INDIVIDUAL PROJECT

HO WENG TIM

TITLE:
SUPERVISED MACHINE LEARNING APPROACHES UNDER DIFFERENT RESAMPLING METHOD IN PREDICTING E-COMMERCE CUSTOMER CHURN

ABSTRACT

E-commerce is a business model that purchases and sells products through online method. The e-commerce business will remain very competitive in the future, as the global trend toward digitization accelerates. Imbalanced datasets in customer churn prediction will always be a concern since it can affect the model performance. However, the optimal machine learning approach for maintaining good performance under various resampling methods in the field of customer churn is yet unknown. Hence, it is important to determine the best machine learning approach that able to give high performance across various resampling methods. In this study, the e-commerce churn data is obtained from Kaggle website. Target variable distribution is explored and the original data is balanced by different resampling method: Synthetic Minority Over-sampling Technique (SMOTE), random oversampling, random under sampling and random both sampling. Supervised machine leaning approach such as support vector machine, logistic regression and random forest are built on original dataset and the four balanced datasets. Random search and cross validation are carried out for hyperparameter tuning to obtain the optimum model. This study showed that tuned random forest is the best machine learning approach to be utilized across different resampling dataset in the e-commerce customer churn prediction.

1.0 Introduction

1.1 General Introduction

Rapid market expansion in every industry has resulted in a larger user base for business vendors. E-commerce, alternatively referred to as electronic commerce, is a business model that purchases and sells products and goods, as well as the transaction of payments through the internet (Chai et al., 2020). The expense of customer acquisition has been estimated to be ten times that of retaining existing customers, emphasizing the necessity of the customer turnover study (Celik & Osmanoglu, 2019). It is consequently necessary for business operators to prevent churn—a situation in which a client chooses to discontinue using a company's services (Kamalraj & Malathi, 2013). Customer attrition is a significant issue and one of the primary worries of big enterprises. Due to the significant impact on firms' earnings, organizations are attempting to develop methods for predicting probable churn rates. Thus, identifying the variables that contribute to client turnover is critical in order to undertake the required steps to decrease this churn (Ahmad et al., 2019).

Machine learning (ML) is one of the popular techniques used to predict customer churn rates (Soumi et al., 2021). ML is a subset of artificial intelligence (AI) that enables technology systems to improve their predictive accuracy and usually are used to estimate upcoming results by using past data as feed. Machine learning is critical since it provides organizations with insight into trends in consumer behaviour and organizational processes, as well as aids in the creation of innovative services. ML can assist businesses in developing a more complete understanding of their clients by discovering correlations and assisting businesses in optimizing business marketing and development strategies to customer demands by gathering customer information and associating it with actions over time (Burn, 2021).

In ML, there are four fundamental strategies are available: reinforcement learning, semi-supervised learning, unsupervised learning, and supervised learning (Burn, 2021). Supervised machine learning is utilized in this project to predict e-commerce customer churn because it is a binary classification task. Supervised machine learning is commonly used to solve classification or regression problems, it relies on labelled data to train algorithms capable of reliably classifying data or predicting outcomes. Supervised learning instructs models to produce the desired output using a training dataset. The training set contains both input and correct outcomes, allowing the model to steadily improve accuracy by making adjustments till the error is suitably reduced. (IBM, 2020).

The most common problem in the customer churn dataset is the data imbalance problem and hence lower the accuracy in machine learning model due to the lack of churn information to train the model (Vera, 2020). The primary issue in this instance is that the number of clients who do not turnover is far higher than the number of clients who do (Bhattara et al., 2019). An unbalanced classification issue is one in which the allocation of instances across recognized classes is uneven or biased. The proportion can range from a little bias to a significant inequity in which there are higher numbers in the majority class but lower in the minority class. Imbalanced classifications provide difficulty for predictive modeling because the majority of ML approaches for classification were created under the premise of an identical sample size for each category. As a result, models with a poor accuracy rate for the minority class was created. This is an issue because, in most cases, the minority class is more significant than the majority class, and hence the problem is more sensitive to minority class classification errors than to majority class classification errors (Brownlee, 2020). There are many different balancing methods are proposed by different studies, but the most popular method are SMOTE and ROSE methods.

In years to come, increased competition, creative and inventive business strategies, and upgraded offerings all contribute to an increase in client acquisition costs. In such a difficult environment, business operators have recognized the critical nature of maintaining existing clients (Kamalraj & Malathi, 2013). The imbalance dataset in the field of customer churn still remains a problem. Hence, data balancing always need to carry out. It is important to address an effective machine learning approach that can accurately predict the customer churn when there are using across different resampling method. In this project, effective machine learning models were developed on different resampling methods datasets to examine the best machine learning model that can give high performance across all different resampling dataset, so that company is able to recognize the appropriate machine learning model that able to effectively identify the churn risk.

1.2 Problem Statement

There is a lack of research that study the performance of machine learning models on datasets that use different balancing methods in the field of e-commerce customer churn. The most common problem in the churn prediction dataset is the imbalance of data. Imbalance data due to there is smaller population of customer churn when compared to customers who are staying.

In this situation, the classifier is more likely to create a biased learning approach with worse forecast validity for minority categories than for majority categories. The role of imbalanced data cannot be ignored, however, based on the literature review, most of the related studies study does not mention clearly about data balancing part, do not carry out data balancing, or just use only one data balancing method. Hence, there is a lack of knowledge to understand the best machine learning algorithm that can provide high performance across datasets that used different balancing method in the field of e-commerce customer churn.

1.3 Research Aim and objectives

The main aim of this project is to determine the best machine learning model that can give high performance across different resampling datasets for the e-commerce customer churn rate prediction.

The project objectives are:

i) To resample the imbalanced dataset with SMOTE, random oversampling, random under sampling and random both samplings.

ii) To build machine learning models for the e-commerce customer churn prediction by using Random Forest (RF), Logistic Regression (LR), and Support Vector Machine (SVM).

iii) To optimize and fine-tune the models to be the best fit for accurately predicting the e-commerce customer churn.

iv) To evaluate the performance of the machine learning models that are built on e-commerce customer churn prediction.

1.4 Research Scope

The scope of this project is using the e-commerse dataset that obtain from Kaggle. This work is limited to predicting the customer churn rate in the field of eCommerce. Due to the nature of the problem which concerns predicting customer churn, the type of problem is a binary classification task. As such, the type of machine learning approach used is supervised machine learning because the target variable is a categorical variable while supervised machine learning is popular approach for classification task. The type of supervised machine learning algorithms that were utilized included logistic regression (LR), random forest (RF), and support vector machine-Radial Basis Function (SVM-RBF).

For the splitting method, only stratified sampling method is used. These models are built on the dataset that was resampling by 4 different resampling methods which include Synthetic Minority Over-sampling Technique (SMOTE), random oversampling, random undersampling and random both sampling methods under the package of Random Over-Sampling Examples (ROSE). Hence, there are total 5 kinds of datasets are utilized which include the original dataset, the dataset after SMOTE resampling, datasets after random oversampling, dataset after random under sampling and dataset after random both sampling.

The hyperparameters tuning method in this project is only limited to random search while cross-validation is limited to only 10-fold. Hyperparameter tuning method is only utilized on the support vector machine algorithm and random forest algorithm. So, in the end of this project, there are total 5 algorithms were built which include the based SVM-RBF model, tuned SVM-RBF model, logistic regression, based random forest model and tuned random forest model. The evaluation matrix only limited on accuracy, sensitivity, specificity, F1-score, confusion matrix, area under curve (AUC) and receiver operating characteristic (ROC) curve. Last but not least, the programming language that used to built the machine learning approaches is R programming.

2.0 Related Work

2.1 Literature Review

Table 2.1 summarises studies that used machine learning algorithms to predict eCommerce customer attrition. Table 2.2 summarises the technique and findings. The objective of conducting a literature review is to gather fundamental information about the study issue, identify gaps, develop one's own approach to the topic, summarise major findings, and discuss the topic.

Table 2.1: Literature Review Matrix in Customer Churn Field

Reference	Dataset & Size (Row x Col)	Methodology	Result (Accuracy %)	Conclusion	Author Recommendation	Own comment
(Lemos et al., 2022)	Financial institution customer churn (500,000 x 35)	**EDA:** Uncover outlier, data types, missing value, summary statistics, visualization method **Pre-processing:** feature selection, feature creation, missing value imputation, standardization **ML algorithm:** DT, KNN, Elastic net, LR, SVM, RF **Cross-validation:** 10-fold	DT: 78.2 KNN: 77.9 Elastic net: 76.2 LR: 76.2 SVM: 80.3 **RF: 82.8**	Random forest has higher performance in term of accuracy, precision, and F-measure	*NA*	Did mention weight tuning method but didn't clearly mention the hyperparameter tuning method Did not explore data balancing in the target variable.

		Weight tuning: ensemble's voting scheme **Evaluation parameter:** AUC-ROC, Accuracy, Precision, and F-measure, confusion matrix				
(Miao & Wang, 2022)	Credit card customer churn dataset (1000 x 21)	**EDA:** Identify patterns, missing value exploration, duplicated values, outlier, visualization for relationship between feature and target and data normality. **Data Pre-processing:** Label encoding, feature selection, train-test split, standardization **ML algorithm:** LR, RF, KNN **Hyperparameter tuning**: Grid Search **Cross-Validation**: 5-fold **Evaluation parameters:** Confusion matrix, ROC,	LR: 90.36 **RF: 96.10** KNN: 90.32		• Random forest is the best performance model • By utilising a more optimal parameter arrangement, the model's accuracy can be enhanced.	Collect more dataset from a variety industry Use ensemble model Try more different algorithm	Used label encoding in nominal data Did not explore data balancing in the target variable.

		AUC, accuracy, precision, recall				
(Wu et al. 2021)	Telco customer churn dataset: 3 Dataset **Dataset 1:** 7032 x 21 **Dataset 2:** 4031 x 20 **Dataset 3**: 51047 x 58	**EDA:** Univariate Analysis & Bivariate Analysis **Data pre-processing:** Data cleaning, data transformation, data normalization, Data balancing (SMOTE), feature selection **ML algorithm:** LR, DT, RF, NB, AdaBoost, Multi-layer Perceptron **Cross-Validation:** 10-fold **Evaluation parameters**: Accuracy, Precision, Recall, F1-score and AUC	**Dataset 1 (Original dataset)** **LR: 80.19** DT:78.33 RF:79.55 NB:75.14 AdaBoost: 80.08 Multi-layer Perceptron: 80.12 **Dataset 1 (After SMOTE balancing)** LR: 74.82 DT:76.74 RF:76.99 NB:73.76 **AdaBoost: 77.19** Multi-layer Perceptron: 75.60	• F1-score is regarded to be an essential statistic to evaluate the models for unbalanced datasets • Random Forest has best performance in dataset 2 and 3.	Study other oversampling and undersampling method for balancing Select more reasonable threshold for ROC Use Bayesian optimization hyperparameter tuning	In overall, this study is quite complete but would like to suggest do hyperparameter tuning and use more balancing method such as random oversampling, random under sampling and random both sampling.

			Dataset 2 (Original dataset) LR: 87.42 DT:94.59 **RF:95.34** NB:88.14 AdaBoost: 88.27 Multi-layer Perceptron: 95.26 **Dataset 2 (After SMOTE balancing)** LR: 77.18 DT:92.58 **RF:93.60** NB:78.64 AdaBoost: 86.21 Multi-layer Perceptron: 90.37			

			<u>**Dataset 3 (Original dataset)**</u> **LR:71.05** DT:68.36 RF:68.85 NB:68.84 AdaBoost: 70.01 Multi-layer Perceptron: 69.56 <u>**Dataset 3 (After SMOTE balancing)**</u> LR: 57.60 DT: 59.38 **RF:63.09** NB:59.29 AdaBoost: 58.63 Multi-layer Perceptron: 53.47			

(Xiahou & Harada, 2021)	E-commerce customer churn dataset published by the Alibaba Cloud Tianchi platform (987994 x 17)	First, divide the dataset customer into three clusters by k-mean clustering. Then, carry out the methodology below for every cluster. **EDA:** *Done but did not mention detail* **Data Pre-processing:** Change data timestamp format, data balancing by SMOTE **ML algorithm:** SVM, LR **Cross-Validation**: 10-fold **Evaluation parameters:** Accuracy, Precision, Recall, AUC	**<u>Before segmentation</u>** **SVM:90.81** LR: 90.65 **<u>After segmentation</u>** **Cluster 1** **SVM:92.56** LR:90.50 **Cluster 2** **SVM:91.58** LR: 90.98 **Cluster 3** **SVM: 90.53** LR: 90.50 **Average** **SVM: 91.56** LR: 90.66	Support vector machine has higher accuracy than logistic regression	Gather and analyze consumer behavior data from several oorganizationsto further improve the model's generalizability. Predict and evaluate continuously.	The exploratory data analysis and pre-processing part is not clear If compare the result before and after balancing with be a good insight. Can use more balancing method The number of variables is less The number of algorithms is less

(Kaur & Kaur, 2020)	Customer churn in banking industry dataset from Kaggle (28382 x 21)	Compare between baseline feature and all features in term of without stratified sampling, stratified sampling and 8-fold validation **EDA:** Explore missing value, data balancing, skewness, and bivariate analysis, visualization **Pre-processing:** Impute missing value, feature selection, data normalization, data balancing **ML algorithm:** LR, DT, KNN, and RF **Cross-Validation**: 8-fold **Evaluation parameter:** Recall, Precision, Area Under the Curve-Receiver Operating	<u>**Baseline feature (After feature selection)**</u> ***Stratified sampling*** LR:82.56 DT:84.89 KNN:81.43 **RF:85.23** ***Without stratified sampling*** LR:82.47 DT:83.65 KNN:81.77 **RF:85.13** **8-fold validation** LR:82.04 **DT:85.20** KNN:81.31 RF:84.55	Random Forest perform has higher performance	• Use more advance machine learning such as ANN and esemble model	The data balancing method is not mentioned. ROC curve is not show

		Characteristics (AUC-ROC), and Accuracy	**All features** ***Stratified sampling*** LR:82.19 **DT:85.25** KNN:81.20 RF:85.21 ***Without stratified sampling*** LR:82.85 DT:82.43 KNN:81.29 **RF:84.79** ***8-fold validation*** LR:82.41 DT:83.37 KNN:81.03 **RF:84.35**			

(Wadikar, 2020)	Customer data of the financial institute (96967 x 48)	**EDA:** uncover missing value, outlier, normality, correlation between dependent and independent variable, visualization method **Data Pre-processing:** Data balancing (SMOTE), feature selection, remove outlier, impute missing values, data transformation, encoding **ML algorithm:** LR, RF, SVM, NN **Evaluation parameter:** Confusion matrix, accuracy, specificity, sensitivity, F1 score, ROC	**Before SMOTE balancing** LR: 87% **RF: 96%** SVM: 86% NN: 91% **After SMOTE balancing** LR: 85% **RF: 97%** SVM: 89% NN: 91%	The random forest has the highest accuracy in both balanced and imbalanced dataset	Explore a variety dataset and other machine learning algorithms. Build time series model for customer churn	Cross-validation, hyperparameter tuning is suggested
(He et al., 2020)	real-life dataset received from Markel Corporation (25275 x 253)	**EDA:** uncover missing value, correlation matrix, a visualization method **Pre-processing:** Missing value imputation, feature	***The authors do not use accuracy as evaluation parameter, hence AUC is reviewed in this session.***	Extra Tree Classifier and Gradient Boost are the optimal models	*NA*	The value of cross validation is not mention Did not discuss about the exploration of

		selection, variable encoding, limit outliers by Winsorizing, standardization, train-test split **ML algorithm**: LR, RF, Extremely Randomized Tree, SVM, AutoML, NN, GBM **Cross-validation**: *Did not mention fold value* **Hyperparameter tuning**: The method is not mentioned **Evaluation Parameter:** AUC, confusion matrix	Split data into two groups: PLC4 and PLC123 **PLC123** LR: 53% RF: 60% Extra Tree Classifier: 60% SVM: 57% **GBM: 61%** NN: 56% **PLC4:** LR: 58% RF: 66% **Extra Tree Classifier: 68%** SVM: 63% GBM: 65% NN: 64%			distribution between churn and not churn The hyperparameter tuning method is not mention The evaluation parameter is not enough

(Bhattarai et al., 2019)	Mobile telecommunication system customer churn dataset (3333 x 20)	**EDA:** Identify patterns, detect outliers, visualization **Data Pre-processing:** drop useless attribute, missing value imputation, data types conversion, feature engineering **ML algorithm:** LR, NB, RF, XGBoost **Evaluation parameters:** Accuracy, Precision, Recall, F1-measure	LR: 85.7 NB: 85.1 RF: 92.3 **XGBoost: 95.5**	Random Forest and XGBoost have better performance for imbalanced class distributions of data	- Data may be analyzed in detail to determine the root causes of client turnover. - Artificial Neural Networks can be used to investigate this issue because they have demonstrated superior performance in a variety of prediction situations.	Did not discuss about the distribution between churn and not churn Need to do outlier treatment or need to explain why outlier treatment is not done Do hyperparameter tuning to built a better performance model. Need to do cross-validation to increase accuracy
(Asthana, 2018)	Customer churn dataset in telecom field obtained from the UCI Machine	**ML algorithm:** ANN, SVM-RBF, SVM-POLY, SVM-RBF with Adaboost, SVM-	SVM-RBF：94.37 SVM-POLY: 89.58 SVM-RBF with Adaboost：96.05	SVM-Poly with Adaboost is the best classifier.	Investigate other simulation approaches for the AdaBoost's low trainer settings.	Did not clearly mention the exploratory data

	Learning Repository (5000 x 20)	POLY with Adaboost, DT, NB, LR **Cross-Validation:** 100-fold **Evaluation parameters**: Precision, recall, accuracy and F-measure	**SVM-POLY with Adaboost: 96.85** DT： 94.15 NB： 86.94 LR: 87.94		Study the potential of other alternative boosting methods. Collect larger dataset to improve statistic significant	analysis and pre-processing steps Did not discuss about the balancing issue that normally happened in the churn dataset
(Ismail et al., 2019)	Telecommunication industry customer churn data obtained from Kaggle (7043 x 21)	**EDA:** Explore missing value, outlier, invalid values, data types, summary statistic and visualization method **Data pre-processing:** impute missing values, outlier, feature selection, one-hot encoding and data trasformation **ML technique:** LR, ANN, RF	**LR: 100** ANN: 85.55 RF: 96	Logistic regression obtains the best result	*NA*	Did not explore target variable data distribution There is a need of hyperparameter tuning and cross-validation to find out the best parameter More different algorithm can be used

		Evaluation parameter: accuracy, precision, recall and error rate, confusion matrix				

2.2 Summary

Table 2.2 Summary of Table 2.1

Reference	Dataset Size (Rows x col)	Best ML algorithms	EDA	Data pre-processing	Confusion matrix	Feature selection	Data Balancing	K-Fold Cross-Validation	Hyperparameter tuning	At least 3 evaluation parameters	Accuracy (%)
(Lemos et al., 2022)	500,000 x 35	Random forest	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	82.8
(Miao & Wang, 2022)	1000 x 21	Random forest	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	96.10
(Wu et al. 2021)	**Dataset 1:** 7032 x 21	**Dataset 1:** Adaboost **Dataset 2:** RF **Dataset 3**: RF	Yes	Yes	No	Yes	Yes	Yes	No	Yes	**Dataset 1**: 77.19 **Dataset 2:** 93.60 **Dataset 3:** 63.09

	Dataset 2: 4031 x 20 **Dataset 3**: 51047 x 58										
(Xiahou & Harada, 2021)	987994 x17	Support vector machine	No	Yes	No	No	Yes	Yes	No	Yes	~91
(Kaur & Kaur, 2020)	28382 x 21	Random Forest	Yes	Yes	No	Yes	Yes	Yes	No	Yes	~85.0
(Wadikar, 2020)	96967 x 48	Random Forest	Yes	Yes	Yes	Yes	Yes	No	No	Yes	97
(Bhattarai et al., 2019)	3333 x 20	XGBoost	Yes	Yes	No	Yes	No	No	No	Yes	95.5
(He et al., 2020)	25275 x 253	Extra Tree Classifier and Gradient Boost	Yes	Yes	Yes	Yes	No	Yes	Yes	No	**AUC (%)** GBM: 61 Extra Tree Classifier: 68

(Asthana, 2018)	(5000 x 20)	SVM-Poly with Adaboost	No	No	Yes	No	No	Yes	Yes	Yes	96.85
(Ismail et al., 2019)	(7043 x 21)	Logistic Regression	Yes	Yes	Yes	Yes	No	No	No	Yes	100

2.3 Comparison between related work

Based on the literature review, the different method that they are using is each step such as exploratory data analysis, data pre-processing, machine learning selection, hyperparameter tuning and evaluation matrix. The commonly used exploratory data analysis method are exploring the data types, data variable, missing value, outliers, normality, statistic summary and visualization. However, these studies do not have clear explanation or even mentioned on the issue of target variable distribution. This is an important step because there is a need to understand the data distribution in the target variable since imbalanced data distribution is the most common issue that happened in customer churn dataset. Imbalanced dataset will lower the model performance in prediction due to less minority group data that can be used to train the model.

In term of data pre-processing, the studies from literature review are mostly missing value imputation, outlier treatment, normalization and feature engineering. Only less than half authors carried out data balancing before model building, but they did not clearly mention the resampling method that was used in their research. There are only 2 studies which is Wadika, (2020) and Wu et al., (2021) did mention the method that is used for data balancing based on the literature review, the method that they are using is SMOTE method which is one of the popular data balancing methods. However, they only use one resampling method for prediction. It leaves a question that: When the machine learning that they proposed work on other resampling method, does the result will still remain high performance as the research? Hence, it addresses a need to examine the best machine learning models which can achieve high performance across different resampling method.

In term of machine learning selection, the common machine learning algorithm that was chosen are supervised machine learning, logistic regression, random forest, artificial neural network, Naïve Bayes and XGBoost. Among these machine learning algorithms, the machine learning models that was mostly claimed by studies as best performance approach is random forest. Around 60% of the literature review pointed out that random forest that build on customer churn dataset outperform other algorithms. Next, in term of hyperparameter tuning, around 80% of the studies did not carry out hyperparameter tuning after the base model building. This might cause the result less reliable because hyperparameter tuning is an important process to obtain the optimal result. After hyperparameter tuning, the other new model might outperform the previous suggested machine learning model. Hence, it addresses a need of research to perform hyperparameter tuning when determining the best machine learning models.

In term of evaluation matrix, the most common use parameter by the studies include accuracy, sensitivity, specificity, confusion matrix, F1-measure, area under curve (AUC) and receiver operating characteristic (ROC) curve. By considering all the context based on the literature review, a new study is developed in this project. This study is to determine the best machine learning approach that can perform well across different resampling methods. In exploratory data analysis part, the most commonly used methods are utilized, which include: explore the data dimension, data types, missing value, normality, summary statistic, visualization method, determine the dependent and independent variable and the most important is explore the data proportion under the target variable.

