Queen Mary
University of London

Assessment Title: BUSM130 Group Project

Student ID: 200859325

Client: Brain Bravo Limited

Project Title: "Credit card application assessment using supervised learning algorithms"

Table of Contents

1. **Introduction.** 4
2. **Project Description.** 5
3. **Literature Review.** 7
   - 3.1. **Background** 7
   - 3.2. **Definitions of Credit scoring models (CSMs)** 7
   - 3.3. **Demerit and merit of Credit scoring models (CSMs)** 8
   - 3.4. **Classification Techniques** 8
   - 3.5. **Regression Model** 8
   - 3.6. **Decision trees** 9
   - 3.7. **Random Forest** 9
   - 3.8. **Summary** 10
4. **Data Pre-Processing for Brain Bravo Dataset.** 10
   - 4.1. **Variable definition** 11
   - 4.2. **Data Pre-Processing** 14
     - 4.2.1. **Data Manipulation (Handling missing values)** 15
     - 4.2.2. **Data Formatting** 16
     - 4.2.3. **Binary Encoding** 17
   - 4.3. **Feature Extraction** 21
   - 4.4. **Splitting of Dataset** 23
5. **Exploratory Data Analysis.** 23
6. **Methodology (Machine learning algorithms).** 38

Confusion / Classification Matrix 38

- 6.1. **Logistic Regression** 40
  - 6.1.1. **Background** 40
  - 6.1.2. **First Iteration** 42
  - 6.1.3. **Second Iteration** 42
- 6.2. **Linear Regression** 43
  - 6.2.1. **Background** 43
  - 6.2.2. **Project** 43
  - 6.2.3. **Dependent variable** 44
  - 6.2.4. **Independent variable** 44
  - 6.2.5. **Formulation** 44
  - 6.2.6. **Each of the steps** 45
- 6.3. **Decision Tree classifier (Classification And Regression Trees)** 47
  - 6.3.1. **Background** 47

6.3.2. Project 47
6.3.3. Dependent variable 47
6.3.4. Independent variable 47
6.3.5. Formulation 47
6.3.6. Each of the steps 48
6.4. Random Forest 52
6.4.1. Background 52
6.4.2. First Iteration (general approach) 54
6.4.3. Second Iteration 55
6.4.4. Third Iteration (Parameter tuning) 55
7. Performance metrics. 56
8. Results 60
8.1. Application Scoring 60
8.1.1. Logistic Regression result 61
8.1.2 Linear regression result 63
8.1.3 Classification and Regression Trees Result 65
8.1.4 Random Forest result 67
8.2 Behavioural scoring part 68
8.2.1 Logistic regression result 68
8.2.2 Linear regression result 71
8.2.3 Classification And Regression Trees Result 71
8.2.4 Random Forest result 72
8.3 The Result Table 74
9. Limitation. 75
10. Conclusion. 76
11. Appendix 77
12. References 164

1. Introduction.

Nowadays, with the update of payment methods, more and more people begin to apply for credit card and use it to pay the bill. At the same time, due to the upsurge in credit card applications, a lot of banks have begun to give more attention to whether people can reply to their bills on time. If customers cannot repay within a limited time, this will have a great impact on the bank's profit. To be able to solve this problem, Brain Bravo, located in London, provides services that can help banks predict credit card risks. In this service, Brain Bravo would provide banks with some professional analytical services. By using Logistic Regression, Decision Tree, Neural Network algorithms and Zero-one Classification, Bucket Prediction and Profit/Loss Prediction models, they can help banks predict and avoid credit risks, to help banks solve daily problems in an unpredictable competitive environment. The forecast of the credit card risk of the company can help the bank to maximize the benefits. In this process, they will use two aspects to predict the credit card risk of different groups of people, such as application scoring, and behaviour scoring. In the first part, they will predict the future bucket of new credit card customers based on the past data. In the second part, they will use past data to predict the future performance of existing customers and the bucket in the second year. Through the analysis of these two parts, it will be able to better help the bank to predict and avoid credit card risks, then maximize the bank's profit. (Brain bravo, Ltd)

In this paper, we will use the credit scoring model, which can help us evaluate the customer's credit risk situation. The credit scoring model uses some important statistical models and algorithms to help banks or financial institutions analyse and predict the credit situation of their customers. The main purpose of the credit scoring model is to be able to give a credit rating to the customer's credit status based on some important factors, such as age, family status, asset status, education level, income status, and real estate status. When the customer's comprehensive index can reach a certain threshold, we classify the customer as good. On the contrary, if they cannot get that threshold, we would classify the customer as bad. In this paper, we will use linear regression, logistic regression, decision tree and random forest methods to help us evaluate the credit situation of the future customer. These algorithms are commonly used in the credit scoring model. Through these methods, we can predict the customer's future credit situation more accurately, and it can also help the bank or financial institutions to predict the credit risk situation of their customers in the future, to help them avoid possible credit risk crises.

Logistic regression, Linear regression， CART (Classification and Regression Trees), and random forest algorithms can obtain accurate data when predicting the possibility of different things. For example, when a hospital is studying the possibility of a certain disease in different populations, these three methods will also need to be used. For analysis, because they can estimate the possibility of a certain thing very well, and they can clearly filter out the factors that affect the different events. In the

meantime, these algorithms can help us predict the probability of customers' buckets happening in the future. Although these algorithms can accurately predict the future bucket of customers, the accuracy of each model is different. In this paper, we will explore the accuracy of each algorithm for model prediction, choose the best one to conclude. The results of each algorithm would be described in detail in the methodology section.

The purpose of this report is to discuss the use of supervised learning algorithms to evaluate credit card risk. Discussing the meaning of the results obtained by these models and analyse the potential impact and risks that these results bring to the bank. The structure of this report is as follows:

- First, state the project description and literature review.
- Then, discuss methodology and analyze the results we got.
- Later, explain the relevant algorithms, evaluate the model we used, and state the limitations of the project.

2. Project Description.

The aim of this project is to predict the creditworthiness and default risk. Default rates is one of the main concerns relate to the Credit Scoring Models' (CSMs) which have been widely used in the financial organisations. To improve the decision-making process and enhance risk management, they identify each new customer who can be accepted or rejected by the model. As for the existing clients, the system conducts further analysis. Organisations can gain their financial benefits by virtue of this evaluation process, which is the main goal that this project must achieve.

In some cases, the problem of build-up CSMs is lack of the personal data in dataset (Fernandez Vidal and Barbon, 2019). Hence, the financial organizations tend to utilize an earlier client data for prediction. For example: 20-years-old credit scoring dataset in this project. The organizations believe that the past result of prediction in customer credit behavior is similar to the result of the future (Abdou and Pointon, 2011; Fernandez Vidal and Barbon, 2019).

Meanwhile, CSMs are not solely for the evaluation of the benefits or losses for the banking industry, but also can identify the customer's lifecycle (ibid). Each cycle is indicated by different scoring types (Figure 1). Thus, application scoring, and behavioural scoring are the two-cycle stages that will be discussed in detail in this project. Application scoring is for the new client, who apply for the credit card, the scoring model can analyse and predict their future credit. As for the behavioural scoring, it is the analysis and prediction of the data for the existing customers' of repayment ability from application start date up to current date.

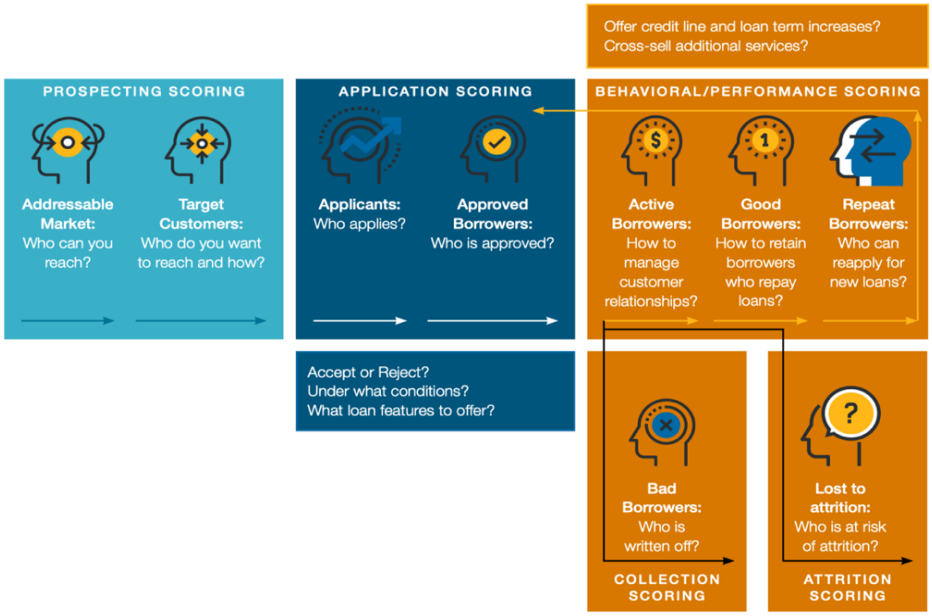

Figure 1: The borrowing process customer lifecycle.

Source: CREDIT SCORING IN FINANCIAL. [online] **Available at:** https://www.cgap.org/sites/default/files/publications/2019_07_Technical_Guide_CreditScore.pdf

Furthermore, the 20-years-ago dataset with 11,329 observations is provided by Brain bravo Ltd. It is necessary to use classification methods to distinguish customer's credit into two categories such as "good" and "bad" in this task. The required statistical techniques include linear regression, logistic regression, CART decision tree classifier and random forest will be employed in this report.

Moreover, the probability of default is based on personal historical data. For the application scoring, the evaluation relies on the information of the applicators, for example, age, agenda, branch location, occupation and etc. As for the behavioural scoring, the explanatory variables are based on personal credit histories such as credit records, default data, backlist record and average balance. Additionally, the definition of the "bucket" is used to classify client's repayment behavioural. The buckets are divided into 0 to 7. The largest number of buckets means the longest period of deferral payment in the dataset. For instance, the number of 7 buckets means the payment deferred at least 7 months, they will be referring as "bad" credit customers. As for the number of 0 buckets, the payment is deferred less than one month, indicating these clients have "good" credit and can make profit for the organizations.

The three subtasks: zero-one classification, bucket prediction and profit and loss prediction will be analysed in the project. The dependent variable, *all max buckets*, code as 0 and 1 in zero-one classification subtask; and order values from 0 to 7 in other two subtasks. Developing models on these three subtasks is to fit four proposed methods, linear regression, logistic regression, decision tree with CART and random forest. Then the confusion matrix is produced/created to compare the model performance based on those four methods by the accuracy of the credit decisions, sensitivity, and specificity.

3. Literature Review.

3.1. Background

"Credit scoring models are widely used to evaluate business, real estate, and consumer loans" (Gup and Kolari, 2005, p. 508). Approximately 97% of banks choose credit scoring to implement their credit card applications (ibid). From enterprise perspective, these techniques decide who will get credit, how much credit they should get, and what operational strategies will enhance the profitability of the borrower to the lenders (Abdou and Pointon, 2011; Samreen and Zaidi, 2012). Logistic Regression, Neural Networks, Decision Trees, Nearest Neighbour Classifiers and Support Vector Machines (SVM) are the commonly used classification methods in recent days for credit scoring models that in the study of Liberati and Camillo (2018) and Ala'raj and Abbod (2016) paper.

3.2. Definitions of Credit scoring models (CSMs)

Before the classification techniques section, the definition of credit scoring will be presented first. According to Anderson (2007), credit scoring should be discussed separately by individual term. The term "credit" is "buy now and pay later". As for the "scoring", it means "grade" or "number". Thus, the meaning of "credit scoring" is a credit decision process of reducing the risk of clients being default via statistical models (Abdou and Pointon, 2011). CSMs process is involved in collecting personal data, transforming data into numerical measures, setting up a criterion for a loan, and analysing the variables to acquire useful decision-making information (Abdou and Pointon, 2011). More precisely, "credit-scoring is used by banks and financial organisations to predict default, make loan decisions, and accordingly estimate the probability of default (PD) and exposure at default (EAD) (BCBS, 2010)." In terms of the definition of credit scoring models, studies of Kaseke and Murairwa (2019) provide that the goal of CSMs is to avoid type I and type II which means rejection of good clients and the acceptance of the bad customers by analysing historical data. Consistent with Abdou and Pointon (2011), the two studies suggest that CSMs is to "avoid large losses (p.8)."

The definition of application scoring, and behavioural scoring will be presented. For the first scoring, it is a score is used to give a decision on new credit application (Liu, 2001). As regards behavioural scoring, this scoring is not merely involved with the personal information but also the behaviour of the client's credit payment and purchasing history. Financial organisations use this static and dynamic process to expand or refuse the client's credibility and loans according to their credit performance in the past (Li and Zhong, 2012).

3.3. Demerit and merit of Credit scoring models (CSMs)

Some studies have summarised the advantages of credit scoring models and explained why these models have become a fashionable tool. This section will focus on the advantages and disadvantages in models and organisations perspectives

For the advantages of credit scoring models, Al Amari (2002), Chandler and Coffman (1979) state that credit scoring models can reduce the misclassification cost, lower the error, minimise the total processing time, support decision-making with less information requirement, model with practical data, and cut off values varies by different scenarios. Crook (1996) gave a further analysis that the credit scoring data with the same weights can be examined in different statistical models, which is easy to conduct. Besides, these models can be applied to a huge dataset. Furthermore, the priority tasks for the bankers are to control credit risk well and costs of credit risk processing. Therefore, some studies indicated that the credit scoring model can help reducing costs of credit products, time, and risk. (Lee et al, 2002; Ong et al, 2005). More importantly, the credit scoring system improves the risk controlling procedure and the accuracy of credit analysis in the operation, and it leads to be more efficient and effective in risk controlling and to enhance the marketing strategy to snowball the sales volume in financial products (ibid). By establishing a stable and robust credit scoring system, financial organisations and banks may achieve a top competitive position in the market

On the contrary, the demerit of the credit scoring models, there are three pieces of advice that studies have engaged to this model. Firstly, a client's data should be updated regularly; otherwise, the system might misidentify the creditworthy applicators (Abdou and Pointon, 2011; Al Amari, 2002; Chandler and Coffman, 1979). Secondly, the economic factors should be included in models (Abdou and Pointon, 2011). Eventually, credit scoring models are costly in purchasing and processing (ibid).

3.4. Classification Techniques

To find the best performance model, analyse the dataset using the following four classifiers: linear regression, logistic regression, decision trees and random forest.

3.5. Regression Model

For the regression models, there are two commonly used statistical techniques, linear and logistic regression.

According to Karimi (2014), using a linear regression model to examine the credit scoring could find the less creditworthy customers, reducing the misclassification cost.

Logistic regression is one of the more prominent statistical models employed in credit scoring. (Ala'raj and Abbod, 201; Deloitte, 2016; Lessmann et al., 2015). When utilizing logistic regression, the dependent variable can be classified into binary results, "good" and "bad", which means a good loan (0) or bad loan (1) (Abdou and Pointon, 2011). Logistic regression is designed for a linear relationship between predictor and log odds (Lee & Chen, 2005). If it is a non-linear relationship, the accuracy will be reduced (Ong et al., 2005).

Compared with linear regression, simple logistic regression can extend to two or more explanatory variables. The more variables in the model, the more complex processing it could be. Thus, Freund and William (1998) suggested the maximum likelihood method can solve this issue of parameter estimation.

On the other hand, the assumptions and parametric distribution of response are the main difference between linear and logistic regression models (Hosmer & Lemeshow, 1989). More studies suggested that logistic regression is appropriate for credit scoring applications (Abdou, et al., 2008; Crook et al, 2007; Baesens et al, 2003). Furthermore, compared with the linear regression, Hand and Henley (1997) stated that the logistic regression is the better statistical tool to describe the results of dichotomous credit.

3.6. Decision trees

Decision trees, also known as recursive partitioning, is another popular model used for credit scoring. Li and Zhong (2012) illustrated the decision trees provide the most accurate outcome of credit risk. According to Bensic, Sarlija and Zekic-Susac (2005), decision trees offer better result compared with other parametric models such as regressions when based on the same dataset. With its simple procedure and easily understandable dendrogram, decision trees have become a standard method in the credit scoring model (Ala'raj and Abbod, 2016; Breiman et al, 1984; Lee et al., 2006; ZekicSusac et al., 2004). CART is an example of the Decision tress that people use. Moreover, decision trees are suitable for qualitative data without strict statistical requirements.

3.7. Random Forest

Random Forest is a combination of several decision trees. The most evident difference between decision tree and random forest is the final classifier is based on a voting process. This voting procedure can improve the accuracy of the outcome compared with other classification

techniques, which is one of the beneficial aspects. There are other merit factors on random forest. Miner, Nisbet and Elder (2009) have provided their aspects. The random forest can handle incomplete data and various-sized data and variables. Moreover, random forest can be executed effectively and efficiently. The only disadvantage is that it is hard to handle large amounts of irrelevant attributes (Grosan & Abraham, 2011).

3.8. Summary

To sum up, credit scoring models have become a frequently used tool in various industries, especially financial organizations. From a business perspective, with the increased use of credit products, financial institutions must improve their monitoring system and enforce tighter controls in order to remain profitable for the long term. Therefore, CSMs for analysing client's past and latest credit data can assist companies in growing their cash and lowering the probability of damages to enhance business credit risk strategies. Furthermore, from a technical perspective, efficiency and effectiveness on processing, budget, misclassification, and risk are the main reasons why this model has gotten its popularity. More importantly, accuracy, size of the dataset, misclassification cost, missing data, simple procedure, and clear outcomes are the improvements that linear regression, logistic regression, decision trees and random forest can make to help researchers analyse the model and produce accurate results.

4. Data Pre-Processing for Brain Bravo Dataset.

The dataset is a collection of variables and observations. The credit card dataset provided by Brain Bravo Ltd has 11329 observations or called as records and 32 variables or known as fields. Each observation / row / record in the dataset are the details of each customer. Each variable / column / field in the dataset represents a different category where the customer information is being filled or the values are entered based on the customer's behaviour. In Data pre-processing section we will be explaining what each variable means, their data type and the process that we followed to clean the discrepancies in the data set.

4.1. Variable definition

There are 32 variables or fields in the dataset provided by Brain Bravo Limited and it is important for us to know what each field / variable mean. This section focuses on the explanation of the variables, their data type with some examples of data present in each variable.

ACC-AQCCOUNT-NO is a variable containing a customer account number, it is a 16-digit number and that is in the Integer format.

NO-OF-CARDS: No of cards variable tells us how many cards does the customer hold from the Brain Bravo Limited bank. That may be credit card or debit card or both. This field shows if the customer owns an additional card from the bank or no. The values in the variable are in the integer value.

MEMBER-SINCE: This variable defines the month and year the customer creates the account with Brain Bravo Ltd, it is represented in the format 'yyyymm' (E.g.: 200111) where the first four digits represented as 'yyyy' is years (E.g.: 2001) and 'mm' stands for months. The last two digits are the months and are represented in terms of numbers varying from 01 to 12 starting from January to December. (E.g.:11) 11th month in the calendar is November and it is an integer value**.**

BIRTH-DATE: contains the date of birth of the customer. The value given in the BIRTH-DATE is in format "yyyymmdd" (Eg: 19520711), The first four digits represent the year and are in format "yyyy" (E.g.: 1952), 5th and 6th digit in a number represents the month "mm" (E.g.: 07). 7th and 8th digit in the example represents date "dd" (E.g.: 11). The value in this field is in integer format.

SEX: The sex field contains detail of customer gender written in M or F where (M) is written for Male and (F) is written if the customer is Female, this field is in character format.

OCCUPATION-CODE: This field represents the customer's occupation code at different jobs. There are 17 occupation codes recognized by Brain Bravo Limited and based on the customer's job title the numbers are being assigned to the customer based on the job title mentioned on the application that the customers have submitted to the bank while opening the bank account. The null value (0) is assigned for the customers, who do not have an occupation, or the field was left blank on the application, it is in integer format.

BLACK-LIST-CODE: There are 7 blacklist codes that are present in the BLACK-LIST-CODE field and the values are assigned to the customers based on the customer's behaviour towards credit payment at Brain Bravo Limited, the value '0' is assigned to the customers whose transaction and the repayment of the credit amount issued from Brain Bravo Limited has been done on or before the deadline and the good customers**.**

ATTRITION-REASON: The loss of customers by the bank due to various reasons mentioned either by the customers or by the bank. There are 15 attrition reason codes present and are assigned to the respective customers. The code 0 is assigned to customers who have active bank account with the Brain Bravo Limited. The datatype is an integer.

BLACK-LIST-DATE: This field gives us the details on which date the customer was blacklisted by the Brain Bravo Limited bank. It is an integer value and is expressed as "20010626". The first four digits of example is year "yyyy" (2001). The next two digits after year is the value of month "mm" (10 is October) and the last two numbers are the value of date "dd" (26^{th}).

WRITE-OFF-DATE: A write-off date is an accounting action that reduces the value of an asset while simultaneously debiting a liabilities account. It is primarily used in its most literal sense by businesses seeking to account for unpaid loan obligations, unpaid receivables, or losses on stored inventory.

SPENDING-LIMIT: The variable defines that the customer can spend using his / her credit card on their necessities. The customer cannot accept the spending limit set by the bank as this limit is set by the bank on each customer account based on considering few parameters. The values are integer datatype.

AVG-PAYMENTS: The average amount that the customer has made payment to the bank on the credit amount issued by them. It is calculated by the formula: [Average amount / Time]. Time is the number of months they have taken to repay the credit amount that they had taken from the bank. The datatype is an integer.

AVG-BALANCES: The Average balance field consists of the average sum of the balance that the customer possessed in his account from the time the customer opened his bank account at the Brain Bravo Bank. The datatype is an integer.

AT-ANNIV-DATE: This field is an anniversary date that defines the subscription date of the customers with the bank. The value of the anniversary date is given in format "yyyymmdd" (Eg: 20011015). The first four digits represents year and is in format "yyyy" (E.g.: 2001), 5^{th} and 6^{th} digit in a number represents the month "mm" (E.g.: 10). 7^{th} and 8^{th} digit in the example represents date "dd" (E.g.: 15). The value in this field is an integer format.

LAST-TRX-DATE: This field tells us the date of last transition that the customer had with his account. It is an integer value and is expressed as "19991021". The first four digits of example is tear "yyyy" (1999). The next two digits after year is the value of month "mm" (10 is October) and the last two number are the value of date "dd" (21^{st}).

BUCKET: Buckets are the scoring given to the customer between 0 to 7 based on the customer's application and the values / entry that they have filled or mentioned on their application while applying

for credit card. Based on the BUCKET score the customer is either granted the credit card or rejects the request of issuing the credit card from Brain Bravo Limited. The score 0, 1 are considered to the good customers as the customers have a good track record and might be profitable to the bank if they provide them credit card as they would make their payment of credit card bills within 0 to 2 months. BUCKET 2 is considered as the neutral customers as they make delay in their payment by 2 – 3 months and their so much profit yielded by the bank. The customers who fall in BUCKET 3 to 7 are considered as bad customers as they delay in making the payment by 3 or more months to make their payment.

OUT-BANK-ACCOUNT: This field consists of the bank account number of customers, if the field has the bank account number is specified that means the repayment amount will automatically be deducted from customers' accounts. If the customer amount is not being mentioned that means that the customer will be making a manual payment to the bank.

OY-AMT-CASH: The cash the customer possesses in this account over a year of calendar time.

PY-AMT-CASH: The amount the customer possesses in his account for the present year. That is the year 2000.

YTD-AMT-CASH: Cash deposited in the customer account for the last 12 months.

YTD-MAX-BUCKET: This is the year-to-date maximum bucket number (where 0 being the maximum and 7 being the least bucket) that the customer has received when compared to the bucket number that the customer has received in the past 12 months. It's a categorical value.

PY-MAX-BUCKET: This field says the present years maximum bucket number the customer had achieved.

ACCOUNT-STATUS: The value in this field depicts the different status of the customer's account. There are 10 account statuses, and each customer has the status attached to the account. There are numeric and alphanumeric values attached to the account E.g.:'0S', '44', '55', etc. This field is considered as a string datatype.

CURRENT-BALANCE: This field shows us the customer current bank balance that the customer possesses in his / her account. The datatype of the values in the field is integer.

BRANCH-CODE: Each branch of Brain Bravo Limited has a different code and from the dataset, we know that there are 52 branch codes available. These branch codes are applied to all the customers to keep track of customers in which branch did they open their accounts. The datatype is in Integer format and is expressed in the form "XXX" (E.g.: 168).

Statistic	N	Mean	St.dev	Min	Pctl(25)	Pctl(75)	Max
Spending_Limit	8,476	9,70,480.20	1,11,330.80	1,00,000	10,00,000	10,00,000	10,00,000
Avg_Payments	8,476	25,058.81	30,263.07	11	13,000	25,058.80	5,09,930
Avg_Balances	8,476	8,85,616.10	2,01,243.70	30	8,75,724.50	10,09,655.00	20,46,746
Oy_Amt_Cash	8,476	8,06,751.80	47,268.61	3,20,000	8,06,751.80	8,06,751.80	10,20,000.00
Py_Amt_Cash	8,476	9,36,875.90	1,31,893.40	1,00,000	9,36,875.90	10,00,000.00	23,40,000.00
Ytd_Amt_Cash	8,476	8,68,206.50	2,30,350.80	93,000	8,68,206.50	10,00,000	37,00,000
Ytd_Max_Bucket	8,476	0.728	1.546	0	0	0	7
Py_Max_Bucket	8,476	0.206	0.888	0	0	0	7
Current_Balance	8,476	9,07,638.00	1,82,597.10	11	9,07,638.00	10,02,734	20,82,069
Bucket_Flag	8,476	0.106	0.308	0	0	0	1
Py_Max_Bucket_Flag	8,476	0.063	0.243	0	0	0	1
Ytd_Max_Bucket_Flag	8,476	0.225	0.418	0	0	0	1
Account_Status_Bucket	8,476	48.568	34.908	0	12	95	95
Age_Issued_Updated	8,476	42.454	10.14	18	35	49	80

Table 1: Summary of the variable in the training dataset.

4.2. Data Pre-Processing

The above section describes the field / variable and what exactly the variable stands for. On observing the data in the dataset of each variable we found there were missing values, wrong entries, wrong data format or datatype for few variables. For this reason, we must carry out the data pre-processing where we need to perform data cleaning, data manipulation (handle the missing values, wrong data entry, adjusting the right data format) must be done before proceeding to Exploratory data analysis.

Consistent data: It can be organized, perused, and superior caught on by giving information in a steady format. You can make beyond any doubt that the information is organized and put away consistently.

Project data: it is fundamental for organizations to be able to utilize chronicled information to extend the long run and to supply more in-depth investigation, particularly when it comes to accounts. Control of information makes it possible for this purpose.

Overall, being able to convert, update, delete, and incorporate data into a database implies you'll do more with the data. -Make more value from the data. It gets to be futile by giving information that remains inactive. But you may have clear experiences to create way better commerce choices after you know how to utilize data to your advantage.

Delete or disregard repetitive information: data that's unusable is continuously displayed and can meddle with what things.

Below are the variables where data cleaning and data manipulations have been done.

4.2.1. Data Manipulation (Handling missing values)

This section focuses on how the missing values are being handled and treated and how they are being filled. In a few instances, the recodes are being deleted and they are filled with values of 0 and NA.

We had to remove the fields which added no value to for modelling and exploratory data analysis. Fields like Account Number. Where each row had its uniquely identified account number for each customer. ACC_ACCOUNT_NO, NO_OF_CARDS, FREE_FEE_FLAG, CYCLE_CODE, INCREASE_LIMIT_FLAG, INCREASE_LIMIT_DATE, CARD_DELIVERY. The reason for removing these fields from the dataset is because these fields have the same value throughout the data set, NO variation in the values and no information was being generated from the values in each field during Exploratory data analysis.

Variables	Count (Before)	Count (After)
ACC_ACCOUNT_NO	11329	11305
NO_OF_CARDS	11329	11305
FREE_FEE_FLAG	11329	11305
CYCLE_CODE	11329	11305
INCREASE_LIMIT_FLAG	11329	11305
INCREASE_LIMIT_DATE	11329	11305
CARD_DELIVERY	11329	11305

Table 2: Variables that had missing values and count before and after removing missing values.

From the BIRTH_SINCE field, customers who had their birth year smaller than 1900 and the customer with birth year greater than 2002 had to be removed from the dataset since the customer is too old or too young to give any output. There are a total of 16 records that had to be removed. E.g.: 12620106, 14970925, 9501111 etc.

In The SPENDING_LIMIT field, the records which had the spending limit less the 1 had to be removed, since they happen to be the outliers. E.g.: The rows / records with spending limit "-" and "1" has been removed from the data set.

The AVG_BALANCES and AVG_PAYMENTS have the values which are not an integer value it has a hyphen "- " at the end of the number for few records. Those values had to be replaced with the NA values.

BEFORE		AFTER	
AVG-PAYMENTS	**AVG-BALANCES**	**Avg-Payments-Flag**	**Avg-Balances-Flag**
4167	274899	1	1
25833	-	1	NA
0	264578	NA	1
45417	-	1	NA
24750	-	1	NA
26000	414781	1	1
-	-	NA	NA

Table 3: Pre-processing of AVG-PAYMENTS and AVG-BALANCES to Avg-Payments-Flag and Avg-Balances-Flag.

4.2.2. Data Formatting

There are variables that have different records in datatype for example in the BIRTHDATE and MEMBER_SINCE variable had records in integer datatype, but it should have had the data in the date datatype. The format had to be changed. ACCOUNT_STATUS had alphanumeric values that had to be changed to integer data type. This section focuses on changing the data type of variable and formatting the record in such a way that it can be used for exploratory data analysis and modelling.

Creating a new field to replace the field MEMBER_SINCE to Menbership_Date field. The value in this field has been expressed in "yyyymm" format and was an integer datatype, "01" is added to each record denoting it as 01 date of the month and the year. The new format is "yyyy-mm-dd". The new datatype of the value is data. The BIRTH_DATE field has the record which is an integer and these values in the field had to be converted to "yyyy-mm-dd" format and set to date datatype.

BEFORE	**AFTER**	**BEFORE**	**AFTER**
BIRTH-DATE	**Birth-Date**	**MEMBER-SINCE**	**Membership-Date**
19520711	1952-07-11	199910	1999-10-01
19661002	1966-10-02	199911	1999-11-01
19510716	1951-07-16	199912	1999-12-01

Table 4: Pre-processing of BIRTH-DATE and MEMBER-SINCE field and changing its format to date.

The ACCOUNT_STATUS field consists of values that show different status of the account and there were three alphanumeric values (0S, ME and NE) that had to be replaced with the numeric values (75, 85 and 95) as shown in the table below.

BEFORE	AFTER
ACCOUNT-STATUS	**Account_Status_Bucket**
0	0
21	21
NE	95
12	12
ME	85
33	33
55	55
44	44
65	65
0S	75
22	22

Table 5: pre-processing of ACCOUNT-STATUS to Account-Status-Bucket.

The LAST_TRX_DATE and AT-ANNIV_DATE had data there in integer format and this data had to be converted to date format in order to see that last time when the customer performed his last transaction and the AT_ANNIV_DATE shows us the customer's anniversary date. The data in this field had to be converted to date format "yyyy-mm-dd' format to get the last transaction date of the customer and the anniversary date of the customer with the bank.

Removing of comma's "," from all the records from the data set and converting the datatype into integer datatype so that it would be easy for use to consider the values as a continuous value and use it for modelling and exploratory data analytics.

4.2.3. Binary Encoding

The binary encoding is nothing but representing the values in terms of 0, 1. This section focuses on how binary encoding is being implied to few variables so that it would be convenient for use to use it for modelling and performing exploratory data analysis.

SEX field represents whether the customer is Male or Female and the values in the field are "F" and "M". The "M" and "F" has been classified as "0" and "1", where "0" represents Female and the Male

represents as “1”. The name field created is Sex_M_Flag and it’s an integer datatype and can be used as a categorical value.

BEFORE	AFTER
SEX	**Sex_M_Flag**
F	0
F	0
M	1
M	1

Table 6: Example of SEX field before and after pre-processing.

Blacklist_Flag: This field is created for the replacement of Black_List_Date where the customer with the blacklist date has been assigned with number ”1” and the customer with no blacklist date is assigned “0”. It is an integer data type.

BEFORE	AFTER
BLACK-LIST-DATE	**Blacklist_Flag**
20001205	1
0	0
20001205	1
0	0
0	0
20010215	1
20001205	1

Table 7: Conversion of BLACK-LIST-DATE to Blacklist_Flag (0,1)

Attriation_Flag: The customer with attrition reason code is being assigned with value “1” and the customer with no attrition code is assigned “0”. It is an integer data type.

BEFORE	AFTER
ATTRITION-REASON	**Blacklist_Flag**
22	1
0	0
22	1
0	0
65	1

Table 8: Conversion of ATTRIATION-DATE to Attriation_Flag (0,1)

Writeoff_Flag: The value "1" is being assigned if the customer has the write-off date mentioned under WRITEOFF_DATE field. If the write-off date has not been mentioned value "0" is being assigned to it. It is an integer data type.

BEFORE	AFTER
WRITE-OFF-DATE	**Blacklist_Flag**
20001227	1
20001227	1
0	0
0	0

Table 9: Conversion of WRITE-OFF-DATE to Writeoff_Flag (0,1)

All_Max_Bucket_Flag_updated: The new file is created replacing the ALL_MAX_BUCKET field. The bucket field had bucket numbers from 0 to 7. In the All_Max_Bucket_Flag_updated field, the buckets 0 and 1 are grouped and assigned to a number "0" and the buckets 2, 3, 4, 5, 6, and 7 are being grouped and assigned to number "1". This field is being treated as a categorical variable.

BEFORE	**AFTER**
ALL-MAX-BUCKET	**All_Max_Bucket_Flag_updated**
0	0
1	0
2	1
3	1
4	1
5	1
6	1
7	1

Table 10: pre-processing of ALL-MAX-BUCKET field to All_Max_Bucket_Flag_updated.

Ytd_Max_Bucket_Flag: The new field is created replacing the YTD-MAX-BUCKET field. The bucket field had bucket numbers from 0 to 7. In the Ytd_Max_Bucket_Flag field, the buckets 0 and 1 are grouped and assigned to a number "0" and the buckets 2, 3, 4, 5, 6, and 7 are being grouped and assigned to number "1". This field is being treated as a categorical variable with datatype as integer.

BEFORE	AFTER
YTD-MAX-BUCKET	Ytd_Max_Bucket_Flag
0	0
1	0
2	1
3	1
4	1
5	1
6	1
7	1

Table 11: Preprocessing of YTD-MAX-BUCKET to Ytd_Max_Bucket_Flag.

Py_Max_Bucket_Flag: The new field is created replacing the PY-MAX-BUCKET field. The bucket field had bucket numbers from 0 to 7. In the Py_Max_Bucket_Flag field, the buckets 0 and 1 are grouped and assigned to number "0" and the buckets 2, 3, 4, 5, 6, and 7 are being grouped and assigned to number "1". This field is being treated as a categorical variable with datatype as an integer.

BEFORE	AFTER
PY-MAX-BUCKET	Py_Max_Bucket_Flag
0	0
1	0
2	1
3	1
4	1
5	1
6	1
7	1

Table 12: Pre-processing of PY-MAX-BUCKET to Py_Max_Bucket_Flag.

4.3. Feature Extraction

Here we discuss on creating new variables from existing variables. This is done to create a new dimension of things and helps us to create a new variable that is required for further analysis and get meaningful insights from the data. This section explains how we created new variables using the existing variable and gives the description of the new variable.

New field had to be created by name "Age" by using BIRTH-DATE field, that is consists of the age of the customer by considering 2000-01-01 as the present year and subtracting that with the BIRTH_SINCE to get the age of customer in years. The formula used to calculate the age.

Age = (BIRTH-DATE) – (Present year).

Note: The present year is considered as (2000-01-01).

The Age_Issued field is created by subtracting the Membership_Data field with the BIRTH_DATE field and it is an integer datatype. The formula used to calculate the Age_Issued is as below. This field would help us know since when the customer created the bank account with Brain Bravo Limited.

Age_Issued = (Membership-Date) – (BIRTH-DATE).

Days_Blacklist_Membership: This field shows us the number of days the customer got into the backlist, and this is calculated by subtracting the date when the customer is blacklisted (BLACKLIST_DATE) from the day when the customer became member of the bank (MEMBER-SINCE).

Days_Blacklist_Membership = (Membership-Date) – (BLACKLIST-DATE)

Days_Writeoff_blacklist: This field depicts the number of day that the customer took to move from backlist date to write-off date. These values in the field are calculated by subtracting the dates from WRITE-OFF-DATE from the dates present in BLACKLIST-DATE field.

Days_Writeoff_blacklist = (WRITE-OFF-DATE) – (BLACKLIST-DATE)

A new variable for BRANCH_CODE was being created. The PNL_All_Max_Bucket field was created by considering the ALL_MAX_BUCKET and BRANCH_CODE field. We calculated the quartile percentage of all the good customer by considering the quartile value of customers who belong to bucket 0 and 1 from ALL_MAX_BUCKET field and updated the value in PNL_All_Max_Bucket field as shown in the table below.

The Branch_Code_Rating is given by the number 1 to 4. Each branch is given rating from 1 to 4 where 1 being the good and 4 being bad rating. The Branch which had very good customers who have the percentile above 75% are categorised to rating 1. Branch which had percentage of good customers

between 51%-75% are categorised to good customers and given assigned as 2 rating. The customers with 26% - 50% are average customers with rating number 3 and customers with 25% and below are consider is poor customers as rating 4.

In the new field Age_Issued_Updated, the customers who with age above 81 and the below 18 had to be replaced with value “A” as customer below 18 years old will not be issued a credit card and the customers above 81 don’t have many transactions to carry out.

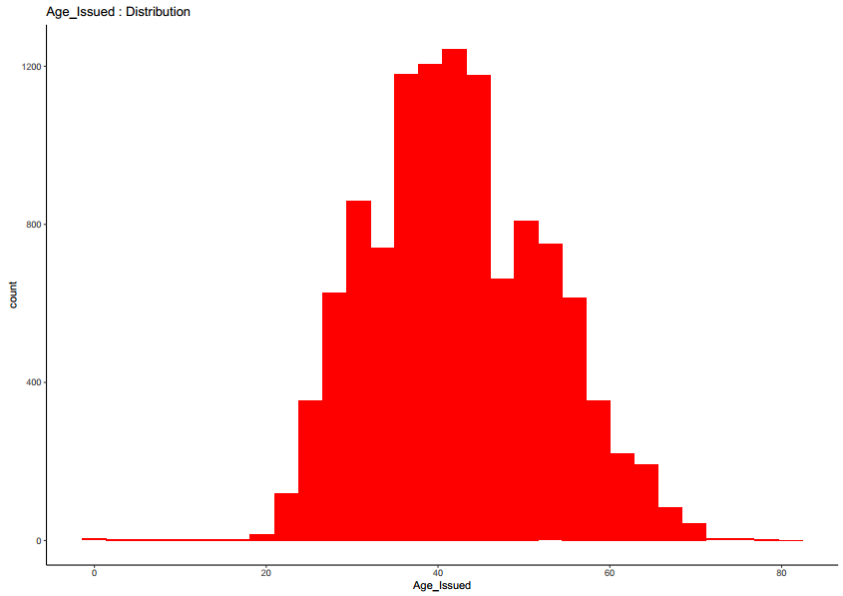

Figure 4.1: Age issued vs Count

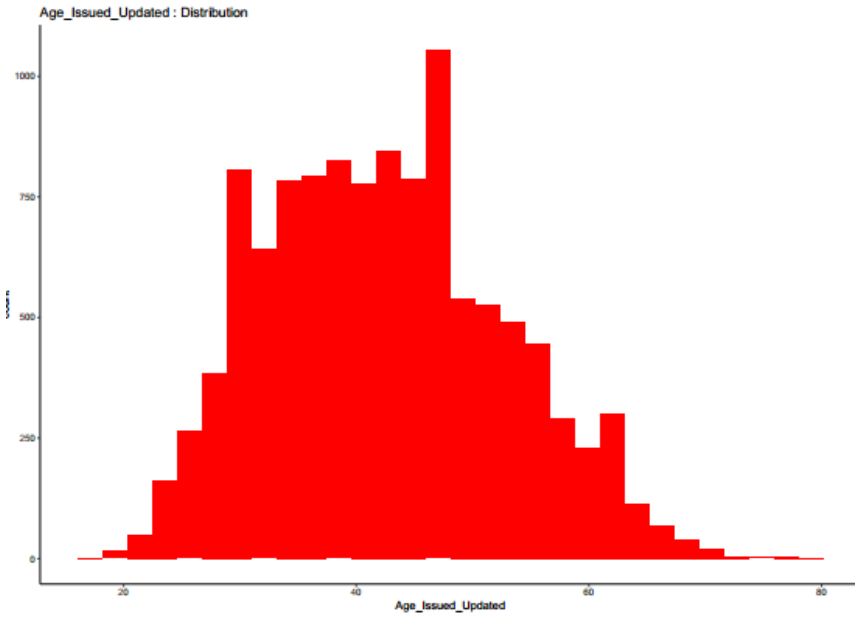

Figure 4.2: Age_Issued_Updated vs Count.

