposed method and sampling) logistic regression

From Table 5.8 it was understood that the proposed method did not cause any significant improvements. The base model with no balancing method was still superior based on AUROC comparison. Comparing the specificity between all the models, the proposed model with under-sampling was able to achieve a marginally higher specificity at a sensitivity of 0.95.

From Figure 5.13 and 5.14 it is observed that the overall performance of all the models is similar. The major difference occurs when the recall is below 0.8. After a recall (sensitivity) of 0.8, all the models were performing almost equally. The difference in the specificity of the models at 0.95 sensitivity is minor. The presented method is time-consuming. Hence it would not be ideal to use this method with a logistic regression based classifier for such a small gain unless a small gain in sensitivity translates to a huge difference in reality.

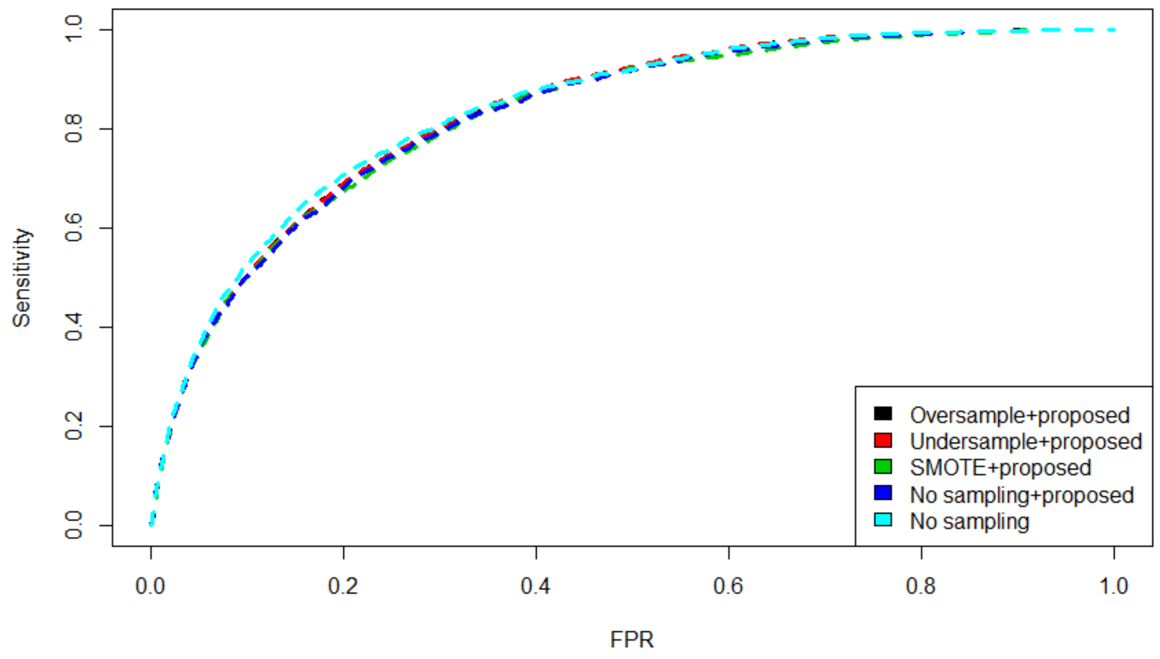


Figure 5. 13 Logistic regression (ROC) Sampling methods and Proposed method

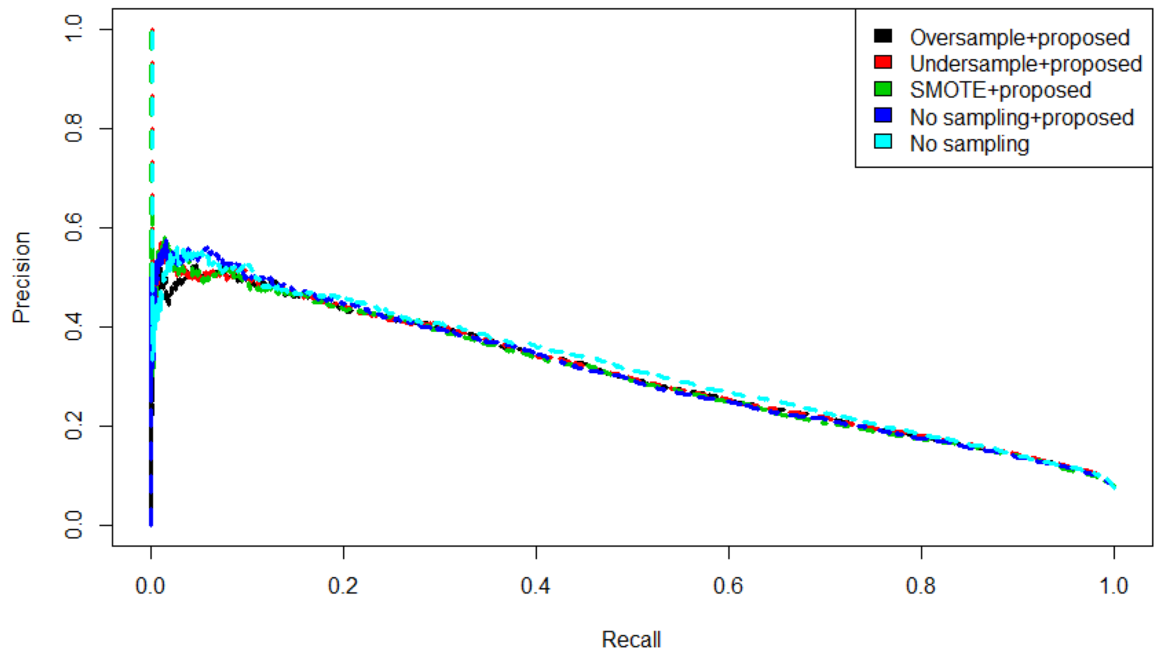


Figure 5. 14 Logistic regression (PR) Sampling methods and the proposed method

5.5 Decision tree (CART) classifier

In a single decision tree, it is impossible to obtain a specific sensitivity as compared to logistic regression; therefore, the model comparison is based on maximum sensitivity (excluding 1) obtainable for a tree. The overall model performance is gauged using the same methods used in section 5.4.

The decision tree model was unable to handle the data imbalance; this is proven by the AUROC of 0.5 as indicated in Table 5.9. Data balancing methods were required to make sensible predictions for a decision tree. (Jo & Japkowicz, 2005) stated that a huge dataset with a reasonable number of minority class might be less affected by the imbalance, but the experiments carried out shows that the algorithm of the classifiers plays a higher role in the imbalance domain. This conclusion was made because using the same dataset, a logistic regression was able to handle the imbalance while a decision tree failed.

Comparing sensitivity, precision and specificity all three data balancing methods have the same performance as shown in Table 5.8. Data rebalanced using SMOTE seems to have a minor advantage as compared to other sampling methods in terms of AUROC and AUPR.

	Training data	Over-sample	Under-sample	SMOTE
Sensitivity	0	0.88	0.88	0.88
Specificity	1	0.574	0.574	0.574
AUROC	0.5	0.78	0.78	0.79
Precision	NA	0.145	0.145	0.145
AUPR	NA	0.19	0.19	0.23
Threshold	0.029	0.4	0.4	0.3

Table 5. 9 Model evaluation summary (sampling and training data) Decision tree

Figure 5.15 shows the ROC plot. From the roc plot, it can be observed that the data balancing method SMOTE was able to perform better overall compared to the over-sampling and under-sampling methods. Both over-sampling and under-sampling have the same overall performance as indicated by the overlapping plot. When compared between these two methods, under-sampling would be a better choice due to the lower computation time. Overall SMOTE sampling method seems to be a better choice to be used alongside a decision tree algorithm.

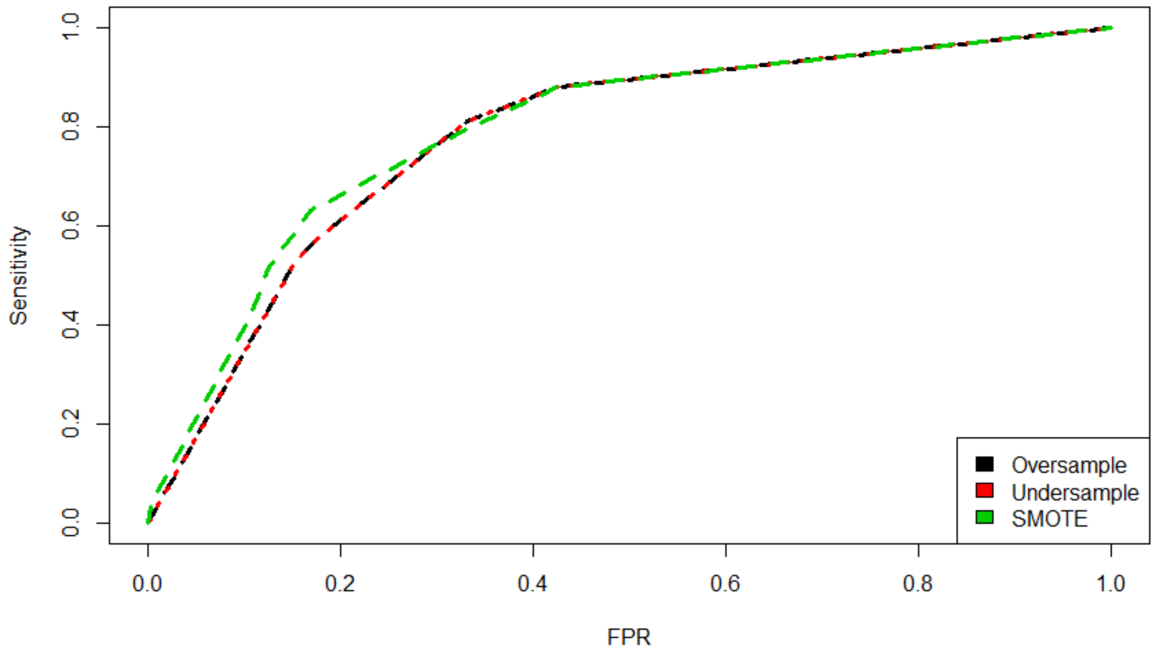


Figure 5. 15 Decision tree (ROC) Sampling methods

Figure 5.16 shows the PR plot. Similar to the ROC plot, both over-sampling and under-sampling had the same overall performance. SMOTE based method has the best overall performance for a precision-recall plot.

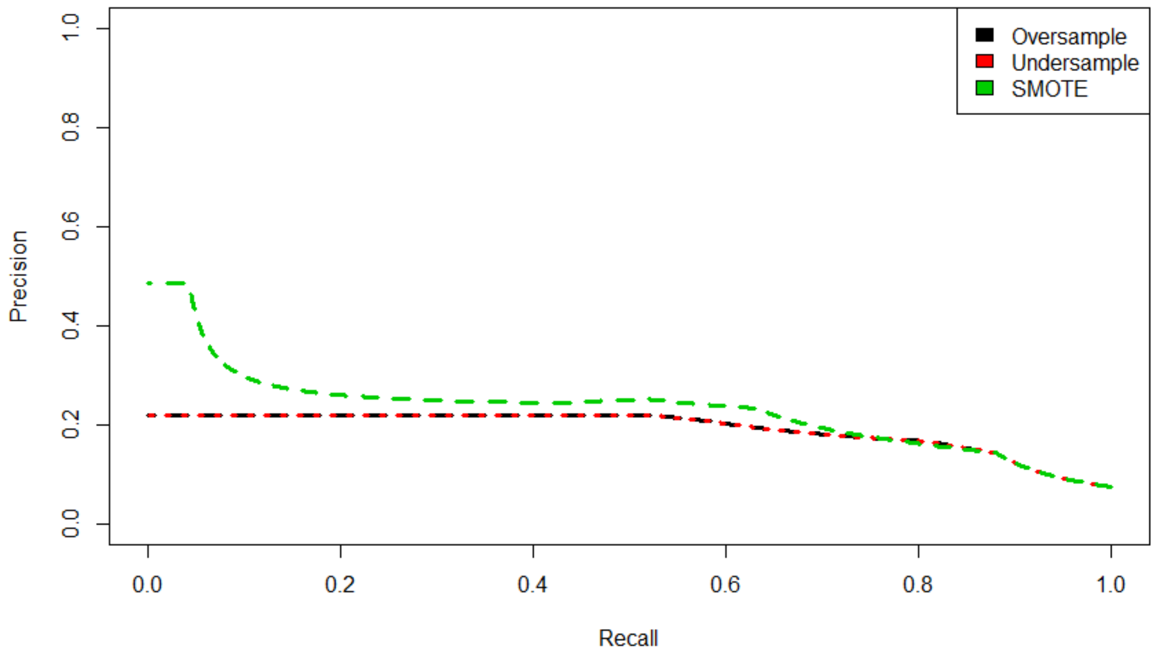


Figure 5. 16 Decision tree (PR) Sampling methods

As SMOTE balanced dataset model has the best overall performance among these models, this model was used as a standard to gauge the models build using decision trees.

5.5.1 Bagged decision tree (CART)

	Bagged (No sampling)	Bagged (Over-sample)	Bagged (Under-sample)	Bagged (SMOTE)
Sensitivity	0.95	0.95	0.95	0.95
Specificity	0.43	0.40	0.44	0.42
AUROC	0.821	0.813	0.833	0.826
Precision	0.12	0.11	0.12	0.12
AUPR	0.321	0.288	0.305	0.298
Threshold	0.09	0.01	0.24	0.05

Table 5. 10 Model evaluation summary (Bagging decision tree)

Bagging with a decision tree base classifier was more effective compared to a single decision tree. Unlike bagged logistic regression, bagged decision tree improved the overall performance of the model in terms of AUROC as shown in Table 5.10. Comparison of specificity is invalid as in a single decision tree; it was impossible to set a specific sensitivity level. However, since the AUC improved, specificity is believed to be better in a bagged decision tree. (Galar, et al., 2011) stated that because of the accuracy-oriented design, ensemble algorithm directly applied to imbalanced datasets do not solve the problem that underlay in the base classifier by themselves, but the experimental study carried out proves otherwise. Bagging without any sampling method was able to deal with the imbalance in data and performed almost as well as bagged models with data balancing methods.

Figure 5.17 shows the roc curves for five different models. From this curve, it is observed that bagging with under-sampling has marginally higher overall performance compared to the other models. At the desired sensitivity of 0.95, all the bagged models performed better than a single model balanced with smote (best model in terms of AUC for a single classifier). In other words, bagging improved models performance.