Next, the pre-processing part in this study include impute the missing value, cell category treatment, normalization and data balancing by using different methods to create different data frame for model building. The resampling methods that were used in this study include SMOTE resampling, random oversampling, random undersampling and random both sampling under the ROSE package. The determination of the chosen resampling method is based on (GreeksforGreeks, 2021) and (avcontentteam, 2016) that proposed that SMOTE and the other resampling method under ROSE are the popular method used in data resampling. Based on the previous literature review, the study that did data resampling also use SMOTE resampling method (Wadika, (2020) and Wu et al., (2021)).

In term of machine learning method, random forest which is the best performance model that determine by other studies is selected, follow by support vector machine and logistic regression which are the best machine under the study of (Xiahou & Harada, 2021) and (Ismail et al., 2019). Hyperparameter tuning and 10-fold cross validation is carried out in this study to make sure the best performance machine learning approach is optimal. Last but not least, the evaluation matrix that was used is accuracy, sensitivity, specificity, F1-measure, confusion matrix, AUC and ROC which is selected based on reference on the literature review previously.

3.0 Method

3.1 Flow Chart

The flow chart below shows the general flow of the experiment:

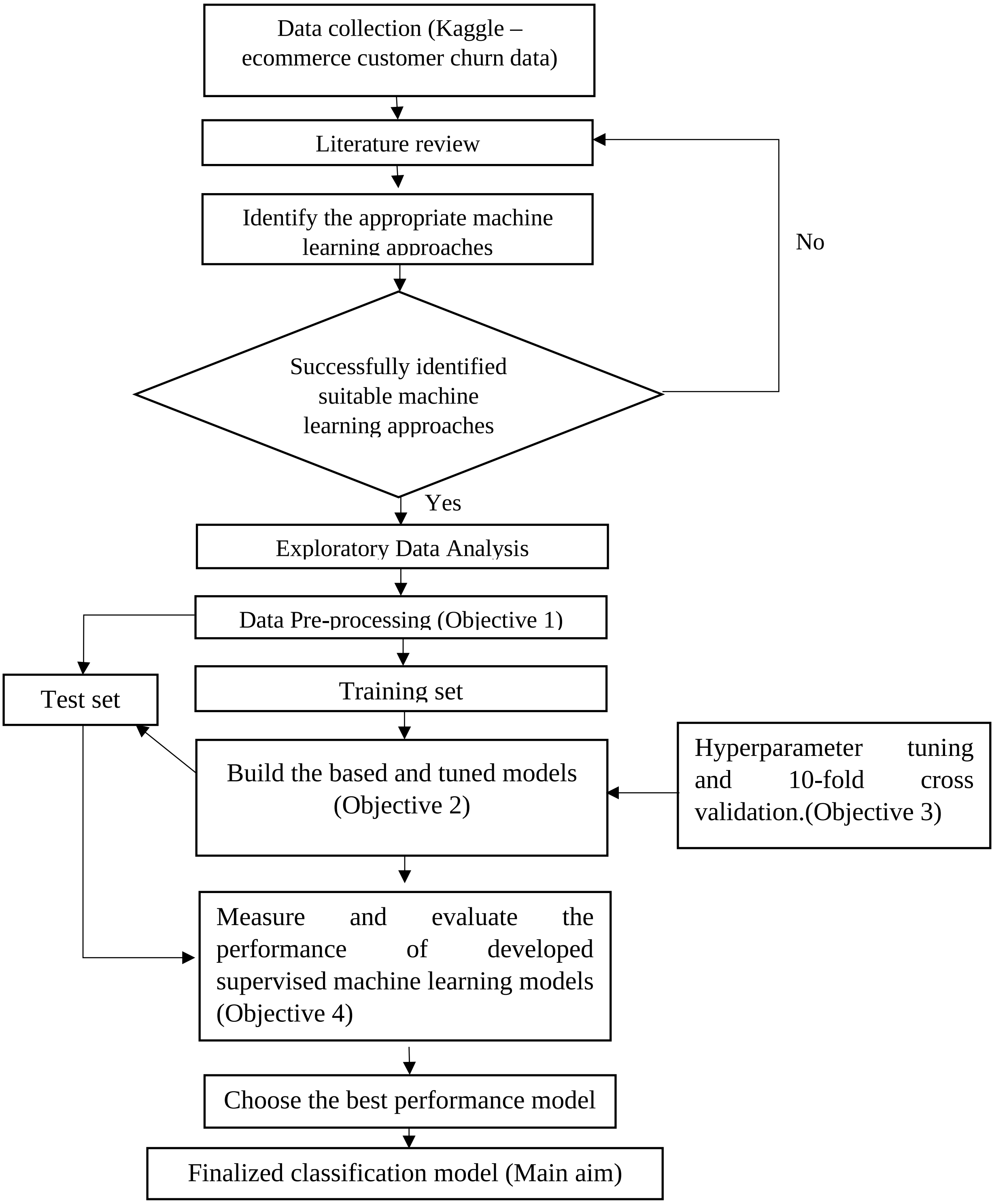


Figure 3.1 Methodology Flow Chart

3.2 Data Collection

3.2.1 Dataset Introduction

The eCommerce customer churn dataset that used in this project is obtained from Kaggle. Here is the url for the dataset: https://www.kaggle.com/datasets/ankitverma2010/ecommerce-customer-churn-analysis-and-prediction. Initially, the dataset is the property of a well-known online E-Commerce corporation that needs to determine which clients are about to churn so that they may contact them with special offers. The dataset consists of 5630 churn records, 19 predictable variables, one target variable. The 19 predictable variables included CustomerID, Tenure, PreferredLoginDevice, CityTier, WarehouseToHome, PreferredPaymentMode, Gender, HourSpendOnApp, NumberOfDevicesRegistered, PreferredOrderCat, SatisfactionScore, MaritalStatus, NumberOfAddress, Complain, OrderAmountHikeFromlastYear, CouponUsed, OrderCount, DaySinceLastOrder and CashbackAmount while the target variable is churn. The churn variable which is the target variable is a binary categorical variable that has only '1' and '0'. For '1', indicate the customer is left while '0' indicates that the customer still stays with the company service.

3.2.2 Dataset Variable Dictionary

Table 3.1: Dataset Attributes Description and Instances.

No	Variable	Discerption	Data Types	Instances
1	CustomerID	Unique customer ID	Numeric	5630 unique ID for customer
2	Churn	Churn Flag	Categorical	'0' – No '1' - Yes
3	Tenure	Tenure of the customer in the organization	Numeric	Within 0 to 61
4	PreferredLoginDevice	Preferred login device of the customer	Categorical	Computer, mobile

				phone, phone
5	CityTier	City tier	Numeric	Within 1 to 3
6	WarehouseToHome	Distance in between warehouse to the home of customer in km	Numeric	Within 5 to 127
7	PreferredPaymentMode	The preferred payment method of customer	Categorical	Credit cards, cash on delivery, debit card, e-wallet, Unified Payments Interface
8	Gender	Gender of customer	Categorical	Male, Female
9	HourSpendOnApp	Number of hours spent on mobile application or website	Numeric	Within 0 to 5
10	NumberOfDeviceRegistered	The total number of deceives is registered on a particular customer	Numeric	Within 1 to 6
11	PreferredOrderCat	Preferred order category of customer in last month	Categorical	Fashion, grocery, mobile phone, laptop & accessory
12	SatisfactionScore	Satisfactory score of customers on service	Numeric	Within 1 to 5

13	MaritalStatus	Marital status of customer	Categorical	Divorced, marriage, single
14	NumberOfAddress	Total number of added on the particular customer	Numeric	Within 1 to 22
15	Complain	Any complaint has been raised in last month	Numeric	Either 0 or 1
16	OrderAmountHikeFromlastYear	Percentage increases in order from last year	Numeric	Within 11 to 26
17	CouponUsed	Total number of coupons has been used in last month	Numeric	Within 0 to 16
18	OrderCount	Total number of orders has been places in last month	Numeric	Within 0 to 16
19	DaySinceLastOrder	Day Since last order by customer	Numeric	Within 0 to 46
20	CashbackAmount	Average cashback in last month	Numeric	Within 0 to 325

3.3 Data Preparation

3.3.1 Exploratory Data Analysis

Exploratory Data Analysis is a vital process that involves conducting preliminary analyses on information in order to identify patterns, detect abnormal using statistical results and visualisations (Patil, 2018). Based on the literature review, most of the methods that was used in the exploratory data analysis are explore the missing value, normality, summary statistic and visualization. In this study, the methods used to explore the dataset include: Explore the data dimension, data type, data variables, skewness, missing value, summary statistic, variables categories, category distribution under target variables and correlation between variables by visualization technique. In the section of exploratory data analysis, two main libraries are used which is the dplyr library and DataExplorer library. The main function of the dplyr library is to explore the information of the data frame such as the number of variable and observation, the data types of the variables, the available variables and statistic summary such as mean, median, and quartile of the numeric data. The function that used in the dplyr libraries such as, mutate() function and etc. While the DataExplorer library is mainly used for visualization of the data frame information such as visualize the frequency of gender variable in bar graph form, visualize the skewness of continuous variable through histogram, and visualize the relationship between variable. The function that used under the DataExplorer library such as plot_str(), plot_histogram(), plot_bar() and etc.

3.3.2. Data Pre-processing

Pre-processing data is mostly used to ensure the data's quality and is the process of converting original data to a usable format. Based on the literature review, the most common use pre-processing technique are imputing missing value, outlier treatment, data transformation, and features engineering. In this study, the data pre-processing involves missing value imputation, data normalization, cell category treatment, data types conversion, feature engineering and data resampling and stratified sampling. The data resampling method included: Synthetic Minority Over-sampling Technique (SMOTE), random oversampling, random under sampling, and random both sampling. The train-test split ratio is 70:30 which mean 70% training set and 30% test set. The package that is used in this stage include package of smotefamily used for SMOTE resampling, ROSE package which is used for random resampling, stringr package used to replace the words in the cell in order for cell category correction, fastDummies package used for one-hot encoding and rsample package for stratified sampling.

3.3 Machine learning technique

In this section, the programming language that used for model building is R programming. Based on the literature review, three machine learning algorithms were chosen which include support vector machine, random forest and logistic regression. The detail of each machine learning approach is shown below:

3.3.1 Logistic Regression

Logistic regression is a technique used to categorize the relationships between target variables with other features. Although it was traditionally utilised in the health sector, it is a sophisticated regression technique that has acquired appeal in the cultural studies in recent years. Due to the inefficiency of the Least Squares Method (LSM) in multivariate models with dependant variables distinction, logistic regression is utilised as a substitute for this technique. The chance of the target features having two outcomes is forecasted using logistic regression model. As a result of this property, it is a popular technique for observations classification (Osmanoglu, 2019). In this section, the glm function is used to build the logistic regression model. The conditional probability of the event Y given the variable X is where Y is a linear function of the independent input variables. Below shows the formula for the logistic regression:

$$\boldsymbol{P\left(\frac{Y}{X}\right) = \frac{1}{1 + e^{Y}}}$$

3.3.2 Support Vector Machine

SVM is a machine learning technique that does forecasting and generalisation on information by conducting training on big information. The SVM's fundamental idea is predicated on the existence of a hyperplane that effectively separates 2 categories of data. The support vector machine is classified into two types based on the data set's linear and nonlinear classification (Osmanoglu, 2019). The support vector machine is built by the package of e1071.

3.3.3 Random Forest

Random forests are an ensemble training technique for categorization, regression, and other types of work. It works by learning a large number of decision trees and then outputs the category that is high probability of the categories of the different trees (Miao & Wang, 2022). Based on the literature review, most of the studies claim that random forest has the highest performance against other models. Hence, random forest is selected in this study. The random forest is built by using the package of randomForest in R.

3.4 Hyperparameter tuning

Modern supervised machine learning algorithms require the setting of hyperparameters prior to execution. The choices for configuring hyperparameters include using the initial values provided by the package, manually defining them, or adjusting them for maximum prediction accuracy. In comparison to the explicit, 1st model parameters that are established during learning, these 2nd tuning parameters frequently require meticulous optimization to attain optimal results (Probst et al., 2019). In this section, method that used for hyperparameter tuning is random search. Only support vector machine and random forest carry out hyperparameter tuning. 10-fold cross validation is utilized in this tuning process to improve the output accuracy. The package used in hyperparameter tuning is caret package. After the hyperparameter tuning, a new model is built by using the tuned parameter.

3.5 Evaluation Metrix

In this section the evaluation matrix that used include the confusion matrix, accuracy, sensitivity, specificity, F1 measure, area under curve (AUC) and receiver operating characteristic (ROC) curve (Wu wt al. 2021).

I. Accuracy

It is the fraction of correct predictions to the total amount of forecasts and is calculated using the equation:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Here TP, TN, FP and FN are the True Positive, True Negative, False Positive and False Negative respectively.

II. F-measure

F-measure as the harmonics average of their precision and recall and is calculated using the equation:

$$F - measure = \frac{2\ x\ Precision\ x\ Recall}{Precision + Recall}$$

III. ROC

ROC is a plot based on the calculation of False Positive Rate (FPR) and True Positive Rate (TPR) of the classifier:

$$FPR = \frac{FP}{FP + TN}$$

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

While the AUC is the area under the curve of ROC.

IV. Sensitivity

Sensitivity is the true positive divided by the total between true positive and false negative.

$$Sensitivity = \frac{TP}{TP + FN}$$

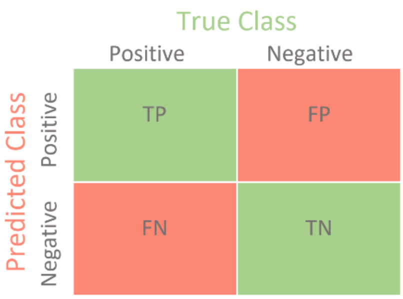
V. Specificity

Specificity is the true negative divided by the total between true negative and false positive.

$$Specificity = \frac{TN}{TN + FP}$$

VI. Confusion Matrix

The Confusion Matrix is a unique table arrangement that enables the visualisation of an algorithm's performances.



4.0 Data Preparation

4.1 Exploratory data analysis

1. View the dataset

```
View(df)
```

Output:

	CustomerID	Churn	Tenure	PreferredLoginDevice	CityTier	WarehouseToHome	PreferredPaymentMode	Gender	HourSpendO
1	50001	1	4	Mobile Phone	3	6	Debit Card	Female	
2	50002	1	NA	Phone	1	8	UPI	Male	
3	50003	1	NA	Phone	1	30	Debit Card	Male	
4	50004	1	0	Phone	3	15	Debit Card	Male	
5	50005	1	0	Phone	1	12	CC	Male	
6	50006	1	0	Computer	1	22	Debit Card	Female	
7	50007	1	NA	Phone	3	11	Cash on Delivery	Male	
8	50008	1	NA	Phone	1	6	CC	Male	
9	50009	1	13	Phone	3	9	E wallet	Male	
10	50010	1	NA	Phone	1	31	Debit Card	Male	
11	50011	1	4	Mobile Phone	1	18	Cash on Delivery	Female	
12	50012	1	11	Mobile Phone	1	6	Debit Card	Male	
13	50013	1	0	Phone	1	11	COD	Male	
14	50014	1	0	Phone	1	15	CC	Male	
15	50015	1	9	Mobile Phone	3	15	Credit Card	Male	
16	50016	1	NA	Phone	2	12	UPI	Male	
17	50017	1	0	Computer	1	12	Debit Card	Female	
18	50018	1	0	Mobile Phone	3	11	E wallet	Male	
19	50019	1	0	Computer	1	13	Debit Card	Male	
20	50020	1	19	Mobile Phone	1	20	Debit Card	Female	
21	50021	1	0	Mobile Phone	3	12	Debit Card	Male	

After import the dataset and name the dataset as "df". By using the code above, the dataset can be view in table form under a new window of R studio. The purpose of this section is to have an initial view of the dataframe in the R studio.

2. Dimension of the dataset

```
dim(df)
```

Output:

```
> dim(df)
[1] 5630   20
> 
```

Next, explore the dimension of the dataset by using the dim() function. Based on the output, it shown that the dataset has 20 variables and 5630 observations.

3. Explore the variable name

```
name(df)
```

Output:

```
> names(df)
 [1] "CustomerID"                  "Churn"                       "Tenure"
 [4] "PreferredLoginDevice"        "CityTier"                    "WarehouseToHome"
 [7] "PreferredPaymentMode"        "Gender"                      "HourSpendOnApp"
[10] "NumberOfDeviceRegistered"    "PreferedOrderCat"            "SatisfactionScore"
[13] "MaritalStatus"               "NumberOfAddress"             "Complain"
[16] "OrderAmountHikeFromlastYear" "CouponUsed"                  "OrderCount"
[19] "DaySinceLastOrder"           "CashbackAmount"
>
```

Next, explore the 20 variables name by using the name(df) function, the purpose of this section is to have an overview in the dataset and understand the customer profile. In this section, the independent variable and dependent variable also can be determined. This project is used to predict the customer churn risk in the e-commerce, hence we can conclude that the dependent variable is the "Churn" variable while the rest variables are independent variables.

4. Explore the data types

Output:

```
glimpse(df)
```

```
Columns: 20
$ CustomerID                  <int> 50001, 50002, 50003, 50004, 50005, 50006, 50007, 50008, 50009, 5001~
$ Churn                       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ Tenure                      <int> 4, NA, NA, 0, 0, 0, NA, NA, 13, NA, 4, 11, 0, 0, 9, NA, 0, 0, 0, 19~
$ PreferredLoginDevice        <chr> "Mobile Phone", "Phone", "Phone", "Phone", "Phone", "Computer", "Ph~
$ CityTier                    <int> 3, 1, 1, 3, 1, 1, 3, 1, 3, 1, 1, 1, 1, 1, 3, 2, 1, 3, 1, 1, 3, 1, 3~
$ WarehouseToHome             <int> 6, 8, 30, 15, 12, 22, 11, 6, 9, 31, 18, 6, 11, 15, 15, 12, 12, 11, ~
$ PreferredPaymentMode        <chr> "Debit Card", "UPI", "Debit Card", "Debit Card", "CC", "Debit Card"~
$ Gender                      <chr> "Female", "Male", "Male", "Male", "Male", "Female", "Male", "Male",~
$ HourSpendOnApp              <int> 3, 3, 2, 2, NA, 3, 2, 3, NA, 2, 2, 3, 2, 3, 3, 3, NA, 2, 3, 3, 3, 3~
$ NumberOfDeviceRegistered    <int> 3, 4, 4, 4, 3, 5, 3, 3, 4, 5, 3, 4, 3, 4, 4, 3, 4, 4, 5, 3, 5, 3, 3~
$ PreferedOrderCat            <chr> "Laptop & Accessory", "Mobile", "Mobile", "Laptop & Accessory", "Mo~
$ SatisfactionScore           <int> 2, 3, 3, 5, 5, 5, 2, 2, 3, 3, 3, 3, 3, 3, 2, 5, 2, 3, 3, 4, 3, 2, 3~
$ MaritalStatus               <chr> "Single", "Single", "Single", "Single", "Single", "Single", "Divorc~
$ NumberOfAddress             <int> 9, 7, 6, 8, 3, 2, 4, 3, 2, 2, 2, 10, 2, 1, 2, 5, 2, 2, 2, 10, 5, 2,~
$ Complain                    <int> 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1~
$ OrderAmountHikeFromlastYear <int> 11, 15, 14, 23, 11, 22, 14, 16, 14, 12, NA, 13, 13, 17, 16, 22, 18,~
$ CouponUsed                  <int> 1, 0, 0, 0, 1, 4, 0, 2, 0, 1, 9, 0, 2, 0, 0, 1, 1, 1, 1, 1, 6, 11, ~
$ OrderCount                  <int> 1, 1, 1, 1, 1, 6, 1, 2, 1, 1, 15, 1, 2, 1, 4, 1, 1, 1, 1, 4, 7, 15,~
$ DaySinceLastOrder           <int> 5, 0, 3, 3, 3, 7, 0, 0, 2, 1, 8, 0, 2, 0, 7, 2, 0, 3, 6, 3, 7, 6, 0~
$ CashbackAmount              <int> 160, 121, 120, 134, 130, 139, 121, 123, 127, 123, 295, 154, 134, 13~
>
```

Next, explore the data types of the variable. In this section, we can understand the data type and correct it if there is wrong data type or encode the categorical variables into numeric in the following pre-processing part. In this section, we can see that the churn variable is in numeric form, however the prediction variable should be in category form. On the other hand, the character data types are suggested to change into factor data types because this data types can be used by different algorithms and able to aid in prevent error. Hence, this point is marked down to make sure the later pre-processing part will carry out data type conversion.