In figure 4.1.1 we see that there are few outliers where the data are at the extreme end where the customers are below 18 or above 80 years old, we are not able to make which age category do we

have many customers. After removing the outliers, we can see that the graph has improved and seen that the majority of customers lie in the 50 age category and see the peak in the histogram graph.

4.4. Splitting of Dataset

The performance evaluation is done by splitting the main data set into Training data set and other one is Testing data set. These main data set is split into 70% of Training data set where we will be trying different set of Machine learning model to find the best fit model and testing the model on the 30% of the original data set called as Test Data set. The 70% of training and 30 % of testing dataset is created by taking equal data from all the buckets.

5. Exploratory Data Analysis.

The main objective of the exploratory data analysis (EDA) is to understand the data distribution better and discover the dataset's patterns using graphs and statistics. The key focus of this activity is to identify the variables having good variance for the dependent variable values, i.e. "All max bucket and flag". This will help us identify the variables that can help predict the customers' application and behaviour scoring buckets. In this section following topics shall be covered:

1. **Univariate Analysis:** To understand the distribution of the variables
2. **Bi-Variate Analysis:** To understand the relationship of the variables wrt All max bucket and flag

Univariate Analysis:

Blacklist date: This variable has the blacklist date of the customers who have been blacklisted by the banks. Around 568 customers of ~11k customers have been blacklisted during the considered time period. The distribution shows that the bank blacklisted many customers during the month of December’2000 and July’ 2001. It will be interesting to understand the reason for the same. It can be due to banks' processes or the sudden change in the customers' behaviour during these months. Customers may have overshopped during Christmas/holiday season in December or during the summer holidays in July and have failed to pay bills on time, resulting in the blacklisting activity. Since there is no transaction data provided, we couldn't check these hypotheses.

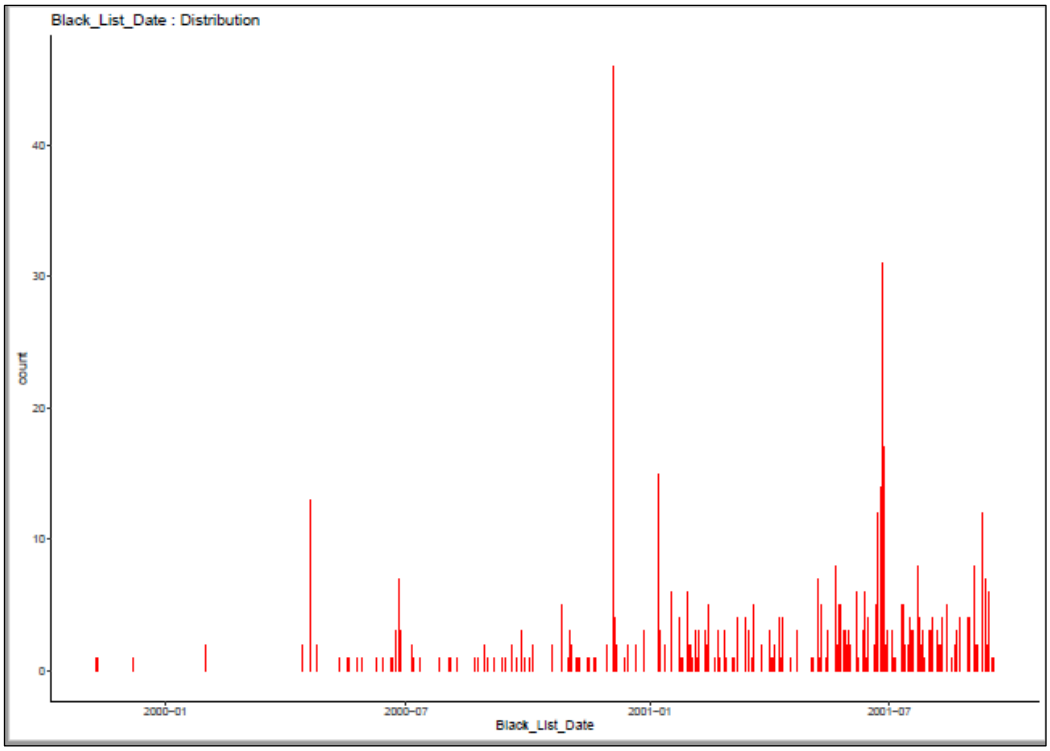

Figure 5.1: Black_List_Date vs Count.

Membership Date: This variable shows the membership date of the customers. It can be seen from the below graph that the bank has rolled out many cards during the last quarter of 2000 and during the month of July 2001. This can be again due to the mark of holiday seasons and customers might be looking for a credit card for additional financial needs.

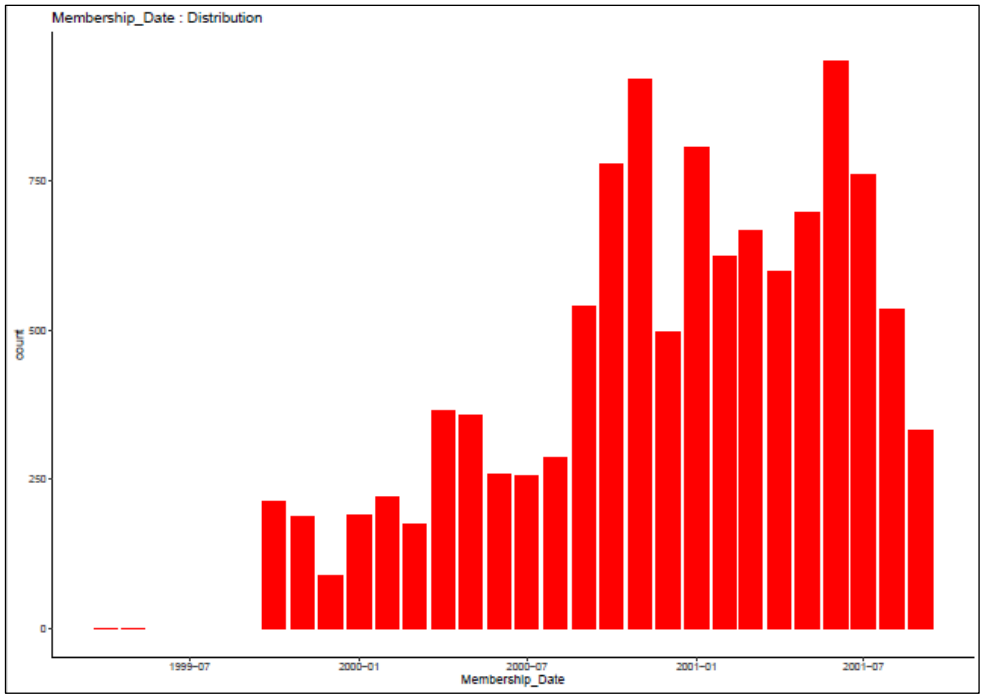

Figure 5.2: Membership_Date vs Count

Sex M Flag: This variable shows whether the person is male or not. The distribution indicates that almost two-third of the bank customers are males i.e. 7,737 of 11K customers. It would be interesting to see the All max bucket and Sex M flag distribution to understand the difference in the all max bucket based on the gender of the person.

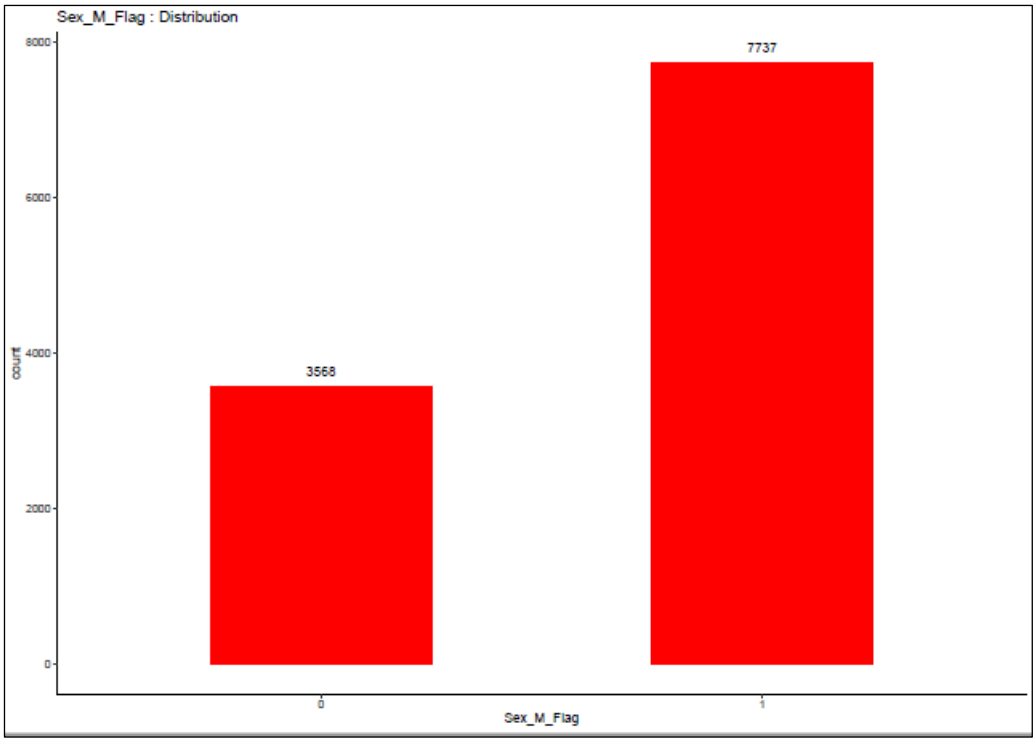

Figure 5.3: Sex_M_Flag vs Count

Out Bank flag: This variable has the values as 1 for the customers who have linked their account with the card for auto-payment feature to pay their credit bills. The below graph shows that almost 40% of the customers have opted for the auto-payment feature. It is expected that these customers would have fewer defaulters and would mostly belong to the good bucket wrt All max bucket since they would be able to pay the bills on time due to the auto payment feature.

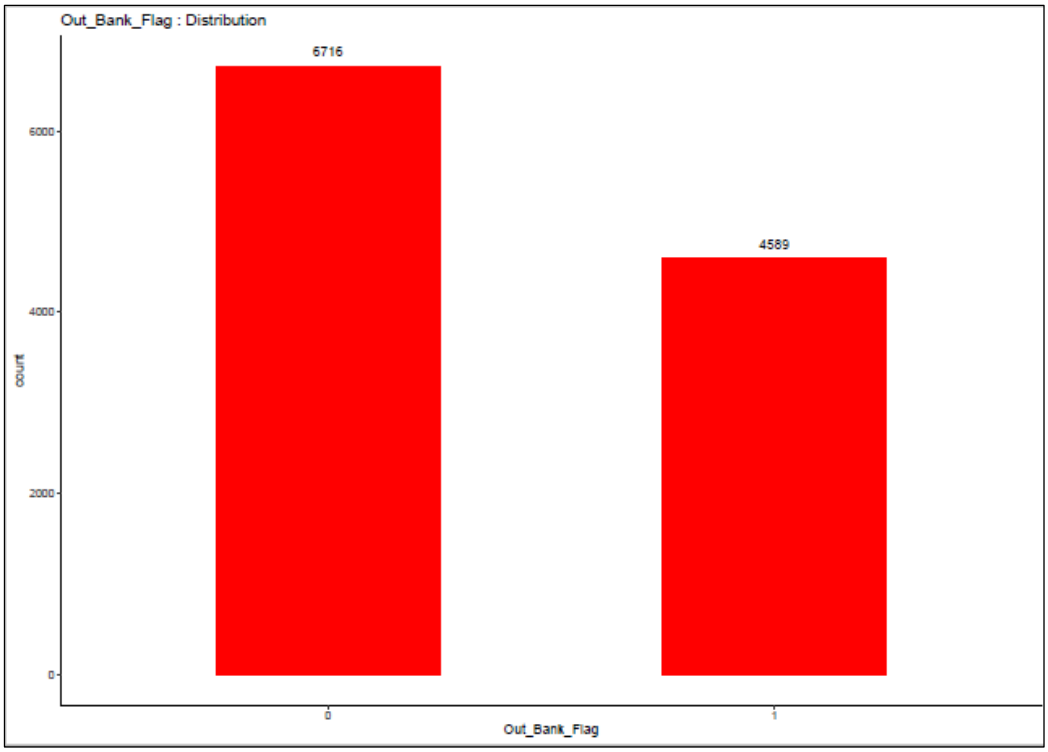

Figure 5.4: Out_Bank_Flag vs Count

All max bucket: This is the main dependent variable, which shows the bank's maximum bucket assigned to the customer. The graphs show that 8,597 customers of 11k total customers belong to 0 category, i.e. good customers. This means ~76% of the customers are good performing customers. This is the case of imbalance class since a disproportionate number of customers present in the 0 category. Therefore, some techniques such as under-sampling major class, Smote etc. should be used to handle this problem. Since predicting the variable having this kind of distribution will mostly result in higher prediction accuracy because the model will tend to predict every customer into 0 category. Also, it can be noticed in the graph that there is a drop in the number of customers for higher buckets except for the "Bucket 7". This is because we have changed the bucket of all the blacklisted customers to "Bucket 7". So, number of customers have increased in this bucket.

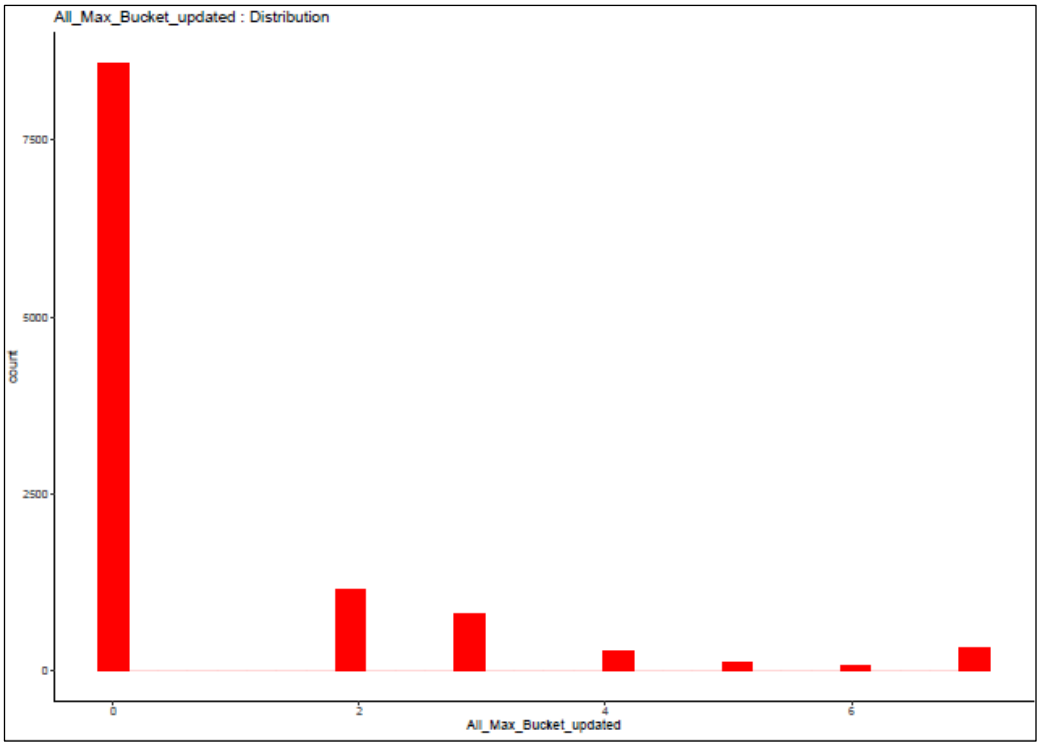

Figure 5.5: All_Max_Bucket_updated vs count

Spending Limit: The graph shows there is almost no variation in the spending limit of the customers. Nearly all of them have a spending limit of 1 Million units. There are only handful of customers who have spending limit < 1million (as seen in the dotted portion of the box plot). A nearly similar trend was observed for variables Avg. Payments, YTD Amt Cash, Py_Amt_Cash. However, it would be interesting to check if these variables have a good variation w.r.t to "All max flag" and "All max bucket" variables.

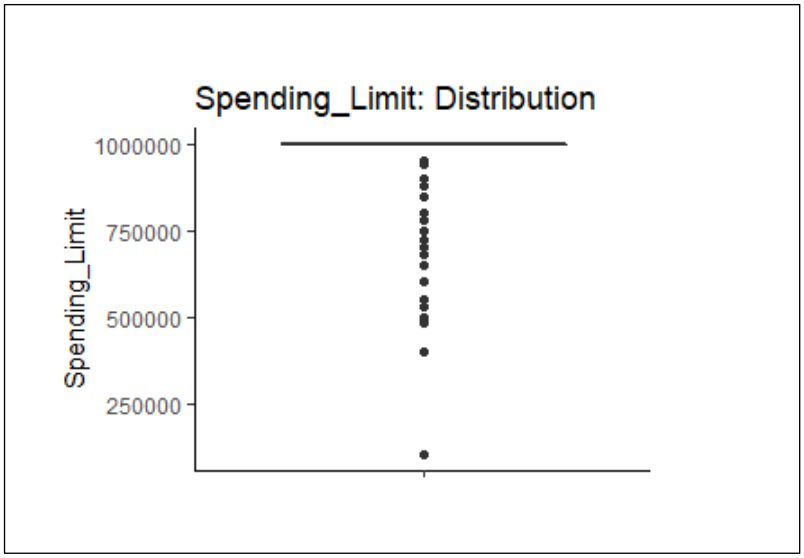

Figure 5.6: Spending limit distribution

Avg. Balances: There is a slight variation in the avg. balances of the customers. Customers have the median average balance value of ~980 k and first quartile and third quartile values as ~860 k and ~101 k, respectively. Similar trend can be noticed for the "Current balances" field as well. We need to check for the correlation between these fields since both the variables seem to have a similar trend and distribution.

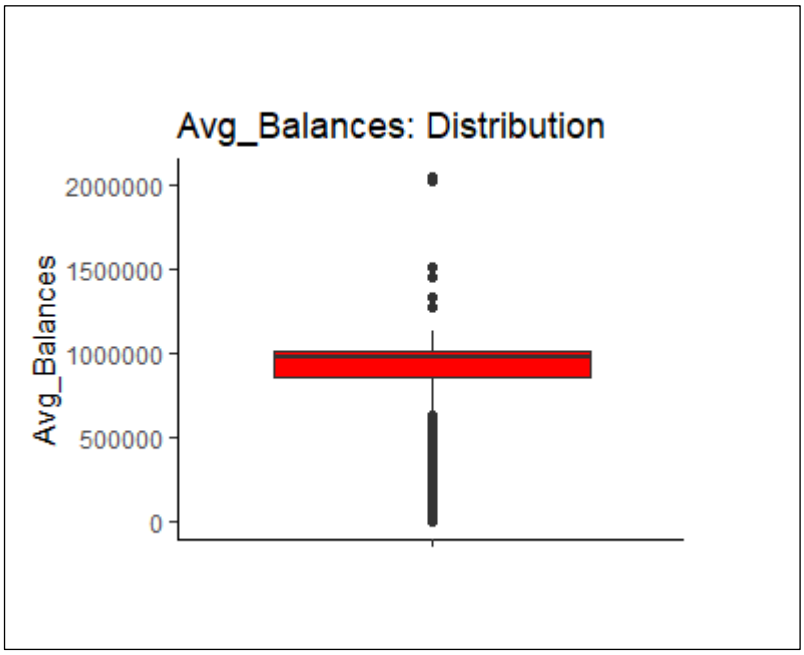

Figure 5.7: Boxplot of Avg_Balances

Days_Anniv_membership: This variable shows the number of day's difference between Anniversary and membership date of the customers. This distribution shows that some cards are having an anniversary < 10 days. This seems to be a data error or some other issues, which should be checked from the bank. Also, most customers have been given 365 days (1 year) or 2 years membership cards. It would be interesting to check if there is any difference in the membership days between "Good category – Bucket 0 and 1" customers and other customers.

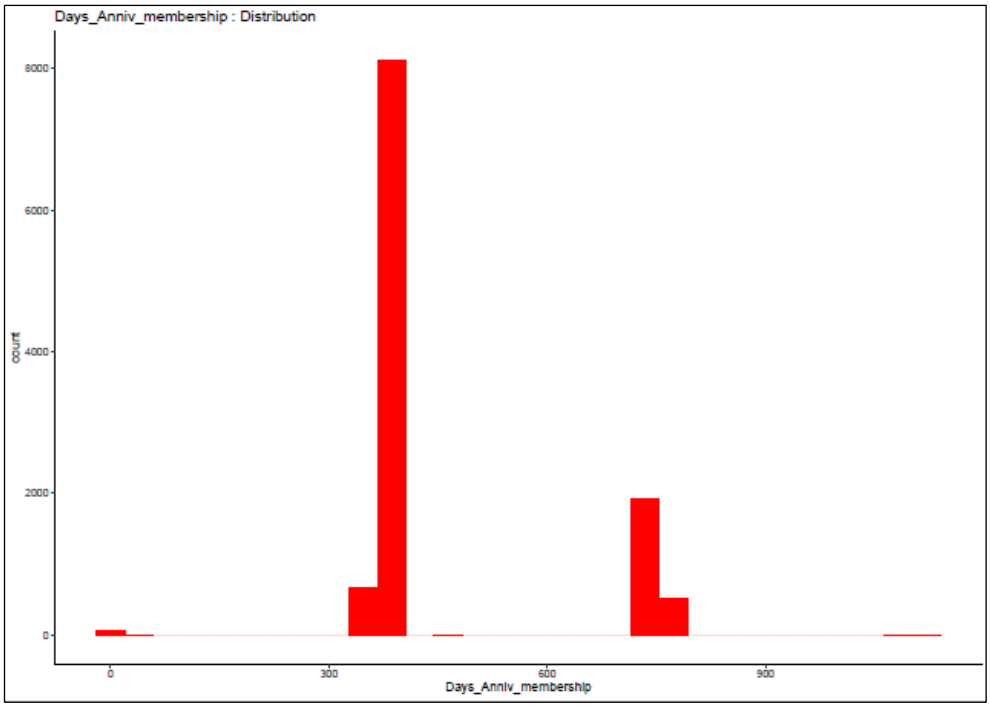

Figure 5.8: Days_Anniv_membership vs Count

Days blacklist membership: This variable shows the day's difference between Blacklisted and membership date of the customers. The data shows that the blacklisted customers' percentage drops significantly post 400 days (> 1 Year) of membership, but we need a few more days of the data for the recent members to test this hypothesis.

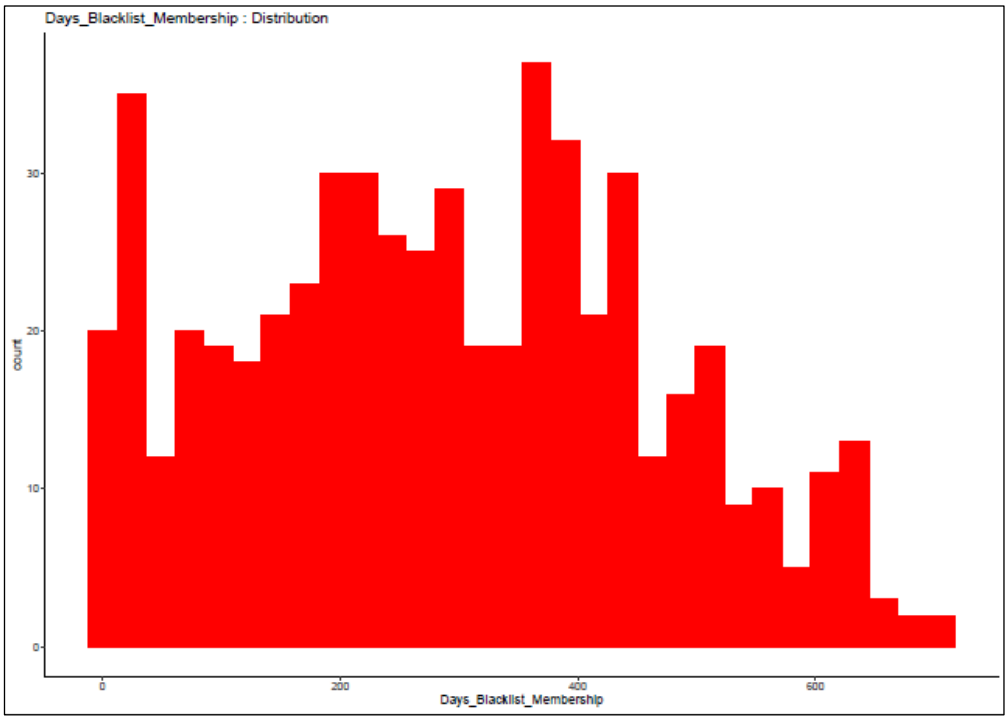

Figure 5.9: Days_Blacklist_Membership vs Count.

Account status bucket: There are around 10 account status buckets present in the data. The chart shows that almost 30% of the customers (3378 of 11K) have account status as "95". We couldn't get much information for this variable, so it's tough to comment on what these account statuses signify.

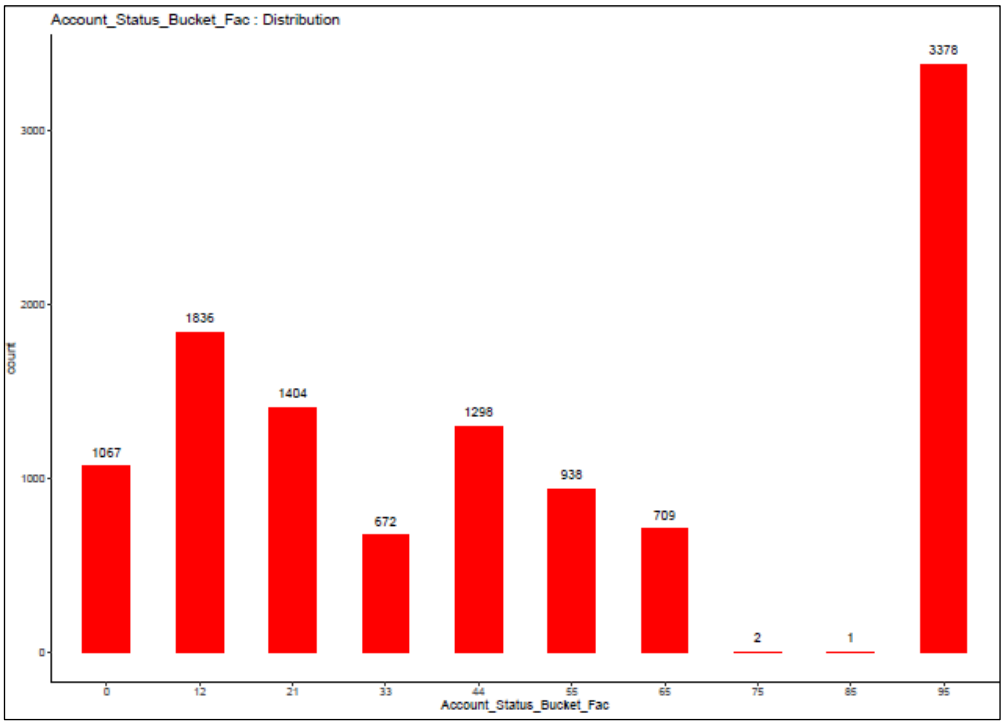

Figure 5.10: Account_Status_Bucket vs Count.

Bivariate Analysis:

Age issued updated and All max flag: The average age issued for the All max flag = 0 or "good customer" is slightly higher than All max flag =1 customer.

The t-test also shows the p-value less than 5%, so we can reject the null hypothesis -"Difference in age issued for all max flag 1 and all max flag 0 is not greater than 0". Hence we can conclude that the age issued for the 0 category "good customers" is significantly higher than other customers. This is an important insight, and thus, the variable will be helpful while modelling for the application and behaviour scoring.

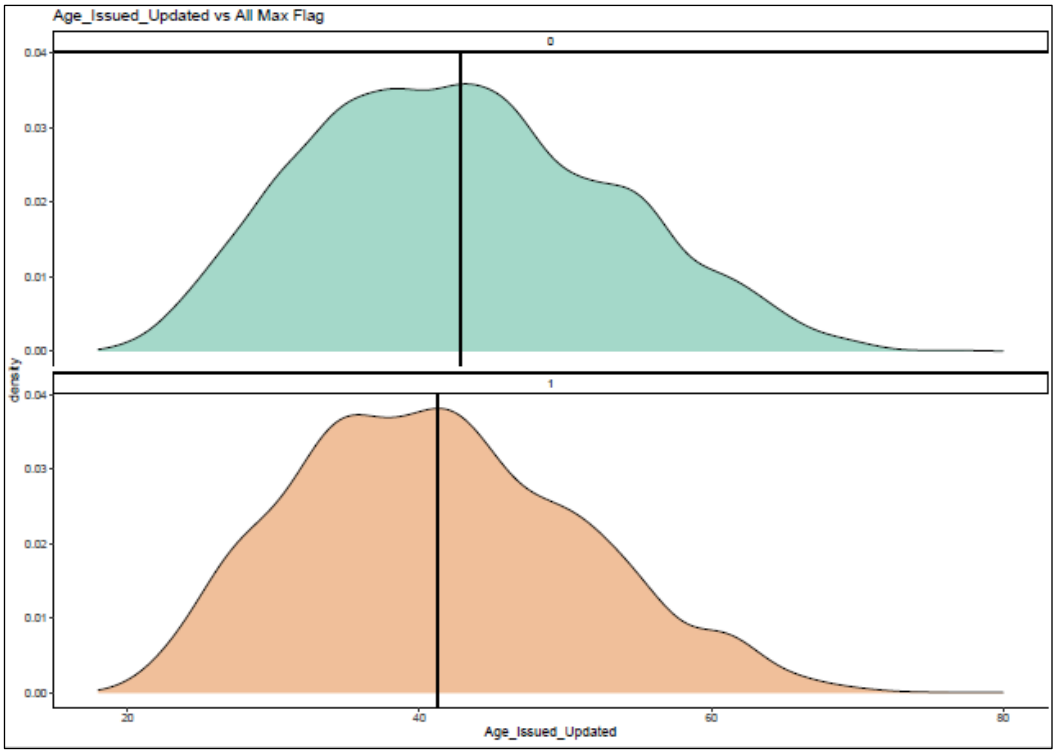

Figure 5.11: Density plot for Age_Issued_Updated vs All max flag

```
        subset(eda_ds$Age_Issued_Updated, eda_ds$All_Max_Bucket_Flag_updated == 0)
and subset(eda_ds$Age_Issued_Updated, eda_ds$All_Max_Bucket_Flag_updated == 1)
        t = 6.8567, df = 4687.3, p-value = 0.000000000003981
        alternative hypothesis: true difference in means is greater than 0
        95 percent confidence interval:
         1.141565      Inf
        sample estimates:
        mean of x mean of y
         42.79960  41.29767
```

Branch code rating and All max flag: Due to the definition of the branch code rating, we can see the branches having higher ratings have a lower percentage of good/profitable customers. Whether we can use the dependent variable information to create a new independent variable and use it in the modelling can be debated. Therefore, we have highlighted and discussed the same with the Brainbravo team before using this variable in the modelling.

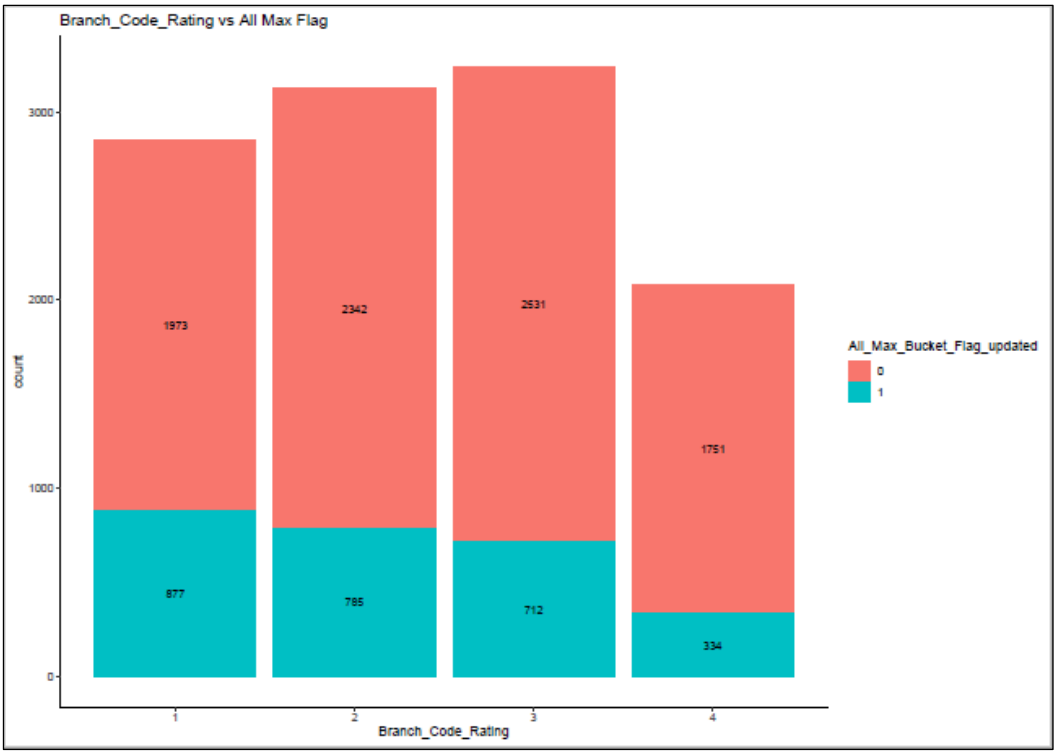

Figure 5.12: Branch_Code_Rating vs All max flag

Membership date and All max flag: It can be seen from the below graph that the % of "bad or All max bucket 1" category customers were higher for the customers enrolled during the holiday season. However, it should be noted during the last few months the percentage of the bad customers is lower. It can be because these people haven't spent much time in the system to be categorised under bad/non-profitable customers. But it seems that this variable can be used in the modelling to get good results to tag customers into profitable (Bucket 0-1) and non-profitable customers (Bucket 2-7).

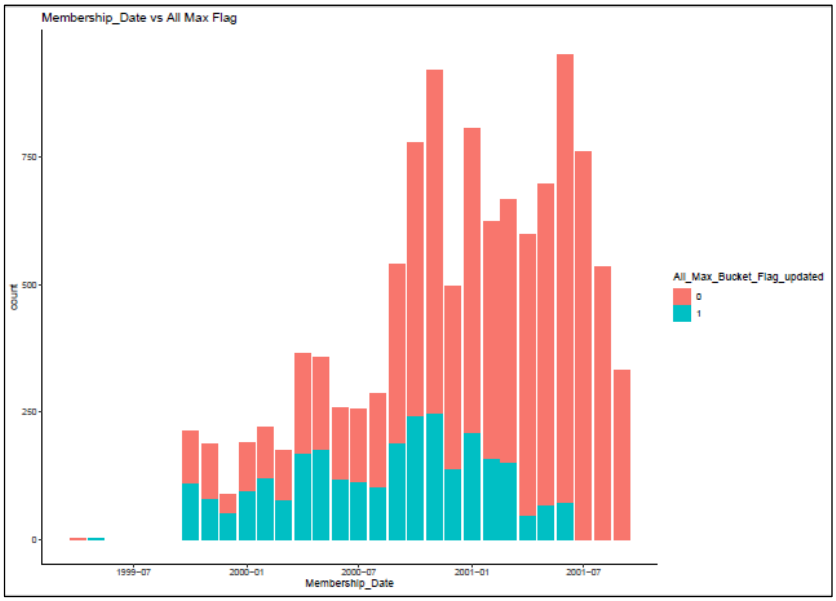

Figure 5.13: Membership_Date vs All maxflag

Sex_M_Flag and All max bucket: The below graph provides crucial insights into the scoring of the customers based on gender. The chart clearly shows that the percentage of female customers is much higher in the good or 0 category bucket. There are almost 33% female customers in the 0 buckets, while for other buckets, it's almost ~20% and drops to ~5% for the last buckets. This can be a good variable to be considered in the modelling and to understand and quantify the impact of gender on the scoring of the customers.

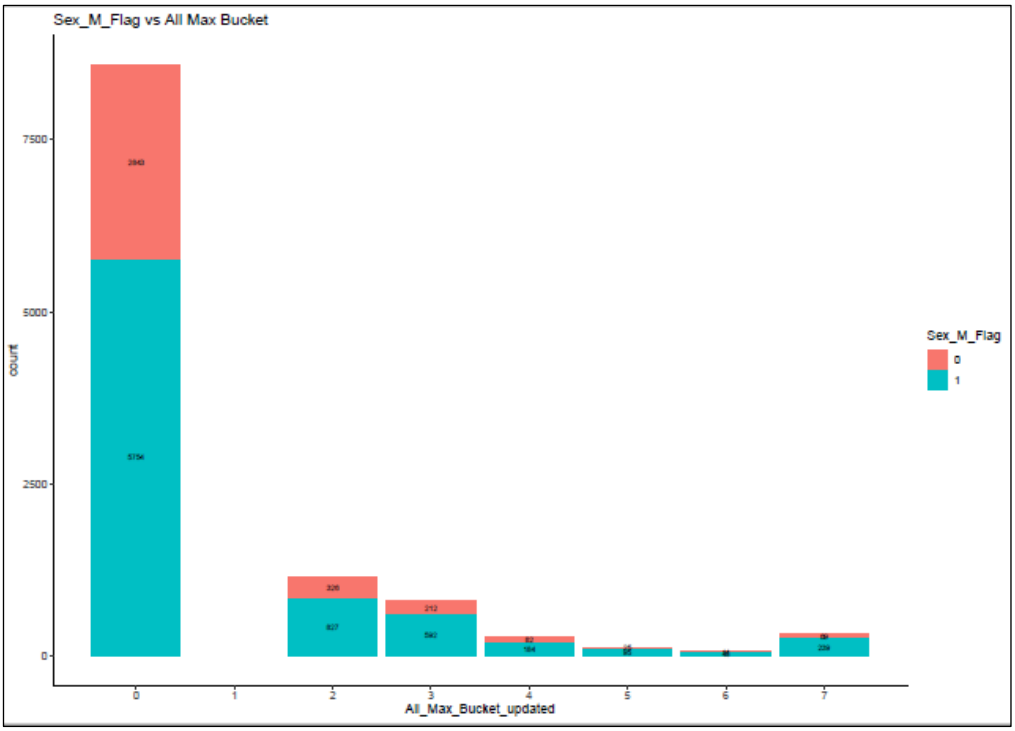

Figure 5.14: All_Max_Bucket_Updated vs All max bucket.

Out_Bank_Flag and All max flag: The result of this graph looks shocking. It was expected that the percentage of customers having auto-payment feature would be higher for "0 category – good/profitable" bucket customers, but unfortunately, that's not the case. The graph shows that for both the categories "0 and 1 All max bucket," the percentage of customers having the auto-repayment feature is the same, i.e. 40%.