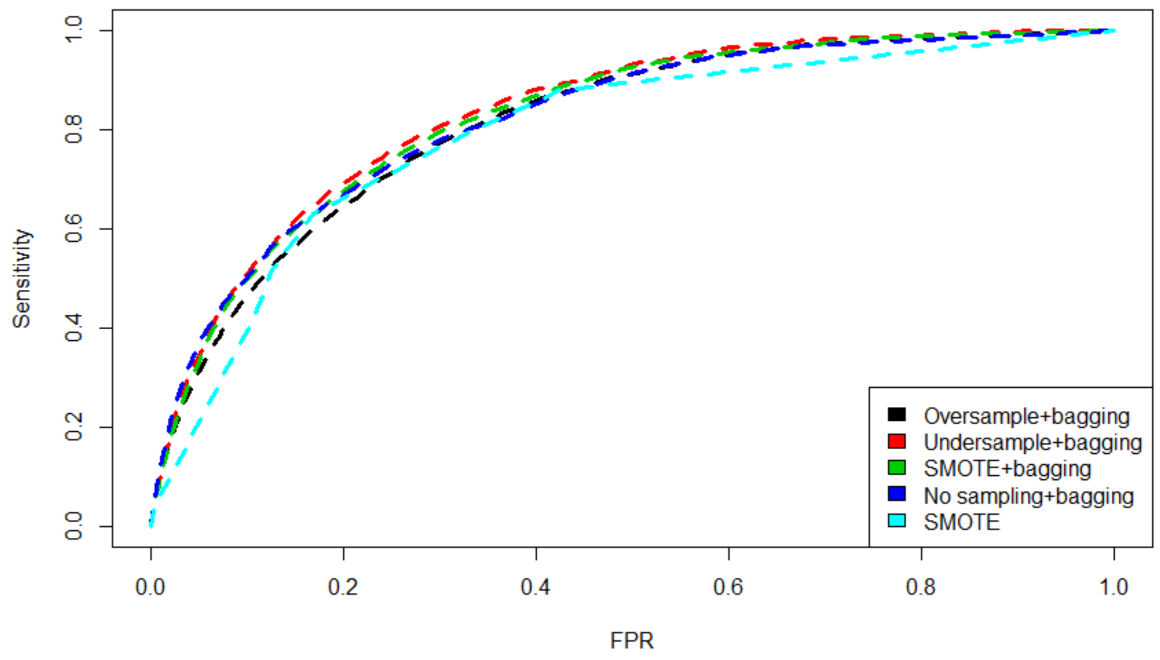


Figure 5. 17 Decision tree (ROC) Sampling methods and bagging

Table 5.18 shows the PR curve for five different models. It is observed that bagging improved the overall area under the precision-recall curve. A single model balanced with SMOTE was performing much lower compared to the bagged models. One interesting thing to note is that a bagged model with no sampling method performed better overall compared to bagged models with sampling. However, after a specificity of 0.8, bagged models with sampling methods had better performance. Since the desired recall(sensitivity) is 0.95, the Under-sample with bagging method would the best choice among these models. (Galar, et al., 2011) carried out different ensemble methods that deal with class imbalanced on a decision tree and SMOTEbagging obtained the best performance, but in this study, it was found that underbagging (undersampling with bagging) had the best performance. This could mean that no fixed data balancing method works well with a specific classifier; the data balancing method that is most suitable could be directly related to the complexity of the data. Another reason that could have lead to a more mediocre performance of SMOTEbagging could be due to the reason that this study only carried out SMOTE with a k value of 2. SMOTEbagging might have better performance with a different k value.

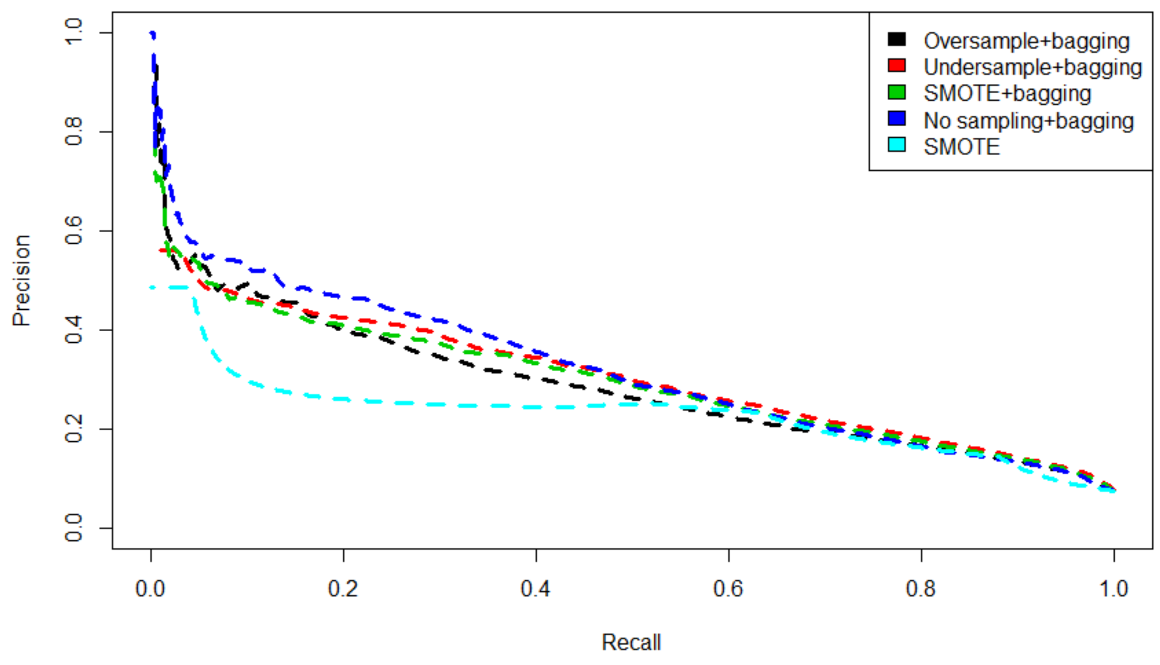


Figure 5. 18 Decision tree (PR) Sampling methods and bagging

5.5.2 Proposed decision tree (CART)

	Proposed (Over-sample)	Proposed (Under-sample)	Proposed (SMOTE)
Sensitivity	0.95	0.95	0.95
Specificity	0.46	0.46	0.42
AUROC	0.843	0.841	0.834
Precision	0.126	0.126	0.119
AUPR	0.311	0.304	0.292
Threshold	0.205	0.238	0.125

Table 5. 11 Model evaluation summary (Proposed method and sampling) decision tree

The proposed method involves a subset selection for each categorical variable and building a model based on this subset. As shown earlier that a single decision tree was unable to handle the class imbalance, this method is faced with the same problem. Data balancing methods have to be used to resample the data subsets. Table 5.11 shows the model evaluation summary of the proposed methods with three different data balancing methods. From this table, it is observed that at a sensitivity of 0.95, the proposed method integrated with the over-sampling method and the under-sampling method was able to perform better than bagged models in terms of specificity. The proposed method was also able to output a higher AUROC compared to bagged models.

Figure 5.19 shows the ROC curves for the proposed models as well as the best performing single model and best performing bagged models in terms of AUROC. From analysing the curves, it is understood that the proposed method is superior to a single classifier. When compared with a bagged model (under-sampling with bagging), the proposed method performed marginally better when integrated with the sampling methods. The major difference occurs at a specificity below 0.8 where the proposed method outperforms the underbagged model.

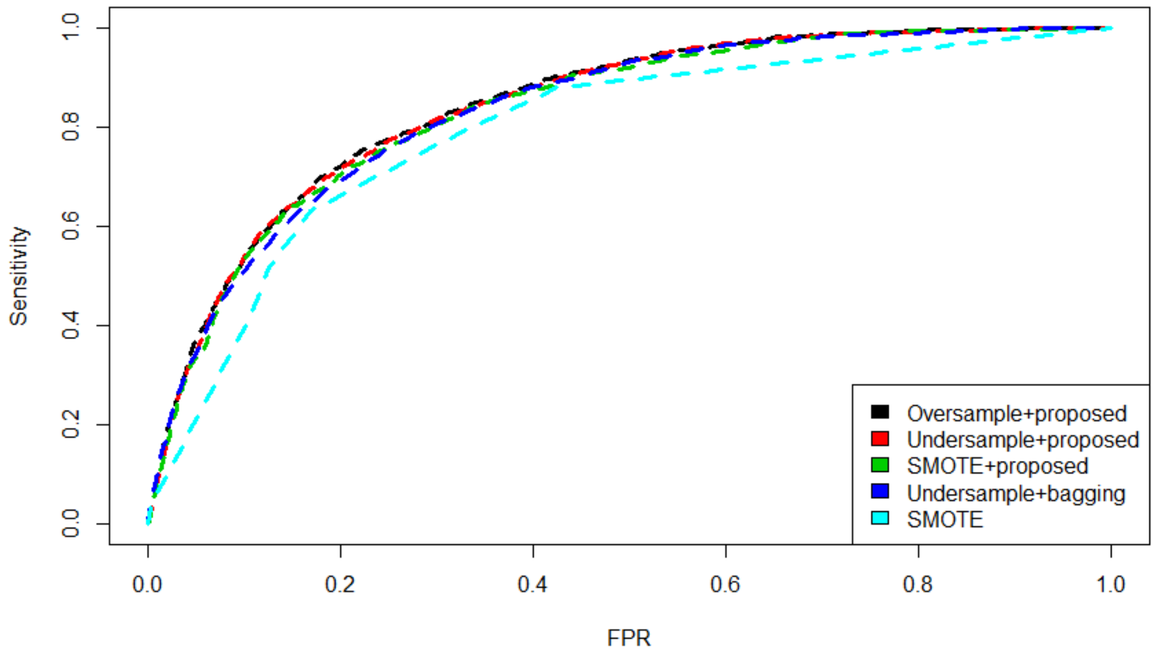


Figure 5. 19 Decision tree (ROC) Sampling methods and the proposed method

Figure 5.20 shows the PR curve for the proposed models as well as the best performing single model and best performing bagged model in terms of AUPR. From the figure, it is understood that the bagged method, as well as the proposed method are much superior compared to a single model balanced with SMOTE (best model for a single classifier). At a recall of less than 0.5 the bagging method performed significantly better compared to the other models. Above a recall of 0.5, the proposed method was much superior. At the desired recall of 0.95, both under-sampling and over-sampling based proposed method performed better compared to the other models.

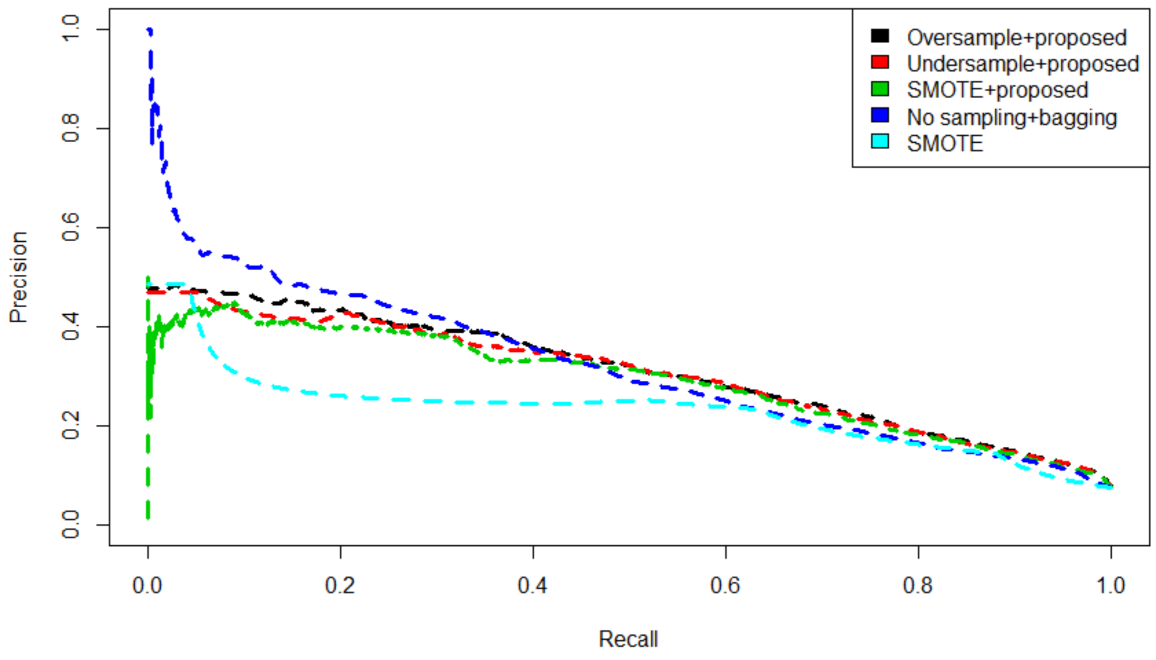


Figure 5. 20 Decision tree (PR) Sampling methods and the proposed method

From the experiments carried out, it is clear that bagging and the proposed method is more effective when dealing with imbalanced data as compared to a single decision tree with data balancing method.

CHAPTER 6
CONCLUSION AND RECOMMENDATION FOR FUTURE WORK

6.1 Conclusion

In this dissertation, we tried to solve the problem of imbalanced data by providing an effective ensemble framework and a series of related methods from different problem-solving angles. We compared the performance of Logistic regression and Decision tree (CART) classifiers with ensemble methods on the CRM dataset obtained from Ocean Holidays. As a comparison metric, we chose area under the receiver-operating characteristic curve (AUROC), the area under the precision-recall curve (AUPR), sensitivity, specificity and precision.

The primary motivation for this thesis, from a statistical point of view was to understand if an ensemble-based method will be a more effective way to deal with data imbalance problem. Based on the literature, we expected to see a more significant increase in prediction accuracy for the tree-based ensemble bagging methods than the models based on logistic regression, as decision trees have low bias but high variance. The experiment carried out has confirmed this. The models based on bagged decision trees performed better compared to a single tree. There were no significant improvements when bagging logistic regression models.

The study carried out proves that when a model is able to deal with the data imbalance in the dataset, no further improvements could be made to the model by data balancing methods. The logistic regression models in section 5.4 prove this finding. Bagging logistic regression models did not result in any improvements in model performance as well; this could be due to the reason that logistic regression is a stable classifier and minor changes in dataset do not affect such models. That means to say diversity among the bagged models is low, resulting in minor or negligible improvements. From these experiments, it can be concluded that the sampling method is not necessary if a classifier can handle the imbalanced data. Regarding bagging, the analysis is inconclusive as the logistic regression was able to handle the imbalanced data. In a situation where data is highly imbalanced bagging logistic regression with sampling method might prove useful. For the logistic regression classifier, the proposed method improved the specificity at a sensitivity of 0.95 and was able to perform marginally better than bagging based methods.

A decision tree (CART) was unable to handle the data imbalance. When integrated with data balancing methods, this model was able to output satisfactory results. Furthermore, it was noted that SMOTE based data balancing was more effective compared to ROS and RUS when used alongside a decision tree. SMOTE is known to be useful when the imbalance ratio is high, but in this study with slightly imbalance data, SMOTE outperformed both over-sampling and under-sampling based methods. A bagging decision tree increased the overall performance of the model compared to a single decision tree, and it was able to perform marginally better than a logistic regression model. One interesting finding was that a bagged decision tree without any data balancing method was able to perform well despite a single decision tree being unable to deal with the data imbalance. Furthermore, the bagged decision tree without data balancing method was able to perform almost as well as bagged decision trees with balancing methods in terms of AUROC and performed the best in terms of AUPR.

In the experiments carried out using the proposed method on a decision tree, we observed that this method was able to perform marginally better than bagged decision trees. The proposed method with over-sampling on a decision tree has the best performance among all the models. At a 0.95 sensitivity, this model was able to obtain the highest specificity of 0.46 as well as the highest AUROC among all the models.

The proposed method was effective when applied to this particular dataset; this might not be the case for all datasets. There is potential in using this method to carry out predictions even though it is time-consuming. In this thesis, the full effect of this method (only subsets from 13 categorical variable were used) was not explored due to the time limitation of the project.

The main contributions of this thesis can be summarised as follows:

- **When dealing with an imbalanced dataset, typical data balancing methods are required only when the algorithm is unable to handle the imbalance in data.**
  The finding from the experimental study presented in Chapter 5 with a logistic regression model proves this finding. The logistic regression model was able to deal with the imbalance in the dataset, applying data balancing methods such as RUS, ROS and SMOTE resulted in a minor reduction in the overall performance of the model. This conclusion was arrived purely based on analysing precision-recall curve as well as a ROC curve.

- **Bagging is ineffective when used with logistic regression base classifier.**
  Bagging did not result in any significant improvements when used with a logistic regression model. This is due to the reason that stable classifiers such as logistic regression only have minor effects on small changes in data that is introduced by the bootstrap method. For bagging to be effective, diversity is the key.

- **A data balancing method is not necessarily required for a bagged model when dealing with imbalanced data.**
  The finding from the experimental study presented in chapter 5 with a bagged CART model proves this finding. A single CART model was unable to hand the data imbalance, but the bagged CART model was able to handle this data imbalance reasonably well and was able to produce impressive results. The bagged model with no data balancing methods even outperformed models utilising ROS, RUS and SMOTE in terms of AUPR. This means that bagging could directly solve data imbalance problem unless the data is highly imbalanced.