5. Explore the summary statistic

```
summary(df)
```

Output:

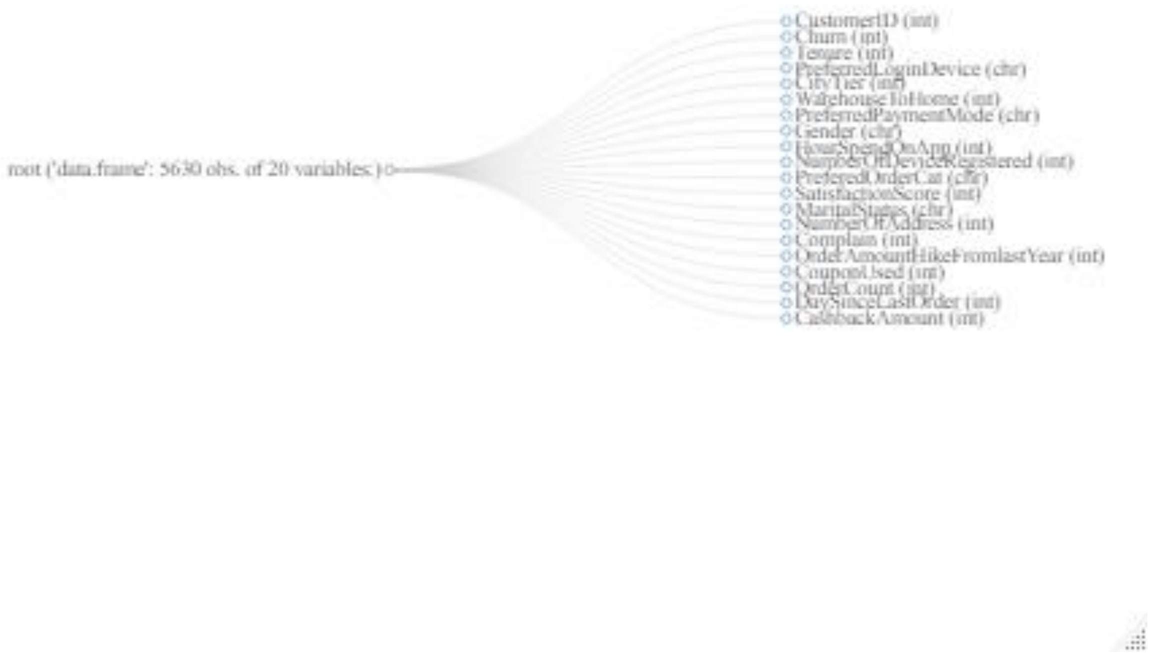
```
> summary(df)
   CustomerID        Churn             Tenure       PreferredLoginDevice    CityTier      WarehouseToHome
 Min.   :50001   Min.   :0.0000   Min.   : 0.00   Length:5630          Min.   :1.000   Min.   :  5.00
 1st Qu.:51408   1st Qu.:0.0000   1st Qu.: 2.00   Class :character     1st Qu.:1.000   1st Qu.:  9.00
 Median :52816   Median :0.0000   Median : 9.00   Mode  :character     Median :1.000   Median : 14.00
 Mean   :52816   Mean   :0.1684   Mean   :10.19                        Mean   :1.655   Mean   : 15.64
 3rd Qu.:54223   3rd Qu.:0.0000   3rd Qu.:16.00                        3rd Qu.:3.000   3rd Qu.: 20.00
 Max.   :55630   Max.   :1.0000   Max.   :61.00                        Max.   :3.000   Max.   :127.00
                                  NA's   :264                                          NA's   :251
 PreferredPaymentMode    Gender           HourSpendOnApp  NumberOfDeviceRegistered PreferedOrderCat
 Length:5630          Length:5630        Min.   :0.000   Min.   :1.000            Length:5630
 Class :character     Class :character   1st Qu.:2.000   1st Qu.:3.000            Class :character
 Mode  :character     Mode  :character   Median :3.000   Median :4.000            Mode  :character
                                         Mean   :2.932   Mean   :3.689
                                         3rd Qu.:3.000   3rd Qu.:4.000
                                         Max.   :5.000   Max.   :6.000
                                         NA's   :255
 SatisfactionScore MaritalStatus      NumberOfAddress     Complain      OrderAmountHikeFromlastYear
 Min.   :1.000     Length:5630        Min.   : 1.000   Min.   :0.0000   Min.   :11.00
 1st Qu.:2.000     Class :character   1st Qu.: 2.000   1st Qu.:0.0000   1st Qu.:13.00
 Median :3.000     Mode  :character   Median : 3.000   Median :0.0000   Median :15.00
 Mean   :3.067                        Mean   : 4.214   Mean   :0.2849   Mean   :15.71
 3rd Qu.:4.000                        3rd Qu.: 6.000   3rd Qu.:1.0000   3rd Qu.:18.00
 Max.   :5.000                        Max.   :22.000   Max.   :1.0000   Max.   :26.00
                                                                        NA's   :265
   CouponUsed       OrderCount      DaySinceLastOrder CashbackAmount
 Min.   : 0.000   Min.   : 1.000   Min.   : 0.000    Min.   :  0.0
 1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 2.000    1st Qu.:146.0
 Median : 1.000   Median : 2.000   Median : 3.000    Median :163.0
 Mean   : 1.751   Mean   : 3.008   Mean   : 4.543    Mean   :177.2
 3rd Qu.: 2.000   3rd Qu.: 3.000   3rd Qu.: 7.000    3rd Qu.:196.0
 Max.   :16.000   Max.   :16.000   Max.   :46.000    Max.   :325.0
 NA's   :256      NA's   :258      NA's   :307
>
```

Next, the summary statistic of the data frame can be view in this section. Based on this section, the minimum, maximum, mean, median, first quartile, third quartile of continuous variable and number of missing values can be seen.

6. Visualize data frame information

```
plot_str(df, fontSize=20)
```

Output:



Next, the data information also can be visualized in one plot. Based on the output, the dimension of the data frame, the data variable names and also the data types can be view. It is more likely a form of summary from the initial data exploration that had done previously but in the form of plot.

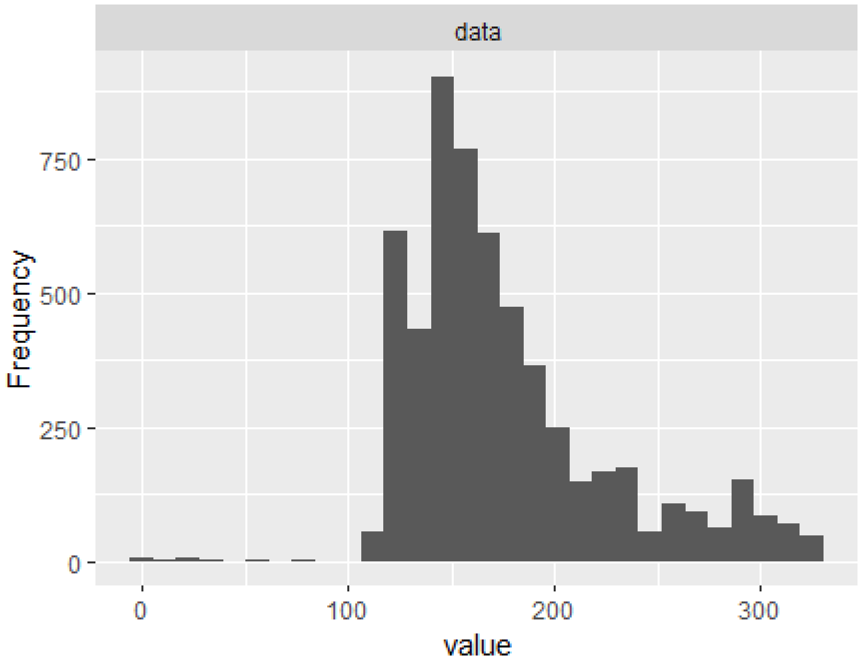
7. Skewness exploration

i) CashbackAmount

```
plot_histogram(df$CashbackAmount)
```

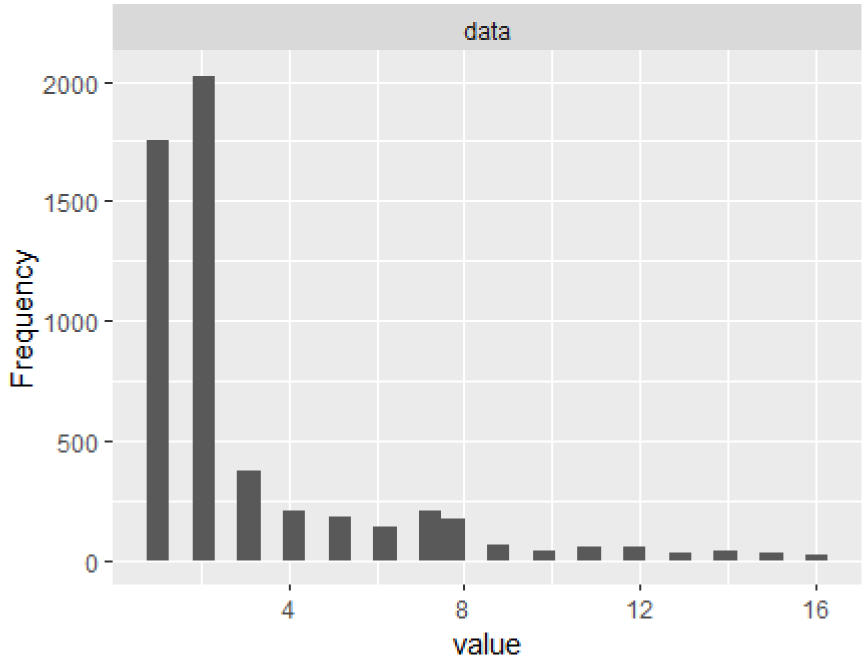
The code snippet above is used for view the histogram of the continuous variable in the dataframe. In stead of view the histogram in one view, viewing one by one is more preferable because the visualization can be clearer. In the code above the variable name is inserted such as if "CashbackAmount" is going to be viewed, then put the variable name in the code, the following section is keeping change the variable name by using the code above.

Output:



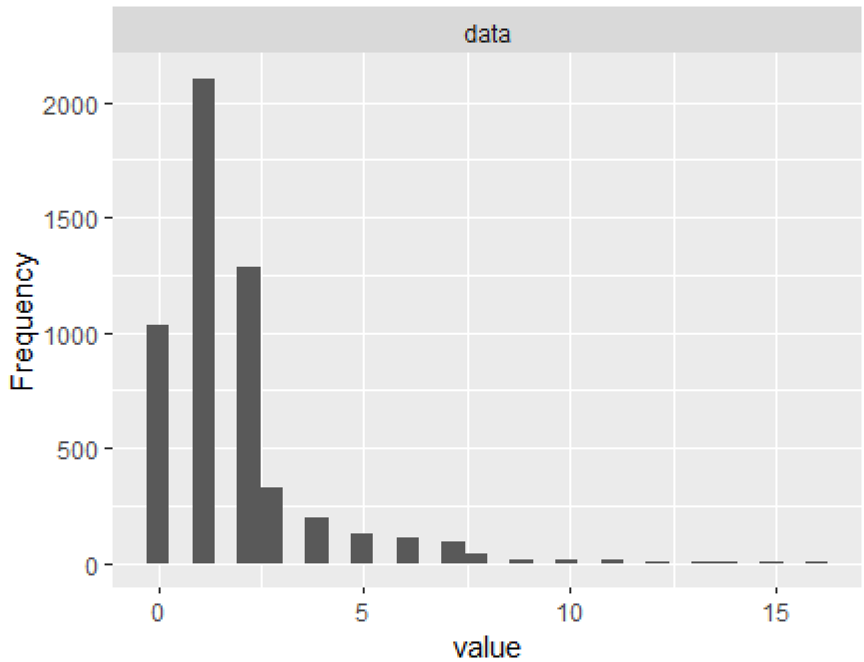
Based on the histogram, it is normal skewness, most of the customers can get their cashback amount at the value of 150. However, there is some imbalance between the edges because it shown there is more customer obtained more than 200 rather than cashback amount less than 100. But this is logical because the cashback amount is depended on the customer behavior and also the items price that they buy. So, it is reasonable to have big different.

ii) OrderCount



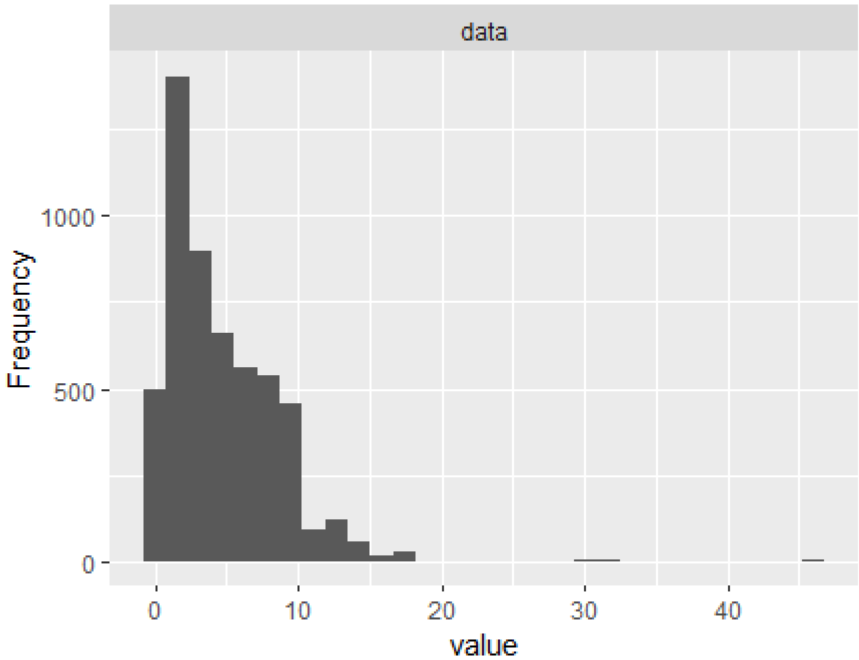
Based on the histogram, it is showing the order count variable is right skewness. Most of the order count frequency are less than 4. This variable is the number of order that the customer done in 1 month. This is highly depends on customer behaviour, so it is reasonable.

iii) CouponUsed



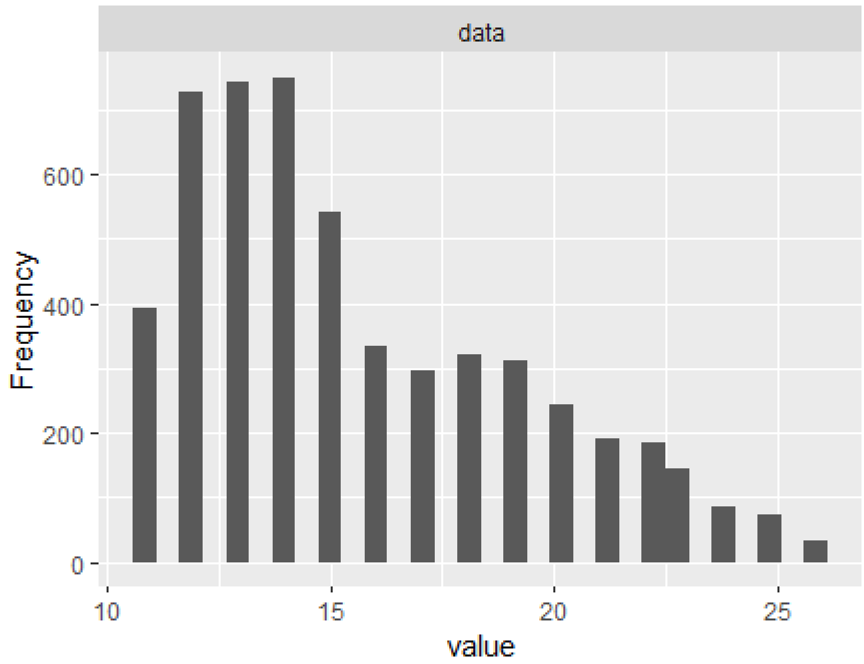
Based on the histogram, it is showing the coupon usage of the e-commerce customer is right skewness. Most of the order coupon usage are less than 4. This is reasonable because low coupon usage in e-commerce has many factors such as need to reach minimum spend, late redemption, limited coupon that can be claimed, some online shop in the e-commerce company do not support certain coupon. Hence, the right skewness is reasonable.

iv) DaySinceLastOrder



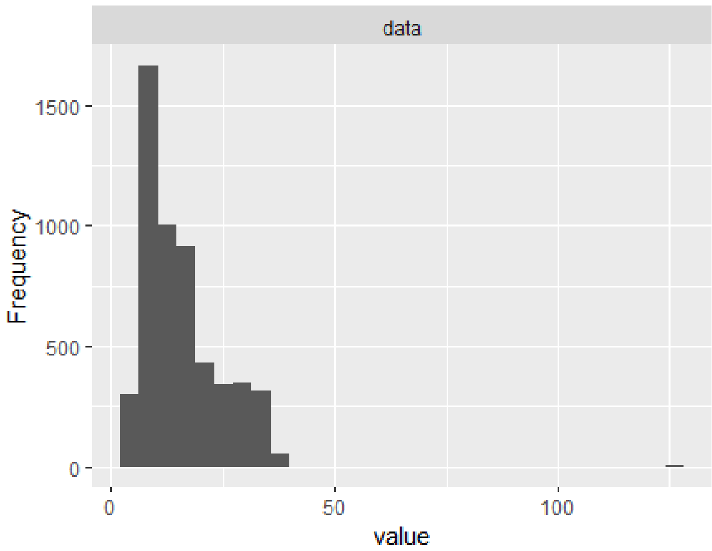
Based on the histogram, it is showing the number of days since the customer order items. The histogram shows the days number is right skewness which indicate that most of the customer are active customers but there is also customer has more than 1 months did not make any order, this value is reasonable, because the customer might tend to churn with long days since last order.

v) OrderAmountHikeFromLastYear



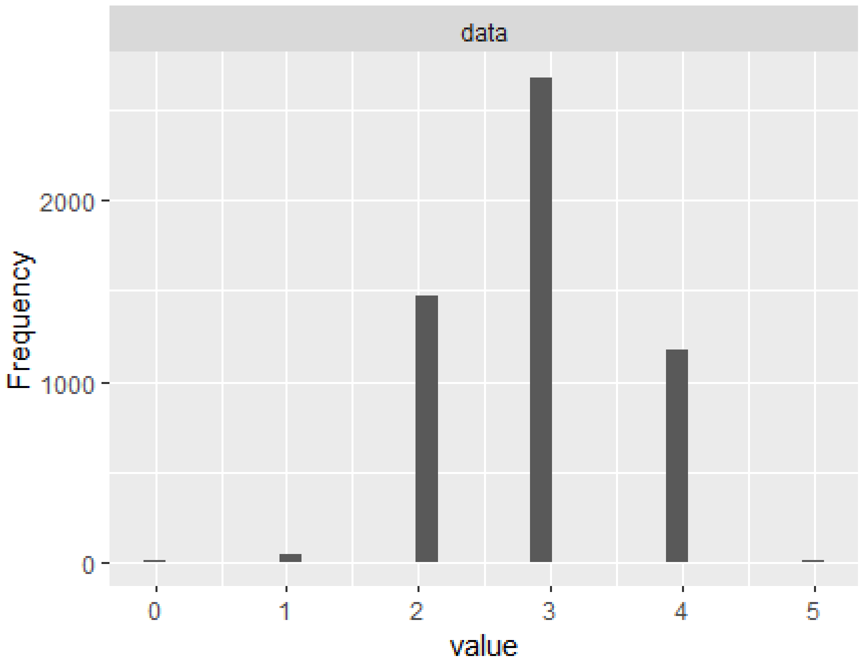
Based on the histogram, it is showing the order amount that hike from last year. The histogram shows the days number is right skewness which indicate that most of the customer are have higher order amount around value of 10 to 15. This is reasonable because this is highly depending of many factors such as promotions, the customer needs and behaviours.

vi) WareHouseToHome



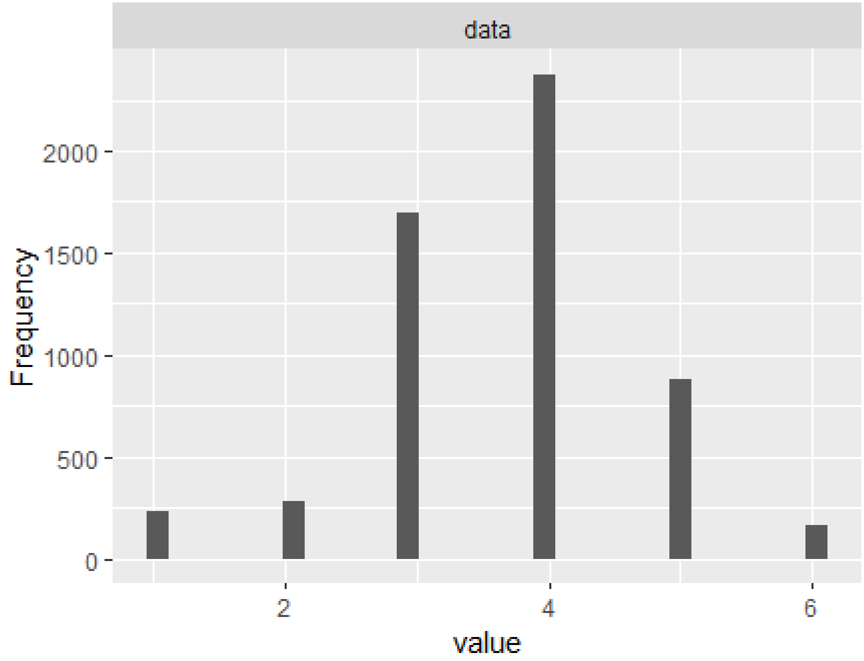
Based on the histogram, it is showing the warehouse to home distance in km of the e-commerce customers. The histogram shows the distances is right skewness which indicate that most of the customer are have less than around 10 to 20 km to the nearest warehouse. variable such as the warehouse to home can have big different depend on the customer address, if the customer is living in village, it is possible for them to reach warehouse in a long distance while customer living is city can be more easily to access to the nearest warehouse.

vii) HourSpendOnApp



Based on the histogram, it is showing the number of hours customer spend on the app. The value is less because the unit is hour instead of minutes, it is showing a normal distribution in this variable as most of the customer spend around 3 hours shopping on the e-commerce site.

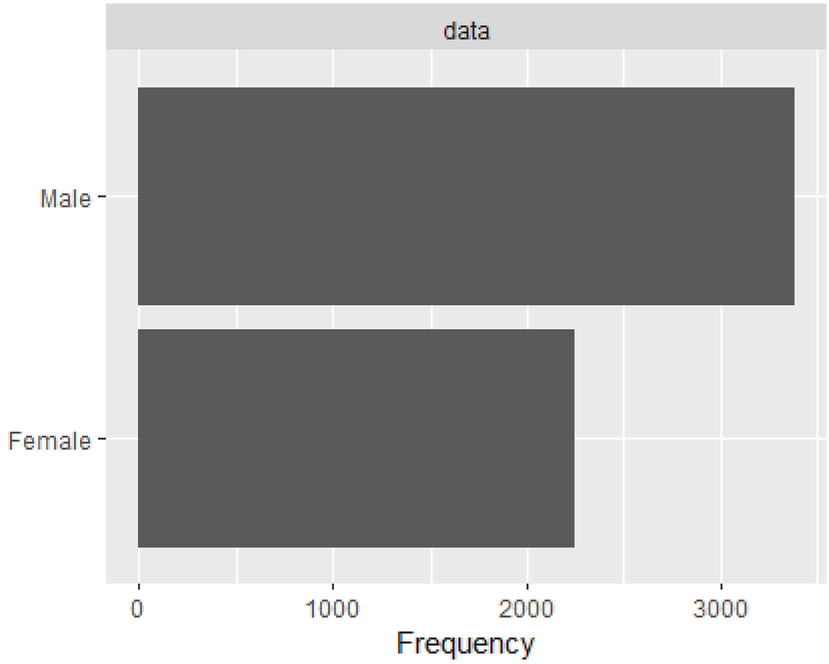
viii) NumberOfDeviceRegistered



Next based on the houtput, it is the number of device registered by customers. The amount this count from 1 to 6, the distribution is normal. Most of the customer have 4 number of device registered. The registration account may be can share with family especially parents who do not have emails. Hence it cause more device registered, it is also mayb due to the customer change their device but still tend to registered with the e-commerce company. In overall, all the variable has logical ranges. Hence, upon explore on the variables of the dataset, it can be concluded that the data variables value is acceptable, in case they have outliers, the outliers can be kept to prevent biased result. So, box plot is not carried out in this session since the outliers are decided to remain.