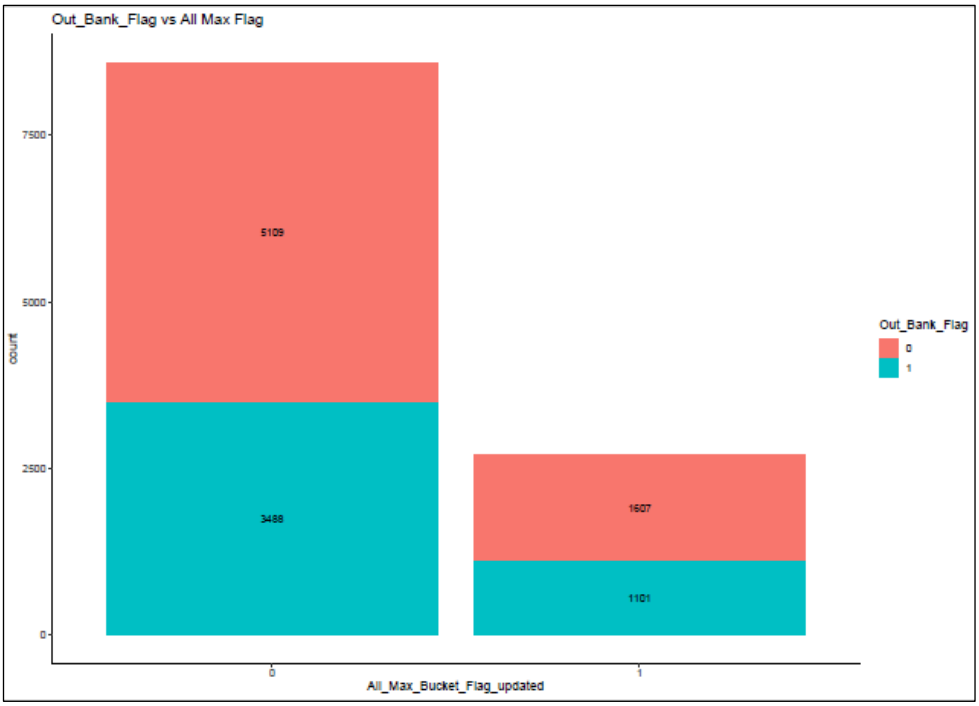

Figure 5.15: All_Max_Flag_updated vs Out bank flag

Spending limit and All max flag: This graph shows that the "0 All max bucket customers" spending limit seems a bit higher than other customers. Though it's not clear, it seems the bank has set higher limits for the cards for potential good customers or customers with a good repayment history. The t-test result also shows that the spending limit of the All max 0 category customers is significantly higher than All max 1 category customers. Hence, this variable can be crucial for application and behaviour scoring.

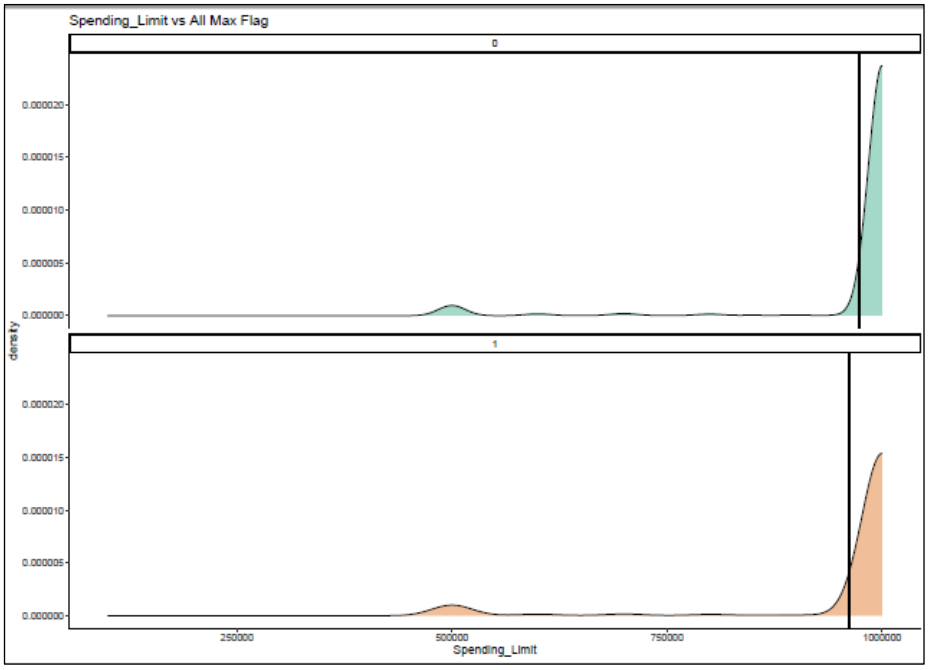

Figure 5.16: Spending_Limit density plot vs All max flag

```
        subset(eda_ds$Spending_Limit, eda_ds$All_Max_Bucket_Flag_updated == 0) and
subset(eda_ds$Spending_Limit, eda_ds$All_Max_Bucket_Flag_updated == 1)
        t = 4.2675, df = 3979.7, p-value = 1.011e-05
        alternative hypothesis: true difference in means is greater than 0
        95 percent confidence interval:
         7062.531      Inf
        sample estimates:
        mean of x mean of y
         972992.9  961499.3
```

P value is less than 5% so we can reject the null hypothesis that true difference in means is not greater than 0.

Account status and All max flag: The chart shows a slight variation in the distribution of the customers' account status belonging to "All max bucket 1" and "All max bucket 0" customers. There are very few customers with Account status as (33,44,55) for the All max bucket 1 customer while there are a good proportion of customers for All max bucket 0 customers for these account status. Due to this variation, Account status might be helpful in the application and behaviour scoring of the customers.

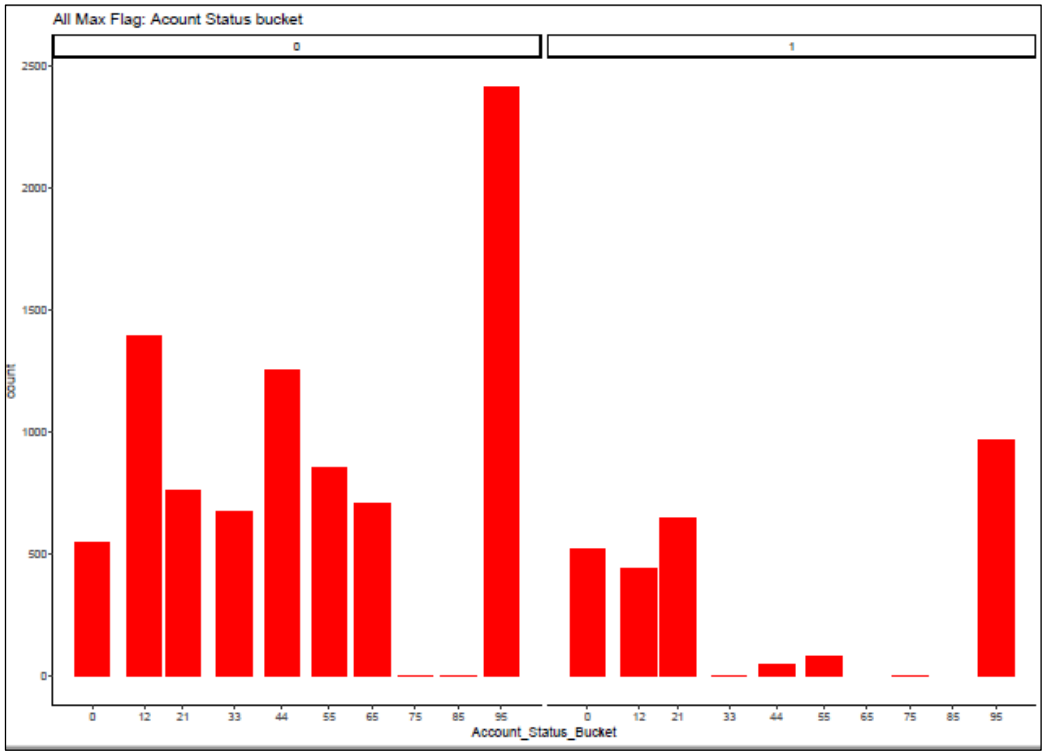

Figure 5.17: Account_Status_Bucket vs All max flag

Average Balances and All max bucket: It can be seen from the below chart that the distribution of average balance across All max bucket is almost the same except for the bucket 7 customers. The first quartile (25%) and third quartile (75%) for All max bucket 0 – 6 is between 900k to 1.1 Mn units. The median value of "All max bucket 7" is 850K, which is much lower than the other "All max buckets". Also, some of the customers in the lower buckets (0-3) have very low values of the average balances. But overall, it can be seen that the average balances for the bucket 7 customers is way lower than the other customers. This information will be helpful while predicting the All max bucket for the customers. Also, no such variation and trends were noticed in the cash variables such as Py_Amt_Cash and Ytd_Amt_cash.

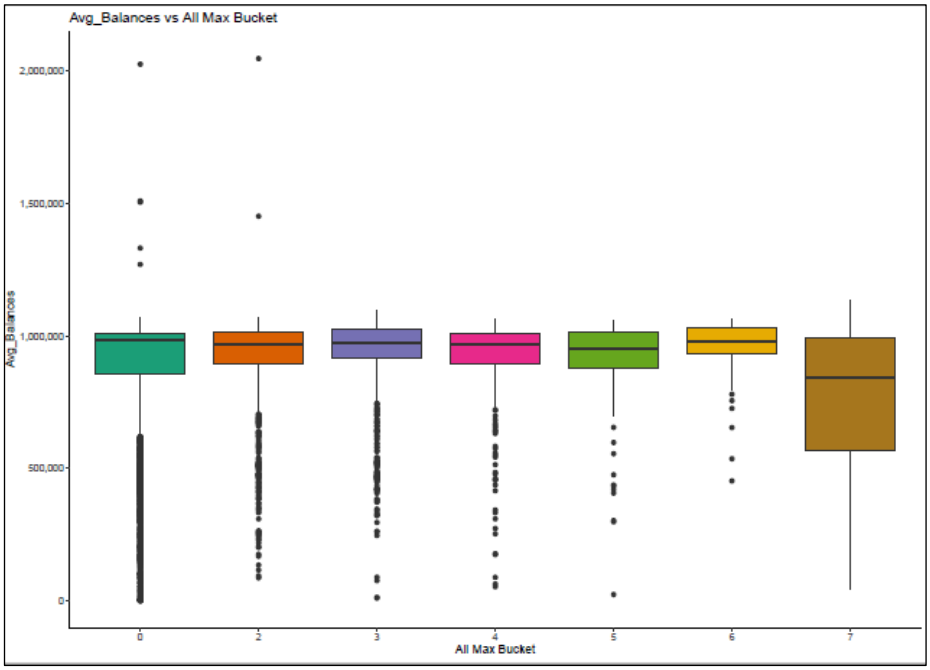

Figure 5.18: All_Max_Bucket vs Avg_Balances

Correlation Matrix: The below matrix shows the correlation between all possible numeric pairings of values in a table. It's a helpful tool for quickly summarizing a large dataset and identifying and visualising trends in the data. A correlation matrix consists of rows and columns that show the variables. The correlation coefficient is contained in each cell of a table. Numeric variables having a very high number of NA values such as Py_Amt_Cash and Ytd_Amt_Cash, have been removed from the data while creating the confusion matrix. The correlation matrix shows that there is no high correlation between all max buckets and other variables. It would be interesting to check the joint associations of these independent variables with All max bucket using the regression model. Also, some flag variables have not been considered in the confusion matrix but can help with the prediction of the buckets. There is a strong correlation between the spending limit, Avg balances and current balances. This might be

due to the design of the definition of these variables. Hence only one of these variables should be used while building the model.

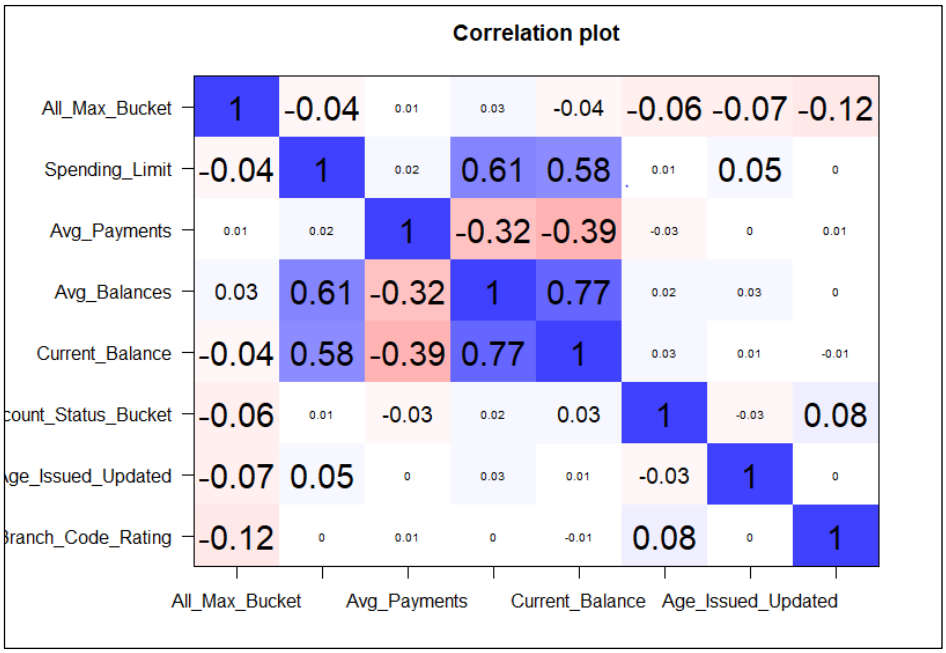

Figure 5.19: Correlation matrix

6. Methodology (Machine learning algorithms).

After completing the pre-processing of the data, we use the training dataset (CS_Train) which is 70% of the original dataset provided by Brain Bravo Ltd. We use the training dataset to try and fit different machine learning algorithms and test the predicted output using the test dataset (CS_Test). Below are the algorithms used for training the dataset.

1. Logistics regression.
2. Linear regression.
3. Classification And Regression Tree (Decision Tree)
4. Random forest.

Off the above-mentioned machine learning methods, there are 3 classification methods, and they are Random Forest, Logistics regression and Decision tree. The only 1 regression method is linear regression. The classification method is used when we have categorical value in the dependent variable and each number is considered as a different number and not a continuous number. The output prediction is required in a categorical value. For the regression method the dependent value needs to be considered as the continuous values.

Confusion / Classification Matrix

A confusion matrix may be a strategy for summarizing the execution of a classification algorithm.

Calculating a confusion matrix can deliver you distant better; a much better; a higher; a stronger; an improved">a distant better thought of what your classification show is getting right and what sorts of mistakes it is making.

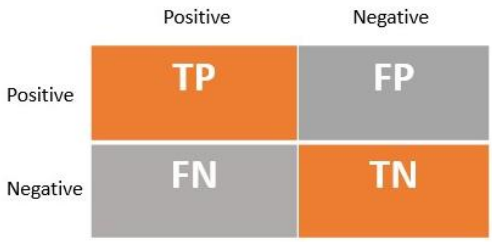

Figure 6.1: Confusion/classification matrix.

Where,

TP = Truly Positive, **FP** = False Positive, **FN** = False Negative, **TN** = Truly Negative.

True Positive (TP): It alludes to the number of expectations where the classifier accurately predicts the positive lesson as positive.

True Negative (TN): It alludes to the number of expectations where the classifier accurately predicts the negative course as negative.

False Positive (FP): It alludes to the number of expectations where the classifier erroneously predicts the negative lesson as positive.

False Negative (FN): It alludes to the number of expectations where the classifier inaccurately predicts the positive course as negative.

Sensitivity: It tells you what division of all positive tests was accurately anticipated as positive by the classifier. It is additionally known as Genuine Positive Rate (TPR), Affectability, and Likelihood of Location. To calculate Sensitivity, utilize the taking after equation:

Sensitivity = TP / (TP + FN)

Specificity: It tells you what division of all negative tests are accurately anticipated as negative by the classifier. It is additionally known as Genuine Negative Rate (TNR). To calculate specificity, utilize the taking after equation:

Specificity = TN / (TN + FP)

Accuracy: It gives you the general accuracy of the demonstration, meaning the division of the overall tests that were accurately classified by the classifier. To calculate exactness, utilize the taking after equation:

Accuracy = (TP+TN)/(TP+FP+FN+TN)

Average Profit / Loss: It gives us a number that represents profit or loss to the bank if they give the credit card to the customer. If we get average profit / Loss as a positive number, that represents that customer will yield profit to the bank if they give him the credit card. If we get a negative number as an answer, that represents the bank will face loss if they provide credit card to that customer. Different methods were proposed by Brainbravo Ltd. to calculate the average profit/loss based on the type of algorithm. This is covered in details in the performance evaluation part of the report.

Profit_rate: number of correct positive (TP) bucket prediction in the confusion matrix.

Loss_Rate: Number of correct negative (TN) bucket prediction in the confusion matrix.

6.1. Logistic Regression

6.1.1. Background

Logistic Regression classification is one of the classification modelling methods used in machine learning that is used for binary classification (0,1). The assumptions that need to be considered for Logistic regression are:

1. Response Variables are in binary form (0,1).
2. The observations in the dataset should not be the same individual or related to each other in anyways
3. There is no multicollinearity among explanatory variables.
4. There should not be extreme outliers or influential observations in the dataset.
5. The sampling size is sufficiently large.
6. There is linear relationship between explanatory variables and the logit of the response variable.

Sigmoid / Logistic Regression function

The Sigmoid Function squishes all its inputs (values on the x-axis) between 0 and 1 as ready to see on the y-axis within the chart below. The sigmoid function formula is given as below.

$$g(z) = \frac{1}{1 + e^{-z}}$$

The run of inputs for this work is the set of all Genuine Numbers and the extend of yields is between and 1.

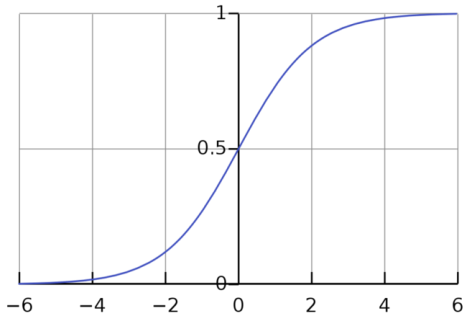

Figure 6.2: Sigmoid function plot.

We can see in figure 6, we can say that as z increments towards positive interminability the yield gets closer to 1, and as z diminishes towards negative limitlessness the yield gets closer to 0.

The Hypothesis for the Linear regression is given below. The output of the linear regression is the set of real numbers.

$$z = w.X + b$$

where,

w = slope of y/slope of x

b = intercept

X = Independent variable

The equation for the logistics regression is given below. The output of this ranges between 0 and 1 because by applying the sigmoid function, we get the output ranging between 0 and 1

$$y = \text{sigmoid}\ (w.X + b)$$

Substituting sigmoid equation in the logistic regression equation, we get the following equation:

$$\frac{1}{1 + e^{-(w \cdot x + b)}}$$

Where,

z = w.X +b

The sigmoid function z is being replaced with the hypothesis of logistics regression equation

Loss/Cost Function

For a binary classification issue, we ought to be able to yield the likelihood of y being 1 or 0.

The likelihood of y is obtained from the sigmoid function and the cost is derived using the below formula.

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)})]$$

Where,

m = number of samples in training data.

y = actual outcome

$\hat{y}$ = Predicted outcome

The above cost function will be maximum for incorrection prediction, and the cost function becomes zero for correct prediction.

The logistic regression uses a gradient descent algorithm to determine parameters that yield minimum cost on the trained data.

6.1.2. First Iteration

At first, we run the logistic regression model for the application bucket prediction using the glm function on the training dataset and use the prediction model on the test data set to see the output of the prediction model.

6.1.3. Second Iteration

We used the polynomial and other transformation of the variable in this iteration. We form the receiver operating characteristic curve (ROC curve is a graphical representation of a binary classifier system's diagnostic capacity when its discriminating threshold is changed) to help us find the threshold of the model. The ROC curve of application and the behavioural prediction model is given below:

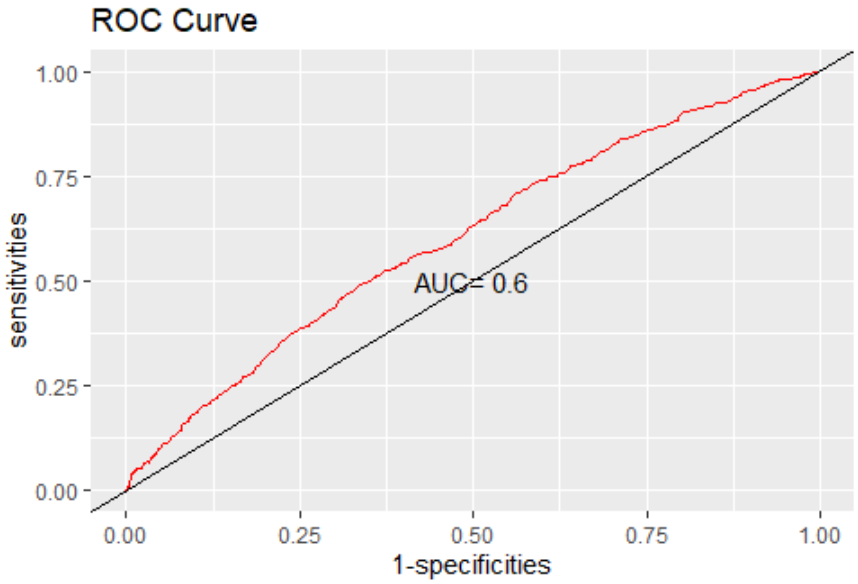

Figure 6.3: ROC curve to find the threshold for the application model.

For the ROC curve we find that the threshold is 0.2577458. Using the threshold value (0.2577458) we develop the logistic regression model to predict the confusion matrix for the application prediction model.

From the confusion matrix we can calculate the sensitivity, specificity and accuracy of the model using the formula mentioned in the confusion matrix section.

For bucket prediction of multi-class variable is not possible logistic regression cannot solve multi-class problem, so we had to use linear regression to help us in analysing the multi-class problem. We first found out the ROC curve for the model and the threshold was 0.8041307.

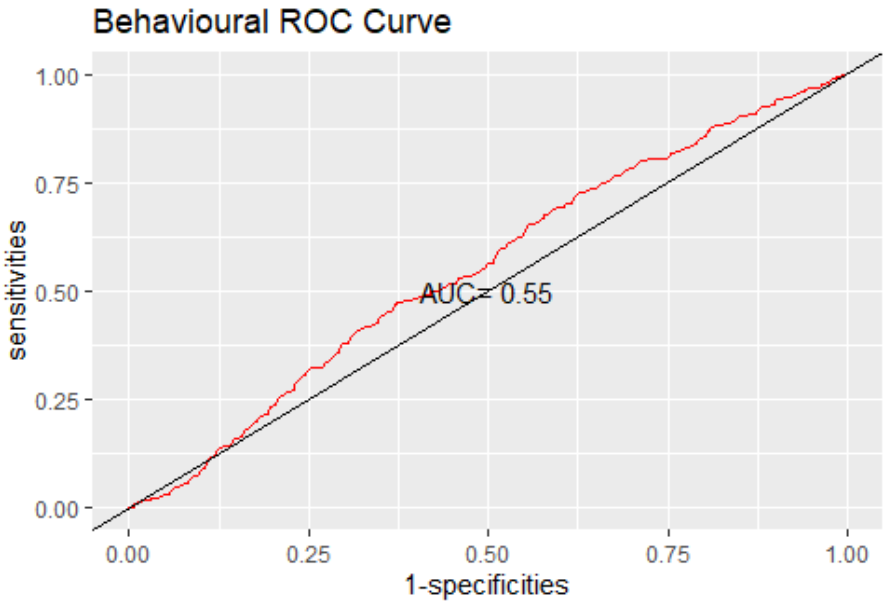

Figure 6.4: ROC curve for Bucket prediction model.

Using the threshold, we find the confusion matrix. Through that we will be finding out the sensitivity, specificity, profit/ loss and the accuracy of the model.

6.2. Linear Regression

6.2.1. Background

Linear regressions are one of the generally used statistical techniques. According to Abdou and Pointon (2011), linear regression is the methods that explain the connection between a predictand variable and one or more predictor variables in any data analysis. Also, the association between a quantitative predictand variable and two or more predictor variables in data analysis. Moreover, using graphs, such as residual vs fitted, normal q-q plot and leverage, can check the assumptions of the linear regression.

6.2.2. Project

In the project, six linear regression models with 24 variables from the dataset in Zero-One Classification, Bucket Prediction, and Profit/Loss Prediction will be provided. This section is divided into four parts: dependent variable, independent variable, formulation and each of the steps.

6.2.3. Dependent variable

Each of the models has different dependent variables. For our dataset, bucket is present to be the dependent variable. Bucket in Zero-One Classification, Bucket Prediction, and Profit/Loss Prediction is divided into different levels. In zero-one classification, All_Max_Bucket_Flag_updated as an dependent variable into 0 (good customers) and 1 (bad customers) classification. As for the bucket, All_Max_Bucket_updated as an dependent variable divided into the range from 0 to 7. For Profit/Loss Prediction, PNL_All_Max_Bucket as an dependent variable where 0-1 is good clients and 2-7 refer to bad client credit classes.

6.2.4. Independent variable

Independent variables refer to different type of scoring. For Application scoring, Branch_code_Rating, Age, Membership_Date, Sex_M_Flag and Out_Bank_Flag have been selected. As for Behavioural scoring, Attrition_Flag, Blacklist_Flag, Writeoff_Flag, Spending_Limit, Avg_Payments, Avg_Balances, Account_Status_Bucket, Current_Balance, Branch_Code_Rating, Out_Bank_Flag, Age_Issued_Updated, and Sex_M_Flag are selected in this project.

6.2.5. Formulation

The linear regression model is formulated as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots\ldots\ldots\ldots + \beta_i X_i + \epsilon$$

where

- Y is the bucket
- X_i is the explanatory factor i
- β_i is the regression coefficient of the explanatory factor i
- i is the number of explanatory variables

In the project, becomes

1. Application scoring in zero-one classification:

$$\text{All\_Max\_Bucket\_Flag\_updated} = \beta_0 + \beta_1 Branch\_Code\_Rating + \beta_2\, Age + \beta_3 Membership\_Date + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \epsilon$$

2. Application scoring in bucket prediction:

$$\text{All\_Max\_Bucket\_updated} = \beta_0 + \beta_1 Branch\_Code\_Rating + \beta_2\, Age + \beta_3 Membership\_Date + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \epsilon$$

3. Application scoring in profit and loss prediction:

$$\text{PNL\_All\_Max\_Bucket} = \beta_0 + \beta_1 Branch\_Code\_Rating + \beta_2\, Age + \beta_3 Membership\_Date + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \epsilon$$

4. Behavioural scoring in zero-one classification:

$$\text{All\_Max\_Bucket\_Flag\_updated} = \beta_0 + \beta_1 Spending\_Limit + \beta_2\, Blacklist\_Flag + \beta_3 Membership\_Date + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \beta_6 Attrition\_Flag + \beta_7 Writeoff\_Flag + \beta_8 Account\_Status\_Bucket + \beta_9 Age + \beta_{10} Branch\_Code\_Rating + \epsilon$$

5. Behavioural scoring in bucket prediction:

$$\text{All\_Max\_Bucket\_updated} = \beta_0 + \beta_1 Spending\_Limit + \beta_2\, Blacklist\_Flag + \beta_3 Membership\_Date + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \beta_6 Attrition\_Flag + \beta_7 Writeoff\_Flag + \beta_8 Account\_Status\_Bucket + \beta_9 Age + \beta_{10} Branch\_Code\_Rating + \epsilon$$

6. Behavioural scoring in profit and loss prediction:

$$\text{PNL\_All\_Max\_Bucket} = \beta_0 + \beta_1 BSpending\_Limit + \beta_2\, Blacklist\_Flag + \beta_3 Membership\_Date + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \beta_6 Attrition\_Flag + \beta_7 Writeoff\_Flag + \beta_8 Account\_Status\_Bucket + \beta_9 Age + \beta_{10} Branch\_Code\_Rating + \epsilon$$

6.2.6. Each of the steps

There are four steps that shows the process of the multiple linear regressions.

Firstly, the missing values in the dataset are replaced by mean. Then, backward stepwise algorithm has been used to find the best variables for the model based on R-square and p value. Meanwhile, with a view to better fitting, checking the polynomial degree for some variables is necessary. Also, we tried

SMOTE to handle the imbalanced class problem. Next, We checked for four assumptions for the linear model as following:

A. Residual vs fitted: To check for the linear relationship assumption.
B. Normal q-q plot: To examine whether the residuals are normally distributed. The graph shows that residuals are not normally distributed.
C. Scale location: To check the homogeneity of variance of the residuals (homoscedasticity). This is not the case in our example, the graph shows that we have a heteroscedasticity problem.
D. Residual vs leverage: Used to identify influential cases, that is extreme values that might influence the regression results. There are few influential points present in the below graph.

Finally, using different thresholds testing to find the best model in good accuracy, Profit and Loss.

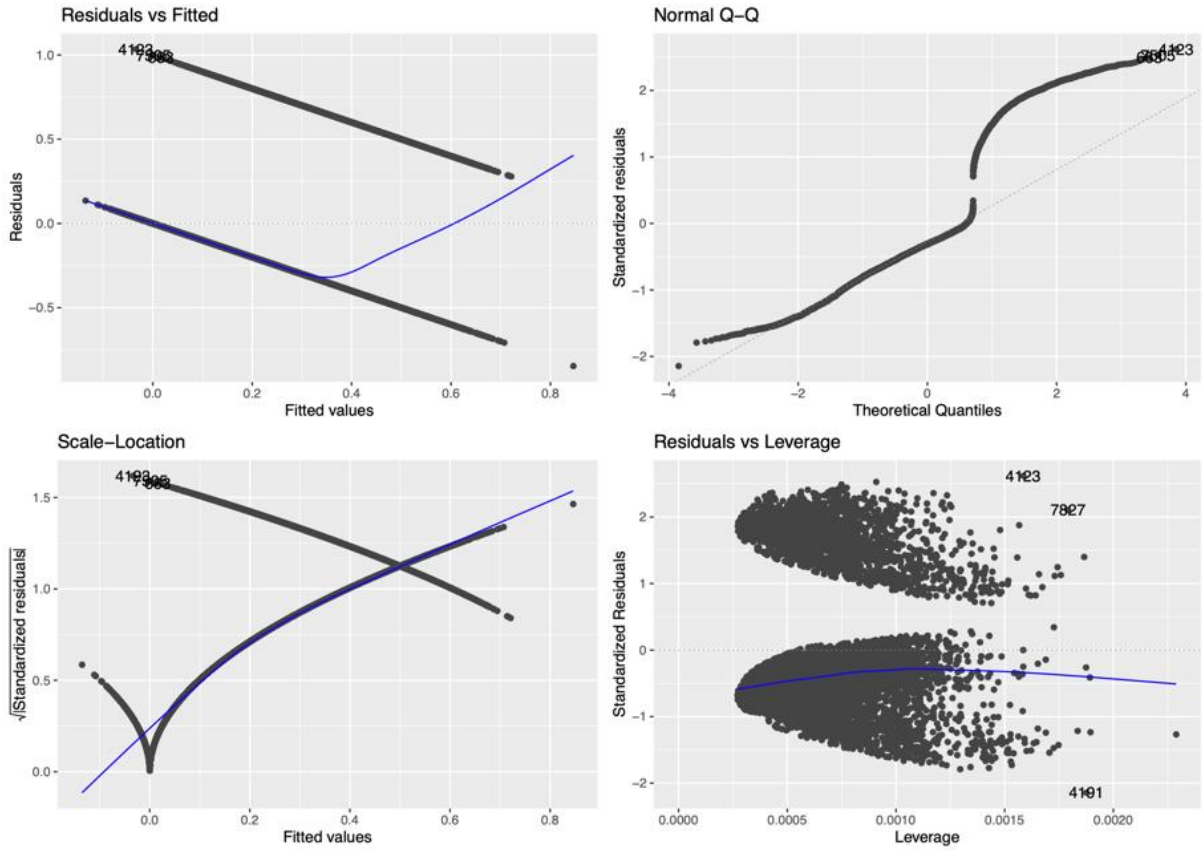

Figure 6.5: Four assumptions for the linear model

6.3. Decision Tree classifier (Classification And Regression Trees)

6.3.1. Background

Classification And Regression Trees (CART) is the algorithm that based on the decision tree. The function of the CART algorithm is that by dividing the data into each node, this binary tree process is one of the widespread methods of credit evaluation to be used in a single input field in decision tree (Bensic, Sarlija and Zekic-Susac, 2005).

6.3.2. Project

In the project, six CART models with 24 variables from the dataset in Zero-One Classification, Bucket Prediction, and Profit/Loss Prediction will be presented. This section is divided into four parts: dependent variable, independent variable, formulation and each of the steps.

6.3.3. Dependent variable

There are two dependent variables in this project. All_Max_Bucket_Flag_updated and PNL_All_Max_Bucket are presented as a dependent variable in Zero-One Classification, Bucket Prediction, and Profit/Loss Prediction classification. These two dependent variables have been divided into different level of classification. 0 (good customers) and 1 (bad customers) buckets are in the zero-one classification. For the Bucket Prediction and Profit/Loss Prediction, the range is from 0 to 7 where 0-1 is good clients and 2-7 refer to bad client credit classes.

6.3.4. Independent variable

Two types of scoring, application and behavioural, are selected in different independent variables. For Application scoring, Branch_code_Rating, Age, Membership_Date, Sex_M_Flag and Out_Bank_Flag have been selected. As for Behavioural scoring, Attrition_Flag, Blacklist_Flag, Writeoff_Flag, Spending_Limit, Avg_Payments, Avg_Balances, Account_Status_Bucket, Current_Balance, Branch_Code_Rating, Out_Bank_Flag, Age_Issued_Updated, and Sex_M_Flag are in this project.

6.3.5. Formulation

Back to the project, the CART model is formulated as

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots\ldots\ldots\ldots + \beta_p X_p + \epsilon$$

where

- Y is the bucket
- X_p is the explanatory factor p and characteristics
- β_i is the regression coefficient of the explanatory factor $i,\ i = 1,2,\dots,p$
- p is the number of explanatory variables and attributes

1. Application scoring in zero-one classification:

$$\text{All\_Max\_Bucket\_Flag\_updated} = \beta_0 + \beta_1 Branch\_Code\_Rating + \beta_2\, Age + \beta_3 Blacklist\_Flag + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \epsilon$$

2. Application scoring in bucket prediction:

$$\text{All\_Max\_Bucket\_updated} = \beta_0 + \beta_1 Branch\_Code\_Rating + \beta_2\, Age + \beta_3 Blacklist\_Flag + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \epsilon$$

3. Application scoring in profit and loss prediction:

$$\text{PNL\_All\_Max\_Bucket} = \beta_0 + \beta_1 Branch\_Code\_Rating + \beta_2\, Age + \beta_3 Blacklist\_Flag + \beta_4 Sex\_M\_Flag + \beta_5 Out\_Bank\_Flag + \epsilon$$

4. Behavioural scoring in zero-one classification:

$$\text{All\_Max\_Bucket\_Flag\_updated} = \beta_0 + \beta_1 Spending\_Limit + \beta_2\, Blacklist\_Flag + \beta_3 Avg\_Balances + \beta_4 Ytd\_Amt\_Cash + \beta_5 Current\_Balance + \epsilon$$

5. Behavioural scoring in bucket prediction:

$$\begin{aligned}\text{All\_Max\_Bucket\_updated} = {} & \beta_0 + \beta_1 Spending\_Limit + \beta_2\, Blacklist\_Flag \\ & + \beta_3 Current\_Balance + \beta_4 Avg\_Payments + \beta_5 Avg\_Balances \\ & + \beta_6 Attrition\_Flag + \beta_7 Writeoff\_Flag + \beta_8 Account\_Status\_Bucket \\ & + \beta_9 Py\_Amt\_Cash + \beta_{10} Ytd\_Amt\_Cash + \beta_{11} Oy\_Amt\_Cash + \epsilon\end{aligned}$$

6. Behavioural scoring in profit and loss prediction:

$$\begin{aligned}\text{PNL\_All\_Max\_Bucket} = {} & \beta_0 + \beta_1 Spending\_Limit + \beta_2\, Ytd\_Max\_Bucket\_Flag \\ & + \beta_3 Current\_Balance + \beta_4 Avg\_Payments + \beta_5 Avg\_Balances \\ & + \beta_6 Attrition\_Flag + \beta_7 Writeoff\_Flag + \beta_8 Account\_Status\_Bucket \\ & + \beta_9 Py\_Amt\_Cash + \beta_{10} Ytd\_Amt\_Cash + \beta_{11} Py\_Max\_Bucket\_Flag + \epsilon\end{aligned}$$

6.3.6. Each of the steps

This section is divided into six parts: clean the dataset, using function, plot, pruning the tree, predicting model, ROC curve, and K-fold Cross-Validation.

6.3.6.1. Clean the dataset

In order to pre-process data, the average values are replaced for missing value into the dataset. Besides, plotting the correlation and coefficients for independent variables is one of the methods to find the proper variables for the model. Next, some of the independent variables must convert the attribute of binary flag into factor such as: SexMFlag, MaxBucketFlag, BlacklistFlag, OutBankFlag, AttritionFlag and WriteoffFlag. It is because the dependent variables need to identify the data as factor attribute.

6.3.6.2. Using function

Using the rpart() function, specifying the model formula, data, and arguments. The process of this function is tried to provide the best result from splitting each node until reaching the requirements of the end node. This limited sub-sets division process begins with a root node and then classify into "good" and "slow" repay classes in credit scoring dataset. Each node splits into two subsets and the winning of the sub-tree is from the decision algorithm which the model searched all the optimized split. This sub-tree with two classes is based on the overall error rate and lowest misclassification cost (Biermann et al., 1984; Thomas, 2000). Besides, based on Li and Zhong (2012) study, there are three elements in the construction of a decision tree:

1) Bifurcation criterion: the function of this criterion is to divide new subsets

2) Stopping criterion: determine whether the subset is an end node or not

3) and the criterion deciding which class the terminal node belongs to.

6.3.6.3 Plot

Using *rparty.tree* to plot the tree to develop the detail of each node in the growing tree. In this Figure 6.6, Writeoff_Flag as a root node. This root node divides into two child nodes. This procedure will be repeated until the stopping criterion (Bastos, 2008).

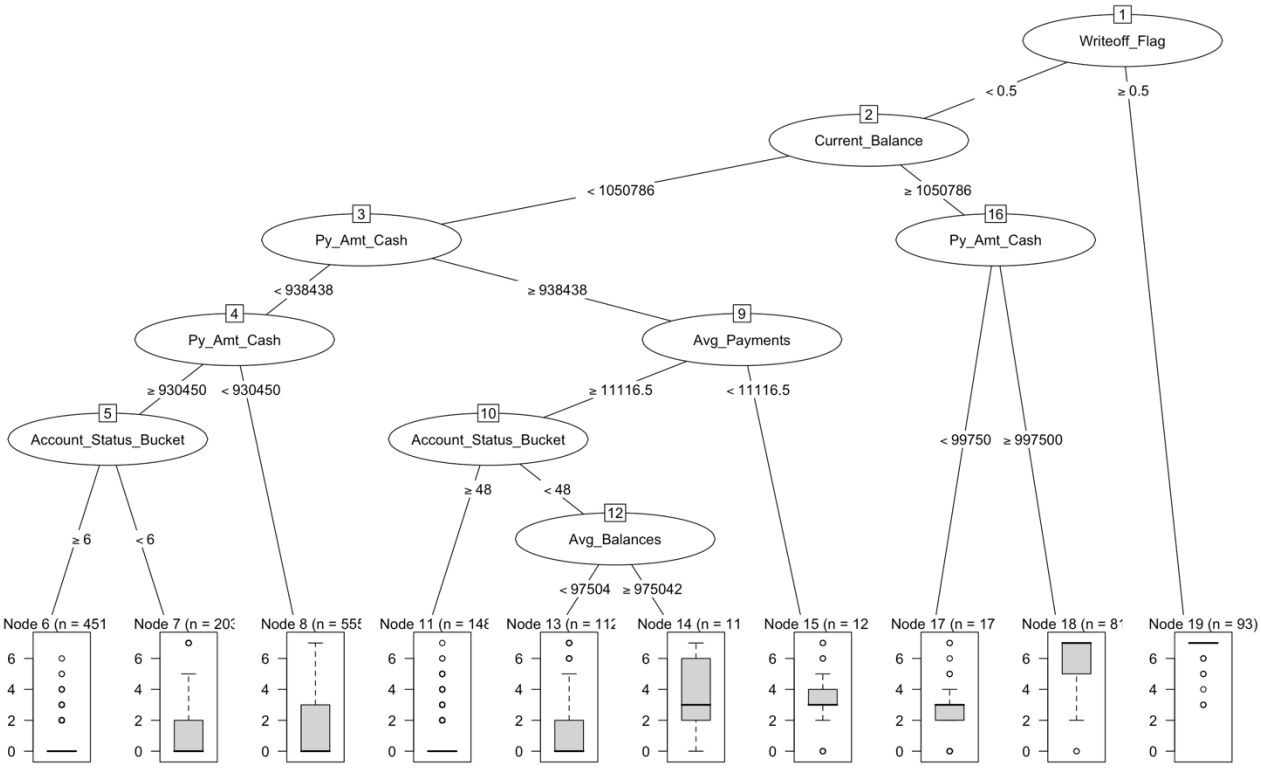

Figure 6.6: CART tree from Behavioural scoring in Bucket Prediction model

6.3.6.3. Pruning the tree

The aim to increase accurate predictions is by pruning the growing tree. This pruning tree procedure is to check the cp (complexity parameter) value. This value is to find the optimal size of tree. Moreover, the purpose of pruning is to balance complexity and accuracy which need to come up with a least additional predictive power per leaf by identifying the branches (Breiman et al. 1984 in Galindo, Tamayo, 2000). Then predict the model.