- **A new method was introduced which was able to outperform bagging when used with a decision tree.**
  The new method is based on a subset selection from each variable level. This method was able to provide impressive results even though it was not tested to its full potential. This method was tested with two different classifiers. The first test was carried out on a logistic regression model which showed minor improvements compared to bagging. This is due to the same reason as to why a bagged logistic regression is ineffective. The next test was carried out using a CART model. In this test, some impressive test results were obtained with the proposed method. The model built using the proposed method with over-sampling outperformed all the other models discussed in Chapter 5 in terms of AUROC. Furthermore, this model was able to perform better than other models with a recall above 0.5; this is observed from Figure 5.20.

6.2 Future works

From the experiments carried out, logistic regression based method was ineffective when used alongside bagging methods. It would be ideal to explore logistic regression with boosting. Boosting is a method that has shown to work well with logistic regression models in many works of literature. The proposed data subset method can be integrated with a boosting method to improve the overall accuracy of the model. The idea would be to create multiple boosted logistic regression models based on subsets and then combine the predictions of each boosted model by averaging. However, this could be computationally expensive.

CART based decision tree performed well when used alongside bagging and was able to perform even better when used with the proposed method. It would be interesting to combine these two methods and carry out predictions. Each data subset should be used to create bagged models, and the predictions from all the bagged models could be averaged to carry out the final prediction. This could lead to further improvements in the performance of a model. The proposed method should be studied in depth using multiple datasets as well as with different models to understand if this method can be potentially beneficial in predictive modelling.

The effect of increasing the number of bootstrapped samples was not explored in this paper. All bagging based models were created using 100 bootstrapped samples. Methods to obtain the best number of bootstraps should be explored. Furthermore, a study could be carried out to understand the effects of increasing the number of bootstrap samples in a bagged model with different data balancing methods. The sampling method SMOTE was not explored with its full potential. All experiments were carried out using a fixed k value of 2; SMOTE might be able to perform well with varying *k* values. Determining the optimal *k* value should also be studied.

REFERENCES

Alirezaee, M., Dehzangi, A. & Mansoori, E., 2012. Ensemble of Neural networks to solve class imbalance problem of protein secondary structure prediction. *International Journal of Artificial Intelligence & Application,* pp. 9-20.

BÃlaszczy´nski, J., Stefanowski, J. & Szajek, M., 2013. Local Neighbourhood in Generalizing Bagging for imbalanced Data. *COPEM ECML-PKKD. Workshop Proceedings.*

Barandela, R., Sa´nchez, J. & Valdovinos, R., 2003. New Applications of Ensembles of Classifiers. *Pattern Anal app. vol 6,* pp. 245-256.

Batista, G. E., C.Prati, R. & Monard, M. C., 2004. *A study of the behavior of several method for balancing machine learning training data, SIGKDD Expl. Newslett., vol. 6,* pp. 20-29.

Batista, G. E., Prati, R. C. & Monard, M. C., 2005. *Balancing strategies and class overlapping. Advances in Intelligent Data Analysis,* pp. 24-35.

Błaszczynski, J. & Stefanowski, J., 2015. Neighbourhood Sampling in Bagging for Imbalanced Data. *Neurocomputing,* pp. 529-542.

Breiman, L., 1996. *Bagging Predictors, Department of Statistics, University of California.*

Breiman, L., 2001. *Random Forests ,Machine learning,* pp. 5-32.

Brown, G., L.Wyatt, J. & Tino, P., 2005. Managing Diveristy in Regression Ensembles. *Journal of machine learning research 6,* pp. 1621-1650.

Chawla, N. V., Japkowicz, N. & Kolcz, A., 2004. *Editorial Special Issue on Learning from Imbalance Data sets, ACM SIGKDD Explorations Newsletter,* pp. 1-6.

Chawla, N. V., Lazarevic, A., Hall, L. O. & Bowyer, K., 2004. SMOTEBoost: Improving prediction of the minority class in Boosting. *Business Analytic Solutions, Canadian Imperial Bank of Commerce (CIBC).*

Chawla, N. V. & Sylvester, J., 2007. Exploiting Diversity in Ensembles: Improving the performance on unabalanced datasets. *Multiple Classifier Systems,* pp. 397-406.

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

Chen, H., 2008. Diveristy and Regulation in Neural Network Ensembles. *PhD thesis, School of Computer Science, University of Birmingham.*

Chujai, P. & Kerdprasop, N., 2015. Ensemble Learning for Imbalanced Data Classification Problem. *Proccedings of the 3rd International Conference on Industrial Application Engineering 2015,* pp. 449-456.

Cioca, M., Cioranu, C. & Gîfu, D., 2012. Computational techniques in management of engineering and business institutions. *International Conference on Engineering & Business Education, Innovation and Entrepreneurship,* pp. 462-472.

Debik, J. B., 2017. Using Ensemble Methods to improve the performance of Prediction. *Statistical Analysis of a myocardial infarction Data Set from the HUNT Study.*

Díez-Pastor, J. F., Rodríguez, J. J., García-Osorio, C. I. & Kuncheva, L. I., 2015. Diversity techniques improve the performance of the best. *Information Sciences,* pp. 98-117.

Domingos, P., 2000. A unified Bias-Variance Decompostion for Zero-One and Squared Loss. *17th National Conference on Artificial Inteligence,* pp. 564-569.

Duque, P. F. J., Fernández, G. M. ,., Troncoso, A. & Martínez, Á. F., 2016. A new methodology based on imbalanced classification for predicting outliers in electricity demand time series. *Article Energies,* pp. 1-10.

Elkan, C., 2013. Predictive analysis and data mining. 28 May.

EstaBrooks, A., Jo, T. & Japkowicz, N., 2004. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence 20, volume 20,* pp. 18-36.

Fawcett, T., 2003. ROC Graphs: Notes and Practical Considerations for Data Mining Researches. *HP Labs Palo Alto, CA, Technical Report* .

Freund, Y. & Schapire, R. E., 1996. Experimens with a new Boosting Algorithm. *In Proc. of the 13th. Int. Conf. on Machine Learning,* pp. 148-156.

Galar, M., Fernandez, A., Barrenechea, E. & Bustince, H., 2011. A Review on Ensembles for Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART C: APPLICATIONS AND REVIEWS,* pp. 1-22.

Ganganwar, V., 2012. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering,* pp. 2250-2259.

Garcia-Pedrajas, N., Hervas-Martinez, C. & Ortiz-Boyer, D., 2005. Cooperative coevolution of artifical neural network ensembles for pattern classification. *IEEE Transations of Evolutionary Computaion,* pp. 271-302.

Geman, S., Bienenstock, E. & Doursat, R., 1992. Neural Networks and Bias/Variance Dilemma. pp. 1-58.

Gončarovs, P., 2017. Data Analytics in CRM processes. *Information Technology and Management Science,* December, Volume 20, pp. 103-108.

Gutierrez, D., 2014. *Classes of predictive Analytics.* [Online] Available at: https://insidebigdata.com/2014/09/18/classes-predictive-analytics/

Hakim, L., Sartono, B. & Saefuddin, A., 2017. Bagging Based Ensemble Classification mEthod on Imbalance Datasets. *IJCSN-Internatonal journal of Computer Science and Network, vol 6,* pp. 670-676.

Han, H., Wang, W.-Y. & Mao, B.-H., 2005. Borderline-SMOTE: A NEW Over-Sampling Method in Imbalanced Data Sets Learning. *Advances in Intellingent Computing,* pp. 878-887.

Hanifah, F. S., Wijayanto, H. & Kurnia, A., 2015. *SMOTEBagging Algorithm for Imbalanced Dataset in Logistic Regression Analaysis, Applied Mathematical Sciences,* pp. 6857-6865.

Hanifah, F. S., Wijayanto, H. & Kurnia, A., 2015. SMOTE Bagging Algorithm for Imbalanced dataset in Logistic regression Analysis. *Applied Mathematical Sciences, Vol. 9,,* pp. 6857-6865.

He, H. & Garcia, E. A., 2009. Learning from Imbalanced Data. *IEEE Transactions on knowledge and data Engineering,* pp. 1263-1284.

Hulse, J. V., Khoshgoftaar, T. M. & Napolitano, A., 2007. Experimental Perspectives on Learning from Imbalanced Data. *ICML '07: Proceedings of the 24th international conference on Machine learning,* pp. 935-942.

Hu, S., Liang, Y., Ma, L. & He, Y., 2009. MSMOTE: Improving classification performance when training data is imbalanced. *Second international workshop on Computer Science and Engineering,* pp. 13-17.

Japkowicz, N., 2000. Learning from Imabalanced Data sets: A comparision of Various Strategies. *AAAI Workshop on Learning from Imbalanced Data Sets,* pp. 10-15.

Joshi, M. V., 2002. On evaluation performance of classifiers for rare classes. *IEEE international Conference on Data Mining,* pp. 641-661.

Joshi, N. & Srivastava, S., 2014. Improving Classification Accuracy Using Ensemble Learning Technique (Using Different Decision Trees). *International Joural of Computer Science and Mobile Computing,* pp. 727-732.

Jo, T. & Japkowicz, N., 2005. Class Imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter, volume 6,* pp. 40-49.

Khoshgoftaar, T., Fazelpour, A., Dittman, D. & Napolitano, A., 2015. Ensemble vs. Data Sampling: Which Option Is Best Suited to Improve Classification Performance of Imbalanced Bioinformatics Data?. pp. 705-712.

Kim, Y. S., 2004. Toward a successful CRM: Variable selection, Sampling and ensemble. *Business Information Systems, Utah State University.*

King, G. & Zeng, L., 2001. Logistic regression in Rare events data. *Political Analysis,* pp. 137-163.

Krawczyk, B., Wozniak, M. & Schaefer, G., 2014. Cost-sensitive decision tree ensemble for effective imbalanced classification. *Applied soft computing,* pp. 554-562.

Krieger, A., Long, C. & Wyner, A., 2002. Boosting Noisy Data.

Kuncheva, L., 2014. *Combining Pattern Classifiers : Methods and Algorithms, Second Edition.*

Kuncheva, L., 2014. *Combining Pattern Classifiers : Methods and Algorithms, Second Edition.*

Kuncheva, L. I. & Whitaker, C. J., 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Pattern Analysis and Applications,* pp. 22-31.

Laurikkala, J., 2001. Improving Identificaiton of Difficult Small classes by Balancing class distribution. *Proc. Conf. AI in Medicine in Europe: Artificial Intelligence Medicine,* pp. 63-66.

Li, C., 2007. Classifying imbalanced data using a bagging ensemble variation (BEV). *Conference: Proceeding of the 45th Annual Southeast Regional Conference.*

Ling, C. X., Huang, J. & Zhang, H., 2003. AUC: a Statistically Consisent and more Discriminating Measure than Accuracy. *In Proccding of 18th International Conference on Artificial Intelligence,* pp. 329-341.

Liu, X.-Y., Wu, J. & Zhou, Z.-H., 2009. Exploratory Under-sampling for Class-Imbalance Learning. *IEEE TRANSACTIONS ON SYSTEMS,MAN AND CYBERNETICS,* pp. 539-550.

Maloof, M. A., 2003. Learning when data sets are imbalanced and when costs are unequal and unknown. *ICML Workshop on Learning from Imbalanced Data Sets II.*

Morelli, T., Shearer, C. & Buecker, A., 2010. IBM SPSS predictive analytics: Optimizing decisions at the point of impact. *Redguides for Business Leaders,* pp. 1-58.

Ngai, E., Xiu, L. & Chau, D., 2009. Application od data mining tecniques in customer relaionship management: A literature review and classification. *Expert systems and application,* pp. 2592-2602.

Opitz, D. & Maclin, R., 2006. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research.*

Oza, N. C. & Tumer, K., 2008. Classifier Ensembles: Select Real-World Applications. *Inf. Fusion, vol. 9, no. 1,* pp. 4-20.

Pandey, M. & Taruna, S., 2014. A comparative study of ensemble methods for students' performance Modeling. *International Journal of Computer Applications,* 103(8), pp. 27-32.

Parvin, H., Alinejad-Rokny, H. & Pravin, S., 2013. A classifier Ensemble of Binary Classifier Ensembles. *International Journal of Learning Management Systems,* pp. 37-43.

Polikar, R., 2007. Bootstap- inspired techiques in computational intelligence: ensemble of classifiers, incremental learning, data fusion and missing features. *IEEE signal processing magazine,* pp. 59-72.

Powers, D. M. W., 2011. Evaluation: From Precision, recall and F-measures to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies,* pp. 37-63.

Pramanik, S., Pram, B. K., Chowdhury, U. N. & Huda, N., 2010. A comparative study of bagging, boosting and C4.5: The recent improvements in Decision Tree Learning Algorithm. pp. 300-306.

Raghupathi, W. & Raju, L. L. S. B. S., 2010. A neural network application for bankruptcy prediction. pp. 147-155.

Raskutti, B. & Kowalczyk, A., 2004. Extreme Re-balancing for SVMs: a case study. *SIGKDD Explorations,* pp. 60-69.

Riksbergen, C. J. v., 1979. Information Retrieval. *Department of Computing Science.*

Seiffert, C., M.Khoshgoftaar, T., Hulse, J. V. & Napolitano, A., 2010. RUSBoost: A Hybrid Apprach to Alleviating class imbalance. *IEEE Transactions on systems, Man and Cybernetics- Part A: Systems and Humans Vol 40,* pp. 185-197.

Sharma, A., Gandhi, A. & Kumar, A., 2016. Prediction of CRM using regression modelling. *International journal of advanced engineering and global technology,* 4(3).

Signorini, D. F. et al., 1999. Predicting survival using simple clinical variables: a case study in traumatic brain injury. *Journal of Neurology, Neurosurgery & Psychiatry* , pp. 20-25.

Smola, A. & Vishwanathan, S., 2008. *Introduction to machine learning.* Cambridge: Cambridge University press.

Teo, T., Devadoss, p. & Pan, S., 2006. Towards a holistic perspective of customer relationship management implementation: A case stud of the housing and development board, Singapore. *Decision Support Systems 42,* pp. 1613-1627.

Ting, K. M., 2002. An instance-Weighting Method to induce Cost-Sensitive Trees. *IEEE Transations on knowledge and data engineering,* pp. 659-665.

Trifonov, R., Gotseva, D. & Angelov, V., 2017. Binary classification algorithms. *International journal of development research,* November, 07(11), pp. 16873-16879.

Visa, S. & Ralescu, A., 2005. Issues in Mining Imbalanced Data Sets - A review paper. *In proceeding of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference,* pp. 67-73.

Wang, H., Xu, Q. & Zhou, L., 2015. *Large Unbalanced Credit Scoring Using Lasso-Logistic Regression ensemble.* [Online] Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0117844

Wang, S., 2011. *ENSEMBLE DIVERSITY FOR CLASS IMBALANCE LEARNING.*

Wang, S. & Yao, X., 2009. *Diversity Analysis on Imbalanced Data Sets by Using Ensemble Models, IEEE Symp.Comput.Intell.Data Mining,* pp. 324-331.

Wan, S. & Yang, H., 2013. Comparison among methods of ensemble learning. *International Sympsium of Biometrics and security technologies,* pp. 286-290.

Zadrozny, B. & Elkan, C., 2001. Learning and Making Decisions When costs and Probabilities are Both Unknown. *KDD ’01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 204-213.

Zhou, L. & Lai, K. K., 2009. Benchmarking binary classification models on dataset with different degree of imbalance. *Department of Management Sciences,* pp. 205-216.