8. Categorical frequency

i) Gender

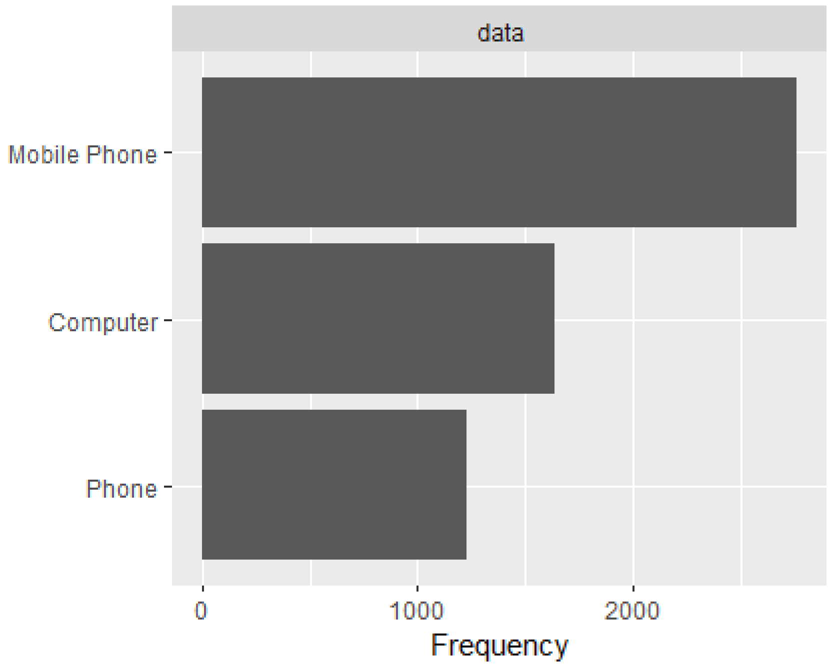



```
   Female      Male 
0.3989343 0.6010657
```

Next, the number and proportion of gender is explored by using bar chart and histogram, based on the output there are higher percentage and frequency of male (60%) than female (30%).

ii) Preferred Login Device

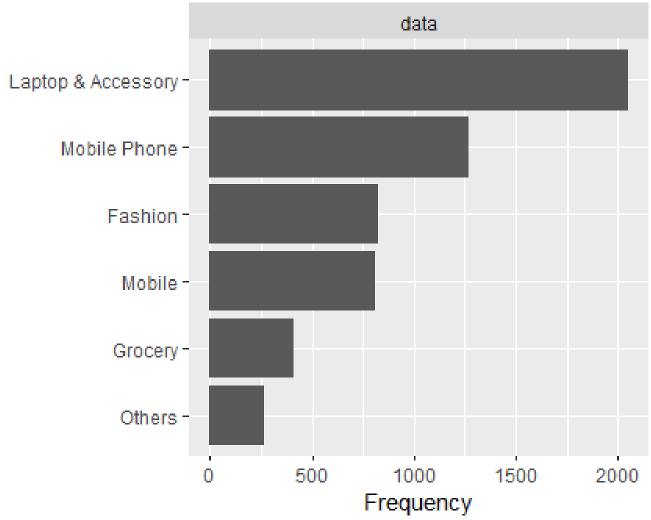
```
 Computer Mobile Phone        Phone
0.2902309    0.4911190    0.2186501
```



Next, the preferred login device of the customers is reviewed. There is total 3 categories in this section which included mobile phone, computer and phone. However, mobile phone and phone are divided into different category may be due to some customer tend to fill in a shorter name when submit their survey. Hence, this problem needs to be marked down so the later pre-processing part can adjust the problem by grouping them into one category.

iii) Preferred order category

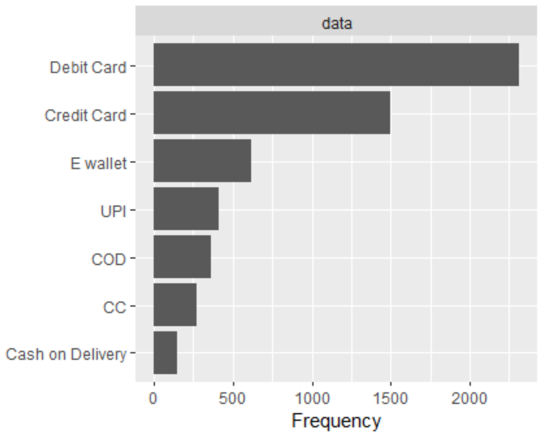
```
          Fashion             Grocery Laptop & Accessory             Mobile       Mobile Phone
       0.14671403         0.07282416         0.36412078         0.14369449         0.22575488
           Others
       0.04689165
>
```



Preferred order category has total 6 categories. However, in this variable, it has the same problem as the preferred login device. The mobile category is actually same as mobile phone. This is may be due to customer behaviour, hence this point is marked down to be adjusted in the following session.

iv) Preferred Payment Mode

```
> prop.table(table(df$PreferredPaymentMode))

Cash on Delivery               CC              COD      Credit Card       Debit Card         E wallet
      0.02646536       0.04849023       0.06483126       0.26660746       0.41101243       0.10905861
             UPI
      0.07353464
```



Preferred payment mode has total 7 categories. However, in this variable, it has the same problem as the preferred login device and preferred order category. The “CC” category is actually same as “Credit Card” while the “COD” category is same as the “Cash on Delivery”. This is may be due to customer behaviour which tend to write short form instead of full term, hence this point is marked down to be adjusted in the following session.

v) City tier

```
         1          2          3
0.65115453 0.04298401 0.30586146
>
```

Since the city tier is based on level, so it is an ordinal variable. Hence, to visualize it, a table form is choosen. Based on the output, it shown that most of the customers are come from city tier 1 (65%) followed by city tier 3 (31%) and city tier 2 (4%).

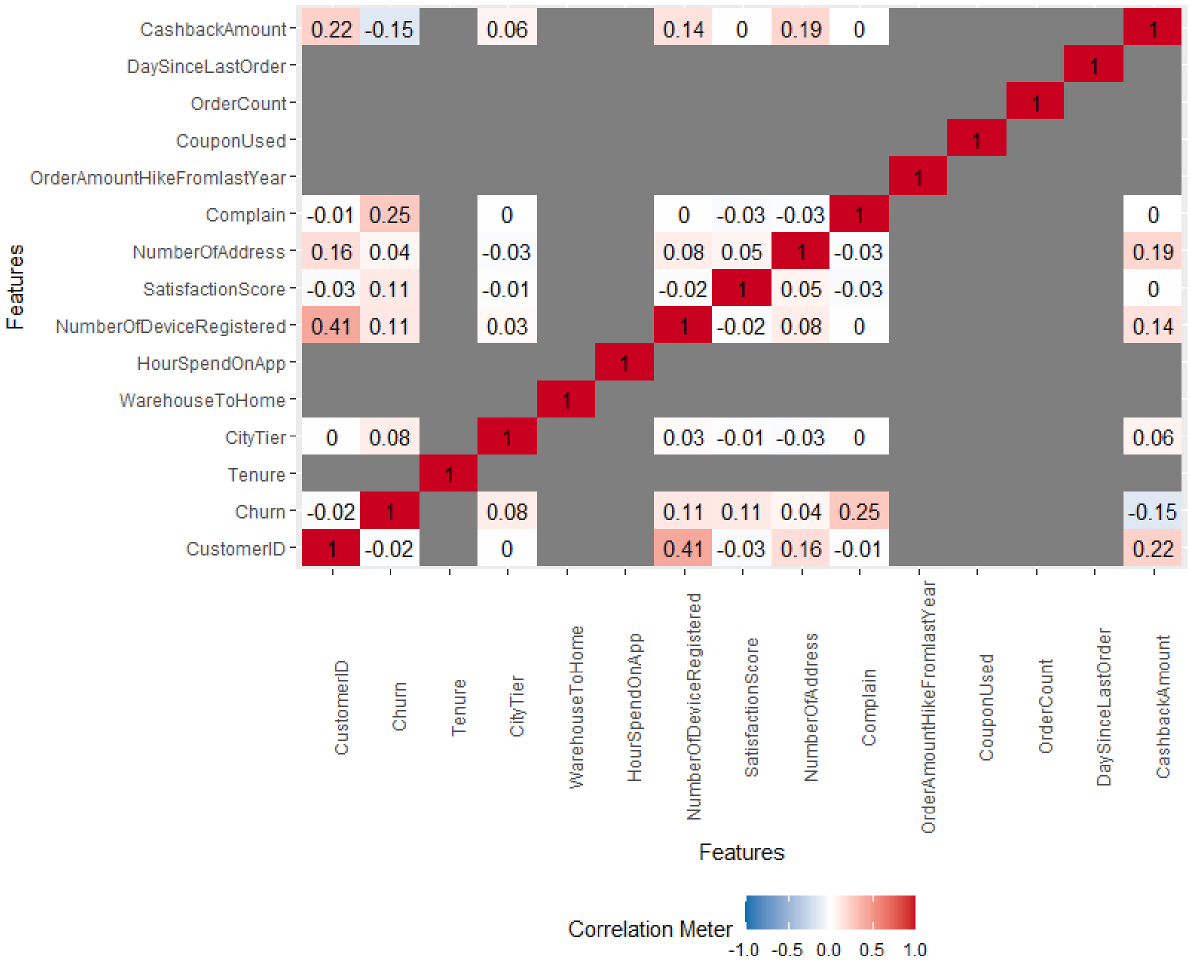
vi) Satisfaction Score

```
        1         2         3         4         5
0.2067496 0.1040853 0.3015986 0.1907638 0.1968028
>
```

Since the city tier is based on level, so it is an ordinal variable. Hence, to visualize it, a table form is choosen. Based on the output, it shown that most of the customer has satisfaction score around value of 3 (30%), followed by value of 1 (21%), value of 5 (20%), value of 4 (19%) and value of 2 (10%).

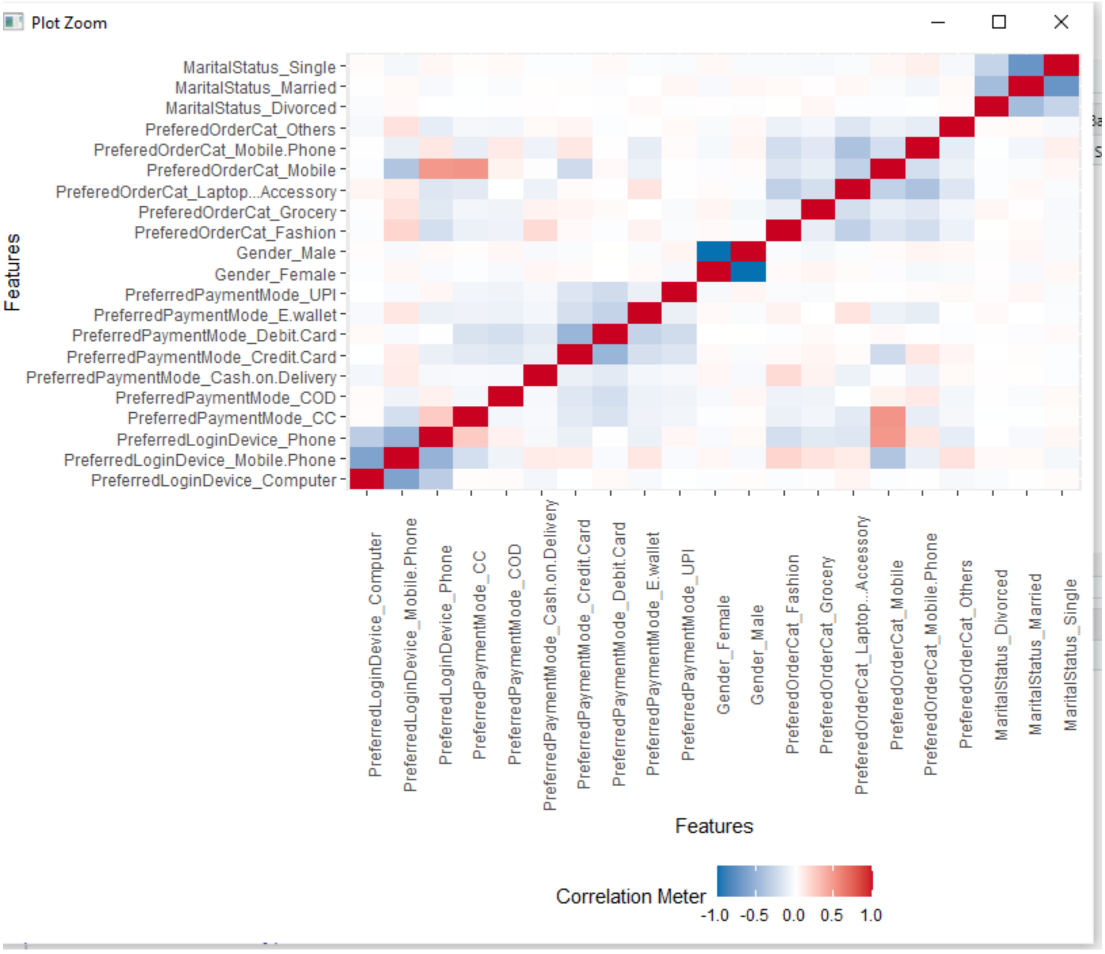
9. Correlation between variable

```
plot_correlation(df, type=c('continuous'))
```



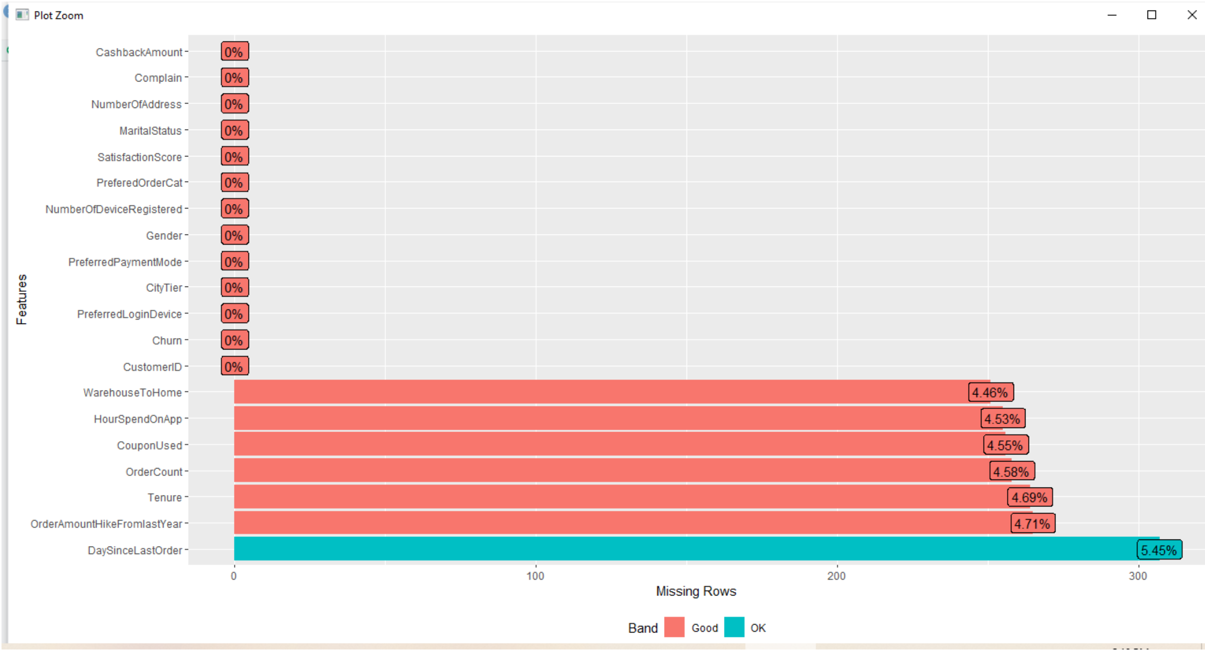
Next, the code above is used to review the correlation between continuous variable, based on the diagram above, it is showing that there is no significant relationship between variable as they have no value more than 0.5 or less than -0.5. Hence, no feature selection in continuous variable is done in this context.

```
plot_correlation(df, type=c('continuous'))
```



Next, the output above is relationship between each category in the categorical variables. Based on the above diagram, there is no much relationship between variable because the box which showing higher relationship are category in each variable. Hence, there is no feature selection in the categorical variable.

10. Missing value exploration




```
> colSums(sapply(df,is.na)) # missing values by columns
                 CustomerID                       Churn                      Tenure
                          0                           0                         264
      PreferredLoginDevice                    CityTier              WarehouseToHome
                          0                           0                         251
       PreferredPaymentMode                      Gender                HourSpendOnApp
                          0                           0                         255
   NumberOfDeviceRegistered             PreferedOrderCat            SatisfactionScore
                          0                           0                           0
              MaritalStatus              NumberOfAddress                     Complain
                          0                           0                           0
OrderAmountHikeFromlastYear                   CouponUsed                   OrderCount
                        265                         256                         258
            DaySinceLastOrder              CashbackAmount
                        307                           0
```

Next, the missing value of the dataset is reviewed by using plot and column. Based on the output there are around 5% missing value in the variable of warehouse to home, hours spend on app, coupon list, order count, tenure, order amount hike from last year and the day since last order variables. The total number of missing values are also shown in the table above. Hence, it is noted down the missing value imputation is needed in the following section.

11. Data distribution exploration

```
   0    1
4682  948
> prop.table(table(df$Churn))

        0         1
0.8316163 0.1683837
>
```

Next is checking the data distribution in the target variable. Based on the result, it is shown that the dataset has imbalanced data distribution. Hence, data balancing need to be carry out later on.

4.2 Data pre-processing

Next, is the data pre-processing part, based on the exploratory data analysis, there are few part need to do data pre-processing such as impute missing value, feature engineering and cell category treatment. First, the customerID is dropped because this is unique variable that does not help in model building. Then the missing or blank column is converted to NA by using matate_all() function. Next is the missing value imputation, the code snippet is showed in the following session:

.1. Data types conversion

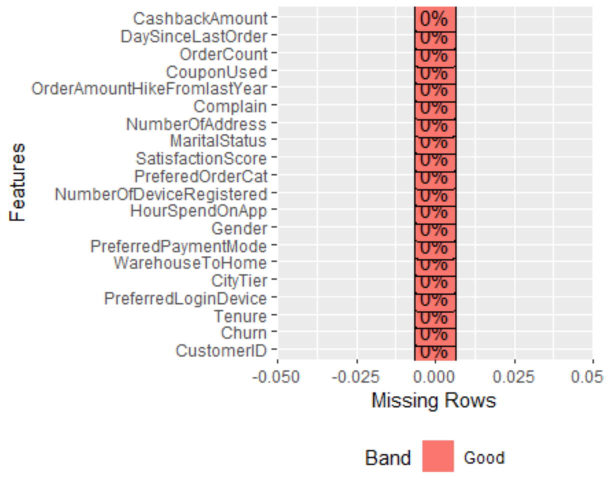
```
df$PreferredLoginDevice <- as.factor(df$PreferredLoginDevice)
df$PreferredPaymentMode <- as.factor(df$PreferredPaymentMode)
df$Gender <- as.factor(df$Gender)
df$PreferedOrderCat <- as.factor(df$PreferedOrderCat)
df$MaritalStatus <- as.factor(df$MaritalStatus)
df$Churn <- as.factor(df$Churn)
str(df)
```

.

Convert the character data types and dependent variable into factor.

2. Missing value imputation

```
### Imputation- Method 1
df$CouponUsed = ifelse(is.na(df$CouponUsed),
                       ave(df$CouponUsed, FUN = function(x) mean(x, na.rm = TRUE)),
                       df$CouponUsed)

df$OrderAmountHikeFromlastYear = ifelse(is.na(df$OrderAmountHikeFromlastYear),
                                        ave(df$OrderAmountHikeFromlastYear, FUN = function(x) mean(x,
                                        df$OrderAmountHikeFromlastYear)


df$OrderCount = ifelse(is.na(df$OrderCount),
                       ave(df$OrderCount, FUN = function(x) mean(x, na.rm = TRUE)),
                       df$OrderCount)

df$WarehouseToHome = ifelse(is.na(df$WarehouseToHome),
                            ave(df$WarehouseToHome, FUN = function(x) mean(x, na.rm = TRUE)),
                            df$WarehouseToHome)

df$Tenure = ifelse(is.na(df$Tenure),
                   ave(df$Tenure, FUN = function(x) mean(x, na.rm = TRUE)),
                   df$Tenure)

df$HourSpendOnApp = ifelse(is.na(df$HourSpendOnApp),
                           ave(df$HourSpendOnApp, FUN = function(x) mean(x, na.rm = TRUE)),
                           df$HourSpendOnApp)

df$DaySinceLastOrder = ifelse(is.na(df$DaySinceLastOrder),
                              ave(df$DaySinceLastOrder, FUN = function(x) mean(x, na.rm = TRUE)),
                              df$DaySinceLastOrder)
```



Above show the process of missing value treatment, based on above diagram, the missing value is imputed by using mean value. After missing value imputation was done, the missing value plot was checked. Upon checking, it showed that there is no more missing value in the dataset.

3. Correction for cells

```
# Correction
library(stringr)
df$PreferredLoginDevice <- print(str_replace_all(df$PreferredLoginDevice,"Mobile ", ""))


df$PreferredPaymentMode <- print(str_replace_all(df$PreferredPaymentMode,"CC", "Credit Card"))
df$PreferredPaymentMode <- print(str_replace_all(df$PreferredPaymentMode,"COD", "Cash on Delivery"))


df$PreferedOrderCat <- print(str_replace_all(df$PreferedOrderCat,"Mobile", "Mobile Phone"))
df$PreferedOrderCat <- print(str_replace_all(df$PreferedOrderCat,"Mobile Phone Phone", "Mobile Phone"))
```

```
 Computer     Phone
0.2902309 0.7097691
> prop.table(table(df$PreferredPaymentMode))

Cash on Delivery      Credit Card       Debit Card         E wallet              UPI
      0.09129663       0.31509769       0.41101243       0.10905861       0.07353464
> prop.table(table(df$PreferedOrderCat))

           Fashion              Grocery Laptop & Accessory         Mobile Phone               Others
        0.14671403           0.07282416           0.36412078           0.36944938           0.04689165
> 
```

Next, its correction of the cell which has mistake in categories such as “CC” and “Credit Card”. This is to make sure the data have true categories instead of redundant categories. After processing, the variable is checked again, and the result showed that the categories are successfully treated.