6.3.6.4. Predict the model

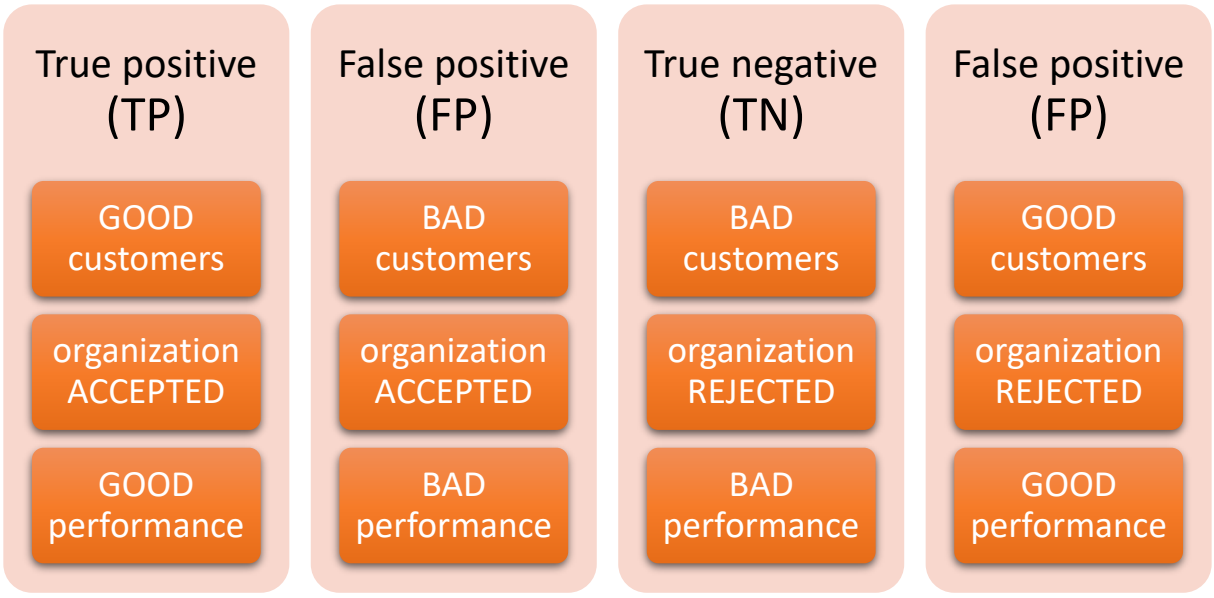

Figure 6.7. The result of Application scoring
Source: Brainbravo Ltd.

From figure 6.7, it is clear that the credit scoring model predicts the acceptance of a good customer in true positive (TP) classification and rejects bad customers in true negative (TN) classification. It should avoid false positives (FP) where the client is uncreditworthy, but the bank predicts the client as a good customer. Also, it should be aware of false negatives (FN) where the client is creditworthy, but the bank predicts the client as a bad customer. The aim of developing a model is to increase the TP and to lower FP.

6.3.6.5. ROC Curve

In order to provide the best threshold, using ROC curve to visualizes a model performance at different threshold values. Also, check the area under the curve (AUC), which quantifies the performance of the model (Figure 6.8). A perfect classifier of the AUC is 1. And if the value is 0, it means that the predictions are completely incorrect. Next, put the threshold into the training and test dataset to get the confusion matrix, so that accuracy, sensitivity, specificity and average profit/loss can be derived.

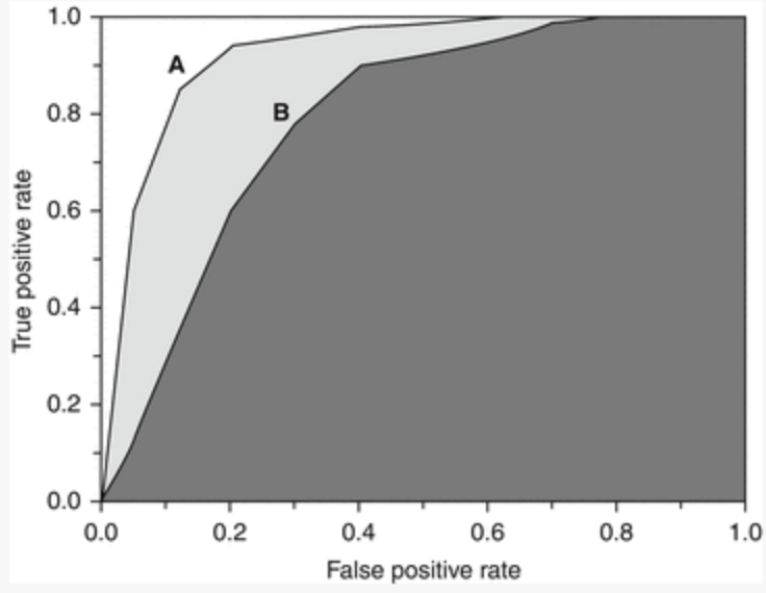

Figure 6.8.: AUC for two different classifiers

Source: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-9863-7_209

6.3.6.6. K-fold Cross-Validation

Overfitting is important in this step. The problem of overfitting models examines in this project via looking for the low bias and variance. Thus, use K-fold Cross-Validation (k = 10) is to correct overfitting and to check the prediction's accuracy. Ideally, the result of the accuracy should be same as after the prune tree.

In finial steps, the best accuracy and good average profit/loss can be selected.

6.4. Random Forest

6.4.1. Background

Random forest classification is one of the classification modelling methods used in machine learning. The random forest calculation works by aggregating the expectations made by different choice trees of changing depth. Each choice tree within the forest is prepared on a subset of the dataset called the bootstrapped dataset.

when choosing the criteria with which to part a decision tree, we measured the impurity delivered by each include utilizing the Gini index or entropy. In random forest, however, we haphazardly select a predefined number of features as candidates. The latter will result in a bigger change between the trees that would something else contain the same features (i.e those which are exceedingly correlated with the target label).

When the random forest is utilized for classification and is displayed with an unused sample, the ultimate expectation is made by taking the larger part of the predictions made by each individual decision tree

within the forest. Within the event, it is used for regression, and it is displayed with an unused test, the ultimate prediction is made by taking the average of the expectations made by each person's decision tree within the forest as shown in the diagram below.

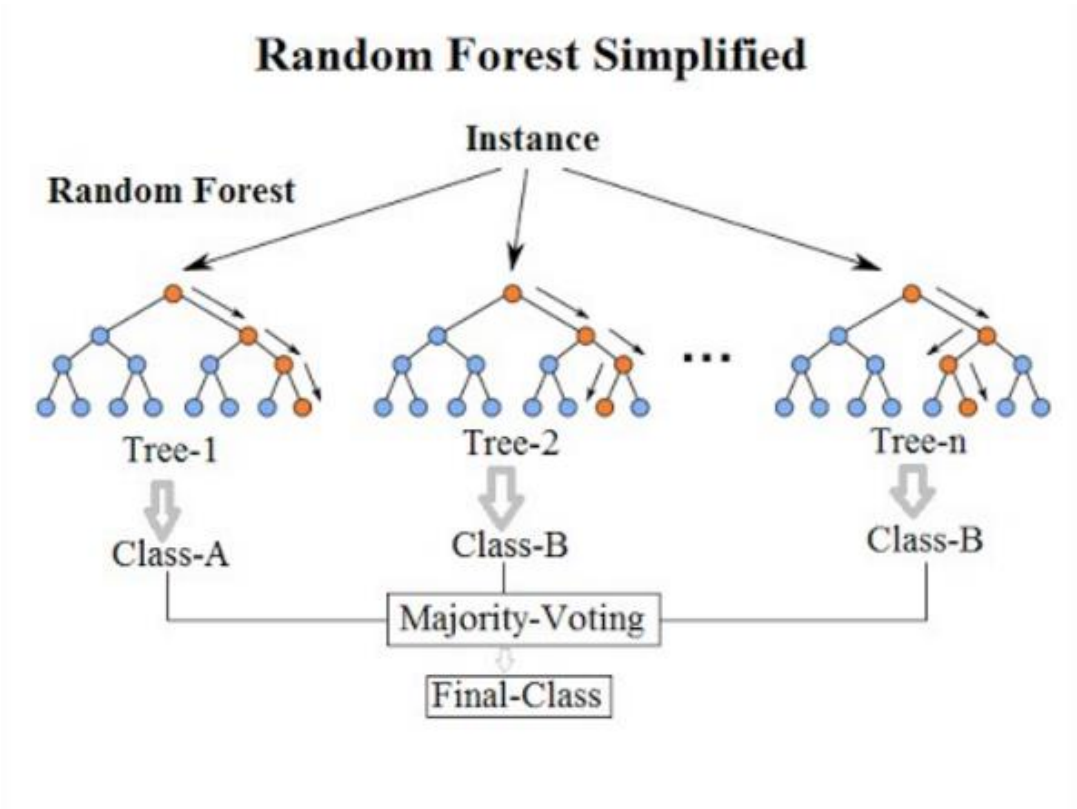

Figure 6.9: Random Forest.

Source:
https://en.wikipedia.org/wiki/Random_forest#/media/File:Random_forest_diagram_complete.png

We have 24 variables available post pre-processing stage of the data set. Now we have to train 6 random forest models. There are 3 categories, they are Zero-One Classification, Bucket Prediction, and Profit/Loss Prediction and each of these categories has two subcategories that is customers application bucket prediction model and customer behavioural bucket prediction model. For predicting the customer application and customer behavioural bucket prediction we use different independent and dependent variables based on the categories we are working on.

The Dependent variables used in the three categories are different based on the requirement. The Zero-One classification has All_Max_Bucket_Flag_updated this variable is used in predicting the bucket in 0's and 1's. The Behavioural classification has a dependent variable as All_Max_Bucket_updated, this is used to get the exact classification of how many customers fall in each bucket and the Profit/Loss classification has PNL_All_Max_Bucket as a dependent variable where this is used to calculate the profit / loss generated when the customers are given the credit card is done based on

The independent variable varies as the application model contains variables that we get when the customer fills the application like Sex, Date of birth, making automatic payment from the bank account and the branch code from where they have opened their account. The independent variable consists of the variable which describes the transaction and repayment behaviour of a customer. They are mentioned in the below table.

Categories	Sub-Categories	Dependent Variable (Y)	Independent variable (X)
Zero-One Classification	Application	All_Max_Bucket_Flag_updated	'Sex_M_Flag', 'Blacklist_Flag', 'Out_Bank_Flag', 'Age_Issued_Updated', 'Branch_Code_Rating'
	Behavioural	All_Max_Bucket_Flag_updated	'Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag', 'Out_Bank_Flag', 'Sex_M_Flag', 'Age_Issued_Updated', 'Branch_Code_Rating'
Bucket Prediction	Application	All_Max_Bucket_updated	'Sex_M_Flag','Blacklist_Flag', 'Out_Bank_Flag', 'Branch_Code_Rating'
	Behavioural	All_Max_Bucket_updated	'Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'Account_Status_Bucket', 'Current_Balance', 'Branch_Code_Rating', 'Out_Bank_Flag', 'Age_Issued_Updated', 'Sex_M_Flag'
Profit/Loss Classification	Application	PNL_All_Max_Bucket	'Sex_M_Flag','Blacklist_Flag', 'Out_Bank_Flag', 'Branch_Code_Rating'
	Behavioural	PNL_All_Max_Bucket	'Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'Account_Status_Bucket',' Current_Balance', 'Branch_Code_Rating', 'Out_Bank_Flag', 'Age_Issued_Updated', 'Sex_M_Flag'

Table 13: Categories and sub-categories with dependent and independent variable.

6.4.2. First Iteration (general approach)

At first all the independent variables were used for modelling for predicting bucket number for the customer's application and customers behavioural. Using the confusion matrix we find the sensitivity, specificity, profit/loss this model, and the accuracy of the model. The formula is mentioned in the confusion matrix section.

6.4.3. Second Iteration

For the second iteration, The Values in all the variables need to be factorised and converted to the categorical values 'Sex_M_Flag', 'Blacklist_Flag', 'Out_Bank_Flag', 'Age_Issued_Updated', 'Branch_Code_Rating', 'Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'Account_Status_Bucket', 'Current_Balance', 'Branch_Code_Rating', 'Out_Bank_Flag', 'Age_Issued_Updated', 'Sex_M_Flag'. After making the changes we run the code again out this procedure we carried out the regressing. We get the following result.

Type of Scoring	Model	sqrt(sensitivity*specificity)	Average_Profit/Loss	Accuracy
Application	Zero-one Classification	6.64%	1.4700%	76.03%
Application	Bucket Prediction	0.00%	0.0200%	75.99%
Application	Proft/Loss Prediction	0.00%	0.0200%	75.99%
Behavioral	Zero-one Classification	23.65%	0.9100%	77.34%
Behavioral	Bucket Prediction	47.17%	0.0079%	80.80%
Behavioral	Proft/Loss Prediction	47.60%	0.0080%	80.80%

Table 14: Result of the random forest classification model in terms of accuracy, profit/loss, sensitivity, and specificity.

6.4.4. Third Iteration (Parameter tuning)

In third iteration, we try to tune the randomForest by adding new parameter mtry vary by inserting the number starting from 1 till the number is equal to the number of independent variables used in the modelling.

Mtry is Number of variables randomly sampled as candidates at each split. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times. Post running the code and varying the mtry we get the following result.

The **ntree** is Number of trees to grow. The default method was used with default ntree = 100. Increasing the number of trees number the answer didn't not vary so we decide to give the default number as 100.

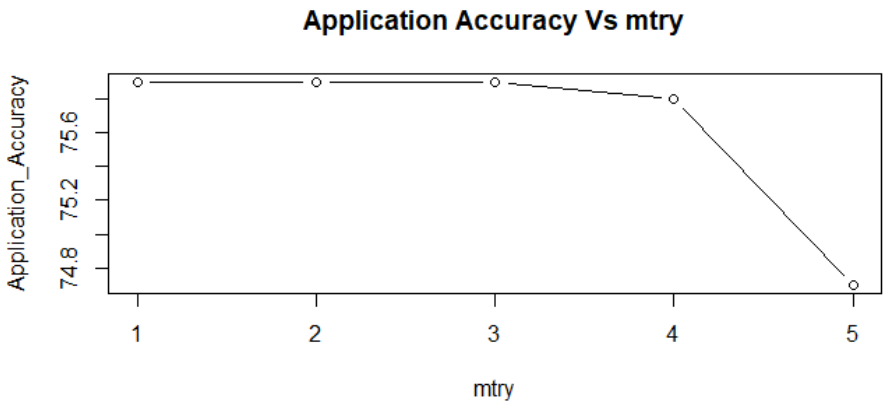

Figure 6.10: The accuracy of the application model vs mtry for zero-one classification.

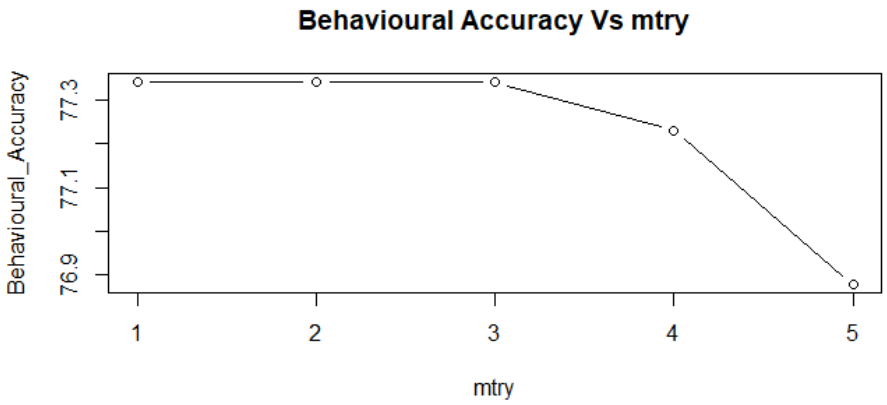

Figure 6.11: The accuracy of the behavioural model vs mtry for zero-one classification.

mtry	Behavioural Accuracy	Application Accuracy
1	77.34	75.90
2	77.34	75.90
3	77.34	75.90
4	77.23	75.80
5	76.88	74.70

Table 15: result by varying the mtry parameter and its accuracy of the model.

From the above table we can conclude that the mtry with 1 to 3 has the same accuracy in application and behavioural model we consider mtry as 1 as it has the highest accuracy of the model.

Training the data and predicting the output using random forest algorithm code has been attached in the appendix.

7. Performance metrics.

The most challenging part of the modelling activity is to decide the best model. Hence, it becomes crucial to determine the parameters for model evaluation before starting the model development since different methods can provide different perspectives. Given the problem at hand, there are various measures available to evaluate the model, such as precision, accuracy, F1 Score, etc. These are the most widely used and recognized metrics for evaluating classification models. Each of them uniquely assesses the model. For this activity model, accuracy, square root of sensitivity and specificity and final profit/loss metrics were considered to model evaluation. Accuracy reveals how well the model can predict, while sensitivity/specificity shows how sensitive the model is to positive/negative classes. Brainbravo suggested breaking down the issue into three significant parts for each application and behavioural scoring for the final profit and loss calculation.

1. Zero One Classification Task (All max flag)
2. Bucket Prediction Task (All max bucket)
3. Profit / Loss Prediction (Assigned PnL for each bucket)

Zero One Classification Task (All max flag): For both application and behavioural scoring, "All max flag" was created by bucketing 0 or 1 to Group=0 "good customer", whilst the others will be grouped to Group=1 "Bad customer". A bad customer is assigned a unit loss of (-45%), and each Good customer is assigned a unit profit of 2%. If a prediction model predicts that a customer is good and if the customer is really good they assign a score of 2% to that but if a customer is really bad then assign a score of (-45%). This is the profit or loss that a Bank would incur by accepting a Good or Bad customer, respectively. If a prediction model predicts that a customer is bad, then assign a score of 0. This means that by rejecting a customer, a bank will incur neither a profit nor a loss. The average profit/loss is the profit per good category multiplied by the percentage of that category in the sample. A model that accepts all customers is the so-called default model, and its performance is the average of scores from accepting all good and bad customers. So, it's not just model parameters such as accuracy and specificity which matters, but also the final profit/loss becomes an important aspect to consider for model evaluation. Setting up a suitable threshold to categorize the customer as good or bad is very important to approach this problem. We have tested various thresholds for all the models building activities to find out the model performance. For example: In linear regression, the output predicted value varies between 0 and 1, so we have created different thresholds (cut-off) and check the model performance on selected parameters:

	Cut-off	Accuracy	Sq_sen_spec	PnL
1	0.1	**0.44**	0.52	**0.68%**
2	0.2	0.62	0.68	-2.12%
3	0.3	0.72	0.67	-4.74%
4	0.4	0.77	0.59	-6.40%
5	0.5	0.77	0.41	-8.04%
6	0.6	0.77	0.25	-8.84%
7	0.7	0.76	0.04	-9.28%

Table 16: Result for Zero- One Classification.

Cut-off - Criteria to classify the predicted value as 0 or 1. For example, a cut-off of 0.3 means that values greater than 0.3 are classified as 1 while the rest are classified as 0.

Accuracy: Ratio of correctly predicted observation to the total observations

Sq_sen_spec: Square root of (Sensitivity * specificity). It reveals how sensitive the model is to positive/negative classes. Sensitivity (TPR) measures how well a test can identify true positives, and specificity (TNR) measures how well a test can identify true negatives.

PnL: Profit and loss calculated based on the above formulae.

The above table results show that the profit and loss would be highest if we use a cut-off value of 0.1. In this scenario, a very few observations will be classified as 0. Since the number of observations is very high for the "0 bucket", the probability of a customer being good is very high. This means that the model has predicted good customers as good but many good customers as bad customers, too, resulting in bad accuracy. But the rule is to penalize only the bad customer classified as a good customer, hence profits are much higher for the low cut-off/threshold values. Though the result looks good, setting a low threshold might give drastic results since the data distribution can change based on the sampling techniques. Hence, we should consider the cut-off, which should have good profit and good accuracy, sensitivity, and specificity. Therefore, it makes more sense to use the cut-off of 0.3 with an accuracy of 72% but PnL of (-4.74%).

Bucket Prediction Task (All max bucket): The dependent variable is "All-max-bucket" with values from 0 to 7. To calculate the profit and loss in the approach, bad customers are assigned bucket values (i.e.-7...-2) based on their respective buckets and the good customers are assigned higher bucket values (i.e. Buckets -1 and 0). If a prediction model predicts that a customer is good, assign the value corresponding to their bucket. This is the corresponding profit/loss equivalent that a bank expects by deciding to accept a customer. If a prediction model predicts that a customer is bad, then assign a score of 0. The average profit/loss is the profit per good Category * % of that Category in the sample. We have come up with different techniques to improve the bucket selection for maximizing profits. We have used different approaches based on the algorithm, such as in linear regression, we have used 2 strategies. 1) Directly using the predicted bucket as classified bucket 2) Using the proportion of original data set's bucket sizes to create new predicted buckets. Ex.: if there are 76% customers from "0 bucket" in the primary dataset, then classify the bottom 76 percentile predicted customers as 0 in the test dataset. Based on this, we got the following results:

Criteria	Sq_sen_spec	PnL
Actual	**0.58**	**-0.521**
Percentile	0.63	-0.45349

Criteria – The approach used Actual means assigning the predicted bucket based on the actual values while percentile means based on the percentile approach discussed above.

Sq_sen_spec: Square root of (Sensitivity * specificity). It reveals how sensitive the model is to positive/negative classes. Sensitivity (TPR) measures how well a test can identify true positives, and specificity (TNR) measures how well a test can identify true negatives.

PnL: Profit and loss calculated based on the above formulae.

In this case, it can be seen that the percentile approach hasn't performed that well.

This is because in the actual model, very few observations (marked in green) predicted as "0 – good customers" actually belongs to higher "4-7 buckets". But, in the percentile approach, there are some observations (marked in red), which are predicted as "0 bucket" actually belongs to higher "4-7 buckets". This has resulted in the heavy penalization for these observations, resulting in the overall profit/loss calculation drop.

	Actual Bucket Classification						
	0	**1**	**2**	**3**	**4**	**6**	**7**
0	1049	963	136	1	1	0	0
2	47	183	58	1	0	0	0
3	12	138	51	0	0	0	0
4	0	51	15	1	0	0	0
5	0	10	14	1	0	0	5
6	0	8	5	0	1	0	1
7	1	17	26	1	0	8	24
	Percentile Approach Classification						
	0	**2**	**3**	**4**	**5**	**6**	**7**
0	1824	165	99	18	25	17	2
2	177	45	37	16	6	7	1
3	104	39	31	9	7	11	0
4	27	18	15	2	4	1	0
5	3	7	6	5	3	0	6
6	5	3	3	0	1	1	2
7	10	6	7	6	11	5	32

Table 17: Count of customers in each bucket both in application and Behavioural

Profit / Loss Prediction (Assigned PnL for each bucket): In this approach, the dependent variable is the profit/loss corresponding to the "All max bucket" variable. As provided by Brainbravo, we have assigned the following unit Profit/Loss values to every bucket:

All Max Bucket 0 ->+3%,

All Max Bucket 1->+1%,

All Max Bucket 2->0%,

All Max Bucket 3 ->-15%,

All Max Bucket 4->-25%,

All Max Bucket 5-> -35%,

All Max Bucket 6-> -45%,

All Max Bucket 7 -> -65%

If a prediction model predicts that a customer is good, assign the value corresponding to their bucket. This is the corresponding profit/loss equivalent that a bank expects by deciding to accept a customer. If a prediction model predicts that a customer is bad, then assign a score of 0. The average profit/loss is the profit per good Category * % of that Category in the sample. In this case also, we have tried different approaches such as percentile-based bucketing, etc., to find out the best models having good accuracy and good profit/loss value. The best results using these criteria have been discussed in the result sheet provided in the later section of this report.

8. Results

8.1. Application Scoring

The variables Brainbravo suggested in the Application Scoring section are: 'All_Max_Bucket_Flag_updated', 'All_Max_Bucket_updated', 'Sex_M_Flag', 'Blacklist_Flag', 'Out_Bank_Flag', 'Age_Issued_Updated', 'PNL_All_Max_Bucket.

In the analysis of the application scoring part, we used four algorithms which are logistic regression, linear regression, decision tree and random forest for analysis. In this section, we will explain the result of application scoring of each algorithm.

8.1.1. Logistic Regression result

In this part, I will analyze the results of application scoring through logistic regression. Logistic regression is one of our commonly used data analysis tools, which can solve the problem that the dependent variable is a two-category categorical variable. Moreover, if the dependent variable is a multi-category categorical variable, it is difficult for us to solve the problem within the logistic regression. In this part, I will introduce multiple logistic regression and linear regression to help me solve the problem of multi-class categorical variables.

In this part, I will calculate different accuracy, sqrt(sensitivity*specificity) and average profit and loss according to different models.

8.1.1.1 0 and 1 classification model

In this part, we get the best model as below:

```
Call:
glm(formula = All_Max_Bucket_Flag_updated ~ Age_Issued_Updated +
    Sex_M_Flag + Branch_Code_Rating, family = "binomial", data = CS_Train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0243  -0.7768  -0.6709  -0.5209   2.0800

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        -0.038065   0.127660  -0.298    0.766
Age_Issued_Updated -0.016031   0.002582  -6.208 5.36e-10 ***
Sex_M_Flag          0.255541   0.056885   4.492 7.05e-06 ***
Branch_Code_Rating -0.268322   0.024740 -10.846  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9308.4  on 8457  degrees of freedom
Residual deviance: 9127.4  on 8454  degrees of freedom
  (18 observations deleted due to missingness)
AIC: 9135.4

Number of Fisher Scoring iterations: 4
```

Figure 8.1: Significant variable.

We can know the best AIC is 9135.4, the reason why we choose this one is because this model has the lowest AIC value. The significant variables are ‘Age_Issued_Updated’, ‘Sex_M_Flag’, ‘Branch_Code_Rating’. Then, we can get the final model equation is (All_Max_Bucket_Flag_updated/(1-All_Max_Bucket_Flag_updated))=-0.038065+Age_Issued_Updated*(-0.016031) +Sex_M_Flag*0.25541+Branch_Code_Rating*(-0.268322), which means the change in the log odds of All_Max_Bucket_Flag_updated per unit change in Age_Issued_Updated is -0.016031, Other variables also like Age_Issued_Updated.

Then, we can calculate the threshold through the ROC curve and classify the prediction of buckets according to the threshold. Among them, we assign 0 to the customer whose bucket’s range is from 0 to 2 and assign 1 to the customer whose bucket’s range is from 3 to 7. We can get the prediction matrix first:

prediction	
0	1
2819	2

Table 18: confusion matrix for Zero- One Classification.

After that, we can get the confusion matrix, as shown below：

	0	1
0	2143	1
1	676	1

Table 19: confusion matrix for Zero- One Classification.

Here, the confusion matrix I called f, and we can get accuracy through f.

accuracy=(f[1,1]+f[2,2])/(f[1,1]+f[2,2]+f[2,1]+f[1,2]), the result is 76%。

At the same time, we can also obtain the sensitivity and the specificity, where Sensitivity=f[2,2]/(f[2,2]+f[2,1]), and the result is 0.00148. Using the same way, we can get specificity=f[1,1]/(f[1,2]+f[1,1]), the result is 0.9995. Through sensitivity and specificity, we can calculate sqrt(sensitivity*specificity)=0.038.

According to the profit and loss coefficient provided by Brainbravo, we can know that if the customer's bucket is in the range which is from 0 to 2, then their profit and loss coefficient is 0.02. On the contrary, if their bucket is in the range which is from 3 to 7, then their coefficient is -0.45. According to our confusion matrix and profit and loss coefficient, we can get profit and loss= (2143/(2143+676))*0.02+(676/(2143+676))*-0.45. So, the result is -0.0927, which means that the company cannot be profitable in this case.

8.1.1.2 Bucket Prediction Model

First, we use multiple linear functions to calculate the sensitivity and specificity of the model. Because logistic regression is difficult to calculate the categorical variables of multiple classifications, we will use multiple linear regression for analysis. And after that, using multivariate logistic regression to calculate the profit and loss.

Through the confusion matrix, we can obtain sqrt(sensitivity*specificity) =0.285.

Finally, according to the coefficient of each bucket given by Brainbravo, we can know that when the value of our bucket is equal to 0, our coefficient is 0; when the bucket is equal to 1, our coefficient is -1. We can

check the table Brainbravo provided to know the coefficient of each bucket whose range is from 2 to 7, and the corresponding coefficient is equal to -1*(bucket). In the end, the result of profit and loss we get is 0, which means that this business will neither bring profit nor loss in the future.

8.1.1.3 Profit and Loss Prediction Model

First, we use the same method to get the confusion matrix and calculate the sensitivity and specificity of the model to get sqrt(sensitivity*specificity) =0.285.

Then we use the multiple logistic regression model to predict the bucket to which the new customer belongs in the future, and then multiplying the profit and loss coefficient by the predicted bucket. According to the coefficient of each bucket given by Brainbravo, we can know that when the value of the bucket is 0, our coefficient is 0.03; when the value of our bucket is 1, our coefficient is 0.01; when the value of our bucket is 2, our coefficient is 0; when the value of the bucket is 3, our coefficient is -0.15; when the value of the bucket is 4; our coefficient is -0.25; when the value of the bucket is 5, our coefficient is -0.35; when the value of the bucket is 6, our coefficient is -0.45; and when the value of the bucket is 7, our coefficient is -0.65. At last, the result of profit and loss we got was 0.03, which shows that our business can be profitable under this model.

8.1.2 Linear regression result

In this section, we will analyze the results of application scoring through Linear regression. Linear regression is one of the simplest data analysis tools. We only need to determine the independent variable and the dependent variable to find which independent variable has a significant impact on the dependent variable. At the same time, we can also predict the result of future customers' application scoring.

8.1.2.1 0 and 1 classification model

First, we calculated the threshold based on the customer's future bucket distribution. The dependent variable we use here is the All Max Bucket Flag, and its range is [0,1]. Simultaneously, we classify the predicted value of the customer's bucket according to the threshold. If the predicted value is greater than or equal to the threshold, we assign it to 1, otherwise assign it to 0. Then, we can also get the confusion matrix. Through the result of the confusion matrix, we can calculate the accuracy, sensitivity, and specificity. Among them, accuracy is equal to 76.5%, and through the sensitivity and specificity, we can get sqrt(sensitivity*specificity) =58.76%,

Then, we use the coefficient of profit and loss given by Brainbravo and the confusion matrix we calculated, we can figure out the profit and loss=-0.064. Here, we can know that under this model, this business will bring loss to the bank.

8.1.2.2 Bucket prediction model

First, we find the best model as below:

```
lm(formula = All_Max_Bucket_updated ~ Branch_Code_Rating + Age_Issued_Updated +
    Membership_Date + Sex_M_Flag + Out_Bank_Flag, data = df_train)

Residuals:
    Min      1Q  Median      3Q     Max
-3.2620 -0.8381 -0.3048  0.2228  6.4529

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       4.137e+01  1.034e+00  39.998  < 2e-16 ***
Branch_Code_Rating -1.091e-01  1.516e-02  -7.196 6.74e-13 ***
Age_Issued_Updated -1.290e-02  1.578e-03  -8.178 3.30e-16 ***
Membership_Date   -3.524e-03  9.121e-05 -38.635  < 2e-16 ***
Sex_M_Flag1        9.160e-02  3.442e-02   2.661   0.0078 **
Out_Bank_Flag1    -6.898e-02  3.250e-02  -2.122   0.0338 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.467 on 8470 degrees of freedom
Multiple R-squared:  0.1687,    Adjusted R-squared:  0.1682
F-statistic: 343.8 on 5 and 8470 DF,  p-value: < 2.2e-16
```

Figure 8.2: Significant variable.

We can know that the R squared is 0.1687, the reason why we choose this model is because this model has the highest R squared, although it's not so high, and all variables are significant. The equation is All_Max_Bucket_updated = 4.137e+01 + Branch_Code_Rating * (-1.091e-01) + Age_Issued_Updated*(-1.290e-02)+ Membership_Date*(-3.524e-03)+Sex_M_Flag1*(9.160e-02)+Out_Bank_Flag1*(-6.898e-02). The result means 1 unit change in the Age_Issued_Updated will result in -1.290e-02 in the All_Max_Bucket_updated, other variables also like Age_Issued_Updated.

After that, we know our dependent variable is All Max Bucket, where the range of All Max Bucket is [0,7]. We classify the predicted value to obtain the confusion matrix of the predicted value, as shown in the following table

	0	1
0	1929	221
1	428	251

Table 20: confusion matrix for Bucket Prediction.

Through the confusion matrix, we can calculate sensitivity and specificity, and then we can get sqrt(sensitivity*specificity) =57.59%.

After that, we use the profit and loss coefficients given by Brainbravo and the predicted buckets of customers we get:

prediction						
0	2	3	4	5	6	7
2357	456	16	0	0	0	0

Table 21:prediction for Bucket Prediction.

Finally, we can figure out the profit and loss=-0.52. We can also know that under this model, this business will bring loss to the bank.

8.1.2.3 Profit and loss prediction model

In this section, our dependent variable is PNL All Max Bucket, where the values of PNL All Max are 0.03, 0.01,0, -0.15, -0.25, -0.35, -0.45, -0.65. We can also get the confusion matrix, then we can calculate the sqrt(sensitivity*specificity) =63.58%.

After that, we use the profit and loss coefficients given by Brainbravo and get the profit and loss=0.08. Meanwhile, we can know that under this model, this business will bring profit to the bank.

8.1.3 Classification and Regression Trees Result

In this part, we use the classification and regression tree algorithm, which is a widely used decision tree learning method. The algorithm can be used for classification and regression. In the calculation process, by setting the threshold and calculating the Gini coefficient, we can get the customer's predicted bucket situation.

8.1.3.1 0 and 1 classification

In this part, we also use the same dependent variable as linear regression and logistic regression which is All Max Bucket Flag, and its range is [0,1]. Then we need to find the threshold that can classify the predicted value of the bucket. According to the threshold, we can get the confusion matrix, then, we can calculate the accuracy, sensitivity, and specificity. Among them, accuracy=51.82%, and we can get sqrt(sensitivity*specificity) =54%.

After that, we use the profit and loss coefficients given by Brainbravo and figure out the profit and loss=-0.09. Here, we can know that under this model, this business will bring loss to the bank.

8.1.3.2 Bucket Prediction Model

In this section, our dependent variable is All Max Bucket, where the range of All Max Bucket is [0,7]. We classify the predicted value to obtain the confusion matrix of the predicted value. Through the confusion matrix, we can calculate sensitivity and specificity, and then we can calculate the sqrt(sensitivity*specificity) =56%.

After that, we use the profit and loss coefficients given by Brainbravo and figure out the profit and loss=-0.78. We can know that under this model, this business will bring loss to the bank.

8.1.3.3 Profit and Loss Prediction Model

In this part, we will find the proper variables for the model first. Then, we will plot the tree and prune the growing tree to check the complexity parameter value, then we use the ROC curve to get the confusion matrix.

In this part, our dependent variable is PNL All Max Bucket, where the values of PNL All Max are 0.03, 0.01, -0.15, -0.25, -0.35, -0.45, -0.65. We can get the confusion matrix:

	0	1
0	91	2348
1	53	337

Table 22: confusion matrix for Profit and Loss Prediction.

Through the confusion matrix, we can calculate the sqrt(sensitivity*specificity) =28%.

After that, we use the profit and loss coefficients given by Brainbravo and the predicted buckets of our customers, as shown in the following table:

prediction						
-0.65	-0.45	-0.35	-0.25	-0.15	0	0.03
77	15	30	67	201	289	2150

Table 23: prediction matrix for Profit and Loss Prediction.

Finally, we can find profit and loss=-0.17. Here, we can know that under this model, this business will bring loss to the bank.

8.1.4 Random Forest result

In this part, we will use the random forest algorithm. The random forest algorithm can efficiently run on large data sets and evaluate the importance of each variable in the classification problem. Each tree in the random forest can be regarded as a CART, so we will get multiple CART, and finally based on the results displayed by different CART, we can draw a comprehensive conclusion.

0 and 1 classification:

In this part, our dependent variable is the All Max Bucket Flag, and its range is [0,1]. Then we need to find the threshold that can classify the predicted value of the bucket. According to the threshold, we can get the confusion matrix and calculate the accuracy, sensitivity, and specificity. Among them, accuracy=76.03%, and we can get the sqrt(sensitivity*specificity) =6.64%.

After that, we use the profit and loss coefficients given by Brainbravo and figure out the profit and loss=0.0147. Here we can know that under this model, this business will bring benefits to the bank.

Bucket Prediction Model:

In this section, our dependent variable is All Max Bucket, where the range of All Max Bucket is [0,7]. We classify the predicted value to obtain the confusion matrix and calculate sensitivity, specificity, and then we get the sqrt(sensitivity*specificity)=0%.

Finally, we use the profit and loss coefficients given by Brainbravo and calculate profit and loss=0.0002. Under this model, this business will bring benefits to the bank.

Profit and Loss Prediction Model:

In this section, our dependent variable is PNL All Max Bucket, where the values of PNL All Max are 0.03, 0.01,0, -0.15, -0.25, -0.35, -0.45, -0.65. We can also get the confusion matrix and get the sqrt(sensitivity*specificity) =0%.

After that, we use the profit and loss coefficients given by Brainbravo and find profit and loss=0.0002. Under this model, this business will bring benefits to the bank.

8.2 Behavioural scoring part

The variables Brainbravo suggested in the Behavioral scoring section are: All_Max_Bucket_Flag_updated,'Attrition_Flag','Blacklist_Flag','Writeoff_Flag','Spending_Limit','Avg_Payments','Avg_Balances','Account_Status_Bucket'_Rlag, ','Age_Issued_Updated' and'Sex_M_Flag'.

8.2.1 Logistic regression result

8.2.1.1 0 and 1 classification

```
Call:
glm(formula = All_Max_Bucket_Flag_updated ~ Ytd_Amt_Cash + Account_Status_Bucket +
    Attrition_Flag + Blacklist_Flag + Current_Balance + Writeoff_Flag +
    Spending_Limit + Avg_Balances, family = "binomial", data = CS_Train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0417  -0.7228  -0.6862  -0.3334   3.2586

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           1.447e-01  2.188e-01   0.661 0.508324
Ytd_Amt_Cash         -1.332e-06  1.125e-07 -11.846  < 2e-16 ***
Account_Status_Bucket -3.488e-03  7.656e-04  -4.556 5.22e-06 ***
Attrition_Flag        -3.046e+00  8.164e-01  -3.731 0.000191 ***
Blacklist_Flag         2.932e+00  8.007e-01   3.662 0.000250 ***
Current_Balance       -8.384e-07  2.228e-07  -3.763 0.000168 ***
Writeoff_Flag          1.761e+01  1.460e+02   0.121 0.904012
Spending_Limit        -1.461e-06  2.984e-07  -4.897 9.71e-07 ***
Avg_Balances           2.366e-06  2.502e-07   9.458  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 9329.8  on 8475  degrees of freedom
Residual deviance: 8758.2  on 8467  degrees of freedom
AIC: 8776.2
```

Figure 8.3: Significant variable.

We know that the Best AIC is 8776.2, the reason why we choose this one is because it has the lowest AIC, and significant variables are except the Writeoff_Flag. The equation is log(All_Max_Bucket_Flag_updated /(1- All_Max_Bucket_Flag_updated))= Ytd_Amt_Cash*(-1.332e-06)+Account_Status_Bucket**(-3.488e-03)+Attrition_Flag*(-3.046e+00)+Blacklist_Flag*2.932e+00+Current_Balance*(-8.384e-07)+Spending_Limit*(-1.46e-06)+Avg_Balances*2.366e-06. The result means the change in the log odds of All_Max_Bucket_Flag+updated per unit change in Ytd_Amt_Cash is 1.477e-01, and other variables all like Ytd_Amt_Cash.