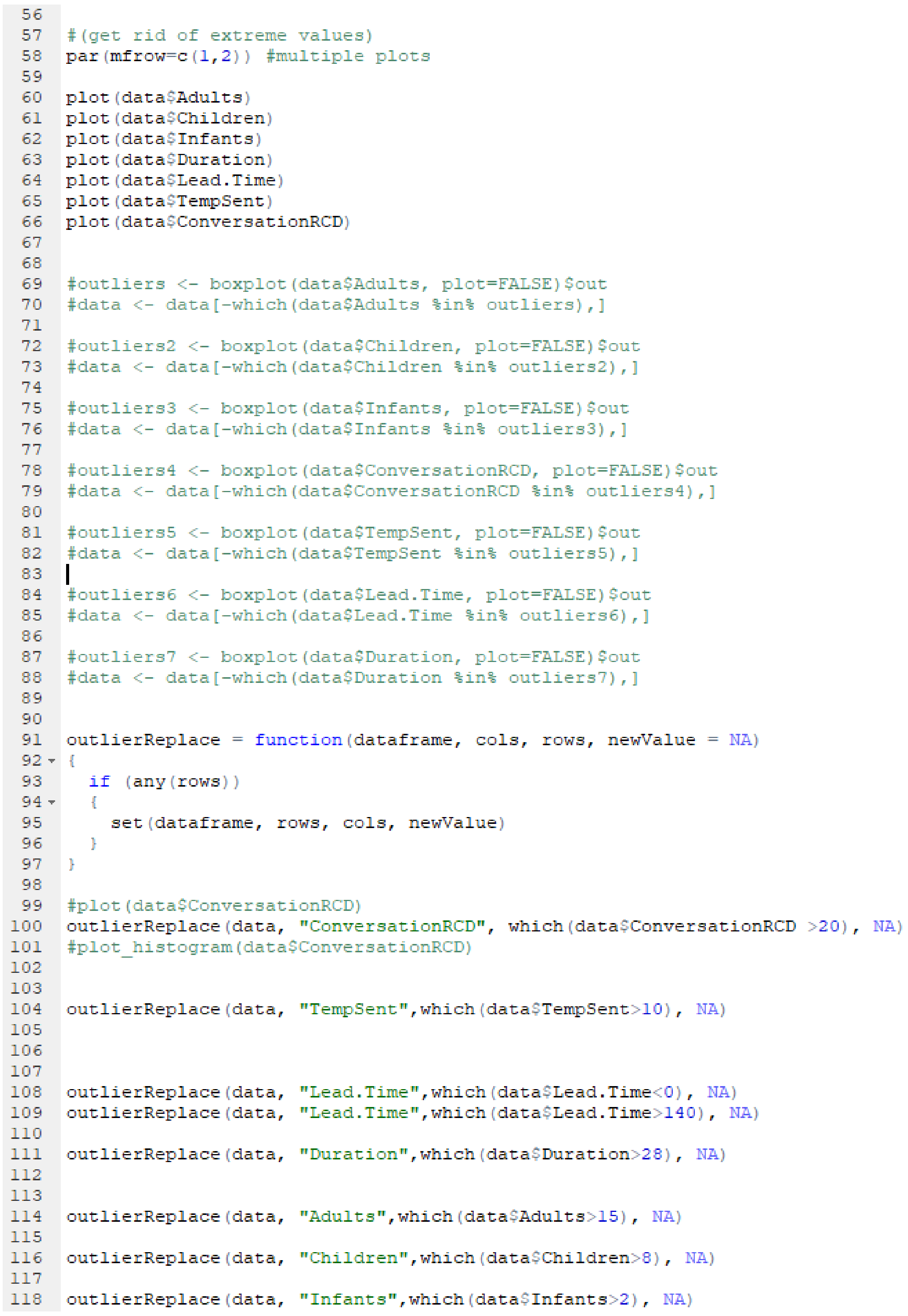
Zhou, Z. H., 2012. *Ensemble Methods Foundation and Algorithm , Machine learning & Pattern recognition series.* Boca Raton: CRC press.

Zhou, Z.-H. & Liu, X.-Y., 2006. Tranining Cost-Sensitive Neural Networks with Methods addressing the class imbalance problem. *IEEE Transactions on knowledge and data engineering vol 18,* pp. 63-77.

APPENDIX

Data preparation

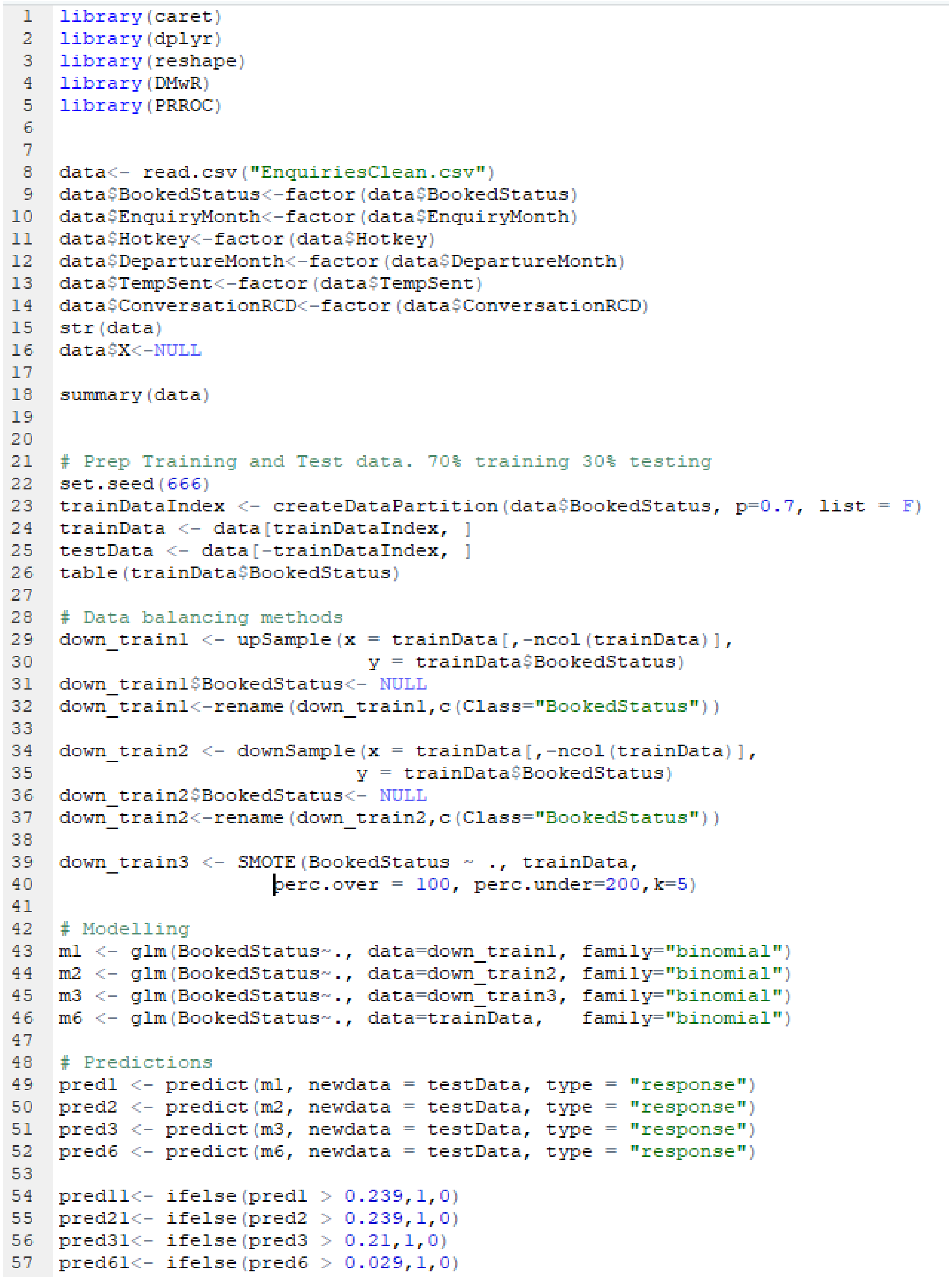
```r
library(lubridate)
library(data.table)
library(DataExplorer)
library(caret)

data<- read.csv("EnquiryTotal2.csv")
#Allocated.Time, not relevant to this analysis
data<-subset(data, Allocated.Time=="ExtremelyFast"| Allocated.Time=="Fast")
data$Allocated.Time<- NULL
summary(data)
str(data)

  ###################EDA#########################

# Change Enquiry.Time to categories
data$Enquiry.Time <- as.numeric(gsub("\\:.*$", "", data$Enquiry.Time))
data$Enquiry.Timecat<-ifelse(data$Enquiry.Time>=9 &
                             data$Enquiry.Time<=21,"Business_Hour","Closed")
data$Enquiry.Timecat<-factor(data$Enquiry.Timecat)

data$Enquiry.Time_class <- with(data,
                                ifelse(Enquiry.Time >= 6 &
                                       Enquiry.Time<=12, "morning",
                                ifelse(Enquiry.Time>12 &
                                       Enquiry.Time<=18, "afternoon", "night")))
data$Enquiry.Time<- NULL
data$Enquiry.Time_class<-factor(data$Enquiry.Time_class)

# Change Enquiry.Date to categories

yq <- as.yearqtr(as.yearmon(data$Enquiry.Date, "%m/%d/%Y") + 1/12)
data$EnquirySeason <- factor(format(yq, "%q"), levels = 1:4,
                    labels = c("winter", "spring", "summer", "fall"))




data$Enquiry.Date<- month(as.POSIXlt(data$Enquiry.Date, format="%m/%d/%Y"))
data$Enquiry.Date<-factor(data$Enquiry.Date)
colnames(data)[colnames(data)=="Enquiry.Date"] <- "EnquiryMonth"

# Change Departure.Date to categories

yq2 <- as.yearqtr(as.yearmon(data$Dep.Date, "%m/%d/%Y") + 1/12)
data$DepartureSeason <- factor(format(yq2, "%q"), levels = 1:4,
                             labels = c("winter", "spring", "summer", "fall"))


data$Dep.Date<- month(as.POSIXlt(data$Dep.Date, format="%m/%d/%Y"))
data$Dep.Date<-factor(data$Dep.Date)
colnames(data)[colnames(data)=="Dep.Date"] <- "DepartureMonth"

# Hotkey missing values change to 0 and 1 for hotkey
data$Hotkey<-ifelse(data$Hotkey %in% 'Hot Key',"1","0")
data$Hotkey<-factor(data$Hotkey)
```



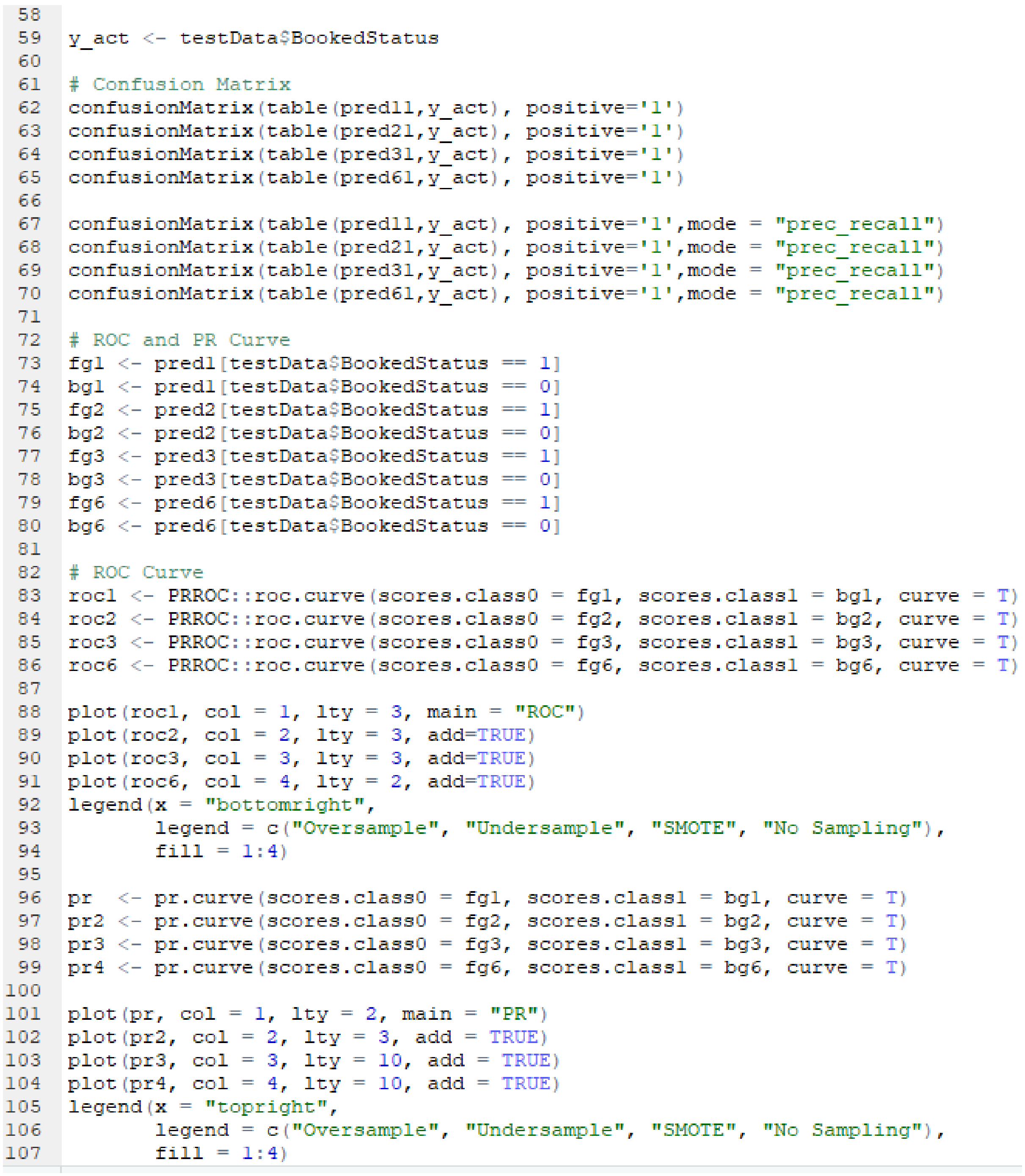
```r

#(get rid of extreme values)
par(mfrow=c(1,2)) #multiple plots

plot(data$Adults)
plot(data$Children)
plot(data$Infants)
plot(data$Duration)
plot(data$Lead.Time)
plot(data$TempSent)
plot(data$ConversationRCD)


#outliers <- boxplot(data$Adults, plot=FALSE)$out
#data <- data[-which(data$Adults %in% outliers),]

#outliers2 <- boxplot(data$Children, plot=FALSE)$out
#data <- data[-which(data$Children %in% outliers2),]

#outliers3 <- boxplot(data$Infants, plot=FALSE)$out
#data <- data[-which(data$Infants %in% outliers3),]

#outliers4 <- boxplot(data$ConversationRCD, plot=FALSE)$out
#data <- data[-which(data$ConversationRCD %in% outliers4),]

#outliers5 <- boxplot(data$TempSent, plot=FALSE)$out
#data <- data[-which(data$TempSent %in% outliers5),]

#outliers6 <- boxplot(data$Lead.Time, plot=FALSE)$out
#data <- data[-which(data$Lead.Time %in% outliers6),]

#outliers7 <- boxplot(data$Duration, plot=FALSE)$out
#data <- data[-which(data$Duration %in% outliers7),]


outlierReplace = function(dataframe, cols, rows, newValue = NA)
{
  if (any(rows))
  {
    set(dataframe, rows, cols, newValue)
  }
}

#plot(data$ConversationRCD)
outlierReplace(data, "ConversationRCD", which(data$ConversationRCD >20), NA)
#plot_histogram(data$ConversationRCD)


outlierReplace(data, "TempSent",which(data$TempSent>10), NA)



outlierReplace(data, "Lead.Time",which(data$Lead.Time<0), NA)
outlierReplace(data, "Lead.Time",which(data$Lead.Time>140), NA)

outlierReplace(data, "Duration",which(data$Duration>28), NA)


outlierReplace(data, "Adults",which(data$Adults>15), NA)

outlierReplace(data, "Children",which(data$Children>8), NA)

outlierReplace(data, "Infants",which(data$Infants>2), NA)
```

```
# Holiday type
plot_bar(data$Holiday.Type)
data<-group_category(data=data,feature = "Holiday.Type",threshold=0.05,update=TRUE)
data$Holiday.Type<-factor(data$Holiday.Type)
str(data)

# Transport type
plot_bar(data$Transport.Type)
data$Transport.Type <- with(data,
ifelse(Transport.Type %in% "Return transfers",
                           "Return transfers",
ifelse(Transport.Type %in% "Fully comp car hire",
                           "Fully comp car hire",
                           "None Required")))
data$Transport.Type<- factor(data$Transport.Type)

# Destination/Departure Airport (Group rare cases together)

plot(data$Destination)
summary(data$Destination)

data<-group_category(data=data, feature = "Destination", threshold=0.05, update=TRUE)
data$Destination<- factor(data$Destination)
str(data)

#plot_bar(data$Dep.Airport)
data<-group_category(data=data, feature = "Dep.Airport", threshold=0.05,update=TRUE)
data$Dep.Airport<-factor(data$Dep.Airport)


# Accommodation type
#plot_bar(data$Accom.type)
data$Accom.type <- with(data,
                  ifelse(Accom.type %in% "Hotel","Hotel",
                   ifelse(Accom.type %in% "Villa", "Villa",
                    ifelse(Accom.type %in% "TownHome", "Apartment",
                      ifelse(Accom.type %in% "Combined","Villa",
                        ifelse(Accom.type %in% "Apartment","Apartment","None"))))))
data$Accom.type<- factor(data$Accom.type)

data$Title<- with(data, ifelse(Title %in% "Dr","M",
                          ifelse(Title %in% "Mr","M",
                            ifelse(Title %in% "Ms","F",
                              ifelse(Title %in% "Mrs","F","F")))))

data$Title<-factor(data$Title)

data$Booked.Status<-with(data,ifelse(Booked.Status %in% "YES","1","0"))
data$BookedStatus<-factor(data$Booked.Status)
data$Booked.Status<- NULL
data<-na.omit(data)

plot_bar(data$Destination)

summary(data)
str(data)

write.csv(data,file='EnquiriesClean.csv')
```