4. Normalization

```
#Normalization
df$Tenure <- (df$Tenure - min(df$Tenure))/(max(df$Tenure) - min(df$Tenure))
df$OrderCount <- (df$OrderCount - min(df$OrderCount))/(max(df$OrderCount) - min(df$OrderCount))
df$CouponUsed <- (df$CouponUsed - min(df$CouponUsed))/(max(df$CouponUsed) - min(df$CouponUsed))
df$DaySinceLastOrder <- (df$DaySinceLastOrder - min(df$DaySinceLastOrder))/(max(df$DaySinceLastOrder) - min(df$DaySinceLastOrder))
df$NumberOfAddress <- (df$NumberOfAddress - min(df$NumberOfAddress))/(max(df$NumberOfAddress) - min(df$NumberOfAddress))
df$OrderAmountHikeFromlastYear <- (df$OrderAmountHikeFromlastYear - min(df$OrderAmountHikeFromlastYear))/
  (max(df$OrderAmountHikeFromlastYear) - min(df$OrderAmountHikeFromlastYear))
df$WarehouseToHome <- (df$WarehouseToHome - min(df$WarehouseToHome))/(max(df$WarehouseToHome) - min(df$WarehouseToHome))
df$CashbackAmount <- (df$CashbackAmount - min(df$CashbackAmount))/(max(df$CashbackAmount) - min(df$CashbackAmount))
```

Next, is the normalization of the continuous variable to get a better skewness.

5. One hot encoding

```
#One-hot encoding

library(fastDummies)

df <- dummy_cols(df, select_columns = 'PreferedOrderCat')
df <- dummy_cols(df, select_columns = 'PreferredLoginDevice')
df <- dummy_cols(df, select_columns = 'PreferredPaymentMode')
df <- dummy_cols(df, select_columns = 'Gender')
df <- dummy_cols(df, select_columns = 'MaritalStatus')

df <- select(df, -3, -6, -7, -10, -12) # drop selected columns
```

Next, convert the categorical variable into numeric by using the fastDummies library. The selected variable is converted while the old variables are dropped afterwards.

6. Balancing

Next is data balancing, since this project decided to work on different data balancing method. Hence, there are total 4 types of data balancing are carried out such as SMOTE, random over sampling, random under sampling and random both sampling. In this section 2 library are used which include smotefamily library and ROSE library. The code of the data balancing is shown below.

i) SMOTE

```
#Balancing
library(smotefamily)
df <- SMOTE(df[-1],df$Churn)
df=df$data
table(df$class)
prop.table(table(df$class))
```

ii) Oversampling

```
# ROSE oversampling balancing
library(ROSE)
df <- ovun.sample(Churn ~ ., data = df, method = "over",N = 10000)$data
table(df$Churn)
prop.table(table(df$Churn))
```

iii) Undersampling

```
library(ROSE)
df <- ovun.sample(Churn ~ ., data = df, method = "under",N = 1200, seed=12345)$data
table(df$Churn)
prop.table(table(df$Churn))
```

iv) Both

```
#Resampling-both method
df <- ovun.sample(Churn ~ ., data = df, method = "both", p=0.5, N=5000, seed = 1)$data
table(df$Churn)
prop.table(table(df$Churn))
```

7. Train-test split

```
library(rsample)
stratify <- initial_split(df,prop = 0.7,strata = class)
stratify

training_set <- training(stratify)
prop.table(table(training_set$class))


test_set <- testing(stratify)
prop.table(table(test_set$class))
```

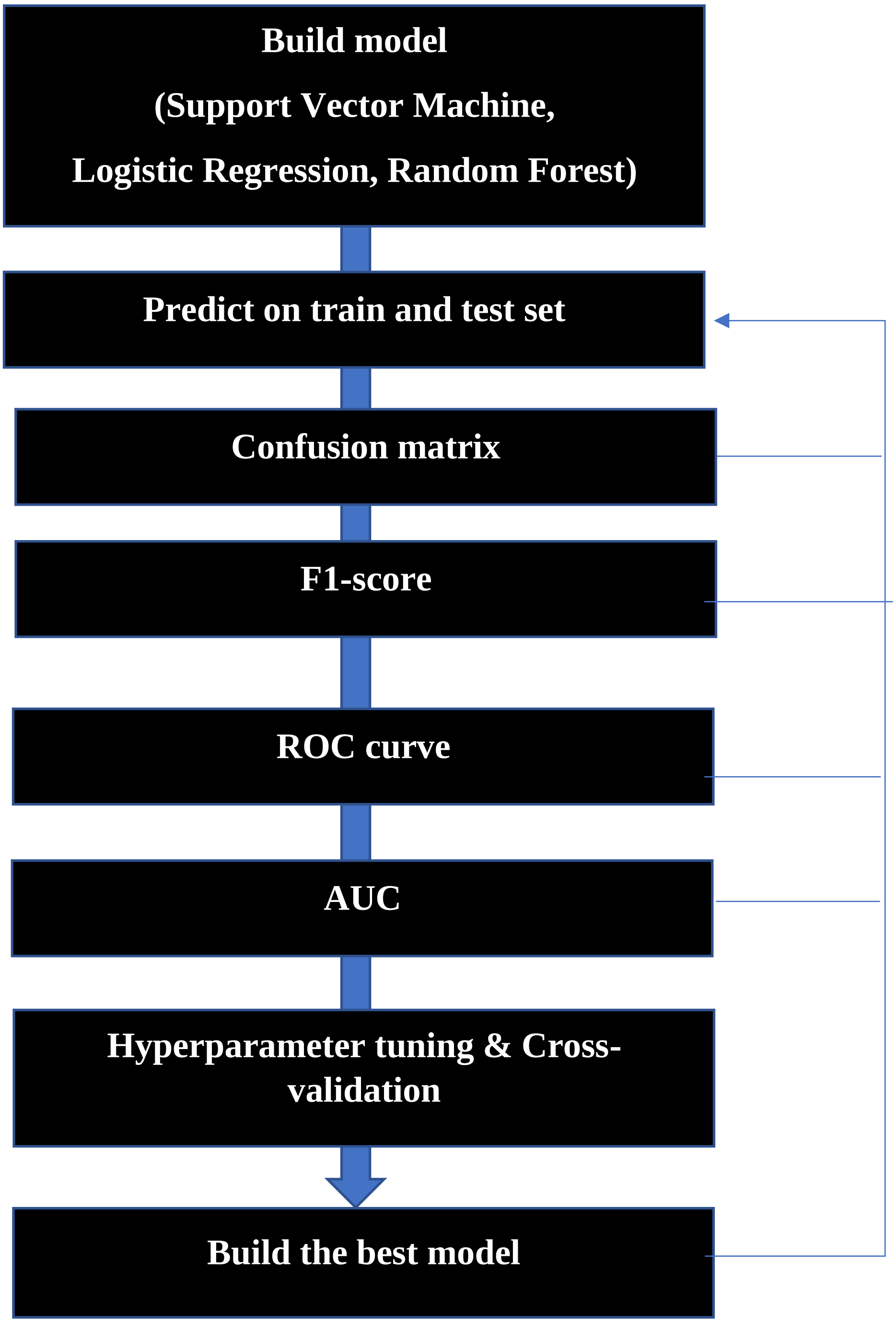
```
> stratify
<Analysis/Assess/Total>
<5931/2543/8474>
> training_set <- training(stratify)
> prop.table(table(training_set$class))

        0         1
0.5525207 0.4474793
> test_set <- testing(stratify)
> prop.table(table(test_set$class))

        0         1
0.5524971 0.4475029
>
```

Next, in the train test split session, the stratified sampling is chosen rather than random sampling. This is because stratified sampling provides a minimum error and greater precision. The above showing the code and output to split the dataset and also view on the proportion of the dataset to make sure the training set has balance data distribution in the target variable in order to build a more accurate model. The proportion of the model is 55% class “0” and 45% class “1” after splitting for both training and testing set. This is to double check and ensure there is no imbalance situation happen before model training.

5.0 Model Implementation and Validation

5.1 Model Building

5.1.1 Model Building Flow



In the section of model building, there are total 3 types of models are built, which include support vector machine, logistic regression and random forest. Next, after building the model, confusion matrix is used to review the evaluation matrix of the model such as accuracy, sensitivity, and specificity. Other than these 3 parameters, another parameter which is F1 measure, AUC and ROC is also display to further understand the effectiveness of the built model. Next, hyperparameter tuning and cross validation is carry out to optimize the model performance. Last but not least, build the best model by using the tuned parameter and compare with the based model performance.

5.1.2 Model Building Demonstration

This section is mainly used for demonstration purpose. It is to demonstrate the flow of model building by using support vector machine as an example. In this project, there are 3 models build on total 4 resampling method. If display every coding and output screenshot together, it will cause a difficult the read. So, this section will demonstrate the coding and output of SVM. The libraries and hyperparameter of each model will be discuss in this part. After that, the output will be collected together into tables for easy reading and make further discussion on the result of each part.

Step 1. Build model

```
svm_rbf <- svm(class~., data = training_set)
```

The figure above shows the code of model building in support vector machine. In support vector machine, there are different kernels can be choose. In our case, the default kernel is chosen, which is the RBF kernel. The RBF Kernel is famous because of its resemblance to the K-Nearest Neighbor Algorithm. It benefits from K-NN and avoids the storage difficulty issue, it only need to keep the support vectors while learning, not the full data set. The library that used to build the SVM model is library e1071. For logistic regression the model is built by using the glm function while random forest is using randomForest library. In the glm code, binomial family is chosen because the target variable is only 2 class, which is “1” and “0” which indicate “Churn” and “Not Churn” respectively. The different coding snippet of this two models are displayed belowed.

i) glm code snippet to build **logistic regression**:

```
classifier = glm(class ~.,
                 training_set,
                 family = binomial)
```

ii) **i) randomForest** code snippet to build **random forest**:

```
rf <- randomForest(class~.,data = training_set)
```

Step 2: Predict on train and test set

```
pred_rbf_training = predict (svm_rbf, training_set)
pred_rbf_test = predict (svm_rbf, test_set)
```

Next, use the model to predict on the train and testing set by using the code above.

Step 3. Confusion matrix

```
cm_training = table(Predicted = pred_rbf_training, Actual = training_set$class)
confusionMatrix(cm_training)
```

```
cm_test = table(Predicted = pred_rbf_test, Actual = test_set$class)
confusionMatrix(cm_test)
```

After prediction is done, the code above is run to view the confusion matrix, this is to understand the performance of the models. In this part, the caret library is used to build confusion matrix. The confusion matrix is divided into training and testing mainly is to understand the fitness of the models. If the training set accuracy is far higher than test set, this indicate that an overfit problem happens while if the test set accuracy is higher than the training set accuracy, then it is underfit problem. Hence, by understand the fitting problem, it also reflects the performance of the models. After running the code above the sample output is shown below:

Output

Training set

```
          Actual
Predicted    0    1
        0 3254  219
        1   23  444

               Accuracy : 0.9386
                 95% CI : (0.9306, 0.9459)
    No Information Rate : 0.8317
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7512

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

            Sensitivity : 0.9930
            Specificity : 0.6697
         Pos Pred Value : 0.9369
         Neg Pred Value : 0.9507
             Prevalence : 0.8317
         Detection Rate : 0.8259
   Detection Prevalence : 0.8815
      Balanced Accuracy : 0.8313
```

Original dataset

```
          Actual
Predicted    0    1
        0 1682   47
        1  112 1658

               Accuracy : 0.9546
                 95% CI : (0.9471, 0.9612)
    No Information Rate : 0.5127
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9091

 Mcnemar's Test P-Value : 3.864e-07

            Sensitivity : 0.9376
            Specificity : 0.9724
         Pos Pred Value : 0.9728
         Neg Pred Value : 0.9367
             Prevalence : 0.5127
         Detection Rate : 0.4807
   Detection Prevalence : 0.4941
```

Random Both Sampling

```
Confusion Matrix and Statistics

          Actual
Predicted    0    1
        0 3109   71
        1  168 2583

               Accuracy : 0.9597
                 95% CI : (0.9544, 0.9646)
    No Information Rate : 0.5525
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9188

 Mcnemar's Test P-Value : 5.308e-10

            Sensitivity : 0.9487
            Specificity : 0.9732
         Pos Pred Value : 0.9777
         Neg Pred Value : 0.9389
             Prevalence : 0.5525
         Detection Rate : 0.5242
   Detection Prevalence : 0.5362
      Balanced Accuracy : 0.9610
```

SMOTE

```
Confusion Matrix and Statistics

          Actual
Predicted    1    0
        1 3683  222
        0   39 3055

               Accuracy : 0.9627
                 95% CI : (0.958, 0.967)
    No Information Rate : 0.5318
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9249

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

            Sensitivity : 0.9895
            Specificity : 0.9323
         Pos Pred Value : 0.9431
         Neg Pred Value : 0.9874
             Prevalence : 0.5318
         Detection Rate : 0.5262
   Detection Prevalence : 0.5579
      Balanced Accuracy : 0.9609
```

Random Oversampling

```
Confusion Matrix and Statistics

         Actual
Predicted   1   0
        1 427  44
        0  26 342

               Accuracy : 0.9166
                 95% CI : (0.8958, 0.9344)
    No Information Rate : 0.5399
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8315

 Mcnemar's Test P-Value : 0.04216

            Sensitivity : 0.9426
            Specificity : 0.8860
         Pos Pred Value : 0.9066
         Neg Pred Value : 0.9293
             Prevalence : 0.5399
         Detection Rate : 0.5089
   Detection Prevalence : 0.5614
```

Random Undersampling

Test set

```
          Actual
Predicted    0    1
        0 1383  127
        1   22  158

               Accuracy : 0.9118
                 95% CI : (0.8973, 0.9249)
    No Information Rate : 0.8314
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6315

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

            Sensitivity : 0.9843
            Specificity : 0.5544
         Pos Pred Value : 0.9159
         Neg Pred Value : 0.8778
             Prevalence : 0.8314
         Detection Rate : 0.8183
   Detection Prevalence : 0.8935
      Balanced Accuracy : 0.7694
```

Original dataset

```
          Actual
Predicted   0   1
        0 699  46
        1  71 685

               Accuracy : 0.9221
                 95% CI : (0.9073, 0.9351)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8441

 Mcnemar's Test P-Value : 0.0265

            Sensitivity : 0.9078
            Specificity : 0.9371
         Pos Pred Value : 0.9383
         Neg Pred Value : 0.9061
             Prevalence : 0.5130
         Detection Rate : 0.4657
   Detection Prevalence : 0.4963
      Balanced Accuracy : 0.9224
```

Random Both Sampling

```
Confusion Matrix and Statistics

          Actual
Predicted    0    1
        0 1302   62
        1  103 1076

               Accuracy : 0.9351
                 95% CI : (0.9248, 0.9444)
    No Information Rate : 0.5525
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8692

 Mcnemar's Test P-Value : 0.001846

            Sensitivity : 0.9267
            Specificity : 0.9455
         Pos Pred Value : 0.9545
         Neg Pred Value : 0.9126
             Prevalence : 0.5525
         Detection Rate : 0.5120
   Detection Prevalence : 0.5364
      Balanced Accuracy : 0.9361
```

SMOTE

```
Confusion Matrix and Statistics

          Actual
Predicted    1    0
        1 1575  125
        0   21 1280

               Accuracy : 0.9513
                 95% CI : (0.943, 0.9588)
    No Information Rate : 0.5318
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9019

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

            Sensitivity : 0.9868
            Specificity : 0.9110
         Pos Pred Value : 0.9265
         Neg Pred Value : 0.9839
             Prevalence : 0.5318
         Detection Rate : 0.5248
   Detection Prevalence : 0.5665
```

Random Over Sampling

```
Confusion Matrix and Statistics

          Actual
Predicted   1   0
        1 177  33
        0  18 133

               Accuracy : 0.8587
                 95% CI : (0.8185, 0.893)
    No Information Rate : 0.5402
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7137

 Mcnemar's Test P-Value : 0.04995

            Sensitivity : 0.9077
            Specificity : 0.8012
         Pos Pred Value : 0.8429
         Neg Pred Value : 0.8808
             Prevalence : 0.5402
         Detection Rate : 0.4903
   Detection Prevalence : 0.5817
```

Random Under Sampling

In this output, it involved both training and testing confusion matrix, the three main performance matrix that be observed are accuracy, specificity, and sensitivity. Based on the output above, there are total 5 output because the model is built on 4 types of resampling method and also the original dataset. Hence, in model building section, there are total 5 R files in this project.

Step 4. F1-score

```
F1_train <- F1_Score(y_pred = pred_rbf_training, y_true = training_set$class, positive = NULL)
F1_train
```

```
F1_test <- F1_Score(y_pred = pred_rbf_test, y_true = test_set$class, positive = NULL)
F1_test
```

Output:

Resampling Method	Training F1 score	Test F1 score
Original	0.9641481	0.9488851
SMOTE	0.9629859	0.9404117
Random oversampling	0.9657795	0.9557039
Random undersampling	0.9242424	0.8740741
Random both sampling	0.9548681	0.9227723

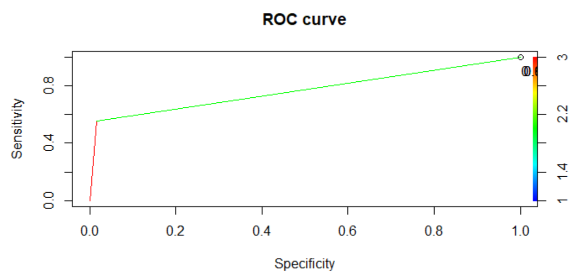
After confusion matrix, the next evaluation parameter for our model is F-score. The library that was used in this part is library(MLmetrics). It is also divided into training and testing set.

Step 5. ROC curve

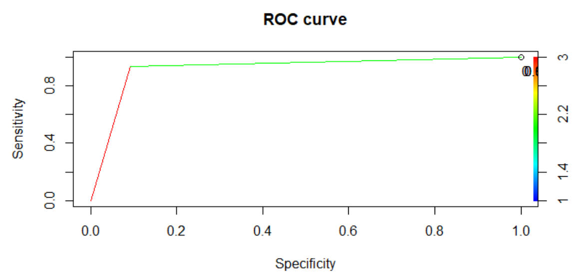
```
pred = prediction(as.numeric(pred_rbf_test), as.numeric(test_set$class))
perf = performance(pred, "tpr", "fpr")
pred
perf
plot(perf, colorize = T)
plot(perf, colorize=T,
    main = "ROC curve",
    ylab = "Sensitivity",
    xlab = "Specificity",
    print.cutoffs.at=seq(0,1,0.3),
    text.adj= c(-0.2,1.7))
```

Next, ROC curve is plot to evaluate the performance of the models. The library that was used in this part is library ROCR. When the plot is closer to the top left, indicate that the model has better performance. The sample output from the code above is shown below.

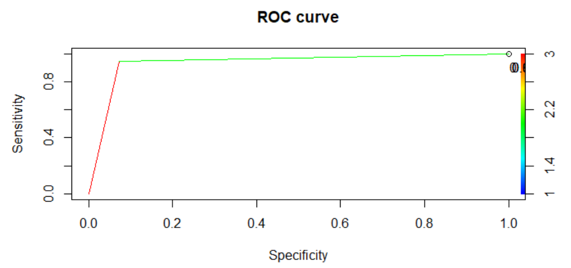
Output



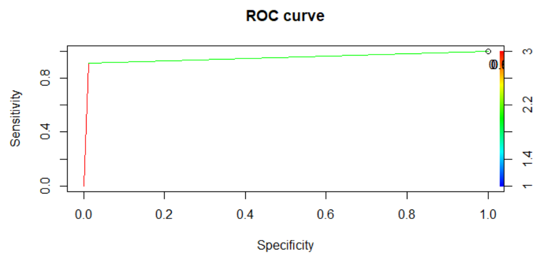
No resampling (Original)



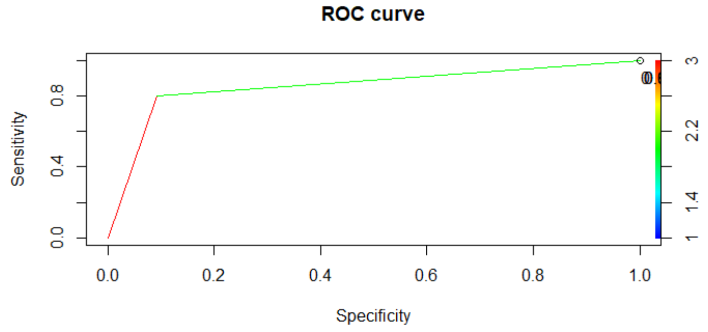
Both resampling



SMOTE



Random Oversampling



Random Undersampling

Based on the figure above, it is obviously showing that the model that do not have any balancing showing poor performance as the curve is far from the left corner.

Step 6. AUC

```
auc <- as.numeric(performance(pred, "auc")@y.values)
auc <-  round(auc, 3)
auc
```

Output:

Resampling Method	AUC
No resampling	0.769
SMOTE	0.936
Random oversampling	0.949
Random undersampling	0.854
Random both sampling	0.922

Next, the area under the curve is view in term of value, when the value is high indicate that the model has higher performance while if the value is low indicate that the model has poor performance. Based on the sample output, we can see that the dataset that never carry out any rebalance bring a poorer performance than others.