Then, we can calculate the threshold through the ROC curve, and classify different buckets according to the threshold to get the matrix of predicted values:

0	1
2791	38

Table 24: confusion matrix for 0 and 1 classification

After that, the confusion matrix can be calculated, as shown below

	0	1
0	2150	0
1	641	38

Table 25: prediction matrix for 0 and 1 classification

The confusion matrix is called f4, and we can get accuracy through f4, the accuracy<-(f4[1,1] +f4[2,2])/(f4[1,1]+f4[2,2]+f4[2,1]+f4[1,2]). We get the accuracy of logistic regression is 76.36%.

At the same time, we can also obtain sensitivity and specificity, where sensitivity<-f4[2,2]/(f4[2,2] +f4[2,1]), the result is 0.056. In the same way, we can get specificity, where formula is specificitiy<-f4[1,1]/(f4[1,2] +f4[1,1]), and the result is 1. Through sensitivity and specificity, we can calculate sqrt(sensitivity*specificity) =0.2366.

According to the coefficient of profit and loss provided by Brainbravo and the confusion matrix we got, we can calculate profit and loss=(2150/(2150+641))*0.02+(641/(2150+641))*(-0.45) and the result is -0.088, which means that the company cannot make a profit under this model.

8.2.1.2 Bucket Prediction Model

First, we used linear regression to find the best model and the best model as below:

```
Call:
lm(formula = All_Max_Bucket_updated ~ Py_Amt_Cash + Ytd_Amt_Cash +
    Current_Balance + Account_Status_Bucket + Attrition_Flag +
    Blacklist_Flag + Writeoff_Flag + Spending_Limit + Avg_Payments +
    Avg_Balances, data = CS_Train)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3355 -0.7466 -0.6516 -0.1028  6.8672

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)             1.692e+00  1.583e-01  10.688  < 2e-16 ***
Py_Amt_Cash            -2.307e-07  1.314e-07  -1.757 0.079016 .
Ytd_Amt_Cash           -6.226e-07  7.356e-08  -8.464  < 2e-16 ***
Current_Balance        -4.455e-07  1.238e-07  -3.598 0.000322 ***
Account_Status_Bucket -2.646e-03  4.602e-04  -5.750 9.24e-09 ***
Attrition_Flag         -3.375e+00  4.156e-01  -8.121 5.27e-16 ***
Blacklist_Flag          3.460e+00  4.059e-01   8.524  < 2e-16 ***
Writeoff_Flag           6.142e+00  1.766e-01  34.770  < 2e-16 ***
Spending_Limit         -7.789e-07  1.889e-07  -4.123 3.77e-05 ***
Avg_Payments            1.675e-06  6.278e-07   2.667 0.007657 **
Avg_Balances            1.151e-06  1.196e-07   9.631  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.459 on 8465 degrees of freedom
Multiple R-squared:  0.1778,    Adjusted R-squared:  0.1768
F-statistic:   183 on 10 and 8465 DF,  p-value: < 2.2e-16
```

Figure 8.4: Significant variable.

R2 is 0.1778, the reason why we choose this model is because it has the highest R2, although is not so high, and all variables are significant. The equation is All_Max_Bucket_ updated = Py_Amt_Cash*(-2.307e-07)+Ytd_Amt_Cash*(-6.226e-07)+Account_Status_Bucket**(-6.226e+00)+Attrition_Flag*(-3.375e+00)+Blacklist_Flag*(3.460e+00)+Writeoff_Flag*(6.142e-07)+Spending_Limit*(-7.789e-07)+Avg_Payments*(1.675e-06)+Avg_Balances*1.151e-06. The result means 1 unit change in the Py_Amt_Cash will result in -2.307e-07 in the All_Max_Bucket_ updated , other variables all like Py_Amt_Cash

First, we get the confusion matrix as follows:

	0	1
0	1465	685
1	285	394

Table 26: confusion matrix for Bucket Prediction

The confusion matrix is called t5_1, through t5_1 we can calculate the sqrt(sensitivity*specificity) =0.6288.

After that, we need to use the multiple logistic regression function to predict the bucket to which the new customer belongs in the future. We can get the following results:

prediction						
0	2	3	4	5	6	7
2787	0	0	1	2	0	39

Table 27: prediction matrix for Bucket Prediction

Finally, according to the coefficient of each bucket given by Brainbravo, the result of profit and loss we get is -0.001014, which means that this business will bring loss to the bank in the future.

8.2.1.3 Profit and Loss Prediction Model

First, we use the same method as Bucket Prediction Model to get the confusion matrix and figure out the sqrt(sensitivity*specificity)=0.6251.

Then we use the multiple logistic regression model to predict the profit and loss coefficient of the bucket to which the new customer belongs in the future. According to the coefficient of each bucket given by

Brainbravo, the result of profit and loss we get is 0.0203, which shows that the business can be profitable under this model.

8.2.2 Linear regression result

8.2.2.1 0 and 1 classification

In this model, the dependent variable is the All Max Bucket Flag, and its range is [0,1]. According to the linear regression, We found the threshold that can classify the predicted bucket. According to the threshold, we can get the confusion matrix, and calculate the accuracy, sensitivity, and specificity. Among them, accuracy=77.48%, and the sqrt(sensitivity*specificity)=58.37%.

Then we use the profit and loss coefficients given by Brainbravo and the confusion matrix we calculated, we can find profit and loss=-0.0641. Under this model, this business will bring loss to the bank.

8.2.2.2 Bucket Prediction Model

In this section, our dependent variable is All Max Bucket, where the range is [0,7]. We classify the predicted value to obtain the confusion matrix and calculate sensitivity and specificity, and then we can calculate sqrt(sensitivity*specificity)=54.05%.

After that, we use the profit and loss coefficients given by Brainbravo and calculate profit and loss=-0.5397. Under this model, this business will bring loss to the bank.

8.2.2.3 Profit and Loss Prediction Model

In this section, our dependent variable is PNL All Max Bucket, where values are equal to 0.03, 0.01,0, -0.15, -0.25, -0.35, -0.45, -0.65. We can also get the confusion matrix. Through the confusion matrix, we can get sqrt(sensitivity*specificity)=65.96%.

Then, we use the profit and loss coefficients given by Brainbravo and find profit and loss=0.0109. Under this model, this business will bring benefits to the bank.

8.2.3 Classification And Regression Trees Result

8.2.3.1 0 and 1 classification

In this part, the dependent variable is the All Max Bucket Flag, and its range is [0,1]. Then we need to find the threshold that can classify the predicted bucket. According to the threshold, we can get the confusion

matrix, and calculate the accuracy, sensitivity and specificity. Among them, accuracy=80%, and we can get sqrt(sensitivity*specificity)=44%.

After that, we use the profit and loss coefficients given by Brainbravo and get profit and loss=-0.0764. Under this model, this business will bring loss to the bank.

8.2.3.2 Bucket Prediction Model

In this section, our dependent variable is All Max Bucket, where the range is [0,7]. We classify the predicted value to obtain the confusion matrix and calculate sensitivity and specificity, and then we can calculate sqrt(sensitivity*specificity) =48%.

We use the profit and loss coefficients given by Brainbravo and find profit and loss=-0.078. Under this model, this business will bring loss to the bank.

8.2.3.3 Profit and Loss Prediction Model

In this section, our dependent variable is PNL All Max Bucket, where the values are equal to 0.03, 0.01,0, -0.15, -0.25, -0.35, -0.45, -0.65. We can also get the confusion matrix, and get sqrt(sensitivity*specificity)=10.2%.

We use the profit and loss coefficients given by Brainbravo and find profit and loss=-0.0175. Under this model, this business will bring loss to the bank.

8.2.4 Random Forest result

8.2.4.1 0 and 1 classification

In this part, the dependent variable is the All Max Bucket Flag, and its range is [0,1]. Then we can find the threshold that can classify the predicted value of the bucket. According to the threshold, we can get the confusion matrix and calculate the accuracy, sensitivity, and specificity. Among them, accuracy=77.34%, and we can get sqrt(sensitivity*specificity) =23.65%.

We use the profit and loss coefficients given by Brainbravo and find profit and loss=0.0091. Under this model, this business will bring benefits to the bank.

8.2.4.2 Bucket Prediction Model

In this section, our dependent variable is All Max Bucket, where the range is [0,7]. We classify the predicted value to obtain the confusion matrix and calculate sensitivity and specificity, and the sqrt(sensitivity*specificity) =47.17%.

Using the profit and loss coefficients given by Brainbravo and get the profit and loss=0.000079. Under this model, this business will bring benefits to the bank.

8.2.4.3 Profit and Loss Prediction Model

We train the train dataset first, and get the equation,:Behavioral_BP <- randomForest(x = CS_Train[c('Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'Account_Status_Bucket','Current_Balance', 'Branch_Code_Rating','Out_Bank_Flag', 'Age_Issued_Updated','Sex_M_Flag')], y= CS_Train$All_Max_Bucket_updated, mtry = 1, ntree = 100) . We selected 100 as our ntree value as our default value and try different Mtry value and find the best value is 1.

In this section, our dependent variable is PNL All Max Bucket, where the values equal to 0.03, 0.01,0, -0.15, -0.25, -0.35, -0.45, -0.65. We can also get the confusion matrix:

	0	1
0	2130	505
1	38	151

Table 28: confusion matrix for Profit and Loss Prediction

Through the confusion matrix, we can get sqrt(sensitivity*specificity) =47.6%.

Using the profit and loss coefficients given by Brainbravo and the predicted buckets of the customers we obtained has shown in the following table:

prediction						
-0.65	-0.45	-0.35	-0.25	-0.15	0	0.03
46	0	3	5	56	41	2130

Table 29: prediction matrix for Profit and Loss Prediction

Finally, we can get the profit and loss=0.00008. Under this model, this business will bring benefits to the bank.

8.3 The Result Table

Type of Scoring	Model	Algorithm	sqrt(sensitivity *specificity)	Average_Profit_Loss	Accuracy
Application	Zero-one Classification	Best Performance	100.00%	1.50%	100.00%
Application	Zero-one Classification	Default	-	-9.50%	76.06%
Application	Zero-one Classification	Linear Regression	58.76%	-6.40%	76.50%
Application	Zero-one Classification	Logistic Regression	3.84%	-9.27%	76.00%
Application	Zero-one Classification	Decision Tree_CART	54.00%	-9.00%	51.82%
Application	Zero-one Classification	Random Forest	6.64%	1.47%	76.03%
Application	Bucket Prediction	Best Performance	100.00%	-	
Application	Bucket Prediction	Default		-0.78	
Application	Bucket Prediction	Linear Regression	57.59%	-0.52	
Application	Bucket Prediction	Logistic Regression	28.50%	0.00	
Application	Bucket Prediction	Decision Tree_CART	56.00%	-0.78	
Application	Bucket Prediction	Random Forest	0.00%	0.00	
Application	Proft/Loss Prediction	Best Performance	100.00%	2.3%	
Application	Proft/Loss Prediction	Default		-1.7%	
Application	Proft/Loss Prediction	Linear Regression	63.58%	0.8%	
Application	Proft/Loss Prediction	Logistic Regression	28.46%	-1.8%	
Application	Proft/Loss Prediction	Decision Tree_CART	28.00%	-17.0%	15%
Application	Proft/Loss Prediction	Random Forest	0.00%	0.0%	75.99%
Behavioral	Zero-one Classification	Default		-9.50%	76.06%
Behavioral	Zero-one Classification	Linear Regression	58.37%	-6.41%	77.48%

Behavioral	Zero-one Classification	Logistic Regression	23.66%	-8.82%	76.36%
Behavioral	Zero-one Classification	Decision Tree_CART	44.00%	-7.64%	80.00%
Behavioral	Zero-one Classification	Random Forest	23.65%	0.91%	77.34%
Behavioral	Bucket Prediction	Best Performance	100.00%	-	
Behavioral	Bucket Prediction	Default		-0.78	
Behavioral	Bucket Prediction	Linear Regression	54.05%	-0.54	
Behavioral	Bucket Prediction	Logistic Regression	62.88%	-0.00	
Behavioral	Bucket Prediction	Decision Tree_CART	48.00%	-0.08	
Behavioral	Bucket Prediction	Random Forest	47.17%	0.00	80.80%
Behavioral	Proft/Loss Prediction	Best Performance	100.00%	2.25%	
Behavioral	Proft/Loss Prediction	Default		-1.74%	
Behavioral	Proft/Loss Prediction	Linear Regression	65.96%	1.09%	
Behavioral	Proft/Loss Prediction	Logistic Regression	62.51%	0.0%	
Behavioral	Proft/Loss Prediction	Decision Tree_CART	10.20%	-1.8%	
Behavioral	Proft/Loss Prediction	Random Forest	47.60%	0.0%	80.80%

Table30: Final result

9. Limitation.

First of all, in the data cleaning part, Brainbravo provided a dataset that has some important variables, like sex, age, but there were several missing data. This included AVG-PAYMENTS, AVG-BALANCES, OY-AMT-CASH, PY-AMT-CASH, and YTD -AMT-CASH. We replaced the data with missing values and the NA part in these variables with the average value of each variable to ensure the integrity of the data set. Although the operation can facilitate our subsequent data processing, the missing parts of the data that we replaced will have an impact on the results of the regression function, which will cause a little difference between the final results we get and the reference results given by the company but will not make a difference in the final interpretation of the results. In addition, the data set provided by Brainbravo lacks some important variables, such as the user's annual income, the user's deposit, the number of user's family members, the number of user's houses, and the user's education

level. Some stuff is missing and it will improve our result if we have it, but our results are still convincing.

Secondly, while analyzing application scoring and behavioural scoring, we found that we could not calculate the specificity and sensitivity of multi-class data directly. To solve this problem, we first classify multiple classes into 0 and 1 according to the range of different bucket values and then calculate the sensitivity and specificity. Although there are a few differences between our result and the result provided by Brainbravo, the difference was not so large, and the way we use can simplify the process when we predict dependent variables with multiply classes.

Finally, We used multiple algorithms for data analysis such as logistic regression, linear regression, CART and random forest algorithms. At the same time, One particular algorithm that is known for its high accuracy is neural networks, which often outperform traditional algorithms. However, due to the large number of missing data in the dataset provided by Brainbrava. We were unable to achieve high accuracy using a neural network, therefore we did not include it in our analysis.

10. Conclusion.

The main objective of this analysis is to build the application and behaviour scoring models to predict the credit risk of the bank's customers. To build these scoring models, Brainbravo Ltd. provided us the dataset containing the information of the customers who have enrolled for the credit card from April'1999 to September'2001. The dataset had 11,329 observations and 32 variables. Each observation in the dataset is the detail of each customer such as Birthdate, gender, membership date, payments and balances information, blacklist date, YTD, customer YTD and All max buckets data, etc. Before starting the analysis, data pre-processing and exploratory data analysis were performed to convert the data into a proper clean format and understand the distribution and trend in the dataset. For both application and behaviour scoring, three approaches were proposed by Brainbravo ltd. All three approaches: zero-one classification, bucket prediction and profit and loss prediction, were analysed in the project. The dependent variable, all max bucket, is coded as 0 and 1 in the zero-one classification approach. In the bucket prediction approach All max bucket (from 0 to 7) was used as dependent variable. Profit and loss bucket assigned to each bucket was considered in the last approach. The main data set was split into 70% - 30% ratios for Training and testing purposes.

We explored different machine algorithms that can be used in the modelling and decided to use linear regression, logistic regression, decision tree with CART (Classification And Regression Trees) and random forest. We also tried smote technique to handle the imbalanced class problem in the regression models. We decided on the performance evaluation methods – Accuracy, square root of sensitivity and specificity and final profit/loss metrics to compare the performance of the models. We

tested different thresholds to improve the model performance. The result shows that different model works best for different algorithms. We noticed that Random Forest works best for the Zero-one classification for both application and behaviour scoring, while linear regression gave the best results for the Profit and loss prediction. The random forest model had an accuracy of 76% and Avg profit/loss of 1.47% for the application scoring Zero-one classification. Linear regression (using cut-off 0.4) gave the second-best result with an accuracy of 76% but Avg. profit/loss of -6.4%. For application scoring – profit/loss prediction linear regression had the sqrt(sensitivity * specificity) of 63.58% and Avg. profit/loss value as 0.8%. Almost the same results were seen in the behavioural scoring – profit/loss prediction too. Logistic regression had the best results for Application scoring bucket prediction, while the decision tree with CART gave the best output for behavioural scoring bucket prediction. For the application scoring – bucket prediction, logistic regression had the avg. profit/loss of 0%. Similarly for the behavioural scoring bucket prediction, decision tree with CART had the highest avg. profit/loss of -0.08% almost same as 0% of the best model. Hence, we would recommend to use Random Forest for Zero-One classification, linear regression for Profit and loss prediction and logistic regression for bucket prediction.

However, These results can be improved using improved data collection techniques 1) By capturing more information regarding the customer's transactional information, 2) Sharing the logic for the already assigned buckets and bucket changed reasons, 3) Having fewer missing values in the important fields such as Spending limit, Cash fields, etc. Furthermore, more sophisticated statistical tests and machine learning models can be tested on this data to improve the scoring of the customers in the future.