Logistic regression (ROS, RUS, SMOTE, No sampling)

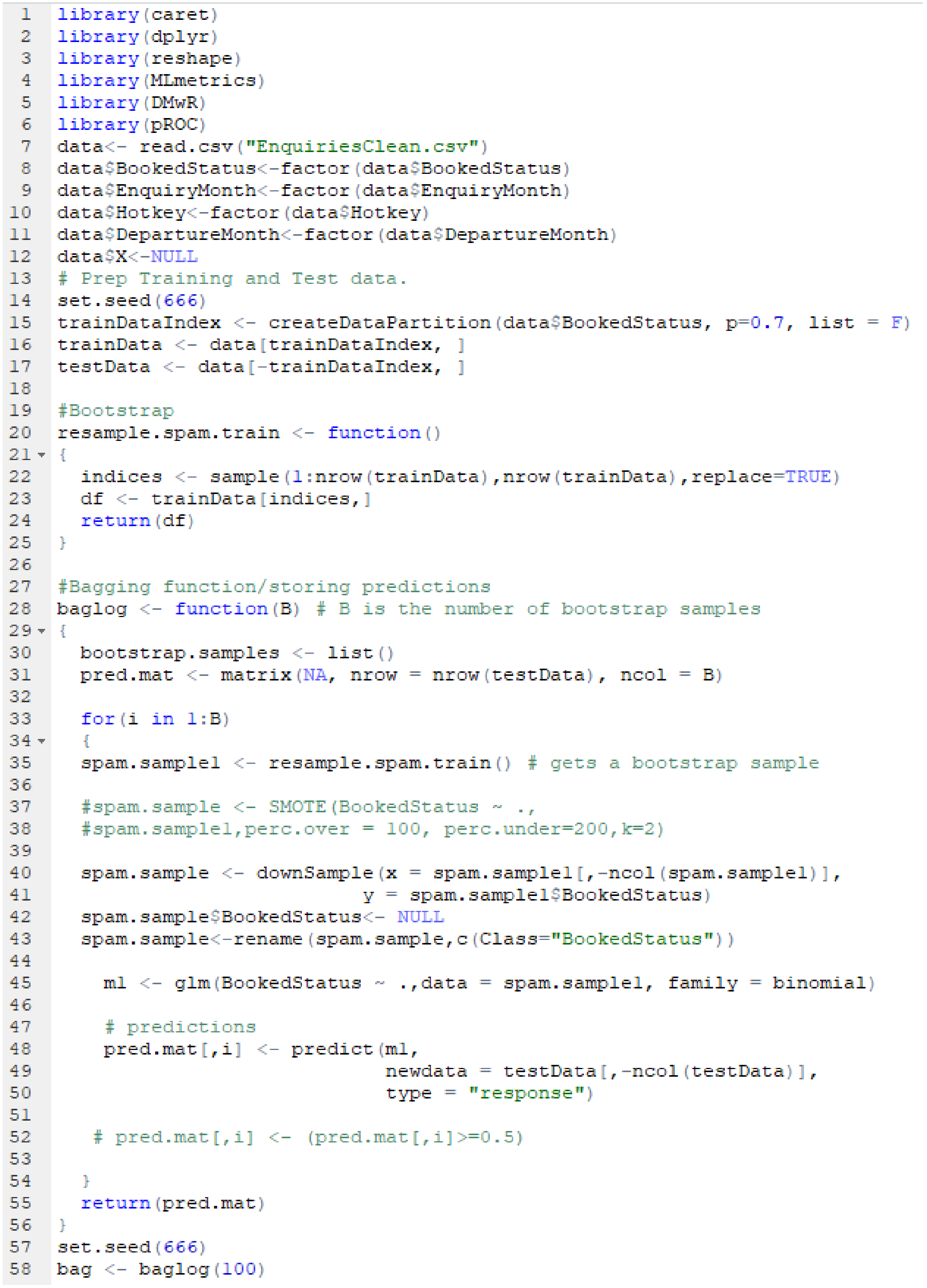


```r
library(caret)
library(dplyr)
library(reshape)
library(DMwR)
library(PRROC)


data<- read.csv("EnquiriesClean.csv")
data$BookedStatus<-factor(data$BookedStatus)
data$EnquiryMonth<-factor(data$EnquiryMonth)
data$Hotkey<-factor(data$Hotkey)
data$DepartureMonth<-factor(data$DepartureMonth)
data$TempSent<-factor(data$TempSent)
data$ConversationRCD<-factor(data$ConversationRCD)
str(data)
data$X<-NULL

summary(data)


# Prep Training and Test data. 70% training 30% testing
set.seed(666)
trainDataIndex <- createDataPartition(data$BookedStatus, p=0.7, list = F)
trainData <- data[trainDataIndex, ]
testData <- data[-trainDataIndex, ]
table(trainData$BookedStatus)

# Data balancing methods
down_train1 <- upSample(x = trainData[,-ncol(trainData)],
                          y = trainData$BookedStatus)
down_train1$BookedStatus<- NULL
down_train1<-rename(down_train1,c(Class="BookedStatus"))

down_train2 <- downSample(x = trainData[,-ncol(trainData)],
                         y = trainData$BookedStatus)
down_train2$BookedStatus<- NULL
down_train2<-rename(down_train2,c(Class="BookedStatus"))

down_train3 <- SMOTE(BookedStatus ~ ., trainData,
                  perc.over = 100, perc.under=200,k=5)

# Modelling
m1 <- glm(BookedStatus~., data=down_train1, family="binomial")
m2 <- glm(BookedStatus~., data=down_train2, family="binomial")
m3 <- glm(BookedStatus~., data=down_train3, family="binomial")
m6 <- glm(BookedStatus~., data=trainData,   family="binomial")

# Predictions
pred1 <- predict(m1, newdata = testData, type = "response")
pred2 <- predict(m2, newdata = testData, type = "response")
pred3 <- predict(m3, newdata = testData, type = "response")
pred6 <- predict(m6, newdata = testData, type = "response")

pred11<- ifelse(pred1 > 0.239,1,0)
pred21<- ifelse(pred2 > 0.239,1,0)
pred31<- ifelse(pred3 > 0.21,1,0)
pred61<- ifelse(pred6 > 0.029,1,0)
```



```r

y_act <- testData$BookedStatus

# Confusion Matrix
confusionMatrix(table(pred11,y_act), positive='1')
confusionMatrix(table(pred21,y_act), positive='1')
confusionMatrix(table(pred31,y_act), positive='1')
confusionMatrix(table(pred61,y_act), positive='1')

confusionMatrix(table(pred11,y_act), positive='1',mode = "prec_recall")
confusionMatrix(table(pred21,y_act), positive='1',mode = "prec_recall")
confusionMatrix(table(pred31,y_act), positive='1',mode = "prec_recall")
confusionMatrix(table(pred61,y_act), positive='1',mode = "prec_recall")

# ROC and PR Curve
fg1 <- pred1[testData$BookedStatus == 1]
bg1 <- pred1[testData$BookedStatus == 0]
fg2 <- pred2[testData$BookedStatus == 1]
bg2 <- pred2[testData$BookedStatus == 0]
fg3 <- pred3[testData$BookedStatus == 1]
bg3 <- pred3[testData$BookedStatus == 0]
fg6 <- pred6[testData$BookedStatus == 1]
bg6 <- pred6[testData$BookedStatus == 0]

# ROC Curve
roc1 <- PRROC::roc.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
roc2 <- PRROC::roc.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
roc3 <- PRROC::roc.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
roc6 <- PRROC::roc.curve(scores.class0 = fg6, scores.class1 = bg6, curve = T)

plot(roc1, col = 1, lty = 3, main = "ROC")
plot(roc2, col = 2, lty = 3, add=TRUE)
plot(roc3, col = 3, lty = 3, add=TRUE)
plot(roc6, col = 4, lty = 2, add=TRUE)
legend(x = "bottomright",
       legend = c("Oversample", "Undersample", "SMOTE", "No Sampling"),
       fill = 1:4)

pr  <- pr.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
pr2 <- pr.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
pr3 <- pr.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
pr4 <- pr.curve(scores.class0 = fg6, scores.class1 = bg6, curve = T)

plot(pr, col = 1, lty = 2, main = "PR")
plot(pr2, col = 2, lty = 3, add = TRUE)
plot(pr3, col = 3, lty = 10, add = TRUE)
plot(pr4, col = 4, lty = 10, add = TRUE)
legend(x = "topright",
       legend = c("Oversample", "Undersample", "SMOTE", "No Sampling"),
       fill = 1:4)
```

Logistic regression Bagging

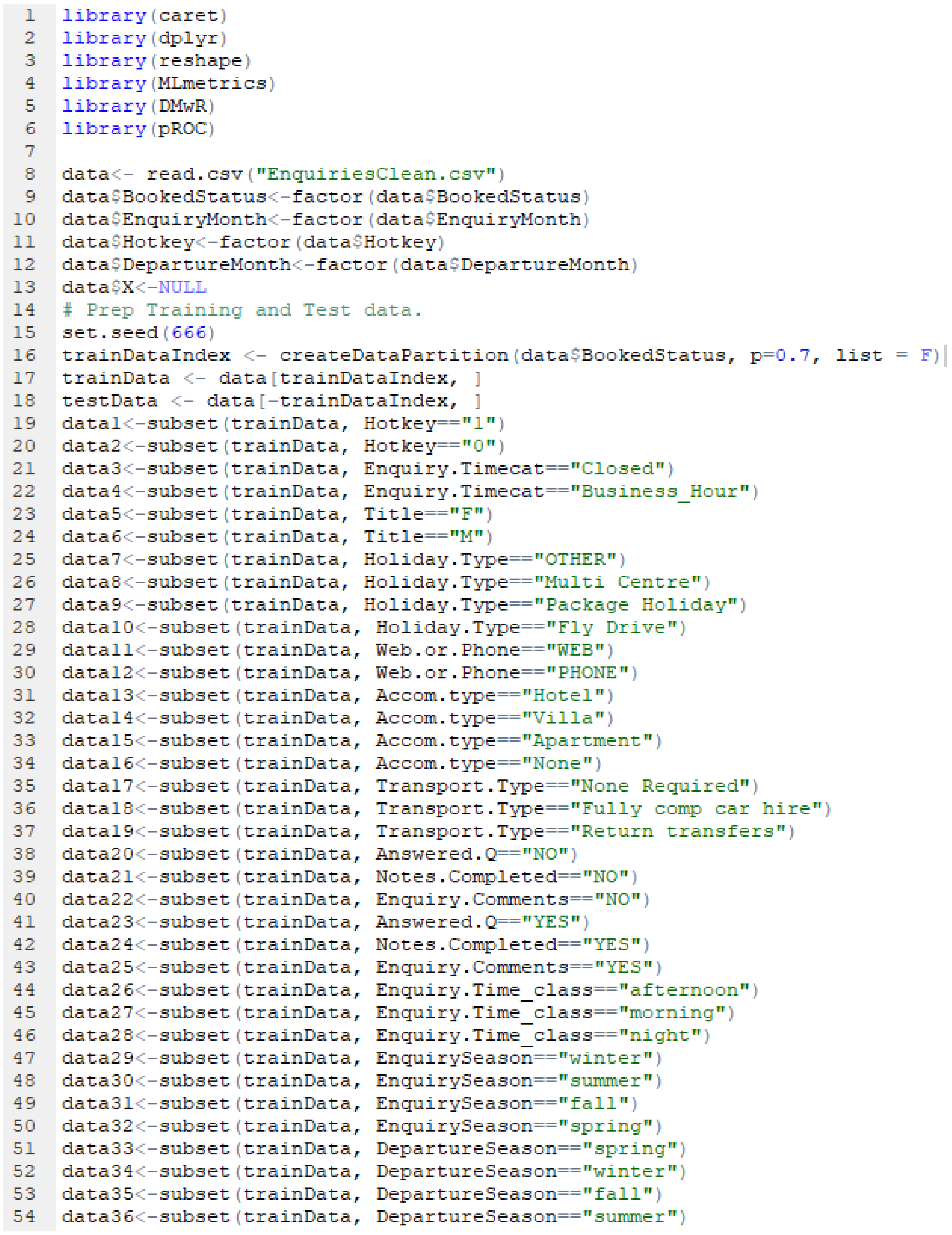


```
library(caret)
library(dplyr)
library(reshape)
library(MLmetrics)
library(DMwR)
library(pROC)
data<- read.csv("EnquiriesClean.csv")
data$BookedStatus<-factor(data$BookedStatus)
data$EnquiryMonth<-factor(data$EnquiryMonth)
data$Hotkey<-factor(data$Hotkey)
data$DepartureMonth<-factor(data$DepartureMonth)
data$X<-NULL
# Prep Training and Test data.
set.seed(666)
trainDataIndex <- createDataPartition(data$BookedStatus, p=0.7, list = F)
trainData <- data[trainDataIndex, ]
testData <- data[-trainDataIndex, ]

#Bootstrap
resample.spam.train <- function()
{
  indices <- sample(1:nrow(trainData),nrow(trainData),replace=TRUE)
  df <- trainData[indices,]
  return(df)
}

#Bagging function/storing predictions
baglog <- function(B) # B is the number of bootstrap samples
{
  bootstrap.samples <- list()
  pred.mat <- matrix(NA, nrow = nrow(testData), ncol = B)

  for(i in 1:B)
  {
  spam.sample1 <- resample.spam.train() # gets a bootstrap sample

  #spam.sample <- SMOTE(BookedStatus ~ .,
  #spam.sample1,perc.over = 100, perc.under=200,k=2)

  spam.sample <- downSample(x = spam.sample1[,-ncol(spam.sample1)],
                          y = spam.sample1$BookedStatus)
  spam.sample$BookedStatus<- NULL
  spam.sample<-rename(spam.sample,c(Class="BookedStatus"))

    m1 <- glm(BookedStatus ~ .,data = spam.sample1, family = binomial)

    # predictions
    pred.mat[,i] <- predict(m1,
                            newdata = testData[,-ncol(testData)],
                            type = "response")

   # pred.mat[,i] <- (pred.mat[,i]>=0.5)

  }
  return(pred.mat)
}
set.seed(666)
bag <- baglog(100)
```

```r
bagged.preds1 <- rowMeans(bag)

confusionMatrix(table(bagged.preds,testData$BookedStatus),
                positive='1')
confusionMatrix(table(bagged.preds,testData$BookedStatus),
                positive='1',mode = "prec_recall")

fg1 <- bagged.preds1[testData$BookedStatus == 1]
bg1 <- bagged.preds1[testData$BookedStatus == 0]
fg2 <- bagged.preds1[testData$BookedStatus == 1]
bg2 <- bagged.preds1[testData$BookedStatus == 0]
fg3 <- bagged.preds1[testData$BookedStatus == 1]
bg3 <- bagged.preds1[testData$BookedStatus == 0]
fg4 <- bagged.preds1[testData$BookedStatus == 1]
bg4 <- bagged.preds1[testData$BookedStatus == 0]


roc1 <- PRROC::roc.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
plot(roc1)
pr1 <- pr.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
plot(pr1)


roc2 <- PRROC::roc.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
plot(roc2)
pr2 <- pr.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
plot(pr2)


roc3 <- PRROC::roc.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
plot(roc3)
pr3 <- pr.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
plot(pr3)


roc4 <- PRROC::roc.curve(scores.class0 = fg4, scores.class1 = bg4, curve = T)
plot(roc4)
pr4 <- pr.curve(scores.class0 = fg4, scores.class1 = bg4, curve = T)
plot(pr4)

plot(roc1, col = 1, lty = 2, main = "ROC")
plot(roc2, col = 2, lty = 3, add=TRUE)
plot(roc3, col = 3, lty = 2, add=TRUE)
plot(roc4, col = 4, lty = 2, add=TRUE)
plot(roc6, col = 5, lty = 3, add=TRUE)
legend(x = "bottomright",
       legend = c("Oversample+bagging", "Undersample+bagging",
                  "SMOTE+bagging", "No Sampling+bagging", "No Sampling"),
       fill = 1:5)



plot(pr1, col = 1, lty = 2, main = "ROC")
plot(pr2, col = 2, lty = 3, add=TRUE)
plot(pr3, col = 3, lty = 2, add=TRUE)
plot(pr4, col = 4, lty = 2, add=TRUE)
plot(pr6, col = 5, lty = 3, add=TRUE)
legend(x = "topright",
       legend = c("Oversample+bagging", "Undersample+bagging",
                  "SMOTE+bagging", "No Sampling+bagging", "No Sampling"),
       fill = 1:5)
```

