

Serial Number xxxxxxxxxx



南京航空航天大学

Undergraduate Graduation Project (Thesis)

Title Classification of A Bank Customer
Satisfaction

Student Name Aanchal Upreti 艾琳

Student ID 191764145

College College of International Education
Software Engineering and

Major Management

Class 1917641

Faculty Adviser Professor Donghai Guan

June, 2021

Nanjing University of Aeronautics and Astronautics
Bachelor Degree Project (Thesis)
The Letter of Commitment

I solemnly declare that my graduation project (Classification of a Bank Customer Satisfaction) submitted is the result of my independent research under the guidance of my supervisor. All I know is that this graduation project(thesis) does not contain any other published or written work by individuals or groups other than those specifically noted and acknowledged in the text. Other individuals and groups that have contributed to the research work involved in this graduation project(thesis) have been clearly identified in the text.

Student's Signature



Date: MAY 19, 2021

Nanjing University of Aeronautics and Astronautics

Bachelor Degree Project (Thesis)

Statement of Authorization

I am fully aware of the regulations of Nanjing University of Aeronautics and Astronautics (NUAA) on the collection, retention and use of the graduation project (thesis). The intellectual property right of the undergraduate graduation project (thesis) belongs to Nanjing University of Aeronautics and Astronautics. NUAA has the right to retain and send the hard copy and soft copy of the graduation project (thesis) to the relevant departments or institutions of the state, allow the thesis to be consulted and borrowed, publish all or part of the content of the thesis, save and compile the thesis by means of photocopying, reducing or scanning. This statement applies to security papers after they are declassified.

Confidentiality of papers:



Not confidential

☐ Confidential, period of confidentiality (from to)

Student's
Signature



Adviser's
signature



Date: 2021/05/30

Date: 2021/05/30

ABSTRACT

Customer satisfaction has been a major concern in banking industries lately as it is the primary criterion used to assess the success of banks in the competitive market. It is believed that dissatisfied customers are more likely to leave and usually it happens without any warning which makes it a difficult job for the bank to anticipate customer dissatisfaction. Thus, to take proactive steps in customer retention and ensure maximum customer satisfaction, Santander bank aims to predict whether the customers are satisfied or not using several machine learning algorithms.

This is a binary classification problem with highly imbalanced and anonymized datasets where several preprocessing, feature engineering and feature selection techniques are implemented using Python programming language to prepare data for training. Logistic Regression, Random Forest, XGBoost, Multi-layer Perceptron, Voting and stacking classifier are used to train the model to classify the unhappy customer as 1 and happy customer as 0. The accuracy of each model is calculated using Area Under the Curve (AUC) matrix and the comparison among these models are done to select the best performing model. The voting classifier ensembled using XGBoost and MLP classifier is found to be the best model with the highest accuracy of 82.62%. Similarly, the feature called Var15 which is assumed to be the age of customer is considered to be the most important factor in determining the customer satisfaction.

In the future, more exploratory data analysis and feature engineering could be done to select the most dependent features for training the data. Also, the research on the application of other models can also be performed and the combination of different models can be implemented to get more optimized accuracy score.

KEY WORDS: Customer Satisfaction, Classification, Feature Engineering, Cross-validation, Tuning

摘要

客户满意度是银行业最近关注的一个主要问题，因为它是衡量银行在竞争激烈的市场中取得成功的主要标准。据信，不满意的客户更容易离开，而且通常在没有任何警告的情况下离开，这使得银行很难预测客户的不满。因此，为了采取主动措施留住客户并确保最大程度的客户满意度，桑坦德银行的目标是使用多种机器学习算法预测客户是否满意。

这是一个具有高度不平衡和匿名数据集的二元分类问题，其中使用 Python 编程语言实现了一些预处理、特征工程和特征选择技术来准备训练数据。Logistic Regression, Random Forest, XGBoost, Multi-Layer Perceptron, Voting and Stacking classifier 对模型进行训练，将不满意顾客分类为 1，快乐顾客分类为 0，利用曲线下面积（AUC）矩阵计算各模型的精度，并对各模型进行比较，选出性能最好的模型。使用 XGBoost 和 MLP 分类器集成的投票分类器是最好的模型，最高准确率为 82.62%。同样，被认为是客户年龄的 Var15 特性被认为是决定客户满意度的最重要因素。

在未来，可以进行更多的探索性数据分析和特征工程来选择最依赖的特征来训练数据。此外，还可以对其他模型的应用进行研究，实现不同模型的组合，以获得更优的精度得分。

关键词：客户满意度、分类、特征工程、交叉验证、调优

Contents

Chapter 1 Introduction	5
1.1 Research Background	5
1.2 Research Objectives	6
1.3 Related Works	6
1.4 Main Tasks	8
1.5 Research Structure	9
Chapter 2 Fundamentals	10
2.1 Classification	10
2.1.1 Logistic Regression	10
2.1.2 MLP Classifier	12
2.2 Decision Trees	13
2.3 Ensemble Modeling	14
2.3.1 Random Forest	15
2.3.2 eXtreme Gradient Boosting	15
2.3.3 Voting and Stacking Classifier	16
2.4 Evaluation Metrics	17
2.4.1 Confusion Matrix	17
2.4.2 Accuracy	18
2.4.3 Area Under the Curve (AUC)	18
2.5 Tools and Libraries used	19
Chapter 3 Exploratory Feature Analysis	20
3.1 Data Source	20
3.2 Feature Exploration and visualization	20
3.3 Feature Dictionary	22
3.4 Individual Feature Analysis	23
3.4.1 Var15	24
3.4.2 Var38	25
3.5 Feature Groupings	27
Chapter 4 Implementation	28
4.1 Data Preprocessing	28
4.1.1 Data Cleaning	28
4.1.2 Feature Engineering	28
4.1.3 Feature Selection	29
4.1.4 Data Scaling	30
4.1.5 Data Splitting	30
4.2 Modeling, Parameter Tuning and Cross-Validating	31
4.2.1 Logistic Regression	31
4.2.2 XGBoost	32
4.2.3 MLP classifier	34
4.2.4 Random Forest	36
4.3 Model Ensemble	38
4.3.1 Use of Voting Classifier	38
4.3.2 Use of Stacking Classifier	39
Chapter 5 Results	40
5.1 Performance Measurement	40
5.1.1 AUC score of Logistic Regression	40
5.1.2 AUC score of MLP classifier	40
5.1.3 AUC score of XGBoost	40

5.1.4	AUC score of Random Forest	41
5.1.5	AUC score of Voting Classifier	41
5.1.6	AUC score of Stacking Classifier	42
5.2	ROC_AUC plot	42
5.2.1	ROC_AUC of XGBoost	42
5.2.2	ROC_AUC of Logistic Regression	43
5.3	Comparison of Accuracy	43
5.3.1	XGBoost, Random Forest and Logistic Regression	43
5.3.2	Voting Classifier and Logistic Regression	44
5.3.3	Overall classifier	44
5.4	Selection of important features	45
5.5	Cross-validation of Voting Classifier	46
Chapter 6	Conclusion	47
References	47
Acknowledgement	51

Chapter 1 Introduction

1.1 Research Background

In this era of globalization, customer service has been an important factor governing businesses and thus, measuring customer satisfaction has been a very crucial step for companies to assess if their business is running in a healthy way. With the advent of technological advancement, the demands of customers in service-oriented industries are increasing rapidly, hence causing stiff competition among businesses. Nowadays businesses are more customer-centric than ever before as assessing customer satisfaction allows companies to enhance customer relationships and give them a competitive edge for sustainable success^[1].

A satisfied customer is one who has their expectations exceeded, whether during the purchase of services, after sales, or when handling tools on a daily basis, and thus assures loyalty to the companies and help businesses increase profits and command convincing market shares within their respective industries.

Similarly, Customer satisfaction has been a major concern in banking industries lately as it is the primary criterion used to assess the relationships of banks with the competitive market^[2]. It is believed that dissatisfied customers are more likely to leave and usually it happens without any prior warning^[3]. This makes it a difficult job for the bank to anticipate customer dissatisfaction. Meanwhile, identifying and analyzing the customer's contentment to improve customer retention can yield many benefits for banks. Keeping a current customer faithful requires five times less effort, time and money than getting a new one. Such a customer is willing to pay higher prices, is a free form of advertising for the bank, and is inclined to purchase further products. The clients also have a better understanding of the organization and can give positive word-of-mouth promotion. Therefore, it is important for commercial banks to ensure maximum customer satisfaction.

Thus, it has become more than necessary for banks to predict whether their customers are satisfied or not beforehand so that they can take proactive steps in customers retention. Usually, customer satisfaction is quantified under survey based metrics that require customers to answer questionnaires and give feedbacks on customer experience manually. Predicting customer satisfaction, is however, a very challenging task as it requires many relevant data for the effective analysis and manual survey-based techniques are found to be not so effective. Thus, the concept of datamining comes handy when it comes to dealing with such large datasets. Data mining allows extracting knowledge from the historical data and predicting outcomes of future situations by

automating the process through different Machine Learning techniques. It helps optimize business decisions, increase the value of each customer, and improve customer satisfaction.

In this thesis, we work on predicting whether a customer is satisfied or not based on the anonymized datasets provided by Santander Bank at Kaggle.com as a Kaggle competition^[4]. Santander Bank is a large corporation focusing principally on the market in the northeast United States. This is a binary classification problem implementing several datamining techniques using Python programming language. It compares the accuracy among several classification models under Area under the curve matrix and thus classify the customer as either satisfied or unsatisfied and aims to help Santander bank take proactive steps to improve a customer's happiness before they would take their business elsewhere.

1.2 Research Objectives

This research work aims on dealing with binary classification problem where its main objectives are given as:

- To classify Santander bank's customer as either satisfied (Labeled as 0) or unsatisfied (labeled as 1) by analyzing the given anonymized dataset.
- To perform exploratory data analysis, feature engineering and feature selection techniques to select the best features for model training.
- To compare the predictive ability of different classifier algorithms and measure the accuracy of the model in terms of Receiver Operating Characteristic Area Under the Curve (ROC AUC).
- To select the best performing classifier model and use it for prediction in unseen data.

1.3 Related Works

Customer satisfaction is defined as “the individual's perception of the performance of the products or services in relation to his or her expectations”^[5]. Customer satisfaction is a key factor in organization that defines long term involvement of customer^[6]. Similarly, Mittal & Kamakura revealed that customer satisfaction is crucial element of customer wishes for upcoming purchases. Therefore, those customers who are satisfied are ready to share their positive thoughts with others^[7]. Customer satisfaction has a great importance in the service sectors especially in banking field because customer satisfaction is directly linked with loyalty and goodwill of the organization ^{[8][9]}.

The following literature review aims to showcase existing approaches to identify customer satisfaction in the service industry in general. The use of machine learning techniques to improve

the customer relationship was prevalent especially in the FMCG retailers^[10], telecommunications^[11] and banking^{[12][13]}.

In most service-oriented industries, such as telecommunications, customer loss is a rare occurrence. In other words, the number of customers lost is much lower than those who continue to use services. From the perspective of a machine learning task, user satisfaction can be treated as a classification of a dichotomous variable. The variable responds to this problem has an unbalanced distribution of classes in which a minority is dissatisfied with services lost customers. The problem of class imbalances complicates the application of standard classification algorithms. The problem with unbalanced classes is that the classifier tends to have minority class observations incorrectly assigned to the majority class. In extreme cases, the classifier can assign all observations for the majority class. This way it achieves a high overall classification accuracy, however unacceptably low accuracy with respect to the minority class of interest. For example, if the model trained for a data set in which 1% is a minority class, then 99% accuracy can be achieved by simply classifying everything in the majority class^[14].

Haibo He et. Al^[15] in their paper provides different solution methods in better handling the problem of class imbalance such as sampling methods, cost-sensitive methods and kernel-based methods. The author also highlights the importance of standardized evaluation metrics such as singular assessment metrics (precision, recall, F-measure, and G-mean), receiver operating characteristics (ROC) curves, precision-recall (PR) curves and cost curves to properly assess the effectiveness of each model.

Modeling is found to be an effective approach in forecasting bank's customer satisfaction levels. Different types of modeling techniques were recorded in classifying bank customer satisfaction and were mainly classified into two main groups i.e., one group encompassing statistical models such as Structural Equation Models (SEMs) and regression models and another group including artificial intelligence models such as Artificial Neural Networks (ANN). The use of such statistical models has been portrayed in the following works^([16], [17], [18], [19], [20], [21], [22], [23],[24]). The limitation for statistical models is the lack of high prediction accuracy since the models have a linear structure whereas customer satisfaction has a highly nonlinear relationship with the influencing factors. In the meanwhile, ANN is an effective tool to model the behavior of nonlinear systems while having a simple structure that makes it computationally efficient. ANN has been utilized to predict customer satisfaction in non-banking organizations such as the auto industry, telecommunications, and healthcare services^[25]. The Performance of the developed ANN model was then compared with that of a linear regression model and results showed more than 73 % higher accuracy for the ANN model compared to the linear regression^[26].

