CT046-3-M

Applied Machine Learning

Individual Assignment

Customer Churn Prediction in the Banking Industry Using Machine Learning Approaches

Student Name: Victor Hew Xin Kai

TP Number: TP078400

Intake Code: APUMF2310DSBA(DE)(PR)

Programme: MSc in Data Science and Business Analytics

Module Lecturer: Prof. Dr. Mandava Rajeswari

Table of Contents

Abstract

Customer churn is one of the main persistent problems every service provider in the banking industry needs to address as this will result in significant financial loss and a drop in brand reputation for the company if left unattended. This prompts the need for the prediction of bank customer churn prediction. Therefore, this project aims to use machine learning approaches for customer churn prediction tasks. The dataset used in this project revolves around customer data for a multinational bank in Europe. Data preprocessing techniques such as the creation of missing values and imputing them with median and mode as well as performing one hot encoding. Three baseline models are implemented using logistic regression, support vector regression and random forest followed by hyperparameter tuning on each model using techniques like L1 regularisation for logistic regression and grid search for support vector machine and random forest. As a result, random forest-based models, both baseline and tuned, produced the best performance across all models in terms of accuracy, precision, recall, F1-score and AUC value.

Keywords: churn, logistic regression, support vector machine, random forest

1.0 Customer Churn Prediction in the Banking Industry Using Machine Learning Approaches

1.1 Introduction

Undoubtedly, customers are prized assets for every company in the world, especially for the service providers in the banking industry as they are the main end-users of the services that directly contribute to the reputation and profitability of the company. Therefore, many bank service providers are shifting their focus towards existing customer retention aside from sourcing new customers for bank service subscriptions by making sure that they are satisfied with the services provided so that the likelihood of customer churn decreases (Lalwani et al., 2022; Singh et al., 2024). Churn in the banking industry context can be defined as a situation where bank customers cut ties with the bank by unsubscribing their services, mainly by closing their bank accounts and shifting towards the services of other bank competitors to satisfy their preferences and demands such as exceptional customer service experience (Singh et al., 2024). Customer churn can be detrimental to the financial side of the bank in terms of profit loss and bad brand reputation which hinders their progress in acquiring new customers whose decision-making process of bank service subscription depends on the bank's reputation and quality of service provided (Singh et al., 2024). Therefore, predicting customer churn is important as it can help banks to identify the motivation behind customers becoming churners and move on to improve their services and products, in addition to locating possible churners among the existing customer base and subsequently devising a personalised customer retention plan, which in turn give them a major competitive edge over their competitors in the banking industry. One of the proven ways for this prediction is machine learning approaches such as logistic regression and random forest which can predict churn outcomes according to historical bank customer data (Lalwani et al., 2022).

1.2 Aim, Objectives and Scope

Therefore, the purpose of this project is to utilise different machine learning approaches to implement predictive models for bank customer churn prediction. Thus, the following objectives are proposed for this project:

1. To better understand the data trend and pattern in the bank customer churn dataset through exploratory data analysis (EDA).
2. To implement machine learning-based models using approaches such as logistic regression (LR), polynomial support vector machine (SVM) and random forest (RF).
3. To perform hyperparameter tuning on the baseline models using L1 regularization for LR and grid search for SVM and RF.
4. To compare and validate the performance of the implemented baseline and tuned models.

In terms of the scope of this project, the bank customer churn dataset used in this project is sourced from Kaggle, which has a dataset usability rating of 10 and has more than 10 columns that are sufficient to carry out model implementation using RStudio.

2.0 Related Works

Past related work revolves around machine learning approaches to customer churn prediction mainly in the banking and telecommunication industries. Singh et al. (2024) wanted to forecast bank customer attrition using different machine learning techniques such as logistic regression, support vector machine, random forest and XGBoost. In terms of data preparation, the class imbalance issue for the target variable in the bank dataset was addressed using SMOTE. As a result, the performance of the model using random forest as the model foundation was greater than XGBoost based on both accuracy and recall metrics. Although the accuracy of XGBoost was the greatest among all implemented models, its percentage of sensitivity was the lowest, hence the recommendation of the random forest as the best-performing model due to the advantage in managing large datasets with multiple attributes for better capture of complex relationships between different attributes. The cash balance in customers' bank accounts as well as their nationality and gender are some of the most important variables in predicting customer attrition. Similarly, Tékouabou et al. (2022) used the same bank churn dataset to conduct the bank customer churn prediction task using KNN, ANN, DT and boosting-related techniques, in addition to the machine learning methods used by Singh et al. (2024). Their results suggested that random forest performed the best in terms of accuracy and F1-measure, which supported the findings from Singh et al. (2024). This also suggested that ensemble learning methods performed better than single classifiers in predicting churners and existing customers. Comparing the random forest model performance of these two studies, the 86% accuracy achieved using random forest in

Tékouabou et al. (2022) was greater than the 78.3% achieved by Singh et al. (2024) after class balancing using SMOTE, which indicated the role of tree numbers in varying the model accuracy of random forest. This was supported by the results from Muneer et al. (2022) who found that random forest achieved better accuracy and F1 score than SVM and adaptive boosting in forecasting the churn of bank customers after SMOTE's involvement in class balancing, although deep learning methods might be able to further extend the results. Besides, Kaur & Kaur (2020) also found that random forest had a better predictive edge than its base classifier along with KNN and logistic regression in predicting bank customer churn, which this performance was contributed by the implementation of stratified sampling and k-fold cross-validation. Moreover, Ullah et al. (2019) further investigated the implementation of a random forest-based model in forecasting attrition of telco customers with the additional help of unsupervised learning methods such as k-means clustering for profiling them. They found that the implemented model was able to better indicate the possibility of being a churner or non-churner using random forest compared to logistic regression and naïve Bayes.

Contrastingly, Pamina et al. (2019) investigated the importance of features that share the strongest relationship with telco customer churn using the IBM Watson telco customer dataset. As a result, XGBoost model exhibited greater performance than KNN and RF models based on accuracy and F-measure, and the most important churn factor was identified to be the total amount of charges imposed on customers who subscribed to fibre optic services from the telco provider. This result was supported by Lalwani et al. (2022) who found that boosting techniques such as XGBoost and adaptive boosting model outperformed other models that used logistic regression, SVM, Naïve Bayes and tree-based classifiers like decision tree and random forest.

Anitha & Sherly (2023) approached the churn prediction problem on the telecom and bank customer datasets with the implementation of the hybrid model which encompassed bagging and boosting methods. As a result, the hybrid model achieved superior predictive performance, beating the likes of single classifiers like logistic regression and decision trees as well as ensemble techniques like random forest and adaptive boosting. However, one weakness of this study revolved around the inability to address class imbalance issues in those two datasets, which might impact the reliability of the hybrid model. Besides, Khodabandehlou & Rahman (2017) used real-life Iranian restaurant data to predict churning customers who are non-frequent restaurant eaters using ANN, SVM and random forest along with their hybrid combinations with each other. They found that a hybrid approach could be one of the effective ways to predict customer churn, as indicated by the combination of ANN with SVM with radial kernel which exhibited the best model accuracy, in which this number could be further increased with the implementation of the boosting method on this model. In terms of single classifier, Karvana et al. (2019) cooperated with a large-scale Indonesian bank on the prediction of bank customer churn using machine learning approaches such as decision tree, neural network, SVM, Naïve Bayes and logistic regression. Using class stratified sampling method with a ratio of 50 to 50, the SVM model performed the best over others, although the inclusion of other possible important churn determinants like complaints and service satisfaction ratings from customers.

The following table provides an overview of the machine learning-based research done in the past relating to customer churn prediction in the food and beverage, telecommunications and banking sectors. Based on this table, it can be concluded that random forest, SVM, hybrid models and boosting-related techniques are commonly used in this forecasting context. However, there seemed to be a lack of implementation of hyperparameter tuning in building the model. Therefore, in this project, logistic regression, polynomial SVM and RF are used to build baseline models, followed by each of their tuned versions such as L1-regularised logistic regression and grid search tuned SVM and RF.

Authors	Dataset Used	Data Process	Classifiers	Accuracy	Precision	Recall	F1-Score	AUC
Singh et al. (2024)	Kaggle's bank customer churn dataset (10,000 observations, 14 variables) - **same dataset used in this assignment**	> SMOTE/No SMOTE > Train-test split > Data normalization > Grid search CV > Feature reduction > Feature engineering -variable creation	LR (no SMOTE)	79.30%	-	17.30%	26.30%	76.30%
			LR (SMOTE)	69.10%	-	71.40%	49.70%	76.70%
			SVM (no SMOTE)	80.20%	-	11.90%	20.50%	75.00%
			SVM (SMOTE)	71.90%	-	67.20%	50.50%	76.50%
			RF (no SMOTE)	84.40%	-	35.30%	49.10%	83.10%
			RF (SMOTE)	78.30%	-	69.30%	57.70%	83.10%
			XGB (no SMOTE)	85.20%	-	44.00%	55.90%	84.20%
			XGB (SMOTE)	83.90%	-	60.10%	61.30%	84.70%
Tékouabou et al. (2022)	Kaggle's bank customer churn dataset (10,000 observations, 12	> Feature reduction > Train-test split > SMOTE/no SMOTE	KNN (no SMOTE)	75.00%	-	-	12.00%	-
			KNN (SMOTE)	68.00%	-	-	70.00%	-

	variables) - **the same dataset used in this assignment**		SVM (no SMOTE)	80.00%	-	-	0.00%	-
			SVM (SMOTE)	57.00%	-	-	64.00%	-
			DT (no SMOTE)	79.00%	-	-	50.00%	-
			DT (SMOTE)	80.00%	-	-	80.00%	-
			LR (no SMOTE)	79.00%	-	-	9.00%	-
			LR (SMOTE)	67.00%	-	-	67.00%	-
			ANN (no SMOTE)	66.00%	-	-	10.00%	-
			ANN (SMOTE)	52.00%	-	-	54.00%	-
			NB (no SMOTE)	79.00%	-	-	13.00%	-
			NB (SMOTE)	72.00%	-	-	74.00%	-
			RF (no SMOTE)	86.00%	-	-	58.00%	-
			RF (SMOTE)	86.00%	-	-	86.00%	-

			Bagging (no SMOTE)	85.00%	-	-	54.00%	-
			Bagging (SMOTE)	84.00%	-	-	83.00%	-
			Gradient Boosting (no SMOTE)	87.00%	-	-	59.00%	-
			Grdadient Boosting (SMOTE)	84.00%	-	-	84.00%	-
			AdaBoost (no SMOTE)	86.00%	-	-	57.00%	-
			AdaBoost (SMOTE)	83.00%	-	-	83.00%	-
Pamina et al. (2019)	IBM Watsom telecom dataset (7031 observations, 21 variables)	Data type conversion	KNN	75.40%	-	-	49.50%	-
			RF	77.50%	-	-	50.60%	-
			XGB	79.80%	-	-	58.20%	-
Anitha & Sherly (2023)	Kaggle's telco dataset (7043 observations, 21 variables)	Data cleaning	LR	81.30%	66.60%	52.30%	58.60%	75.20%
			DT	78.70%	58.90%	51.80%	55.10%	71.60%
			RF	79.30%	61.50%	48.10%	54.00%	72.60%

			SVM	79.90%	64.50%	45.60%	53.40%	73.90%
			ANN	80.90%	65.40%	51.80%	57.80%	75.10%
			AdaBoost	80.20%	63.60%	50.70%	56.40%	74.00%
			Hybrid (Bag-of-Learners)	87.10%	77.30%	64.40%	70.20%	
			Associative Classification	84.40%	72.80%	61.80%	66.90%	86.10%
	Kaggle's bank dataset (10000 observations, 15 variables) - **an older version of the same dataset used in this assignment**		LR	78.10%	36.30%	4.70%	8.30%	57.80%
			DT	85.70%	71.20%	54.30%	61.60%	79.80%
			RF	86.20%	78.80%	47.50%	59.20%	83.00%
			SVM	77.50%	31.90%	5.40%	9.20%	55.60%
			ANN	86.30%	73.40%	55.50%	63.20%	81.10%
			AdaBoost	86.10%	75.00%	53.10%	60.90%	81.50%
			Associative Classification	89.50%	83.20%	63.70%	72.20%	87.30%
			Hybrid (Bag-of-Learners)	91.10%	85.30%	63.20%	73.00%	-
Karvana et al. (2019)	Indonesian XYZ Bank dataset (131548	> Data cleaning > Data type conversion	DT (Stratified)	91.58%	13.73%	41.05%	-	71.50%

	observations, 57 variables)	> Different sampling technique tested	DT (50:50)	69.04%	70.58%	16.39%	-	74.50%
			DT (30:70)	88.47%	39.99%	31.98%	-	75.90%
			NN (Stratified)	89.47%	34.61%	34.02%	-	78.00%
			NN (50:50)	70.79%	69.35%	17.07%	-	76.90%
			NN (30:70)	81.64%	50.60%	21.79%	-	76.70%
			SVM (Stratified)	92.65%	13.92%	68.41%	-	75.00%
			SVM (50:50)	73.68%	73.24%	19.39%	-	81.10%
			SVM (30:70)	89.17%	45.01%	35.65%	-	80.40%
			NB (Stratified)	79.65%	60.27%	21.79%	-	79.30%
			NB (50:50)	76.33%	65.72%	19.96%	-	79.50%
			NB (30:70)	78.58%	62.43%	21.20%	-	79.50%
			LR (Stratified)	82.18%	21.03%	52.05%	-	80.40%
			LR (50:50)	74.57%	72.68%	19.89%	-	81.50%
			LR (30:70)	89.65%	40.52%	36.41%	-	81.60%
Muneer et al. (2022)	Kaggle's bank customer churn dataset (10127 observations, 23 variables)	> SMOTE + under + oversampling > Data preporcessing	RF (no SMOTE)	63.70%	-	64.00%	63.00%	-
			RF (SMOTE)	88.70%	-	89.00%	91.00%	-
			AdaBoost (no SMOTE)	62.20%	-	62.00%	57.00%	-

			AdaBoost (SMOTE)	87.20%	-	87.00%	88.00%	-
			SVM (no SMOTE)	56.20%	-	75.00%	55.00%	-
			SVM (SMOTE)	77.60%	-	100.00%	89.00%	-
Khodabandehlou & Rahman (2017)	Actual data from an Iranian food store (1050 customers and 577200 records)	> Cleaning > Integration > Transformation > New variables created > Train-test split > Propose new framework RFMITDSP to predict	ANN-MLP (Boosting)	97.07%	98.15%	97.69%	97.92%	-
			ANN-MLP (Bagging)	93.81%	95.83%	95.39%	95.61%	-
			ANN-MLP	90.55%	94.34%	92.16%	93.24%	-
			ANN-RBF (Boosting)	94.46%	96.73%	95.39%	95.05%	-
			ANN-RBF (Bagging)	92.38%	95.35%	94.47%	94.91%	-
			ANN-RBF	88.60%	92.92%	90.78%	91.84%	-
			SVM Poly (Boosting)	93.48%	95.81%	94.93%	95.37%	-

			SVM Poly (Bagging)	93.48%	95.39%	95.39%	95.39%	-
			SVM Poly	88.92%	89.61%	95.39%	92.41%	-
			SVM RBF (Boosting)	92.83%	95.77%	94.01%	94.88%	-
			SVM RBF (Bagging)	91.53%	94.42%	93.55%	93.98%	-
			SVM RBF	85.67%	88.44%	91.70%	90.04%	-
			DT - C5.0 (Boosting)	89.90%	92.66%	93.09%	92.87%	-
			DT - C5.0 (Bagging)	86.64%	92.31%	88.47%	90.36%	-
			DT - C5.0	90.13%	87.50%	93.87%	85.65%	-
Ullah et al. (2019)	South Asia GSM telecom company (64107 observations, 29 variables)	10-fold CV applied	RF	-	89.30%	88.80%	88.20%	94.70%
			J48	-	90.20%	88.70%	88.00%	94.00%
			NB	-	71.50%	47.30%	45.60%	62.90%
			LR	-	49.00%	70.00%	57.70%	49.60%
	Publicly available churn dataset (3333 observations, 16 variables)		RF	-	89.10%	89.60%	87.60%	83.50%
			J48	-	90.60%	91.20%	90.40%	79.80%
			NB	-	86.60%	88.10%	86.80%	79.20%
			LR	-	81.00%	85.40%	80.90%	79.70%

Kaur & Kaur (2020)	Kaggle's bank customer dataset (28382 observations, 21 variables)	> Data preprocessing > Train-test split 70:30	LR (No stratified)	82.85%	69.34%	15.17%	-	76.66%
			LR (Stratified)	82.19%	59.87%	11.91%	-	73.19%
			LR (8-fold CV)	82.41%	57.97%	18.26%	-	75.18%
			DT (No stratified)	82.43%	64.61%	7.43%	-	49.60%
			DT (Stratified)	85.25%	72.30%	33.08%	-	77.15%
			DT (8-fold CV)	83.37%	60.18%	30.14%	-	73.81%
			KNN (No stratified)	81.29%	17.30%	7.80%	-	50.34%
			KNN (Stratified)	81.20%	33.33%	14.60%	-	74.89%
			KNN (8-fold CV)	81.03%	16.67%	6.10%	-	53.02%
			RF (No stratified)	84.79%	71.62%	41.09%	-	49.61%
			RF (Stratified)	85.21%	74.15%	43.09%	-	77.15%

			RF (8-fold CV)	84.35%	69.54%	41.70%	-	82.02%
Lemos et al. (2022)	Large financial institution in Brazil dataset (500000 observations, 35 variables)	> Data preprocessing > Feature selection > Repeated 10-fold CV	DT	78.20%	78.50%	-	78.05%	
			KNN	77.90%	79.20%	-	77.36%	
			Elastic net	76.20%	73.90%	-	77.29%	
			LR	76.20%	74.00%	-	77.26%	
			SVM	80.30%	81.00%	-	80.09%	
			RF	82.80%	84.40%	-	82.25%	
Lalwan et al. (2022)	Telco churn dataset (7000 observations, 21 variables)	> Gravitational search for feature selection > k-fold CV	LR	80.45%	79.11%	80.23%	78.89%	82.00%
			LR (Adaboost)	76.57%	56.61%	75.57%	64.71%	78.00%
			DT	80.14%	78.81%	80.10%	78.89%	83.00%
			Adaboost Classifier	81.71%	80.14%	81.21%	80.28%	84.00%
			Adaboost Classifier (Extra Tree)	81.14%	80.57%	81.64%	80.60%	72.00%
			KNN	79.64%	78.38%	79.71%	77.00%	80.00%
			RF	78.04%	77.54%	78.68%	77.91%	82.00%

			RF (Adaboost)	81.21%	80.19%	81.28%	80.29%	82.00%
			NB (Gaussian)	77.07%	77.60%	77.12%	77.31%	80.00%
			SVM Linear	79.14%	78.67%	79.89%	78.86%	79.00%
			SVM Poly	80.21%	79.66%	80.64%	78.11%	80.00%
			SVM (Adaboost)	74.07%	54.91%	74.43%	63.17%	80.00%
			XGBoost	80.80%	80.30%	80.70%	78.70%	84.00%
			CatBoost	81.80%	81.20%	82.20%	79.60%	82.00%

3.0 Methods

In terms of procedure, the customer churn records dataset is first read from the CSV file using RStudio and the data frame is named as Bank_Customer_Churn, hereinafter referred to as bank customer churn. Data preprocessing is done on this dataset which includes tasks such as variable reductions and creation and data type conversions followed by exploratory data analysis (EDA). Next, 20% missing values are randomly introduced to this dataset with no missing values initially, before mode and median imputations are done to handle the missing values. Later, one-hot encoding, class balancing, and min-max normalization are performed before the 70/30 train-test split is introduced. Six models are developed for bank customer churn prediction based on machine learning approaches such as logistic regression, lasso regression, polynomial support vector machine (SVM), random forest (RF) and tuned SVM and RF with grid search before proceeding to the presentation of final findings.