11. Appendix

```
####Set the working directory#######
setwd("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code")
###Clear the screen and previosly loaded dataset#####
rm(list = ls())
cat("\14")
####Install the required packages###################################
if(!require('haven')){
  install.packages('haven')
  library('haven')}
if(!require('stringr')){
```

```
  install.packages('stringr')
  library('stringr')}
if (!require('stargazer')) {
  install.packages('stargazer')
  library('stargazer')}
if (!require('funModeling')) {
  install.packages('funModeling')
  library('funModeling')}
if (!require('lubridate')) {
  install.packages('lubridate ')
  library('lubridate ')}
if (!require('plyr')) {
  install.packages('plyr')
  library('plyr')}
if (!require('dplyr')) {
  install.packages('dplyr')
  library('dplyr')}
if (!require('ggplot2')) {
  install.packages('ggplot2')
  library('ggplot2')}
if (!require('caret')) {
  install.packages('caret')
  library('caret')}
if (!require('reshape2')) {
  install.packages('reshape2')
  library('reshape2')}
if (!require('ggfortify')) {
  install.packages('ggfortify')
  library('ggfortify')}
if (!require('stargazer')) {
  install.packages('stargazer')
```

```
 library('stargazer')}
options(scipen = 999)
#####################Import the dataset#######################
CS_Data <- read.csv("Data.csv")
#view(CS_Data)
####Rename variables and remove unwanted columns from the data
colnames(CS_Data) <- c('Account_No', 'No_Of_Cards', 'Member_Since', 'Birth_Date', 'Sex',
                       'Occupation_Code', 'Black_List_Code', 'Attrition_Reason', 'Black_List_Date',
                       'Write_Off_Date', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'At_Anniv_Date',
                       'Free_Fee_Flag', 'Cycle_Code', 'Last_Trx_Date', 'Bucket',
                       'Out_Bank_Account', 'Y4_Amt_Cash', 'Oy_Amt_Cash', 'Py_Amt_Cash',
                       'Ytd_Amt_Cash', 'Ytd_Max_Bucket', 'Py_Max_Bucket', 'All_Max_Bucket',
                       'Account_Status', 'Current_Balance', 'Increase_Limit_Flag', 'Increase_Limit_Date',
                       'Card_Delivery', 'Branch_Code')
CS_Data <- subset(CS_Data,select = -c(Free_Fee_Flag, Cycle_Code, Y4_Amt_Cash,
Increase_Limit_Flag,
                                      Increase_Limit_Date, Card_Delivery))


########Create membership date and birthdate
CS_Data <- transform(CS_Data, Membership_Date =
as.Date(paste(as.character(Member_Since),"01", sep=""), "%Y%m%d"))
#CS_Data <- CS_Data[, c(1:3, 27, 4:26)]
CS_Data <- transform(CS_Data, Birth_Date = as.Date(as.character(Birth_Date), "%Y%m%d"))
########Create age card issued
CS_Data$Age_Issued <- round(time_length(difftime(CS_Data$Membership_Date,
CS_Data$Birth_Date), "years"))
#CS_Data <- CS_Data[, c(1:5, 28, 6:27)]
########Create gender flag
CS_Data <- within(CS_Data, {Sex_M_Flag <- ifelse(Sex == "M", 1, 0)})
#CS_Data <- CS_Data[, c(1:7, 29, 8:28)]
########Create blacklist, writeoff and attrition flag
CS_Data <- within(CS_Data, {Blacklist_Flag <- ifelse(Black_List_Code == 0, 0, 1)})
```

```
CS_Data <- within(CS_Data, {Attrition_Flag <- ifelse(Attrition_Reason == 0, 0, 1)})

CS_Data <- within(CS_Data, {Writeoff_Flag <- ifelse(Write_Off_Date == 0, 0, 1)})

#CS_Data <- CS_Data[, c(1:13, 30:32, 14:29)]

########Convert blacklist and writeoff to date format

CS_Data <- transform(CS_Data, Black_List_Date = as.Date(as.character(Black_List_Date),
"%Y%m%d"))

CS_Data <- transform(CS_Data, Write_Off_Date = as.Date(as.character(Write_Off_Date),
"%Y%m%d"))

########Created days to blacklist and membership variables

CS_Data$Days_Writeoff_blacklist <- round(time_length(difftime(CS_Data$Write_Off_Date,
CS_Data$Black_List_Date), "days"))

CS_Data$Days_Blacklist_Membership <- round(time_length(difftime(CS_Data$Black_List_Date,
CS_Data$Membership_Date), "days"))

#CS_Data <- CS_Data[, c(1:16, 33:34, 17:32)]

########Convert at aniv and last transx to date varaibles

CS_Data <- transform(CS_Data, At_Anniv_Date = as.Date(as.character(At_Anniv_Date), "%Y%m%d"))

CS_Data <- transform(CS_Data, Last_Trx_Date = as.Date(as.character(Last_Trx_Date), "%Y%m%d"))



########Create out bank flag to date varaibles

CS_Data <- within(CS_Data, {Out_Bank_Flag <- ifelse(Out_Bank_Account >= 1, 1, 0)})

#CS_Data <- CS_Data[, c(1:25, 35, 26:34)]

########Remove all max bucket

#CS_Data <- subset(CS_Data,select = -c(All_Max_Bucket))

check <- as.data.frame(round(time_length(difftime(CS_Data$At_Anniv_Date,
CS_Data$Membership_Date), "days")))

rm(check)

########Create buckets for regression

CS_Data <- within(CS_Data, {Bucket_Flag <- ifelse(Bucket >= 2, 1,0)})

CS_Data <- within(CS_Data, {Py_Max_Bucket_Flag <- ifelse(Py_Max_Bucket >= 2, 1,0)})

#check <- as.data.frame(round(time_length(difftime(CS_Data$Last_Trx_Date,
CS_Data$Membership_Date), "days")))

#rm(check)
```

```
#unique(CS_Data[c("Py_Max_Bucket_Flag","Py_Max_Bucket")])

###Remove wrong birthdate  (11312 from 11329 rows)

CS_Data <- subset(CS_Data, year(Birth_Date) >= 1900 & year(Birth_Date) <= 2002)

describe(CS_Data$Spending_Limit)

###Remove spending limit 0 and 1  (11305 from 11312 rows)

CS_Data <- subset(CS_Data, Spending_Limit > 1)

##############################################

########Created days to last transc and membership variables

########Created anniv to membership variables

CS_Data$Days_lasttrx_membership <- round(time_length(difftime(CS_Data$Last_Trx_Date,
CS_Data$Membership_Date), "days"))

#CS_Data <- CS_Data[, c(1:16,41, 17:40)]

#CS_Data <- CS_Data[, c(1:16,18:24,17,25:41)]

CS_Data$Days_Anniv_membership <- round(time_length(difftime(CS_Data$At_Anniv_Date,
CS_Data$Membership_Date), "days"))

#CS_Data <- CS_Data[, c(1:22,42, 23:41)]

########Create buckets for YTD max bucket


CS_Data <- within(CS_Data, {Ytd_Max_Bucket_Flag <- ifelse(Ytd_Max_Bucket >= 2, 1,0)})

#CS_Data <- CS_Data[, c(1:32,43, 33:42)]

#####create values for account status

CS_Data <- within(CS_Data, {Account_Status_Bucket <-

  ifelse(Account_Status == "0", 0,

       ifelse(Account_Status == "12", 12,

            ifelse(Account_Status == "21", 21,

                 ifelse(Account_Status == "33", 33,

                      ifelse(Account_Status == "44", 44,

                           ifelse(Account_Status == "55", 55,

                                ifelse(Account_Status == "65", 65,

                                     ifelse(Account_Status == "0S", 75,

                                          ifelse(Account_Status == "ME", 85, 95)))))))))})

#unique(CS_Data[c("Account_Status","Account_Status_Bucket")])
```

```
#####Create all max buckt flag
CS_Data <- within(CS_Data, {All_Max_Bucket_Flag <- ifelse(All_Max_Bucket >= 2, 1,0)})
#CS_Data <- CS_Data[, c(1:35,45, 36:44)]
######Age issued updated column
CS_Data <- within(CS_Data, {Age_Issued_Updated <-
  ifelse(Age_Issued < 18 | Age_Issued >= 81, NA, Age_Issued)})
##############################################
#a <- unique(CS_Data[c("All_Max_Bucket_Flag","Writeoff_Flag")])
#rm(a)
#######Create new buckets based on the write off dates ##############
CS_Data <- within(CS_Data, {All_Max_Bucket_updated <-
  ifelse(Writeoff_Flag == 1 & All_Max_Bucket <= 1, 7, All_Max_Bucket)})
# CS_Data <- within(CS_Data, {Py_Max_Bucket_Updated <-
#   ifelse(Writeoff_Flag == 1 & Py_Max_Bucket <= 1, 7, Py_Max_Bucket)})
########Create buckets for regression
CS_Data <- within(CS_Data, {All_Max_Bucket_Flag_updated <- ifelse(All_Max_Bucket_updated >= 2,
1,0)})
#CS_Data <- within(CS_Data, {Py_Max_Bucket_Updated_Flag <- ifelse(Py_Max_Bucket_Updated >=
2, 1,0)})
####create values for P&L
CS_Data <- within(CS_Data, {PNL_All_Max_Bucket <-
  ifelse(All_Max_Bucket_updated == 0, 0.03,
       ifelse(All_Max_Bucket_updated == 1, 0.01,
            ifelse(All_Max_Bucket_updated == 2, 0,
                 ifelse(All_Max_Bucket_updated == 3, -.15,
                      ifelse(All_Max_Bucket_updated == 4, -.25,
                           ifelse(All_Max_Bucket_updated == 5, -.35,
                                ifelse(All_Max_Bucket_updated == 6, -.45, -.65)))))))})
##############################################
##########Create variables for branch code analysis
####try three buckets in the modeling##### and report the result
##############Bucket Change varaible
```

```
CS_Data$Bucket_All_Max_Updated <-
paste(CS_Data$Bucket_Flag,CS_Data$All_Max_Bucket_Flag_updated,sep="-")

#CS_Data <- CS_Data[, c(2:35,1, 36:40)]

############################################################

#typeof(CS_Data$Bucket)

#a <- unique(CS_Data[c("Py_Max_Bucket","Py_Max_Bucket_Updated","Writeoff_Flag")])

#rm(a)

rite.csv(CS_Data,"Check.csv")

#######################################33

#Create New DF for payment and balance as NA

CS_Data1 <- CS_Data

CS_Data1$Avg_Payments <-  str_replace_all(CS_Data1$Avg_Payments, "-", "NA")

CS_Data1$Avg_Balances <-  str_replace_all(CS_Data1$Avg_Balances, "-", "NA")

CS_Data1$Ytd_Amt_Cash <-  str_replace_all(CS_Data1$Ytd_Amt_Cash, "-", "NA")

CS_Data1$Current_Balance <-  str_replace_all(CS_Data1$Current_Balance, "-", "NA")

##Converting numeric values to numeric by removing commas

CS_Data1$Spending_Limit <- as.numeric(gsub(",", "", CS_Data1$Spending_Limit))

CS_Data1$Avg_Payments <- as.numeric(gsub(",", "", CS_Data1$Avg_Payments))

CS_Data1$Avg_Balances <- as.numeric(gsub(",", "", CS_Data1$Avg_Balances))

CS_Data1$Oy_Amt_Cash <- as.numeric(gsub(",", "", CS_Data1$Oy_Amt_Cash))

CS_Data1$Py_Amt_Cash <- as.numeric(gsub(",", "", CS_Data1$Py_Amt_Cash))

CS_Data1$Ytd_Amt_Cash <- as.numeric(gsub(",", "", CS_Data1$Ytd_Amt_Cash))

CS_Data1$Current_Balance <- as.numeric(gsub(",", "", CS_Data1$Current_Balance))

######Remove the unwanted columns from the data

CS_Data1 <- subset(CS_Data1,select = -c(Account_No, No_Of_Cards, Member_Since, Sex,
                                        Black_List_Code, Attrition_Reason,Out_Bank_Account,
                                        Occupation_Code))

#######create test and train dataset based on the bucket


df_train = data.frame()

df_test = data.frame()
```

```
for (i in 0:length(unique(CS_Data1$All_Max_Bucket_updated))) {
  df1 <- subset(CS_Data1, All_Max_Bucket_updated == (i))
  set.seed(3456)
  sample <- sample.int(n = nrow(df1), size = floor(.75*nrow(df1)), replace = F)
  train <- df1[sample, ]
  test  <- df1[-sample, ]
  df_train <- rbind(df_train,train)
  df_test <- rbind(df_test,test)
}

#as.data.frame(table(CS_Data1$All_Max_Bucket_updated))
rm(df1, test, train, i, sample)
#unique(df_test[c("All_Max_Bucket_updated","All_Max_Bucket")])
check <- dcast(CS_Data, Branch_Code ~ All_Max_Bucket_Flag_updated, value.var="Age_Issued",
fun.aggregate=length)
colnames(check) <- c('Branch_Code', 'Bucket_Flag_0', 'Bucket_Flag_1')

# check1 <- dcast(CS_Data, Branch_Code ~ Py_Max_Bucket_Updated_Flag, value.var="Age_Issued",
fun.aggregate=length)
# colnames(check1) <- c('Branch_Code', 'Py_Max_Bucket_Flag_0', 'Py_Max_Bucket_Flag_1')
#
# check2 <- merge(check, check1, by = "Branch_Code")
#rm(check1, check)
check$Bucket_Pe1 <- check$Bucket_Flag_0/(check$Bucket_Flag_0 + check$Bucket_Flag_1)
check <- within(check, Branch_Code_Rating <- as.integer(cut(Bucket_Pe1, quantile(Bucket_Pe1,
probs=0:4/4), include.lowest=TRUE)))
check <- check [-c(2:4)]

CS_Data <- merge(x=CS_Data, y=check, by = "Branch_Code", all.x = TRUE)
rm(check)
eda_ds <- subset(CS_Data1,select = c(Birth_Date,Black_List_Date,Write_Off_Date,Spending_Limit,
                                     Avg_Payments,Avg_Balances,At_Anniv_Date,Last_Trx_Date,Bucket,
```

```
                    Oy_Amt_Cash,Py_Amt_Cash,Ytd_Amt_Cash,Ytd_Max_Bucket,Py_Max_Bucket,
                    All_Max_Bucket, Current_Balance,Membership_Date,Age_Issued,Sex_M_Flag,
                    Blacklist_Flag,Attrition_Flag,Writeoff_Flag,Days_Writeoff_blacklist,
                    Days_Blacklist_Membership,Out_Bank_Flag,Bucket_Flag,Py_Max_Bucket_Flag,
                    Days_lasttrx_membership,Days_Anniv_membership,Ytd_Max_Bucket_Flag,
                    Account_Status_Bucket,All_Max_Bucket_Flag,Age_Issued_Updated,
                    All_Max_Bucket_updated,All_Max_Bucket_Flag_updated,PNL_All_Max_Bucket,
                    Branch_Code_Rating,Bucket_All_Max_Updated))
CS_Data1 <- subset(CS_Data1,select =
c(Spending_Limit,Avg_Payments,Avg_Balances,Bucket,Oy_Amt_Cash,

Py_Amt_Cash,Ytd_Amt_Cash,Ytd_Max_Bucket,Py_Max_Bucket,Current_Balance,
                     Sex_M_Flag,Blacklist_Flag,Attrition_Flag,Writeoff_Flag,Out_Bank_Flag,

Bucket_Flag,Py_Max_Bucket_Flag,Ytd_Max_Bucket_Flag,Account_Status_Bucket,

Age_Issued_Updated,All_Max_Bucket_updated,All_Max_Bucket_Flag_updated,
                     PNL_All_Max_Bucket,Branch_Code_Rating))
df_test <- subset(df_test,select =
c(Spending_Limit,Avg_Payments,Avg_Balances,Bucket,Oy_Amt_Cash,

Py_Amt_Cash,Ytd_Amt_Cash,Ytd_Max_Bucket,Py_Max_Bucket,Current_Balance,
                    Sex_M_Flag,Blacklist_Flag,Attrition_Flag,Writeoff_Flag,Out_Bank_Flag,

Bucket_Flag,Py_Max_Bucket_Flag,Ytd_Max_Bucket_Flag,Account_Status_Bucket,
                    Age_Issued_Updated,All_Max_Bucket_updated,All_Max_Bucket_Flag_updated,
                    PNL_All_Max_Bucket,Branch_Code_Rating))
df_train <- subset(df_train,select =
c(Spending_Limit,Avg_Payments,Avg_Balances,Bucket,Oy_Amt_Cash,

Py_Amt_Cash,Ytd_Amt_Cash,Ytd_Max_Bucket,Py_Max_Bucket,Current_Balance,
                     Sex_M_Flag,Blacklist_Flag,Attrition_Flag,Writeoff_Flag,Out_Bank_Flag,

Bucket_Flag,Py_Max_Bucket_Flag,Ytd_Max_Bucket_Flag,Account_Status_Bucket,
```

```
Age_Issued_Updated,All_Max_Bucket_updated,All_Max_Bucket_Flag_updated,
                        PNL_All_Max_Bucket,Branch_Code_Rating))
saveRDS(CS_Data, file = "CS_Data.RDS")
saveRDS(CS_Data1, file = "CS_Data1.RDS")
saveRDS(df_test, file = "CS_Test.RDS")
saveRDS(df_train, file = "CS_Train.RDS")
saveRDS(eda_ds, file = "eda_ds.RDS")
#write.csv(CS_Data1, "Check.csv")
###########Creating fields for Month=year in R
#eda_ds <- CS_Data1
#saveRDS(eda_ds, file = "eda_ds.RDS")
rm(list = ls())
cat("\14")
eda_ds <- readRDS("eda_ds.RDS")
#sapply(eda_ds, class)

###COnvert flag to factor variables
eda_ds$Sex_M_Flag <- as.factor(eda_ds$Sex_M_Flag)
eda_ds$Blacklist_Flag <- as.factor(eda_ds$Blacklist_Flag)
eda_ds$Attrition_Flag <- as.factor(eda_ds$Attrition_Flag)
eda_ds$Writeoff_Flag <- as.factor(eda_ds$Writeoff_Flag)
eda_ds$Out_Bank_Flag <- as.factor(eda_ds$Out_Bank_Flag)
eda_ds$Ytd_Max_Bucket_Flag <- as.factor(eda_ds$Ytd_Max_Bucket_Flag)
eda_ds$All_Max_Bucket_Flag <- as.factor(eda_ds$All_Max_Bucket_Flag)
eda_ds$All_Max_Bucket_Flag_updated <- as.factor(eda_ds$All_Max_Bucket_Flag_updated)
eda_ds$Bucket_Flag <- as.factor(eda_ds$Bucket_Flag)
eda_ds$Py_Max_Bucket_Flag <- as.factor(eda_ds$Py_Max_Bucket_Flag)
eda_ds$Bucket_All_Max_Updated <- as.factor(eda_ds$Bucket_All_Max_Updated)
eda_ds$Account_Status_Bucket_Fac <- eda_ds$Account_Status_Bucket
eda_ds$Account_Status_Bucket_Fac <- as.factor(eda_ds$Account_Status_Bucket_Fac)
```

```
############Bucket EDA
#for date variable
for(i in 1:length(names(eda_ds))) {
 if(is.Date(eda_ds [,i])) {
  tryCatch({
   pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\Graphsv1\\",
               paste(colnames(eda_ds)[i], "Histogram.pdf", sep ="-"), sep=""),
      width = 11.69, # The width of the plot in inches
      height = 8.27) # The height of the plot in inches
   print(ggplot(eda_ds, aes(x = eda_ds[,i])) +
         geom_bar(fill = "Red")  +
         labs(x=colnames(eda_ds)[i]) +
         ggtitle(paste(colnames(eda_ds)[i], ": Distribution")) +
         theme_classic())
   dev.off()
  }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
 } else if(is.factor(eda_ds [,i]) | is.character(eda_ds [,i])) {
  tryCatch({
   pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\Graphsv1\\",
               paste(colnames(eda_ds)[i], "Histogram.pdf", sep ="-"), sep=""),
      width = 11.69, # The width of the plot in inches
      height = 8.27) # The height of the plot in inches
   print(ggplot(eda_ds, aes(x = eda_ds[,i])) +
         geom_bar(width = 0.5, fill = "Red")  +
         geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
         labs(x=colnames(eda_ds)[i]) +
         ggtitle(paste(colnames(eda_ds)[i], ": Distribution")) +
         theme_classic())
   dev.off()
  }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
 }else {}
```

```
}

a <- as.data.frame(sapply(eda_ds, class))
rm(a)

rm(i)
######For integer type data types
#i=35
for(i in 1:length(names(eda_ds))) {
  if(is.integer(eda_ds [,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\Graphsv1\\",
                       paste(colnames(eda_ds)[i], "Histogram.pdf", sep ="-"), sep=""),
          width = 11.69, # The width of the plot in inches
          height = 8.27) # The height of the plot in inches
      x_axis_labels <- min(eda_ds[,i]):max(eda_ds[,i])
      print(ggplot(eda_ds, aes(x = eda_ds[,i])) +
              geom_bar(fill = "Red")  +
              scale_x_continuous(labels = x_axis_labels, breaks = x_axis_labels) +
              labs(x=colnames(eda_ds)[i]) +
              ggtitle(paste(colnames(eda_ds)[i], ": Distribution")) +
              theme_classic())
      rm(x_axis_labels)
      dev.off()
    }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}
rm(i)

###for numeric type datatype
```

```
for(i in 1:length(names(eda_ds))) {
  if(is.numeric(eda_ds [,i]) & ! is.integer(eda_ds[,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\Graphsv1\\",
                       paste(colnames(eda_ds)[i], "Histogram.pdf", sep ="-"), sep=""),
          width = 11.69, # The width of the plot in inches
          height = 8.27) # The height of the plot in inches
      print(ggplot(eda_ds, aes(x = eda_ds[,i])) +
              geom_histogram(bins=30, fill = "Red")  +
              labs(x=colnames(eda_ds)[i]) +
              ggtitle(paste(colnames(eda_ds)[i], ": Distribution")) +
              theme_classic())
      dev.off()
    }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}


rm(i)


#freq(eda_ds)


#sapply(eda_ds, class)


##################EDA for bucket variable
#####Bucket flag and Age issued


vline <-  summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated), qtl =
quantile(Age_Issued_Updated, .5,na.rm = TRUE))
vline1 <- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated), mean1 =
mean(Age_Issued_Updated,na.rm = TRUE))
myplota3 <- ggplot(eda_ds, aes(x = Age_Issued_Updated, fill = All_Max_Bucket_Flag_updated)) +
  geom_density(alpha = 0.4) +
```

```
  scale_fill_brewer(palette = "Dark2") +
  ggtitle("All Max Flag: Age issued distribution") +
  facet_wrap(. ~ All_Max_Bucket_Flag_updated, dir = "v") +
  geom_vline(data = vline1,aes(xintercept = mean1), color = "black", size = 1) +
  geom_vline(data = vline,aes(xintercept = qtl), color = "darkgrey", size = 1) +
  theme_classic() +
  theme(legend.position = "none")
rm(vline)
rm(vline1)

print(myplota3)
ggsave("EDA\\Bucket flag_Age issued updated.png", plot = myplota3)
rm(myplota3)

####################################
#####Bucket flag sex m flag

myplota4 <- ggplot(eda_ds, aes(fill = Sex_M_Flag, x = All_Max_Bucket_Flag_updated)) +
  geom_bar(position = "stack", stat = "count") +
  geom_text(stat = 'count', aes(label = ..count..), size = 3, position = position_stack(vjust = 0.5)) +
  ggtitle("All Max Flag: Sex flag distribution") +
  theme_classic()

print(myplota4)
ggsave("EDA\\Bucket flag_Sex flag updated.png", plot = myplota4)
rm(myplota4)

####################################
#####Bucket flag repayment flag

myplota5 <- ggplot(eda_ds, aes(fill = Out_Bank_Flag, x = All_Max_Bucket_Flag_updated)) +
```

```
  geom_bar(position = "stack", stat = "count") +
  geom_text(stat = 'count', aes(label = ..count..), size = 3, position = position_stack(vjust = 0.5)) +
  ggtitle("All Max Flag: Repayemnt \n (Out_Bank_Flag) distribution") +
  theme_classic()

print(myplota5)
ggsave("EDA\\Bucket flag_Out_Bank_Flag updated.png", plot = myplota5)
rm(myplota5)

########Branch flag ###############

myplota6 <- ggplot(eda_ds, aes(fill = All_Max_Bucket_Flag_updated, x = Branch_Code_Rating)) +
  geom_bar(position = "stack", stat = "count") +
  geom_text(stat = 'count', aes(label = ..count..), size = 3, position = position_stack(vjust = 0.5)) +
  ggtitle("All Max Flag: Branch Code Rating") +
  theme_classic()

print(myplota6)
ggsave("EDA\\Bucket flag_Branch_Code_Rating updated.png", plot = myplota6)
rm(myplota6)

#########################################################

################EDA for FLag field################################################
##########For date graphs
for(i in 1:length(names(eda_ds))) {
  if(is.Date(eda_ds [,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                       paste(colnames(eda_ds)[i], "All_Max_Flag.pdf", sep ="-"), sep=""),
          width = 11.69, # The width of the plot in inches
```

```
        height = 8.27) # The height of the plot in inches
    print(ggplot(eda_ds, aes(x = eda_ds[,i], fill = All_Max_Bucket_Flag_updated)) +
            geom_bar(position = "stack", stat = "count")   +
            labs(x=colnames(eda_ds)[i]) +
            ggtitle(paste(colnames(eda_ds)[i], "vs All Max Flag")) +
            theme_classic())
    dev.off()
   }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}
rm(i)


##########For factor graphs
for(i in 1:length(names(eda_ds))) {
  if(is.factor(eda_ds [,i])) {
   tryCatch({
    pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                     paste(colnames(eda_ds)[i], "All_Max_Flag.pdf", sep ="-"), sep=""),
        width = 11.69, # The width of the plot in inches
        height = 8.27) # The height of the plot in inches
    print(  ggplot(eda_ds, aes(fill = eda_ds[,i], x = All_Max_Bucket_Flag_updated)) +
              geom_bar(position = "stack", stat = "count") +
              geom_text(stat = 'count', aes(label = ..count..), size = 3, position = position_stack(vjust =
0.5)) +
              ggtitle(paste(colnames(eda_ds)[i], "vs All Max Flag")) +
              labs(fill = colnames(eda_ds)[i]) +
              theme_classic()
    )
    dev.off()
   }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
```

```
}
rm(i)


##########For integer graphs
for(i in 1:length(names(eda_ds))) {
  if(is.integer(eda_ds [,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                       paste(colnames(eda_ds)[i], "All_Max_Flag.pdf", sep ="-"), sep=""),
          width = 11.69, # The width of the plot in inches
          height = 8.27) # The height of the plot in inches
      x_axis_labels <- min(eda_ds[,i], na.rm = TRUE):max(eda_ds[,i], na.rm = TRUE)
      print(  ggplot(eda_ds, aes(fill = All_Max_Bucket_Flag_updated, x = eda_ds[,i])) +
                geom_bar(position = "stack", stat = "count") +
                geom_text(stat = 'count', aes(label = ..count..), size = 3, position = position_stack(vjust =
0.5)) +
                scale_x_continuous(labels = x_axis_labels, breaks = x_axis_labels) +
                labs(x=colnames(eda_ds)[i]) +
                ggtitle(paste(colnames(eda_ds)[i], "vs All Max Flag")) +
                theme_classic()
      )
      rm(x_axis_labels)
      dev.off()
    }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}
rm(i)


#################For numeirc froelds
vline1<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean1=
mean(Spending_Limit, na.rm = TRUE))
```

```
vline2<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean2=
mean(Avg_Payments, na.rm = TRUE))

vline3<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean3= mean(Avg_Balances,
na.rm = TRUE))

vline4<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean4=
mean(Oy_Amt_Cash, na.rm = TRUE))

vline5<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean5= mean(Py_Amt_Cash,
na.rm = TRUE))

vline6<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean6=
mean(Ytd_Amt_Cash, na.rm = TRUE))

vline7<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean7=
mean(Current_Balance, na.rm = TRUE))

vline8<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean8= mean(Age_Issued,
na.rm = TRUE))

vline9<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean9=
mean(Days_Writeoff_blacklist, na.rm = TRUE))

vline10<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean10=
mean(Days_Blacklist_Membership, na.rm = TRUE))

vline11<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean11=
mean(Days_lasttrx_membership, na.rm = TRUE))

vline12<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean12=
mean(Days_Anniv_membership, na.rm = TRUE))

vline13<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean13=
mean(Account_Status_Bucket, na.rm = TRUE))

vline14<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean14=
mean(Age_Issued_Updated, na.rm = TRUE))

vline15<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean15=
mean(All_Max_Bucket_updated, na.rm = TRUE))

vline16<- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated),mean16=
mean(PNL_All_Max_Bucket, na.rm = TRUE))


vline <- cbind(vline1, vline2[,2], vline3[,2], vline4[,2], vline5[,2], vline6[,2],

        vline7[2], vline8[,2], vline9[,2], vline10[,2], vline11[,2], vline12[,2], vline13[,2],

        vline14[2], vline15[,2], vline16[,2])

rm(vline1,vline2,vline3,vline4, vline5, vline6, vline7, vline8,

  vline9, vline10, vline11, vline12, vline13, vline14, vline15, vline16)
```

```
#rm(vline,vline1, vline2, numdata)
#rm(i,x)


j=1
###For numeric density graphs
for(i in 1:length(names(eda_ds))) {
  if(is.numeric(eda_ds [,i]) & ! is.integer(eda_ds[,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                       paste(colnames(eda_ds)[i], "All_Max_Flag.pdf", sep ="-"), sep=""),
          width = 11.69, # The width of the plot in inches
          height = 8.27) # The height of the plot in inches
      j=j+1
      #vline <-  summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated), qtl = quantile(eda_ds[,i],
.5, na.rm = TRUE))
      #vline1 <- summarise(group_by(eda_ds, All_Max_Bucket_Flag_updated), mean1 =
mean(eda_ds[,i], na.rm = TRUE))
      print ( ggplot(eda_ds, aes(x = eda_ds[,i], fill = All_Max_Bucket_Flag_updated)) +
                geom_density(alpha = 0.4) +
                scale_fill_brewer(palette = "Dark2") +
                facet_wrap(. ~ All_Max_Bucket_Flag_updated, dir = "v") +
                geom_vline(data = vline,aes(xintercept = vline[,j]), color = "black", size = 1) +
                #geom_vline(data = vline,aes(xintercept = qtl), color = "darkgrey", size = 1) +
                labs(x=colnames(eda_ds)[i]) +
                ggtitle(paste(colnames(eda_ds)[i], "vs All Max Flag")) +
                theme_classic() +
                theme(legend.position = "none")
      )
      #rm(vline, vline1)
      dev.off()
    }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
```

```
  } else{}
}
rm(i,j,vline)

################################################################################################
######EDA for Max bucket

##########For factor graphs
for(i in 1:length(names(eda_ds))) {
  if(is.factor(eda_ds [,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                       paste(colnames(eda_ds)[i], "All_Max_Bucket.pdf", sep ="-"), sep=""),
          width = 11.69, # The width of the plot in inches
          height = 8.27) # The height of the plot in inches
      x_axis_labels <- min(eda_ds$All_Max_Bucket_updated, na.rm =
TRUE):max(eda_ds$All_Max_Bucket_updated, na.rm = TRUE)
      print(  ggplot(eda_ds, aes(fill = eda_ds[,i], x = All_Max_Bucket_updated)) +
                geom_bar(position = "stack", stat = "count") +
                geom_text(stat = 'count', aes(label = ..count..), size = 2, position = position_stack(vjust =
0.5)) +
                scale_x_continuous(labels = x_axis_labels, breaks = x_axis_labels) +
                labs(fill = colnames(eda_ds)[i]) +
                ggtitle(paste(colnames(eda_ds)[i], "vs All Max Bucket")) +
                theme_classic()
      )
      rm(x_axis_labels)
      dev.off()
    }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}
rm(i)
```

```
##########For box plot graphs
eda_ds$All_Max_Bucket_updated_fc <- as.factor(eda_ds$All_Max_Bucket_updated)
for(i in 1:length(names(eda_ds))) {
  if(is.numeric(eda_ds [,i]) & ! is.integer(eda_ds[,i])) {
    tryCatch({
      pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                  paste(colnames(eda_ds)[i], "All_Max_Bucket.pdf", sep ="-"), sep=""),
        width = 11.69, # The width of the plot in inches
        height = 8.27) # The height of the plot in inches

      print (ggplot(eda_ds, aes(x=All_Max_Bucket_updated_fc, y=eda_ds[,i], fill =
All_Max_Bucket_updated_fc )) +
             geom_boxplot() +
             labs(y = colnames(eda_ds)[i], x = "All Max Bucket") +
             scale_fill_brewer(palette="Dark2") +
             ggtitle(paste(colnames(eda_ds)[i], "vs All Max Bucket")) +
             theme_classic() +
             scale_y_continuous(labels = scales::comma) +
             theme(legend.position = "none")
      )
      dev.off()
    }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}
rm(i)

#############################################################################################
3
##########EDA for bucket all max bucket

##########For factor graphs
for(i in 1:length(names(eda_ds))) {
```

```
 if(is.factor(eda_ds [,i])) {
   tryCatch({
     pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                 paste(colnames(eda_ds)[i], "Bucket_All_Max_Updated.pdf", sep ="-"), sep=""),
        width = 11.69, # The width of the plot in inches
        height = 8.27) # The height of the plot in inches
     #x_axis_labels <- min(eda_ds$Bucket_All_Max_Updated, na.rm =
TRUE):max(eda_ds$Bucket_All_Max_Updated, na.rm = TRUE)
     print(  ggplot(eda_ds, aes(fill = eda_ds[,i], x = Bucket_All_Max_Updated)) +
             geom_bar(position = "stack", stat = "count") +
             geom_text(stat = 'count', aes(label = ..count..), size = 2, position = position_stack(vjust =
0.5)) +
             #scale_x_continuous(labels = x_axis_labels, breaks = x_axis_labels) +
             labs(fill = colnames(eda_ds)[i]) +
             ggtitle(paste(colnames(eda_ds)[i], "vs Bucket_All_Max_Updated")) +
             theme_classic()
     )
     dev.off()
   }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
 } else{}
}
rm(i)




##########For box plot graphs
for(i in 1:length(names(eda_ds))) {
 if(is.numeric(eda_ds [,i]) & ! is.integer(eda_ds[,i])) {
   tryCatch({
     pdf(file = paste("C:\\Users\\Akii\\Documents\\Sem1\\Group Project\\Code\\EDA\\New\\",
                 paste(colnames(eda_ds)[i], "Bucket_All_Max_Updated.pdf", sep ="-"), sep=""),
        width = 11.69, # The width of the plot in inches
```

```
        height = 8.27) # The height of the plot in inches


    print (ggplot(eda_ds, aes(x=Bucket_All_Max_Updated, y=eda_ds[,i], fill =
Bucket_All_Max_Updated )) +
            geom_boxplot() +
            labs(y = colnames(eda_ds)[i], x = "All Max Bucket") +
            scale_fill_brewer(palette="Dark2") +
            ggtitle(paste(colnames(eda_ds)[i], "vs All Max Bucket")) +
            theme_classic() +
            scale_y_continuous(labels = scales::comma) +
            theme(legend.position = "none")
    )
    dev.off()
   }, error=function(e){cat("ERROR :",conditionMessage(e), "\n")})
  } else{}
}
rm(i)
#############################################################################################


############Extra Graphs and Hypothessi
testing#################################################


ggplot(eda_ds, aes(y = Spending_Limit, x="")) +
  geom_boxplot(fill = "Red")  +
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Spending_Limit", x="") +
  ggtitle("Spending_Limit: Distribution") +
  theme_classic()



ggplot(eda_ds, aes(y = Avg_Payments, x="")) +
  geom_boxplot(fill = "Red")  +
```

```
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Avg_Payments", x="") +
  ggtitle("Avg_Payments: Distribution") +
  theme_classic()


ggplot(eda_ds, aes(y = Avg_Balances, x="")) +
  geom_boxplot(fill = "Red")  +
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Avg_Balances", x="") +
  ggtitle("Avg_Balances: Distribution") +
  theme_classic()


ggplot(eda_ds, aes(y = Py_Amt_Cash, x="")) +
  geom_boxplot(fill = "Red")  +
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Py_Amt_Cash", x="") +
  ggtitle("Py_Amt_Cash: Distribution") +
  theme_classic()


ggplot(eda_ds, aes(y = Ytd_Amt_Cash, x="")) +
  geom_boxplot(fill = "Red")  +
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Ytd_Amt_Cash", x="") +
  ggtitle("Ytd_Amt_Cash: Distribution") +
  theme_classic()
```

```
ggplot(eda_ds, aes(y = Current_Balance, x="")) +
  geom_boxplot(fill = "Red")  +
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Current_Balance", x="") +
  ggtitle("Current_Balance: Distribution") +
  theme_classic()



summary(eda_ds$Avg_Balances)


t.test(
  subset(
    eda_ds$Age_Issued_Updated,
    eda_ds$All_Max_Bucket_Flag_updated == 0),
  subset(
    eda_ds$Age_Issued_Updated,
    eda_ds$All_Max_Bucket_Flag_updated == 1),
  alternative = "greater"
)



t.test(
  subset(
    eda_ds$Spending_Limit,
    eda_ds$All_Max_Bucket_Flag_updated == 0),
  subset(
    eda_ds$Spending_Limit,
    eda_ds$All_Max_Bucket_Flag_updated == 1),
  alternative = "greater"
)
```

```
rm(Prop_Data_Num)

my_data2 <- Filter(is.numeric, eda_ds)

my_data2 <- my_data2[, -c(4,5:7,8,9,12:16,19,20,22)]
my_data2 <- my_data2[, c(4,1:3,5:8)]
my_data2 <- na.omit(my_data2)

M <- cor(na.omit(my_data2))



col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(M, method = "color", col = col(100),
         type = "upper",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "darkblue", tl.srt = 45, #Text label color and rotation
         # Combine with significance level
         # hide correlation coefficient on the principal diagonal
         diag = FALSE
)
plot.new()

corrplot(M, method="number")



col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(M, method="color", col=col(200),
         type="upper", order="hclust",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col="black", tl.srt=45, #Text label color and rotation
```

```
        # Combine with significance
        p.mat = p.mat, sig.level = 0.01, insig = "blank",
        # hide correlation coefficient on the principal diagonal
        diag=FALSE
)



library(psych)

corPlot(my_data2, cex = 1.2)



ggplot(data = eda_ds) +
  geom_histogram(aes(x = Account_Status_Bucket), colour = "black", fill = "white") +
  facet_wrap(~ All_Max_Bucket_updated_fc)



ggplot(eda_ds, aes(y = Current_Balance, x="")) +
  geom_boxplot(fill = "Red")  +
  #geom_text(stat = 'count', aes(label = ..count..), vjust = -1) +
  labs(y="Current_Balance", x="") +
  ggtitle("Current_Balance: Distribution") +
  theme_classic()

x_axis_labels <- min(eda_ds$Account_Status_Bucket):max(eda_ds$Account_Status_Bucket)
ggplot(eda_ds, aes(x = Account_Status_Bucket)) +
  facet_wrap(~ All_Max_Bucket_Flag_updated) +
  #xlab('Account_Status_Bucket')
  geom_bar(fill = "Red")  +
```

```
 scale_x_continuous(breaks = unique(eda_ds$Account_Status_Bucket)) +
 labs(x="Account_Status_Bucket") +
 ggtitle("All Max Flag: Acount Status bucket") +
 theme_classic()


x_axis_labels <- min(eda_ds[,i]):max(eda_ds[,i])
print(ggplot(eda_ds, aes(x = eda_ds[,i])) +
      geom_bar(fill = "Red")  +
      scale_x_continuous(labels = x_axis_labels, breaks = x_axis_labels)


##############################################################################

eda_ds <- readRDS("eda_ds.RDS")


###COnvert flag to factor variables

eda_ds$Sex_M_Flag <- as.factor(eda_ds$Sex_M_Flag)
eda_ds$Blacklist_Flag <- as.factor(eda_ds$Blacklist_Flag)
eda_ds$Attrition_Flag <- as.factor(eda_ds$Attrition_Flag)
eda_ds$Writeoff_Flag <- as.factor(eda_ds$Writeoff_Flag)

eda_ds$Out_Bank_Flag <- as.factor(eda_ds$Out_Bank_Flag)
eda_ds$Ytd_Max_Bucket_Flag <- as.factor(eda_ds$Ytd_Max_Bucket_Flag)
eda_ds$All_Max_Bucket_Flag <- as.factor(eda_ds$All_Max_Bucket_Flag)
eda_ds$All_Max_Bucket_Flag_updated <- as.factor(eda_ds$All_Max_Bucket_Flag_updated)
eda_ds$Bucket_Flag <- as.factor(eda_ds$Bucket_Flag)
eda_ds$Py_Max_Bucket_Flag <- as.factor(eda_ds$Py_Max_Bucket_Flag)
eda_ds$Bucket_All_Max_Updated <- as.factor(eda_ds$Bucket_All_Max_Updated)
```

```
eda_ds$Account_Status_Bucket_Fac <- eda_ds$Account_Status_Bucket

eda_ds$Account_Status_Bucket_Fac <- as.factor(eda_ds$Account_Status_Bucket_Fac)



#####For modeling purpose


eda_ds$All_Max_Bucket_Flag_updated1 <-
as.numeric(as.character(eda_ds$All_Max_Bucket_Flag_updated))


eda_ds[is.na(eda_ds$Spending_Limit),"Spending_Limit"]=mean(eda_ds$Spending_Limit,na.rm=T)

eda_ds[is.na(eda_ds$Avg_Payments),"Avg_Payments"]=mean(eda_ds$Avg_Payments,na.rm=T)

eda_ds[is.na(eda_ds$Avg_Balances),"Avg_Balances"]=mean(eda_ds$Avg_Balances,na.rm=T)

eda_ds[is.na(eda_ds$Py_Amt_Cash),"Py_Amt_Cash"]=mean(eda_ds$Py_Amt_Cash,na.rm=T)

eda_ds[is.na(eda_ds$Ytd_Amt_Cash),"Ytd_Amt_Cash"]=mean(eda_ds$Ytd_Amt_Cash,na.rm=T)

eda_ds[is.na(eda_ds$Current_Balance),"Current_Balance"]=mean(eda_ds$Current_Balance,na.rm=T
)

eda_ds[is.na(eda_ds$Age_Issued_Updated),"Age_Issued_Updated"]=mean(eda_ds$Age_Issued_Up
dated,na.rm=T)

#write.csv(eda_ds,"Check11.csv")

df_train = data.frame()

df_test = data.frame()

for (i in 0:length(unique(eda_ds$All_Max_Bucket_updated))) {

  df1 <- subset(eda_ds, All_Max_Bucket_updated == (i))

  set.seed(3456)

  sample <- sample.int(n = nrow(df1), size = floor(.75*nrow(df1)), replace = F)

  train <- df1[sample, ]

  test  <- df1[-sample, ]

  df_train <- rbind(df_train,train)

  df_test <- rbind(df_test,test)

}

#as.data.frame(table(CS_Data1$All_Max_Bucket_updated))
```

```
rm(df1, test, train, i, sample)

################################################################################

df_test <- subset(df_test,select = -c(Birth_Date,Black_List_Date,Write_Off_Date
                          ,At_Anniv_Date,Last_Trx_Date
                          ,Oy_Amt_Cash
                          ,All_Max_Bucket
                          ,Age_Issued
                          ,Days_Writeoff_blacklist,Days_Blacklist_Membership
                          ,Days_lasttrx_membership,Days_Anniv_membership
                          ,All_Max_Bucket_Flag
                          ,Bucket_All_Max_Updated))
df_train <- subset(df_train,select = -c(Birth_Date,Black_List_Date,Write_Off_Date
                           ,At_Anniv_Date,Last_Trx_Date
                           ,Oy_Amt_Cash
                           ,All_Max_Bucket
                           ,Age_Issued
                           ,Days_Writeoff_blacklist,Days_Blacklist_Membership
                           ,Days_lasttrx_membership,Days_Anniv_membership
                           ,All_Max_Bucket_Flag
                           ,Bucket_All_Max_Updated))
saveRDS(df_test, file = "CS_Test.RDS")
saveRDS(df_train, file = "CS_Train.RDS")
################################################################################
####
############Modeling Linear ###################################
#####Application scoring - Flag############################

model1 =
lm(All_Max_Bucket_Flag_updated1~Branch_Code_Rating+Age_Issued_Updated+Membership_Date
+Sex_M_Flag+Out_Bank_Flag,
```

```
          data=df_train)
model1
stargazer(model1,  type="text", align=TRUE)
###Poor R2 as expected becuase not much variation present in the data
plot(x=model1$fitted.values, y=df_train$All_Max_Bucket_Flag_updated1, col="blue")
###Same thing can be noticed in the graph

##Check for assumptions
autoplot(model1)
####residual clear pattern expected violation
##Non linear qunatile - not normal data ##expected violation
###scale location pattern even after standardization of residuals
#very few influential points noticed



######################Check poly for the age issued ###################################

n = nrow(df_train)

max_degree = 5
PRESS = vector(length=max_degree)

for(k in 1:max_degree){
  cv_k = vector(length=n)
  for(i in 1:n){
    fiti = lm(All_Max_Bucket_Flag_updated1 ~
poly(Age_Issued_Updated,k)+Branch_Code_Rating+Membership_Date+Sex_M_Flag+Out_Bank_Flag
,
              data=df_train[-i,])
    predi = predict(fiti, newdata=df_train[i,])
    cv_k[i] = df_train$All_Max_Bucket_Flag_updated1[i] - predi
  }
```

```
  PRESS[k] = sum(cv_k^2)
}



plot(x=1:max_degree, y=PRESS)
####Best for 1 degree only


rm(cv_k, i, k, max_degree,n, predi, fiti, PRESS)
################################################################################


model1 =
lm(All_Max_Bucket_Flag_updated1~Branch_Code_Rating+Age_Issued_Updated+Membership_Date
+Sex_M_Flag+Out_Bank_Flag,
          data=df_train)
model1
stargazer(model1,  type="text", align=TRUE)


###
####Calculate pnl for the threshold to figure out the best model#####
test_pred=cbind(as.data.frame(predict(model1,
newdata=df_test)),df_test$All_Max_Bucket_Flag_updated1)
colnames(test_pred) <- c('Predicted_All_Max_Bucket_Flag', 'Actual_All_Max_Bucket_Flag')


result = data.frame()


i=.4
for (i in 1:10) {
  test_pred<-within(test_pred,{Predicted_All_Max_Bucket_Flag1<-
ifelse(Predicted_All_Max_Bucket_Flag>= i*0.1, 1,0)})
  test_pred1 <- dcast(test_pred, Actual_All_Max_Bucket_Flag ~ Predicted_All_Max_Bucket_Flag1,
                value.var="Predicted_All_Max_Bucket_Flag1", fun.aggregate=length)
  Accu <- (test_pred1[1,2]+test_pred1[2,3])/nrow(df_test)
```

```
  Sq_sen_spec <- ((test_pred1[1,2]/(test_pred1[1,2] + test_pred1[1,3])) *
(test_pred1[2,3]/(test_pred1[2,2] + test_pred1[2,3])))^0.5

  PnL <- ((test_pred1[1,2]/(test_pred1[1,2] + test_pred1[2,2]))*0.02) +

    ((test_pred1[2,2]/(test_pred1[1,2] + test_pred1[2,2]))*-0.45)


  result1 <- as.data.frame(cbind(i*0.1, Accu, Sq_sen_spec, PnL))

  result <- rbind(result,result1)

  rm(Accu, Sq_sen_spec, PnL,i,result1)

}

rm(Accu, Sq_sen_spec, PnL,i,result1)

##################################################


write.csv(result, "Result1.csv")

write.csv(test_pred1, "CM1.csv")


###Best for 0.3 value, high accuracy and good pnl


rm(model1,test_pred, test_pred1, result)


#####################################################################

##########Application - All max bucket


model2 =
lm(All_Max_Bucket_updated~Branch_Code_Rating+Age_Issued_Updated+Membership_Date+Sex_
M_Flag+Out_Bank_Flag,

          data=df_train)

#model2

summary(model2)

stargazer(model2,  type="text", align=TRUE)


###Poor R2 as expected becuase not much variation present in the data

plot(x=model2$fitted.values, y=df_train$All_Max_Bucket_updated, col="blue")
```

```
###values going in -ive range also ... max is 3 so need to assign the buckets accordingly


##Check for assumptions

autoplot(model2)

#####values in -ive

##Non linear qunatile - not normal data ##expected violation

#very few influential points noticed



test_pred=cbind(as.data.frame(predict(model2,
newdata=df_test)),df_test$All_Max_Bucket_updated)

colnames(test_pred) <- c('Predicted_All_Max_Bucket', 'Actual_All_Max_Bucket')


####New bucket based on predicted values


test_pred$Predicted_All_Max_Bucket1 <- ifelse(test_pred[,1] < 0, 0, round(test_pred[,1]))


result = data.frame()


test_pred1 <- dcast(test_pred, Actual_All_Max_Bucket ~ Predicted_All_Max_Bucket1,

                value.var="Predicted_All_Max_Bucket1", fun.aggregate=length)


Accu <-
(test_pred1[1,2]+test_pred1[1,3]+sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5]))/nrow(df_test)

Sq_sen_spec <- (((test_pred1[1,2]+test_pred1[1,3])/sum(test_pred1[1,])) *

              ((sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5]))/

                (sum(test_pred1[2:7,2])+sum(test_pred1[2:7,3])+

                  sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5]))))^0.5

a <- sum(test_pred1[,2]) + sum(test_pred1[,3])

PnL <- ((((test_pred1[2,2]+test_pred1[2,3])/a)*-2) + (((test_pred1[3,2]+test_pred1[3,3])/a)*-3) +
(((test_pred1[4,2]+test_pred1[4,3])/a)*-4) + (((test_pred1[5,2]+test_pred1[5,3])/a)*-5) +
(((test_pred1[6,2]+test_pred1[6,3])/a)*-6) + (((test_pred1[7,2]+test_pred1[7,3])/a)*-7))
```

```
result <- as.data.frame(cbind("Actual", Accu, Sq_sen_spec, PnL))
#result <- rbind(result,result1)
rm(a,Accu, Sq_sen_spec, PnL,test_pred1)
write.csv(test_pred1, "CM2.csv")

#####Different bucketing instead of round off appraoch

a <- count(eda_ds, 'All_Max_Bucket_updated')
a$Perc <- a$freq / sum(a$freq) * 100

####Based on the actual distribution of the values of all max bucket
test_pred <- test_pred %>% mutate(Predicted_All_Max_Bucket2 = cut(Predicted_All_Max_Bucket,
                                        quantile(Predicted_All_Max_Bucket,
                                              c(0,.76, .86, .93, .95,.97,.985,1)),
                                        labels=c(0,2,3,4,5,6,7),
                                        include.lowest = TRUE))
rm(a)


test_pred1 <- dcast(test_pred, Actual_All_Max_Bucket ~ Predicted_All_Max_Bucket2,
                    value.var="Predicted_All_Max_Bucket2", fun.aggregate=length)

Accu <- (test_pred1[1,2]+test_pred1[2,3]+test_pred1[3,4]+test_pred1[4,5]+
           test_pred1[5,6]+test_pred1[6,7]+test_pred1[7,8])/nrow(df_test)
Sq_sen_spec <- (((test_pred1[1,2]/sum(test_pred1[1,])) *

((sum(test_pred1[2:7,3])+sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])+sum(test_pred1[2:7,6])
                    +sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8]))/

(sum(test_pred1[2:7,2])+sum(test_pred1[2:7,3])+sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])+su
m(test_pred1[2:7,6])
                     +sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8])))))^0.5
```

```
a <- sum(test_pred1[,2])
PnL <- (((test_pred1[2,2]/a)*-2) + ((test_pred1[3,2]/a)*-3) + ((test_pred1[4,2]/a)*-4)
      + ((test_pred1[5,2]/a)*-5) + ((test_pred1[6,2]/a)*-6) + ((test_pred1[7,2]/a)*-7))
result1 <- as.data.frame(cbind("Calcu", Accu, Sq_sen_spec, PnL))
result <- rbind(result,result1)
write.csv(result, "Result2.csv")

rm(a,Accu, Sq_sen_spec, PnL,test_pred1,test_pred,model2,result1,result)

#############################################################################
Application - PnL All max bucket

model3 =
lm(PNL_All_Max_Bucket~Branch_Code_Rating+Age_Issued_Updated+Membership_Date+Sex_M_Flag+Out_Bank_Flag,
        data=df_train)
#model3
summary(model3)
stargazer(model3,  type="text", align=TRUE)

###Poor R2 as expected becuase not much variation present in the data
plot(x=model3$fitted.values, y=df_train$PNL_All_Max_Bucket, col="blue")
###values going in +ive range also ... max is .03 so need to assign the buckets accordingly

##Check for assumptions
autoplot(model3)
#####values outside 0.03 but downward pattern noticed
##Non linear qunatile - not normal data ##expected violation
# a couple of influential points noticed

test_pred=cbind(as.data.frame(predict(model3, newdata=df_test)),df_test$PNL_All_Max_Bucket)
colnames(test_pred) <- c('Predicted_PNL_All_Max_Bucket', 'Actual_PNL_All_Max_Bucket')
```

```
#write.csv(test_pred, "checkl.csv")

####New bucket based on predicted values

lbls <- c(-0.65,-0.45,-0.35,-0.25,-0.15,0,0.03)
test_pred$Predicted_PNL_All_Max_Bucket1 <- cut(test_pred[ ,1],
                                breaks = c(-Inf,-0.65,-0.45,-0.35,-0.25,-0.15,0,+Inf), right = TRUE,
                                include.lowest = TRUE, labels = lbls)
rm(lbls)



result = data.frame()

test_pred1 <- dcast(test_pred, Actual_PNL_All_Max_Bucket ~ Predicted_PNL_All_Max_Bucket1,
                value.var="Predicted_PNL_All_Max_Bucket1", fun.aggregate=length)

Accu <- (test_pred1[7,4]+test_pred1[6,3]+test_pred1[5,2])/nrow(df_test)
Sq_sen_spec <- (((test_pred1[7,4]/sum(test_pred1[7,2:4])) *
              ((sum(test_pred1[1:6,2])+sum(test_pred1[1:6,3]))/
                (sum(test_pred1[1:6,2])+sum(test_pred1[1:6,3])+
                   sum(test_pred1[1:6,4])))))^0.5

a <- sum(test_pred1[,4])
PnL <- ((test_pred1[7,4]/a)*test_pred1[7,1]) + ((test_pred1[6,4]/a)*test_pred1[6,1])+
((test_pred1[5,4]/a)*test_pred1[5,1])+ ((test_pred1[4,4]/a)*test_pred1[4,1])+
((test_pred1[3,4]/a)*test_pred1[3,1])+ ((test_pred1[2,4]/a)*test_pred1[2,1])+
((test_pred1[1,4]/a)*test_pred1[1,1])
result <- as.data.frame(cbind("Actual", Accu, Sq_sen_spec, PnL))
#result <- rbind(result,result1)
rm(a,Accu, Sq_sen_spec, PnL,test_pred1)
write.csv(test_pred1, "CM3.csv")
```

```
#####Different bucketing instead of round off appraoch


a <- count(eda_ds, 'All_Max_Bucket_updated')

a$Perc <- a$freq / sum(a$freq) * 100


####Based on the actual distribution of the values of all max bucket

test_pred <- test_pred %>% mutate(Predicted_PNL_All_Max_Bucket2 =
cut(Predicted_PNL_All_Max_Bucket,
                                          quantile(Predicted_PNL_All_Max_Bucket,
                                                   c(0,.015, .03, .05, .07,.14,.24,1)),
                                          labels=c(-0.65, -0.45,-0.35,-0.25,-0.15,0,0.03),
                                          include.lowest = TRUE))

rm(a)


test_pred1 <- dcast(test_pred, Actual_PNL_All_Max_Bucket ~ Predicted_PNL_All_Max_Bucket2,
                    value.var="Predicted_PNL_All_Max_Bucket2", fun.aggregate=length)


Accu <- (test_pred1[1,2]+test_pred1[2,3]+test_pred1[3,4]+test_pred1[4,5]+
           test_pred1[5,6]+test_pred1[6,7]+test_pred1[7,8])/nrow(df_test)

Sq_sen_spec <- (((test_pred1[7,8]/sum(test_pred1[7,2:8])) *

((sum(test_pred1[1:6,3])+sum(test_pred1[1:6,4])+sum(test_pred1[1:6,5])+sum(test_pred1[1:6,6])
                    +sum(test_pred1[1:6,7])+sum(test_pred1[1:6,2]))/

(sum(test_pred1[1:6,2])+sum(test_pred1[1:6,3])+sum(test_pred1[1:6,4])+sum(test_pred1[1:6,5])+su
m(test_pred1[1:6,6])
                     +sum(test_pred1[1:6,7])+sum(test_pred1[1:6,8])))))^0.5

a <- sum(test_pred1[,8])

PnL <- ((test_pred1[7,8]/a)*test_pred1[7,1]) + ((test_pred1[6,8]/a)*test_pred1[6,1])+
((test_pred1[5,8]/a)*test_pred1[5,1])+ ((test_pred1[4,8]/a)*test_pred1[4,1])+
((test_pred1[3,8]/a)*test_pred1[3,1])+ ((test_pred1[2,8]/a)*test_pred1[2,1])+
((test_pred1[1,8]/a)*test_pred1[1,1])
```

```
result1 <- as.data.frame(cbind("Calcu", Accu, Sq_sen_spec, PnL))
result <- rbind(result,result1)
write.csv(result, "Result3.csv")

###Bad results in the calulated once since we don;t have -65% in the original models
###but are there in the calculated ones beciase of the qunatile bucketing

rm(a,Accu, Sq_sen_spec, PnL,test_pred1,test_pred,model3,result1,result)

##############################################################################
########Application scoring#########################################################
#######Flag ########################

model11 = lm(All_Max_Bucket_Flag_updated1~
             Spending_Limit+
             #Avg_Payments+
             #Avg_Balances+
             #Py_Amt_Cash+
             #Ytd_Amt_Cash+
             #Current_Balance+
             Membership_Date+
             Sex_M_Flag+
             Blacklist_Flag+
             Attrition_Flag+
             Writeoff_Flag+
             Out_Bank_Flag+
             Account_Status_Bucket+
             Age_Issued_Updated+
             Branch_Code_Rating,
           data=df_train)
stargazer(model11,  type="text", align=TRUE)
```

```
###Poor R2 as expected becuase not much variation present in the data

plot(x=model11$fitted.values, y=df_train$All_Max_Bucket_Flag_updated1, col="blue")

###Same thing can be noticed in the graph


##Check for assumptions

autoplot(model11)

##Non linear qunatile - not normal data ##expected violation

###scale location pattern even after standardization of residuals

#Many influential points noticed - can try remove them and re-run the model



###

####Calculate pnl for the threshold to figure out the best model#####

test_pred=cbind(as.data.frame(predict(model11,
newdata=df_test)),df_test$All_Max_Bucket_Flag_updated1)

colnames(test_pred) <- c('Predicted_All_Max_Bucket_Flag', 'Actual_All_Max_Bucket_Flag')


result = data.frame()


for (i in 1:10) {

 test_pred<-within(test_pred,{Predicted_All_Max_Bucket_Flag1<-
ifelse(Predicted_All_Max_Bucket_Flag>= i*0.1, 1,0)})

 test_pred1 <- dcast(test_pred, Actual_All_Max_Bucket_Flag ~ Predicted_All_Max_Bucket_Flag1,

                value.var="Predicted_All_Max_Bucket_Flag1", fun.aggregate=length)

 Accu <- (test_pred1[1,2]+test_pred1[2,3])/nrow(df_test)

 Sq_sen_spec <- ((test_pred1[1,2]/(test_pred1[1,2] + test_pred1[1,3])) *
(test_pred1[2,3]/(test_pred1[2,2] + test_pred1[2,3])))^0.5

 PnL <- ((test_pred1[1,2]/(test_pred1[1,2] + test_pred1[2,2]))*0.02) +

  ((test_pred1[2,2]/(test_pred1[1,2] + test_pred1[2,2]))*-0.45)


 result1 <- as.data.frame(cbind(i*0.1, Accu, Sq_sen_spec, PnL))
```

```
  result <- rbind(result,result1)
  rm(Accu, Sq_sen_spec, PnL,i,result1)
}
rm(Accu, Sq_sen_spec, PnL,i,result1)
#####################################################
write.csv(test_pred1, "CM11.csv")
write.csv(result, "Result11.csv")

###Best for 0.3 value, high accuracy and good pnl

rm(model11,test_pred, test_pred1, result)

#####################################################################
#####################################################################
##########Behavior - All max bucket

model21 = lm(All_Max_Bucket_updated~Spending_Limit+
             Avg_Payments+
             Avg_Balances+
             Ytd_Amt_Cash+
             Current_Balance+
             Membership_Date+
             Sex_M_Flag+
             Blacklist_Flag+
             Attrition_Flag+
             Writeoff_Flag+
             Out_Bank_Flag+
             Account_Status_Bucket+
             Age_Issued_Updated+
             Branch_Code_Rating,
           data=df_train)
```

```
summary(model21)
stargazer(model21,  type="text", align=TRUE)

###Poor R2 as expected becuase not much variation present in the data
plot(x=model21$fitted.values, y=df_train$All_Max_Bucket_updated, col="blue")
###values going in -ive range also ..so need to assign the buckets accordingly

##Check for assumptions
autoplot(model21)
#####values in -ive seems some pattern
##Non linear qunatile - not normal data ##expected violation
#lot of influential points noticed

test_pred=cbind(as.data.frame(predict(model21,
newdata=df_test)),df_test$All_Max_Bucket_updated)
colnames(test_pred) <- c('Predicted_All_Max_Bucket', 'Actual_All_Max_Bucket')

####New bucket based on predicted values

test_pred$Predicted_All_Max_Bucket1 <- ifelse(test_pred[,1] < 0, 0, ifelse(test_pred[,1] > 7, 7,
round(test_pred[,1])))

result = data.frame()

test_pred1 <- dcast(test_pred, Actual_All_Max_Bucket ~ Predicted_All_Max_Bucket1,
                    value.var="Predicted_All_Max_Bucket1", fun.aggregate=length)

Accu <- (test_pred1[1,2]+test_pred1[1,3]+sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])
      +sum(test_pred1[2:7,6])+sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8]))/nrow(df_test)
Sq_sen_spec <- (((test_pred1[1,2]+test_pred1[1,3])/sum(test_pred1[1,])) *
```

```
            ((sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])+sum(test_pred1[2:7,6])
              +sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8]))/
               (sum(test_pred1[2:7,2])+sum(test_pred1[2:7,3])+
                  sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])+sum(test_pred1[2:7,6])
               +sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8])))^0.5
a <- sum(test_pred1[,2]) + sum(test_pred1[,3])
PnL <- ((((test_pred1[2,2]+test_pred1[2,3])/a)*-2) + (((test_pred1[3,2]+test_pred1[3,3])/a)*-3)
      + (((test_pred1[4,2]+test_pred1[4,3])/a)*-4) + (((test_pred1[5,2]+test_pred1[5,3])/a)*-5)
      + (((test_pred1[6,2]+test_pred1[6,3])/a)*-6) + (((test_pred1[7,2]+test_pred1[7,3])/a)*-7))
result <- as.data.frame(cbind("Actual", Accu, Sq_sen_spec, PnL))
#result <- rbind(result,result1)
rm(a,Accu, Sq_sen_spec, PnL,test_pred1)

#####Different bucketing instead of round off appraoch

a <- count(eda_ds, 'All_Max_Bucket_updated')
a$Perc <- a$freq / sum(a$freq) * 100

####Based on the actual distribution of the values of all max bucket
test_pred <- test_pred %>% mutate(Predicted_All_Max_Bucket2 = cut(Predicted_All_Max_Bucket,
                                        quantile(Predicted_All_Max_Bucket,
                                              c(0,.76, .86, .93, .95,.97,.985,1)),
                                        labels=c(0,2,3,4,5,6,7),
                                        include.lowest = TRUE))
rm(a)


test_pred1 <- dcast(test_pred, Actual_All_Max_Bucket ~ Predicted_All_Max_Bucket2,
              value.var="Predicted_All_Max_Bucket2", fun.aggregate=length)

Accu <- (test_pred1[1,2]+test_pred1[2,3]+test_pred1[3,4]+test_pred1[4,5]+
```

```
        test_pred1[5,6]+test_pred1[6,7]+test_pred1[7,8])/nrow(df_test)
Sq_sen_spec <- (((test_pred1[1,2]/sum(test_pred1[1,])) *

((sum(test_pred1[2:7,3])+sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])+sum(test_pred1[2:7,6])
              +sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8]))/

(sum(test_pred1[2:7,2])+sum(test_pred1[2:7,3])+sum(test_pred1[2:7,4])+sum(test_pred1[2:7,5])+su
m(test_pred1[2:7,6])
               +sum(test_pred1[2:7,7])+sum(test_pred1[2:7,8])))))^0.5
a <- sum(test_pred1[,2])
PnL <- (((test_pred1[2,2]/a)*-2) + ((test_pred1[3,2]/a)*-3) + ((test_pred1[4,2]/a)*-4)
     + ((test_pred1[5,2]/a)*-5) + ((test_pred1[6,2]/a)*-6) + ((test_pred1[7,2]/a)*-7))
result1 <- as.data.frame(cbind("Calcu", Accu, Sq_sen_spec, PnL))
result <- rbind(result,result1)
write.csv(result, "Result21.csv")

rm(a,Accu, Sq_sen_spec, PnL,test_pred1,test_pred,model21,result1,result)

#############################################################################
###############Behavior profit and loss#######################################

model31 = lm(PNL_All_Max_Bucket~Spending_Limit+
          Avg_Payments+
          Avg_Balances+
          Py_Amt_Cash+
          Ytd_Amt_Cash+
          Current_Balance+
          Membership_Date+
          Sex_M_Flag+
          Blacklist_Flag+
          Attrition_Flag+
          Writeoff_Flag+
```

```
            Out_Bank_Flag+
            Account_Status_Bucket+
            Age_Issued_Updated+
            Branch_Code_Rating,
          data=df_train)

stargazer(model31,  type="text", align=TRUE)



###Poor R2 as expected becuase not much variation present in the data
plot(x=model31$fitted.values, y=df_train$PNL_All_Max_Bucket, col="blue")
###values going outside -ive range... so need to assign the buckets accordingly

##Check for assumptions
autoplot(model31)
#####values outside -0.65...pattern noticed
##Non linear qunatile - not normal data ##expected violation
# too many influential points noticed

test_pred=cbind(as.data.frame(predict(model31, newdata=df_test)),df_test$PNL_All_Max_Bucket)
colnames(test_pred) <- c('Predicted_PNL_All_Max_Bucket', 'Actual_PNL_All_Max_Bucket')

#write.csv(test_pred, "checkl.csv")

####New bucket based on predicted values

lbls <- c(-0.65,-0.45,-0.35,-0.25,-0.15,0,0.03)
test_pred$Predicted_PNL_All_Max_Bucket1 <- cut(test_pred[ ,1],
                                breaks = c(-Inf,-0.65,-0.45,-0.35,-0.25,-0.15,0,+Inf), right = TRUE,
                                include.lowest = TRUE, labels = lbls)
rm(lbls)
```

```
result = data.frame()

test_pred1 <- dcast(test_pred, Actual_PNL_All_Max_Bucket ~ Predicted_PNL_All_Max_Bucket1,
                    value.var="Predicted_PNL_All_Max_Bucket1", fun.aggregate=length)

Accu <-
(test_pred1[7,7]+test_pred1[6,6]+test_pred1[5,5]+test_pred1[1,2]+test_pred1[2,3])/nrow(df_test)
Sq_sen_spec <- (((test_pred1[7,7]/sum(test_pred1[7,2:7])) *
                  ((sum(test_pred1[1:6,2])+sum(test_pred1[1:6,3])+sum(test_pred1[1:6,4])+
                      sum(test_pred1[1:6,5])+sum(test_pred1[1:6,6]))/
                     (sum(test_pred1[1:6,2])+sum(test_pred1[1:6,3])+sum(test_pred1[1:6,4])+
                        sum(test_pred1[1:6,5])+sum(test_pred1[1:6,6])+sum(test_pred1[1:6,7])))))^0.5

a <- sum(test_pred1[,7])
PnL <- ((test_pred1[7,7]/a)*test_pred1[7,1]) + ((test_pred1[6,7]/a)*test_pred1[6,1])+
((test_pred1[5,7]/a)*test_pred1[5,1])+ ((test_pred1[4,7]/a)*test_pred1[4,1])+
((test_pred1[3,7]/a)*test_pred1[3,1])+ ((test_pred1[2,7]/a)*test_pred1[2,1])+
((test_pred1[1,7]/a)*test_pred1[1,1])
result <- as.data.frame(cbind("Actual", Accu, Sq_sen_spec, PnL))
#result <- rbind(result,result1)
rm(a,Accu, Sq_sen_spec, PnL,test_pred1)

#####Different bucketing instead of round off appraoch

a <- count(eda_ds, 'All_Max_Bucket_updated')
a$Perc <- a$freq / sum(a$freq) * 100

####Based on the actual distribution of the values of all max bucket
test_pred <- test_pred %>% mutate(Predicted_PNL_All_Max_Bucket2 =
cut(Predicted_PNL_All_Max_Bucket,
                                  quantile(Predicted_PNL_All_Max_Bucket,
```

```
                                        c(0,.015, .03, .05, .07,.14,.24,1)),
                                      labels=c(-0.65, -0.45,-0.35,-0.25,-0.15,0,0.03),
                                      include.lowest = TRUE))
rm(a)

test_pred1 <- dcast(test_pred, Actual_PNL_All_Max_Bucket ~ Predicted_PNL_All_Max_Bucket2,
            value.var="Predicted_PNL_All_Max_Bucket2", fun.aggregate=length)

Accu <- (test_pred1[1,2]+test_pred1[2,3]+test_pred1[3,4]+test_pred1[4,5]+
      test_pred1[5,6]+test_pred1[6,7]+test_pred1[7,8])/nrow(df_test)
Sq_sen_spec <- (((test_pred1[7,8]/sum(test_pred1[7,2:8])) *

((sum(test_pred1[1:6,3])+sum(test_pred1[1:6,4])+sum(test_pred1[1:6,5])+sum(test_pred1[1:6,6])
            +sum(test_pred1[1:6,7])+sum(test_pred1[1:6,2]))/

(sum(test_pred1[1:6,2])+sum(test_pred1[1:6,3])+sum(test_pred1[1:6,4])+sum(test_pred1[1:6,5])+su
m(test_pred1[1:6,6])
             +sum(test_pred1[1:6,7])+sum(test_pred1[1:6,8])))))^0.5
a <- sum(test_pred1[,8])
PnL <- ((test_pred1[7,8]/a)*test_pred1[7,1]) + ((test_pred1[6,8]/a)*test_pred1[6,1])+
((test_pred1[5,8]/a)*test_pred1[5,1])+ ((test_pred1[4,8]/a)*test_pred1[4,1])+
((test_pred1[3,8]/a)*test_pred1[3,1])+ ((test_pred1[2,8]/a)*test_pred1[2,1])+
((test_pred1[1,8]/a)*test_pred1[1,1])
result1 <- as.data.frame(cbind("Calcu", Accu, Sq_sen_spec, PnL))
result <- rbind(result,result1)
write.csv(result, "Result31.csv")

###Bad results in the calulated once since very few -65 bucket values present in the
######orginal model
###but are there in the calculated ones beciase of the qunatile bucketing

rm(a,Accu, Sq_sen_spec, PnL,test_pred1,test_pred,model31,result1,result)
```