Step 7. Hyperparameter tuning

Next, hyperparameter tuning is carry out to find out the best hyperparameter to optimize the models. In this part, the library of Caret and Kernel are used. During tuning, 10-fold validation is selected to improve the tuning accuracy. In this support vector machine, the method of "svmRadial" is chosen, which is specific for SVM-RBF. The tuning method chosen is random search which is a popular tuning method. There are different parameters that can be tuned. In this project, the parameters that are tuned in support vector machine are sigma and Cost. The output of this code showing the best parameter, and we can insert this value into our optimize model.

```
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
set.seed(123)
tuning_cv <- train(class~., data = training_set, method = "svmRadial", trControl = ctrl)
```

Output:

No resampling (Original)

```
Support Vector Machines with Radial Basis Function Kernel

3940 samples
  30 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 3546, 3546, 3545, 3545, 3546, 3546, ...
Resampling results across tuning parameters:

  sigma       C             Accuracy   Kappa
  0.01053689    0.09435281  0.8317267  0.0000000
  0.02794108  117.98276415  0.9667608  0.8807414
  0.04398335   22.88246468  0.9705680  0.8933279

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.04398335 and C = 22.88246.
```

Both Resampling Method

```
Support Vector Machines with Radial Basis Function Kernel

3499 samples
  30 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 3150, 3149, 3150, 3150, 3149, 3149, ...
Resampling results across tuning parameters:

  sigma       C            Accuracy   Kappa
  0.01378538    0.0868607  0.8153769  0.6307222
  0.02074597   93.6328619  0.9748570  0.9497204
  0.05423493  539.3333940  0.9814268  0.9628531

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.05423493 and C = 539.3334.
```

SMOTE

```
Support Vector Machines with Radial Basis Function Kernel

5931 samples
  30 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 5338, 5337, 5338, 5338, 5339, 5337, ...
Resampling results across tuning parameters:

  sigma        C           Accuracy   Kappa
  0.006728567  48.3013377  0.9430130  0.8852712
  0.017161062   0.1552584  0.8487612  0.6940074
  0.060137756   1.5305825  0.9735332  0.9466858

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.06013776 and C = 1.530583.
```

Random Oversampling

```
> tuning_cv
Support Vector Machines with Radial Basis Function Kernel

6999 samples
  30 predictor
   2 classes: '1', '0'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 6299, 6300, 6298, 6299, 6298, 6300, ...
Resampling results across tuning parameters:

  sigma        C            Accuracy   Kappa
  0.005495655  22.24559893  0.9121295  0.8229818
  0.038596210   2.81041014  0.9769971  0.9536751
  0.060080460   0.07631673  0.8706950  0.7393996

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.03859621 and C = 2.81041.
> F1_test <- F1_Score(y_pred = pred_rbf_test, y_true = test_set$class, positive
  = NULL)
```

Random undersampling

```
> tuning_cv
Support Vector Machines with Radial Basis Function Kernel

839 samples
 30 predictor
  2 classes: '1', '0'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 755, 756, 755, 755, 756, 754, ...
Resampling results across tuning parameters:

  sigma        C           Accuracy   Kappa
  0.008382848   1.0994750  0.7962586  0.5883571
  0.010343437  10.8163886  0.7855150  0.5664133
  0.043047783   0.5970589  0.7915248  0.5796628

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.008382848 and C
 = 1.099475.
```

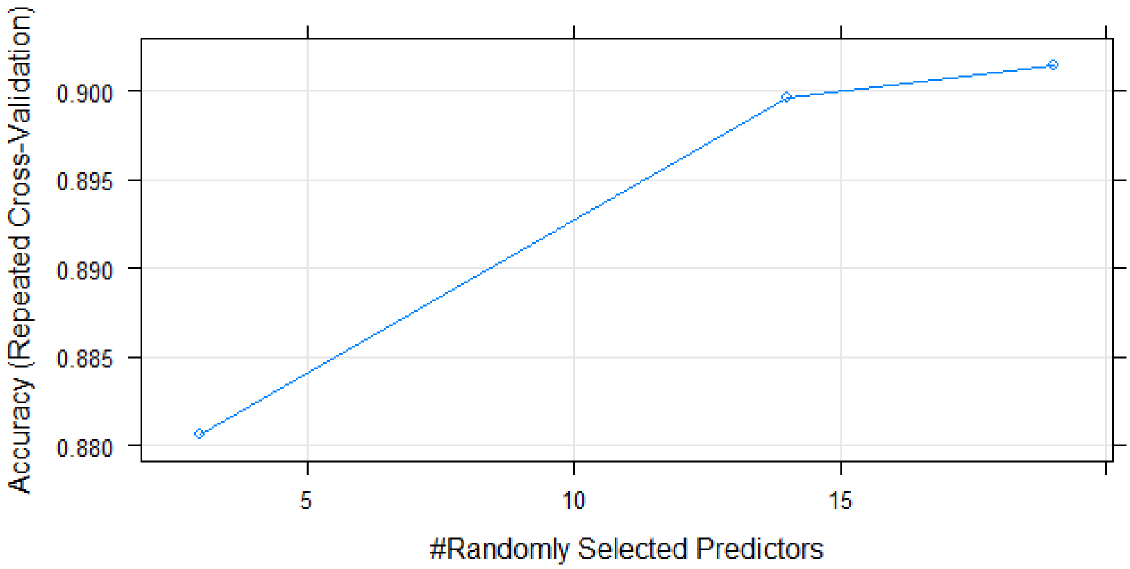
For Logistic regression, there is no hyperparameter free for tuning, hence, there is no hyperparameter tuning in logistic regression. For random forest, the hyperparameter that was tuned is mtry. mtry specifies the number of features picked randomly as candidates at each split. The sample code snippet and output are shown below. The method that was chosen is “rf” which is specific for random forest in caret tuning while the metric chosen is accuracy. In the tuning of random forest, random search with 10-fold cross validation and 3 times repetition is selected with “repeatedcv” method by using trainControl function. Then this function is insert into the trControl under train function.

```
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
set.seed(123)
metric <- "Accuracy"
rf_random <- train(class ~ ., data=training_set, method="rf", metric=metric, trControl=control)
```

Output:

```
Random Forest

909 samples
 30 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 818, 818, 818, 819, 818, 819, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   3    0.8805734  0.6566035
  14    0.8996147  0.7278391
  19    0.9014543  0.7334507

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 19.
```

Based on the above output, the optimal mtry value is 19. And this can be visualized through plots below:



Based on the plot it can be see that the accuracy of the model is highest when the mtry value is 19. Hence, the mtry=19 result is display for us to use this code for following optimization process.

Step 8. Build the best model

In the previous part, it had mentioned that the parameter tuning used for SVM-RBF are sigma and Cost. After getting the optimal value from the hyperparameter tuning part, then the value is inserted into a new model to build the best fit model. The code below showing both svm and random forest code snippet the understand how to insert the value to build new models. After the model building, the following steps are repeat from steps 1 that mentioned just now until the step of display the area under curve (AUC). Then the result is compared with the based model to review the difference.

```
svm_best <- svm (class~., data = training_set, sigma = 0.06013776, cost = 1.530583)
```

```
rf_tuned <- randomForest(class~.,data = training_set, mtry = 19)
```

Output:

Both sampling

Training

```
Confusion Matrix and Statistics

          Actual
Predicted    0    1
        0 1794    0
        1    0 1705

               Accuracy : 1
                 95% CI : (0.9989, 1)
    No Information Rate : 0.5127
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.5127
         Detection Rate : 0.5127
   Detection Prevalence : 0.5127
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

Test

```
Confusion Matrix and Statistics

          Actual
Predicted   0   1
        0 750   8
        1  20 723

               Accuracy : 0.9813
                 95% CI : (0.9732, 0.9876)
    No Information Rate : 0.513
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9627

 Mcnemar's Test P-Value : 0.03764

            Sensitivity : 0.9740
            Specificity : 0.9891
         Pos Pred Value : 0.9894
         Neg Pred Value : 0.9731
             Prevalence : 0.5130
         Detection Rate : 0.4997
   Detection Prevalence : 0.5050
      Balanced Accuracy : 0.9815

       'Positive' Class : 0
```

SMOTE

Training

```
> confusionMatrix(cm_training)
Confusion Matrix and Statistics

          Actual
Predicted    0    1
        0 3140   48
        1  137 2606

               Accuracy : 0.9688
                 95% CI : (0.9641, 0.9731)
    No Information Rate : 0.5525
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9371

 Mcnemar's Test P-Value : 9.808e-11

            Sensitivity : 0.9582
            Specificity : 0.9819
         Pos Pred Value : 0.9849
         Neg Pred Value : 0.9501
             Prevalence : 0.5525
         Detection Rate : 0.5294
   Detection Prevalence : 0.5375
      Balanced Accuracy : 0.9701

       'Positive' Class : 0
```

Test

```
Confusion Matrix and Statistics

         Actual
Predicted    0    1
        0 1307   50
        1   98 1088

               Accuracy : 0.9418
                 95% CI : (0.932, 0.9506)
    No Information Rate : 0.5525
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8828

 Mcnemar's Test P-Value : 0.0001118

            Sensitivity : 0.9302
            Specificity : 0.9561
         Pos Pred Value : 0.9632
         Neg Pred Value : 0.9174
             Prevalence : 0.5525
         Detection Rate : 0.5140
   Detection Prevalence : 0.5336
      Balanced Accuracy : 0.9432

       'Positive' Class : 0
```

Random oversampling

Training

```
Confusion Matrix and Statistics

          Actual
Predicted    1    0
        1 3720  101
        0    2 3176

               Accuracy : 0.9853
                 95% CI : (0.9822, 0.988)
    No Information Rate : 0.5318
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9704

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

            Sensitivity : 0.9995
            Specificity : 0.9692
         Pos Pred Value : 0.9736
         Neg Pred Value : 0.9994
             Prevalence : 0.5318
         Detection Rate : 0.5315
   Detection Prevalence : 0.5459
      Balanced Accuracy : 0.9843

       'Positive' Class : 1
```

Test

```
Confusion Matrix and Statistics

          Actual
Predicted    1    0
        1 1595   69
        0    1 1336

               Accuracy : 0.9767
                 95% CI : (0.9706, 0.9818)
    No Information Rate : 0.5318
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.953

 Mcnemar's Test P-Value : 1.166e-15

            Sensitivity : 0.9994
            Specificity : 0.9509
         Pos Pred Value : 0.9585
         Neg Pred Value : 0.9993
             Prevalence : 0.5318
         Detection Rate : 0.5315
   Detection Prevalence : 0.5545
      Balanced Accuracy : 0.9751

       'Positive' Class : 1
```

Random Under Sampling

Training

```
Confusion Matrix and Statistics

          Actual
Predicted   1   0
        1 428  42
        0  25 344

               Accuracy : 0.9201
                 95% CI : (0.8997, 0.9376)
    No Information Rate : 0.5399
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8387

 Mcnemar's Test P-Value : 0.05062

            Sensitivity : 0.9448
            Specificity : 0.8912
         Pos Pred Value : 0.9106
         Neg Pred Value : 0.9322
             Prevalence : 0.5399
         Detection Rate : 0.5101
   Detection Prevalence : 0.5602
      Balanced Accuracy : 0.9180

       'Positive' Class : 1
```

Test

```
Confusion Matrix and Statistics

          Actual
Predicted   1   0
        1 180  31
        0  15 135

               Accuracy : 0.8726
                 95% CI : (0.8337, 0.9052)
    No Information Rate : 0.5402
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7416

 Mcnemar's Test P-Value : 0.02699

            Sensitivity : 0.9231
            Specificity : 0.8133
         Pos Pred Value : 0.8531
         Neg Pred Value : 0.9000
             Prevalence : 0.5402
         Detection Rate : 0.4986
   Detection Prevalence : 0.5845
      Balanced Accuracy : 0.8682

       'Positive' Class : 1
```

No Resampling (Original)

Training

```
Confusion Matrix and Statistics

          Actual
Predicted    0    1
        0 3276   11
        1    1  652

               Accuracy : 0.997
                 95% CI : (0.9947, 0.9984)
    No Information Rate : 0.8317
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9891

 Mcnemar's Test P-Value : 0.009375

            Sensitivity : 0.9997
            Specificity : 0.9834
         Pos Pred Value : 0.9967
         Neg Pred Value : 0.9985
             Prevalence : 0.8317
         Detection Rate : 0.8315
   Detection Prevalence : 0.8343
      Balanced Accuracy : 0.9916

       'Positive' Class : 0
```

Test

```
Confusion Matrix and Statistics

          Actual
Predicted    0    1
        0 1389   30
        1   16  255

               Accuracy : 0.9728
                 95% CI : (0.9639, 0.98)
    No Information Rate : 0.8314
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.901

 Mcnemar's Test P-Value : 0.05527

            Sensitivity : 0.9886
            Specificity : 0.8947
         Pos Pred Value : 0.9789
         Neg Pred Value : 0.9410
             Prevalence : 0.8314
         Detection Rate : 0.8219
   Detection Prevalence : 0.8396
      Balanced Accuracy : 0.9417

       'Positive' Class : 0
```

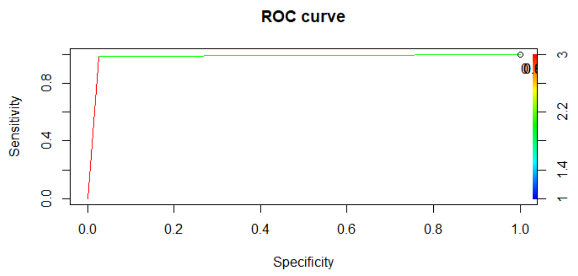
Tuned F1 Score

Data resampling method	Training	Test
Original	0.9981718	0.9837111
SMOTE	0.9713844	0.9464156
Random oversampling	0.986345	0.9785276
Random undersampling	0.9274106	0.8866995
Random both sampling	0.954868	0.9227723

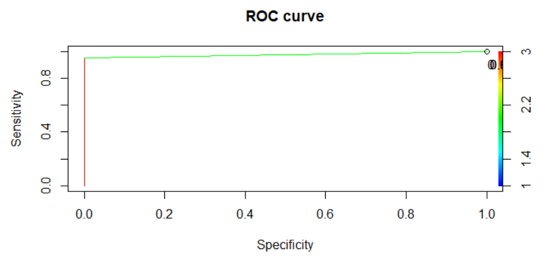
Tuned AUC

Resampling Method	AUC
No resampling	0.942
SMOTE	0.943
Random oversampling	0.975
Random undersampling	0.868
Random both sampling	0.922

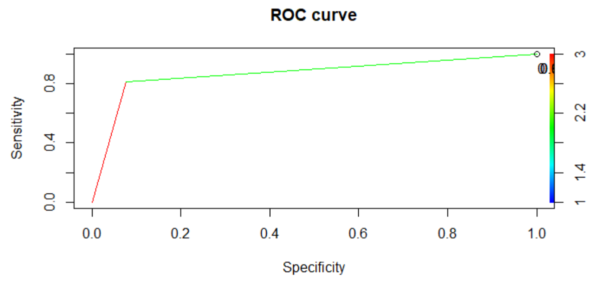
Tuned ROC



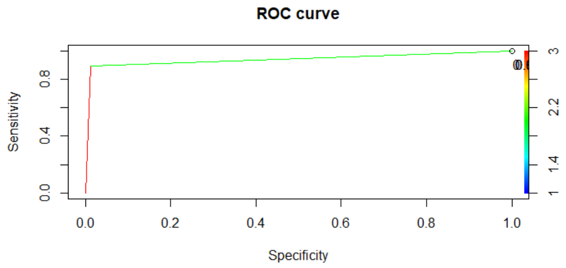
Random Both Sampling



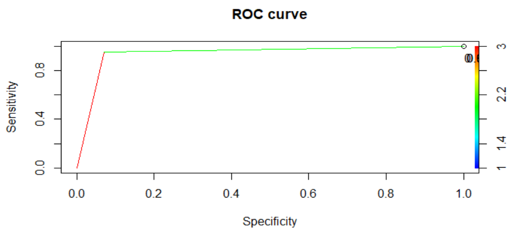
Random Oversampling



Random undersampling



No resampling (Original)



SMOTE

Based on above demonstration, the steps after the hyperparameter tuning are totally same as the based model building. Demonstration of the model building flow is end. In the next section, it is going to display the results of each machine learning model on different balancing dataset in table form.

5.2 Output/Results

5.2.1 Support Vector Machine

Table 5.1: Evaluation calculation (Training Set)

Based Support Vector Machine (Radial Basis Function kernel)					
	Accuracy (%)	Sensitivity (%)	Specificity (%)	F1-score (%)	Confusion Matrix
Original data	93.86	**99.30**	66.97	96.41	Actual Predicted 0 1 0 3254 219 1 23 444
SMOTE	95.97	94.87	**97.32**	96.30	Actual Predicted 0 1 0 3109 71 1 168 2583
Random over sampling	**96.27**	98.95	93.23	**96.58**	Actual Predicted 1 0 1 3683 222 0 39 3055
Random under sampling	91.66	94.26	88.60	92.42	Actual Predicted 1 0 1 427 44 0 26 342
Random both sampling	95.46	93.76	97.24	95.48	Actual Predicted 0 1 0 1682 47 1 112 1658
Tuned Support Vector Machine (Radial Basis Function kernel)					
Original data	99.7	99.97	98.34	99.82	Actual Predicted 0 1 0 3276 11 1 1 652
SMOTE	96.88	95.82	98.19	97.14	Actual Predicted 0 1 0 3140 48 1 137 2606
Random over sampling	98.53	99.95	96.92	98.63	Actual Predicted 1 0 1 3720 101 0 2 3176
Random under sampling	92.01	94.48	89.12	92.74	Actual Predicted 1 0 1 428 42 0 25 344
Random both sampling	**100**	**100**	**100**	**100**	Actual Predicted 0 1 0 1794 0 1 0 1705

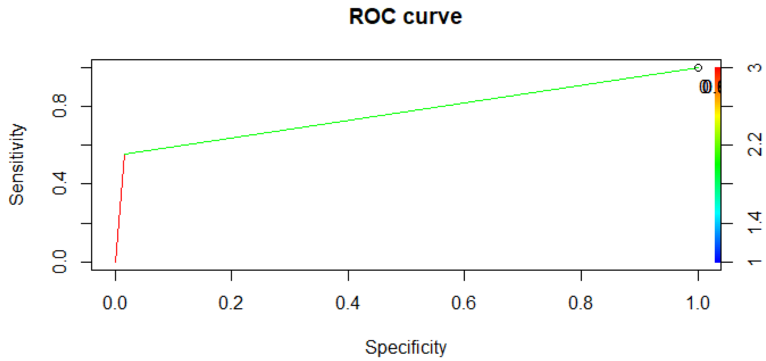
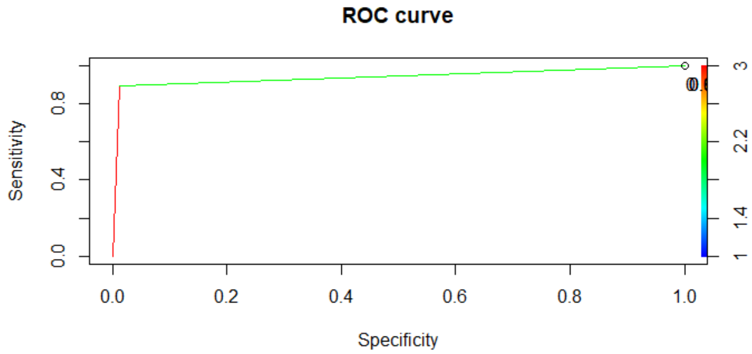
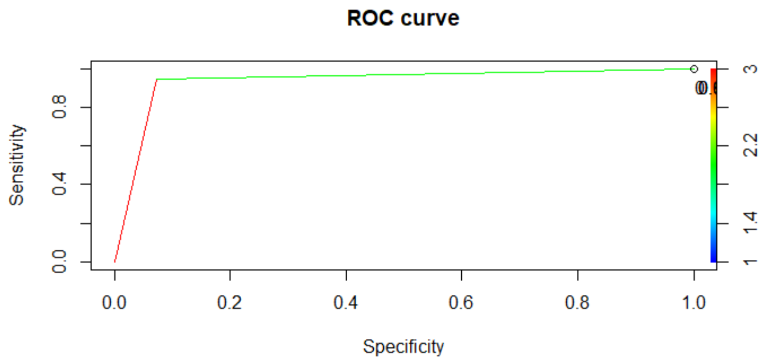
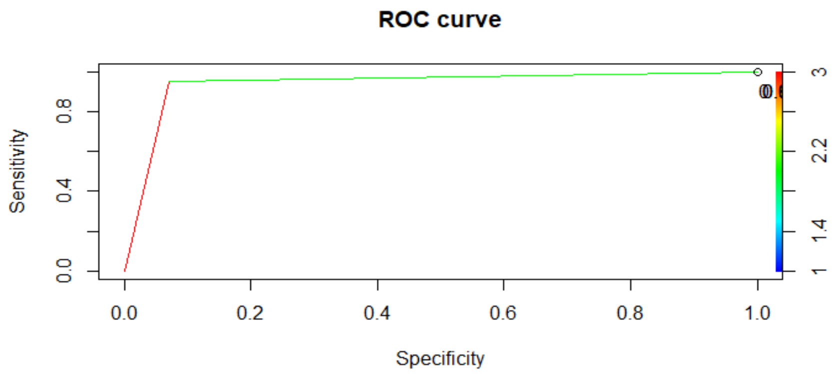
Table 5.2: Evaluation calculation (Test Set)

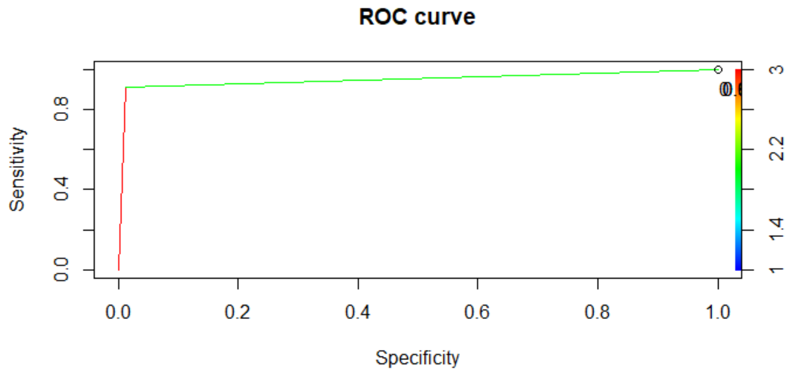
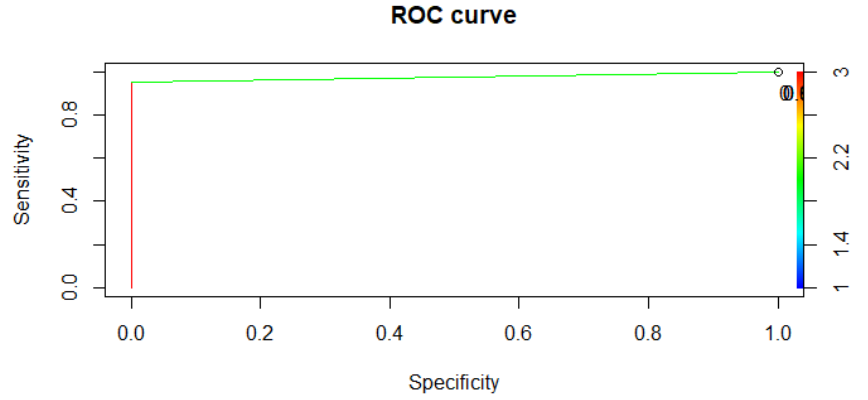
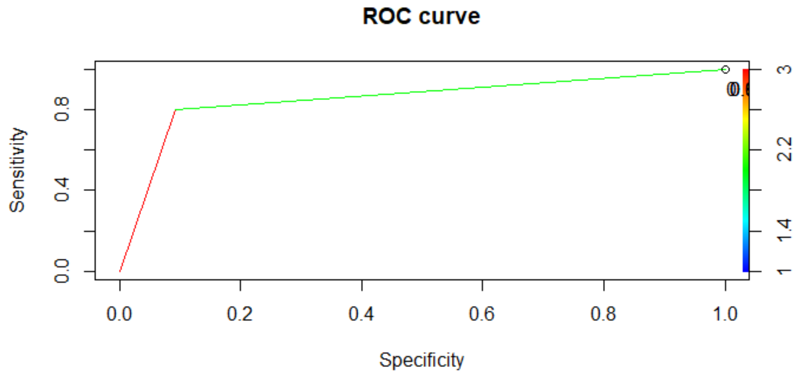
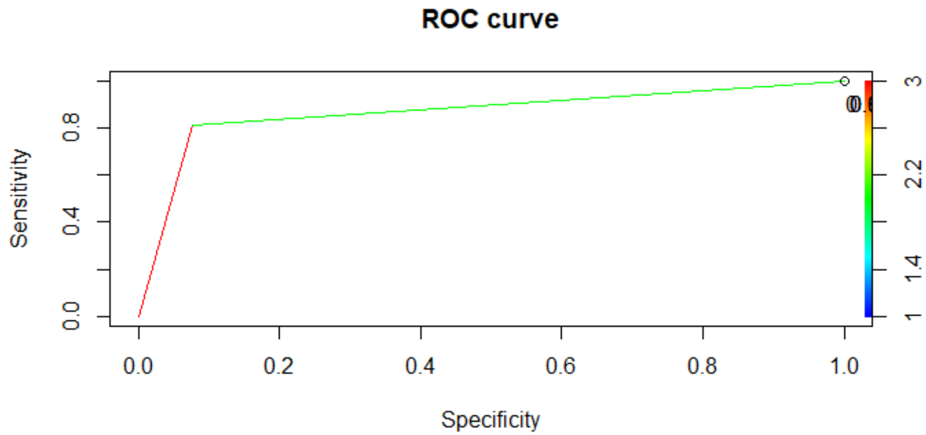
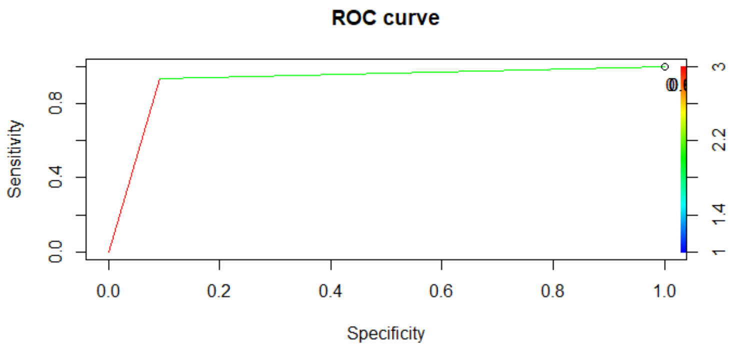
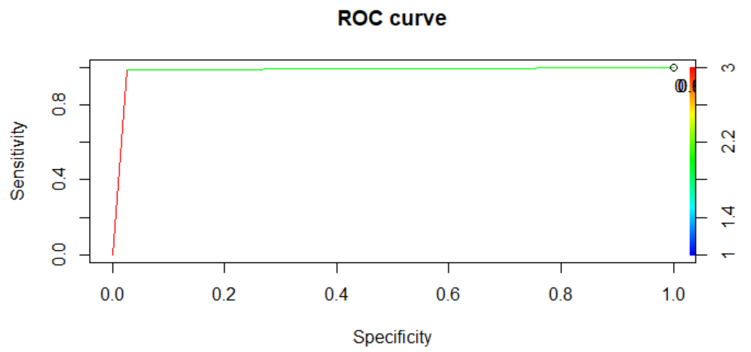
Support Vector Machine (Radial Basis Function kernel)					
	Accuracy (%)	**Sensitivity (%)**	**Specificity (%)**	**F1-score (%)**	**Confusion Matrix**
Original data	91.18	98.43	55.44	94.89	Actual Predicted 0 1 0 1383 127 1 22 158
SMOTE	93.51	92.67	**94.55**	94.04	Actual Predicted 0 1 0 1302 62 1 103 1076
Random over sampling	**95.13**	**98.68**	91.10	**95.57**	Actual Predicted 1 0 1 1575 125 0 21 1280
Random under sampling	85.87	90.77	80.12	87.40	Actual Predicted 1 0 1 177 33 0 18 133
Random both sampling	92.21	90.78	93.71	92.28	Actual Predicted 0 1 0 699 46 1 71 685
Tuned Support Vector Machine (Radial Basis Function kernel)					
Original data	97.28	98.86	89.47	98.37	Actual Predicted 0 1 0 1389 30 1 16 255
SMOTE	94.18	93.02	95.61	96.64	Actual Predicted 0 1 0 1307 50 1 98 1088
Random over sampling	97.67	**99.94**	95.09	**97.85**	Actual Predicted 1 0 1 1595 69 0 1 1336
Random under sampling	87.26	92.31	81.33	88.67	Actual Predicted 1 0 1 180 31 0 15 135
Random both sampling	**98.13**	97.40	**98.91**	97.83	Actual Predicted 0 1 0 750 8 1 20 723

Table 5.3 Fitness summary

Data sampling method	Fitness
Based SVM-RBF model	
Original dataset	Good
SMOTE	Good
random oversampling	Good
random undersampling	Good
random both sampling	Good
Tuned SVM-RBF model	
Original dataset	Good
SMOTE	Good
random oversampling	Good
random undersampling	Good
random both sampling	Good

Table 5.4 AUC and ROC

	Based SVM-RBF model		**Tuned SVM-RBF model**	
	AUC (%)	**ROC**	**AUC**	**ROC**
Original data	76.9		94.2	
SMOTE	93.6		94.3	

random over sampling	94.9		97.5	
random under sampling	85.4		86.8	
random both sampling	92.2		98.2	

Based on table 5.1 and 5.2 above, it showed the evaluation matrix for the training set and test set in the based model and tuned model of SVM-RBF. In the original dataset, it shows a poor AUC (76.9%) , poor ROC performance, significant low specificity (66.97%) in training set and only 55.44% specificity in test set. However, it has highest sensitivity which is 99.30% in training set and also high percentage sensitivity in test set which is about 98.43%. This is because in the original imbalance dataset, there are almost 85% class "0" and 15% class "1" in the dependent variable. Class "0" indicates not churn, class "1" indicates customer churn. In the model building, the "0" in the target variable is consider as positive (Figure 5.1).