This bank customer churn dataset revolves around the data for multinational bank customer attrition across three European countries and all age groups. The source of this dataset is the Kaggle website (https://www.kaggle.com/datasets/radheshyamkollipara/bank-customer-churn/data). It contains 10,000 rows and 18 columns in total. Specifically, the number of character and numeric variables is six and 12 respectively, with 'Exited' being the target variable.

Variables	Data Types	Descriptions
RowNumber	Numeric	Row identifiers
CustomerId	Numeric	The unique ID number of bank customers
Surname	Character	The surnames of bank customers
CreditScore	Numeric	Scores on bank customers' creditworthiness
Geography	Character	The location where bank customer currently resides (France, Spain and Germany)
Gender	Character	Gender of bank customers (male and female)
Age	Numeric	Age of bank customers
Tenure	Numeric	The length of time, measured in years, the customer shares a relationship with the bank as a client
Balance	Numeric	Balance in bank customers' bank customer

NumOfProducts	Numeric	The number of bank products bought by bank customers
HasCrCard	Character	Whether the bank customer possesses credit card (0 = No and 1 = Yes)
IsActiveMember	Character	Whether bank customers are active bank members (0 = No and 1 = Yes)
EstimatedSalary	Numeric	The estimated salary of bank customers measured in euros
Exited (Target Variable)	Numeric	Whether the bank customer has unsubscribed the bank (0 = No and 1 = Yes)
Complain	Numeric	Whether the bank customer complain (0 = No and 1 = Yes)
Satisfaction Score	Numeric	Satisfaction ratings derived from complaint resolution made by bank customers (1 = very poor, 2 = poor, 3 = average, 4 = good, 5 = excellent)
Card Type	Character	The type of card possessed by bank customers (Silver > Gold > Diamond > Platinum in ascending order)
Points Earned	Numeric	The accumulation of points resulting from the usage of credit card

3.1 Data Preprocessing Techniques

Imputations are important as they help the replacement of missing values without sacrificing the size of the dataset through missing data deletion. There are different types of imputations, some of which are used in this assignment are mode and median imputations to deal with missing values in categorical and numeric variables respectively.

One hot encoding is used to manage categorical data encoding by splitting the levels in the categorical variable into different columns. For example, using the caret package, male and female levels of gender can be divided into two columns named gender_male and gender_female. If the data point belongs to a male, the value ‘1’ is assigned to the gender_male column or otherwise ‘0’, and the same goes for the female gender.

Class balancing is important as it strikes a balance in numbers between the two instances of the target variable through oversampling and undersampling methods. In this assignment, the technique of oversampling the minority class using the ROSE package is used. Min-max normalization is one of the two data scaling methods aside from standardisation so that the values fall between the range of 0 as the minimum and 1 as the maximum value.

3.2 Machine Learning Techniques

Logistic regression is one of the supervised machine-learning techniques used for classification tasks by utilising the built-in logistic function of the linear model to represent the rate at which an input under consideration is probably a member of a specific category, therefore making it suitable for the churn prediction task in terms of modelling probabilities of churn as well as simplicity. L1-regularised logistic regression is different from the baseline logistic regression as some arbitrary coefficient sizes of less important variables are being penalized under an L1 or Lasso penalty, shrinking them to exactly zero. It is suitable in this context due to the boost in the interpretability of the L1-regularised logistic regression model which prompts the identification of the most relevant churn factors.

Besides, in a high-dimensional region, polynomial SVM can be used to divide classes using the most efficient hyperplane. This method is chosen as the higher-order polynomial kernels can capture the non-linear associations between different attributes and whether the customer has unenrolled from the given program or services. The fourth technique revolves around tuning SVM with grid search followed by 10-fold cross-validation by performing an exhaustive search over a predefined grid of parameters for the best hyperparameter set using different value combinations of C and Gamma.

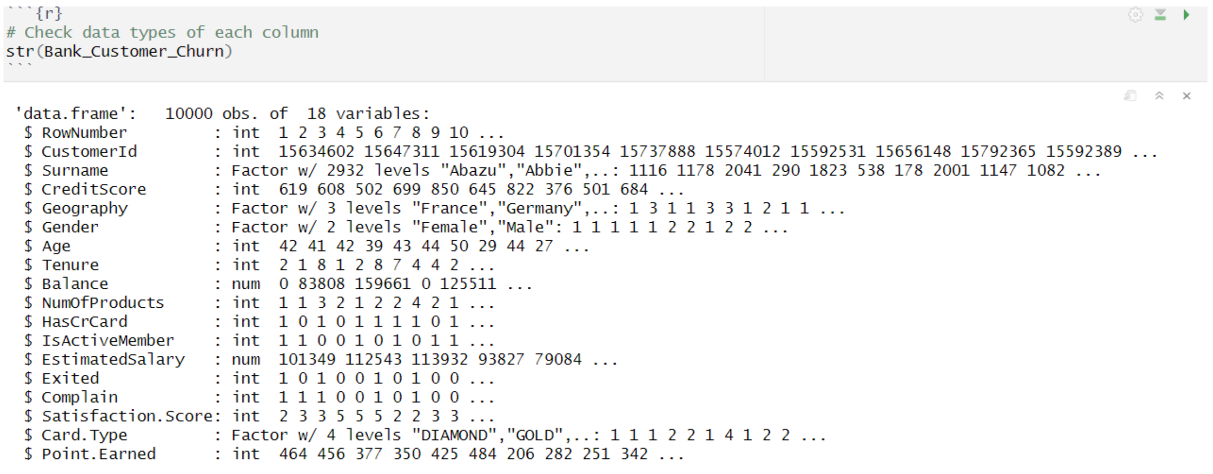
Just like a forest that is formed by a collection of trees, Random Forest is essentially made up of a collection of decision trees which can aggregate the forecasts generated by an individual decision tree and capture the complex, non-linear connections between multiple features and customer attrition with lowered overfitting risk. Identification of churn-related variable importance using this method is also feasible. The sixth technique revolves around grid search tuned random forest which tunes hyperparameters such as the number of trees and minimum samples for every split over a specified grid of parameters followed by 10-fold cross-validation, making sure that they are at an optimal model performance level that is good enough to generalise the model to forecast customer churn.

3.3 Performance Metrics

Customer churn prediction is a classification task, therefore classification-based performance metrics will be used in this study. The first one will be accuracy, which can be defined as the ratio of the number of correct churn predictions over the total number of churn predictions. However, one of the main weaknesses of this metric revolves around class imbalance through the prediction of the majority class of churn, so addressing class imbalance is important. The second metric is precision which revolves around the accuracy of forecasting the number of churned customers that have actually exited the service or program. The third metric is recall also known as sensitivity, which revolves around the ratio of the number of true positives over the combination of the number of true positives and false negatives. F1-score revolves around striking a balance between precision and recall scores in measuring model performance. Another metric can be the Receiver Operating Characteristic (ROC) Curve, which visualises the trade-off between sensitivity and false positive rate over a spectrum of thresholds.

4.0 Dataset Preparation

The bank customer churn dataset is first read from the original CSV file using RStudio, with all character variables being defined as factor variables at the beginning. An overview of the dataset in terms of the data types of each variable and their values or levels is shown in the figure below.

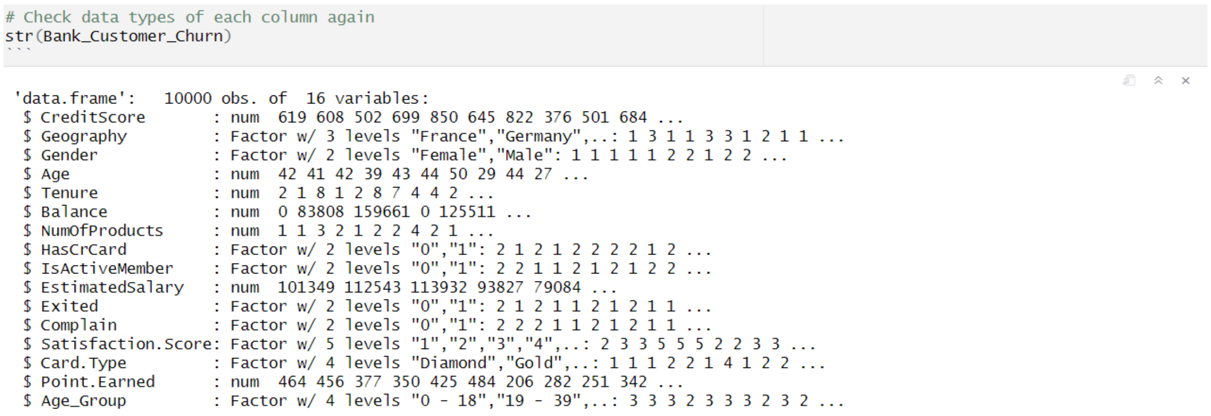


~~~~
```{r}
# Check data types of each column
str(Bank_Customer_Churn)
```
~~~~

```
'data.frame':	10000 obs. of  18 variables:
 $ RowNumber         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ CustomerId        : int  15634602 15647311 15619304 15701354 15737888 15574012 15592531 15656148 15792365 15592389 ...
 $ Surname           : Factor w/ 2932 levels "Abazu","Abbie",..: 1116 1178 2041 290 1823 538 178 2001 1147 1082 ...
 $ CreditScore       : int  619 608 502 699 850 645 822 376 501 684 ...
 $ Geography         : Factor w/ 3 levels "France","Germany",..: 1 3 1 1 3 3 1 2 1 1 ...
 $ Gender            : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 2 2 1 2 2 ...
 $ Age               : int  42 41 42 39 43 44 50 29 44 27 ...
 $ Tenure            : int  2 1 8 1 2 8 7 4 4 2 ...
 $ Balance           : num  0 83808 159661 0 125511 ...
 $ NumOfProducts     : int  1 1 3 2 1 2 2 4 2 1 ...
 $ HasCrCard         : int  1 0 1 0 1 1 1 1 0 1 ...
 $ IsActiveMember    : int  1 1 0 0 1 0 1 0 1 1 ...
 $ EstimatedSalary   : num  101349 112543 113932 93827 79084 ...
 $ Exited            : int  1 0 1 0 0 1 0 1 0 0 ...
 $ Complain          : int  1 1 1 0 0 1 0 1 0 0 ...
 $ Satisfaction.Score: int  2 3 3 5 5 5 2 2 3 3 ...
 $ Card.Type         : Factor w/ 4 levels "DIAMOND","GOLD",..: 1 1 1 2 2 1 4 1 2 2 ...
 $ Point.Earned      : int  464 456 377 350 425 484 206 282 251 342 ...
```

Figure 1: Initial overview of dataset

Data cleaning and preprocessing need to be done on the dataset before displaying its summary. First, meaningless variables such as RowNumber, CustomerId and Surname are dropped from the data frame as they are deemed unnecessary for model implementation. A new variable called Age_Group is created. Four levels of age group are derived based on the values from Age variable. For example, customers who are 18 years of age and below are considered as adolescents, with the age group being labelled as "0 – 18". Customers whose age are greater than 18 but less than or equal to 39 are considered as young adults, with the age group being labelled as "19 – 39". Customers whose age are greater than 39 but less than or equal to 59 are considered as middle adults, with the age group being labelled as "40 – 59". Customers whose age are greater than 59 are considered as late adults, with the age group being labelled as "60+". The values in Card.Type variables are converted from upper case to sentence or lower case for standardisation purposes. Unique rows are kept in the data frame to remove any possible duplicates. Conversion of data types is also done by converting variables with the wrong data type to their correct version. The figure below demonstrates the summary of the cleaned dataset, in which there are seven numeric variables and nine factor variables ultimately.



```
# Check data types of each column again
str(Bank_Customer_Churn)
```

```
'data.frame':	10000 obs. of  16 variables:
 $ CreditScore       : num  619 608 502 699 850 645 822 376 501 684 ...
 $ Geography         : Factor w/ 3 levels "France","Germany",..: 1 3 1 1 3 3 1 2 1 1 ...
 $ Gender            : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 2 2 1 2 2 ...
 $ Age               : num  42 41 42 39 43 44 50 29 44 27 ...
 $ Tenure            : num  2 1 8 1 2 8 7 4 4 2 ...
 $ Balance           : num  0 83808 159661 0 125511 ...
 $ NumOfProducts     : num  1 1 3 2 1 2 2 4 2 1 ...
 $ HasCrCard         : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 2 1 2 ...
 $ IsActiveMember    : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 1 2 2 ...
 $ EstimatedSalary   : num  101349 112543 113932 93827 79084 ...
 $ Exited            : Factor w/ 2 levels "0","1": 2 1 2 1 1 2 1 2 1 1 ...
 $ Complain          : Factor w/ 2 levels "0","1": 2 2 2 1 1 2 1 2 1 1 ...
 $ Satisfaction.Score: Factor w/ 5 levels "1","2","3","4",..: 2 3 3 5 5 5 2 2 3 3 ...
 $ Card.Type         : Factor w/ 4 levels "Diamond","Gold",..: 1 1 1 2 2 1 4 1 2 2 ...
 $ Point.Earned      : num  464 456 377 350 425 484 206 282 251 342 ...
 $ Age_Group         : Factor w/ 4 levels "0 - 18","19 - 39",..: 3 3 3 2 3 3 3 2 3 2 ...
```

Figure 2: Summary of the cleaned dataset

Figure 3 visualises the continuous variables in the form of histograms. The distributions of age and credit score are positive and negative skewed respectively. The distribution of balance is approximately normal. The distributions for estimated salary, points earned, and tenure is uniform.

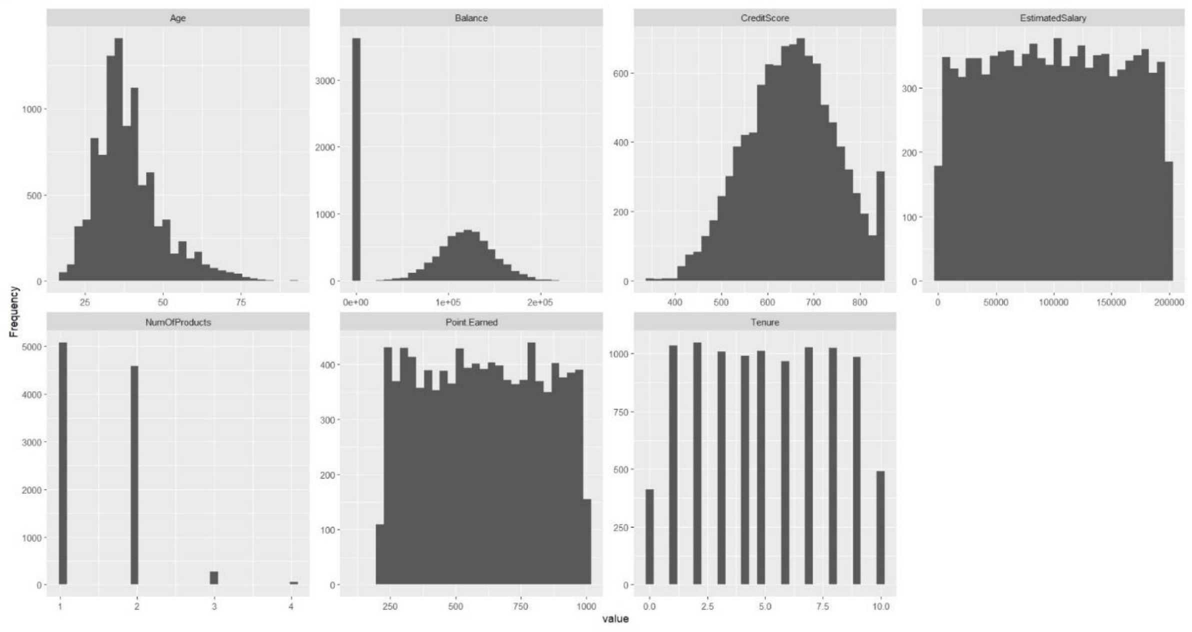


Figure 3: Histograms

Figure 4 visualises the factor variables in the form of horizontal bar charts. Most bank customers are males, in the young adulthood stage, live in France, has credit card, are active bank members, make no complaints to the bank and remain in relationship with the bank.