Logistic regression part

```
#————————load data

CS_Data1<-readRDS("E:/R/group project/CS_Data1.RDS")

CS_Test<-readRDS("E:/R/group project/CS_Test.RDS")

CS_Train<-readRDS("E:/R/group project/CS_Train.RDS")

View(CS_Data1)

glimpse(CS_Data1)

#———logistic regression/ 0&1 classification/application scoring

log_model_multi <-
glm(All_Max_Bucket_Flag_updated~Age_Issued_Updated+Sex_M_Flag+Out_Bank_Flag+
Branch_Code_Rating, family="binomial", data=CS_Train)

summary(log_model_multi)

model <- step(object=log_model_multi,trace=0)

summary(model)


#The model's prediction accuracy on  test set

prob<-predict(object=model,newdata=CS_Test,type="response")

summary(prob)


#ROC

roc_curve<-roc(CS_Test$All_Max_Bucket_Flag_updated,prob)

roc_result<-coords(roc_curve,"best")

roc_result

names(roc_curve)

x<-1-roc_curve$specificities

y<-roc_curve$sensitivities

p2<-ggplot(data=NULL,aes(x=x,y=y))+geom_line(color="red")+geom_abline(intercept=0,slope=1)+

  annotate("text",x=0.5,y=0.5,label=paste("AUC=",round(roc_curve$auc,2)))+labs(x='1-
specificities',y='sensitivities',title='ROC Curve')

p2

#auc=0.6, not accurate,remove some variable
```

```
result = data.frame()
for (i in 1:10){
pred<-ifelse(prob>=i*0.1,1,0)
  pred<-factor(pred,levels=c(0,1),order=TRUE)
  f<-table(CS_Test$All_Max_Bucket_Flag_updated,pred)

  #sensitivity&specificity

  sensitivity<-f[2,2]/(f[2,2]+f[2,1])
  specificitiy<-f[1,1]/(f[1,2]+f[1,1])
sensi_spe<-(sensitivity*specificitiy)^(1/2)
  accuracy=(f[1,1]+f[2,2])/(f[1,1]+f[2,2]+f[2,1]+f[1,2])
  PnL <- ((f[1,1]/(f[1,1] +f[2,1]))*0.02) +
    ((f[2,1]/(f[1,1] + f[2,1]))*-0.45)
  result1 <- as.data.frame(cbind(i*0.1, accuracy, sensi_spe, PnL))
  result <- rbind(result,result1)
  rm(accuracy, sensi_spe, PnL,i,result1)
  }
result

pred<-ifelse(prob>=0.4,1,0)
pred<-factor(pred,levels=c(0,1),order=TRUE)
f<-table(CS_Test$All_Max_Bucket_Flag_updated,pred)
table(pred)

#sensitivity&specificity

sensitivity<-f[2,2]/(f[2,2]+f[2,1])
specificitiy<-f[1,1]/(f[1,2]+f[1,1])
sensi_spe<-(sensitivity*specificitiy)^(1/2)
accuracy=(f[1,1]+f[2,2])/(f[1,1]+f[2,2]+f[2,1]+f[1,2])
```

```
PnL <- ((f[1,1]/(f[1,1] +f[2,1]))*0.02) +
  ((f[2,1]/(f[1,1] + f[2,1]))*-0.45)


sensi_spe

#bucket
prediction_____________________________________________________________________________
_

#Data clean

CS_Train$All_Max_Bucket_updated2<-relevel(as.factor(CS_Train$All_Max_Bucket_updated),ref=1L)

CS_Train[is.na(CS_Train$Age_Issued_Updated),"Age_Issued_Updated"]=mean(CS_Train$Age_Issued
_Updated,na.rm=T)

CS_Train[is.na(CS_Train$Sex_M_Flag ),"Sex_M_Flag "]=mean(CS_Train$Sex_M_Flag ,na.rm=T)

CS_Train[is.na(CS_Train$Blacklist_Flag),"Blacklist_Flag"]=mean(CS_Train$Blacklist_Flag,na.rm=T)

CS_Train[is.na(CS_Train$Out_Bank_Flag),"Out_Bank_Flag"]=mean(CS_Train$Out_Bank_Flag,na.rm=
T)

CS_Train[is.na(CS_Train$Branch_Code_Rating),"Branch_Code_Rating"]=mean(CS_Train$Branch_Cod
e_Rating,na.rm=T)

#clean data_test

CS_Test$All_Max_Bucket_updated2<-relevel(as.factor(CS_Test$All_Max_Bucket_updated),ref=1L)

CS_Test[is.na(CS_Test$Age_Issued_Updated),"Age_Issued_Updated"]=mean(CS_Test$Age_Issued_U
pdated,na.rm=T)

CS_Test[is.na(CS_Test$Sex_M_Flag ),"Sex_M_Flag "]=mean(CS_Test$Sex_M_Flag ,na.rm=T)

CS_Test[is.na(CS_Test$Blacklist_Flag),"Blacklist_Flag"]=mean(CS_Test$Blacklist_Flag,na.rm=T)

CS_Test[is.na(CS_Test$Out_Bank_Flag),"Out_Bank_Flag"]=mean(CS_Test$Out_Bank_Flag,na.rm=T)

CS_Test[is.na(CS_Test$Branch_Code_Rating),"Branch_Code_Rating"]=mean(CS_Test$Branch_Code_
Rating,na.rm=T)



#___prediction_linear regression

#preparation

View(CS_Train)

View(CS_Test)
```

```
mult.model <- lm(All_Max_Bucket_updated
~Age_Issued_Updated+Sex_M_Flag+Blacklist_Flag+Out_Bank_Flag+ Branch_Code_Rating,
data=CS_Train)

mult.model <- step(object=mult.model,trace=0)

summary(mult.model)

#Make prediction

pre_logistic<-predict(mult.model,newdata=CS_Test)

summary(pre_logistic)

#roc

roc_curve2<-roc(CS_Test$All_Max_Bucket_updated,pre_logistic)

roc_result2<-coords(roc_curve2,"best")

roc_result2

names(roc_curve2)

x2<-1-roc_curve2$specificities

y2<-roc_curve2$sensitivities

p2<-ggplot(data=NULL,aes(x=x2,y=y2))+geom_line(color="red")+geom_abline(intercept=0,slope=1)+

  annotate("text",x=0.5,y=0.5,label=paste("AUC=",round(roc_curve2$auc,2)))+labs(x='1-
specificities',y='sensitivities',title='ROC Curve')

p2

#auc=0.55

pred2<-ifelse(pre_logistic>0.8041307,0,1)

table(pred2)

t1<-table(CS_Test$All_Max_Bucket_Flag_updated,pred2)

t1

#(Specificity*sensitivity)^(1/2)

(t1[1,1]/sum(t1[1,])*t1[2,2]/sum(t1[2,]))^(1/2)



#Profit and Lost

mult.model_0 <- multinom(All_Max_Bucket_updated
~Age_Issued_Updated+Sex_M_Flag+Blacklist_Flag+Out_Bank_Flag+ Branch_Code_Rating,
data=CS_Train)

mult.model_0 <- step(object=mult.model_0,trace=0)
```

```
summary(mult.model_0)

pre_logistic_0<-predict(mult.model_0,newdata=CS_Test)

summary(pre_logistic_0)

f2<-table(pre_logistic_0)

f2

(-2)*(f2[2]/sum(f2))+(-3)*(f2[3]/sum(f2))+(-4)*(f2[4]/sum(f2))+(-5)*(f2[5]/sum(f2))+(-
6)*(f2[6]/sum(f2))+(-7)*(f2[7]/sum(f2))


#PNL

mult.model1 <- lm(PNL_All_Max_Bucket ~
Age_Issued_Updated+Sex_M_Flag+Blacklist_Flag+Out_Bank_Flag+ Branch_Code_Rating,
data=CS_Train)

mult.model1 <- step(object=mult.model1,trace=0)

summary(mult.model1)

#prediction

pre_logistic1<-predict(mult.model1,newdata=CS_Test)

summary(pre_logistic1)

roc_curve3<-roc(CS_Test$PNL_All_Max_Bucket,pre_logistic1)

roc_result3<-coords(roc_curve3,"best")

roc_result3

names(roc_curve3)

x3<-1-roc_curve3$specificities

y3<-roc_curve3$sensitivities

p3<-ggplot(data=NULL,aes(x=x3,y=y3))+geom_line(color="red")+geom_abline(intercept=0,slope=1)+

  annotate("text",x=0.5,y=0.5,label=paste("AUC=",round(roc_curve3$auc,2)))+labs(x='1-
specificities',y='sensitivities',title='ROC Curve')

p3

#AUC=0.7

pre_logistic1

f3<-ifelse(pre_logistic1<(-0.1351264),1,0)

t_2<-table(CS_Test$All_Max_Bucket_Flag_updated,f3)

t_2
```

```
#(Specificity*sensitivity)^(1/2)

((t_2[1,1]/sum(t_2[1,]))*(t_2[2,2]/sum(t_2[2,])))^(1/2)

#Profit and Lost

mult.model1_0 <- multinom(PNL_All_Max_Bucket ~
Age_Issued_Updated+Sex_M_Flag+Blacklist_Flag+Out_Bank_Flag+ Branch_Code_Rating,
data=CS_Train)

mult.model1_0 <- step(object=mult.model1_0,trace=0)

summary(mult.model1)

pre_logistic1_0<-predict(mult.model1_0,newdata=CS_Test)

summary(pre_logistic1_0)

f3<-table(pre_logistic1_0)

f3


0.03*(f3[7]/sum(f3))+(-0.15)*(f3[5]/sum(f3))+(-0.25)*(f3[4]/sum(f3))+(-0.35)*(f3[3]/sum(f3))+(-
0.45)*(f3[2]/sum(f3))+(-0.65)*(f3[1]/sum(f3))


#——————————————————————————————————————————
——————————————————————————————————————————
——————————————

#———logistic regression/ 0&1 classification/behaviour scoring

#clean data_train

CS_Train[is.na(CS_Train$Oy_Amt_Cash),"Oy_Amt_Cash"]=mean(CS_Train$Oy_Amt_Cash,na.rm=T)

CS_Train[is.na(CS_Train$All_Max_Bucket_updated),"All_Max_Bucket_updated"]=mean(CS_Train$All
_Max_Bucket_updated,na.rm=T)

CS_Train[is.na(CS_Train$Py_Amt_Cash),"Py_Amt_Cash"]=mean(CS_Train$Py_Amt_Cash,na.rm=T)

CS_Train[is.na(CS_Train$Ytd_Amt_Cash),"Ytd_Amt_Cash"]=mean(CS_Train$Ytd_Amt_Cash
,na.rm=T)

CS_Train[is.na(CS_Train$Ytd_Max_Bucket),"Ytd_Max_Bucket"]=mean(CS_Train$Ytd_Max_Bucket
,na.rm=T)

CS_Train[is.na(CS_Train$Account_Status_Bucket),"Account_Status_Bucket"]=mean(CS_Train$Accou
nt_Status_Bucket ,na.rm=T)

CS_Train[is.na(CS_Train$Attrition_Flag),"Attrition_Flag"]=mean(CS_Train$Attrition_Flag ,na.rm=T)

CS_Train[is.na(CS_Train$Blacklist_Flag),"Blacklist_Flag"]=mean(CS_Train$Blacklist_Flag,na.rm=T)

CS_Train[is.na(CS_Train$Writeoff_Flag),"Writeoff_Flag"]=mean(CS_Train$Writeoff_Flag ,na.rm=T)
```

```
CS_Train[is.na(CS_Train$Spending_Limit),"Spending_Limit"]=mean(CS_Train$Spending_Limit
,na.rm=T)

CS_Train[is.na(CS_Train$Avg_Payments),"Avg_Payments"]=mean(CS_Train$Avg_Payments
,na.rm=T)

CS_Train[is.na(CS_Train$Avg_Balances),"Avg_Balances"]=mean(CS_Train$Avg_Balances ,na.rm=T)

CS_Train[is.na(CS_Train$PNL_All_Max_Bucket),"PNL_All_Max_Bucket"]=mean(CS_Train$PNL_All_M
ax_Bucket ,na.rm=T)

CS_Train[is.na(CS_Train$Current_Balance),"Current_Balance"]=mean(CS_Train$Current_Balance
,na.rm=T)

#clean data_test

CS_Test[is.na(CS_Test$Oy_Amt_Cash),"Oy_Amt_Cash"]=mean(CS_Test$Oy_Amt_Cash,na.rm=T)

CS_Test[is.na(CS_Test$All_Max_Bucket_updated),"All_Max_Bucket_updated"]=mean(CS_Test$All_
Max_Bucket_updated,na.rm=T)

CS_Test[is.na(CS_Test$Py_Amt_Cash),"Py_Amt_Cash"]=mean(CS_Test$Py_Amt_Cash,na.rm=T)

CS_Test[is.na(CS_Test$Ytd_Amt_Cash),"Ytd_Amt_Cash"]=mean(CS_Test$Ytd_Amt_Cash ,na.rm=T)

CS_Test[is.na(CS_Test$Ytd_Max_Bucket),"Ytd_Max_Bucket"]=mean(CS_Test$Ytd_Max_Bucket
,na.rm=T)

CS_Test[is.na(CS_Test$Account_Status_Bucket),"Account_Status_Bucket"]=mean(CS_Test$Account_
Status_Bucket ,na.rm=T)

CS_Test[is.na(CS_Test$Attrition_Flag),"Attrition_Flag"]=mean(CS_Test$Attrition_Flag ,na.rm=T)

CS_Test[is.na(CS_Test$Blacklist_Flag),"Blacklist_Flag"]=mean(CS_Test$Blacklist_Flag,na.rm=T)

CS_Test[is.na(CS_Test$Writeoff_Flag),"Writeoff_Flag"]=mean(CS_Test$Writeoff_Flag ,na.rm=T)

CS_Test[is.na(CS_Test$Spending_Limit),"Spending_Limit"]=mean(CS_Test$Spending_Limit ,na.rm=T)

CS_Test[is.na(CS_Test$Avg_Payments),"Avg_Payments"]=mean(CS_Test$Avg_Payments ,na.rm=T)

CS_Test[is.na(CS_Test$Avg_Balances),"Avg_Balances"]=mean(CS_Test$Avg_Balances ,na.rm=T)

CS_Test[is.na(CS_Test$PNL_All_Max_Bucket),"PNL_All_Max_Bucket"]=mean(CS_Test$PNL_All_Max
_Bucket ,na.rm=T)

CS_Test[is.na(CS_Test$Current_Balance),"Current_Balance"]=mean(CS_Test$Current_Balance
,na.rm=T)


log_model_multi1 <- glm(All_Max_Bucket_Flag_updated~
Py_Amt_Cash+Ytd_Amt_Cash+Account_Status_Bucket+Attrition_Flag+


Blacklist_Flag+Current_Balance+Writeoff_Flag+Spending_Limit+Avg_Payments+Avg_Balances,
family="binomial", data=CS_Train)
```

```
summary(log_model_multi1)
log_model_multi1 <- step(object=log_model_multi1,trace=0)
summary(log_model_multi1)
log_model_multi1 <- step(object=log_model_multi1,trace=0)
#The model's prediction accuracy on  test set
prob2<-predict(object=log_model_multi1,newdata=CS_Test,type="response")
summary(prob2)

#ROC
roc_curve4<-roc(CS_Test$All_Max_Bucket_Flag_updated,prob2)
roc_result4<-coords(roc_curve4,"best")
roc_result4
names(roc_curve4)
x4<-1-roc_curve4$specificities
y4<-roc_curve4$sensitivities
p4<-ggplot(data=NULL,aes(x=x4,y=y4))+geom_line(color="red")+geom_abline(intercept=0,slope=1)+
  annotate("text",x=0.5,y=0.5,label=paste("AUC=",round(roc_curve4$auc,2)))+labs(x='1-
specificities',y='sensitivities',title='ROC Curve')
p4
#auc=0.6, not accurate,remove some variable
result = data.frame()
for (i in 1:10){
  pred2<-ifelse(prob2>=i*0.1,1,0)
  pred2<-factor(pred2,levels=c(0,1),order=TRUE)
  f4<-table(CS_Test$All_Max_Bucket_Flag_updated,pred2)

  #sensitivity&specificity

  sensitivity4<-f4[2,2]/(f4[2,2]+f4[2,1])
  specificitiy4<-f4[1,1]/(f4[1,2]+f4[1,1])
  sensi_spe4<-(sensitivity4*specificitiy4)^(1/2)
```

```
  accuracy4=(f4[1,1]+f4[2,2])/(f4[1,1]+f4[2,2]+f[2,1]+f4[1,2])
  PnL4 <- ((f4[1,1]/(f4[1,1] +f4[2,1]))*0.02) +
    ((f4[2,1]/(f[1,1] + f4[2,1]))*-0.45)
  result2 <- as.data.frame(cbind(i*0.1, accuracy4, sensi_spe4, PnL4))
  result <- rbind(result,result2)
  rm(accuracy4, sensi_spe4, PnL4,i,result2)
}
result

pred2<-ifelse(prob2>=0.9,1,0)
pred2<-factor(pred2,levels=c(0,1),order=TRUE)
f4<-table(CS_Test$All_Max_Bucket_Flag_updated,pred2)

#sensitivity&specificity

sensitivity4<-f4[2,2]/(f4[2,2]+f4[2,1])
specificitiy4<-f4[1,1]/(f4[1,2]+f4[1,1])
sensi_spe4<-(sensitivity4*specificitiy4)^(1/2)
accuracy4=(f4[1,1]+f4[2,2])/(f4[1,1]+f4[2,2]+f[2,1]+f4[1,2])
PnL4 <- ((f4[1,1]/(f4[1,1] +f4[2,1]))*0.02) +
  ((f4[2,1]/(f[1,1] + f4[2,1]))*-0.45)

f4
#sensitivity&specificity
sensitivity4<-f4[2,2]/(f4[2,2]+f4[2,1])
sensitivity4
specificitiy4<-f4[1,1]/(f4[1,2]+f4[1,1])
specificitiy4
accuracy2<-(f4[1,1]+f4[2,2])/(f4[1,1]+f4[2,2]+f4[2,1]+f4[1,2])
accuracy2
(sensitivity4*specificitiy4)^(1/2)
```

```
pred_table4=table(unlist(pred2))

pred_table4

profit_rate4=pred_table4[1]/(pred_table4[1]+pred_table4[2])

loss_rate4=1-profit_rate4

profit_lost4=profit_rate4*0.02+loss_rate4*(-0.45)

profit_lost4

profit_lost4_1= (1763/(1763+1066))*0.02+(1066/(1763+1066))*-0.45

profit_lost4_1


#bucket
prediction_____________________________________________________________________________
_
mult.model3 <- lm(All_Max_Bucket_updated ~
Py_Amt_Cash+Ytd_Amt_Cash+Current_Balance+Account_Status_Bucket+Attrition_Flag+

              Blacklist_Flag+Writeoff_Flag+Spending_Limit+Avg_Payments+Avg_Balances,
data=CS_Train)

mult.model3<-step(object=mult.model3,trace=0)

summary(mult.model3)

pre_logistic5<-predict(mult.model3,newdata=CS_Test)

#roc

roc_curve5<-roc(CS_Test$All_Max_Bucket_updated,pre_logistic5)

roc_result5<-coords(roc_curve5,"best")

roc_result5

names(roc_curve5)

x5<-1-roc_curve5$specificities

y5<-roc_curve5$sensitivities


p5<-ggplot(data=NULL,aes(x=x5,y=y5))+geom_line(color="red")+geom_abline(intercept=0,slope=1)+

  annotate("text",x=0.5,y=0.5,label=paste("AUC=",round(roc_curve4$auc,2)))+labs(x='1-
specificities',y='sensitivities',title='ROC Curve')

p5

#auc=0.6, not accurate,remove some variable

pred5<-ifelse(pre_logistic5>0.7581839,1,0)
```

```
table(pred5)

t5<-table(CS_Test$All_Max_Bucket_updated,pre_logistic5)

t5_1<-table(CS_Test$All_Max_Bucket_Flag_updated,pred5)

t5_1

sensitivity5<-t5_1[2,2]/(t5_1[2,2]+t5_1[2,1])

specificitiy5<-t5_1[1,1]/(t5_1[1,2]+t5_1[1,1])

(sensitivity5*specificitiy5)^(1/2)

#Profit and Lost

mult.model3_0 <- multinom(All_Max_Bucket_updated ~
Py_Amt_Cash+Ytd_Amt_Cash+Current_Balance+Account_Status_Bucket+Attrition_Flag+

                    Blacklist_Flag+Writeoff_Flag+Spending_Limit+Avg_Payments+Avg_Balances,
data=CS_Train)

mult.model3_0 <- step(object=mult.model3_0,trace=0)

summary(mult.model3_0)

pre_logistic3_0<-predict(mult.model3_0,newdata=CS_Test)

summary(pre_logistic3_0)

f5<-table(pre_logistic3_0)

(-2)*(f5[2]/sum(f5))+(-3)*(f5[3]/sum(f5))+(-4)*(f5[4]/sum(f5))+(-5)*(f5[5]/sum(f5))+(-
6)*(f5[6]/sum(f5))+(-7)*(f5[7]/sum(f5))


#PNL

mult.model4 <- lm(PNL_All_Max_Bucket  ~Current_Balance+
Py_Amt_Cash+Ytd_Amt_Cash+Account_Status_Bucket+Attrition_Flag+

                Blacklist_Flag+Writeoff_Flag+Spending_Limit+Avg_Payments+Avg_Balances,
data=CS_Train)

summary(mult.model4)

pre_logistic4<-predict(mult.model4,newdata=CS_Test)

summary(pre_logistic4)

#roc

roc_curve6<-roc(CS_Test$All_Max_Bucket_updated,pre_logistic4)

roc_result6<-coords(roc_curve6,"best")

roc_result6

names(roc_curve6)
```

```
x6<-1-roc_curve6$specificities

y6<-roc_curve6$sensitivities


p6<-ggplot(data=NULL,aes(x=x6,y=y6))+geom_line(color="red")+geom_abline(intercept=0,slope=1)+

  annotate("text",x=0.5,y=0.5,label=paste("AUC=",round(roc_curve5$auc,2)))+labs(x='1-
specificities',y='sensitivities',title='ROC Curve')

p6

#auc=0.59

pred6<-ifelse(pre_logistic4<(-0.01449005),1,0)

table(pred6)

t6<-table(CS_Test$All_Max_Bucket_Flag_updated,pred6)

t6

sensitivity6<-t6[1,1]/sum(t6[1,])

specificitiy6<-t6[2,2]/sum(t6[2,])

(sensitivity6*specificitiy6)^(1/2)


#Profit and Lost

mult.model4_0 <- multinom(PNL_All_Max_Bucket  ~Current_Balance+
Py_Amt_Cash+Ytd_Amt_Cash+Account_Status_Bucket+Attrition_Flag+

                    Blacklist_Flag+Writeoff_Flag+Spending_Limit+Avg_Payments+Avg_Balances,
data=CS_Train)

mult.model4_0 <- step(object=mult.model4_0,trace=0)

summary(mult.model4_0)

pre_logistic4_0<-predict(mult.model4_0,newdata=CS_Test)

summary(pre_logistic4_0)

0.03*(f6[7]/sum(f6))+(-0.15)*(f6[5]/sum(f6))+(-0.25)*(f6[4]/sum(f6))+(-0.35)*(f6[3]/sum(f6))+(-
0.45)*(f6[2]/sum(f6))+(-0.65)*(f6[1]/sum(f6))
```