Further, Stephan Dreiseitl et. al^[27], points out difference between several classifications models and highlights their advantages and disadvantages. The paper states that the white box models such as Decision trees, Logistic regression offer better interpretability whereas black box models such as SVM and Neural networks have better predictability. The authors have found that the performance of logistic regression and ANN is higher than KNN and decision trees in most of the experimental cases whereas SVM shows comparable results and logistic regression is popular mainly for its low generalization error.

Hossein Abbasimehr et.al^[28] stated that oversampling improved the prediction performance and different ensemble techniques significantly improved the predictive power of the base learners. The authors implemented two types of sampling & four types of ensemble techniques on the base learning algorithms. For, churn prediction tasks, boosting algorithms were found to be the best fit among bagging, stacking and voting techniques.

Isabelle Guyon et.al^[29] in their paper discusses the problem of dealing with thousands of variables in the dataset and discusses the importance of selecting variables that would improve the prediction and computational performance of the chosen classification algorithm. The authors give an overview of relevant feature selection techniques and highlights the fact that the implementation of them solely depends on domain knowledge and there is no unifying criterion. The author recommends the use of filter methods for preprocessing followed by wrapper/embedded techniques.

1.4 Main Tasks

This research is based on binary classification problem where the implementation of different python tools and libraries are done in different machine learning and data mining techniques. This research is mainly based on the utilization of different feature engineering and feature selection techniques to handle anonymous dataset with imbalanced classs distribution for machine learning. The different machine learning techniques includes the modeling of classifiers such as logistic regression, Multilayer perceptron, XGBoost, Random Forest and the usage of ensemble techniques such as voting and stacking classifier. The model is first trained on selected features and then checked for overfitting by using cross-validation and hyperparameter tuning methods and later tested on validation set. The performance of the model is measured on the accuracy score obtained from Receiver Operating Characteristics Area Under the Curve (ROC AUC). Comparison among several models is done and the best model is selected among all for predicting the customer satisfaction in unknown data set.

1.5 Research Structure

This research work consists of 6 chapters. The first chapter gives a brief introduction of our research. It also gives an overview on research objectives, related works, and the research contents.

Chapter 2 explains about the fundamentals of our research and provides theoretical background of our research contents.

Chapter 3 gives an overview of understanding our data through exploratory data analysis and visualization.

Chapter 4 describes the implementation part in sequence. It gives an insight on model training, testing and explains the techniques and code used in our work.

Chapter 5 shows the results of our implementation part and several experimentations. It also compares among the results obtained in different experimentation parts.

Finally, Chapter 6 is the last chapter where the conclusion of whole research is presented and recommendations for future improvements are given.

Chapter 2 Fundamentals

2.1 Classification

Classification^[30] is a supervised machine learning process where a given dataset consisting target is categorized into different classes. It deals with both structured or unstructured datasets and involves in the prediction of the class of the given data instances. The classes are also named as either target, label, or categories. The classification modeling task uses a mapping function (f) from input training data's variables (X) to discrete output variables (Y) to forecast the likelihood or probability of that data to fall into one of the predetermined categories.

There are two different types of supervised classification i.e., Binary classification and Multi-class classification. Binary classification results in only two outcomes whereas multiclass classification results in more than two class outcomes with each sample assigned to one and only one label or target.

Classification consists of two major processes i.e., training process and testing process. In the training phase, the model is developed based on given training data and during the prediction phase, the model is used to predict the response for given data. There are many kinds of classification algorithms available for binary classification, also known as classifiers. The most used methods are logistic regressions, decision trees, random forest, support vector machines, neural networks, Bayesian networks etc.

In this work, we deal with supervised binary classification problem and the major classifiers used are described below:

2.1.1 Logistic Regression

Logistic regression is a widely used supervised learning classification algorithm used to forecast the probability of a target variable. It measures the relationship between a categorical dependent variable(Y) as log of odds and set of p independent variables $X_1, X_2 \dots X_p$ by estimating probabilities of occurrence using a logistic function(27). The nature of target or dependent variable is binary, which means there would be only two possible labels i.e., either Yes or No, 0 or 1, true or False, etc. The output is categorical or discrete value, but it does not give the exact value as 0 and 1, instead gives the probabilistic values lying between 0 and 1.

Logistic regression is an extended version of linear regression where linear regression model gives the predicted probability outcomes as any number ranging from negative to positive infinity, whereas probability of an outcome can only lie between 0 and 1 which is justified by logistic regression. Linear Regression is used for solving regression problems, whereas logistic regression is used for solving the classification problems. The dependent variable in logistic regression

follows Bernoulli Distribution and their estimation is done through maximum likelihood whereas using linear regression is done through Ordinary Least Squares (OLS) approach.

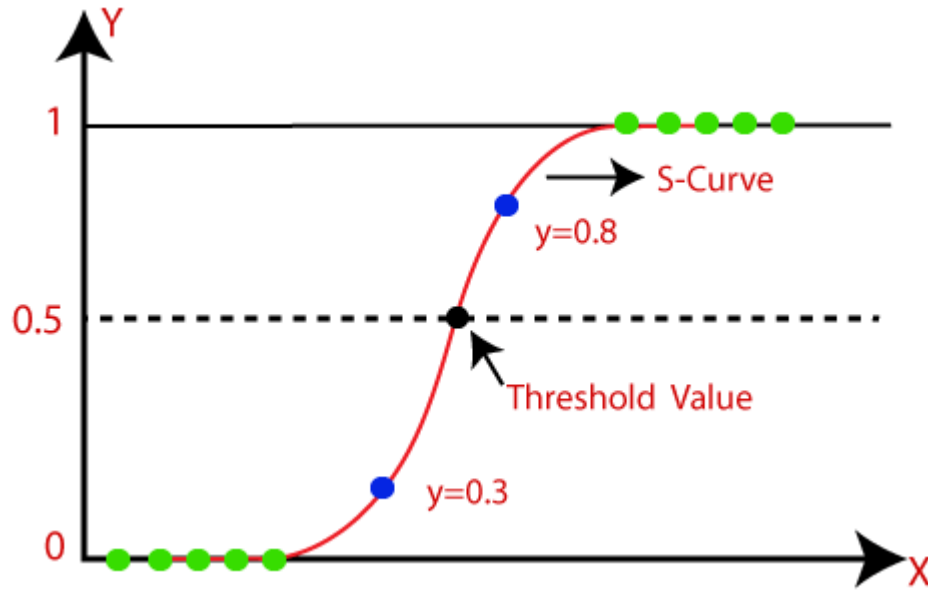


Figure 2.1 Linear Regression

Linear Regression Equation:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n \quad (2.1)$$

Where, Y is dependent variable and $X_1, X_2 \dots$ and X_n are explanatory variables.

Similarly, the probability that the output is 1 given its input can be represented as:

$$P(Y = \frac{1}{X}) \quad (2.2)$$

For logistic regression, sigmoid function is represented as:

$$P(Y) = \frac{1}{1 + e^{-Y}} \quad (2.3)$$

Applying sigmoid function on linear regression, we get the equation for logistic regression:

$$P(Y) = \frac{1}{1 + e^{-\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n}} \quad (2.4)$$

The sigmoid function or logistic function gives an ‘S’ shaped curve that maps real numbers to a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than let’s say 0.5, which is a threshold value, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO.

In this work, we deal with binary classification problem and as the logistic regression serves as one of the best predictive models for binary categorical dependent variables, it can be used as the baseline classification model for training our data.

2.1.2 MLP Classifier

Multi-layer perceptron (MLP) is a type of feed forward artificial neural network which comprises of three layers of perceptron—the input layer, output layer and hidden layer, with weight associated for each perceptron. The input layer receives the input signal for processing, output layer performs the task such as prediction and classification. There are multiple number of hidden layers in between the input and output layer and are the major computational core of the MLP. The MLP classifies datasets which are not linearly separable and are mostly used in solving pattern classification, recognition, prediction and approximation problems^[31].

Artificial neurons are the building block for neural networks. Layers are defined to be a row of neurons and one network can have multiple layers. The architecture of the neurons in the network is called as the network topology.

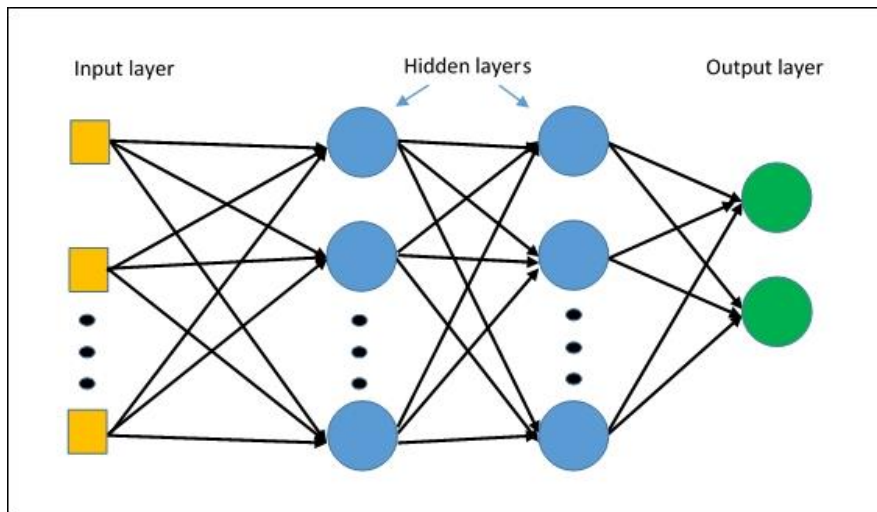


Figure 2.2 MLP Classifier

Neurons are basic computational units that have weighted input signals and produce an output signal using an activation function. An activation function is a simple mapping of summed weighted input to the output of the neuron. It is called an activation function because it governs the threshold at which the neuron is activated and strength of the output signal.

The data flows in the forward direction from input to output layer like a feed forward network and the neurons are trained with the back propagation learning algorithm and gradient descent to lower the final output error.

The final output error is given as:

$$E(n) = \frac{1}{2} \sum_j e_j^2(n) \quad (2.5)$$

Where e is the error at the n th data point

The weights can be written as:

$$weight = weight + learning_rate * (expected - predicted) * x$$

$$\Delta w_{ji}(n) = -\eta \frac{\partial \epsilon(n)}{\partial v_j(n)} y_i(n) \quad (2.6)$$

Where η represents learning rate and y_i denotes the output of the previous layer.

2.2 Decision Trees

Decision Trees are a type of non-parametric supervised predictive modeling tool widely used for both classification and regression problems^[30]. Decision trees are constructed using algorithmic methods which find ways to split a data set continuously based on different conditions. The decision rules are usually if-then-else statements. It creates model that forecast the value of a target variable by learning simple decision rules obtained from the data features.

Decision tree uses the tree representation following a top-down approach with decision node and leaf node as the main entities. Decision nodes or root nodes are from where the data is split and are used to make decisions and represents entire datasets, whereas Leaf nodes consists of the output of those decisions and have no further branches. Branches represent the decision rules and are formed by splitting the tree. The more branches, the tree gets more complex, and fitter is the model.

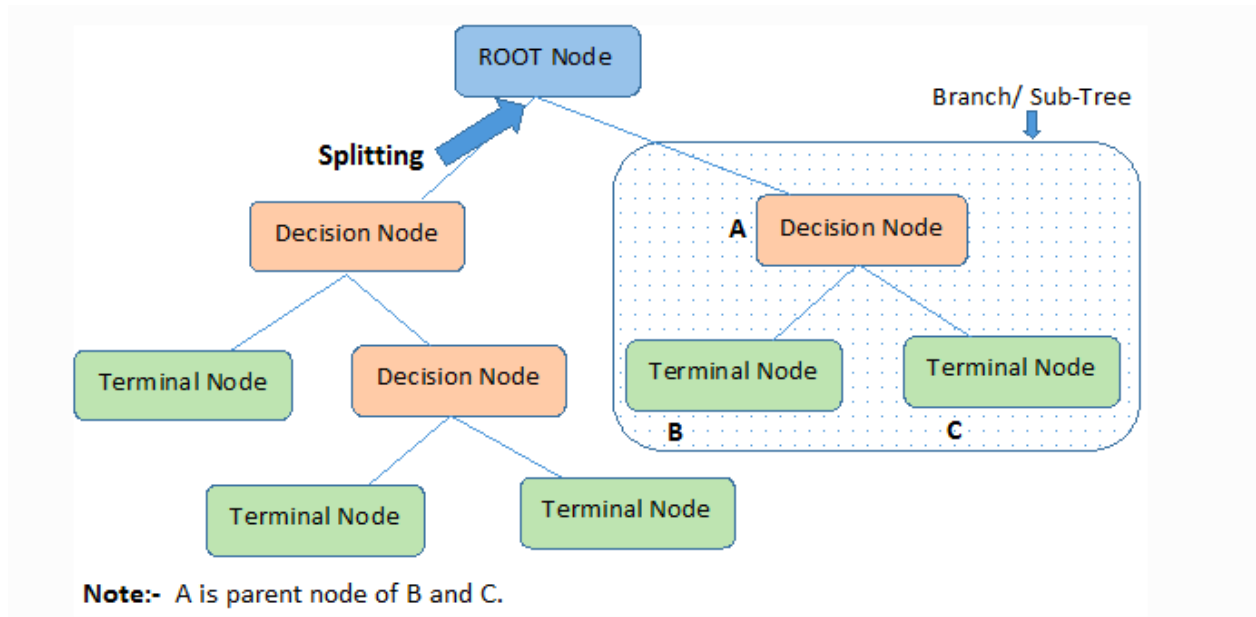


Figure 2.3 Tree representation of Decision Trees

Decision trees are widely used because of its speed in decision making and easy interpretability, however as our dataset consists of large number of features, the trees get complex with high chances of overfitting the model, reducing the ability of model to predict outcomes accurately. Thus, to avoid this issue, we use a combination of decision trees to make ensemble models for better predictive performance than just utilizing a single decision tree. The major logic

behind the ensemble model is that a group of weak learners come together to form a strong learner. Random Forest and Extreme Gradient Boosting, formed by combining different decision trees are used in our work as ensemble models.