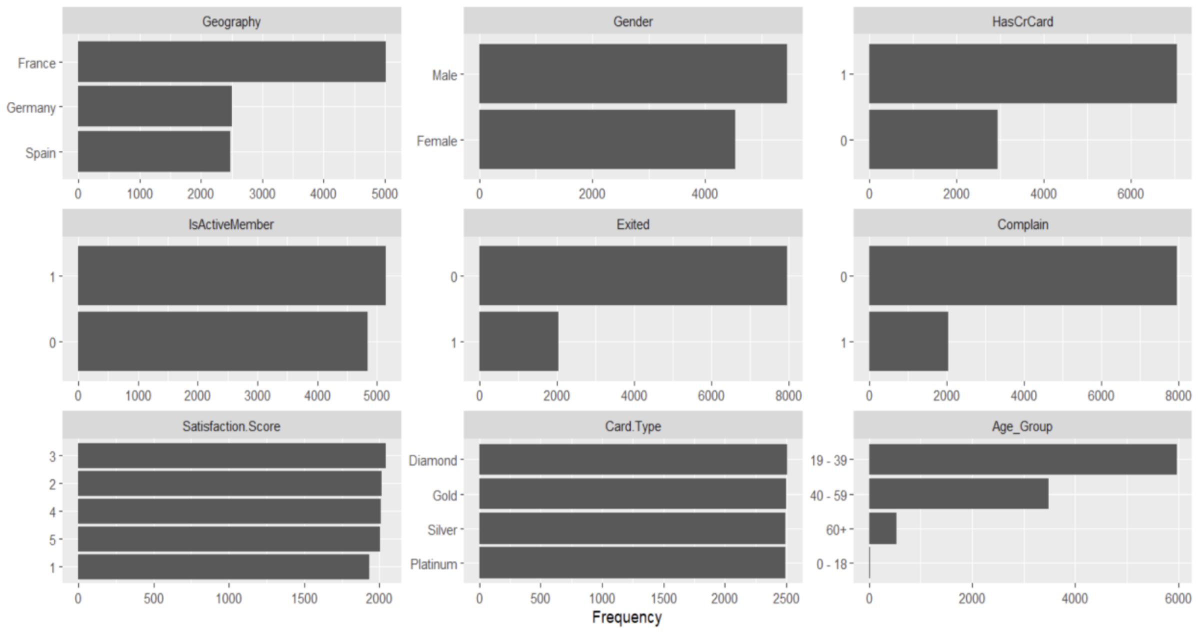


Figure 4: Horizontal bar charts

Figure 5 demonstrates the density distribution of all continuous variables.

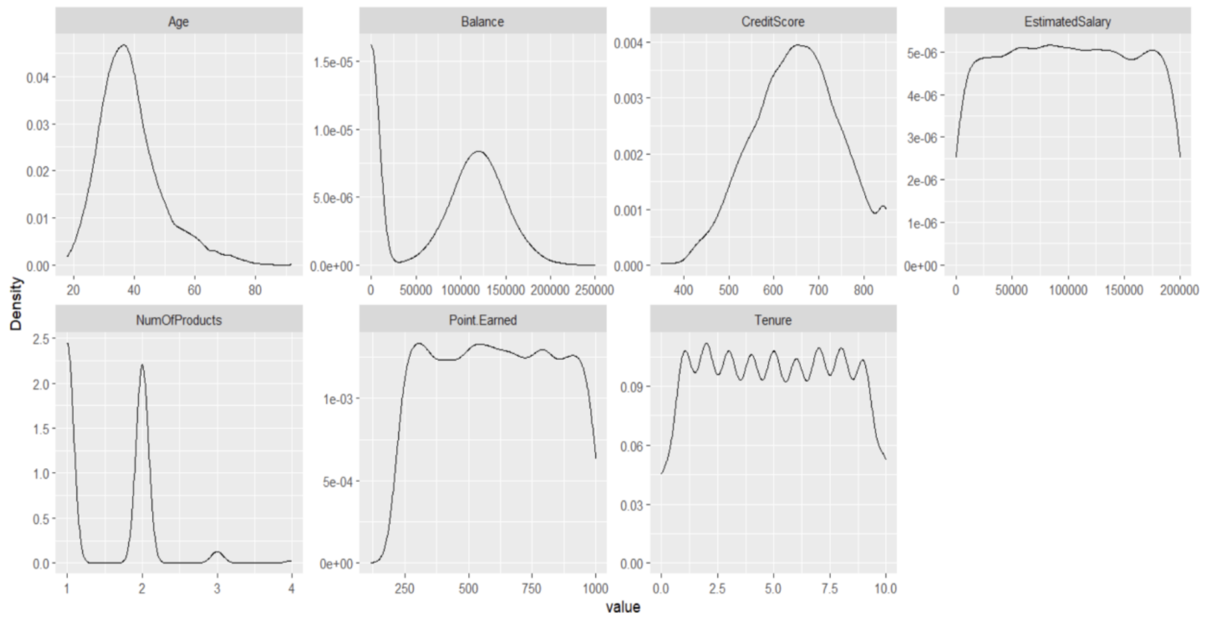


Figure 5: Density plots

Figure 6 demonstrates the boxplots of all continuous variables by Exited. There are outliers in the boxplots for age, credit score and number of bank products.

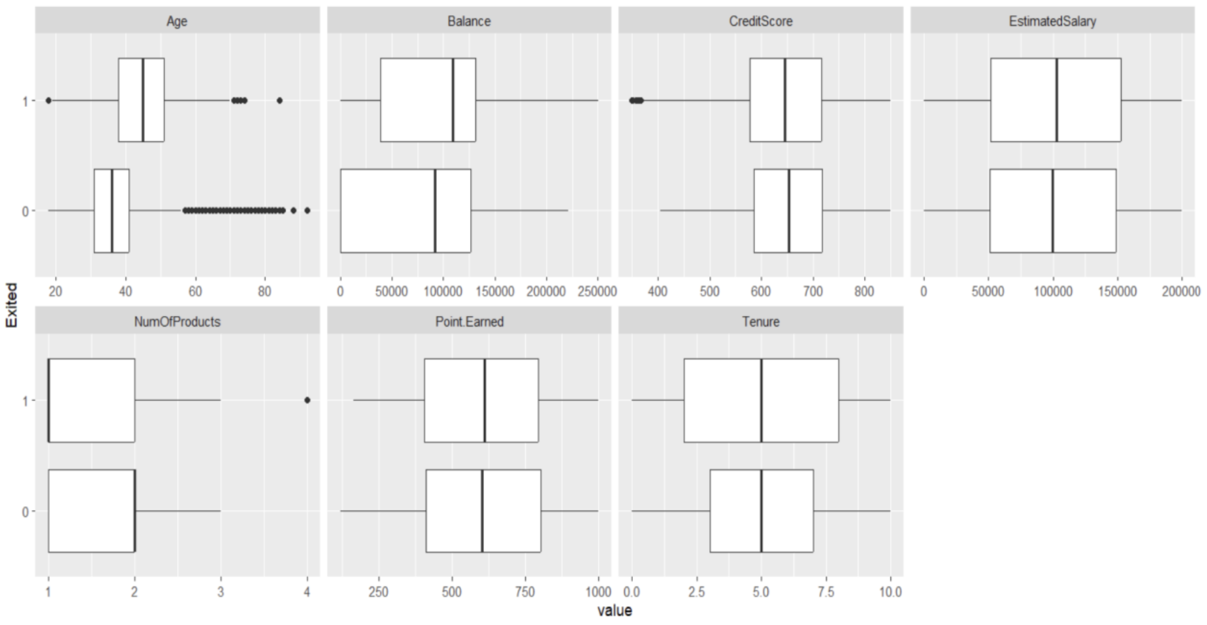


Figure 6: Boxplots

Figures 7 and 8 below display the correlation heatmaps of the categorical and continuous variables respectively. There seems to be weak negative correlation between number of products and balance.

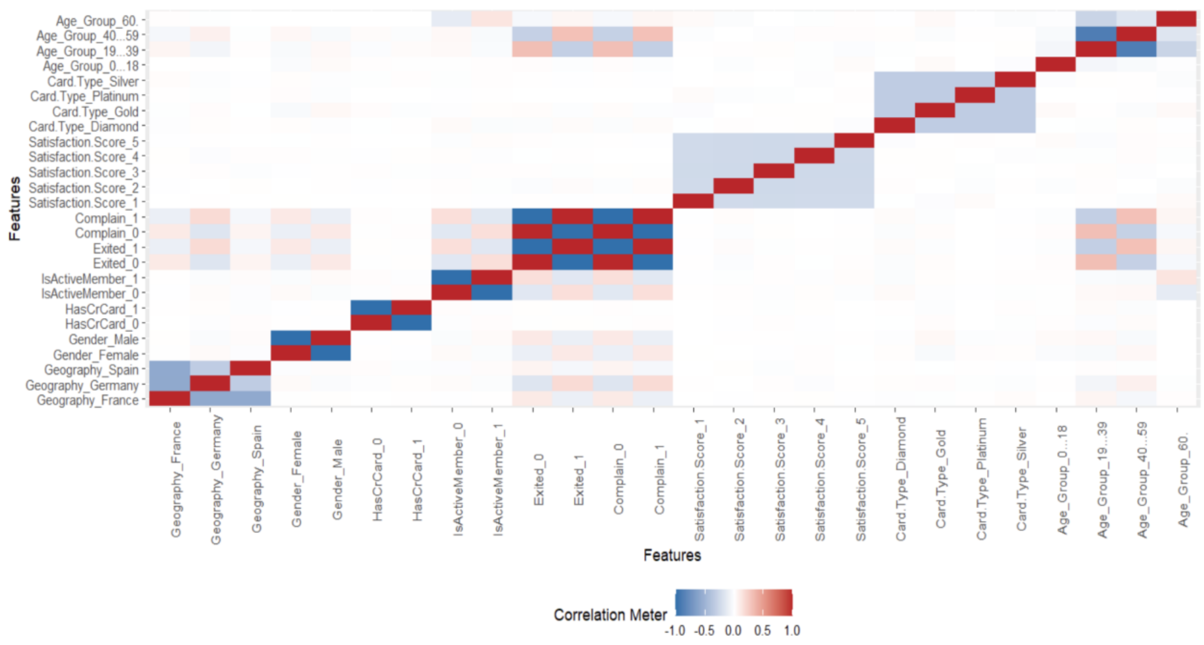


Figure 7: Correlation heatmap for categorical variables

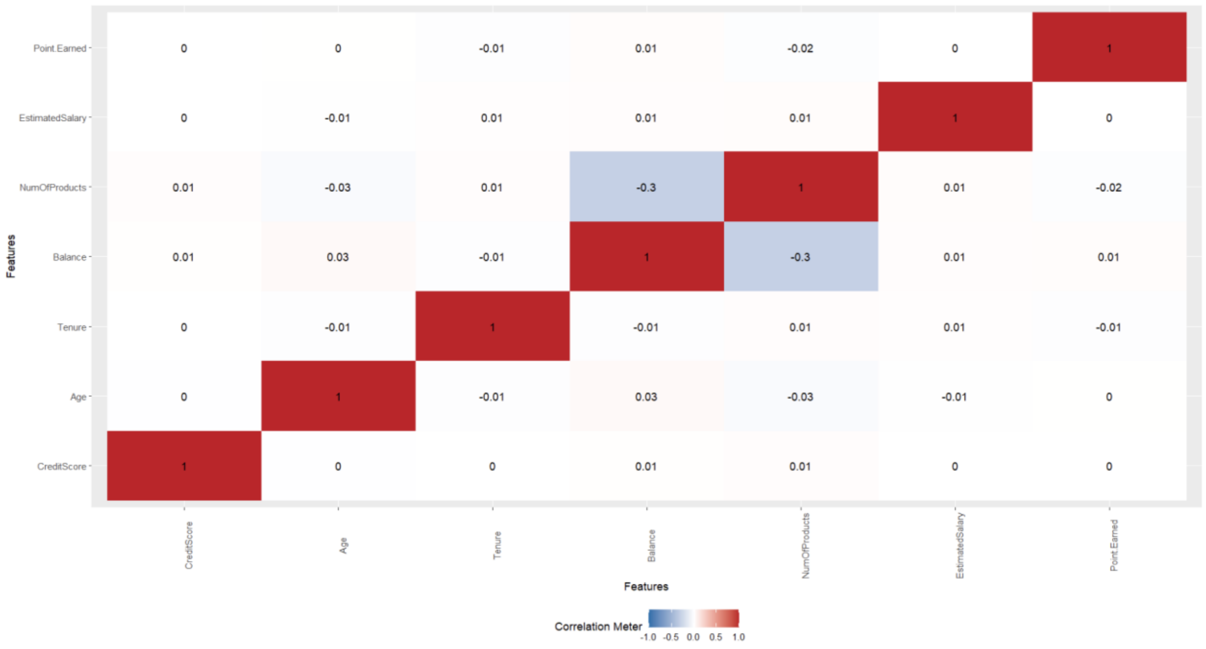
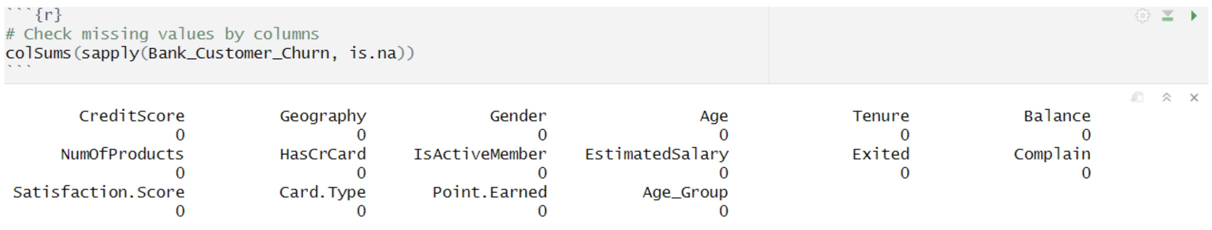
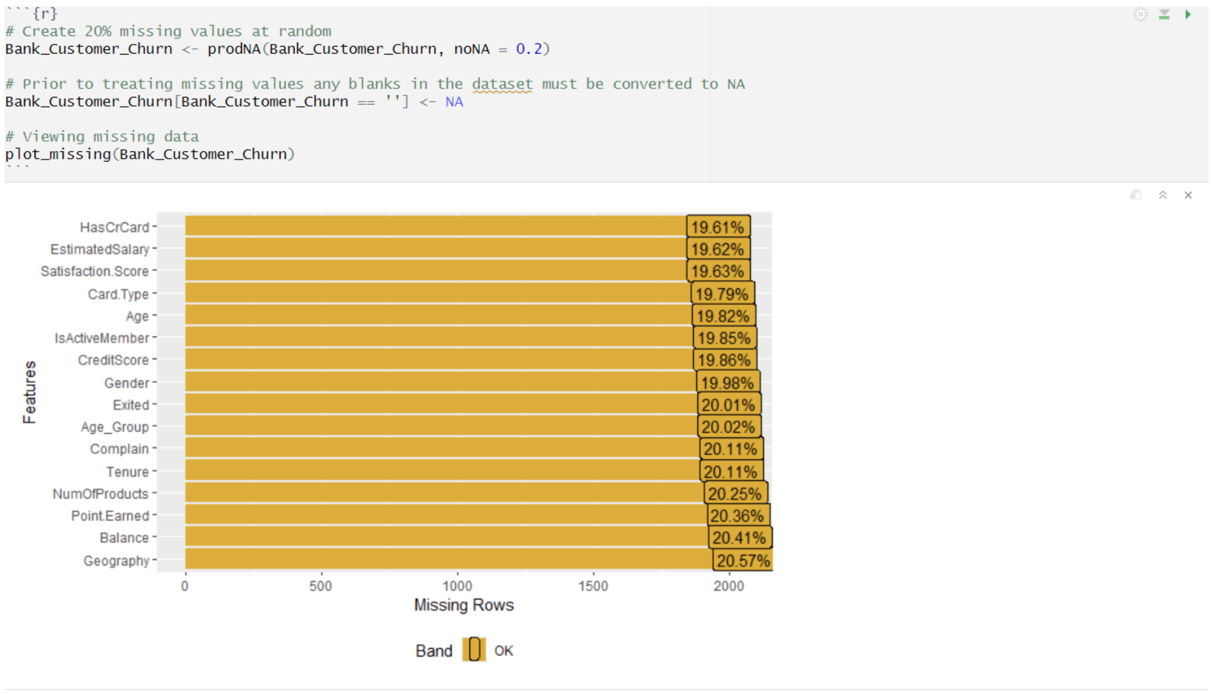
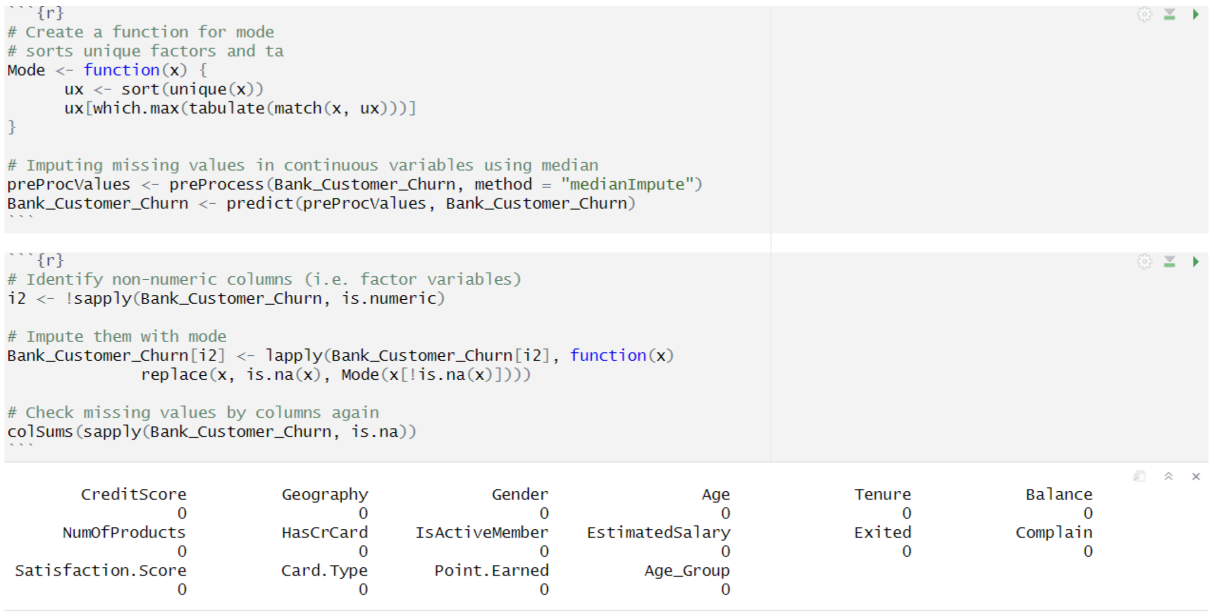