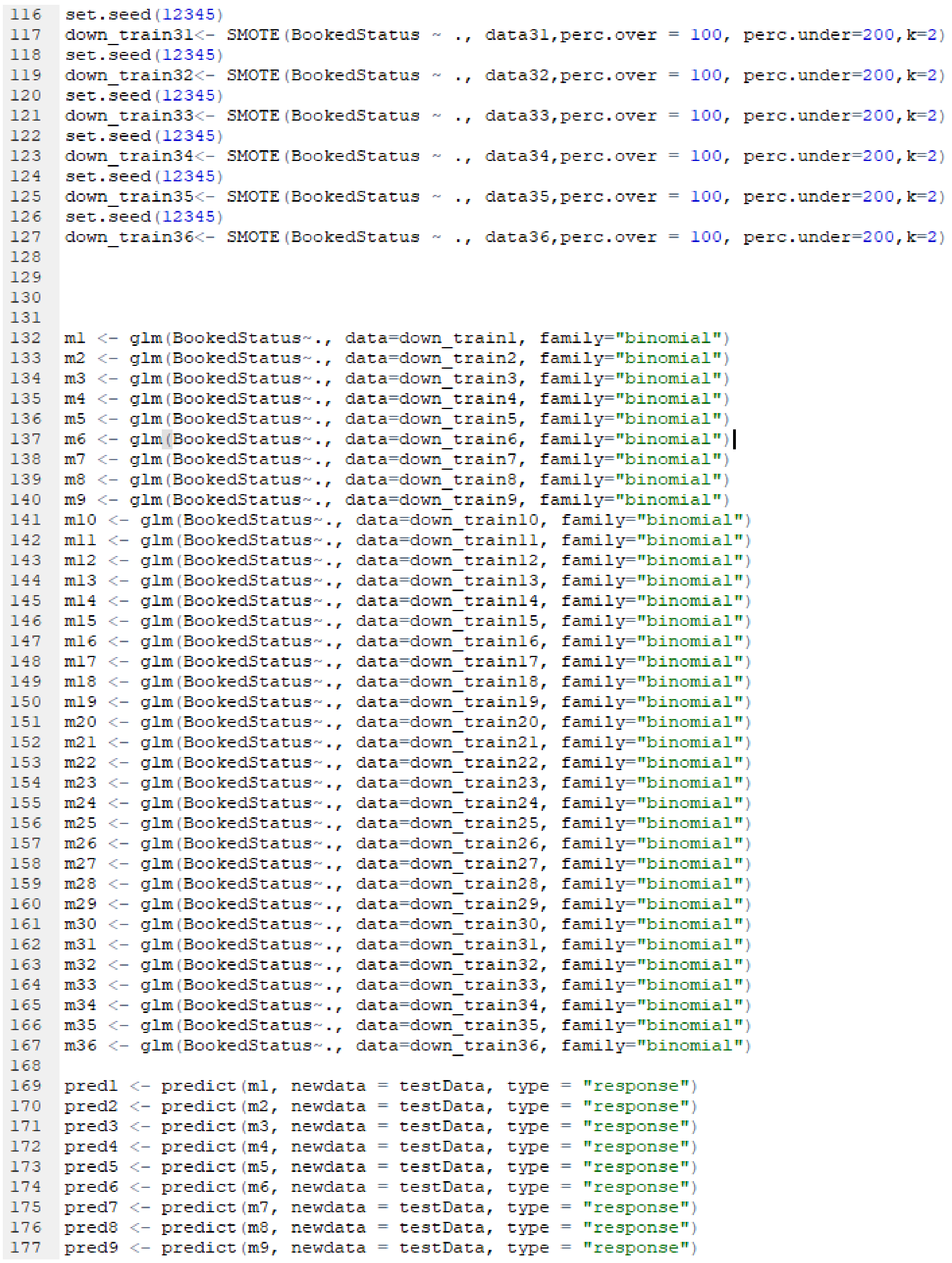
Logistic regression proposed method

Only SMOTE based proposed method is shown as only the data balancing methods (line 56-127) should be changed to carry out an over-sampling or under-sampling based proposed method. Under-sampling and over-sampling methods shown in the first page of the appendix is used for under-sampling and proposed method and over-sampling and proposed method.



```
1   library(caret)
2   library(dplyr)
3   library(reshape)
4   library(MLmetrics)
5   library(DMwR)
6   library(pROC)
7
8   data<- read.csv("EnquiriesClean.csv")
9   data$BookedStatus<-factor(data$BookedStatus)
10  data$EnquiryMonth<-factor(data$EnquiryMonth)
11  data$Hotkey<-factor(data$Hotkey)
12  data$DepartureMonth<-factor(data$DepartureMonth)
13  data$X<-NULL
14  # Prep Training and Test data.
15  set.seed(666)
16  trainDataIndex <- createDataPartition(data$BookedStatus, p=0.7, list = F)
17  trainData <- data[trainDataIndex, ]
18  testData <- data[-trainDataIndex, ]
19  data1<-subset(trainData, Hotkey=="1")
20  data2<-subset(trainData, Hotkey=="0")
21  data3<-subset(trainData, Enquiry.Timecat=="Closed")
22  data4<-subset(trainData, Enquiry.Timecat=="Business_Hour")
23  data5<-subset(trainData, Title=="F")
24  data6<-subset(trainData, Title=="M")
25  data7<-subset(trainData, Holiday.Type=="OTHER")
26  data8<-subset(trainData, Holiday.Type=="Multi Centre")
27  data9<-subset(trainData, Holiday.Type=="Package Holiday")
28  data10<-subset(trainData, Holiday.Type=="Fly Drive")
29  data11<-subset(trainData, Web.or.Phone=="WEB")
30  data12<-subset(trainData, Web.or.Phone=="PHONE")
31  data13<-subset(trainData, Accom.type=="Hotel")
32  data14<-subset(trainData, Accom.type=="Villa")
33  data15<-subset(trainData, Accom.type=="Apartment")
34  data16<-subset(trainData, Accom.type=="None")
35  data17<-subset(trainData, Transport.Type=="None Required")
36  data18<-subset(trainData, Transport.Type=="Fully comp car hire")
37  data19<-subset(trainData, Transport.Type=="Return transfers")
38  data20<-subset(trainData, Answered.Q=="NO")
39  data21<-subset(trainData, Notes.Completed=="NO")
40  data22<-subset(trainData, Enquiry.Comments=="NO")
41  data23<-subset(trainData, Answered.Q=="YES")
42  data24<-subset(trainData, Notes.Completed=="YES")
43  data25<-subset(trainData, Enquiry.Comments=="YES")
44  data26<-subset(trainData, Enquiry.Time_class=="afternoon")
45  data27<-subset(trainData, Enquiry.Time_class=="morning")
46  data28<-subset(trainData, Enquiry.Time_class=="night")
47  data29<-subset(trainData, EnquirySeason=="winter")
48  data30<-subset(trainData, EnquirySeason=="summer")
49  data31<-subset(trainData, EnquirySeason=="fall")
50  data32<-subset(trainData, EnquirySeason=="spring")
51  data33<-subset(trainData, DepartureSeason=="spring")
52  data34<-subset(trainData, DepartureSeason=="winter")
53  data35<-subset(trainData, DepartureSeason=="fall")
54  data36<-subset(trainData, DepartureSeason=="summer")
```

```r
set.seed(12345)
down_train1 <- SMOTE(BookedStatus ~ ., data1,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train2 <- SMOTE(BookedStatus ~ ., data2,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train3 <- SMOTE(BookedStatus ~ ., data3,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train4 <- SMOTE(BookedStatus ~ ., data4,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train5 <- SMOTE(BookedStatus ~ ., data5,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train6 <- SMOTE(BookedStatus ~ ., data6,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train7 <- SMOTE(BookedStatus ~ ., data7,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train8 <- SMOTE(BookedStatus ~ ., data8,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train9 <- SMOTE(BookedStatus ~ ., data9,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train10<- SMOTE(BookedStatus ~ .,data10,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train11<- SMOTE(BookedStatus ~ .,data11,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train12<- SMOTE(BookedStatus ~ ., data12,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train13<- SMOTE(BookedStatus ~ ., data13,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train14<- SMOTE(BookedStatus ~ ., data14,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train15<- SMOTE(BookedStatus ~ ., data15,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train16<- SMOTE(BookedStatus ~ ., data16,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train17<- SMOTE(BookedStatus ~ ., data17,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train18<- SMOTE(BookedStatus ~ ., data18,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train19<- SMOTE(BookedStatus ~ ., data19,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train20<- SMOTE(BookedStatus ~ ., data20,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train21<- SMOTE(BookedStatus ~ ., data21,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train22<- SMOTE(BookedStatus ~ ., data22,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train23<- SMOTE(BookedStatus ~ ., data23,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train24<- SMOTE(BookedStatus ~ ., data24,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train25<- SMOTE(BookedStatus ~ ., data25,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train26<- SMOTE(BookedStatus ~ ., data26,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train27<- SMOTE(BookedStatus ~ ., data27,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train28<- SMOTE(BookedStatus ~ ., data28,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train29<- SMOTE(BookedStatus ~ ., data29,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train30<- SMOTE(BookedStatus ~ ., data30,perc.over = 100, perc.under=200,k=2)
```