2.3 Ensemble Modeling

Ensemble modeling^[32] is a machine learning method that uses the combination of two or more base models to produce one optimized predictive model. The main theme of ensemble learning is circles around the concept of crowds' wisdom, where many different independent decisions, choices or estimates are mixed into an outcome which is usually more accurate than any single method. Ensemble learning algorithms ensures optimization of the model, better robustness and better prediction ability as compared to that of single model.

There are many kinds of ensemble learning algorithms described as follows:

- Bagging:

Bagging or Bootstrap Aggregating combines Bootstrapping and Aggregation on homogenous weak learners to form one ensemble model and is used to reduce the variance of a decision tree. It involves creating different subsets of data from training set chosen randomly with replacement and then trains each subset in parallel way to combines them to get average result at last. Random forest uses bagging ensemble techniques.

- Boosting:

It involves the collection of homogeneous predictive models trained sequentially, where the model learns from the mistakes made by the previous model and creates a better predictive model. Gradient boosting algorithm is an example of boosting method.

- Stacking:

This involves the collection of heterogeneous weak learners to train them in parallel and combine using a meta-model to output a predictive model based on the different weak models' predictions. Voting classifier and stacking classifier falls under stacking ensemble technology.

2.3.1 Random Forest

Random forest^[33] is an ensemble of multiple decision trees, usually trained with the “bagging” method in a parallel fashion on various subsets of the given dataset, where the average of the models is taken into consideration to improve the predictive accuracy of that dataset. It solves the problems of overfitting occurred in a single decision tree.

Suppose we have a training set $X = X_1, \dots, X_n$ with outputs $Y = Y_1, \dots, Y_n$, bagging repeatedly (N times) selects a random sample with replacement of the training set and fits trees to these samples:

For $n = 1, \dots, N$:

Sample, with replacement, t training examples from X, Y ; call these X_n, Y_n .

Train a classification tree f_n on X_n, Y_n .

After training, predictions for testing samples X' can be done by averaging the predictions from all the individual trees on X' :

$$\hat{f} = \frac{1}{N} \sum_{n=1}^N f_n(X') \quad (2.7)$$

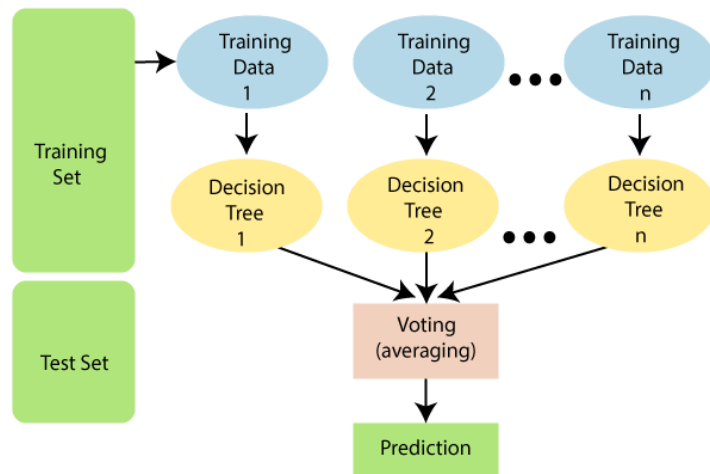


Figure 2.4 Representation of Random Forest

2.3.2 eXtreme Gradient Boosting

XGBoost or eXtreme Gradient Boosting algorithm is a scaled and improved implementation of the already existing gradient boosting decision tree algorithm that ensures optimized, fast and improved performance of the model. It involves the ensemble of multiple decision trees, built iteratively. It assigns weights using gradient (a direction in the loss function), which iteratively optimizes the errors made by model by updating weights. Loss usually refers to the difference

between the estimated value and actual value. The use of additive modeling ensures minimization of loss and the issues of overfitting^[34].

The task of training the model aims to find the best parameters that best fit the training data X_i and labels Y_i . The objective function to measure how well the model fit the training data is given as:

$$\min \text{Obj}(\Omega) = \text{Training Loss Function} + \text{Regularization}$$

$$\min \text{Obj}(\Omega) = \sum_{i=1}^n l(y_i, \hat{y}_i) + \alpha \sum_{i=1}^k |w_i| + \lambda \sum_{i=1}^k w_i^2 + \gamma T \quad (2.8)$$

Where REG_ALPHA is for L1 regularization, LAMBDA for L2 regularization, GAMMA as parameter for regularization that multiplies itself with the number of leaves. The regularization helps in avoiding overfitting by controlling the complexity of the model whereas the training loss measures how predictive is the model with respect to the training data.

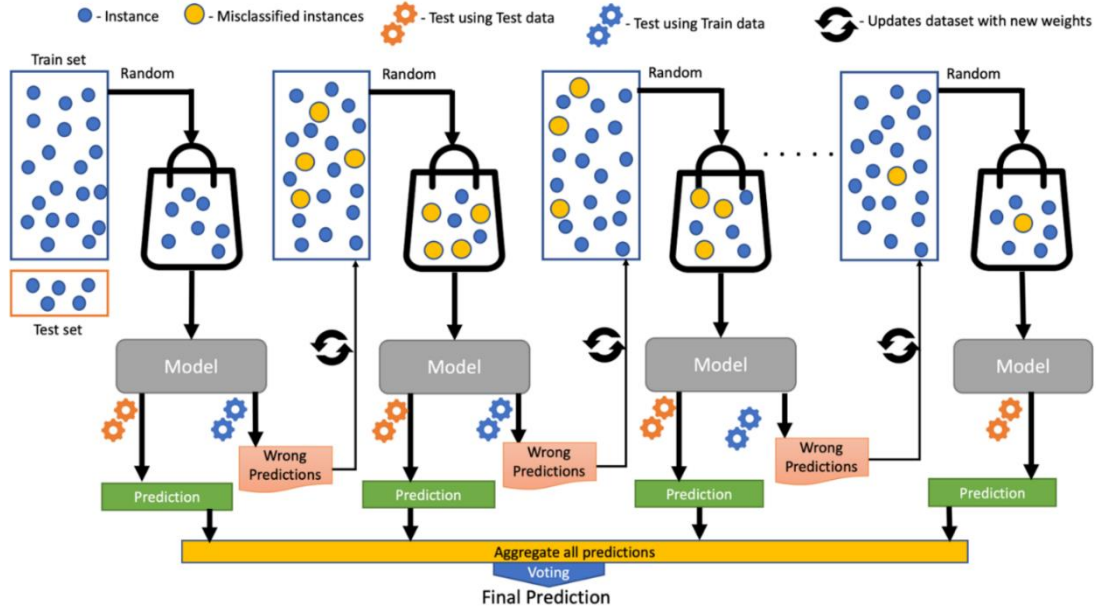


Figure 2.5 Final Prediction representation of XGBoost

2.3.3 Voting and Stacking Classifier

A Voting Classifier^[35] involves the ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It combines the outputs of each classifier passed into Voting Classifier and forecast the output class based on most of the voting. It follows the stacking methods where the heterogenous models are trained parallelly and the probabilities of each classifier to map the input data to certain class is averaged based on the weight specified to classifier. Then, the class with the highest probability is chosen.

Stacking^[36] is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training

set; then, the meta-classifier is fitted based on the outputs -- meta-features -- of the individual classification models in the ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.

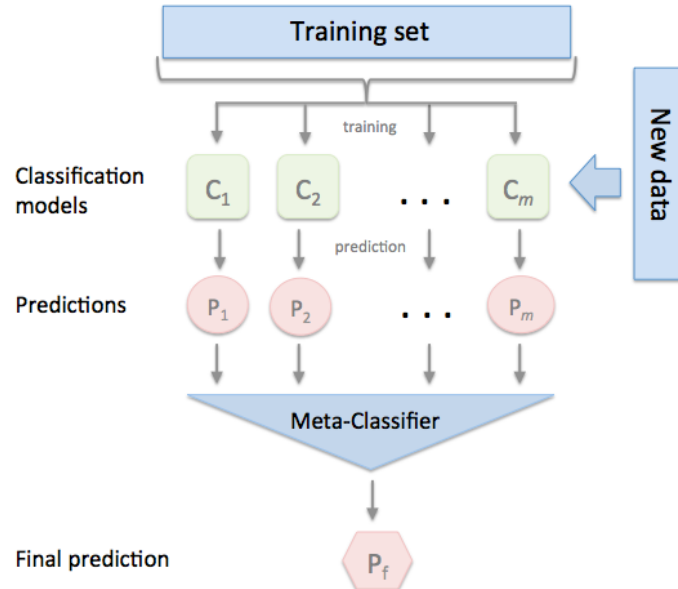


Figure 2.6 Stacking Classifier

The architecture of a stacking model consists of two levels i.e., base models, and a meta-model. Level-0 Models (*Base-Models*) gives prediction based on the fed training data whereas Level-1 Model (*Meta-Model*) is trained on the predictions of the base models and uses the data not used to train base models for prediction. To prepare the training dataset for the meta-model, k-fold cross-validation of the base models is commonly used.

2.4 Evaluation Metrics

The performance of the models is measured under the Area under the receiver operating characteristic curve, AUROC or AUC. The performance is measured in terms of accuracy.

2.4.1 Confusion Matrix

A Confusion matrix is an $N \times N$ matrix for evaluating the performance of a classification model, where N is the number of target classes. The matrix makes comparison between the actual target values and the predicted value. It is used in understanding how well the classification model is performing and what kinds of errors it is making.

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Figure 2.7 Confusion Matrix Representation

From the matrix, we can understand following concepts:

- True Positives: The cases in which we predicted 1 and the actual output was also 1.
- True Negatives: The cases in which we predicted 0 and the actual output was 0.
- False Positives: The cases in which we predicted 1 and the actual output was 0.
- False Negatives: The cases in which we predicted 0 and the actual output was 1.

2.4.2 Accuracy

Accuracy is one metric for evaluating classification models and is defined as the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{\text{Number of correct Predictions}}{\text{Total Number of Predictions}} \quad (2.9)$$

Accuracy can also be calculated in terms of positives and negatives as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.10)$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

2.4.3 Area Under the Curve (AUC)

The Area under the Receiver Operator Characteristic Curve (AUCROC) is an evaluation metric or a probability curve for binary classification problems where the True Positive Rate (TFR) is plotted against False Positive Rate (FPR) at various threshold points^[37].

$$TPR (\text{True Positive Rate}) / \text{Recall} / \text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{True Negative Rate (Specificity)} = \frac{TN}{TN + FP}$$

$$\text{False Positive Rate} = \frac{FP}{TN + FP}$$

$$\text{False Positive Rate} = 1 - \text{Specificity} \quad (2.11)$$

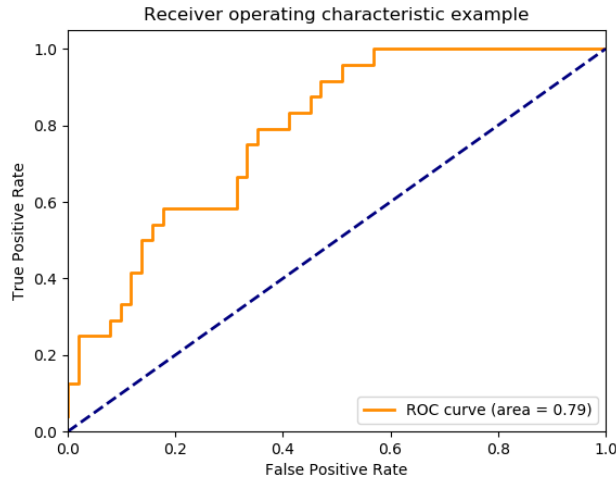


Figure 2.8 ROC AUC Curve

AUC value lies between 0 and 1. The performance of the model at distinguishing between the positive and negative classes is considered higher if the AUC value is higher.