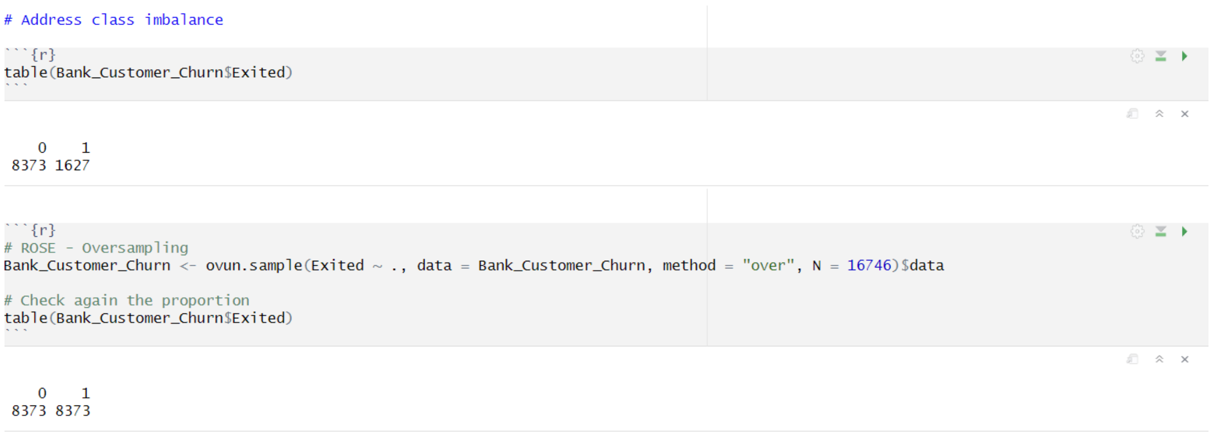
Figure 8: Correlation heatmap for continuous variables



~~~r
```{r}
# Check missing values by columns
colSums(sapply(Bank_Customer_Churn, is.na))
~~~
```

```
      CreditScore         Geography            Gender               Age            Tenure           Balance 
                0                 0                 0                 0                 0                 0 
    NumOfProducts         HasCrCard    IsActiveMember   EstimatedSalary            Exited          Complain 
                0                 0                 0                 0                 0                 0 
Satisfaction.Score         Card.Type      Point.Earned         Age_Group 
                0                 0                 0                 0 
~~~

Figure 9: Missing values in the dataset



```r
```{r}
# Create 20% missing values at random
Bank_Customer_Churn <- prodNA(Bank_Customer_Churn, noNA = 0.2)

# Prior to treating missing values any blanks in the dataset must be converted to NA
Bank_Customer_Churn[Bank_Customer_Churn == ''] <- NA

# Viewing missing data
plot_missing(Bank_Customer_Churn)
~~~
```

Figure 10: Percentage of created missing values in each column

There are no missing values in any columns based on figure 9 above. Therefore, approximately 20% missing values are created at random on each column of the dataset as shown in Figure 10 to demonstrate further imputation of missing values.
```



~~~
```{r}
# Create a function for mode
# sorts unique factors and ta
Mode <- function(x) {
      ux <- sort(unique(x))
      ux[which.max(tabulate(match(x, ux)))]
}

# Imputing missing values in continuous variables using median
preProcValues <- preProcess(Bank_Customer_Churn, method = "medianImpute")
Bank_Customer_Churn <- predict(preProcValues, Bank_Customer_Churn)
~~~

```{r}
# Identify non-numeric columns (i.e. factor variables)
i2 <- !sapply(Bank_Customer_Churn, is.numeric)

# Impute them with mode
Bank_Customer_Churn[i2] <- lapply(Bank_Customer_Churn[i2], function(x)
              replace(x, is.na(x), Mode(x[!is.na(x)])))

# Check missing values by columns again
colSums(sapply(Bank_Customer_Churn, is.na))
```
```

```
       CreditScore          Geography             Gender                Age             Tenure            Balance 
                 0                  0                  0                  0                  0                  0 
     NumOfProducts          HasCrCard     IsActiveMember    EstimatedSalary             Exited           Complain 
                 0                  0                  0                  0                  0                  0 
Satisfaction.Score          Card.Type       Point.Earned          Age_Group 
                 0                  0                  0                  0 
```

Figure 11: Codes for imputing missing values using mode and median and the output

Based on Figure 11 above, the missing values in categorical and continuous variables are imputed with mode and median values respectively.

Next, the eight categorical variables are one-hot encoded to create the corresponding dummy variables. The separated dummy variables are then bind with the original data frame and the original categorical variables are deleted later. To avoid duplication of the two binary columns for the gender variable, the two columns, Gender.Male.1 and Gender.Female.1, are dropped from the dataset.



```
# Address class imbalance

```{r}
table(Bank_Customer_Churn$Exited)
```

   0   1
8373 1627

```{r}
# ROSE - Oversampling
Bank_Customer_Churn <- ovun.sample(Exited ~ ., data = Bank_Customer_Churn, method = "over", N = 16746)$data

# Check again the proportion
table(Bank_Customer_Churn$Exited)
```

   0   1
8373 8373
```

*Figure 12: Codes for addressing class imbalance*

Based on Figure 12 above, there is class imbalance issues on the target variable, Exited. There are 83.73% of bank customers who remain as members of the bank and 16.27% of
```

them left, so the proportion of class between minority and majority is highly imbalanced. Therefore, the oversampling technique is done on the minority class until it reaches 8387 using the ROSE package to strike a balance in numbers on both classes. As a result, the number of observations for each class of Exited is 8373 and the total number of observations in the balanced dataset is 16,746.

Finally, before the end of the dataset preparation stage, min-max normalization is done on the continuous variables of the dataset by scaling their values to the zero-one spectrum. The normalized values are then rounded up to three decimal points. All in all, the final dataset contains 30 columns and 16,746 rows, and is ready for model implementation using machine learning approaches.

5.0 Model Implementation

5.1 Train-Test Split

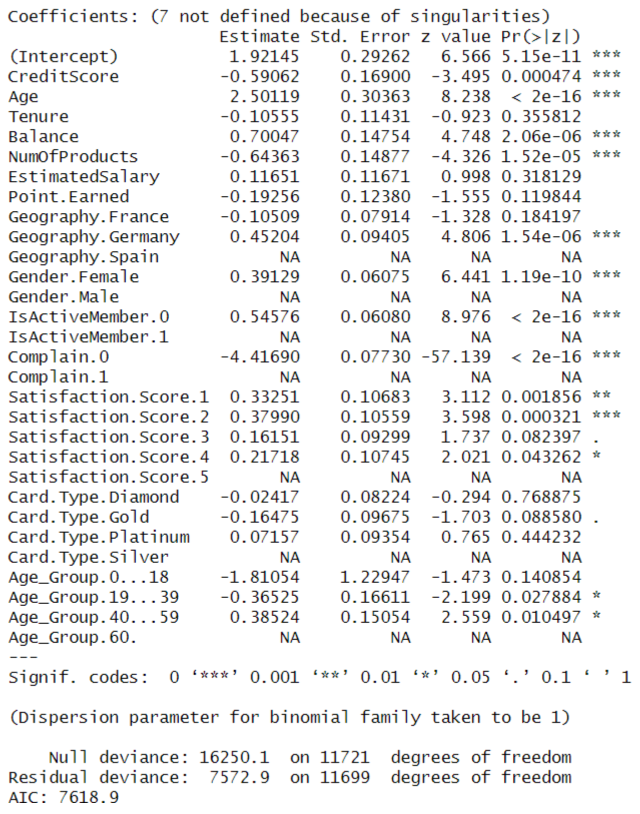
In terms of stratified sampling, the bank customer churn dataset is split into training and test sets according to the 70:30 ratio, termed train_bank and test_bank respectively. As a result, the training and test sets contain 11,722 and 5,024 observations respectively. Different number of seeds are set for different machine learning methods.

5.2 Baseline Logistic Regression

5.2.1 Training Set

The baseline logistic regression method will be used to implement the first model of this assignment. First, a classifier needs to be created to fit the logistic regression model using glm function. Inside this function, Exited is inputted as the target variable and the remaining variables are inputted as the predictors in the formula, in addition to the inclusion of the training set as the data frame. Customer churn prediction is a classification task, so the value of the family is set as binomial.

The summary of the result is shown below in Figure 13. In terms of numeric variables, credit score, age, tenure, balance, and number of bank products purchased by bank customers are the statistically significant predictors of churn as their p-values are less than the 0.001 significance level. For categorical variables, bank customers living in Germany, female customers, customers belonging to the young and middle adult age group, customers expressing above-average dissatisfaction and good satisfaction towards the bank as well as non-active and no-complaint bank members are statistically significant predictors of customer churn.

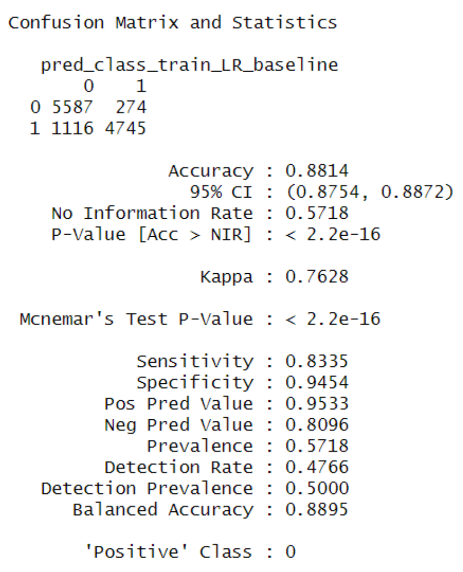


```
Coefficients: (7 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)    
(Intercept)           1.92145    0.29262   6.566 5.15e-11 ***
CreditScore          -0.59062    0.16900  -3.495 0.000474 ***
Age                   2.50119    0.30363   8.238  < 2e-16 ***
Tenure               -0.10555    0.11431  -0.923 0.355812    
Balance               0.70047    0.14754   4.748 2.06e-06 ***
NumOfProducts        -0.64363    0.14877  -4.326 1.52e-05 ***
EstimatedSalary       0.11651    0.11671   0.998 0.318129    
Point.Earned         -0.19256    0.12380  -1.555 0.119844    
Geography.France     -0.10509    0.07914  -1.328 0.184197    
Geography.Germany     0.45204    0.09405   4.806 1.54e-06 ***
Geography.Spain            NA         NA      NA       NA    
Gender.Female         0.39129    0.06075   6.441 1.19e-10 ***
Gender.Male                NA         NA      NA       NA    
IsActiveMember.0      0.54576    0.06080   8.976  < 2e-16 ***
IsActiveMember.1           NA         NA      NA       NA    
Complain.0           -4.41690    0.07730 -57.139  < 2e-16 ***
Complain.1                 NA         NA      NA       NA    
Satisfaction.Score.1  0.33251    0.10683   3.112 0.001856 ** 
Satisfaction.Score.2  0.37990    0.10559   3.598 0.000321 ***
Satisfaction.Score.3  0.16151    0.09299   1.737 0.082397 .  
Satisfaction.Score.4  0.21718    0.10745   2.021 0.043262 *  
Satisfaction.Score.5       NA         NA      NA       NA    
Card.Type.Diamond    -0.02417    0.08224  -0.294 0.768875    
Card.Type.Gold       -0.16475    0.09675  -1.703 0.088580 .  
Card.Type.Platinum    0.07157    0.09354   0.765 0.444232    
Card.Type.Silver           NA         NA      NA       NA    
Age_Group.0...18     -1.81054    1.22947  -1.473 0.140854    
Age_Group.19...39    -0.36525    0.16611  -2.199 0.027884 *  
Age_Group.40...59     0.38524    0.15054   2.559 0.010497 *  
Age_Group.60.              NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 16250.1  on 11721  degrees of freedom
Residual deviance:  7572.9  on 11699  degrees of freedom
AIC: 7618.9
```

Figure 13: Results summary of the created logistic regression classifier

Next, using the predict function, the forecasted probabilities with the logistic regression classifier as the input are generated for the training set, which then undergo transformation into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the confusion matrix of the training set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 14 demonstrates the confusion matrix and logistic regression model performance metrics using the training set. For true positives, there are 5,587 instances for the prediction that the customer has not churned, and it fits with the true case. For true negatives, there are 4,745 instances for the prediction that the customer has churned, and it fits with the true case. For false negatives, there are 274 instances of the prediction that the customer has churned when churn has not happened actually. For false positives, there are 1116 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 88.14% and 95.33% respectively. The percentage of recall is 83.35% and the percentage of f1-score is 88.94%.



```
Confusion Matrix and Statistics

   pred_class_train_LR_baseline
       0    1
  0 5587  274
  1 1116 4745

               Accuracy : 0.8814
                 95% CI : (0.8754, 0.8872)
    No Information Rate : 0.5718
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7628

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8335
            Specificity : 0.9454
         Pos Pred Value : 0.9533
         Neg Pred Value : 0.8096
             Prevalence : 0.5718
         Detection Rate : 0.4766
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8895

       'Positive' Class : 0
```

Figure 14: Confusion matrix and logistic regression model performance metrics (Training set)

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 15. As a result, the value of AUC for the baseline logistic regression model on the training set is 0.881.

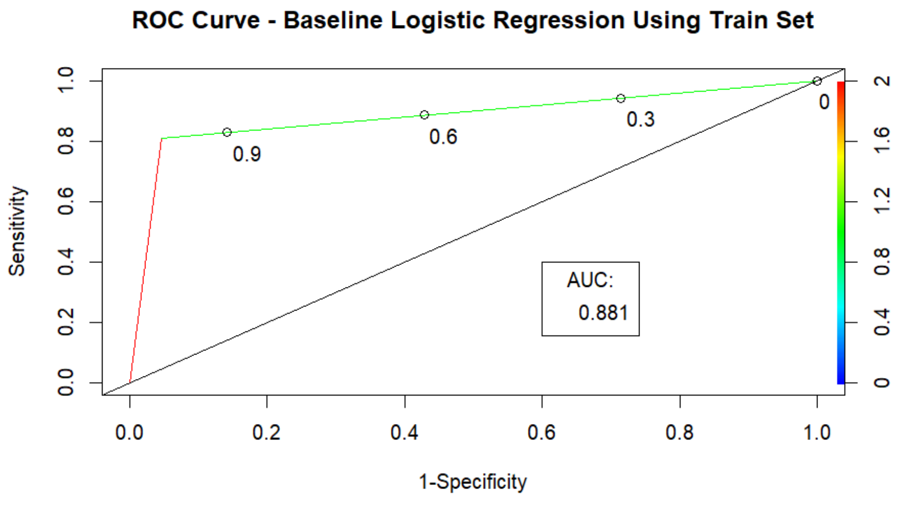
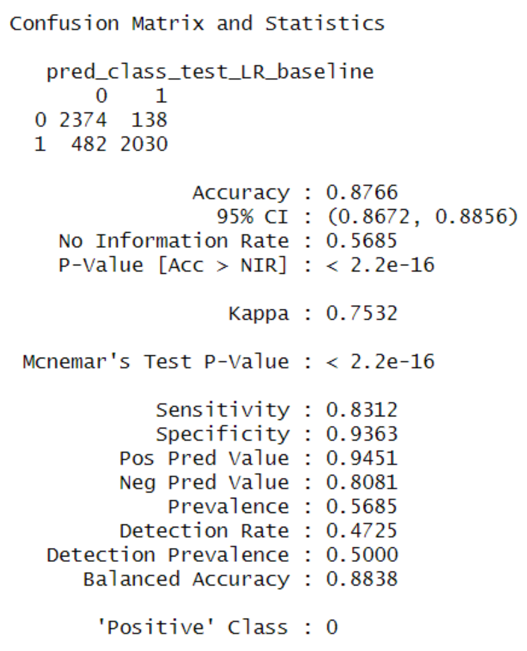


Figure 15: ROC curve for baseline logistic regression model using training set

5.2.2 Test Set

Using the predict function, the forecasted probabilities with the logistic regression classifier as the input are generated for the test set, which then undergo transformation into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the confusion matrix of the test set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 16 demonstrates the confusion matrix and logistic regression model performance metrics using the test set. For true positives, there are 2,374 instances for the prediction that the customer has not churned, and it fits with the true case. For true negatives, there are 2,030 instances for the prediction that the customer has churned, and it fits with the true case. For false negative, there are 138 instances of the prediction that the customer has churned when churn has not happened actually. For false positives, there are 482 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 87.66% and 94.51% respectively. The percentage of recall is 83.12% and the percentage of f1-score is 88.45%.



```
Confusion Matrix and Statistics

   pred_class_test_LR_baseline
       0    1
  0 2374  138
  1  482 2030

               Accuracy : 0.8766
                 95% CI : (0.8672, 0.8856)
    No Information Rate : 0.5685
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7532

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8312
            Specificity : 0.9363
         Pos Pred Value : 0.9451
         Neg Pred Value : 0.8081
             Prevalence : 0.5685
         Detection Rate : 0.4725
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8838

       'Positive' Class : 0
```

Figure 16: Confusion matrix and logistic regression model performance metrics (Test set)

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 17. As a result, the value of AUC for the baseline logistic regression model on the test set is 0.877.

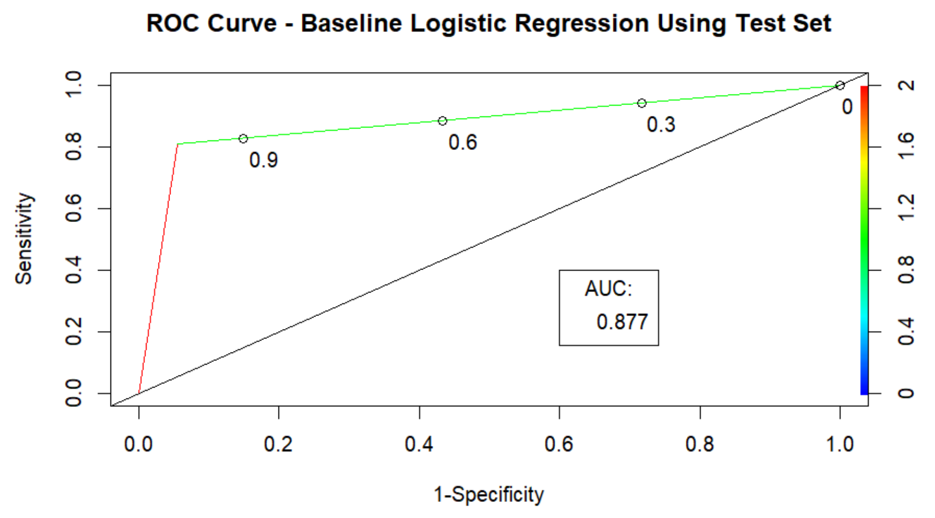


Figure 17: ROC curve for baseline logistic regression model using test set

5.3 L1-Regularised Logistic Regression

The L1-regularised logistic regression method will be used to implement the second model of this assignment. First, the training and test dataset needs to be converted to matrix form. The training set without the target variable is converted to matrix form and assigned to the variable ‘a’, and the target variable’s column in the training set is converted to character type followed by numeric type and assigned to the variable ‘b’. The test set without the target variable is converted to matrix form and assigned to the variable ‘c’, and the target variable’s column in the testing set is converted to character type followed by numeric type and assigned to the variable ‘d’.