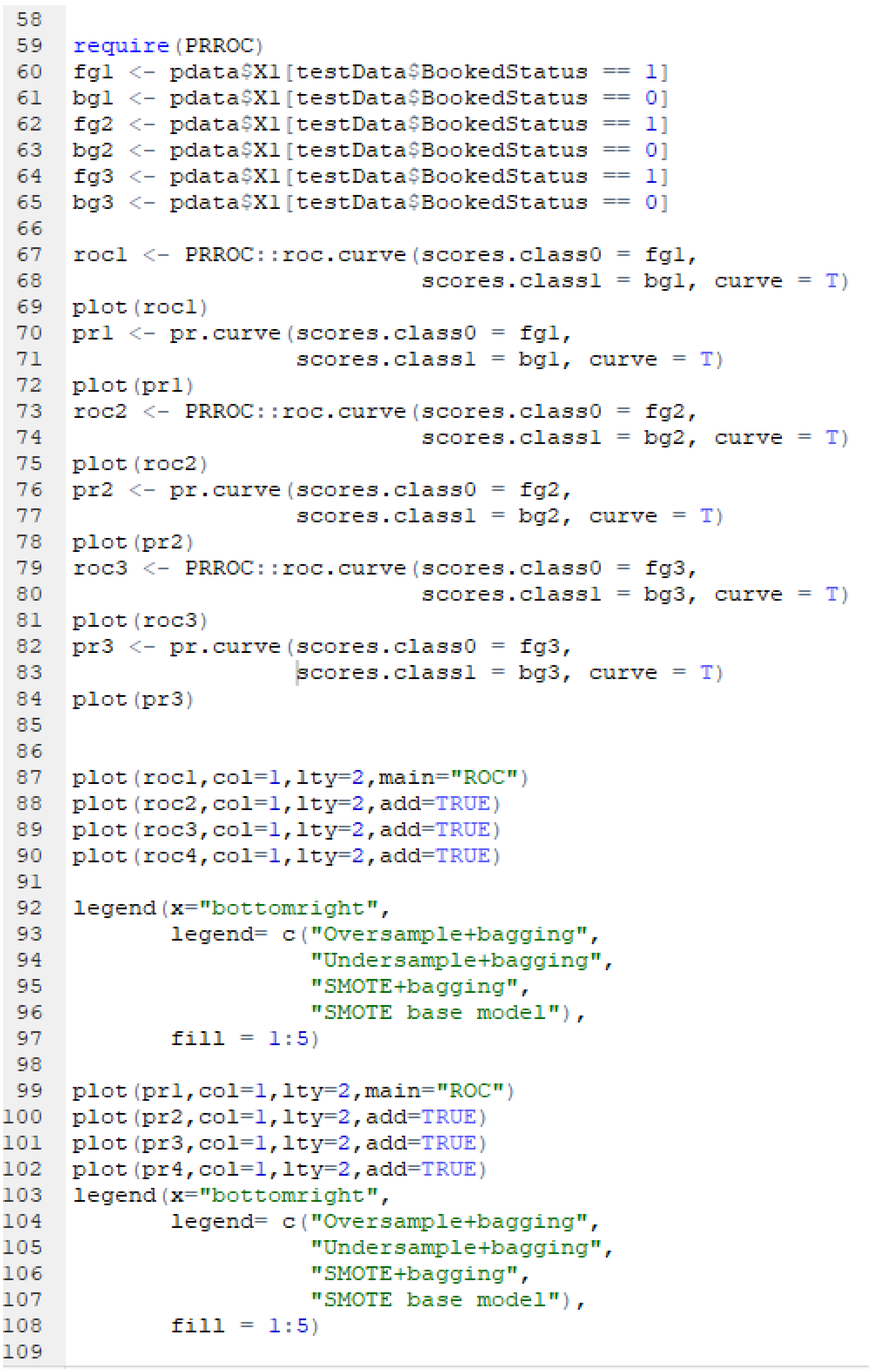


```r
set.seed(12345)
down_train31<- SMOTE(BookedStatus ~ ., data31,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train32<- SMOTE(BookedStatus ~ ., data32,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train33<- SMOTE(BookedStatus ~ ., data33,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train34<- SMOTE(BookedStatus ~ ., data34,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train35<- SMOTE(BookedStatus ~ ., data35,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train36<- SMOTE(BookedStatus ~ ., data36,perc.over = 100, perc.under=200,k=2)




m1 <- glm(BookedStatus~., data=down_train1, family="binomial")
m2 <- glm(BookedStatus~., data=down_train2, family="binomial")
m3 <- glm(BookedStatus~., data=down_train3, family="binomial")
m4 <- glm(BookedStatus~., data=down_train4, family="binomial")
m5 <- glm(BookedStatus~., data=down_train5, family="binomial")
m6 <- glm(BookedStatus~., data=down_train6, family="binomial")
m7 <- glm(BookedStatus~., data=down_train7, family="binomial")
m8 <- glm(BookedStatus~., data=down_train8, family="binomial")
m9 <- glm(BookedStatus~., data=down_train9, family="binomial")
m10 <- glm(BookedStatus~., data=down_train10, family="binomial")
m11 <- glm(BookedStatus~., data=down_train11, family="binomial")
m12 <- glm(BookedStatus~., data=down_train12, family="binomial")
m13 <- glm(BookedStatus~., data=down_train13, family="binomial")
m14 <- glm(BookedStatus~., data=down_train14, family="binomial")
m15 <- glm(BookedStatus~., data=down_train15, family="binomial")
m16 <- glm(BookedStatus~., data=down_train16, family="binomial")
m17 <- glm(BookedStatus~., data=down_train17, family="binomial")
m18 <- glm(BookedStatus~., data=down_train18, family="binomial")
m19 <- glm(BookedStatus~., data=down_train19, family="binomial")
m20 <- glm(BookedStatus~., data=down_train20, family="binomial")
m21 <- glm(BookedStatus~., data=down_train21, family="binomial")
m22 <- glm(BookedStatus~., data=down_train22, family="binomial")
m23 <- glm(BookedStatus~., data=down_train23, family="binomial")
m24 <- glm(BookedStatus~., data=down_train24, family="binomial")
m25 <- glm(BookedStatus~., data=down_train25, family="binomial")
m26 <- glm(BookedStatus~., data=down_train26, family="binomial")
m27 <- glm(BookedStatus~., data=down_train27, family="binomial")
m28 <- glm(BookedStatus~., data=down_train28, family="binomial")
m29 <- glm(BookedStatus~., data=down_train29, family="binomial")
m30 <- glm(BookedStatus~., data=down_train30, family="binomial")
m31 <- glm(BookedStatus~., data=down_train31, family="binomial")
m32 <- glm(BookedStatus~., data=down_train32, family="binomial")
m33 <- glm(BookedStatus~., data=down_train33, family="binomial")
m34 <- glm(BookedStatus~., data=down_train34, family="binomial")
m35 <- glm(BookedStatus~., data=down_train35, family="binomial")
m36 <- glm(BookedStatus~., data=down_train36, family="binomial")

pred1 <- predict(m1, newdata = testData, type = "response")
pred2 <- predict(m2, newdata = testData, type = "response")
pred3 <- predict(m3, newdata = testData, type = "response")
pred4 <- predict(m4, newdata = testData, type = "response")
pred5 <- predict(m5, newdata = testData, type = "response")
pred6 <- predict(m6, newdata = testData, type = "response")
pred7 <- predict(m7, newdata = testData, type = "response")
pred8 <- predict(m8, newdata = testData, type = "response")
pred9 <- predict(m9, newdata = testData, type = "response")
```

```r
pred10 <- predict(m10, newdata = testData, type = "response")
pred11 <- predict(m11, newdata = testData, type = "response")
pred12 <- predict(m12, newdata = testData, type = "response")
pred13 <- predict(m13, newdata = testData, type = "response")
pred14 <- predict(m14, newdata = testData, type = "response")
pred15 <- predict(m15, newdata = testData, type = "response")
pred16 <- predict(m16, newdata = testData, type = "response")
pred17 <- predict(m17, newdata = testData, type = "response")
pred18 <- predict(m18, newdata = testData, type = "response")
pred19 <- predict(m19, newdata = testData, type = "response")
pred20 <- predict(m20, newdata = testData, type = "response")
pred21 <- predict(m21, newdata = testData, type = "response")
pred22 <- predict(m22, newdata = testData, type = "response")
pred23 <- predict(m23, newdata = testData, type = "response")
pred24 <- predict(m24, newdata = testData, type = "response")
pred25 <- predict(m25, newdata = testData, type = "response")
pred26 <- predict(m26, newdata = testData, type = "response")
pred27 <- predict(m27, newdata = testData, type = "response")
pred28 <- predict(m28, newdata = testData, type = "response")
pred29 <- predict(m29, newdata = testData, type = "response")
pred30 <- predict(m30, newdata = testData, type = "response")
pred31 <- predict(m31, newdata = testData, type = "response")
pred32 <- predict(m32, newdata = testData, type = "response")
pred33 <- predict(m33, newdata = testData, type = "response")
pred34 <- predict(m34, newdata = testData, type = "response")
pred35 <- predict(m35, newdata = testData, type = "response")
pred36 <- predict(m36, newdata = testData, type = "response")

y_act <- testData$BookedStatus

new_pred<- (pred1+pred2+pred3+pred4+pred5+pred6+pred7+pred8+pred9+
              pred10+pred11+pred12+pred13+pred14+pred15+pred16+pred17+
              pred18+pred19+pred20+pred21+pred22+pred23+pred24+pred25+
              pred26+pred27+pred28+pred29+pred30+pred31+pred32+pred33+
              pred34+pred35+pred36)/36
new_pred2<- ifelse(new_pred > 0.20,1,0)
new_pred3<-factor(new_pred2)
confusionMatrix(table(new_pred2,y_act), positive='1')
confusionMatrix(table(new_pred2,y_act), positive='1',mode = "prec_recall")
fg7 <- new_pred[testData$BookedStatus == 1]
bg7 <- new_pred[testData$BookedStatus == 0]
roc3 <- PRROC::roc.curve(scores.class0 = fg7, scores.class1 = bg7, curve = T)
plot(roc3)
pr3 <- pr.curve(scores.class0 = fg7, scores.class1 = bg7, curve = T)
plot(pr3)
plot(roc2, col = 1, lty = 2, main = "ROC")
plot(roc, col = 2, lty = 3, add=TRUE)
plot(roc3, col = 3, lty = 2, add=TRUE)
plot(roc4, col = 4, lty = 2, add=TRUE)
plot(roc6, col = 5, lty = 3, add=TRUE)
legend(x = "bottomright",
       legend = c("Oversample+bagging", "Undersample+bagging",
                  "SMOTE+bagging", "No Sampling+bagging", "No Sampling"),
       fill = 1:5)
plot(pr2, col = 1, lty = 2, main = "ROC")
plot(pr, col = 2, lty = 3, add=TRUE)
plot(pr3, col = 3, lty = 2, add=TRUE)
plot(pr4, col = 4, lty = 2, add=TRUE)
plot(pr6, col = 5, lty = 3, add=TRUE)
legend(x = "topright",
       legend = c("Oversample+bagging", "Undersample+bagging",
                  "SMOTE+bagging", "No Sampling+bagging", "No Sampling"),
       fill = 1:5)
```

Decision tree (ROS, RUS, SMOTE, No sampling)

```r
library(caret)
library(dplyr)
library(reshape)
library(DMwR)

library(ipred)

data<- read.csv("EnquiriesClean.csv")
data$BookedStatus<-factor(data$BookedStatus)
data$EnquiryMonth<-factor(data$EnquiryMonth)
data$Hotkey<-factor(data$Hotkey)
data$DepartureMonth<-factor(data$DepartureMonth)

str(data)
data$X<-NULL

summary(data)

# Prep Training and Test data.
set.seed(666)
trainDataIndex <- createDataPartition(data$BookedStatus,
                                      p=0.7, list = F)
down_train1 <- data[trainDataIndex, ]
testData <- data[-trainDataIndex, ]


down_train1 <- upSample(x = trainData[,-ncol(trainData)],
                        y = trainData$BookedStatus)
down_train1$BookedStatus<- NULL
down_train1<-rename(down_train1,c(Class="BookedStatus"))

down_train2 <- downSample(x = trainData[,-ncol(trainData)],
                          y = trainData$BookedStatus)
down_train2$BookedStatus<- NULL
down_train2<-rename(down_train2,c(Class="BookedStatus"))

down_train3 <- SMOTE(BookedStatus ~ ., trainData,
                     perc.over = 100, perc.under=200,k=2)

m1 <- rpart(BookedStatus~.,
             method="class", data=down_train1)
m2 <- rpart(BookedStatus~.,
            method="class", data=down_train2)
m3 <- rpart(BookedStatus~.,
            method="class", data=down_train3)
m4 <- rpart(BookedStatus~.,
            method="class", trainData)


pdata <- as.data.frame(predict(m6, newdata = testData, type = "p"))
pdata$my_custom_predicted_class <- ifelse(pdata$X1 > .5, 1, 0)
pdata$my_custom_predicted_class<-factor(pdata$my_custom_predicted_class)
testData$BookedStatus<-factor(testData$BookedStatus)
# confusion matrix
caret::confusionMatrix(data = pdata$my_custom_predicted_class,
                       reference = testData$BookedStatus, positive = "1")
```



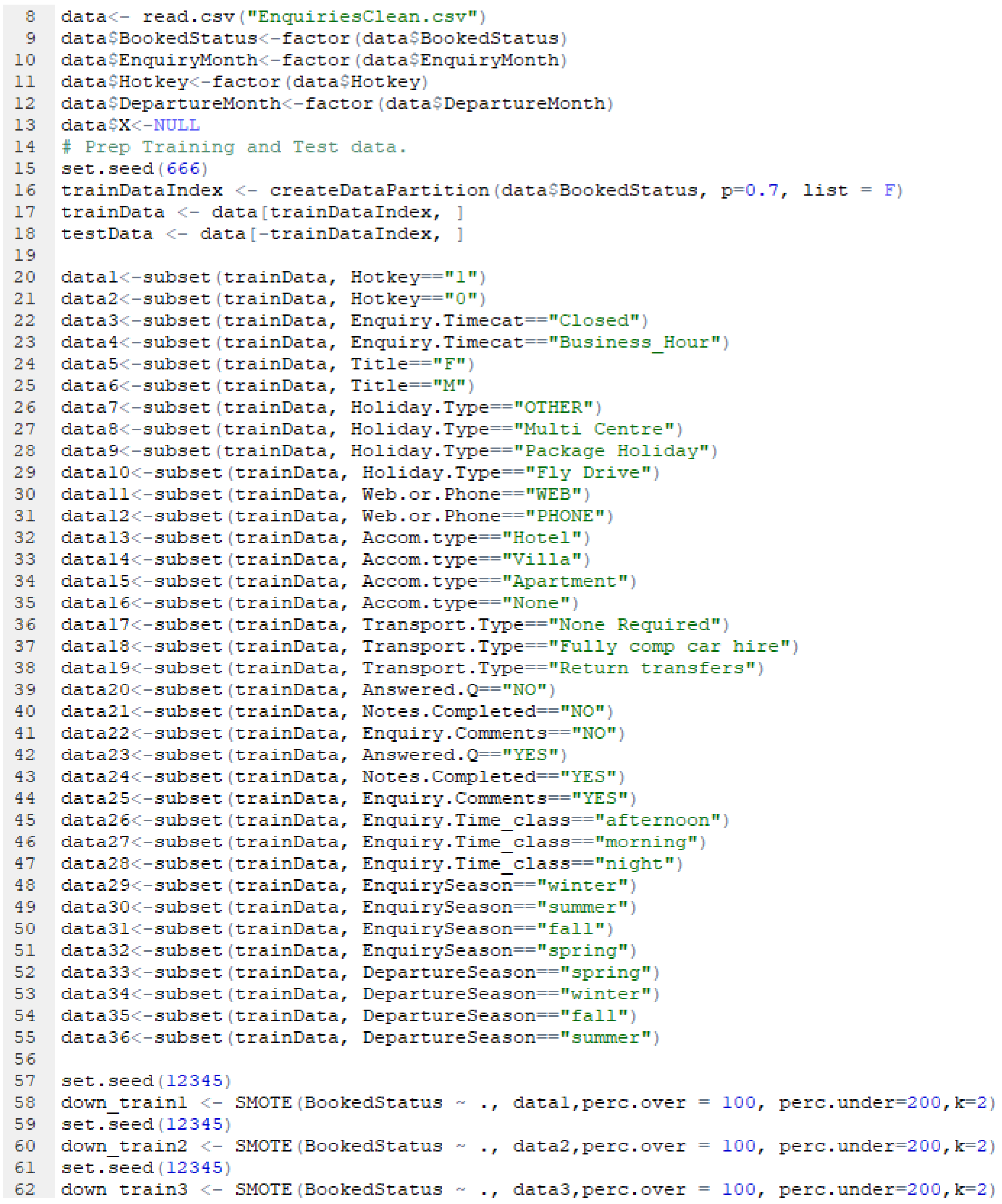
```r

require(PRROC)
fg1 <- pdata$X1[testData$BookedStatus == 1]
bg1 <- pdata$X1[testData$BookedStatus == 0]
fg2 <- pdata$X1[testData$BookedStatus == 1]
bg2 <- pdata$X1[testData$BookedStatus == 0]
fg3 <- pdata$X1[testData$BookedStatus == 1]
bg3 <- pdata$X1[testData$BookedStatus == 0]

roc1 <- PRROC::roc.curve(scores.class0 = fg1,
                         scores.class1 = bg1, curve = T)
plot(roc1)
pr1 <- pr.curve(scores.class0 = fg1,
                scores.class1 = bg1, curve = T)
plot(pr1)
roc2 <- PRROC::roc.curve(scores.class0 = fg2,
                         scores.class1 = bg2, curve = T)
plot(roc2)
pr2 <- pr.curve(scores.class0 = fg2,
                scores.class1 = bg2, curve = T)
plot(pr2)
roc3 <- PRROC::roc.curve(scores.class0 = fg3,
                         scores.class1 = bg3, curve = T)
plot(roc3)
pr3 <- pr.curve(scores.class0 = fg3,
                scores.class1 = bg3, curve = T)
plot(pr3)


plot(roc1,col=1,lty=2,main="ROC")
plot(roc2,col=1,lty=2,add=TRUE)
plot(roc3,col=1,lty=2,add=TRUE)
plot(roc4,col=1,lty=2,add=TRUE)

legend(x="bottomright",
       legend= c("Oversample+bagging",
                 "Undersample+bagging",
                 "SMOTE+bagging",
                 "SMOTE base model"),
       fill = 1:5)

plot(pr1,col=1,lty=2,main="ROC")
plot(pr2,col=1,lty=2,add=TRUE)
plot(pr3,col=1,lty=2,add=TRUE)
plot(pr4,col=1,lty=2,add=TRUE)
legend(x="bottomright",
       legend= c("Oversample+bagging",
                 "Undersample+bagging",
                 "SMOTE+bagging",
                 "SMOTE base model"),
       fill = 1:5)