AUC is not much affected by the imbalances in data and thus is preferred over other standard accuracy measure for classification problem.

2.5 Tools and Libraries used

The implementation of several analysis and modeling techniques in our research work is done by using Python programming language and its in-built libraries which are mentioned below.

- **NumPy** usually provides array objects of n-dimension. NumPy also provides mathematical functions for mathematical calculations.
- **pandas** are mainly used for data analysis purpose. Multi-dimensional arrays are taken as input for producing charts/graphs. pandas can read tables with columns of different datatypes and can read data from various data files and database like SQL, Excel, CSV etc.
- **matplotlib** is a library basically used in scientific plotting of data for visualization. It is mainly used for plotting histograms, pie charts, box plots, lines etc.
- **SKlearn** is mainly used in statistical modeling such as regression, clustering, classification and dimensionality reduction. It provides many different classes and modules for machine learning.
- **Jupyter Notebook** is used to write program code and is very interactive to display output and other multimedia for better explanation in a single file.

Chapter 3 Exploratory Feature Analysis

3.1 Data Source

The datasets are provided by Santander Bank, a Spanish multinational corporation bank and financial based company operating in Europe, North and South America, and Asia. The dataset are extracted from Kaggle.com(4) – a platform for predictive modelling and analytics competitions. The datasets include historical data of Santander Bank's customers and was made public through the means of Kaggle Competition to predict the customer satisfaction, conducted in the year 2016 with prize total worth \$60,000 for winners.

There are 3 different datasets i.e., train dataset, test dataset and dataset for sample submission with all 3 of the files in CSV format, a comma-separated values file, which allows data to be saved in a tabular format.

The train.csv contains data for training the models. Similarly, the test.csv contains data for testing the prediction models. Furthermore, sample_submissions.csv file is for examining the correct format in which we should upload our predictions to the Kaggle platform. It maps ID with predicted target value as either 0 or 1.

3.2 Feature Exploration and visualization

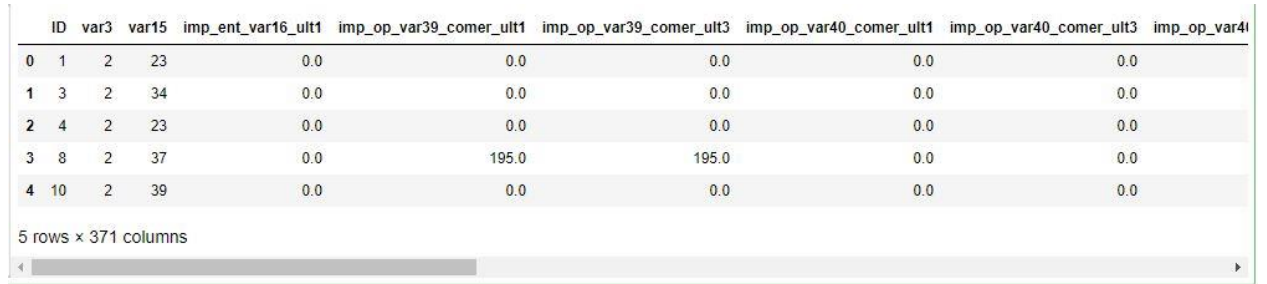
The features are highly anonymized for both test and train datasets. Thus, it is very unclear to understand what is represented by each feature and hence requires detailed exploratory data analysis for further analysis.

After importing the datasets using the read_csv() function of pandas library, the dataset is further analyzed using head() function of pandas to find out the number of observations in the dataset.

```
#Importing modules for data preprocessing
import pandas as pd
import numpy as np

#importing modules for visualization
import seaborn as sns
import matplotlib.pyplot as plt
# Loading the training and test datasets
train = pd.read_csv("Data/train.csv")
test = pd.read_csv("Data/test.csv")
```

```
#Seeing whats inside the training set
train.head()
```

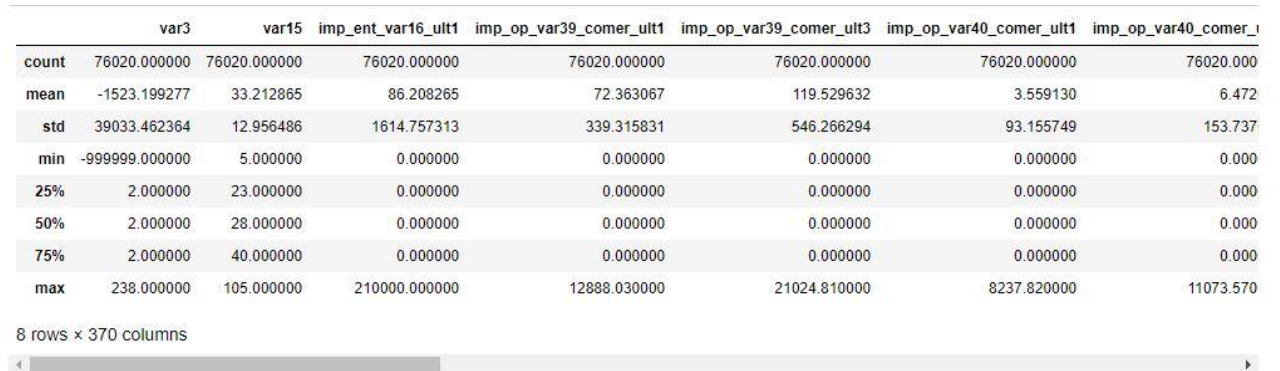
	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1	imp_op_var40_comer_ult3	imp_op_var40_comer_ult4
0	1	2	23	0.0	0.0	0.0	0.0	0.0	0.0
1	3	2	34	0.0	0.0	0.0	0.0	0.0	0.0
2	4	2	23	0.0	0.0	0.0	0.0	0.0	0.0
3	8	2	37	0.0	195.0	195.0	0.0	0.0	0.0
4	10	2	39	0.0	0.0	0.0	0.0	0.0	0.0

5 rows x 371 columns

Figure 3.1 Displaying columns

The figure 3.1 explains that the dataset consists of 371 features including the ID and target variable.

```
#Dataset description after removing ID and target
train.describe()
```



	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_var40_comer_ult1	imp_op_var40_comer_ult3
count	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000	76020.000000
mean	-1523.199277	33.212865	86.208265	72.363067	119.529632	3.559130	6.472
std	39033.462364	12.956486	1614.757313	339.315831	546.266294	93.155749	153.737
min	-999999.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	2.000000	23.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	2.000000	28.000000	0.000000	0.000000	0.000000	0.000000	0.000
75%	2.000000	40.000000	0.000000	0.000000	0.000000	0.000000	0.000
max	238.000000	105.000000	210000.000000	12888.030000	21024.810000	8237.820000	11073.570

8 rows x 370 columns

Figure 3.2 Describing Features

Further, the figure 3.2 explains that the dataset consists of 76020 observations and 370 features or columns with 1 binary feature for target, after removing Id variable. It also represents the statistical distribution of different variables. The target variable classifies the satisfied customers as 0 and unsatisfied as 1.

The test dataset size is almost same as that of train.csv with 75818 observations and 370 features or columns without any target variable.

```
# Checking data types
train.dtypes

ID                int64
var3              int64
var15             int64
imp_ent_var16_ult1 float64
imp_op_var39_comer_ult1 float64
...
saldo_medio_var44_hace3 float64
saldo_medio_var44_ult1 float64
saldo_medio_var44_ult3 float64
var38             float64
TARGET            int64
Length: 371, dtype: object
```

Figure 3.3 Displaying Data Types

Further, the fig 3.3 explains that most of our data types is numerical with some as integer and some as float numbers.

3.3 Feature Dictionary

As the dataset is highly anonymous with very unclear information on what the feature represents, it is quite difficult to understand the meaning of each feature and thus it is difficult to know the importance of that feature on the target variable. To solve this, explorative analysis of datasets using domain knowledge was done to figure out the data definitions and following assumptions were made.

Since Santander is a Spanish bank, some of the variable names are found to be originated from Spanish words. After exploring Kaggle's discussion forum, a data dictionary was proposed by Andreu(4) based on his observations. Most of the variables had substring num, ind, saldo, imp, delta, meses etc. in their name.

The substring 'Num' is assumed to represent numbers, usually with values 0, 3, 9 and most commonly the multiples of 3. There are a total of 155 observations.

Similarly, the substring 'ind' is referred to be an indicator variable with all values to be either 1 or 0. The total observations is found to be 75.

The substring 'saldo' is derived from Spanish word meaning current balance for certain financial products. Altogether there were 71 observations for 'saldo' with a significant number of zeroes (0) as value, possibly referring unused or finished financial products. The values are normally numeric with the highest value as 6119500 and some values as negative numbers. Similarly, 'imp' refers to a Spanish word 'importe' and likely stands for 'amount'. There are altogether 63 observations.

The substring ‘delta’ refers to difference in the variables between two periods. There are 26 observations in total for delta. Similarly, ‘meses’ in Spanish signifies months. There are in total 11 observations. All meses and delta are taken as subgroup of ‘num’ variable.

Similarly, the substring ‘amort’ in Spanish signifies loan (amortization) in English. Other Spanish words like ‘corto’, ‘medio’ and ‘largo’ likely stands for minimum, average and maximum respectively. The substring ‘ult’ likely corresponds to the last period/value. Some variables have substring "hace", "n" which seems like the value "n" periods ago with hace meaning ago/before. Other substrings such as ‘efect’ seems to be derived from ‘efectivo’ meaning cash, ‘reemb’ referring refund is found to be derived from ‘reembolso’, ‘trasp’ derived from traspaso referring transfer and similarly ‘emit’ and ‘recib’ derived from ‘emitido’ and ‘recib’ referring emission and reception, respectively.

3.4 Individual Feature Analysis

Different features with usually short name like var3, var15, var38 etc. suggested that they are possibly not derived from other variables and thus were assumed to be an individual entity. Thus, to further explore and understand them, individual analysis of those variables was done and following conclusions were made.

3.4.1 Target variable

There is a huge class imbalance in datasets with 73012(96.04%) as satisfied customers and 3008(3.96%) as unsatisfied customers. The unsatisfied customers are represented as 1 and satisfied as 0.

	TARGET	Percentage
0	73012	96.043147
1	3008	3.956853

Figure 3.4 Displaying TARGET

The below pie chart obtained from pie() function of matplotlib library, displays the distribution of target variable.

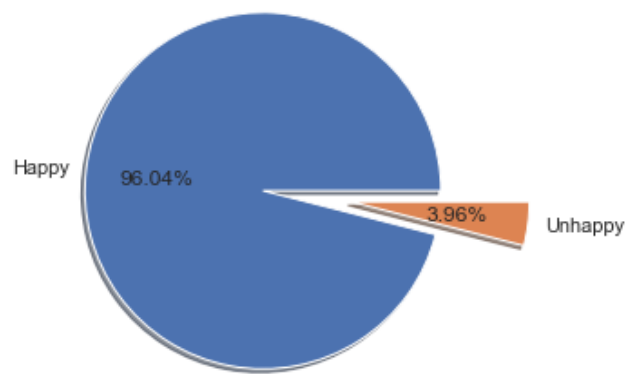


Figure 3.5 Piechart for TARGET VARIABLE

3.4.2 Var15

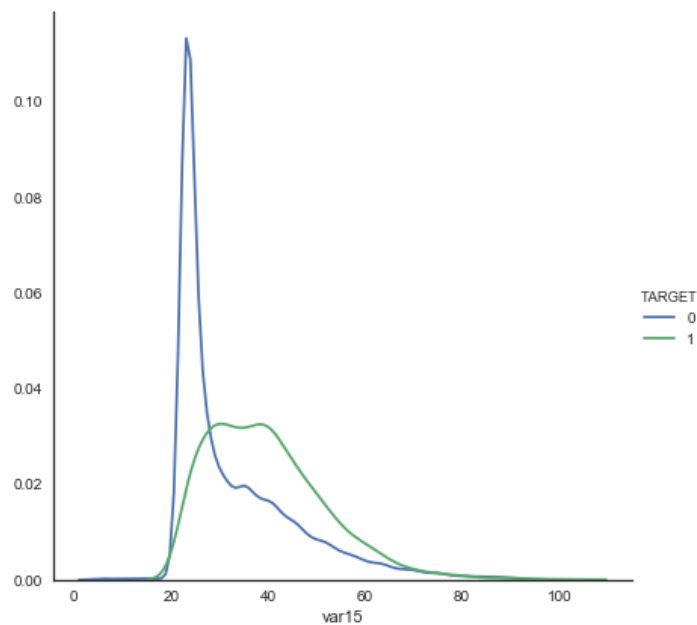


Figure 3.6 Displaying Var15

While analyzing the distribution of satisfied and unsatisfied customer in fig 3.6, var15 most likely represented the age of customers since the minimum and maximum values are found to be 5 and 105, respectively. However, most of the data were over 21. For satisfied customer, the most age common age group seem to be 23 and for unsatisfied ones, it seems to be above 40. This denotes that the young, aged customers are more satisfied whereas customers above 40 are found to be more dissatisfied.