5.3.1 Training Set

First, the logistic regression model is fitted with Lasso or L1 regularisation by inputting ‘a’ and ‘b’ into the glmnet function, in addition to indicating the family as binomial and alpha equal to one. The best lambda value can be discovered through the plot of the Lasso path, that is the plot of mean squared error (MSE) against log lambda, as shown in Figure 18. It can be observed that the Lasso path is stable until it approximately reaches the log lambda value of -4 where the MSE value experiences an exponential increase.

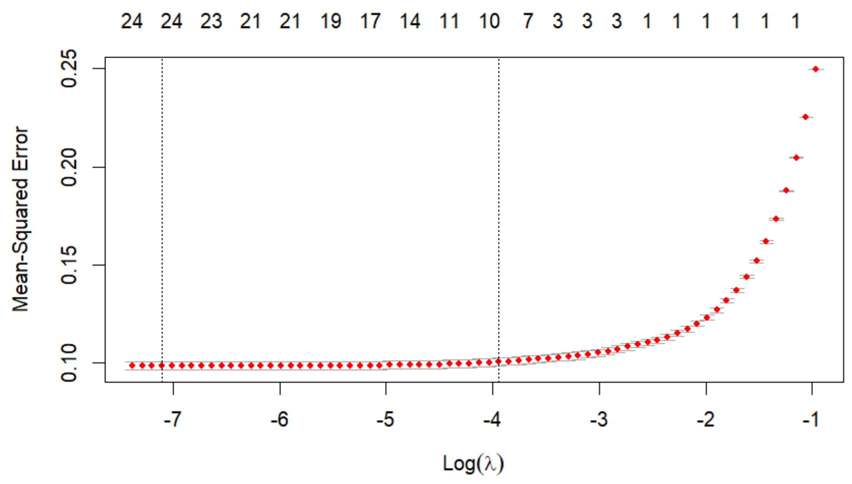


Figure 18: Plot of MSE against log(λ) using training set

According to the coefficient path shown in Figure 19, the minimum error-lambda value (lambda.min) is 0.000822 and the one standard error-lambda value (lambda.1se) is 0.0194465. Based on this, the lambda.1se value will be used to construct the L1-regularised logistic regression model for prediction using the training set. The coefficient values for this model are displayed in plots in Figure 20. The highest coefficient predictor is age of bank customers (index = 3) followed by those who lodge complaints to the bank (index = 17).

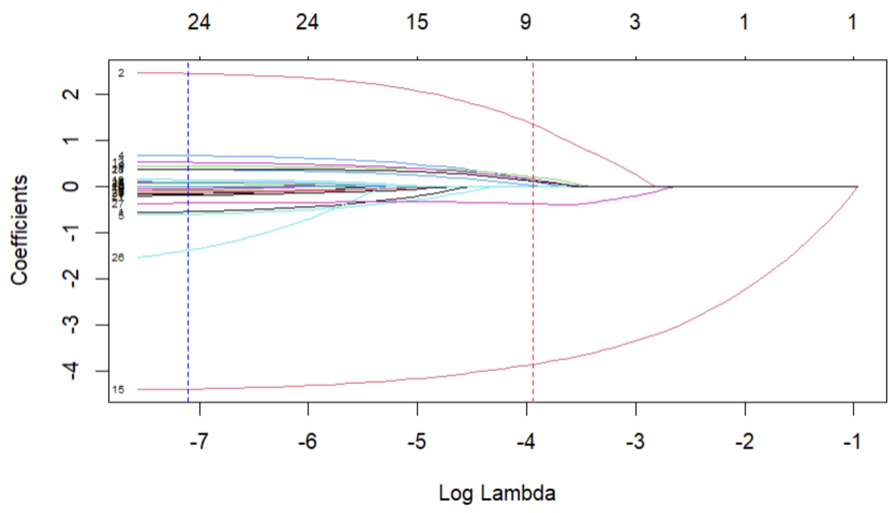


Figure 19: Coefficient path using training set

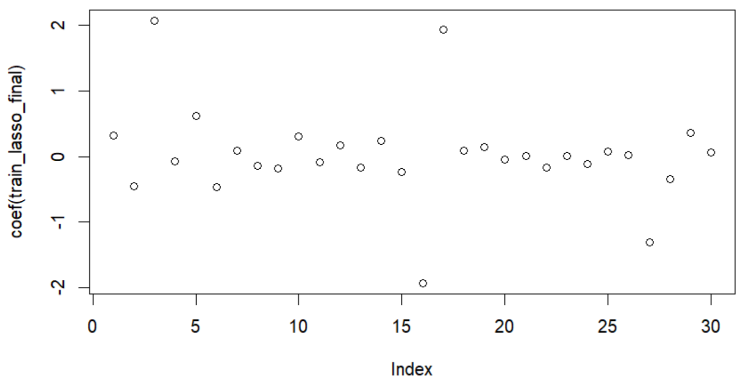
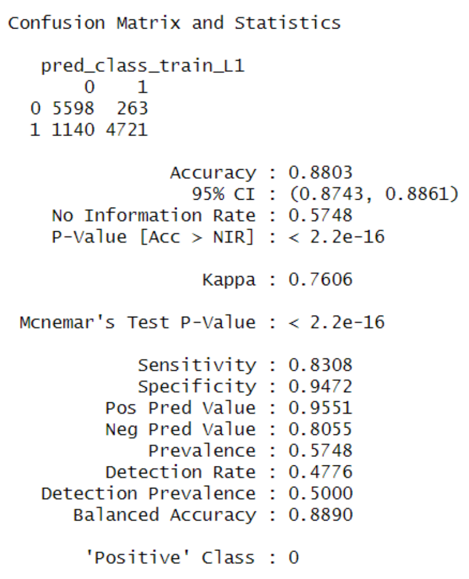


Figure 20: Fitted L1-regularised logistic regression model coefficient plot using training set

Next, using the predict function, the forecasted probabilities with the L1-regularised logistic regression classifier as the input along with the data matrix form for the training set without target variable (that is variable “a”) are generated, which then undergo transformation into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the confusion matrix of the training set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 21 demonstrates the confusion matrix and L1-regularised logistic regression model performance metrics using the training set. For true positives, there are 5,598 instances for the prediction that the customer has not churned, and it fits with the true case. For true negatives, there are 4,721 instances for the prediction that the customer has churned, and it fits with the true case. For false negatives, there are 263 instances of the prediction that the customer has churned when churn has not happened actually. For false positives, there are 1,140 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 88.03% and 95.51% respectively. The percentage of recall is 83.08% and the percentage of f1-score is 89.04%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 22. As a result, the value of AUC for the L1-regularised logistic regression model on the train set is 0.88.



```
Confusion Matrix and Statistics

   pred_class_train_L1
       0    1
  0 5598  263
  1 1140 4721

               Accuracy : 0.8803
                 95% CI : (0.8743, 0.8861)
    No Information Rate : 0.5748
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7606

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8308
            Specificity : 0.9472
         Pos Pred Value : 0.9551
         Neg Pred Value : 0.8055
             Prevalence : 0.5748
         Detection Rate : 0.4776
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8890

       'Positive' Class : 0
```

Figure 21: Confusion matrix and L1-regularised logistic regression model performance metrics (Training set)

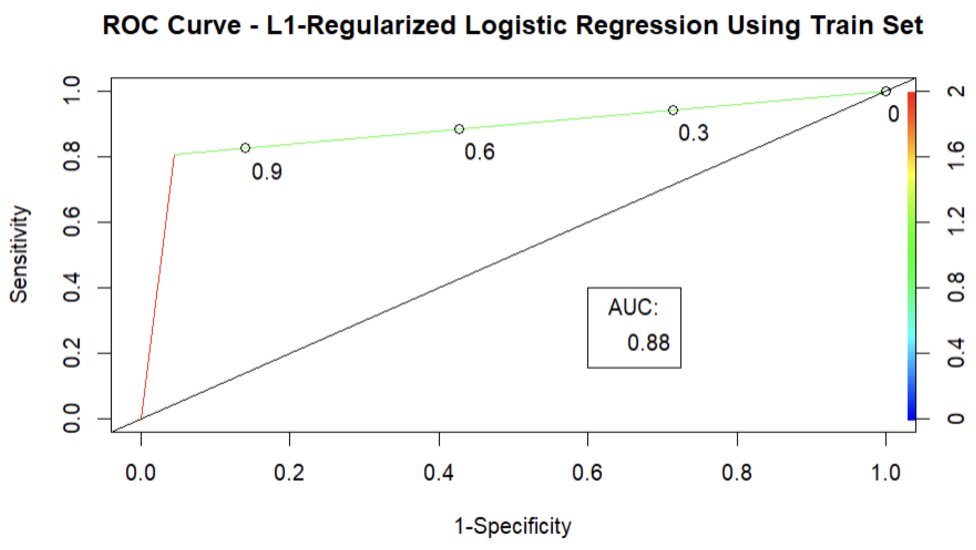


Figure 22: ROC curve for L1-regularised logistic regression model using the training set

5.3.2 Test Set

For the test set, the logistic regression model is fitted with Lasso or L1 regularisation by inputting ‘c’ and ‘d’ into the glmnet function, in addition to indicating the family as binomial and alpha equal to one. The best lambda value for the test set can be discovered through the plot of the Lasso path, that is the plot of MSE against log lambda, as shown in Figure 23. Almost like the training set, it can be observed that the Lasso path is stable until it exceeds the log lambda value of -4 by a little where the MSE value experiences an exponential increase.

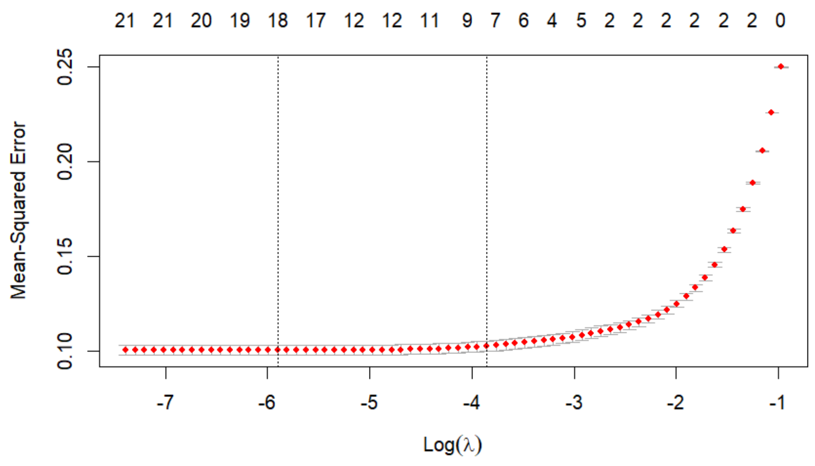


Figure 23: Plot of MSE against log(λ) using test set

According to the coefficient path shown in Figure 24, the minimum error-lambda value (lambda.min) is 0.00274 and the one standard error-lambda value (lambda.1se) is 0.0212. Based on this, the lambda.1se value will be used to construct the L1-regularised logistic regression model for prediction using the test set. The coefficient values for this model are displayed in plots in Figure 25. The highest coefficient predictor is age of bank customers (index = 3).

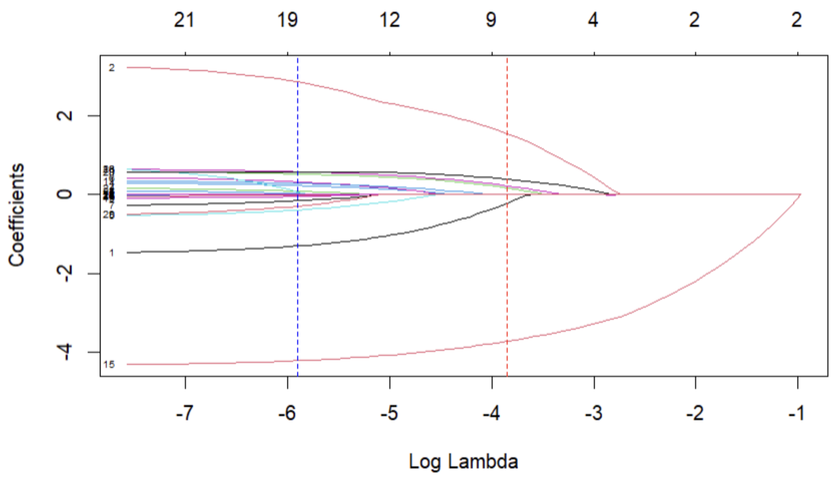


Figure 24: Coefficient path using test set

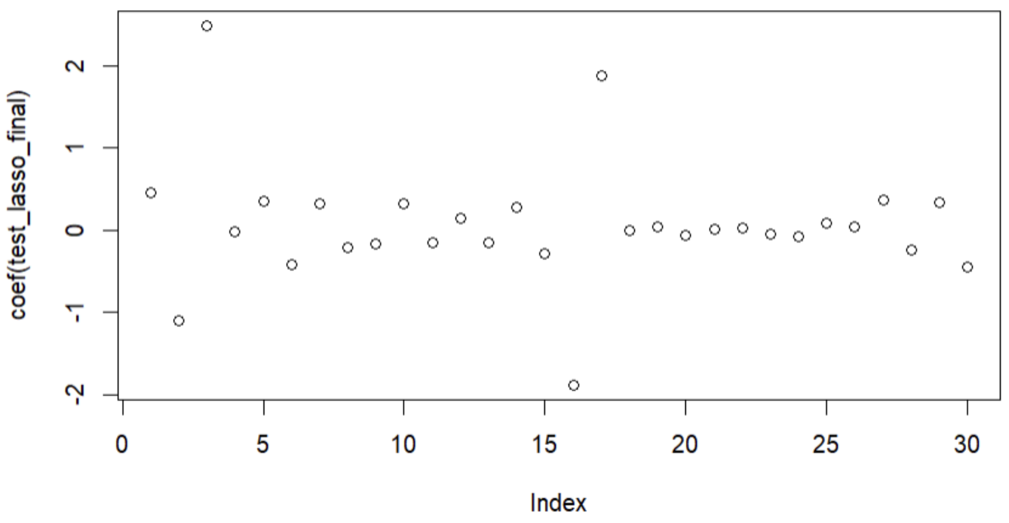
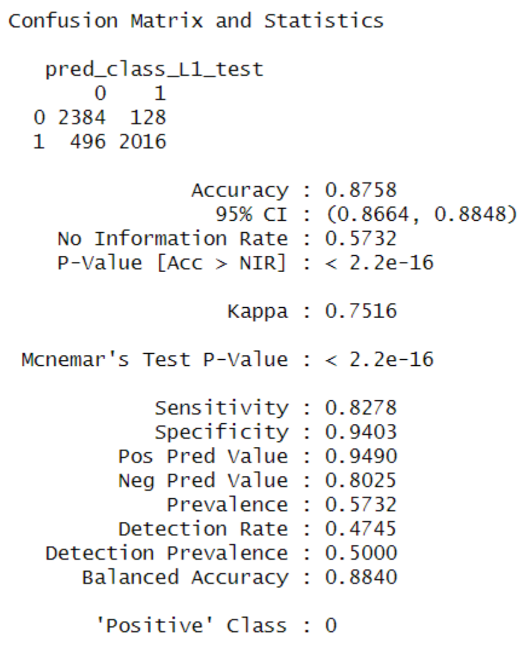


Figure 25: Fitted L1-regularised logistic regression model coefficient plot using test set

Next, using the predict function, the forecasted probabilities with the L1-regularised logistic regression classifier as the input along with the data matrix form for the test set without target variable (that is variable “c”) are generated, which then undergo transformation into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the confusion matrix of the test set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 26 demonstrates the confusion matrix and L1-regularised logistic regression model performance metrics using the test set. For true positives, there are 2,384 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 2,016 instances for the prediction that the customer has churned, and it fits with the true case. For false negatives, there are 128 instances of the prediction that the customer has churned when churn has not happened actually. For false positives, there are 496 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 87.58% and 94.9% respectively. The percentage of recall is 82.78% and the percentage of f1-score is 88.43%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 27. As a result, the value of AUC for the L1-regularised logistic regression model on the test set is 0.876.



```
Confusion Matrix and Statistics

   pred_class_L1_test
       0    1
  0 2384  128
  1  496 2016

               Accuracy : 0.8758
                 95% CI : (0.8664, 0.8848)
    No Information Rate : 0.5732
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7516

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8278
            Specificity : 0.9403
         Pos Pred Value : 0.9490
         Neg Pred Value : 0.8025
             Prevalence : 0.5732
         Detection Rate : 0.4745
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.8840

       'Positive' Class : 0
```

Figure 26: Confusion matrix and L1-regularised logistic regression model performance metrics (Test set)

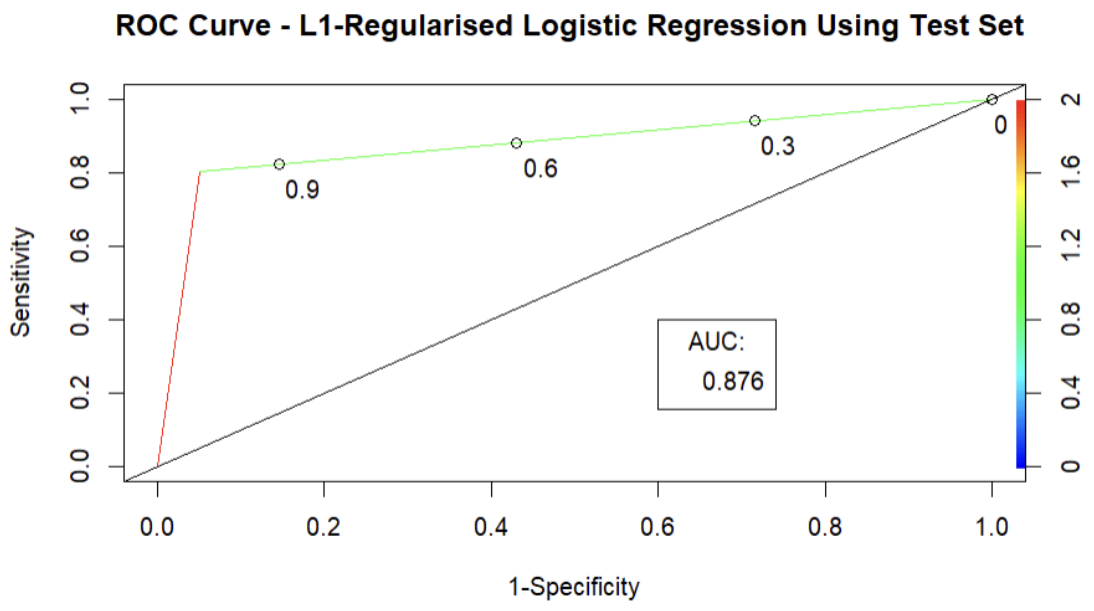


Figure 27: ROC curve for L1-regularised logistic regression model using test set

5.4 Polynomial Support Vector Machine (SVM)

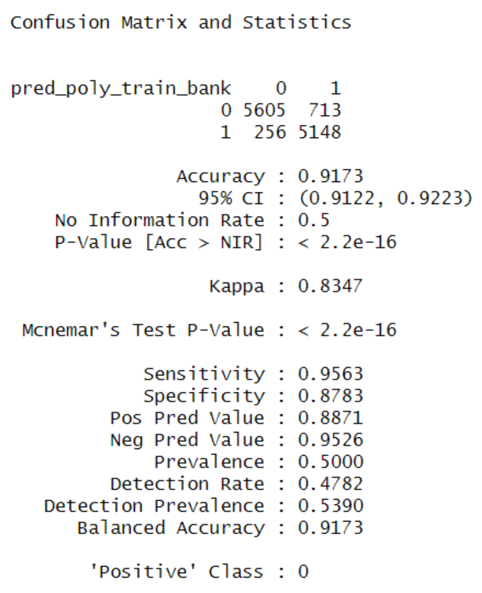
The polynomial SVM method will be used to implement the third model of this assignment. First, the polynomial SVM model is created using svm function. Inside this function, Exited is inputted as the target variable and the remaining variables are inputted as the predictors in the formula, in addition to the inclusion of the train_bank dataset as the data frame and setting the SVM kernel as poly. As a result, the cost of the model equals to zero and the degree value is three. The number of support vectors produced is 3,539.