```

Decision tree Bagging

```
library(caret)
library(reshape)
data<- read.csv("EnquiriesClean.csv")
data$BookedStatus<-factor(data$BookedStatus)
data$EnquiryMonth<-factor(data$EnquiryMonth)
data$Hotkey<-factor(data$Hotkey)
data$DepartureMonth<-factor(data$DepartureMonth)
data$X<-NULL
# Prep Training and Test data.
set.seed(666)
trainDataIndex <- createDataPartition(data$BookedStatus, p=0.7, list = F)
trainData<- data[trainDataIndex, ]
testData <- data[-trainDataIndex, ]
levels(trainData$BookedStatus)<-make.names(levels(factor(trainData$BookedStatus)))
ctrl <- trainControl(classProbs = TRUE,summaryFunction = twoClassSummary,sampling = "up")
set.seed(5627)
down_inside <- train(BookedStatus ~ ., data = trainData, method = "treebag",
                     nbagg = 100,metric = "ROC",trControl = ctrl)

pdata <- as.data.frame(predict(down_inside, newdata = testData, type = "prob"))
pdata$my_custom_predicted_class <- ifelse(pdata$X1 > .205, 1, 0)
pdata$my_custom_predicted_class<-factor(pdata$my_custom_predicted_class)
testData$BookedStatus<-factor(testData$BookedStatus)
# confusion matrix
caret::confusionMatrix(data = pdata$my_custom_predicted_class,
        reference = testData$BookedStatus, positive = "1")
caret::confusionMatrix(data = pdata$my_custom_predicted_class,
        reference = testData$BookedStatus, positive = "1",mode="prec_recall")

require(PRROC)
fg1 <- pdata$X1[testData$BookedStatus == 1]
bg1 <- pdata$X1[testData$BookedStatus == 0]
fg2 <- pdata$X1[testData$BookedStatus == 1]
bg2 <- pdata$X1[testData$BookedStatus == 0]
fg3 <- pdata$X1[testData$BookedStatus == 1]
bg3 <- pdata$X1[testData$BookedStatus == 0]

roc1 <- PRROC::roc.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
plot(roc1)
pr1 <- pr.curve(scores.class0 = fg1, scores.class1 = bg1, curve = T)
plot(pr1)
roc2 <- PRROC::roc.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
plot(roc2)
pr2 <- pr.curve(scores.class0 = fg2, scores.class1 = bg2, curve = T)
plot(pr2)
roc3 <- PRROC::roc.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
plot(roc3)
pr3 <- pr.curve(scores.class0 = fg3, scores.class1 = bg3, curve = T)
plot(pr3)
plot(roc1,col=1,lty=2,main="ROC")
plot(roc2,col=1,lty=2,add=TRUE)
plot(roc3,col=1,lty=2,add=TRUE)
plot(roc4,col=1,lty=2,add=TRUE)

legend(x="bottomright", legend= c("Oversample+bagging",  "Undersample+bagging",
                 "SMOTE+bagging","SMOTE base model"),fill = 1:5)
plot(pr1,col=1,lty=2,main="ROC")
plot(pr2,col=1,lty=2,add=TRUE)
plot(pr3,col=1,lty=2,add=TRUE)
plot(pr4,col=1,lty=2,add=TRUE)
legend(x="bottomright", legend= c("Oversample+bagging", "Undersample+bagging",
                 "SMOTE+bagging","SMOTE base model"),fill = 1:5)
.
```

Decision tree proposed method

Only SMOTE based proposed method is shown as only the data balancing methods (line 56-127) should be changed to carry out an over-sampling or under-sampling based proposed method. Under-sampling and over-sampling methods shown in the first page of the appendix is used for under-sampling and proposed method and over-sampling and proposed method.



```
 8  data<- read.csv("EnquiriesClean.csv")
 9  data$BookedStatus<-factor(data$BookedStatus)
10  data$EnquiryMonth<-factor(data$EnquiryMonth)
11  data$Hotkey<-factor(data$Hotkey)
12  data$DepartureMonth<-factor(data$DepartureMonth)
13  data$X<-NULL
14  # Prep Training and Test data.
15  set.seed(666)
16  trainDataIndex <- createDataPartition(data$BookedStatus, p=0.7, list = F)
17  trainData <- data[trainDataIndex, ]
18  testData <- data[-trainDataIndex, ]
19
20  data1<-subset(trainData, Hotkey=="1")
21  data2<-subset(trainData, Hotkey=="0")
22  data3<-subset(trainData, Enquiry.Timecat=="Closed")
23  data4<-subset(trainData, Enquiry.Timecat=="Business_Hour")
24  data5<-subset(trainData, Title=="F")
25  data6<-subset(trainData, Title=="M")
26  data7<-subset(trainData, Holiday.Type=="OTHER")
27  data8<-subset(trainData, Holiday.Type=="Multi Centre")
28  data9<-subset(trainData, Holiday.Type=="Package Holiday")
29  data10<-subset(trainData, Holiday.Type=="Fly Drive")
30  data11<-subset(trainData, Web.or.Phone=="WEB")
31  data12<-subset(trainData, Web.or.Phone=="PHONE")
32  data13<-subset(trainData, Accom.type=="Hotel")
33  data14<-subset(trainData, Accom.type=="Villa")
34  data15<-subset(trainData, Accom.type=="Apartment")
35  data16<-subset(trainData, Accom.type=="None")
36  data17<-subset(trainData, Transport.Type=="None Required")
37  data18<-subset(trainData, Transport.Type=="Fully comp car hire")
38  data19<-subset(trainData, Transport.Type=="Return transfers")
39  data20<-subset(trainData, Answered.Q=="NO")
40  data21<-subset(trainData, Notes.Completed=="NO")
41  data22<-subset(trainData, Enquiry.Comments=="NO")
42  data23<-subset(trainData, Answered.Q=="YES")
43  data24<-subset(trainData, Notes.Completed=="YES")
44  data25<-subset(trainData, Enquiry.Comments=="YES")
45  data26<-subset(trainData, Enquiry.Time_class=="afternoon")
46  data27<-subset(trainData, Enquiry.Time_class=="morning")
47  data28<-subset(trainData, Enquiry.Time_class=="night")
48  data29<-subset(trainData, EnquirySeason=="winter")
49  data30<-subset(trainData, EnquirySeason=="summer")
50  data31<-subset(trainData, EnquirySeason=="fall")
51  data32<-subset(trainData, EnquirySeason=="spring")
52  data33<-subset(trainData, DepartureSeason=="spring")
53  data34<-subset(trainData, DepartureSeason=="winter")
54  data35<-subset(trainData, DepartureSeason=="fall")
55  data36<-subset(trainData, DepartureSeason=="summer")
56
57  set.seed(12345)
58  down_train1 <- SMOTE(BookedStatus ~ ., data1,perc.over = 100, perc.under=200,k=2)
59  set.seed(12345)
60  down_train2 <- SMOTE(BookedStatus ~ ., data2,perc.over = 100, perc.under=200,k=2)
61  set.seed(12345)
62  down_train3 <- SMOTE(BookedStatus ~ ., data3,perc.over = 100, perc.under=200,k=2)
```

```
set.seed(12345)
down_train4 <- SMOTE(BookedStatus ~ ., data4,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train5 <- SMOTE(BookedStatus ~ ., data5,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train6 <- SMOTE(BookedStatus ~ ., data6,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train7 <- SMOTE(BookedStatus ~ ., data7,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train8 <- SMOTE(BookedStatus ~ ., data8,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train9 <- SMOTE(BookedStatus ~ ., data9,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train10 <- SMOTE(BookedStatus ~ ., data10,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train11 <- SMOTE(BookedStatus ~ ., data11,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train12<- SMOTE(BookedStatus ~ ., data12,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train13<- SMOTE(BookedStatus ~ ., data13,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train14<- SMOTE(BookedStatus ~ ., data14,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train15<- SMOTE(BookedStatus ~ ., data15,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train16<- SMOTE(BookedStatus ~ ., data16,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train17<- SMOTE(BookedStatus ~ ., data17,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train18<- SMOTE(BookedStatus ~ ., data18,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train19<- SMOTE(BookedStatus ~ ., data19,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train20<- SMOTE(BookedStatus ~ ., data20,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train21<- SMOTE(BookedStatus ~ ., data21,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train22<- SMOTE(BookedStatus ~ ., data22,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train23<- SMOTE(BookedStatus ~ ., data23,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train24<- SMOTE(BookedStatus ~ ., data24,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train25<- SMOTE(BookedStatus ~ ., data25,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train26<- SMOTE(BookedStatus ~ ., data26,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train27<- SMOTE(BookedStatus ~ ., data27,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train28<- SMOTE(BookedStatus ~ ., data28,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train29<- SMOTE(BookedStatus ~ ., data29,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train30<- SMOTE(BookedStatus ~ ., data30,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train31<- SMOTE(BookedStatus ~ ., data31,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train32<- SMOTE(BookedStatus ~ ., data32,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train33<- SMOTE(BookedStatus ~ ., data33,perc.over = 100, perc.under=200,k=2)
```

```
set.seed(12345)
down_train34<- SMOTE(BookedStatus ~ ., data34,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train35<- SMOTE(BookedStatus ~ ., data35,perc.over = 100, perc.under=200,k=2)
set.seed(12345)
down_train36<- SMOTE(BookedStatus ~ ., data36,perc.over = 100, perc.under=200,k=2)

m1 <- rpart(BookedStatus~., method="class", data=down_train1 )
m2 <- rpart(BookedStatus~., method="class", data=down_train2 )
m3 <- rpart(BookedStatus~., method="class", data=down_train3 )
m4 <- rpart(BookedStatus~., method="class", data=down_train4 )
m5 <- rpart(BookedStatus~., method="class", data=down_train5 )
m6 <- rpart(BookedStatus~., method="class", data=down_train6 )
m7 <- rpart(BookedStatus~., method="class", data=down_train7 )
m8 <- rpart(BookedStatus~., method="class", data=down_train8 )
m9 <- rpart(BookedStatus~., method="class", data=down_train9 )
m10 <- rpart(BookedStatus~., method="class", data=down_train10 )
m11 <- rpart(BookedStatus~., method="class", data=down_train11 )
m12 <- rpart(BookedStatus~., method="class", data=down_train12 )
m13 <- rpart(BookedStatus~., method="class", data=down_train13 )
m14 <- rpart(BookedStatus~., method="class", data=down_train14 )
m15 <- rpart(BookedStatus~., method="class", data=down_train15 )
m16 <- rpart(BookedStatus~., method="class", data=down_train16 )
m17 <- rpart(BookedStatus~., method="class", data=down_train17 )
m18 <- rpart(BookedStatus~., method="class", data=down_train18 )
m19 <- rpart(BookedStatus~., method="class", data=down_train19 )
m20 <- rpart(BookedStatus~., method="class", data=down_train20 )
m21 <- rpart(BookedStatus~., method="class", data=down_train21 )
m22 <- rpart(BookedStatus~., method="class", data=down_train22 )
m23 <- rpart(BookedStatus~., method="class", data=down_train23 )
m24 <- rpart(BookedStatus~., method="class", data=down_train24 )
m25 <- rpart(BookedStatus~., method="class", data=down_train25 )
m26 <- rpart(BookedStatus~., method="class", data=down_train26 )
m27 <- rpart(BookedStatus~., method="class", data=down_train27 )
m28 <- rpart(BookedStatus~., method="class", data=down_train28 )
m29 <- rpart(BookedStatus~., method="class", data=down_train29 )
m30 <- rpart(BookedStatus~., method="class", data=down_train30 )
m31<-  rpart(BookedStatus~., method="class", data=down_train31 )
m32<-  rpart(BookedStatus~., method="class", data=down_train32 )
m33 <- rpart(BookedStatus~., method="class", data=down_train33 )
m34<-  rpart(BookedStatus~., method="class", data=down_train34 )
m35<-  rpart(BookedStatus~., method="class", data=down_train35 )
m36 <- rpart(BookedStatus~., method="class", data=down_train36 )

pred1 <- predict(m1, newdata = testData, type = "prob")
pred2 <- predict(m2, newdata = testData, type = "prob")
pred3 <- predict(m3, newdata = testData, type = "prob")
pred4 <- predict(m4, newdata = testData, type = "prob")
pred5 <- predict(m5, newdata = testData, type = "prob")
pred6 <- predict(m6, newdata = testData, type = "prob")
pred7 <- predict(m7, newdata = testData, type = "prob")
pred8 <- predict(m8, newdata = testData, type = "prob")
pred9 <- predict(m9, newdata = testData, type = "prob")
pred10 <- predict(m10, newdata = testData, type = "prob")
pred11 <- predict(m11, newdata = testData, type = "prob")
pred12 <- predict(m12, newdata = testData, type = "prob")
pred13 <- predict(m13, newdata = testData, type = "prob")
pred14 <- predict(m14, newdata = testData, type = "prob")
pred15 <- predict(m15, newdata = testData, type = "prob")
pred16 <- predict(m16, newdata = testData, type = "prob")
pred17 <- predict(m17, newdata = testData, type = "prob")
```

```
pred18 <- predict(m18, newdata = testData, type = "prob")
pred19 <- predict(m19, newdata = testData, type = "prob")
pred20 <- predict(m20, newdata = testData, type = "prob")
pred21 <- predict(m21, newdata = testData, type = "prob")
pred22 <- predict(m22, newdata = testData, type = "prob")
pred23 <- predict(m23, newdata = testData, type = "prob")
pred24 <- predict(m24, newdata = testData, type = "prob")
pred25 <- predict(m25, newdata = testData, type = "prob")
pred26 <- predict(m26, newdata = testData, type = "prob")
pred27 <- predict(m27, newdata = testData, type = "prob")
pred28 <- predict(m28, newdata = testData, type = "prob")
pred29 <- predict(m29, newdata = testData, type = "prob")
pred30 <- predict(m30, newdata = testData, type = "prob")
pred31 <- predict(m31, newdata = testData, type = "prob")
pred32 <- predict(m32, newdata = testData, type = "prob")
pred33 <- predict(m33, newdata = testData, type = "prob")
pred34 <- predict(m34, newdata = testData, type = "prob")
pred35 <- predict(m35, newdata = testData, type = "prob")
pred36 <- predict(m36, newdata = testData, type = "prob")

y_act <- testData$BookedStatus

new_pred<- (pred1+pred2+pred3+pred4+pred5+pred6+pred7+pred8+pred9+pred10+
              pred11+pred12+pred13+pred14+pred15+pred16+pred17+pred18+pred19+
              pred20+pred21+pred22+pred23+pred24+pred25+pred26+pred27+pred28+
              pred29+pred30+pred31+pred32+pred33+pred34+pred35+pred36)/36

new_pred<-data.frame(new_pred)

new_pred2<- ifelse(new_pred$X1 > 0.125,1,0)

#new_pred3<-factor(new_pred2)
confusionMatrix(table(new_pred2,y_act), positive='1')
confusionMatrix(table(new_pred2,y_act), positive='1',mode = "prec_recall")

fgA <- new_pred$X1[testData$BookedStatus == 1]
bgA <- new_pred$X1[testData$BookedStatus == 0]
fgB <- new_pred$X1[testData$BookedStatus == 1]
bgB <- new_pred$X1[testData$BookedStatus == 0]
fgC <- new_pred$X1[testData$BookedStatus == 1]
bgC <- new_pred$X1[testData$BookedStatus == 0]

rocA <- PRROC::roc.curve(scores.class0 = fgA, scores.class1 = bgA, curve = T)
plot(rocA)
prA <- pr.curve(scores.class0 = fgA, scores.class1 = bgA, curve = T)
plot(prA)

rocB <- PRROC::roc.curve(scores.class0 = fgB, scores.class1 = bgB, curve = T)
plot(rocB)
prB <- pr.curve(scores.class0 = fgB, scores.class1 = bgB, curve = T)
plot(prB)

rocC <- PRROC::roc.curve(scores.class0 = fgC, scores.class1 = bgC, curve = T)
plot(rocC)
prC <- pr.curve(scores.class0 = fgC, scores.class1 = bgC, curve = T)
plot(prC)
```