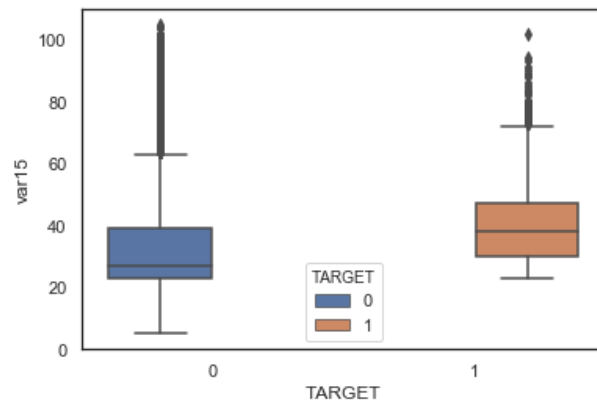


Figure 3.7 Displaying Var15 with respect to TARGET

The boxplot proves that the mean age of the happy customers is lower as compared to the unhappy ones.

3.4.3 Var38

The variable var38 when analyzed was found to be heavily skewed towards right with very less indication of satisfied customers.

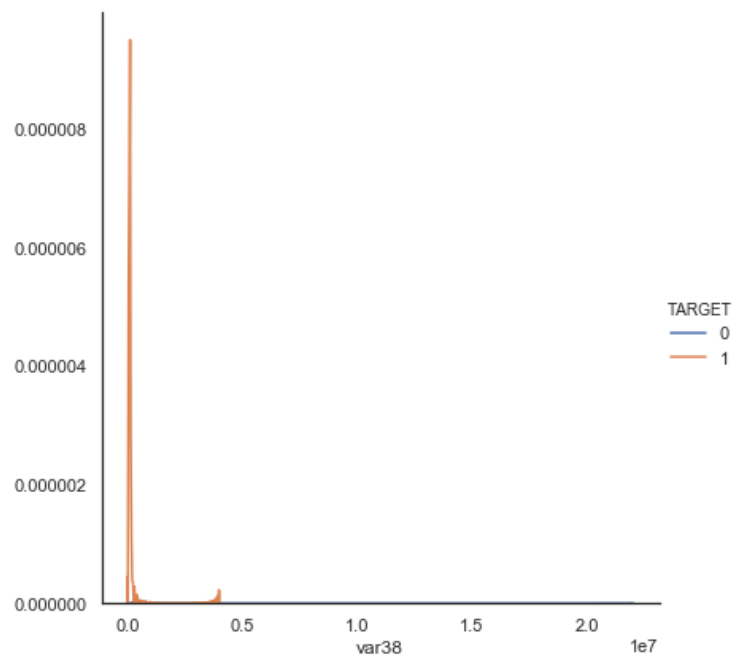


Figure 3.8 Describing Var38

Further, the statistics of the var 38 shows that the data type is float with very large number, which most likely represents the mortgage values.

Table 3.1 Analysis of var 38

Maximum	2.203474e+07
Minimum	5.163750e+03
Mean	1.172358e+05
Standard deviation	1.826646e+05
25%	6.787061e+04
50%	1.064092e+05
75%	1.187563e+05
Total count	7.602000e+04

Thus, to analyze in more detail, the logarithmic value of this variable was used instead. Later, a new feature called LogVar38 was generated from log of var 38 and was included in the dataset containing selected features. The histogram below shows the logarithmic distribution of Var38.

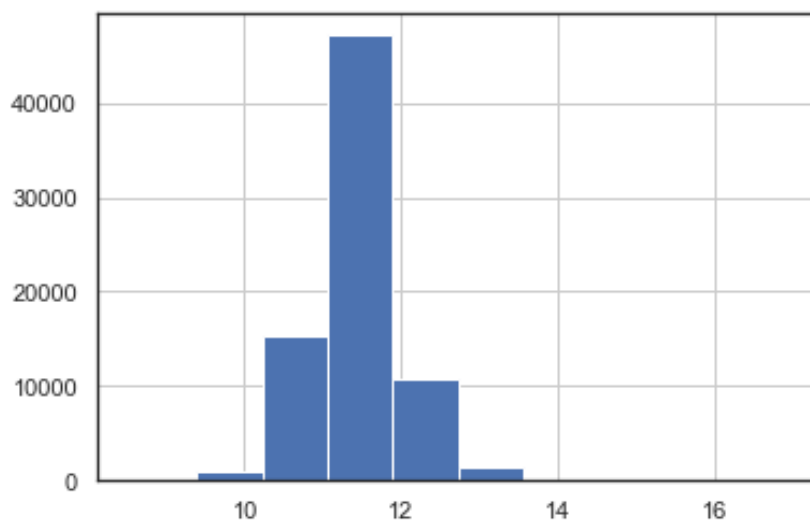


Figure 3.9 logarithmic distribution of Var38

3.5 Feature Groupings

The dataset is very large with 371 features which makes it difficult to analyze and discuss features individually, so instead different patterns were identified in the variable name and thus groupings of variables with common substrings were made. The major substrings present in the variable names were found to be ‘num’, ‘ind’, ‘saldo’, ‘imp’, ‘delta’, ‘meses’ etc. Several variables were found to have overlapping relations with each other.

For instance, `imp_op_var39_comer_ult1`, `imp_op_var40_comer_ult1`, `imp_op_var40_efect_ult1` seemed to be derived from a common base variable of `imp_op_comer_ult1`.

	original	var	base
0	var3	var3	
1	var15	var15	
2	imp_ent_var16_ult1	var16	imp_ent_ult1
3	imp_op_var39_comer_ult1	var39	imp_op_comer_ult1
4	imp_op_var39_comer_ult3	var39	imp_op_comer_ult3
5	imp_op_var40_comer_ult1	var40	imp_op_comer_ult1
6	imp_op_var40_comer_ult3	var40	imp_op_comer_ult3
7	imp_op_var40_efect_ult1	var40	imp_op_efect_ult1
8	imp_op_var40_efect_ult3	var40	imp_op_efect_ult3
9	imp_op_var40_ult1	var40	imp_op_ult1

Figure 3.10 Displaying imp_op related variables.

Similarly, the ‘meses’ category is supposed to be associated with ‘num’ features as it represents months which is most likely numeric value. Also, ‘delta’ features which represents the difference between two variables is most likely to be either ‘num’ or ‘imp’ category.

Thus, groupings of features according to common variable substrings helped categorizing important features and understand the pattern to further explore their relation and importance towards target variable.

Chapter 4 Implementation

4.1 Data Preprocessing

During exploratory data analysis, we found out that the data consists of large number of features and is highly imbalanced, causing noise for data training. Hence, different preprocessing techniques for data cleaning, data transformation and data reduction are implemented in order to train model for accurate prediction.

4.1.1 Data Cleaning

The dataset consists of 371 anonymous features including ID and Target variables. However, the dataset consists of a lot of noise and data cleaning ensures high quality data by removing errors, inconsistency, and duplicate data. The following measures were implemented to clean our data for accurate decision making:

- Removing ID variable: The ID variable acts as an index for all the observations in our data and is present in order. However, as we want our training data to not appear in a particular order but randomly. Thus, the ID variable is dropped, and the feature numbers are reduced to 370 from 371.
- Removing features with zero variance: There were some features found to have constant value of zero in all observations. Such features with zero variance or standard deviation are not useful for model training as they cannot teach anything to our model. So, these 34 variables were dropped and thus the number of features was reduced from 370 to 336.
- Removing duplicate or identical features: Duplicate features cause redundancy in our training data and thus create noise in model training. There was total 58 features found to be duplicate and among them only the first feature was kept, hence dropping 29 features. The total number of features then got reduced to 307 from 336.

The `drop()` function of Python's pandas library is used to remove these features.

4.1.2 Feature Engineering

Feature engineering is the process of defining and generating a new feature from raw, unstructured data to use in our statistical analysis.

The data seemed to contain a lot of zero value, which basically referred that the bank does not have information or lacks interaction with the customer, which makes it a difficult job to know if the customer is satisfied or not. However, as this value is high in numbers, it plays a significant

role in prediction of customers satisfaction. Thus, creation of a new feature called 'n0' was done to count the total number of zeroes appearing in the row.

4.1.3 Feature Selection

Feature selection is the process of selecting only the relevant features from the pool of large number of features to construct a machine learning model. Feature selection helps in simplifying the model to interpret by researchers/users, enhancing generalization by reducing variance, and lower the computation time by reducing the dimension of feature.

Though there are 307 features, but the target variable is dependent on only a few of them. In order to select only the most dependent variable and to reduce the dimension of feature, two different types of feature selection algorithms are used i.e., chi-square test and ANOVA-f test.

(1) Chi-square test

The `chi2()` function of `sklearn.feature_selection` module is used to perform chi-square test for feature selection. It measures the dependency between stochastic variables and removes the features that are independent on target variable, considering it as an irrelevant feature for model training.

The total number of features selected by `chi2` was 76 out of 307 features.

```
from sklearn.feature_selection import SelectPercentile
from sklearn.feature_selection import f_classif, chi2
from sklearn.preprocessing import Binarizer, scale

X_bin = Binarizer().fit_transform(scale(X))
selectChi2 = SelectPercentile(chi2, percentile=25).fit(X_bin, y)
selectF_ClassIf = SelectPercentile(f_classif, percentile=25).fit(X, y)
chi2_selected = selectChi2.get_support()
chi2_selected_features = [f for i, f in enumerate(X.columns)
                          if chi2_selected[i]]
print('Chi2 selected features {}, {}'.format(chi2_selected.sum(), chi2_selected_features))
```

(2) ANOVA F-Value

The ANOVA F-value test is performed using the `f_classif` class of `sklearn.feature_selection` module. The `f_classif` looks for and only considers linear relationships among features and target variable. The value is high for highly correlated features whereas the value is less for less correlated features.

A total of 77 features out of 307 were selected by `f_classif` class.

```
f_classif_selected = selectF_ClassIf.get_support()
f_classif_selected_features = [f for i, f in enumerate(X.columns)
                              if f_classif_selected[i]]
print('F_classif selected {} features : {}'.format(f_classif_selected.sum(), f_classif_selected_features))
```

(3) Combining the selected features:

At last, the features obtained from both chi2 and f_classif is combined and only the common features are further selected as the features important for training our model.

```
f_classif_selected = selectF_ClassIf.get_support()
f_classif_selected_features = [f for i, f in enumerate(X.columns)
                              if f_classif_selected[i]]
print('F_classif selected {} features : {}'.format(f_classif_selected.sum(), f_classif_selected_features))
```

There were altogether 60 features selected out of 370 total features as a combined selected feature.

4.1.4 Data Scaling

The StandardScaler() class of sklearn.preprocessing module is imported in order to standardize the data set using same range. StandardScaler standardizes the dataset by removing the outliers and by assigning the value of mean as 0 and standard deviation as 1.

The code shows the scaling of dataset containing the selected features.

```
# Scaling the columns in X_sel
from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
scaled_X_sel = ss.fit_transform(X_sel)
```

The output obtained shows the range for the selected features after scaling.

```
scaled_X_sel

array([[ -0.78824863, -0.01349292, -0.01553825, ..., -0.15086176,
         1.11465278, -1.61560274],
       [ 0.0607526 , -0.01349292, -0.01553825, ..., -0.1414472 ,
        -0.38424268, -1.20769065],
       [-0.78824863, -0.01349292, -0.01553825, ..., -0.15086176,
         0.2499054 , -0.65080256],
       ...,
       [-0.78824863, -0.01349292, -0.01553825, ..., -0.15086176,
         0.30755522, -0.48172311],
       [-0.63388477, -0.01349292, -0.01553825, ..., -0.15086176,
         0.2499054 , -0.25039843],
       [ 0.98693575, -0.01349292, -0.01553825, ..., -0.15086176,
         1.11465278,  0.33953018]])
```

Figure 4.1 Displaying scaled selected features.

4.1.5 Data Splitting

As the training data is highly imbalanced with 96% of satisfied target, this will create a problem in training the model and might incur overfitting. Also, the train set given by Santander does not contain target variable, so it can't be used to validate the classification result. Hence, the

training set is rather divided in the ratio of 70:30 respectively as train and validation set. The splitting is done in the scaled data containing selected features.

```
# Splitting into training and validation set
X_train, X_valid, y_train, y_valid = train_test_split(scaled_X_sel, y, test_size=0.30, random_state=7)
```

The classification model is fit using the training set with 70% of scaled data, obtained after splitting and later is tested in the validation set containing the remaining 30% of scaled data.

```
xgbClassifier.fit(X_train, y_train, eval_metric='auc')
print('Overall AUC on scaled validation set:', roc_auc_score(y_valid, xgbClassifier.predict_proba(X_valid)[:, 1]))
```

The code gives an example of using training and validation set for training and testing of the models.