5.4.1 Training Set

Next, using the predict function, the forecasted probabilities with the polynomial SVM model as the input are generated for the training set, which then transforms into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the polynomial SVM-based confusion matrix of the training set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 28 demonstrates the confusion matrix and polynomial SVM model performance metrics using the training set. For true positives, there are 5,605 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 5,148 instances for the prediction that the customer has churned, and it fits with the true case. For false positives, there are 713 instances of the prediction that the customer has churned when churn has not happened actually. For false negatives, there are 256 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 91.73% and 88.71% respectively. The percentage of recall is 95.63% and the percentage of f1-score is 92.04%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 29. As a result, the value of AUC for the polynomial SVM model on the training set is 0.917.



```
Confusion Matrix and Statistics

pred_poly_train_bank    0    1
                   0 5605  713
                   1  256 5148

               Accuracy : 0.9173
                 95% CI : (0.9122, 0.9223)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8347

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9563
            Specificity : 0.8783
         Pos Pred Value : 0.8871
         Neg Pred Value : 0.9526
             Prevalence : 0.5000
         Detection Rate : 0.4782
   Detection Prevalence : 0.5390
      Balanced Accuracy : 0.9173

       'Positive' Class : 0
```

Figure 28: Confusion matrix and polynomial SVM model performance metrics (Train set)

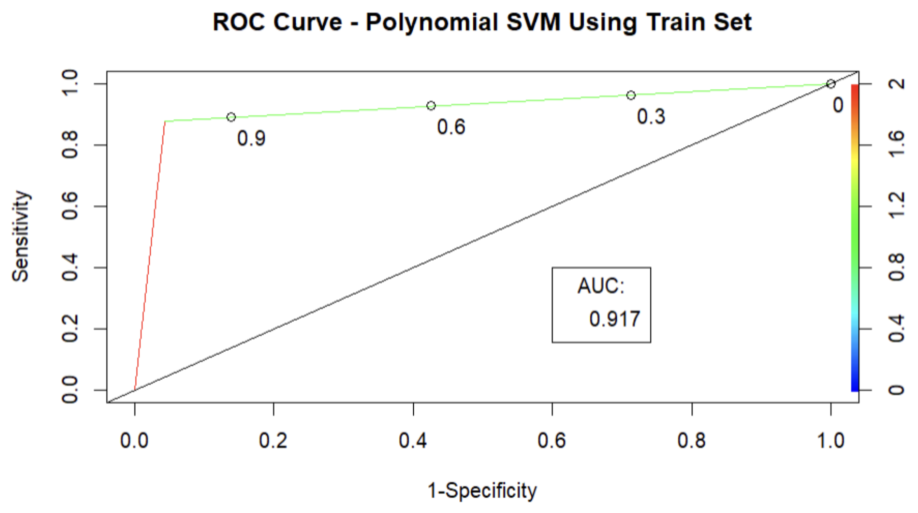


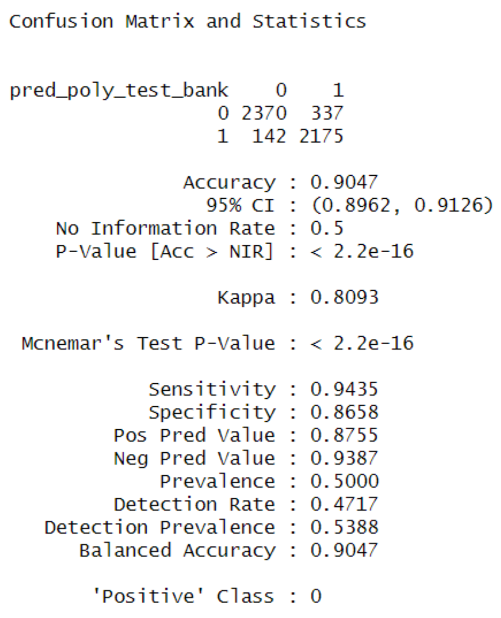
Figure 29: ROC curve for polynomial SVM model using training set

5.4.2 Test Set

Using the predict function, the forecasted probabilities with the polynomial SVM model as the input are generated for the test set, which then transforms into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the polynomial SVM-based confusion matrix of the test set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 30 demonstrates the confusion matrix and polynomial SVM model performance metrics using the test set. For true positives, there are 2,370 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 2,175 instances for the prediction that the customer has churned, and it fits with the true case. For false positives, there are 337 instances of the prediction that the customer has churned when churn has not happened actually. For false negatives, there are 142 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 90.47% and 87.55% respectively. The percentage of recall is 94.35% and the percentage of f1-score is 90.82%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 31. As a result, the value of AUC for the polynomial SVM model on the test set is 0.905.



```
Confusion Matrix and Statistics


pred_poly_test_bank    0    1
                  0 2370  337
                  1  142 2175

               Accuracy : 0.9047
                 95% CI : (0.8962, 0.9126)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8093

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9435
            Specificity : 0.8658
         Pos Pred Value : 0.8755
         Neg Pred Value : 0.9387
             Prevalence : 0.5000
         Detection Rate : 0.4717
   Detection Prevalence : 0.5388
      Balanced Accuracy : 0.9047

       'Positive' Class : 0
```

Figure 30: Confusion matrix and polynomial SVM model performance metrics (Test set)

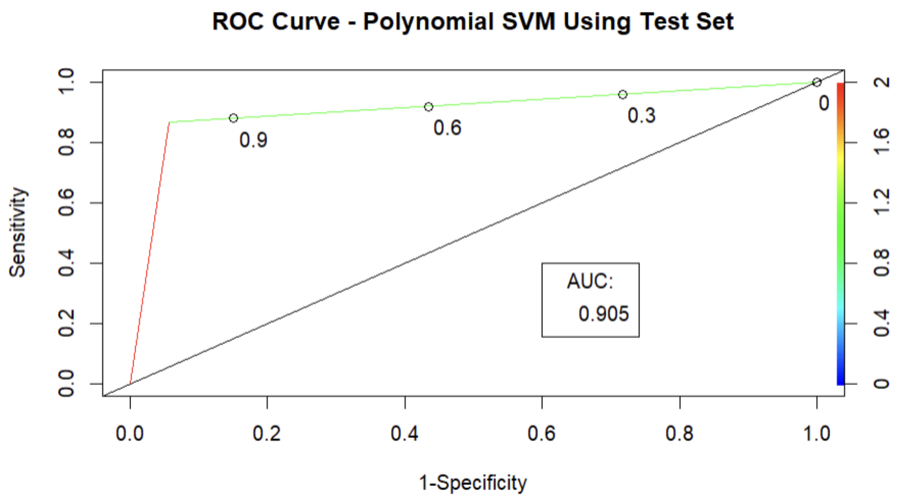


Figure 31: ROC curve for polynomial SVM model using test set

5.5 Tuned SVM with Grid Search

The fourth model to be implemented is hyperparameter tuned SVM using grid search. 10-fold cross validation is implemented in building the model to prevent overfitting of the tuned model using the trainControl function. To build the tuned model, the tuning grid needs to be defined. In this case, all epsilon value pairs that are within the zero-one spectrum in one-tenth steps, in addition to the cost parameter which encompasses the values of 1, 2 and 4 that are derived from the powers of two, lay the foundation of the tuning grid. Then, tune.svm function is used for hyperparameter tuning by including Exited as the target variable for prediction and the remaining variables are inputted as the predictors in the formula, in addition to the inclusion of the train_bank dataset as the target data frame and tuning grid as the tuning range. As a result, the cross-validated SVM predicted error for the tuned model is 0.0961. The cost value for the optimal tuned radial SVM model is 1, and there are 3,652 support vectors in this model which is then used to build the best SVM model using the svm function. Inside this function, Exited is inputted as the target variable and the remaining variables are inputted as the predictors in the formula, in addition to setting the epsilon and cost value to 0 and 1 respectively.

5.5.1 Training Set

Next, using the predict function, the forecasted probabilities with the best-tuned SVM model as the input are generated for the training set, which then transforms into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the best-tuned SVM-based confusion matrix of the training set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 32 demonstrates the confusion matrix and the best-tuned SVM model performance metrics using the training set. For true positives, there are 5,604 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 5,107 instances for the prediction that the customer has churned, and it fits with the true case. For false positives, there are 754 instances of the prediction that the customer has churned when churn has not happened actually. For false negatives, there are 257 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 91.38% and 88.14% respectively. The percentage of recall is 95.62% and the percentage of f1-score is 91.73%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 33. As a result, the value of AUC for the best-tuned SVM model on the training set is 0.914.

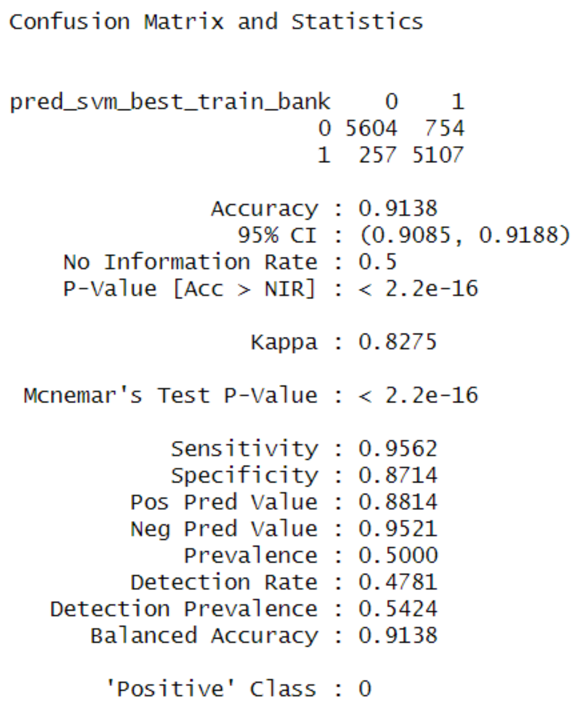

```
Confusion Matrix and Statistics


pred_svm_best_train_bank    0    1
                       0 5604  754
                       1  257 5107

               Accuracy : 0.9138
                 95% CI : (0.9085, 0.9188)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8275

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9562
            Specificity : 0.8714
         Pos Pred Value : 0.8814
         Neg Pred Value : 0.9521
             Prevalence : 0.5000
         Detection Rate : 0.4781
   Detection Prevalence : 0.5424
      Balanced Accuracy : 0.9138

       'Positive' Class : 0
```

Figure 32: Confusion matrix and best tuned SVM model performance metrics (Train set)

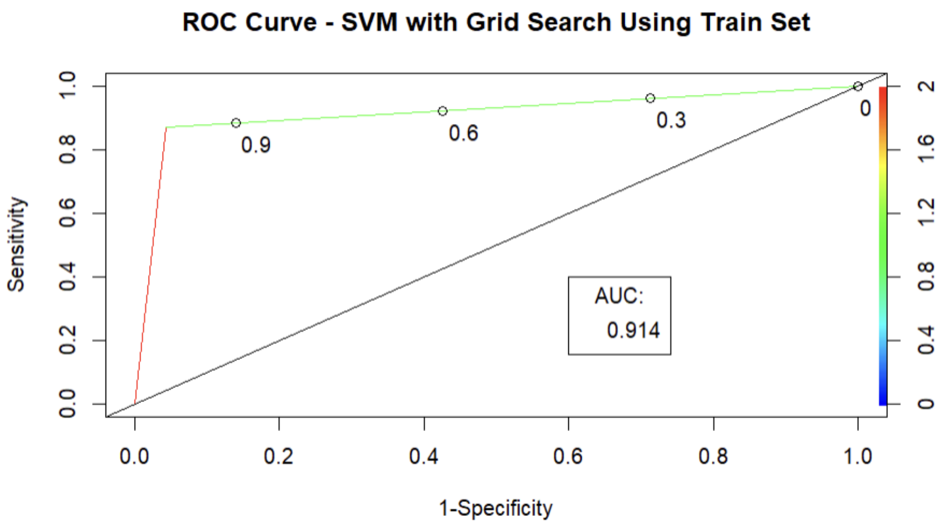


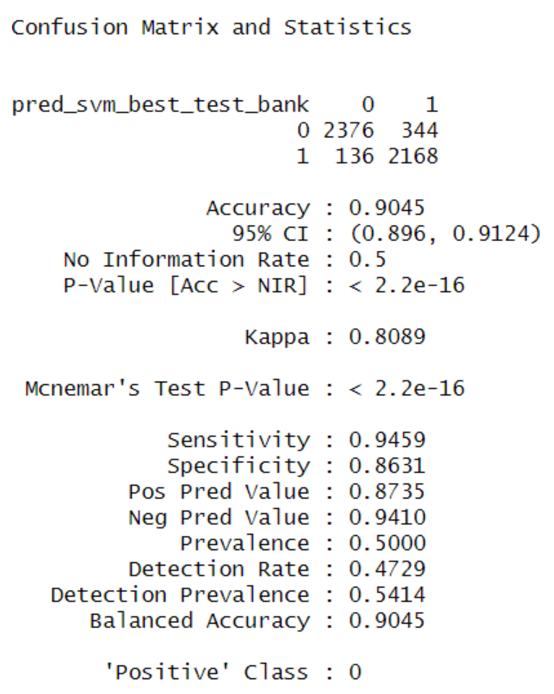
Figure 33: ROC curve for grid search tuned SVM model using training set

5.5.2 Test Set

The forecasted probabilities with the best tuned SVM model as the input are generated for the test set, which then transforms into binary class predictions such that class 0 is assigned under the condition of the predicted probability being less than 0.5, otherwise the assignment of class 0 is performed. Finally, the best-tuned SVM-based confusion matrix of the test set is formed in addition to its accuracy through the comparison between predicted and actual labels of class.