```
'Positive' Class : 0
```

Figure 5.1: Class 0 as positive

With a high proportion of "0" data in the imbalance dataset, the model can learn the class "0" better than other dataset, hence result in high true positive value and minimum false negative value as shown in the confusion matrix. Hence, result in high sensitivity. On the other hand, since the proportion of "1" data in the original data is only around 15%, hence the models have poor learning on this class. Hence, it results is a low specificity percentage which indicate that the class negative which is class "1" cannot accurately identify by this model. Hence, this model that build on this imbalance dataset is not suitable for usage since the main purpose of the model is used to detect the churn risk, if the detection accuracy is low, then it might cause the company cannot prevent customer churn risk.

In the training set, the random under sampling dataset specificity is ranked the number two lowest which is 88.60% beyond the imbalance dataset, while all the other performance matrix such as accuracy, sensitivity and F1 score in test set are the lowest when compared with others dataset's model. While in the testing set, it has the lowest performance in all performance matrix. Although the dataset is done resampling by using under sampling method, however since the data is lower down to around few hundred in both training and testing set only, this causes there is a lack of information for the SVM model to learn and result in lower performance. In term of fitness, all models show good fitness as all the training set accuracy is slightly higher than the testing set accuracy.

On the other hand, models build on the random over sampling datasets is outperform other models since it shown that the random oversampling has highest performance in AUC (94.9%), ROC, both the accuracy (96.27%) and F1-score (96.58%) in training set, accuracy (95.13%), sensitivity (98.68%) and F1-score (95.7%) in the training set. In term of specificity, the SMOTE ranked highest percentage which is 97.32%. But in overall, SVM-RBF models that built on random both sampling, random oversampling and smote sampling dataset are performed well with higher than 90% in all performance matrix in both training and test set. In this context, it can summarize that the based SVM models can perform well in oversampling dataset but not the original and under sampling dataset. To further look into the potential of SVM model, hyperparameter tuning and 10-fold cross validation is carried out.

After hyperparameter tuning and 10-fold cross validation by using the random search under Caret library, the coming result show significant improvement especially the random both sampling which achieve 100% in all performance matrix in testing set and highest performance in AUC (98.2%). Other than random both sampling, another dataset that has obviously improvement is the original dataset. In term of specificity, the issue of imbalance dataset is solved by improved more than 30% specificity in both training set and test set. Other than specificity, the imbalance dataset also improved in AUC value and ROC performance. The AUC improved from 76.9% to 94.2% which more than 15%. Hence, it is obviously brought information that hyperparameter tuning in SVM able to handle imbalance dataset. In this context, it can be concluded that tuned SVM-RBF model is more ideal model for predicting customer churn risk in e-commerce than the based SVM-RBF model.

Logistic Regression

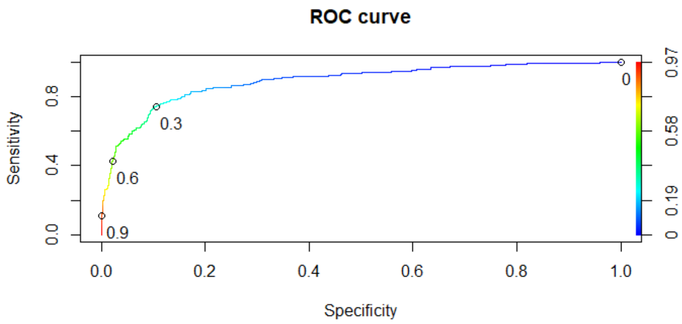
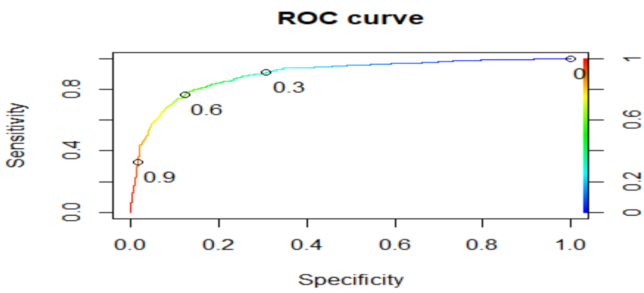
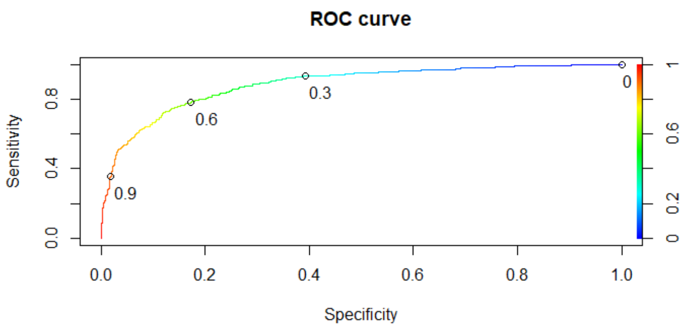
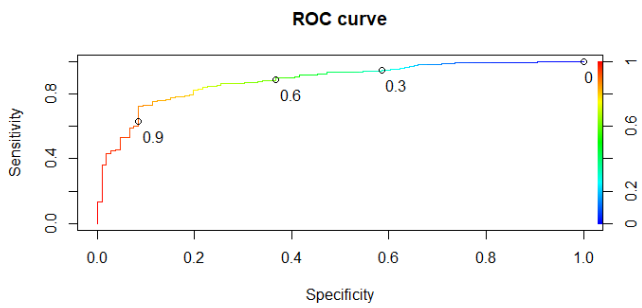
Table 5.5: Evaluation metrix

Logistic Regression (Training Set)					
	Accuracy (%)	**Sensitivity (%)**	**Specificity (%)**	**F1-score (%)**	**Confusion Matrix**
Original data	**89.29**	**90.82**	77.20	**93.77**	0 1 0 3176 101 1 321 342
SMOTE	82.50	84.23	80.18	81.67	0 1 0 2755 522 1 516 2138
Random oversampling	81.81	81.63	81.96	80.25	0 1 0 2586 691 1 582 3140
Random undersampling	85.37	76.78	**87.97**	70.90	0 1 0 162 84 1 49 614
Random combination sampling	80.68	80.88	80.46	81.24	0 1 0 1464 330 1 346 1359
Logistic Regression (Test set)					
Original data	**88.99**	**91.21**	73.46	**93.55**	0 1 0 1349 56 1 130 155
SMOTE	82.42	85.12	79.30	83.86	0 1161 244 1 203 935
Random over sampling	80.27	80.77	79.88	78.28	0 1 0 1067 338 1 254 1342
Random undersampling	82.35	71.26	**85.53**	64.25	0 1 0 62 44 1 25 260
Random combination sampling	81.15	82.86	79.47	81.27	0 1 0 614 156 1 127 604

Table 5.6 Fitness summary

Data sampling method	Fitness
Logistic regression	
Original dataset	Good
SMOTE	Good
random oversampling	Good
random undersampling	Good
random both sampling	Good

Table 5.7 AUC and ROC

Logistic Regression		
	AUC (%)	**ROC**
Original data	89.0	
SMOTE	90.0	
Random over sampling	88.8	
Random under sampling	88.2	

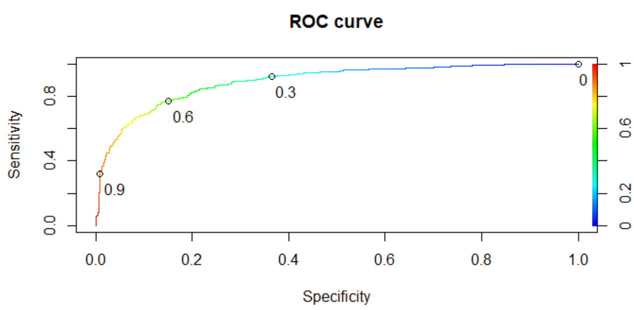
Random both sampling	89.3	

Table 5.5 above shows the result of the logistic regression, in overall, it seems less effective than support vector machine model as most of the performance result are stand around 80%. Based on the result, there is an interesting scenario which is the imbalance dataset achieve the highest performance in accuracy (89.29%), sensitivity (90.82%) and F1-score (93.77%) in training set and 88.99% accuracy, 91.21% sensitivity and 93.55% F1 score in testing set. The high sensitivity can be understood due to the imbalance dataset has highest proportion of class “0” lead the model has better learning in this class to give higher true positive result which lead to higher performance in sensitivity. However, in term of specificity, the model built on original dataset still remain the lowest which is only 77.20% in training set and 73.46% in testing set.

Based on the study of Rahman et al. (2021) which is research that study on the effect of imbalanced dataset on Logistic Regression. This study pointed out that synthetic datasets were created with regulated imbalanced ratios (IR) values ranging from 1% to 50% and depend on sample sizes. The author elaborated that when the sample size is large, the impact of imbalanced dataset on the parameter estimates of the covariate reduced as size of the sample grew. Although the author finds out that the data imbalance issue can be weaken when the data size is larger, but they still conclude that imbalance dataset has impact on the model accuracy even with larger dataset. In their conclusion, they suggested the balancing method which are SMOTE, random undersampling and random oversampling that are using in this project to balance dataset before building model.

Cheruku, (2019) also mentioned that imbalanced dataset can affect the logistic regression model performance by giving false result. The author also suggested data balancing before model building by using few balancing methods such as SMOTE, random oversampling and undersampling. In addition, another study from Eljatib et al., 2018 also pointed out the unbalanced dataset did lower down the performance of logistic regression, and the model performance such as sensitivity and specificity are improved after data balancing. The issue of highest accuracy and F1 score may be can be explained by the sensitivity percentage is significantly higher than other models with at least 5-10% but the specificity only 2-3% slightly lower than other models. Hence, through the calculation format, this condition indirectly increases the accuracy and F-score of the models in imbalanced dataset.

Based on table 5.6, all the models have good fitness because all their training accuracy is slightly higher than testing accuracy. Based on table 5.7, the ROC performance seems no significant different between models while the Area under curve (AUC) are in the midst of 88%-90%. In overall, the performance of the model that builtd on different dataset are around 80% which is consider good. In this context, it can be concluded that logistic regression model gives good performance in all types of datasets but not as excellent as the support vector machine that we previously investigated.

Random Forest

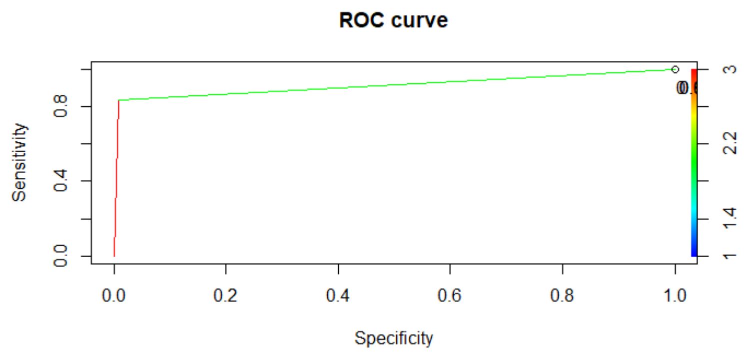
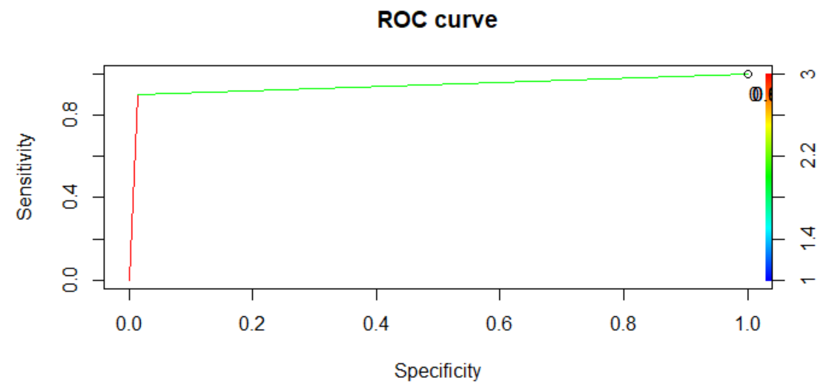
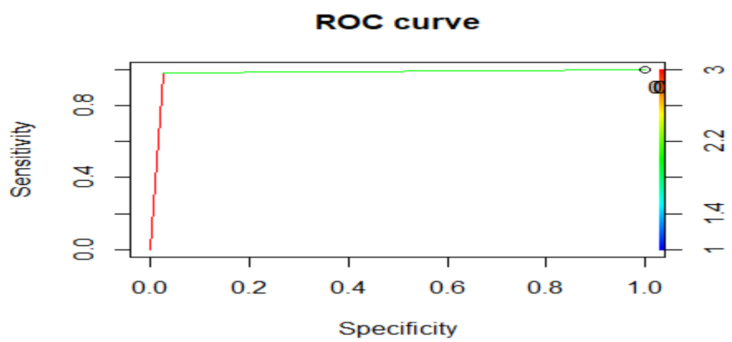
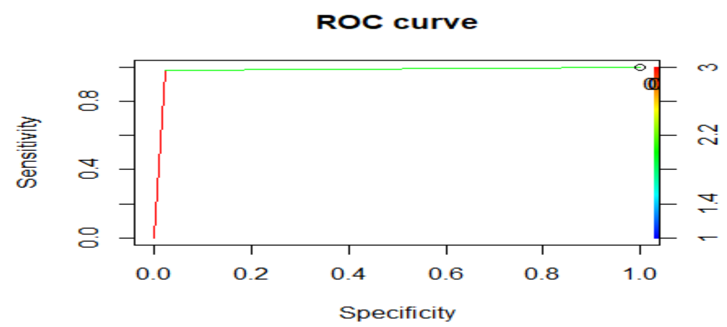
Table 5.8: Random Forest Evaluation Metrix (Training Set)

Based Random Forest Model					
	Accuracy (%)	**Sensitivity (%)**	**Specificity (%)**	**F1-score (%)**	**Confusion Matrix**
Original data	100	100	100	100	p1 0 1 0 3277 0 1 0 663
SMOTE	100	100	100	100	p1 0 1 0 3277 0 1 0 2654
Random over sampling	100	100	100	100	p1 0 1 0 3277 0 1 0 3722
Random under sampling	100	100	100	100	p1 0 1 0 398 0 1 0 441
Random both sampling	100	100	100	100	p1 0 1 0 1794 0 1 0 1705
Tuned Random Forest Model					
Original data	100	100	100	100	p1 0 1 0 3277 0 1 0 663
SMOTE	100	100	100	100	p1 0 1 0 3277 0 1 0 2654
Random over sampling	100	100	100	100	p1 0 1 0 3277 0 1 0 3722
Random under sampling	100	100	100	100	p1 0 1 0 398 0 1 0 441
Random both sampling	100	100	100	100	p1 0 1 0 1794 0 1 0 1705

Table 5.9: Random Forest Evaluation Metrix (Test Set)

Based Random Forest Model					
	Accuracy (%)	**Sensitivity (%)**	**Specificity (%)**	**F1-score (%)**	**Confusion Matrix**
Original data	96.57	**99.15**	83.86	97.96	p2 0 1 0 1393 46 1 12 239
SMOTE	97.84	97.30	98.51	98.02	p2 0 1 0 1367 17 1 38 1121
Random over sampling	**98.70**	97.22	**100**	**98.59**	p2 0 1 0 1366 0 1 39 1596
Random under sampling	91.97	87.13	96.32	91.13	p2 0 1 0 149 7 1 22 183
Random both sampling	97.73	97.14	98.36	97.78	p2 0 1 0 748 12 1 22 719
Tuned Random Forest Model					
Original data	97.16	**98.65**	89.82	98.30	p2 0 1 0 1386 29 1 19 256
SMOTE	97.99	97.86	98.15	98.18	p2 0 1 0 1375 21 1 30 1117
Random over sampling	**99.17**	98.22	**100**	**99.10**	p2 0 1 0 1380 0 1 25 1596
Random under sampling	92.8	87.72	97.37	92.02	p2 0 1 0 150 5 1 21 185
Random both sampling	97.87	97.01	98.77	97.90	p2 0 1 0 747 9 1 23 722

Table 5.10: ROC and AUC in random forest

<table>
<tr><th></th><th colspan="2">Based Random Forest model</th><th colspan="2">Tuned Random Forest model</th></tr>
<tr><th></th><th>AUC (%)</th><th>ROC</th><th>AUC</th><th>ROC</th></tr>
<tr><td>Original data</td><td>91.5</td><td>
</td><td>98.30</td><td>
</td></tr>
<tr><td>SMOTE</td><td>97.9</td><td>
</td><td>98.0</td><td>
</td></tr>
</table>

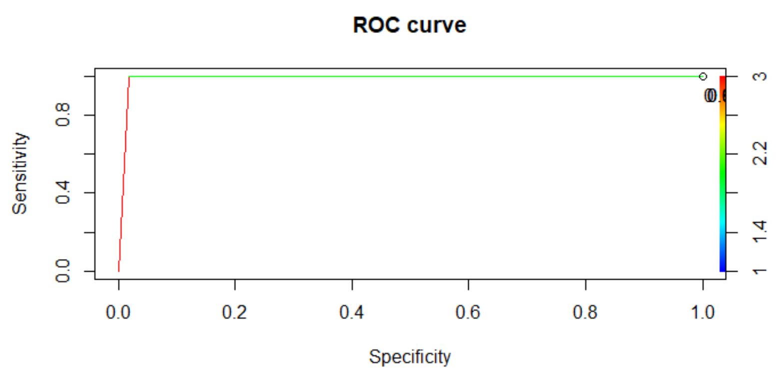
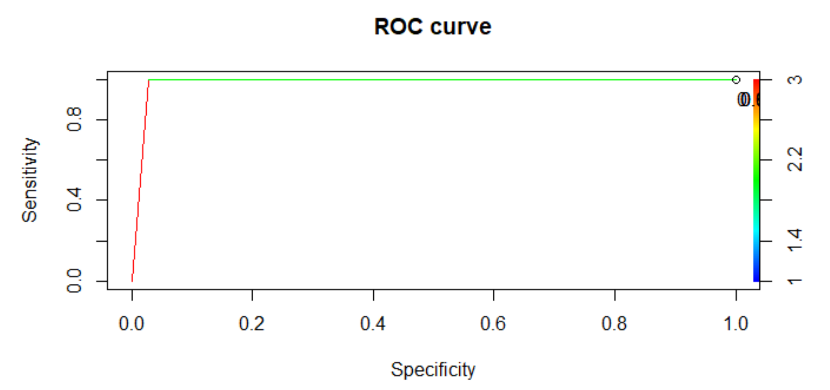
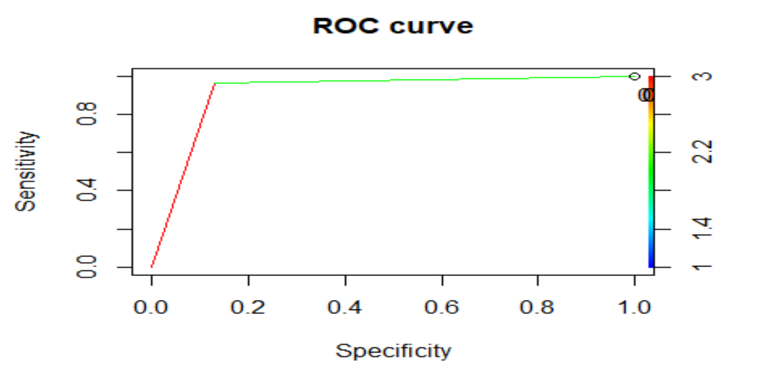
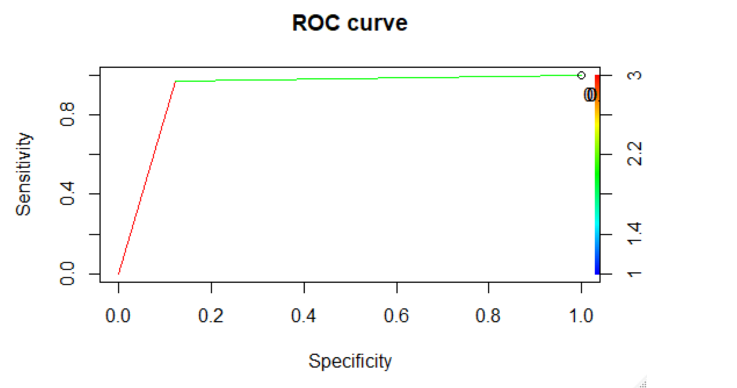
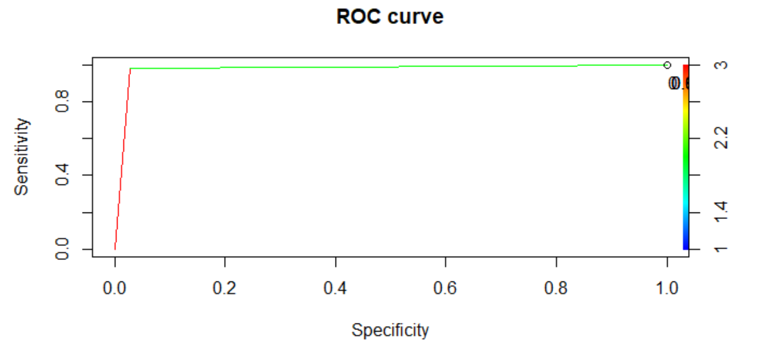
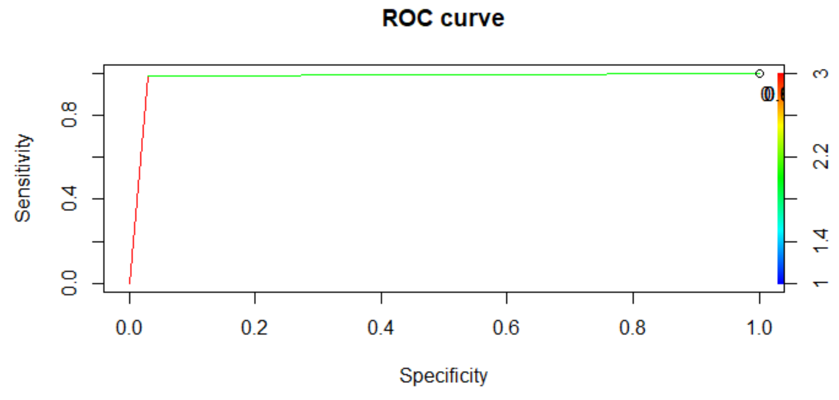
Random over sampling	**99.1**		**98.6**	
Random under sampling	91.7		92.5	
Random both sampling	97.8		97.9	

Table 5.11 Random Forest Fitness summary

Data sampling method	Fitness
Based Random Forest model	
Original dataset	Good
SMOTE	Good
random oversampling	Good
random undersampling	Good
random both sampling	Good
Tuned Random Forest model	
Original dataset	Good
SMOTE	Good
random oversampling	Good
random undersampling	Good
random both sampling	Good

Based on the table above, it shows that the based model of random forest is powerful to give 100% in accuracy, sensitivity, specificity and F1 score in all kinds of balanced and imbalance datasets in this experiment. By using the testing set, the random forest shows more than 90% of performance in mostly all evaluation parameter such as accuracy, sensitivity, specificity, F1-score and area under curve in all kind of dataset except the sensitivity of random under sampling (87.13%) and specificity (83.86%) of original dataset. In the random under sampling dataset, it is only less than 900 data available for training, in the dataset, only 398 is under positive class (47%) while 441 (53%) is under negative class. In this case, the positive class is 6% less than the negative class. It is not a significant percentage to give a big difference, but in the case of random under sampling dataset, since the dataset size is small, so the difference can be obviously seen from the result.

In the original data, it has highest percentage in sensitivity (99.15%) while lowest percentage in specificity (83.86%). This is because the imbalance data set has around 85% data in the positive class “0”. Hence, the models in this dataset have more positive class data to train their learning ability and give better results when compare to others balanced dataset which has only 45-50% positive class data. Hence, the result of models in original dataset able to give the highest accuracy in sensitivity because it is able to get higher true positive and less false negative value. However, the specificity of the original dataset is the lowest when compared to another model in test set. This is because the imbalance dataset only has around 15% class “1” data available for the algorithm to train. In term of number, it is only 663 data in the negative class is used for training, but there are more than 3200 data in the positive class is trained. The number of negative classes is five-fold lower than the positive class in the training set, although it shows 100% performance in the training set, but when come to the test set, the difference can be seen between imbalance dataset and balanced dataset.

In term of random forest model build on oversampling and both sampling, it gives excellent result by more than 97% in all parameter even before hyperparameter tuning. Among them, the based random over sampling outperforms other models with 98.70% accuracy, 97.22% sensitivity, 100% specificity, 98.59% F1-score and 99.1% area under curve. After hyperparameter tuning, all the performance of the models in different datasets are remained 100% in the training set and improved in the test set. The model that has most significant improvement is the model in the original dataset which has 6-7% improvement in area under curve (from 91.5% to 98.3%) and specificity (83.86% to 89.82%). This pointed that random forest model after hyperparameter tuning is able to aid in the performance in imbalanced dataset. In overall, both based random forest model and tuned random forest model has good fitness. There is no overfitting or underfitting issue happen because all the prediction on the training data is slightly higher than the prediction on the test data. Hence, it can be summarized that both based random forest model and tuned random forest model are performed well in all kind of dataset.

7.0 Discussion and recommendation

Table 7.1: Comparison between model