4.2 Modeling, Parameter Tuning and Cross-Validating

4.2.1 Logistic Regression

Logistic regression is the baseline model for the accuracy measurement of all other models because of its simplicity and lesser computation time. The implementation of logistic regression in our work is done in several steps:

(1) Creating logistic regression model:

Logistic regression is the baseline model for the accuracy measurement of all other models because of its simplicity and lesser computation time.

The LogisticRegression class imported from sklearn.linear_model module was used to create a logistic regression model with certain values set as it's parameter.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

logisticReg = LogisticRegression(penalty='l2', random_state=555,
                                max_iter=10000, tol=10, C=1.0,
                                solver='newton-cg')
logisticReg.fit(X_train, y_train)
```

The parameters play a great role in determining the accuracy of the model. 'Penalty' is used to specify the norm (L1 or L2) that is used in penalization (regularization). Similarly, 'tol' denotes the tolerance for stopping criteria, 'C' denotes the inverse of regularization strength, which should always be a positive float number. 'Solver' is used to decide which algorithm to use in the optimization problem. The 'newton-cg' solver handles L2 penalty only and is for multi-class problem. Similarly, 'random_state' refers to the seed of the pseudo random numbers used by random number generator while shuffling the data.

(2) Tuning hyperparameter and cross validating:

In order to validate the accuracy of predictive ability of the model, the K-fold cross validation technique was implemented using GridSearchCV() class, where the datasets were divided into 10 folds of training and testing datasets.

```
fold = KFold(len(y_train), n_folds=10, shuffle=True, random_state=555)
gs = GridSearchCV(log_grid_clf, grid, scoring='roc_auc', cv=fold)
```

Further, in order to tune the parameters, different range of values were given for the GridSearchCV() class to choose the best among the range by trying out all the possible combination of parameter's value. The best value among them was then chosen for model evaluation.

Table 4.1 Parameters for Logistic Regression

Parameters	Range	Values before tuning	Best Value after tuning
penalty	['l1', 'l2', 'elasticnet', 'none']	l2	l2
random_state	None	None	555
max_iter	100	10000	10000
tol	1e-4	0.1	10
C	1.0	1.0	1.0
solver	['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']	lbfgs	newton-cg

(3) Fitting model & comparing accuracy:

The model was then fit using fit() function into both training and test dataset and it's accuracy was measured in terms of roc_auc.

```
gs.fit(X_train, y_train)
```

The accuracy of logistic regression model is measured without tuning the parameter and without grid search cross validation and is compared with the accuracy obtained from the model with cross validation and tuned hyperparameter.

4.2.2 XGBoost

The implementation of XGBoost model follows several steps which is going to be described in this section.

(1) Creation of XGBoost Model:

At first, the creation of XGBoost model was done where the parameters were set to some certain values. The model was created using the XGBClassifier() function of sklearn library.

```
## Creating a basic XGBoost model
xgTrain = xgb.DMatrix(X_sel, label=y)
xgbClassifier = xgb.XGBClassifier(missing=9999999999,
                                  max_depth=7,
                                  n_estimators=700,
                                  learning_rate=0.1,
                                  nthread=4,
                                  subsample=1.0,
                                  colsample_bytree=0.5,
                                  min_child_weight=3,
                                  seed=1301)
xgbParam = xgbClassifier.get_xgb_params()
```

The parameters have a great importance and thus need to be tuned properly. N_ESTIMATORS and MAX_DEPTH have same function as that of Random Forest model. LEARNING_RATE limits the speed and preciseness of the model, where the lower value signifies more accuracy but higher computation time. SUBSAMPLE refers to the ratio of observations to be sampled randomly. COLSAMPLE_BYTREE stands for the numbers of features to consider during the random sampling for each tree. MIN_CHILD_WEIGHT denotes the minimum sum of weights of all observations in a child tree. SCALE_POS_WEIGHT deals with imbalance data.

(2) Cross-validation of model using Grid Search:

Moreover, the model was then checked for overfitting using cross-validation techniques. GridSearchCV() function of sklearn library used stratified K-fold cross validation method where the datasets were divided into 15 folds with each fold having the same proportion of target variable.

```
# Cross validation
print('Starting cross validation')
cvResult = xgb.cv(xgbParam, xgTrain, num_boost_round=5000, nfold=15, metrics=['auc'],
                  early_stopping_rounds=50, stratified=True, seed=1301)
```

(3) Hyperparameter tuning and finding best parameter:

Later, the parameters of XGBoost model were tuned using GridSearchCV() where all the combination of the parameter's value from the given dictionary were tried. The best value among them was then chosen for model evaluation. The model was then fit into both training and test dataset and it's accuracy was measured in terms of AUC.

Table 4.2 Parameters for XGBoost

Parameters	Range	Values before tuning	Best Value after tuning
learning_rate	[0.001, 0.01, 0.1, 0.5]	0.1	0.01
max_depth	[4, 5, 6, 7, 8]	7	7
min_child_weight	[4, 5, 6, 7, 8]	3	5
gamma	[i / 10.0 for i in range (0, 5)]	0	0.1

subsample	[i / 10.0 for i in range (6, 10)]	1.0	0.6
colsample_bytree	[i / 10.0 for i in range (6, 10)]	0.5	0.6
reg_alpha	[1e-5, 1e-2, 0.1, 1, 10, 100]	None	0.0095
n_estimator	[100,1000]	700	700

Table 4.2 (continued)

(4) Fitting the model into train and validation set:

It was then followed by fitting the model in the scaled data to measure its accuracy. Finally, the scaled data is further split into train and validation set and thus, the accuracy of the model was measured using the split dataset.

(5) Comparison of model before and after tuning

Finally, the accuracy of the model was further cross-validated without tuning the hyperparameter and the results were compared with the model containing tuned hyperparameters.

4.2.3 MLP classifier

The implementation of the Multi-Layer Perceptron (MLP) is done in several steps:

(1) Creating MLP model:

At first, the MLP model is created using the `MLPClassifier()` class imported from `sklearn.neural_network` module. The parameter of the model is set to some certain values.

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV

# Building a Basic MLP Classifier model
mlp_clf1 = MLPClassifier(activation='logistic', alpha=1.0, batch_size='auto',
    early_stopping=False,
    epsilon=1e-08,
    hidden_layer_sizes=(30,12), learning_rate='invscaling',
    learning_rate_init=0.1, max_iter=1000, momentum=0.9,
    nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
    solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
    warm_start=False)
```

The parameter ‘hidden_layer_sizes’ is used to set the number of layers and the number of nodes we want to have in the Neural Network Classifier. Each element in the tuple denotes the number of nodes at the i^{th} position where i is the index of the tuple. Thus, the total number of hidden layers in the network is denoted by the length of tuple. The value of this parameter is set to be 30 at first.

Similarly, the ‘solver’ parameter specifies the algorithm for weight optimization across the nodes. It is set as the default value of ‘adam’ which refers to a stochastic gradient-based optimizer. The value ‘lbfgs’ is used for optimization in the quasi-Newton methods.

Further, the parameter ‘random_state’ allows us to set a seed for reproducing the same results. The parameter ‘activation’ acts as the activation function for the hidden layers, which used set to ‘logistic’ uses the logistic sigmoid function. Similarly, the ‘alpha’ parameter denotes L2 penalty (regularization) and is set to default value of 0.001. ‘Learning rate’ is used to schedule weight updates, where ‘constant’ is a constant learning rate given by ‘learning_rate_init’ and ‘invscaling’ is used to gradually decrease the learning rate at each time frame ‘t’ using an inverse scaling exponent of ‘power_t’. The parameter ‘max_iter’ denotes the maximum number of iterations.

(2) Tuning hyperparameter and cross-validating

Cross-validating the model plays a significant role in tuning hyperparameters to select the best among the range of parameters. For this, GridSearchCV() is used where k-fold cross validation is performed using 5 folds of data given as cv = 5. This splits the trainset into 5 different folds with same amount of data in each split.

```
param_grid_mlp = {
    "alpha":10.0 ** -np.arange(-4,7)
}
mlp_grid= GridSearchCV(estimator=mlp_clf,param_grid=param_grid_mlp,
                        cv=5)
```

Further, a range of values to choose from is given to parameters where the GridSearchCV() module search for the best parameter by checking all the possible values for the dataset. The best parameter is then used to train the model and checked for its accuracy level.

Table 4.3 Parameters for MLP Classifier

Parameters	Range	Values before tuning	Best Values after tuning
activation	['identity', 'logistic', 'tanh', 'relu']	logistic	logistic
alpha	10.0 ** -np.arange(-4,7)	0.0001	1.0 (Manually changed from 1000.0)
batch_size	'auto'	auto	auto
early_stopping	[True, False]	False	False
epsilon	1e-8	1e-8	1e-08
hidden_layer_sizes	n_layers - 2	(30, 30, 30)	(30, 12)
learning_rate	['constant', 'invscaling', 'adaptive']	constant	invscaling
learning_rate_init	0.001	0.001	0.1
max_iter	200	200	1000

momentum	0.9	0.9	0.9
nesterovs_momentum	[True, False]	True	True
power_t	0.5	0.5	0.5
random_state	None	None	1
shuffle	[True, False]	True	True
solver	['lbfgs', 'adam']	'sgd', adam	lbfgs
tol	1e-4	1e-4	0.0001
validation_fraction	0.1	0.1	0.1
verbose	[True, False]	False	False
warm_start	[True, False]	False	False

Table 4.3 (continued)

In our work, the best value for alpha was given as 1000 by our GridSearch method with very high accuracy in terms of training dataset. However, on running the model on the validation set it showed a very low AUC. Thus, the value of alpha was changed manually to 1 for simplification in model training.

(3) Fitting the model and checking accuracy:

The model is checked for its' accuracy level before cross validating and the score is obtained in terms of roc_auc score. The fit() module is called to fit the module into training and testing dataset.

Further, after obtaining the best value for parameters from cross-validation, the model is again tested for its accuracy level by fitting in the train and test datasets.

4.2.4 Random Forest

The modeling of Random Forest model was done following several steps described below:

(1) Creating Random Forest model

RandomForestClassifier() imported from sklearn.ensemble library was used to create model for Random Forest. The hyperparameters were at first set to certain values.

```
rf_clf = RandomForestClassifier(random_state=None,
                               max_features='auto',
                               max_depth=6,
                               n_estimators=100,
                               criterion='entropy',
                               min_samples_leaf=1,
                               min_samples_split=3,
                               bootstrap=True)
```


Some of the hyperparameters of sklearn's built-in random forest function are `N_ESTIMATORS` to determine the number of trees in the forest, `MAX_FEATURES` which gives the maximum number of features to consider when determining a best split during the algorithm. `MAX_DEPTH` controls the depth of the decision tree. `MIN_SAMPLES_SPLIT` gives the minimum number of observations required in a node to be considered for splits. `MIN_SAMPLES_LEAF` determines the minimum number of leafs to split a node. `N_JOBS` limits the number of processors, where -1 denotes there's no limits of processor and 1 denotes there's only one processor. As the random forest models are trained in parallel, the more working processor ensures lesser computation time. `CLASS_WEIGHT` is used to work with imbalanced datasets and can be set to 'balanced'.

(2) Tuning parameters and cross-validating

The hyperparameter in random forest are important in increasing the predictive ability of the model or to increase the speed of the model and should be tuned properly.

Table 4.4 Parameters for Random Forest

Parameters	Range	Values before tuning	Best Value after tuning
<code>n_estimators</code>	[100,200]	100	200
<code>criterion</code>	['entropy']	entropy	entropy
<code>max_features</code>	["auto"]	auto	auto
<code>max_depth</code>	[6]	6	6
<code>min_samples_leaf</code>	[1, 2, 3]	1	1
<code>min_samples_split</code>	[2, 3]	3	3
<code>bootstrap</code>	[True, False]	True	True

Since, it is not easy to determine what set of parameters is optimal for a specific problem, we use the `GridSearchCV()` function with 5-fold stratified cross validation and measured the accuracy score under the area under the ROC curve to find the answer. This technique tries on all possible combinations of a set of predetermined parameter ranges and selects the best parameter for model training.

(3) Model fitting and measuring accuracy:

The model is then fitted on training and validation set and the accuracy was measured without tuning the hyperparameters. However, as the model's accuracy for training data was found to be higher than that of validation data, the model was again fit into dataset after tuning the parameters and cross-validating.

4.3 Model Ensemble

To further optimize the predictive ability of the classifier, different ensemble learning methods were implemented on already existing models. Such models were combined to form a meta-model to improve the performance of the model. Two major stacking techniques of ensemble were used which are described below:

4.3.1 Use of Voting Classifier

Voting classifier combines the probability of each classifier to predict the model and calculates the average of all models and the classifier with highest probability is given as the input. In our work, the parameter of the classifier is set as 'soft' which implies that it predicts the class with the largest average probability from models. Soft voting usually has better performance than hard voting because it gives more weight to highly confident votes.