Figure 34 demonstrates the confusion matrix and the best-tuned SVM model performance metrics using the test set. For true positives, there are 2,376 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 2,168 instances for the prediction that the customer has churned, and it fits with the true case. For false positives, there are 344 instances of the prediction that the customer has churned when churn has not happened actually. For false negatives, there are 136 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 90.45% and 87.35% respectively. The percentage of recall is 94.59% and the percentage of f1-score is 90.83%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 35. As a result, the value of AUC for the best-tuned SVM model on the test set is 0.904.



```
Confusion Matrix and Statistics

pred_svm_best_test_bank    0    1
                      0 2376  344
                      1  136 2168

               Accuracy : 0.9045
                 95% CI : (0.896, 0.9124)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8089

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9459
            Specificity : 0.8631
         Pos Pred Value : 0.8735
         Neg Pred Value : 0.9410
             Prevalence : 0.5000
         Detection Rate : 0.4729
   Detection Prevalence : 0.5414
      Balanced Accuracy : 0.9045

       'Positive' Class : 0
```

Figure 34: Confusion matrix and best tuned SVM model performance metrics (Test set)

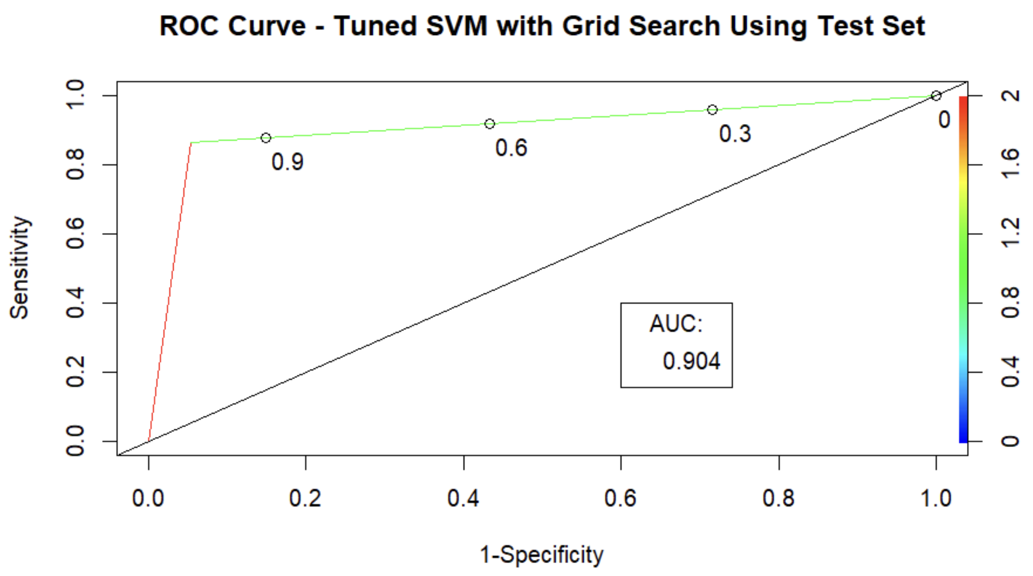


Figure 35: ROC curve for grid search tuned SVM model using test set

5.6 Baseline Random Forest

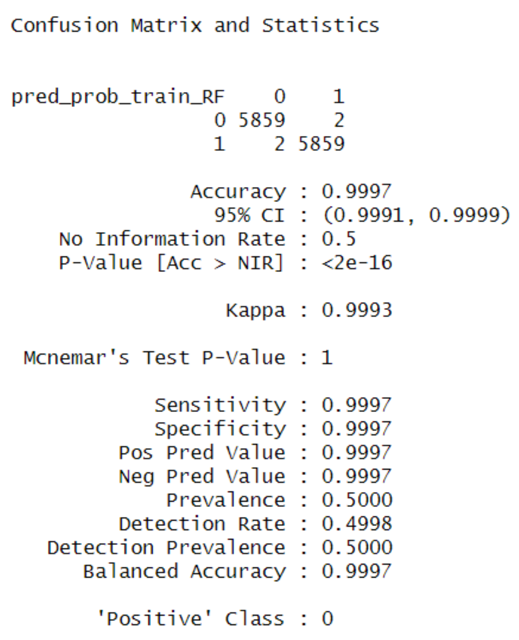
The fifth model to be implemented is baseline random forest. randomForest function is used to build baseline random forest model. Inside this function, the variable Exited is inputted as the target variable to be predicted and the remaining variables are inputted as the predictors in the formula, in addition to the inclusion of the train_bank dataset as the data frame. As a result, 500 trees are generated (ntree = 500), with five variables being attempted per split (mtry = 5). The estimated rate of prediction error of baseline random forest is 2.35%. The classification error for true and false positives is 4.14%, and 0.55% for true and false negatives.

5.6.1 Training Set

Using the predict function, the forecasted probabilities with the baseline random forest model as the input are generated for the training set, which is then combined in a table along with the target variables' values in the training set to form the confusion matrix ultimately.

Figure 36 demonstrates the confusion matrix and the baseline random forest model performance metrics using the training set. There are 5,859 instances of both true positives and negatives respectively, whereas there are two false positives and two false negatives. The percentages of accuracy, recall, f1-score and precision of the training set are 99.97% each.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 37. As a result, the value of AUC for the baseline random forest model on the training set is 1.



```
Confusion Matrix and Statistics


pred_prob_train_RF    0    1
                 0 5859    2
                 1    2 5859

               Accuracy : 0.9997
                 95% CI : (0.9991, 0.9999)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9993

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9997
            Specificity : 0.9997
         Pos Pred Value : 0.9997
         Neg Pred Value : 0.9997
             Prevalence : 0.5000
         Detection Rate : 0.4998
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.9997

       'Positive' Class : 0
```

Figure 36: Confusion matrix and baseline random forest model performance metrics (Train set)

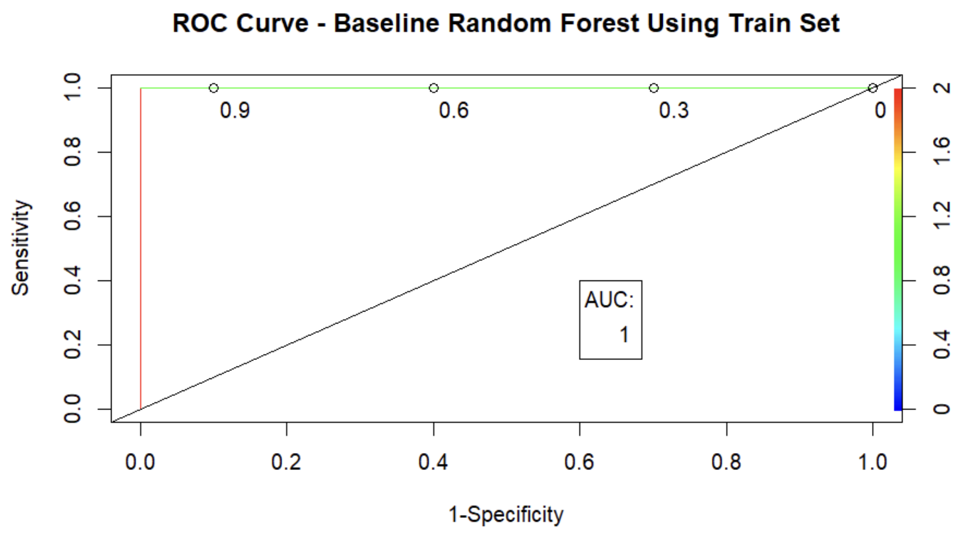


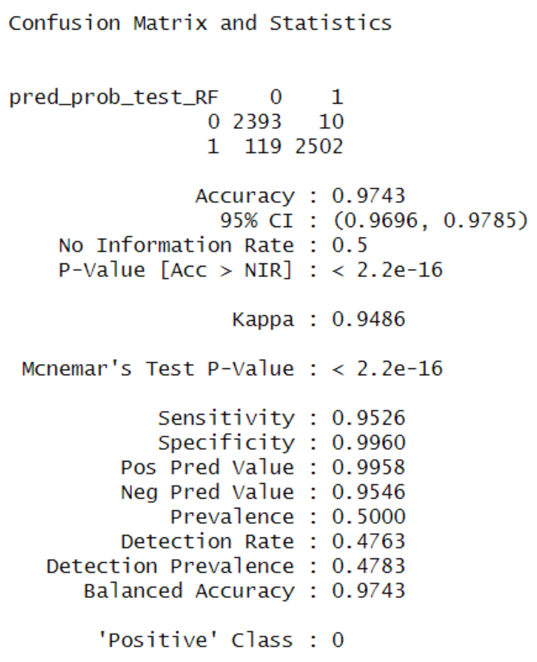
Figure 37: ROC curve for baseline random forest model using training set

5.6.2 Test Set

Using the predict function, the forecasted probabilities with the baseline random forest model as the input are generated for the training set, which is then combined in a table along with the target variables' values in the training set to form the confusion matrix ultimately.

Figure 38 demonstrates the confusion matrix and the baseline random forest model performance metrics using the test set. For true positives, there are 2,393 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 2,502 instances for the prediction that the customer has churned, and it fits with the true case. For false positives, there are 10 instances of the prediction that the customer has churned when churn has not happened actually. For false negatives, there are 119 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 97.43% and 99.58% respectively. The percentage of recall is 95.26% and the percentage of f1-score is 97.37%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 39. As a result, the value of AUC for the baseline random forest model on the test set is 0.974. In terms of variable importance, complaint lodgment is the most important variable in predicting customer churn. Specifically, the coefficient values of the mean decrease in Gini for making and not making complaints are 1478.58 and 1549.98 respectively (see Figure 40).



```
Confusion Matrix and Statistics


pred_prob_test_RF    0    1
                0 2393   10
                1  119 2502

               Accuracy : 0.9743
                 95% CI : (0.9696, 0.9785)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9486

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9526
            Specificity : 0.9960
         Pos Pred Value : 0.9958
         Neg Pred Value : 0.9546
             Prevalence : 0.5000
         Detection Rate : 0.4763
   Detection Prevalence : 0.4783
      Balanced Accuracy : 0.9743

       'Positive' Class : 0
```

Figure 38: Confusion matrix and baseline random forest model performance metrics (Test set)

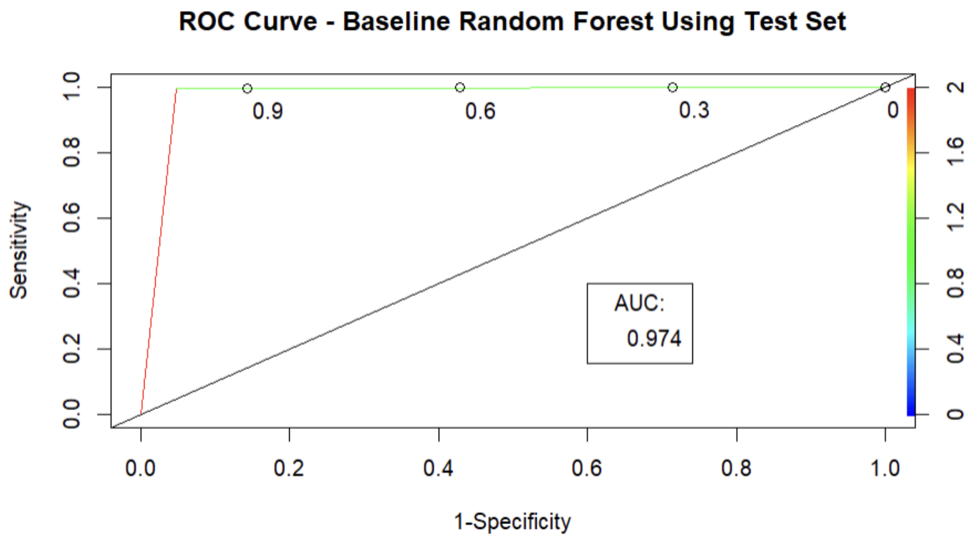


Figure 39: ROC curve for baseline random forest model using test set

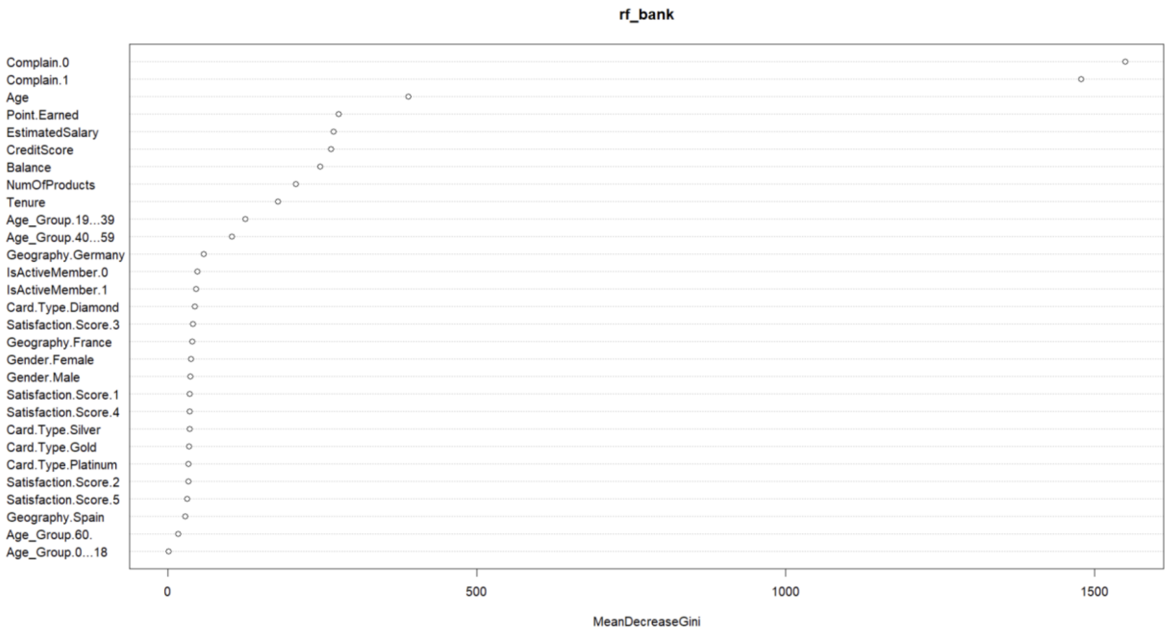


Figure 40: Variable importance plot

5.7 Tuned Random Forest with Grid Search

The final model to be implemented is hyperparameter tuned random forest using grid search. 10-fold cross validation is implemented in building the model to prevent overfitting of the tuned model using the trainControl function. To build the tuned random forest model, the tuning grid needs to be defined. In this case, 15 mtry values, or the number of variables attempted per splits, from one to 15, is included in the tuning grid. Then, the train function is used for hyperparameter tuning under the accuracy metric by including Exited as the target variable for prediction and the remaining variables are inputted as the predictors in the formula, in addition to the inclusion of the train_bank dataset as the target data frame, 10-fold cross-validation and tuning grid as the tuning range. Based on Figure 41, the greatest accuracy of 97.59% is achieved by mtry equals 7, all values of which are used in the final model.

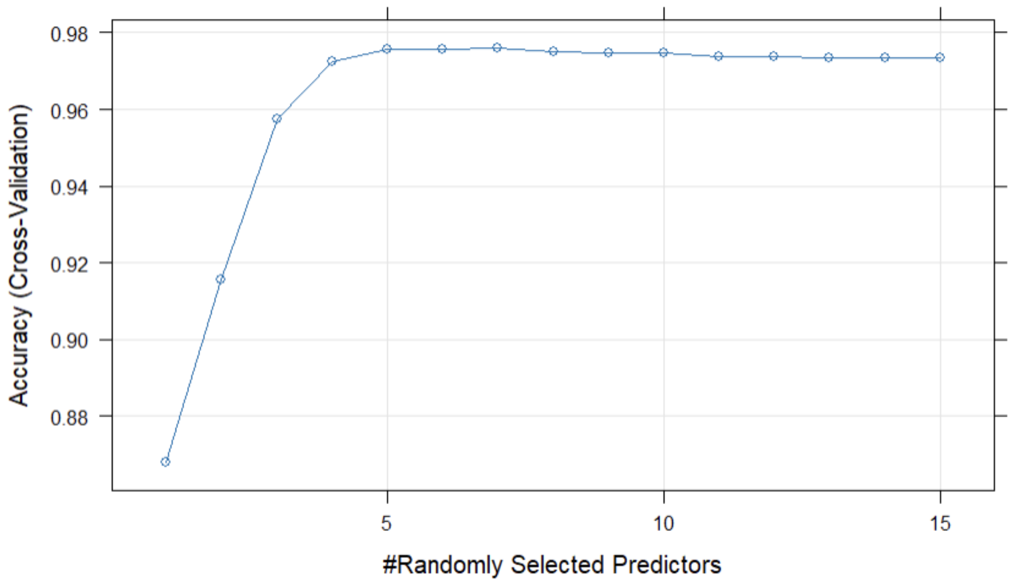
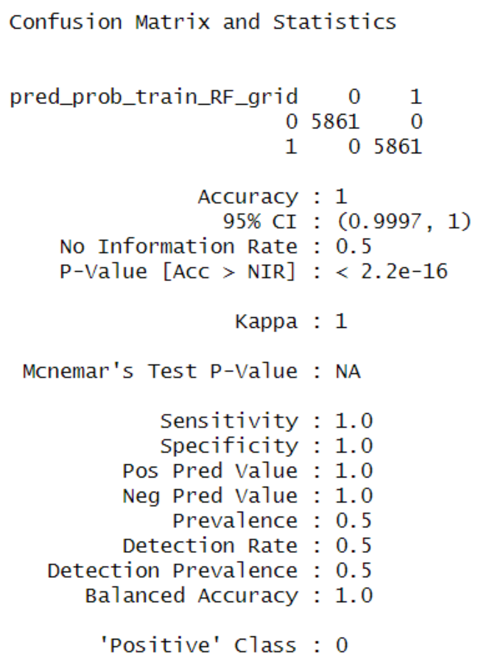


Figure 41: The plot of accuracy (cross-validation) against number of randomly selected predictors

5.7.1 Training Set

Using the predict function, the forecasted probabilities with the baseline random forest model as the input are generated for the training set, which is then combined in a table along with the target variables' values in the training set to form the confusion matrix ultimately. Figure 41 demonstrates the confusion matrix and the tuned random forest model performance metrics using the training set. There are 5,861 instances of both true positives and negatives respectively, whereas there are no false positives and negatives. The percentages of accuracy, recall, f1-score and precision of the training set are 100% each. To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 42. As a result, the value of AUC for the tuned random forest model on the training set is 1.



```
Confusion Matrix and Statistics


pred_prob_train_RF_grid    0    1
                      0 5861    0
                      1    0 5861

               Accuracy : 1
                 95% CI : (0.9997, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.5
         Detection Rate : 0.5
   Detection Prevalence : 0.5
      Balanced Accuracy : 1.0

       'Positive' Class : 0
```

Figure 41: Confusion matrix and tuned random forest model performance metrics (Train set)

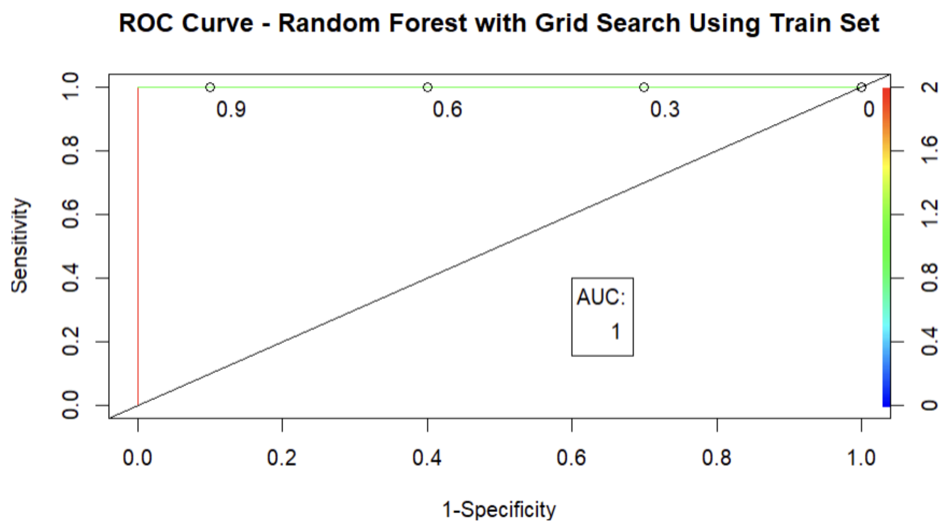


Figure 42: ROC curve for grid search tuned random forest model using training set

5.7.2 Test Set

Using the predict function, the forecasted probabilities with the baseline random forest model as the input are generated for the training set, which is then combined in a table along with the target variables' values in the training set to form the confusion matrix ultimately. Figure 43 demonstrates the confusion matrix and the tuned random forest model performance metrics using the test set. For true positives, there are 2,391 instances for the prediction that the customer has not churned, and it fits with the true case. For true negative, there are 2,506 instances for the prediction that the customer has churned, and it fits with the true case. For false positives, there are 6 instances of the prediction that the customer has churned when churn has not happened actually. For false negatives, there are 121 instances of the prediction that the customer has not churned when churn has happened actually. The percentages of accuracy and precision of the training set are 97.47% and 99.75% respectively. The percentage of recall is 95.18% and the percentage of f1-score is 97.41%.