	Based SVM-RBF				Tuned SVM-RBF				Logistic Regression				Based Random Forest				Tuned Random Forest			
Training set																				
	Acc	**Sen**	**Spec**	**F1**	**Acc**	**Sen**	**Spec**	**F1**	**Acc**	**Sen**	**Spec**	**F1**	**Acc**	**Sen**	**Spec**	**F1**	**Acc**	**Sen**	**Spec**	**F1**
Original	93.86	99.30	66.97	96.41	99.7	99.97	98.34	99.82	89.29	90.82	77.20	93.77	100	100	100	100	100	100	100	100
SMOTE	95.97	94.87	97.32	96.30	96.88	95.82	98.19	97.14	82.50	84.23	80.18	81.67	100	100	100	100	100	100	100	100
Random oversampling	96.27	98.95	93.23	96.58	98.53	99.95	96.92	98.63	81.81	81.63	81.96	80.25	100	100	100	100	100	100	100	100
Random under sampling	91.66	94.26	88.60	92.42	92.01	94.48	89.12	92.74	85.37	76.78	87.97	70.90	100	100	100	100	100	100	100	100
Random both sampling	95.46	93.76	97.24	95.48	100	100	100	100	80.68	80.88	80.46	81.24	100	100	100	100	100	100	100	100
Test set																				
Original	91.18	98.43	55.44	94.89	97.28	98.86	89.47	98.37	88.99	91.21	73.46	93.55	96.57	99.15	83.86	97.96	97.16	98.65	89.82	98.30
SMOTE	93.51	92.67	94.55	94.04	94.18	93.02	95.61	96.64	82.42	85.12	79.30	83.86	97.84	97.30	98.51	98.02	97.99	97.86	98.15	98.18
Random oversampling	95.13	98.68	91.10	95.57	97.67	99.94	95.09	97.85	80.27	80.77	79.88	78.28	98.70	97.22	100	98.59	99.17	98.22	100	99.10
Random under sampling	85.87	90.77	80.12	87.40	87.26	92.31	81.33	88.67	82.35	71.26	85.53	64.25	91.97	87.13	96.32	91.13	92.8	87.72	97.37	92.02
Random both sampling	92.21	90.78	93.71	92.28	98.13	97.40	98.91	97.83	81.15	82.86	79.47	81.27	97.73	97.14	98.36	97.78	97.87	97.01	98.77	97.90

Table 7.2: Comparison between AUC value

	Based SVM-RBF	Tuned SVM-RBF	Logistic Regression	Based Random Forest	Tuned Random Forest
Original	76.9	94.2	89.0	91.5	98.30
SMOTE	93.6	94.3	90.0	97.9	98.0
Random oversampling	94.9	97.5	88.8	99.1	98.6
Random undersampling	85.4	86.8	88.2	91.7	92.5
Random both sampling	92.2	98.2	89.3	97.8	97.9

Table 7.3: Comparison between own model with related works

Reference	Dataset Size (Rows x col)	Best ML algorithms	EDA	Data pre-processing	Confusion matrix	Feature engineering	Data Balancing	K-Fold Cross-Validation	Hyperparameter tuning	At least 3 evaluation parameters	Accuracy (%)
(Lemos et al., 2022)	500,000 x 35	Random forest	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	82.8
(Miao & Wang, 2022)	1000 x 21	Random forest	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	96.10
(Wu et al. 2021)	**Dataset 1:** 7032 x 21 **Dataset 2:** 4031 x 20 **Dataset 3**: 51047 x 58	**Dataset 1:** Adaboost **Dataset 2:** RF **Dataset 3**: RF	Yes	Yes	No	Yes	Yes	Yes	No	Yes	**Dataset 1**: 77.19 **Dataset 2:** 93.60 **Dataset 3:** 63.09
(Xiahou & Harada, 2021)	987994 x17	Support vector machine	No	Yes	No	No	Yes	Yes	No	Yes	~91

(Kaur & Kaur, 2020)	28382 x 21	Random Forest	Yes	Yes	No	Yes	Yes	Yes	No	Yes	~85.0
(Wadikar, 2020)	96967 x 48	Random Forest	Yes	Yes	Yes	Yes	Yes	No	No	Yes	97
(Bhattarai et al., 2019)	3333 x 20	XGBoost	Yes	Yes	No	Yes	No	No	No	Yes	95.5
(He et al., 2020)	25275 x 253	Extra Tree Classifier and Gradient Boost	Yes	Yes	Yes	Yes	No	Yes	Yes	No	**AUC (%)** GBM: 61 Extra Tree Classifier: 68
(Asthana, 2018)	(5000 x 20)	SVM-Poly with Adaboost	No	No	Yes	No	No	Yes	Yes	Yes	96.85
(Ismail et al., 2019)	(7043 x 21)	Logistic Regression	Yes	Yes	Yes	Yes	No	No	No	Yes	100
Own work	(5630 x 20)	Tuned Random Forest	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Original: 97.16 SMOTE: 97.99 Random oversampling:99.17

											Random undersampling:92.8 Random both sampling: 97.87

In this project, there are total 5 models are built which include the based SVM-RBF model, tuned SVM-RBF model, logistic regression, based random forest, and tuned random forest model. Based on the table 7.1, it is clearly show that in term of the prediction on training set, the performance of based random forest and tuned random forest significantly outperformed other models with 100% accuracy, sensitivity, specificity and F1-score in all kind of datasets which include the imbalanced dataset, SMOTE resampling dataset, random oversampling dataset, random under sampling dataset and the random both sampling dataset. In term of prediction on the test set, tuned-random forest model is the best models while based random forest is the second top by achieved more than 90% in all the performance under original data and different resampling method. The result is aligned with the literature review that was previously done which pointed out that the random forest is the best machine learning approach that can be used to predict customer churn. However, this study result is more reliable because the previous study the pointed random forest is the best machine learning approach did not carry out hyperparameter tuning, cross-validation, refer the confusion matrix or do data balancing. In the study of this project, all the steps are completed to determine the optimal approach.

Tuned random forest model provides excellent result in the original and other resampling dataset. Among the various resampling method, random forest achieved highest test performance in the random oversampling data with 99.17% accuracy, 98.22% sensitivity,100% specificity, 99.10% F1-measure and 98.6% AUC. The second high performance resampling dataset under the tuned random forest is SMOTE oversampling data with 97.99% accuracy, 97.86% sensitivity, 98.15% specificity, 98.18% F1-measure and 98% AUC. Follow by the third high performance which is random both sampling which achieve 97.87% accuracy, 97.01% sensitivity, 98.77% specificity, 97.90% F1 measure and 97.9% AUC. Although there is imbalance distribution in the original data, but the tuned random forest model still able to handle it with high performance which is 97.16% accuracy, 98.65% sensitivity, 89.82 specificity, 98.30% F1 measure and 98.30% AUC. However, the top three highest performance dataset under random forest are the data after resampling method, this bring out the information that there is still a need to resampling the data before model building in order to get a better performance. Although random forest has better ability to handle the imbalance data, but the challenge of imbalance data cannot be ignored because when comparing with other resampling dataset, the specificity of the original dataset under random forest is the lowest, this indicate that the imbalance data still affect the result and data resampling before model building is recommended.

Random forest that has better imbalance data handling can be prove by the result of based random forest which has the highest performance in the original dataset before hyperparameter tuning with 96.57% accuracy, 99.15% sensitivity, 83.86% specificity, 97.96% F1-measure and 91.5% AUC. The most obvious parameter that can emphasize this point is the specificity. As the original dataset has just around 15% class negative which is class “1” data, hence it is difficult for model to achieve high performance in specificity because there is more false result will be generated due to the inadequate of information for model training. This can be observed from the result of logistic regression and based support vector machine which has only 73.46% and 55.44% specificity respectively. However, in this case, the random forest able to achieve 83.36% specificity when using the same dataset before hyperparameter tuning. This around 10% - 25% better performance which is consider a lot.

In addition, in term of under sampling data, since the dataset is under sampling into only 1200 observations in total, the small sample size of the data did affect the performance of the models. For example, the logistic regression has only 82.35% accuracy, 71.26% sensitivity, 85.53% specificity, 64.25% sensitivity and 88.2% AUC. However, the tuned random forest still outperforms other models in this issue with 92.8% accuracy, 87.72% sensitivity, 97.37% specificity, 92.02% F1 measure and 92.5% AUC. Hence, after comparing this model with previous literature review and also other models it was built, it determines that the tuned random forest is the best machine learning model that can predict e-commerce customer churn across various resampling method.

8.0 Conclusion

Based on the experiments in this study, the best machine learning model that can be used to predict e-commerce customer churn across various resampling method is tuned random forest model. Random forest has higher performance and better ability to handle various kind of resampling data and also imbalanced data when compared to other models in this study. Although random forest can better handle the imbalance data than other models, but when compared between the balanced and imbalanced dataset, the imbalanced dataset still has lower performance. So, there is still a need of data resampling before model building to improve performance. In the case of customer churn, the minority group which is the churn customer is the main target that a company aim to spot. In this context, a high specificity is important for the models to perform. In term of resampling method, random forest model that built on random oversampling dataset is outperform in this project by giving 100% specificity which is the most important parameter in the customer churn detection context. Last but not least, the accuracy of this models is more reliable than previous literature review because this study carries out hyperparameter tuning, cross validation, and data balancing which is the steps that previous studies miss out. In conclusion, the objective and the main aim of this study is successfully achieved.

9.0 Future Recommendation

In future, research which investigate more advance machine learning such as Artificial Neural Network, XGboost, and essemble machine learning model can be proposed to study on different sampling datasets. In addition, more resampling method can be explored to examine the performance of the machine learning. Last but not least, obtained a larger dataset is suggested to get more information to train the machine learning approach.

Reference

Asthana. (2018). A comparison of machine learning techniques for customer churn prediction. *International Journal of Pure and Applied Mathematics. 119(10), 1149-1169.* https://www.semanticscholar.org/paper/A-comparison-of-machine-learning-techniques-for-Asthana/e4bacc5ebc276da59f1816b5eaec0dc2f2da10ac

Ahmad, A.K., Jafar, A. & Aljoumaa, K. (2019). Customer churn prediction in telecom using machine learning in big data platform. *Journal of Big Data*. 6(28),1-24. https://doi.org/10.1186/s40537-019-0191-6

Avcontentteam. (2016, March 28). *Practical Guide to deal with Imbalanced Classification Problems in R.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/

Brownlee, J. (2020, December 23). *A Gentle Introduction to Imbalanced Classification.* Machine Learning Mastery. https://machinelearningmastery.com/what-is-imbalanced-classification/

Burn, E. (March, 2021). *Machine Learning. Tech Target.* https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values.

Bhattara, A., Shrestha, E., Sapkota, R.P. (2019). Customer Churn Prediction for Imbalanced Class Distribution of Data in Business Sector. *Journal of Advanced College of Engineering and Management.* https://doi.org/10.3126/jacem.v5i0.26693

Celik, O. & Osmanoglu, E.O. (2019). Comparing to Techniques Used in Customer Churn Analysis. *Journal of Multidisciplinary Developments.* 4(1), 30-38. https://www.researchgate.net/publication/337103029

Chai, W., Holak, B. & Cole, B. (2020). *e-commerce.* https://www.techtarget.com/searchcio/definition/e-commerce

GreeksforGreeks. (2021, September 27). *Introduction to Resampling methods.* https://www.geeksforgeeks.org/introduction-to-resampling-methods/

He, Y., Xiong, Y. & Tsai, Y. (2020). Machine Learning Based Approaches to Predict Customer Churn for an Insurance Company. *IEEE Xplorer.* https://doi.org/10.1109/SIEDS49339.2020.9106691

Ismail, A.S., Kama, M.N., & Mohammad, N,I. (2019). Customer Churn Prediction In Telecommunication Industry Using Machine Learning Classifiers. *2019 Association for*

Computing Machinery. https://doi.org/10.1145/3387168.3387219

Miao, X. & Wang, H. (2022). Customer Churn Prediction on Credit Card Services using Random Forest Method. *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development.* https://dx.doi.org/10.2991/aebmr.k.220307.104

Mishra, K. & Rani, D.R. (2017). Churn Prediction in Telecommunication using Machine Learning. *International Conference on Energy, Communication, Data Analytics and Soft Computing.* https://doi.org/10.1109/ICECDS.2017.8389853

Kamalraj, N & Malathi, A. (2013). A Survey on Churn Prediction Techniques in Communication Sector E-mail. *International Journal of Computer Applications*. 64(5),39-42 doi: 10.5120/10633-5373

Osmanoglu, O. O. (2019). Comparing to Techniques Used in Customer Churn Analysis. *Journal of Multidisciplinary Developments*, 4(1), 30–38. https://www.researchgate.net/publication/337103029

Patil, P. (2018, March 24). *What is Exploratory Data Analysis*? Towards Data Science. https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15#:~:text=Exploratory%20Data%20Analysis%20refers%20to,summary%20statistics%20and%20graphical%20representations.

Rahman1, H.A.A., Wah, Y.B., & Huat, O.S. (2021). Predictive Performance of Logistic Regression for Imbalanced Data with Categorical Covariate. *Pertanika.* https://doi.org/10.47836/pjst.29.1.10

Soumi, D., Prabu, P. & Paulose, J. (2021). Effective ML Techniques to Predict Customer Churn. *Third International Conference on Inventive Research in Computing Applications (ICIRCA).* https://doi.org/10.1109/ICIRCA51532.2021.9544785

Vera, A.P. (2020, May 25). *Dealing with Imbalanced Data in Churn Analysis*. Towards Data Science. https://towardsdatascience.com/dealing-with-imbalanced-data-in-churn-analysis-6ea1afba8b5e

Wu, S., Yau, W., Ong, T., Chong, S.C. (2021). Integrated Churn Prediction and Customer Segmentation Framework for Telco Business. *IEEEAccess.* https://doi.org/10.1109/ACCESS.2021.3073776

Xiahou, X. & Harada, Y. (2022). Customer Churn Prediction Using AdaBoost Classifier and BP Neural Network Techniques in the E-Commerce Industry. *American Journal of Industrial and Business Management*, 2022, 12, 277-293. https://doi.org/10.4236/ajibm.2022.123015

Yijit, I.O., & Shourabizadeh, H. (2017). An Approach for Predicting Employee Churn by Using Data Mining. *IEEE Xplorer.* https://doi.org/10.1109/IDAP.2017.8090324

Zhang, T., Moro, S., & Ramos, R.F. (2022). A Data-Driven Approach to Improve Customer Churn

Prediction Based on Telecom Customer Segmentation. *Future internet.* https://doi.org/10.3390/fi14030094