(1) Creating Voting classifier

The VotingClassifier() class of sklearn.ensemble library is used to create a voting classifier model.

(2) Combining models

The model then combines other two classifier i.e., MLP classifier and XGBoost and assign weight of 1 and 9 respectively to each classifier. The average of the probability of class is calculated based on the assigned weight.

(3) Fit model and measure score

The model after ensemble is then fit into the training and validation dataset. The accuracy of the model is then calculated using roc_auc score on validation set and is compared with that of individual classifier.

```
#Voting Classifier using mlp, xgboost and randomforest
from sklearn.ensemble import VotingClassifier
mlp_clf2 = MLPClassifier(activation='logistic', alpha=1.0, batch_size='auto',
                        early_stopping=False,
                        epsilon=1e-08,
                        hidden_layer_sizes=(30,12), learning_rate='invscaling',
                        learning_rate_init=0.1, max_iter=1000, momentum=0.9,
                        nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
                        solver='lbfgs', tol=0.0001, validation_fraction=0.1, verbose=False,
                        warm_start=False)

clf_xgb2 = xgb.XGBClassifier(missing=9999999999,
                            max_depth = 4,
                            n_estimators=700,
                            learning_rate=0.01,
                            nthread=4,
                            subsample=0.6,
                            colsample_bytree=0.6,
                            min_child_weight = 6,
```

```
seed=1000)

vot_clf = VotingClassifier(estimators=[('mlp', mlp_clf2),
                                      ('xgb', clf_xgb2)
                                      ],
                          voting='soft',
                          weights=[1, 9])
```

4.3.2 Use of Stacking Classifier

The `StackingClassifier()` class of `mlxtend.classifier` library is used to create the stacking classifier at first.

The stacking classifier then uses the combination of XGBoost, `MLPClassifier` and the Voting Classifier model to form an ensemble of model. This technique uses two level of prediction, where the first-level prediction was done between XGBoost and voting classifier and the MLP classifier was then taken as the meta-classifier. The combination of prediction of two level is done to achieve the final prediction.

The model was then fit into training and validation dataset and its accuracy was measured in terms of `roc_auc` score.

```
## Stacking classifier with xgboost, mlpclassifier and the voting classifier model.

from mlxtend.classifier import StackingClassifier

stck_clf = StackingClassifier(classifiers=[clf_xgb2, mlp_clf2, vot_clf],
                             use_probas=True,
                             average_probas=False,
                             meta_classifier=mlp_clf2)

stck_clf.fit(X_train, y_train)

print('Overall AUC for stacking classifier:', roc_auc_score(y_valid, stck_clf.predict_proba(X_valid)[:,1]))
```

Chapter 5 Results

5.1 Performance Measurement

The performance of the predictive ability of all the models were calculated in terms of AUC score and following results were observed for each model under different criterion.

5.1.1 AUC score of Logistic Regression

Table 5.1 Logistic Regression Result Analysis

Dataset	Condition	AUC Score
Train Dataset of Scaled Dataset	Basic Logistic Regression model	0.7699440031650385
Train Dataset of Scaled Dataset	Tuned Model with GridSearch cross-validation	0.7916586445104051

5.1.2 AUC score of MLP classifier

Table 5.2 MLP Result Analysis

Dataset	Condition	AUC Score
Validation Dataset of Scaled Data	Basic MLP model	0.8071928035951591
Train Dataset of Scaled Data	Tuned model with $\alpha = 1000.0$	0.9603863644905476
Validation dataset of scaled data	Tuned model with $\alpha = 1.0$	0.8099035271919413

5.1.3 AUC score of XGBoost

Table 5.3 XGBoost Result Analysis

Dataset	Condition	AUC Score
Selected Features Dataset	No tuning (Basic Model)	0.8823285757055674
Train Dataset after splitting into Train and	No tuning (Basic Model)	0.8217468070645283

Validation Datasets		
Scaled Dataset	No tuning (Basic Model)	0.8844654657650273
Train Dataset of Scaled Dataset	Tuned Parameters	0.8563345867216041
Validation Set of Scaled Dataset	Tuned parameters	0.8254972052304289
Train Dataset of Scaled Dataset	No tuning	0.9452765918453078 (Over-fitting)
Validation Set of Scaled Dataset	No tuning	0.8062612221917893 (Over-fitting)

The basic XGBoost model without tuned hyperparameter gave an accuracy of 0.94 and 0.806 for training and validation set respectively, which clearly indicates overfitting with the accuracy score greater for training set as compared to validation set.

The accuracy obtained after tuning parameter's value using grid search cross validation for train set was found to be 0.85, which is lower as compared to that of previous model without tuning. However, the tuned model gave an accuracy of 0.82 for validation set which is better than that of previous one.

5.1.4 AUC score of Random Forest

Table 5.4 Random Forest Result Analysis

Dataset	Condition	AUC Score
Validation Dataset of Scaled Dataset	Grid Search CV without tuned Hyperparameters	0.7950716445012528
Train Dataset of Scaled Dataset	Tuned model with Best Parameters	0.8160131251674362

5.1.5 AUC score of Voting Classifier

The accuracy of the voting classifier formed by combining MLP classifier and XGBoost classifier gave an average value of the accuracy of both models i.e., 0.8261. The XGBoost with weight assigned as 9 had the accuracy value of 0.8264 on validation set.

5.1.6 AUC score of Stacking Classifier

The stacking classifier combines the MLP classifier, XGBoost and voting classifier as the ensemble model and gives the accuracy of 0.825, which is less as compared to that of voting classifier. Hence, for our problem, Voting classifier is preferred over stacking classifier.

5.2 ROC_AUC plot

The visual representation of the accuracy of XGBoost and Logistic regression model is represented through this receiver operating characteristics area under the curve where the accuracy is measured against true positive and false positive rate.

5.2.1 ROC_AUC of XGBoost

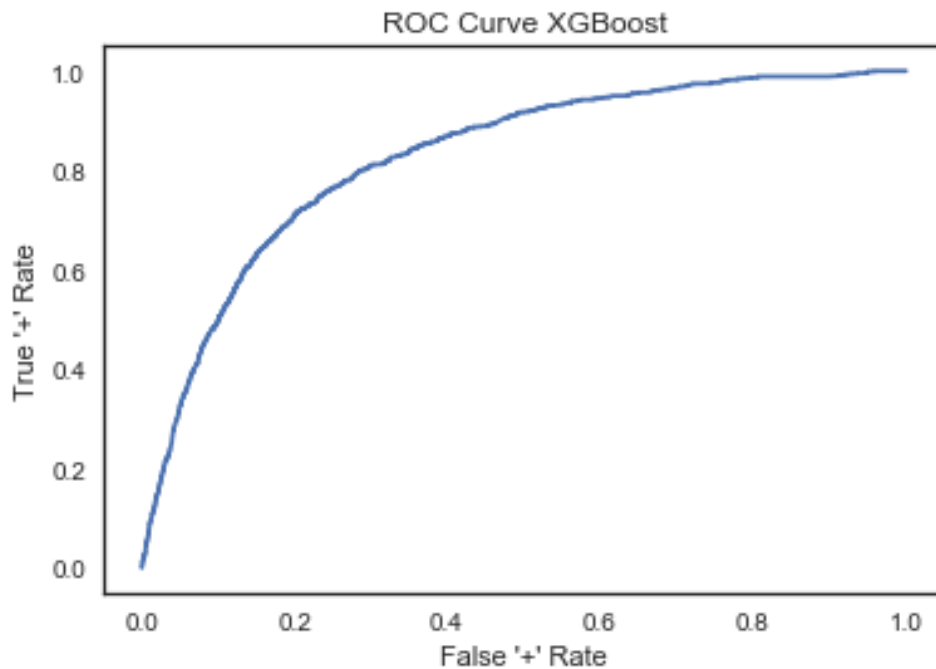


Figure 5.1 ROC AUC of XGBoost

The auc_roc score as plotted in this curve is 0.825.

5.2.2 ROC_AUC of Logistic Regression

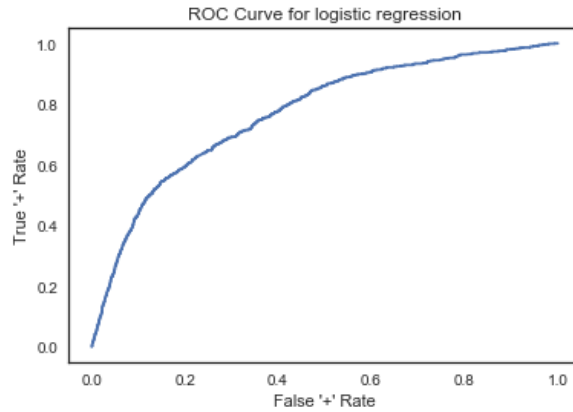


Figure 5.2 ROC Curve for Logistic Regression

The model has roc_auc score of 0.769 as plotted in the curve below.

5.3 Comparison of Accuracy

5.3.1 XGBoost, Random Forest and Logistic Regression

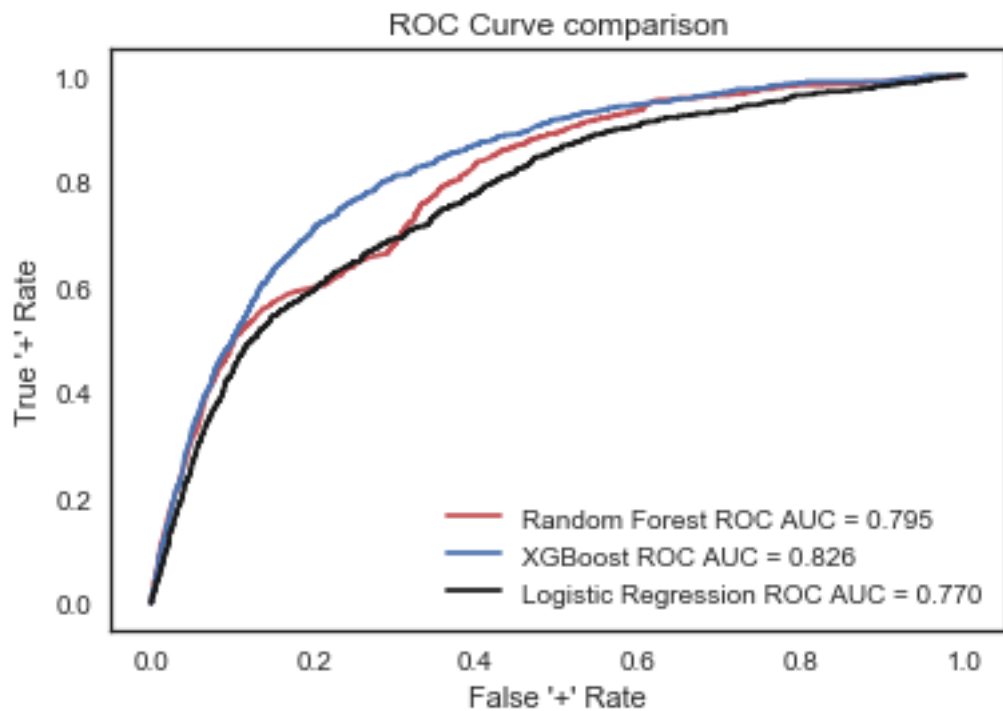


Figure 5.3 ROC Curve Comparison

The accuracy of different classifiers was plotted under the ROC Area under the curve and the comparison were made among them. XGBoost was found to be most effective model for predicting the class of target variable accurately with accuracy of 0.826.

5.3.2 Voting Classifier and Logistic Regression

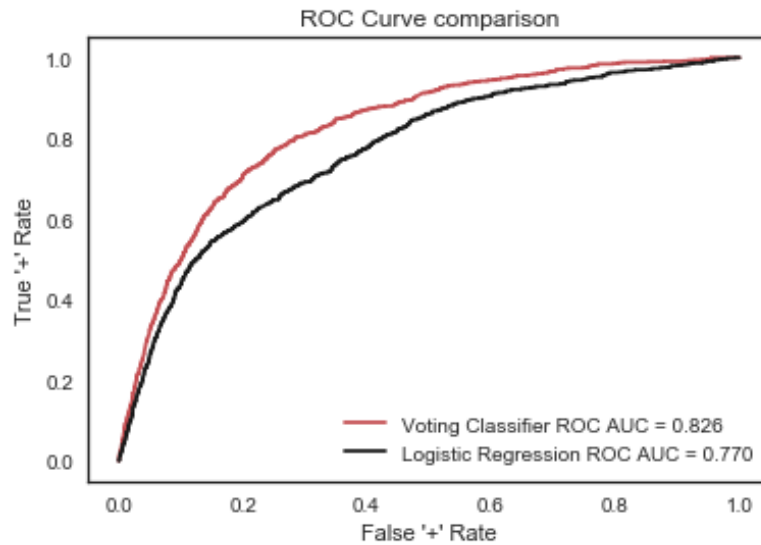


Figure 5.4 ROC Curve Comparison of Voting Classifier and Logistic Regression

As Logistic regression is considered as the baseline model for our dataset, it was used to compare the accuracy with voting classifier. The performance of voting classifier was found to be higher when compared with the logistic regression and hence can be considered as the best performing model.