To draw the ROC curve, we need to obtain the sensitivity value and rate of false positive from the true and forecasted labels for the Exited class followed by the AUC calculation then round up to three decimal points, as shown in Figure 44. As a result, the value of AUC for the tuned random forest model on the test set is 0.975.

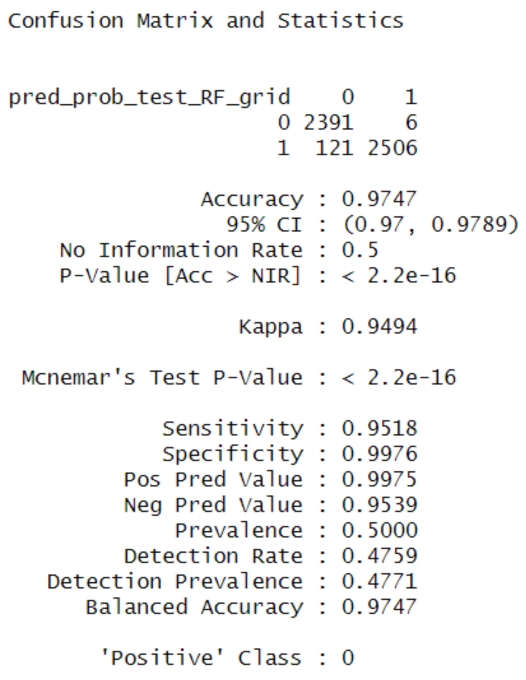

```
Confusion Matrix and Statistics


pred_prob_test_RF_grid    0    1
                     0 2391    6
                     1  121 2506

               Accuracy : 0.9747
                 95% CI : (0.97, 0.9789)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9494

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9518
            Specificity : 0.9976
         Pos Pred Value : 0.9975
         Neg Pred Value : 0.9539
             Prevalence : 0.5000
         Detection Rate : 0.4759
   Detection Prevalence : 0.4771
      Balanced Accuracy : 0.9747

       'Positive' Class : 0
```

Figure 43: Confusion matrix and tuned random forest model performance metrics (Test set)

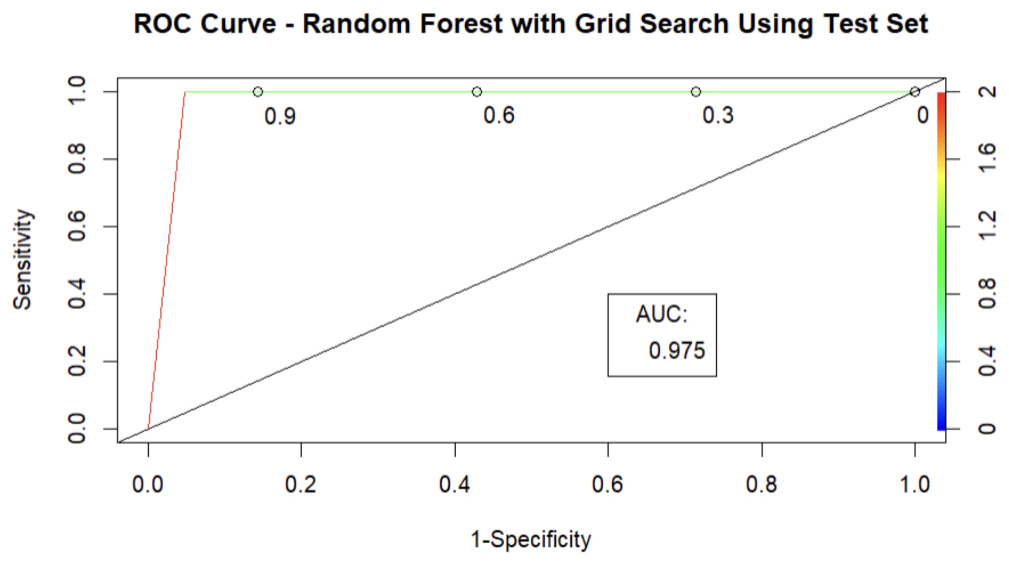


Figure 44: ROC curve for grid search tuned random forest model using the test set

6.0 Model Validation

Models Implemented		**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**AUC**
LR (Baseline)	Train	88.14	95.33	83.35	88.94	0.881
	Test	87.66	94.51	83.12	88.45	0.877
L1-Regularised LR	Train	88.03	95.51	83.08	89.04	0.880
	Test	82.78	94.9	82.78	88.43	0.876
Polynomial SVM (Baseline)	Train	91.73	88.71	95.63	92.04	0.917
	Test	90.47	87.55	94.35	90.82	0.905
Grid Search Tuned SVM	Train	91.38	88.14	95.62	91.73	0.914
	Test	90.45	87.35	94.59	90.83	0.904
RF (Baseline)	Train	99.97	99.97	99.97	99.97	1
	Test	97.43	99.58	95.26	97.37	0.974
Grid Search Tuned RF	Train	100	100	100	100	1
	Test	97.47	99.75	95.18	97.41	0.975

Based on the result summary table above, the overall performance for baseline random forest and grid search tuned random forest models is the greatest across all models. In terms of comparison between these two models, the accuracy, precision, F1-measure, and AUC value for the tuned random forest model are greater than the baseline model except for sensitivity despite the small margin between each metric. The second best-performing model revolves around polynomial SVM and grid search-tuned SVM. Specifically, similar to the between-metric small margin exhibited by random forest and its tuned model, the accuracy, precision, and AUC value of polynomial SVM is slightly greater than the SVM model tuned with grid search. In terms of baseline, the model performance of the random forest is greater than SVM and logistic regression. For grid search optimization, the tuned model for the random forest is better in performance than SVM. Overall, the accuracy, precision, sensitivity, F1-measure and AUC value for the training sets across all models are greater than the test sets.

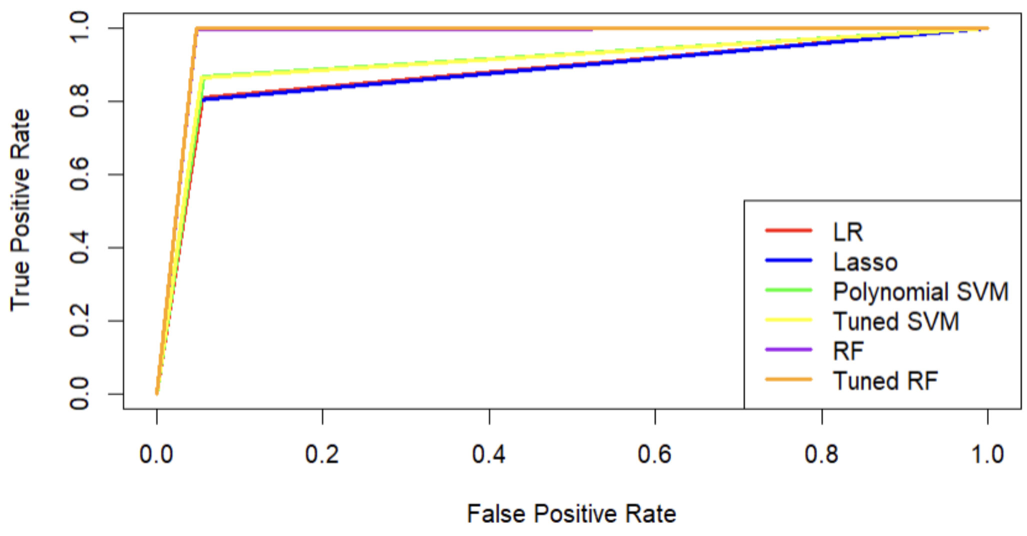


Figure 45: Combined ROC curves across all models

According to Figure 45 above, the area under the curve for baseline and tuned random forest models is greater than other models. Therefore, it can be concluded that the best-performing models are baseline and tuned random forests.

7.0 Analysis & Recommendations

7.1 Logistic Regression

Authors	Accuracy	Precision	Recall	F1-Score	AUC
Baseline LR (current)	87.66%	94.51%	83.12%	88.45%	87.7%
Tuned LR (current)	82.78%	94.9%	82.78%	88.43%	87.6%
Singh et al. (2014)	69.10%	-	71.40%	49.70%	76.70%
Tékouabou et al. (2022)	67.00%	-	-	67.00%	-
Anitha & Sherly (2023) – telco dataset	81.30%	66.60%	52.30%	58.60%	75.20%
Anitha & Sherly (2023) – bank dataset	78.10%	36.30%	4.70%	8.30%	57.80%
Ullah et al. (2019)	-	49.00%	70.00%	57.70%	49.60%
Kaur & Kaur (2020)	82.19%	59.87%	11.91%	-	73.19%
Lemos et al. (2022)	76.20%	74.00%	-	77.26%	-
Lalwan et al. (2022)	80.45%	79.11%	80.23%	78.89%	82.00%

Based on the table above detailing the results comparison between this project and other past studies, the accuracy, precision, recall, F1-score and AUC value for the baseline logistic regression model produced in the current project are greater than all studies listed above. However, there are some differences in terms of methods. For example, Ullah et al. (2019) and Lemos et al. (2022) performed 10-fold cross-validation and repeated 10-fold cross-validation on their logistic regression model respectively but in this study, cross-validation is not applied to the baseline model. This suggests that the baseline logistic regression model in this study has the possibility of overfitting. Besides, all the studies above did not perform Lasso regularization or any hyperparameter tuning on the model, so comparison with other studies for the current tuned logistic regression model is difficult. One thing to note here is that the difference in accuracy between Kaur & Kaur (2020) and the tuned model for this study is the smallest of all.

7.2 Support Vector Machine

Authors	Accuracy	Precision	Recall	F1-Score	AUC
Baseline SVM (current)	90.47%	87.55%	94.35%	90.82%	90.5%
Tuned SVM (current)	90.45%	87.35%	94.59%	90.83%	90.4%
Singh et al. (2014)	71.90%	-	67.20%	50.50%	76.50%
Tékouabou et al. (2022)	57.00%	-	-	64.00%	-
Anitha & Sherly (2023) – telco dataset	79.90%	64.50%	45.60%	53.40%	73.90%
Anitha & Sherly (2023) – bank dataset	77.50%	31.90%	5.40%	9.20%	55.60%
Muneer et al. (2022)	77.60%	-	100.00%	89.00%	-
Lalwani et al. (2022)	80.21%	79.66%	80.64%	78.11%	80.00%
Lemos et al. (2022)	80.30%	81.00%	-	80.09%	-

Based on the table above detailing the results comparison between this project and other past studies, the accuracy, precision, recall, F1-score and AUC value for the baseline and tuned SVM model produced in the current project are greater than all studies listed above. The exception here is Muneer et al. (2022) whose baseline SVM model produced greater recall than this project, and the F1-score difference is small too. This can be because of the use of methods to address class imbalance. This project uses ROSE for oversampling while Muneer et al. (2022) used SMOTE. In terms of polynomial kernel, the tuned model performance for SVM in the current project is greater than Lalwani et al. (2022). For other studies listed above, it is unknown what type of kernel they used for their SVM model.

7.3 Random Forest

Authors	Accuracy	Precision	Recall	F1-Score	AUC
Baseline RF (current)	97.43%	99.58%	95.26%	97.37%	97.40%
Tuned RF (current)	97.47%	99.75%	95.18%	97.41%	97.50%
Singh et al. (2014)	78.30%	-	69.30%	57.70%	83.10%
Tékouabou et al. (2022)	86.00%	-	-	86.00%	-
Anitha & Sherly (2023) – telco dataset	79.30%	61.50%	48.10%	54.00%	72.60%
Anitha & Sherly (2023) – bank dataset	86.20%	78.80%	47.50%	59.20%	83.00%
Muneer et al. (2022)	88.70%		89.00%	91.00%	
Lalwan et al. (2022)	78.04%	77.54%	78.68%	77.91%	82.00%
Lemos et al. (2022)	82.80%	84.40%	-	82.25%	-
Pamina et al. (2019)	77.50%	-	-	50.60%	-
Ullah et al. (2019)	-	89.30%	88.80%	88.20%	94.70%

Based on the table above detailing the results comparison between this project and other past studies, the accuracy, precision, recall, F1-score and AUC value for the baseline random forest regression model produced in the current project are greater than all studies listed above. In terms of hyperparameter tuning, we can compare our tuned model results with Singh et al. (2024) who share the same dataset and same grid search tuning. The performance of our tuned model is better than Singh et al. (2024). This can be because of the feature selection technique which is not applied here.

7.4 Discussion of Results

Based on the results of this project, the random forest model, both baseline and tuned, is the best-performing model. It is not surprising as this is in line with most of the past studies listed in the related works section. This performance superiority can be due to the fact that random forest is able to better model the highly non-linear and complex patterns between features for customer churn compared to the other two methods. Besides, SVM and logistic regression are single classifiers, while random forest is an ensemble learning method that can build the model with greater accuracy after taking the collective decisions of decision trees into account.

There is little difference in performance between the baseline and tuned SVM model. This can be attributed to the tuning process for the SVM model as the tuning process is defined with the radial kernel by default instead of changing it to polynomial. One of the recommendations is setting the kernel to polynomial during SVM tuning so that the results can be compared directly. The worst performing model is L1-regularised logistic regression as the accuracy of this model is less than the baseline logistic regression model. One of the possible reasons for this accuracy difference can be the improper tuning of the hyperparameters as the logistic regression model may be regularised too much or too little which affects the result, therefore, proper tuning is needed in the future.

8.0 Conclusion

One of the things that went wrong at the beginning of this project is that my model performance for all models was within the 99% and 100% zone even after I addressed the class imbalance, normalization and cross-validation. Fortunately, one of the suggestions given by our classmate and also Prof. Mandava is to create missing values to the dataset with no missing values at all. As a result, the scores of the performance metrics drop to a logical level. One of the things that went right was choosing a random forest to build the model and it is the best-performing model ultimately which is in line with the results in most of the articles I found at the beginning of this subject when I didn’t even know the type of models.

To summarise, throughout this experiment, I learned how to do data preprocessing, especially on the creation of missing values and one hot encoding. I also learned how to build baseline and tuned model. The tuning process is indeed very long but thankfully it comes to fruition at the end. In the future, I would like to try doing this experiment again but using Python so that I can compare the results from R and Python. Besides, I also want to learn building hybrid model as this approach is also proven effective in predicting churn.

References

Anitha, M. A., & Sherly, K. K. (2023). An Efficient Hybrid Classifier Model for Customer Churn Prediction. *International Journal of Electronics and Telecommunications*, *69*. https://doi.org/10.24425/ijet.2023.144325

de Lima Lemos, R. A., Silva, T. C., & Tabak, B. M. (2022). Propension to customer churn in a financial institution: A machine learning approach. *Neural Computing and Applications*, *34*(14), 11751-11768. https://doi.org/10.1007/s00521-022-07067-x

Karvana, K. G. M., Yazid, S., Syalim, A., & Mursanto, P. (2019, October). Customer churn analysis and prediction using data mining models in banking industry. In *2019 international workshop on big data and information security (IWBIS)* (pp. 33-38). IEEE. https://doi.org/10.1109/IWBIS.2019.8935884

Kaur, I., & Kaur, J. (2020, November). Customer churn analysis and prediction in banking industry using machine learning. In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 434-437). IEEE. https://doi.org/10.1109/PDGC50313.2020.9315761

Khodabandehlou, S., & Zivari Rahman, M. (2017). Comparison of supervised machine learning techniques for customer churn prediction based on analysis of customer behavior. *Journal of Systems and Information Technology*, *19*(1/2), 65-93. https://doi.org/10.1108/JSIT-10-2016-0061

Lalwani, P., Mishra, M. K., Chadha, J. S., & Sethi, P. (2022). Customer churn prediction system: a machine learning approach. *Computing*, 1-24. https://doi.org/10.1007/s00607-021-00908-y

Muneer, A., Ali, R. F., Alghamdi, A., Taib, S. M., Almaghthawi, A., & Ghaleb, E. A. (2022). Predicting customers churning in banking industry: A machine learning approach. *Indones. J. Electr. Eng. Comput. Sci*, *26*(1), 539-549. https://www.academia.edu/download/92955405/58_27450_v26i1_Apr22.pdf

Pamina, J., Raja, B., SathyaBama, S., Sruthi, M. S., & VJ, A. (2019). An effective classifier for predicting churn in telecommunication. *Jour of Adv Research in Dynamical & Control Systems*, *11*. https://ssrn.com/abstract=3399937

Singh, P. P., Anik, F. I., Senapati, R., Sinha, A., Sakib, N., & Hossain, E. (2024). Investigating customer churn in banking: A machine learning approach and visualization app for data science and management. *Data Science and Management*, *7*(1), 7-16. https://doi.org/10.1016/j.dsm.2023.09.002

Tékouabou, S. C., Gherghina, Ș. C., Toulni, H., Mata, P. N., & Martins, J. M. (2022). Towards explainable machine learning for bank churn prediction using data balancing and ensemble-based methods. *Mathematics*, *10*(14), 2379. https://doi.org/10.3390/math10142379

Ullah, I., Raza, B., Malik, A. K., Imran, M., Islam, S. U., & Kim, S. W. (2019). A churn prediction model using random forest: analysis of machine learning techniques for churn prediction and factor identification in telecom sector. *IEEE access*, *7*, 60134-60149. https://doi.org/10.1109/ACCESS.2019.2914999

Acknowledgement

I would like to express my gratitude to Prof Mandava for allowing me to conduct this really fun project. I want to also thank my friends who provided suggestions for my code so that I can improve on my coding which are one of my weaknesses. I also want to thank Kaggle for the dataset provided.