5.3.3 Overall classifier

Table 5.5 Classifiers with Accuracy

Model	Accuracy on validation set
Logistic regression (Benchmark Model)	76.9%
MLP Classifier	80.7%
Random Forest	79.9%
XGBoost	82.54%
Voting Classifier	82.62%
Stacking Classifier	82.61%

While comparing among all the classifiers, the voting classifier was found to be the best effective model in terms of accuracy with the highest score of 0.8262.

5.4 Selection of important features

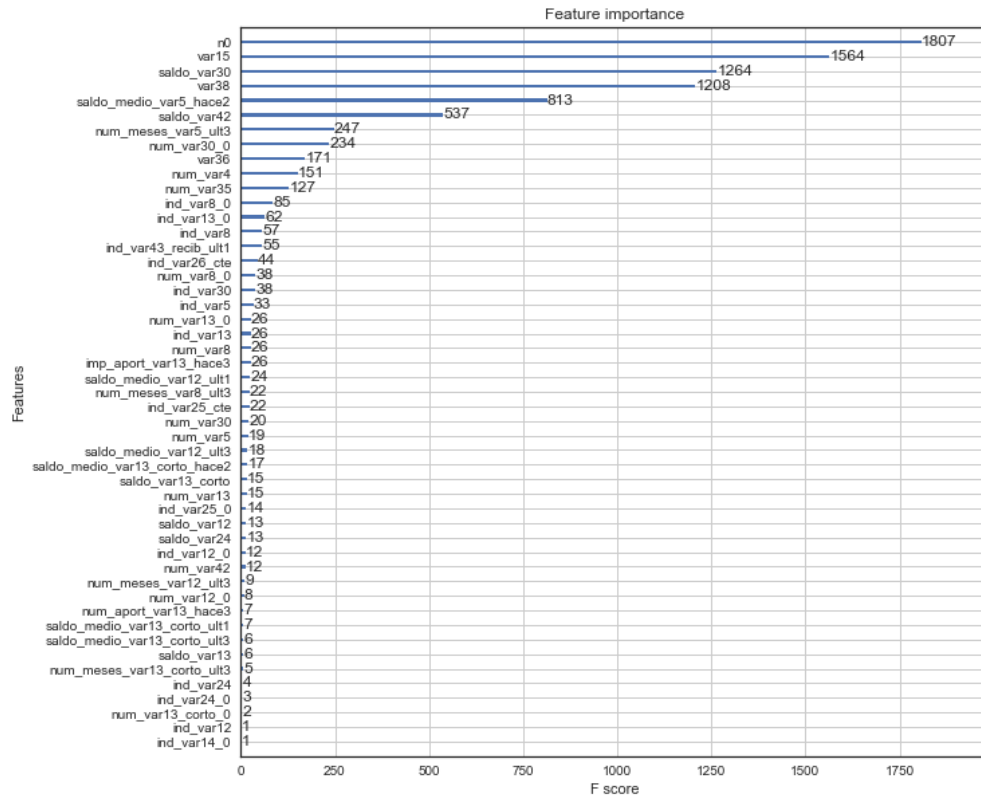


Figure 5.5 Displaying importance of Var 15

As XGBoost was the best performing model with very high accuracy, the features important for target variable were mapped against the F-score using XGBoost model. The result showed that the newly created feature called ‘n0’ which measures the sum of all zeroes observations in our data seemed to be the most important feature. Similarly, the Var15 which represents the age group also seemed to have high importance. Thus, the age of the customer was a major factor in determining the satisfaction towards the bank service. Moreover, Var38 which represented the mortgage value also seemed to be important feature for customer satisfaction.

5.5 Cross-validation of Voting Classifier

Finally, the voting classifier, which was the best performing model of all, was furthered cross-validated to check if the model is overfitted or not. The result showed that the model had the mean accuracy of 83.4% when cross-validated using 5 folds of data. The model only had the standard deviation of 0.003 which indicates that our model is fairly accurate in terms of classifying the customer as satisfied or dissatisfied.

```
1 from sklearn.model_selection import cross_val_score
2 from sklearn.model_selection import KFold

1 fold = KFold(5,shuffle=True,random_state=555)
2

1 scores = cross_val_score(vot_clf,scaled_X_sel,y,cv=fold,scoring='roc_auc')

1 scores.mean()
0.8343827732036913

1 scores.var()
8.913290326869497e-06

1 scores.std()
0.002985513410934457

1 scores
array([0.8324489 , 0.83255108, 0.83510804, 0.83188273, 0.83992311])
```

Figure 5.6 Cross-validation of Voting classifier

Chapter 6 Conclusion

This research work focused on anticipating if the customers of Santander Bank will be dissatisfied in the future by using several data mining and machine learning techniques. The large volume of dataset, anonymized features and huge imbalance in class served as the major challenge in understanding the data for further analysis. However, these issues were solved through exploratory analysis of data using domain knowledge in selecting only the relevant features and the implementation of techniques such as scaling and splitting of datasets to deal with imbalanced data. This is a binary classification problem where several Classification techniques such as Logistic regression, random forest, XGBoost, etc. were implemented using python tools and libraries. The classification models were cross-validated and tuned to select the best hyperparameter for model training and optimizing the accuracy. Finally, Comparison among several classification models were done to select the best performing model to predict the customer satisfaction.

In future, more exploratory data analysis and feature engineering could be done to select the most dependent features for training the data. Also, the research on the application of other models can also be performed and the combination of different models can be implemented in order to get more optimized accuracy score.

References

- [1] Schnaars S P. Marketing strategy: a customer-driven approach. New York: Free Press, 1991.
- [2] Munari L, Ielasi F, Bajetta L. Customer satisfaction management in Italian banks [J]. Qualitative research in financial markets. 2013, 5(2): 139-160
- [3] Hoq M Z, Muslim A. The role of customer satisfaction to enhance customer loyalty [J]. Eurasian journal of Business and Economics. 2009, 2(4): 139-154.
- [4] Kaggle. Santander customer satisfaction. 2016[2021-04-28]. <https://www.kaggle.com/c/santander-customer-satisfaction>.
- [5] Schiffman L G, Kanuk L L. Consumer Behavior. 8th ed. Prentice-Hall, 2004: 587.
- [6] Ndubisi N O. Understanding the salience of cultural dimensions on relationship marketing, it's underpinnings and aftermaths [J]. Cross Cultural Management: An International Journal. 2004, 11(3): 70-80.
- [7] Mittal V, Kamakura W A. Satisfaction, repurchase intent, and repurchase behavior: Investigating the moderating effect of customer characteristics [J]. Journal of marketing research. 2001, 38(1): 131-142.
- [8] Ravichandran K, Prabhakaran S, Kumar S A. Application of Servqual model on measuring service quality: A Bayesian approach [J]. Enterprise Risk Management. 2010, 2(1): 145.
- [9] Söderlund M. Customer satisfaction and its consequences on customer behaviour revisited: The impact of different levels of satisfaction on word-of-mouth, feedback to the supplier and loyalty [J]. International Journal of Service Industry Management. 1998, 9(2): 169-188.
- [10] Buckinx W, Van D. Customer base analysis: partial defection of behaviourally loyal clients in a non-contractual FMCG retail setting [J]. European journal of operational research. 2005, 164(1): 252-268.
- [11] Mozer M C, Wolniewicz R, Grimes D B, Johnson E, Kaushansky H. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry [J]. IEEE Transactions on neural networks. 2000, 11(3): 690-696.
- [12] Clemes M D, Gan C, Zhang D. Customer switching behaviour in the Chinese retail banking industry [J]. International Journal of Bank Marketing. 2010, 28(7): 519-546.
- [13] Xie Y, Li X, Ngai T, Ying W. Customer churn prediction using improved balanced random forests [J]. Expert Systems with Applications. 2009, 36(3): 5445-5449.
- [14] Zhu B, Baesens B, Backiel A, Vanden Broucke S K. Benchmarking sampling techniques for imbalance learning in churn prediction [J]. Journal of the Operational Research Society. 2018, 69(1): 49-65.
- [15] He H, Garcia E A. Learning from imbalanced data [J]. IEEE Transactions on knowledge and data engineering. 2009, 21(9): 1263-1284.
- [16] Kaura V. Antecedents of customer satisfaction: a study of Indian public and private sector banks [J]. International Journal of Bank Marketing. 2013, 31(3): 167-186;
- [17] Ibok N, John S. Investigating customer satisfaction driven values in the retail banking industry [J]. International Journal of Finance and Accounting. 2013, 2(5): 292-296.
- [18] Singh J, Kaur G. Customer satisfaction and universal banks: an empirical study [J]. International Journal of Commerce and Management. 2011, 21(4): 327-348.

- [19] Ravichandran K, Mani B T, Kumar S A, Prabhakaran S. Influence of service quality on customer satisfaction application of servqual model [J]. *International Journal of Business and Management*. 2010, 5(4): 117.
- [20] Herington C, Weaven S. E-retailing by banks: e-service quality and its importance to customer satisfaction [J]. *European journal of marketing*. 2009, 43(9): 1220-1231.
- [21] Manrai L A, Manrai A K. A field study of customers' switching behavior for bank services [J]. *Journal of retailing and consumer services*. 2007, 14(3): 208-215.
- [22] Amin M, Isa Z. An examination of the relationship between service quality perception and customer satisfaction: A SEM approach towards Malaysian Islamic banking [J]. *International Journal of Islamic and Middle Eastern Finance and Management*. 2008, 1(3): 191-209.
- [23] Hawari M, Ward T. The effect of automated service quality on Australian banks' financial performance and the mediating role of customer satisfaction [J]. *Marketing Intelligence & Planning*. 2006, 24(2): 127-147.
- [24] Moutinho L, Smith A. Modelling bank customer satisfaction through mediation of attitudes towards human and automated banking [J]. *International Journal of bank marketing*. 2000, 18(3): 124-134
- [25] Cui Q, Wang X, Li H, Kang X U. Using PCA and ANN to identify significant factors and modeling customer satisfaction for the complex service processes [C]. *Proceedings of the IEEE 18th International Conference on Industrial Engineering and Engineering Management*. IEEE, 2011: 1800-1804.
- [26] Zeinalizadeh N, Shojaie A, Shariatmadari M. Modeling and analysis of bank customer satisfaction using neural networks approach [J]. *International Journal of Bank Marketing*. 2015, 33(6): 717-732.
- [27] Dreiseitl S, Ohno L. Logistic regression and artificial neural network classification models: a methodology review [J]. *Journal of biomedical informatics*. 2002, 35(5): 352-359.
- [28] Abbasimehr H, Setak M, Tarokh M J. A comparative assessment of the performance of ensemble learning in customer churn prediction [J]. *Int Arab J Inf Technol*. 2014, 11(6): 599-606.
- [29] Guyon I, Elisseeff A. An Introduction to Variable and Feature Selection [J]. *Journal of Machine Learning Research*. 2003, 3: 1157-1182.
- [30] Breiman L, Friedman, Olshen, Stone C J. Classification and Regression Trees [J]. *Biometrics*. 1984, 40(3): 874.
- [31] Pedregosa F, Varoquaux G. Scikit-learn: Machine Learning in Python [J]. *Journal of Machine Learning Research*. 2011, 12: 2825-2830.
- [32] Lopez F. Ensemble Learning: Stacking, Blending & Voting. Medium. 2020 [2021-05-20]. <https://towardsdatascience.com/ensemble-learning-stacking-blending-voting-b37737c4f483>.
- [33] Breiman L. Random Forests [J]. *Machine Learning*. 2001, 45(1): 5-32.
- [34] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System [C]. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery, 2016: 785-794.
- [35] Afolabi S. Ensemble Methods: Comparing Scikit Learn's Voting Classifier to The Stacking Classifier. Medium. 2020 [2021-05-20]. <https://towardsdatascience.com/ensemble-methods-comparing-scikit-learns-voting-classifier-to-the-stacking-classifier-f5ab1ed1a29d>.
- [36] Brownlee J. Stacking Ensemble Machine Learning With Python. *Machine Learning Mastery*. 2020

[2021-05-20]. <https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/>.

- [37] Narkhede S. Understanding AUC - ROC Curve. Medium. 2021 [2021-04-09].
<https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>.

Acknowledgement

This thesis would not have been possible without the help of many people in so many ways. Firstly, I would like to thank my supervisor Professor Donghai Guan for giving me such a great opportunity to work in this interesting thesis topic. Without his constant guidance, this project would not have been possible. Further, I would like to thank my friends and family for their continuous support and encouragement. Finally, I would also like to appreciate College of International Education and College of Computer Science and Technology of Nanjing University of Aeronautics and Astronautics for providing such a research-oriented topic for thesis. Lastly, my sincere gratitude to my coordinator, Ma Shuai for her untiring efforts and support during my undergraduate studies.