Random Forest Classification

```
# Importing the dataset and splitting them into training and test dataset.

CS_Data <- readRDS("F:/GP/CS_Data.RDS")

CS_Data1 <- readRDS("F:/GP/CS_Data1.RDS")

CS_Train <- readRDS("F:/GP/CS_Train.RDS")
```

```
CS_Test <- readRDS("F:/GP/CS_Test.RDS")

################################################################################
# Encoding the target feature as factor

CS_Train$Sex_M_Flag <- factor(CS_Train$Sex_M_Flag, levels = c(0, 1))
CS_Train$Blacklist_Flag <- factor(CS_Train$Blacklist_Flag, levels = c(0, 1))
CS_Train$Out_Bank_Flag <- factor(CS_Train$Out_Bank_Flag, levels = c(0, 1))
CS_Train$All_Max_Bucket_Flag_updated <- factor(CS_Train$All_Max_Bucket_Flag_updated)
CS_Train$PNL_All_Max_Bucket <- factor(CS_Train$PNL_All_Max_Bucket)
CS_Train$Attrition_Flag <- factor(CS_Train$Attrition_Flag)
CS_Train$Writeoff_Flag <- factor(CS_Train$Writeoff_Flag)
CS_Train$Bucket <- factor(CS_Train$Bucket)
CS_Train$Branch_Code_Rating <- factor(CS_Train$Branch_Code_Rating)
CS_Train$All_Max_Bucket_updated <- factor(CS_Train$All_Max_Bucket_updated)

summary((CS_Train))

CS_Test$Sex_M_Flag <- factor(CS_Test$Sex_M_Flag, levels = c(0, 1))
CS_Test$Blacklist_Flag <- factor(CS_Test$Blacklist_Flag, levels = c(0, 1))
CS_Test$Out_Bank_Flag <- factor(CS_Test$Out_Bank_Flag, levels = c(0, 1))
CS_Test$All_Max_Bucket_Flag_updated <- factor(CS_Test$All_Max_Bucket_Flag_updated)
CS_Test$PNL_All_Max_Bucket <- factor(CS_Test$PNL_All_Max_Bucket)
CS_Test$Attrition_Flag <- factor(CS_Test$Attrition_Flag)
CS_Test$Writeoff_Flag <- factor(CS_Test$Writeoff_Flag)
CS_Test$Bucket <- factor(CS_Test$Bucket)
CS_Test$Branch_Code_Rating <- factor(CS_Test$Branch_Code_Rating)
################################################################################
#replacing missing values (NA) wiith Median

#CS_Test
```

```
CS_Test$Age_Issued_Updated[is.na(CS_Test$Age_Issued_Updated)]<-
mean(CS_Test$Age_Issued_Updated,na.rm=TRUE)

CS_Test$Current_Balance[is.na(CS_Test$Current_Balance)]<-
mean(CS_Test$Current_Balance,na.rm=TRUE)

CS_Test$Ytd_Amt_Cash[is.na(CS_Test$Ytd_Amt_Cash)]<-
mean(CS_Test$Ytd_Amt_Cash,na.rm=TRUE)

CS_Test$Avg_Payments[is.na(CS_Test$Avg_Payments)]<-
mean(CS_Test$Avg_Payments,na.rm=TRUE)

CS_Test$Avg_Balances[is.na(CS_Test$Avg_Balances)]<-mean(CS_Test$Avg_Balances,na.rm=TRUE)

CS_Test$Oy_Amt_Cash[is.na(CS_Test$Oy_Amt_Cash)]<-mean(CS_Test$Oy_Amt_Cash,na.rm=TRUE)

CS_Test$Py_Amt_Cash[is.na(CS_Test$Py_Amt_Cash)]<-mean(CS_Test$Py_Amt_Cash,na.rm=TRUE)


#CS_Train


CS_Train$Age_Issued_Updated[is.na(CS_Train$Age_Issued_Updated)]<-
mean(CS_Train$Age_Issued_Updated,na.rm=TRUE)

CS_Train$Current_Balance[is.na(CS_Train$Current_Balance)]<-
mean(CS_Train$Current_Balance,na.rm=TRUE)

CS_Train$Ytd_Amt_Cash[is.na(CS_Train$Ytd_Amt_Cash)]<-
mean(CS_Train$Ytd_Amt_Cash,na.rm=TRUE)

CS_Train$Avg_Payments[is.na(CS_Train$Avg_Payments)]<-
mean(CS_Train$Avg_Payments,na.rm=TRUE)

CS_Train$Avg_Balances[is.na(CS_Train$Avg_Balances)]<-mean(CS_Train$Avg_Balances,na.rm=TRUE)

CS_Train$Oy_Amt_Cash[is.na(CS_Train$Oy_Amt_Cash)]<-
mean(CS_Train$Oy_Amt_Cash,na.rm=TRUE)

CS_Train$Py_Amt_Cash[is.na(CS_Train$Py_Amt_Cash)]<-
mean(CS_Train$Py_Amt_Cash,na.rm=TRUE)


#############################################################################

# Feature Scaling


summary(CS_Train)
#CS_Train$Age_Issued_Updated <- scale(CS_Train$Age_Issued_Updated)
#CS_Train$Branch_Code_Rating <- scale(CS_Train$Branch_Code_Rating)
```

```
#CS_Test$Age_Issued_Updated <- scale(CS_Test$Age_Issued_Updated)
#CS_Test$Branch_Code_Rating <- scale(CS_Test$Branch_Code_Rating)

summary(CS_Train)
summary(CS_Test)

#CS_Train <- na.omit(CS_Train)
##############################################################################################

# Fitting Random Forest Classification to the Training set: Application (Bucket Prediction
Classification)


# install.packages('randomForest')

library(randomForest)

#set.seed(123)

names(CS_Train)

Application <- randomForest(x = CS_Train[c('Sex_M_Flag', 'Blacklist_Flag', 'Out_Bank_Flag',
'Age_Issued_Updated', 'Branch_Code_Rating')],

                            y = CS_Train$All_Max_Bucket_Flag_updated,

                            mtry = 1, ntree =100)


# Predicting the Test set results: Application

y_pred_app = predict(Application, newdata = CS_Test[c('Sex_M_Flag', 'Blacklist_Flag',
'Out_Bank_Flag', 'Age_Issued_Updated', 'Branch_Code_Rating')])

y_pred_app


# Making the Confusion Matrix: Application

cm_app = table(CS_Test[, 22], y_pred_app)
cm_app



####################


Behavioral <- randomForest(x = CS_Train[c('Attrition_Flag', 'Blacklist_Flag',
'Writeoff_Flag','Out_Bank_Flag','Sex_M_Flag','Age_Issued_Updated', 'Branch_Code_Rating')],

                           y = CS_Train$All_Max_Bucket_Flag_updated,
```

```
                    mtry = 1, ntree = 100)


# Predicting the Test set results: Behavioral

y_pred_Beh = predict(Behavioral, newdata = CS_Test[c('Attrition_Flag', 'Blacklist_Flag',
'Writeoff_Flag','Out_Bank_Flag','Sex_M_Flag','Age_Issued_Updated', 'Branch_Code_Rating')])

y_pred_Beh


# Making the Confusion Matrix: Behavioral

cm_Beh = table(CS_Test[, 22], y_pred_Beh)
cm_Beh

#############################################################################

# Fitting Random Forest Classification to the Training set: Application (Profit and Loss Classification)


# install.packages('randomForest')

library(randomForest)

#set.seed(123)


Application_BP <- randomForest(x = CS_Train[c('Sex_M_Flag','Blacklist_Flag', 'Out_Bank_Flag',
'Branch_Code_Rating')],

                    y = CS_Train$All_Max_Bucket_updated, mtry = 1, ntree =100)


# Predicting the Test set results: Application

y_pred_app_bp = predict(Application_BP, newdata = CS_Test[c('Sex_M_Flag','Blacklist_Flag',
'Out_Bank_Flag', 'Branch_Code_Rating')])

y_pred_app_bp


# Making the Confusion Matrix: Application

cm_app_bp = table(CS_Test[, 21], y_pred_app_bp)
cm_app_bp

####################
```

```
Behavioral_BP <- randomForest(x = CS_Train[c('Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag',
'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'Account_Status_Bucket','Current_Balance',
'Branch_Code_Rating','Out_Bank_Flag', 'Age_Issued_Updated','Sex_M_Flag')],

                    y = CS_Train$All_Max_Bucket_updated, mtry = 1,

                   ntree = 100)


# Predicting the Test set results: Behavioral

y_pred_Beh_bp = predict(Behavioral_BP, newdata = CS_Test[c('Attrition_Flag', 'Blacklist_Flag',
'Writeoff_Flag', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances',
'Account_Status_Bucket','Current_Balance', 'Branch_Code_Rating','Out_Bank_Flag',
'Age_Issued_Updated','Sex_M_Flag')])

y_pred_Beh_bp


# Making the Confusion Matrix: Behavioral

cm_Beh_bp = table(CS_Test[, 21], y_pred_Beh_bp)
cm_Beh_bp

#############################################################################################

#Profit/Loss prediction : Application


Application_PL <- randomForest(x = CS_Train[c('Sex_M_Flag','Blacklist_Flag', 'Out_Bank_Flag',
'Branch_Code_Rating')],

                    y = CS_Train$PNL_All_Max_Bucket, mtry = 1,

                    ntree =100)


# Predicting the Test set results: Application

y_pred_app_pl = predict(Application_PL, newdata = CS_Test[c('Sex_M_Flag','Blacklist_Flag',
'Out_Bank_Flag', 'Branch_Code_Rating')])

y_pred_app_pl


# Making the Confusion Matrix: Application

cm_app_pl = table(CS_Test[, 23], y_pred_app_pl)
cm_app_pl


####################
```

```
Behavioral_PL <- randomForest(x = CS_Train[c('Attrition_Flag', 'Blacklist_Flag', 'Writeoff_Flag',
'Spending_Limit', 'Avg_Payments', 'Avg_Balances', 'Account_Status_Bucket','Current_Balance',
'Branch_Code_Rating','Out_Bank_Flag', 'Age_Issued_Updated','Sex_M_Flag')],
                    y = CS_Train$PNL_All_Max_Bucket, mtry = 1,
                    ntree = 100)



# Predicting the Test set results: Behavioral
y_pred_Beh_pl = predict(Behavioral_PL, newdata = CS_Test[c('Attrition_Flag', 'Blacklist_Flag',
'Writeoff_Flag', 'Spending_Limit', 'Avg_Payments', 'Avg_Balances',
'Account_Status_Bucket','Current_Balance', 'Branch_Code_Rating','Out_Bank_Flag',
'Age_Issued_Updated','Sex_M_Flag')])
y_pred_Beh_pl



# Making the Confusion Matrix: Behavioral
cm_Beh_pl = table(CS_Test[, 23], y_pred_Beh_pl)
cm_Beh_pl
```

```
####################Decision Tree_CART ##############################
install.packages("pROC")
install.packages("ggplot2")
install.packages("dplyr")
install.packages("rpart")
install.packages("rpart.plot")
install.packages("partykit")
install.packages("rpart.predict")
install.packages("caret")
install.packages("e1071y")
install.packages("randomForest")
install.packages("ROCR")
install.packages("tree")
```

```
install.packages("ISLR")
install.packages("corrplot")
install.packages("Dforest")
install.packages("cutpointr")

rm(list=ls())
library(dplyr)
library(pROC)
library(ggplot2)
library(readr)
library(rpart)
library(rpart.plot)
library(partykit)
library(caret)
library(e1071)
library(randomForest)
library(ROCR)
library(tree)
library(ISLR)
library(rpart.plot)
library(corrplot)
library(Dforest)
library(cutpointr)
library(Hmisc)
library(plyr)
library(lattice)
library(nnet)
library(reshape2)

### import data ###
CS_Data1 <- readRDS("Desktop/CS_Data1.RDS")
CS_Data <- readRDS("Desktop/CS_Data.RDS")
CS_Train <- readRDS("Desktop/CS_Train.RDS")
```

```
CS_Test <- readRDS("Desktop/CS_Test.RDS")


View(CS_Data1)

View(CS_Train)

View(CS_Test)

View(CS_Data)


### NA --> Mean replace ###

### NA --> Mean replace_Train ###

CS_Train[is.na(CS_Train$Spending_Limit),"Spending_Limit"] = mean(CS_Train$Spending_Limit, na.rm
 = TRUE)

CS_Train[is.na(CS_Train$Oy_Amt_Cash),"Oy_Amt_Cash"] = mean(CS_Train$Oy_Amt_Cash, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Current_Balance),"Current_Balance"] = mean(CS_Train$Current_Balance,
 na.rm = TRUE)

CS_Train[is.na(CS_Train$Avg_Payments),"Avg_Payments"] = mean(CS_Train$Avg_Payments, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Avg_Balances),"Avg_Balances"] = mean(CS_Train$Avg_Balances, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Ytd_Amt_Cash),"Ytd_Amt_Cash"] = mean(CS_Train$Ytd_Amt_Cash, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Py_Amt_Cash),"Py_Amt_Cash"] = mean(CS_Train$Py_Amt_Cash, na.rm =
 TRUE)


CS_Train[is.na(CS_Train$Attrition_Flag),"Attrition_Flag"] = mean(CS_Train$Attrition_Flag, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Writeoff_Flag),"Writeoff_Flag"] = mean(CS_Train$Writeoff_Flag, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Blacklist_Flag),"Blacklist_Flag"] = mean(CS_Train$Blacklist_Flag, na.rm =
 TRUE)

CS_Train[is.na(CS_Train$Account_Status_Bucket),"Account_Status_Bucket"] =
 mean(CS_Train$Account_Status_Bucket, na.rm = TRUE)


# NA --> Mean replace_Test

CS_Test[is.na(CS_Test$Spending_Limit),"Spending_Limit"] = mean(CS_Test$Spending_Limit, na.rm =
 T)

CS_Test[is.na(CS_Test$Oy_Amt_Cash),"Oy_Amt_Cash"] = mean(CS_Test$Oy_Amt_Cash,na.rm = T)
```

```
CS_Test[is.na(CS_Test$Current_Balance),"Current_Balance"] = mean(CS_Test$Current_Balance,na.rm =
 T)

CS_Test[is.na(CS_Test$Avg_Payments),"Avg_Payments"] = mean(CS_Test$Avg_Payments,na.rm = T)

CS_Test[is.na(CS_Test$Avg_Balances),"Avg_Balances"] = mean(CS_Test$Avg_Balances, na.rm = T)

CS_Test[is.na(CS_Test$Ytd_Amt_Cash),"Ytd_Amt_Cash"] = mean(CS_Test$Ytd_Amt_Cash, na.rm = T)

CS_Test[is.na(CS_Test$Py_Amt_Cash),"Py_Amt_Cash"] = mean(CS_Test$Py_Amt_Cash, na.rm = T)


CS_Test[is.na(CS_Test$Writeoff_Flag),"Writeoff_Flag"] = mean(CS_Test$Writeoff_Flag, na.rm =
 TRUE)

CS_Test[is.na(CS_Test$Attrition_Flag),"Attrition_Flag"] = mean(CS_Test$Attrition_Flag, na.rm =
 TRUE)

CS_Test[is.na(CS_Test$Blacklist_Flag),"Blacklist_Flag"] = mean(CS_Test$Blacklist_Flag, na.rm =
 TRUE)

CS_Test[is.na(CS_Test$Account_Status_Bucket),"Account_Status_Bucket"] =
 mean(CS_Test$Account_Status_Bucket, na.rm = TRUE)


#################################################################

###### Application

############ Zero-one Classification_All_Max_Bucket_Flag_updated

#################################################################

cart.model_app_zc <- rpart(formula = All_Max_Bucket_Flag_updated ~ Age_Issued_Updated +
 Sex_M_Flag + Blacklist_Flag + Out_Bank_Flag + Branch_Code_Rating, data = CS_Train)

summary(cart.model_app_zc)

cart.model_app_zc

rpart.plot(cart.model_app_zc)

plot(cart.model_app_zc)

text(cart.model_app_zc)

attributes(summary(cart.model_app_zc))


attributes(CS_Train$Age_Issued_Updated)

class(CS_Train$Branch_Code_Rating)


rparty.tree_app_zc <- as.party(cart.model_app_zc)

rparty.tree_app_zc

plot(rparty.tree_app_zc)
```

```
##############################

############################## predict_Train

pred_app_zc_train <- predict(object = cart.model_app_zc, newdata = CS_Train, type = "vector")

table(real = CS_Train$All_Max_Bucket_Flag_updated, predict=pred_app_zc_train)

summary(pred_app_zc_train)


confus.matrix_app_zc_train <- table(real= CS_Train$All_Max_Bucket_Flag_updated,
 predict=pred_app_zc_train)

confus.matrix_app_zc_train

##############################

############################## predict_Test

pred_card_app_zc_test <- predict(object = cart.model_app_zc, newdata = CS_Test, type = "vector")

table(real = CS_Test$All_Max_Bucket_Flag_updated, predict=pred_card_app_zc_test)

summary(pred_card_app_zc_test)


confus.matrix_app_zc_test <- table(real= CS_Test$All_Max_Bucket_Flag_updated,
 predict=pred_card_app_zc_test > 0.2)

confus.matrix_app_zc_test

sum(diag(confus.matrix_app_zc_test))/sum(confus.matrix_app_zc_test)


### accuracy, sensi_spe, PnL

result = data.frame ()


for (i in 1:10){

 pred_app_zc_test_ak1<-ifelse(pred_card_app_zc_test>=i*0.1,1,0)

 pred_app_zc_test_ak2<-factor(pred_app_zc_test_ak1,levels=c(0,1),order=TRUE)

 f<-table(CS_Test$All_Max_Bucket_Flag_updated,pred_app_zc_test_ak2)


 #sensitivity&specificity


 sensitivity<-f[2,2]/(f[2,2]+f[2,1])

 specificitiy<-f[1,1]/(f[1,2]+f[1,1])

 sensi_spe<-(sensitivity*specificitiy)^(1/2)

 accuracy=(f[1,1]+f[2,2])/(f[1,1]+f[2,2]+f[2,1]+f[1,2])
```

```
  PnL <- ((f[1,1]/(f[1,1] +f[2,1]))*0.02) +
    ((f[2,1]/(f[1,1] + f[2,1]))*-0.45)
  result1 <- as.data.frame(cbind(i*0.1, accuracy, sensi_spe, PnL))
  result <- rbind(result,result1)
  rm(accuracy, sensi_spe, PnL,i,result1)
}
result

#### Classification tree
printcp(cart.model_app_zc)
plotcp(cart.model_app_zc)
cart.model_app_zc$cptable
prunetree_cart.model_app_zc_test <- prune(cart.model_app_zc, cp =
 cart.model_app_zc$cptable[which.min(cart.model_app_zc$cptable[,"xerror"]),"CP"])
prunetree_pred_app_zc_test <- predict(prunetree_cart.model_app_zc_test, newdata=CS_Test,
 type="vector")
table(real=CS_Test$All_Max_Bucket_Flag_updated, predict=prunetree_pred_app_zc_test)

prunetree_confus.matrix_app_zc_test <- table(real=CS_Test$All_Max_Bucket_Flag_updated,
 predict=prunetree_pred_app_zc_test)
sum(diag(prunetree_confus.matrix_app_zc_test))/sum(prunetree_confus.matrix_app_zc_test)

# SexMFlag
SexMFlag <-ifelse(CS_Test$Sex_M_Flag == 0 , "F", "M" )
SexMFlag
CS_Test <- cbind(CS_Test, SexMFlag)
summary(SexMFlag)
attributes(SexMFlag)
SexMFlag <- factor(SexMFlag)
attributes(SexMFlag)

# MaxBucketFlag2
MaxBucketFlag2 <-ifelse(CS_Test$All_Max_Bucket_Flag_updated == 0 ,"first", "second" )
MaxBucketFlag2
```

```
CS_Test <- cbind(CS_Test, MaxBucketFlag2)
summary(MaxBucketFlag2)
attributes(MaxBucketFlag2)
MaxBucketFlag2 <- factor(MaxBucketFlag2)
attributes(MaxBucketFlag2)

# BlacklistFlag
BlacklistFlag <-ifelse(CS_Test$Blacklist_Flag == 0 ,"A", "B" )
BlacklistFlag
CS_Test <- cbind(CS_Test, BlacklistFlag)
summary(BlacklistFlag)
attributes(BlacklistFlag)
BlacklistFlag <- factor(BlacklistFlag)
attributes(BlacklistFlag)

# OutBankFlag
OutBankFlag <-ifelse(CS_Test$Out_Bank_Flag == 0 ,"OUT", "IN" )
OutBankFlag
CS_Test <- cbind(CS_Test,OutBankFlag)
summary(OutBankFlag)
attributes(OutBankFlag)
OutBankFlag <- factor(OutBankFlag)
attributes(OutBankFlag)

### K-fold Cross-Validation_overfitting
traincontrolappzctest <- trainControl(method="cv", number=4, classProbs = FALSE)
train_control.model_app_zc_test <- train(MaxBucketFlag2  ~ BlacklistFlag + SexMFlag +
 Age_Issued_Updated + Branch_Code_Rating + OutBankFlag, data = CS_Test, method = "rpart",parms =
 list(split = "gini"), trControl = traincontrolappzctest, na.action = na.rpart)
train_control.model_app_zc_test

############################################################
###### Bucket Prediction_All_Max_Bucket_updated
############################################################
```

```
CM_App_BP <- rpart(formula = All_Max_Bucket_updated ~ Age_Issued_Updated + Sex_M_Flag +
 Blacklist_Flag + Out_Bank_Flag + Branch_Code_Rating, data = CS_Train)

summary(CM_App_BP)

CM_App_BP

rpart.plot(CM_App_BP)

plot(CM_App_BP)

text(CM_App_BP)


rparty.tree_App_BP <- as.party(CM_App_BP)

rparty.tree_App_BP

plot(rparty.tree_App_BP)

##############################

############################## predict_Train

pred_card_app_bp_train <- predict(object = CM_App_BP, newdata = CS_Train, type = "vector")

table(real = CS_Train$All_Max_Bucket_updated, predict=pred_card_app_bp_train)

summary(pred_card_app_bp_train)


confus.matrix_app_bp_train <- table(real= CS_Train$All_Max_Bucket_updated,
 predict=pred_card_app_bp_train)

confus.matrix_app_bp_train


##############################

############################## predict_Test

pred_card_app_test_bp <- predict(object = CM_App_BP, newdata = CS_Test, type = "vector")

table(real = CS_Test$All_Max_Bucket_updated, predict=pred_card_app_test_bp)

summary(pred_card_app_test_bp)


confus.matrix_app_bp_test <- table(real= CS_Test$All_Max_Bucket_updated,
 predict=pred_card_app_test_bp)

confus.matrix_app_bp_test


one_cm_app_bptt_L <-
 confus.matrix_app_bp_test[2,1]+confus.matrix_app_bp_test[3,1]+confus.matrix_app_bp_test[4,1]+confu
 s.matrix_app_bp_test[5,1]+confus.matrix_app_bp_test[6,1]+confus.matrix_app_bp_test[7,1]
```

```
one_cm_app_bptt_R<-
 confus.matrix_app_bp_test[2,2]+confus.matrix_app_bp_test[3,2]+confus.matrix_app_bp_test[4,2]+confu
 s.matrix_app_bp_test[5,2]+confus.matrix_app_bp_test[6,2]+confus.matrix_app_bp_test[7,2]

one_cm_app_bptt_L

one_cm_app_bptt_R


#### Accuracy ####

(confus.matrix_app_bp_test[1,1] + one_cm_app_bptt_R)/sum(confus.matrix_app_bp_test)

sen_cm_app_bp_test <- confus.matrix_app_bp_test[1,1]/(confus.matrix_app_bp_test[1,1] +
 one_cm_app_bptt_L)

sen_cm_app_bp_test

spe_cm_app_bp_test <- one_cm_app_bptt_R /(confus.matrix_app_bp_test[1,2] + one_cm_app_bptt_R)

spe_cm_app_bp_test

sqrt(sen_cm_app_bp_test*spe_cm_app_bp_test)


## Average Profit/Loss ##

zero_app_bp_test <-
 (confus.matrix_app_bp_test[1,1]+confus.matrix_app_bp_test[1,2])/sum(confus.matrix_app_bp_test)

N2_app_bp_test <-
 (confus.matrix_app_bp_test[2,1]+confus.matrix_app_bp_test[2,2])/sum(confus.matrix_app_bp_test)

N3_app_bp_test <-
 (confus.matrix_app_bp_test[3,1]+confus.matrix_app_bp_test[3,2])/sum(confus.matrix_app_bp_test)

N4_app_bp_test <-
 (confus.matrix_app_bp_test[4,1]+confus.matrix_app_bp_test[4,2])/sum(confus.matrix_app_bp_test)

N5_app_bp_test <-
 (confus.matrix_app_bp_test[5,1]+confus.matrix_app_bp_test[5,2])/sum(confus.matrix_app_bp_test)

N6_app_bp_test <-
 (confus.matrix_app_bp_test[6,1]+confus.matrix_app_bp_test[6,2])/sum(confus.matrix_app_bp_test)

N7_app_bp_test <-
 (confus.matrix_app_bp_test[7,1]+confus.matrix_app_bp_test[7,2])/sum(confus.matrix_app_bp_test)


0*zero_app_bp_test-2*N2_app_bp_test -3*N3_app_bp_test -4*N4_app_bp_test -5*N5_app_bp_test -
 6*N6_app_bp_test -7*N7_app_bp_test


#### Classification tree

printcp(CM_App_BP)

plotcp(CM_App_BP)
```

```
prunetree_cart.model_app_bp <- prune(CM_App_BP, cp =
 CM_App_BP$cptable[which.min(CM_App_BP$cptable[,"xerror"]),"CP"])

prunetree_pred_app_bp <- predict(prunetree_cart.model_app_bp, newdata=CS_Test, type="vector")

table(real=CS_Test$All_Max_Bucket_updated, predict=prunetree_pred_app_bp)


prunetree_confus.matrix_app_bp <- table(real=CS_Test$All_Max_Bucket_updated,
 predict=prunetree_pred_app_bp)

sum(diag(prunetree_confus.matrix_app_bp))/sum(prunetree_confus.matrix_app_bp)


### K-fold Cross-Validation_overfitting

train_control_app_bp_test <- trainControl(method="cv", number=10, classProbs = FALSE)

train_control.model_app_bp_test <- train(MaxBucketFlag2  ~ BlacklistFlag + SexMFlag +
 Age_Issued_Updated + Branch_Code_Rating + OutBankFlag, data = CS_Test, method = "rpart",parms =
 list(split = "gini"), trControl = train_control_app_bp_test, na.action = na.rpart)

train_control.model_app_bp_test


############################################################

###### Profit/Loss Prediction_PNL_All_Max_Bucket

############################################################

CM_App_PL <- rpart(formula = PNL_All_Max_Bucket ~ Age_Issued_Updated + Sex_M_Flag +
 Blacklist_Flag + Out_Bank_Flag + Branch_Code_Rating, data = CS_Train)

summary(CM_App_PL)

CM_App_PL

rpart.plot(CM_App_PL)

plot(CM_App_PL)

text(CM_App_PL)


rparty.tree_App_PL <- as.party(CM_App_PL)

rparty.tree_App_PL

plot(rparty.tree_App_PL)


##############################

############################## predict_Train

pred_card_app_pl_train <- predict(object = CM_App_PL, newdata = CS_Train, type = "vector")

table(real = CS_Train$PNL_All_Max_Bucket, predict=pred_card_app_pl_train)
```

```
summary(pred_card_app_pl_train)


confus.matrix_app_pl_train <- table(real= CS_Train$PNL_All_Max_Bucket,
 predict=pred_card_app_pl_train)

confus.matrix_app_pl_train


### K-fold Cross-Validation_overfitting

# overfitting

train_control_app_pl_train <- trainControl(method="cv", number=10, classProbs = FALSE)

train_control.model_app_pl_train <- train(MaxBucketFlag2  ~ BlacklistFlag + SexMFlag +
 Age_Issued_Updated + Branch_Code_Rating + OutBankFlag, data = CS_Train, method = "rpart",parms
 = list(split = "gini"), trControl = train_control_app_pl_train, na.action = na.rpart)

train_control.model_app_pl_train


##############################

############################## predict_Test

pred_card_app_pl_test <- predict(object = CM_App_PL, newdata = CS_Test, type = "vector")

table(real = CS_Test$PNL_All_Max_Bucket, predict=pred_card_app_pl_test)

summary(pred_card_app_pl_test)


confus.matrix_app_pl_test <- table(real= CS_Test$PNL_All_Max_Bucket,
 predict=pred_card_app_pl_test)

confus.matrix_app_pl_test


one_cm_app_pltt_L <-
 confus.matrix_app_pl_test[1,1]+confus.matrix_app_pl_test[2,1]+confus.matrix_app_pl_test[3,1]+confus.
 matrix_app_pl_test[4,1]+confus.matrix_app_pl_test[5,1]

one_cm_app_pltt_R <-
 confus.matrix_app_pl_test[1,2]+confus.matrix_app_pl_test[2,2]+confus.matrix_app_pl_test[3,2]+confus.
 matrix_app_pl_test[4,2]+confus.matrix_app_pl_test[5,2]

one_cm_app_pltt_L

one_cm_app_pltt_R


zero_cm_app_pltt_L <- confus.matrix_app_pl_test[6,1]+confus.matrix_app_pl_test[7,1]

zero_cm_app_pltt_R <- confus.matrix_app_pl_test[6,2]+confus.matrix_app_pl_test[7,2]

zero_cm_app_pltt_L
```

```
zero_cm_app_pltt_R


#### Accuracy ####

(zero_cm_app_pltt_L + one_cm_app_pltt_R)/sum(confus.matrix_app_pl_test)


sen_cm_app_pl_test <- zero_cm_app_pltt_L/(zero_cm_app_pltt_L + one_cm_app_pltt_L)

sen_cm_app_pl_test

spe_cm_app_pl_test <- one_cm_app_pltt_R /(zero_cm_app_pltt_R + one_cm_app_pltt_R)

spe_cm_app_pl_test

sqrt(sen_cm_app_pl_test*spe_cm_app_pl_test)


## Average Profit/Loss ##

N65_app_pl_test <-
 (confus.matrix_app_pl_test[1,1]+confus.matrix_app_pl_test[1,2])/sum(confus.matrix_app_pl_test)

N45_app_pl_test <-
 (confus.matrix_app_pl_test[2,1]+confus.matrix_app_pl_test[2,2])/sum(confus.matrix_app_pl_test)

N35_app_pl_test <-
 (confus.matrix_app_pl_test[3,1]+confus.matrix_app_pl_test[3,2])/sum(confus.matrix_app_pl_test)

N25_app_pl_test <-
 (confus.matrix_app_pl_test[4,1]+confus.matrix_app_pl_test[4,2])/sum(confus.matrix_app_pl_test)

N15_app_pl_test <-
 (confus.matrix_app_pl_test[5,1]+confus.matrix_app_pl_test[5,2])/sum(confus.matrix_app_pl_test)

P0_app_pl_test <-
 (confus.matrix_app_pl_test[6,1]+confus.matrix_app_pl_test[6,2])/sum(confus.matrix_app_pl_test)

P3_app_pl_test <-
 (confus.matrix_app_pl_test[7,1]+confus.matrix_app_pl_test[7,2])/sum(confus.matrix_app_pl_test)


0.03*P3_app_pl_test + 0*P0_app_pl_test -0.15*N15_app_pl_test - 0.25*N25_app_pl_test -
 0.35*N35_app_pl_test -0.45*N45_app_pl_test -0.65*N65_app_pl_test


#### Classification tree

printcp(CM_App_PL)

plotcp(CM_App_PL)

prunetree_cart.model_app_pl_test <- prune(CM_App_PL, cp =
 CM_App_PL$cptable[which.min(CM_App_PL$cptable[,"xerror"]),"CP"])

prunetree_pred_app_pl_test <- predict(prunetree_cart.model_app_pl_test, newdata=CS_Test,
 type="vector")
```

```
table(real=CS_Test$PNL_All_Max_Bucket, predict=prunetree_pred_app_pl_test)


prunetree_confus.matrix_app_pl_test <- table(real=CS_Test$PNL_All_Max_Bucket,
 predict=prunetree_pred_app_pl_test)

sum(diag(prunetree_confus.matrix_app_pl_test))/sum(prunetree_confus.matrix_app_pl_test)


### K-fold Cross-Validation_overfitting

# overfitting

train_control_app_pl_test <- trainControl(method="cv", number=10, classProbs = TRUE)

train_control.model_app_pl_test <- train(MaxBucketFlag2  ~ BlacklistFlag + SexMFlag +
 Age_Issued_Updated + Branch_Code_Rating + OutBankFlag, data = CS_Test, method = "rpart",parms =
 list(split = "gini"), trControl = train_control_app_pl_test, na.action = na.rpart)

train_control.model_app_pl_test


Pred_app_pl_test <- predict(train_control.model_app_pl_test ,newdata = CS_Test, type = "prob")


#################################################################

###### Behavioral

#################################################################

#corrlation

cor(CS_Train$Attrition_Flag, CS_Train$Writeoff_Flag)

cor(CS_Train$Attrition_Flag, CS_Train$Blacklist_Flag)

cor(CS_Train$Attrition_Flag, CS_Train$Account_Status_Bucket)

cor(CS_Train$Writeoff_Flag, CS_Train$Blacklist_Flag)

cor(CS_Train$Writeoff_Flag, CS_Train$Account_Status_Bucket)

cor(CS_Train$Blacklist_Flag, CS_Train$Account_Status_Bucket)


source("http://www.sthda.com/upload/rquery_cormat.r")

new.cor <- cbind(CS_Train$Attrition_Flag, CS_Train$Writeoff_Flag, CS_Train$Blacklist_Flag,
 CS_Train$Spending_Limit, CS_Train$Avg_Payments, CS_Train$Avg_Balances,
 CS_Train$Py_Amt_Cash, CS_Train$Ytd_Amt_Cash, CS_Train$Account_Status_Bucket,
 CS_Train$Oy_Amt_Cash, CS_Train$Current_Balance)

require("corrplot")

new.cor.a <- cor(new.cor)

new.cor.a
```

```
############

x.model <- glm(CS_Train$All_Max_Bucket_Flag_updated ~ CS_Train$Spending_Limit +
 CS_Train$Writeoff_Flag + CS_Train$Blacklist_Flag + CS_Train$Attrition_Flag +
 CS_Train$Avg_Payments + CS_Train$Avg_Balances + CS_Train$Py_Amt_Cash +
 CS_Train$Ytd_Amt_Cash + CS_Train$Oy_Amt_Cash + CS_Train$Current_Balance +
 CS_Train$Account_Status_Bucket, family = binomial)

summary(x.model)

attributes(CS_Train$Attrition_Flag)

class(CS_Train$Attrition_Flag)


#############################################################

###### Zero-one Classification

#############################################################

cart.model_behav_zc <- rpart(All_Max_Bucket_Flag_updated ~ Spending_Limit + Avg_Balances +
 Ytd_Amt_Cash + Current_Balance + BlacklistFlag, data = CS_Train)

summary(cart.model_behav_zc)

cart.model_behav_zc

rpart.plot(cart.model_behav_zc)

plot(cart.model_behav_zc)

text(cart.model_behav_zc)


rparty.tree_behav_zc <- as.party(cart.model_behav_zc)

rparty.tree_behav_zc

plot(rparty.tree_behav_zc)

##############################

############################## predict_Train

pred_behav_zc_train <- predict(object = cart.model_behav_zc, newdata = CS_Train, type = "vector")

tablepred_behav_zc_train<-table(real = CS_Train$All_Max_Bucket_Flag_updated,
 predict=pred_behav_zc_train > 0.102)

tablepred_behav_zc_train

summary(pred_behav_zc_train)


confus.matrix_behav_zc_train <- table(real= CS_Train$All_Max_Bucket_Flag_updated,
 predict=pred_behav_zc_train)

confus.matrix_behav_zc_train
```

```
sum(diag(confus.matrix_behav_zc_train))/sum(confus.matrix_behav_zc_train)


##############################

############################## predict_Test

pred_card_behav_zc_test <- predict(object = cart.model_behav_zc, newdata = CS_Test, type = "vector")

tablepred_behav_zc_test <- table(real = CS_Test$All_Max_Bucket_Flag_updated,
 predict=pred_card_behav_zc_test)

tablepred_behav_zc_test

summary(pred_card_behav_zc_test)


confus.matrix_behav_zc_test <- table(real= CS_Test$All_Max_Bucket_Flag_updated,
 predict=pred_card_behav_zc_test)

confus.matrix_behav_zc_test


#### accuracy, sensi_spe, PnL

result = data.frame ()


for (i in 1:100){

 pred<-ifelse(pred_card_behav_zc_test>=i*0.01,1,0)

 pred1<-factor(pred,levels=c(0,1),order=TRUE)

 f<-table(CS_Test$All_Max_Bucket_Flag_updated,pred1)


 #sensitivity&specificity


 sensitivity<-f[2,2]/(f[2,2]+f[2,1])

 specificitiy<-f[1,1]/(f[1,2]+f[1,1])

 sensi_spe<-(sensitivity*specificitiy)^(1/2)

 accuracy=(f[1,1]+f[2,2])/(f[1,1]+f[2,2]+f[2,1]+f[1,2])

 PnL <- ((f[1,1]/(f[1,1] +f[2,1]))*0.02) +

  ((f[2,1]/(f[1,1] + f[2,1]))*-0.45)

 result1 <- as.data.frame(cbind(i*0.01, accuracy, sensi_spe, PnL))

 result <- rbind(result,result1)

 rm(accuracy, sensi_spe, PnL,i,result1)

}
```

```
result

################################################

# Computing and plotting ROC curve

################################################

resroc_behav_zc_test <- roc(CS_Test$All_Max_Bucket_Flag_updated, pred_card_behav_zc_test, levels =
 c(0, 1), direction = "<")

rocobj <- plot.roc(resroc_behav_zc_test, print.auc = TRUE)

rocobj

coords(rocobj, x="best", input="threshold", best.method="youden")


#### Classification tree

printcp(cart.model_behav_zc)

plotcp(cart.model_behav_zc)

prunetree_cart.model_behav_zc_test <- prune(cart.model_behav_zc, cp =
 cart.model_behav_zc$cptable[which.min(cart.model_behav_zc$cptable[,"xerror"]),"CP"])

prunetree_pred_behav_zc_test <- predict(prunetree_cart.model_behav_zc_test, newdata=CS_Test,
 type="vector")

table(real=CS_Test$All_Max_Bucket_Flag_updated, predict=prunetree_pred_behav_zc_test)


prunetree_confus.matrix_behav_zc_test <- table(real=CS_Test$All_Max_Bucket_Flag_updated,
 predict=prunetree_pred_behav_zc_test)


### K-fold Cross-Validation_overfitting

# AttritionFlag

AttritionFlag <-ifelse(CS_Test$Attrition_Flag == 0 ,"x", "y" )

AttritionFlag

CS_Test <- cbind(CS_Test, AttritionFlag)

summary(AttritionFlag)

attributes(AttritionFlag)

AttritionFlag <- factor(AttritionFlag)

attributes(AttritionFlag)


# WriteoffFlag

WriteoffFlag  <-ifelse(CS_Test$Writeoff_Flag  == 0 ,"a", "b" )

WriteoffFlag
```

```
CS_Test <- cbind(CS_Test, WriteoffFlag )

summary(WriteoffFlag )

attributes(WriteoffFlag )

WriteoffFlag  <- factor(WriteoffFlag )

attributes(WriteoffFlag )


# MaxBucketFlag2

MaxBucketFlag2 <-ifelse(CS_Test$All_Max_Bucket_Flag_updated == 0 ,"first", "second" )

MaxBucketFlag2

CS_Test <- cbind(CS_Test, MaxBucketFlag2)

summary(MaxBucketFlag2)

attributes(MaxBucketFlag2)

MaxBucketFlag2 <- factor(MaxBucketFlag2)

attributes(MaxBucketFlag2)


train_control_behav_zc_test <- trainControl(method="cv", number=10, classProbs = TRUE,
 savePredictions = TRUE)

train_control.model_behav_zc_test <- train(MaxBucketFlag2 ~ Spending_Limit + Avg_Balances +
 Ytd_Amt_Cash + Current_Balance + BlacklistFlag, data = CS_Test, method="rpart", metric = "ROC",
 parms = list(split = "gini"), trControl=train_control_behav_zc_test, na.action = na.rpart)

train_control.model_behav_zc_test


#############################################################

###### Bucket Prediction

#############################################################

cart.model_behav_bp <- rpart(formula = All_Max_Bucket_updated ~ Spending_Limit + Current_Balance
 + Avg_Payments + Avg_Balances + Py_Amt_Cash + Ytd_Amt_Cash + Attrition_Flag + Writeoff_Flag +
 Account_Status_Bucket + Blacklist_Flag + Oy_Amt_Cash, data = CS_Train)

summary(cart.model_behav_bp)

cart.model_behav_bp

rpart.plot(cart.model_behav_bp)

plot(cart.model_behav_bp)

text(cart.model_behav_bp)


rparty.tree_behav_bp <- as.party(cart.model_behav_bp)
```

```
rparty.tree_behav_bp

plot(rparty.tree_behav_bp)


##############################

############################## predict_Train

pred_behav_bp_train <- predict(object = cart.model_behav_bp, newdata = CS_Train, type = "vector")

table(real = CS_Train$All_Max_Bucket_updated, predict=pred_behav_bp_train)

summary(pred_behav_bp_train)


confus.matrix_behav_bp_train <- table(real= CS_Train$All_Max_Bucket_updated,
 predict=pred_behav_bp_train)

confus.matrix_behav_bp_train


##############################

############################## predict_Test

pred_card_behav_bp_test <- predict(object = cart.model_behav_bp, newdata = CS_Test, type = "vector")

table(real = CS_Test$All_Max_Bucket_updated, predict=pred_card_behav_bp_test > 0.9394314)

summary(pred_card_behav_bp_test)


## NOTE:  predict=pred_card_behav_bp_test > cutoff

confus.matrix_behav_bp_test <- table(real= CS_Test$All_Max_Bucket_updated,
 predict=pred_card_behav_bp_test > 0.9394314)

confus.matrix_behav_bp_test

one_cm_behav_bptt_L <-
 confus.matrix_behav_bp_test[2,1]+confus.matrix_behav_bp_test[3,1]+confus.matrix_behav_bp_test[4,1]
 +confus.matrix_behav_bp_test[5,1]+confus.matrix_behav_bp_test[6,1]+confus.matrix_behav_bp_test[7,1
 ]

one_cm_behav_bptt_R <-
 confus.matrix_behav_bp_test[2,2]+confus.matrix_behav_bp_test[3,2]+confus.matrix_behav_bp_test[4,2]
 +confus.matrix_behav_bp_test[5,2]+confus.matrix_behav_bp_test[6,2]+confus.matrix_behav_bp_test[7,2
 ]

one_cm_behav_bptt_L

one_cm_behav_bptt_R


#### Accuracy ####

(confus.matrix_behav_bp_test[1,1] + one_cm_behav_bptt_R)/sum(confus.matrix_behav_bp_test)
```

```
sen_cm_behav_bp_test <- confus.matrix_behav_bp_test[1,1]/(confus.matrix_behav_bp_test[1,1] +
 one_cm_behav_bptt_L)

sen_cm_behav_bp_test

spe_cm_behav_bp_test <- one_cm_behav_bptt_R /(confus.matrix_behav_bp_test[1,2] +
 one_cm_behav_bptt_R)

spe_cm_behav_bp_test

sqrt(sen_cm_behav_bp_test*spe_cm_behav_bp_test)


## Average Profit/Loss ##

zero_behav_bp_test <-
 (confus.matrix_behav_bp_test[1,1]+confus.matrix_behav_bp_test[1,2])/sum(confus.matrix_behav_bp_tes
 t)

N2_behav_bp_test <-
 (confus.matrix_behav_bp_test[2,1]+confus.matrix_behav_bp_test[2,2])/sum(confus.matrix_behav_bp_tes
 t)

N3_behav_bp_test <-
 (confus.matrix_behav_bp_test[3,1]+confus.matrix_behav_bp_test[3,2])/sum(confus.matrix_behav_bp_tes
 t)

N4_behav_bp_test <-
 (confus.matrix_behav_bp_test[4,1]+confus.matrix_behav_bp_test[4,2])/sum(confus.matrix_behav_bp_tes
 t)

N5_behav_bp_test <-
 (confus.matrix_behav_bp_test[5,1]+confus.matrix_behav_bp_test[5,2])/sum(confus.matrix_behav_bp_tes
 t)

N6_behav_bp_test <-
 (confus.matrix_behav_bp_test[6,1]+confus.matrix_behav_bp_test[6,2])/sum(confus.matrix_behav_bp_tes
 t)

N7_behav_bp_test <-
 (confus.matrix_behav_bp_test[7,1]+confus.matrix_behav_bp_test[7,2])/sum(confus.matrix_behav_bp_tes
 t)


0*zero_behav_bp_test-2*N2_behav_bp_test -3*N3_behav_bp_test -4*N4_behav_bp_test -
 5*N5_behav_bp_test -6*N6_behav_bp_test -7*N7_behav_bp_test


#####################################################

# Computing and plotting ROC curve

#####################################################

res.roc_behav_bp_test<- roc(CS_Test$All_Max_Bucket_Flag_updated, pred_card_behav_bp_test, levels =
 c(0, 1), direction = "<")
```

```
rocobj_bp <- plot.roc(res.roc_behav_bp_test, print.auc = TRUE)

rocobj_bp

coords(rocobj_bp, x="best", input="threshold", best.method="youden") # Same than last line


#### Classification tree

printcp(cart.model_behav_bp)

plotcp(cart.model_behav_bp)

prunetree_cart.model_behav_bp_test <- prune(cart.model_behav_bp, cp =
 cart.model_behav_bp$cptable[which.min(cart.model_behav_bp$cptable[,"xerror"]),"CP"])

prunetree_pred_behav_bp_test <- predict(prunetree_cart.model_behav_bp_test, newdata=CS_Test,
 type="vector")

table(real=CS_Test$All_Max_Bucket_updated, predict=prunetree_pred_behav_bp_test)


prunetree_confus.matrix_behav_bp_test <- table(real=CS_Test$All_Max_Bucket_updated,
 predict=prunetree_pred_behav_bp_test)

sum(diag(prunetree_confus.matrix_behav_bp_test))/sum(prunetree_confus.matrix_behav_bp_test)


### K-fold Cross-Validation_overfitting

train_control_behav_bp_test <- trainControl(method="cv", number=10)

train_control.model_behav_bp_test <- train(All_Max_Bucket_updated ~ Spending_Limit +
 Avg_Payments + Avg_Balances + Py_Amt_Cash + Ytd_Amt_Cash + Current_Balance + Attrition_Flag
 + Writeoff_Flag + Account_Status_Bucket + Py_Max_Bucket_Flag + Ytd_Max_Bucket_Flag,
 data=CS_Test, method="rpart", trControl=train_control_app_zc)

train_control.model_behav_bp_test


#############################################################

###### Proft/Loss Prediction

#############################################################

cart.model_behav_pl <- rpart(formula = PNL_All_Max_Bucket ~ Spending_Limit + Avg_Payments +
 Avg_Balances + Py_Amt_Cash + Ytd_Amt_Cash + Current_Balance + Attrition_Flag + Writeoff_Flag +
 Account_Status_Bucket + Py_Max_Bucket_Flag + Ytd_Max_Bucket_Flag, data = CS_Train)

summary(cart.model_behav_pl)

cart.model_behav_pl

rpart.plot(cart.model_behav_pl)

plot(cart.model_behav_pl)

text(cart.model_behav_pl)
```

```
rparty.tree_behav_pl <- as.party(cart.model_behav_pl)

rparty.tree_behav_pl

plot(rparty.tree_behav_pl)


##############################

############################## predict_Train

pred_behav_pl_train <- predict(object = cart.model_behav_pl, newdata = CS_Train, type = "vector")

table(real = CS_Train$PNL_All_Max_Bucket, predict=pred_behav_pl_train)

summary(pred_behav_pl_train)


confus.matrix_behav_pl_train <- table(real= CS_Train$PNL_All_Max_Bucket,
 predict=pred_behav_pl_train)

sum(diag(confus.matrix_behav_pl_train))/sum(confus.matrix_behav_pl_train)


##############################

############################## predict_Test

pred_card_behav_pl_test <- predict(object = cart.model_behav_pl, newdata = CS_Test, type = "vector")

table(real = CS_Test$PNL_All_Max_Bucket, predict=pred_card_behav_pl_test)

summary(pred_card_behav_pl_test)


# NOTE: predict=pred_card_behav_pl_test > cutoff

confus.matrix_behav_pl_test <- table(real= CS_Test$PNL_All_Max_Bucket,
 predict=pred_card_behav_pl_test > -0.157857)

confus.matrix_behav_pl_test


one_cm_behav_pltt_L <-
 confus.matrix_behav_pl_test[1,1]+confus.matrix_behav_pl_test[2,1]+confus.matrix_behav_pl_test[3,1]+
 confus.matrix_behav_pl_test[4,1]+confus.matrix_behav_pl_test[5,1]

one_cm_behav_pltt_R <-
 confus.matrix_behav_pl_test[1,2]+confus.matrix_behav_pl_test[2,2]+confus.matrix_behav_pl_test[3,2]+
 confus.matrix_behav_pl_test[4,2]+confus.matrix_behav_pl_test[5,2]

one_cm_behav_pltt_L

one_cm_behav_pltt_R
```

```
zero_cm_behav_pltt_L <- confus.matrix_behav_pl_test[6,1]+confus.matrix_behav_pl_test[7,1]

zero_cm_behav_pltt_R <- confus.matrix_behav_pl_test[6,2]+confus.matrix_behav_pl_test[7,2]

zero_cm_behav_pltt_L

zero_cm_behav_pltt_R


#### Accuracy ####

(zero_cm_behav_pltt_L + one_cm_behav_pltt_R)/sum(confus.matrix_behav_pl_test)


sen_cm_behav_pl_test <- zero_cm_behav_pltt_L/(zero_cm_behav_pltt_L + one_cm_behav_pltt_L)

sen_cm_behav_pl_test

spe_cm_behav_pl_test <- one_cm_behav_pltt_R /(zero_cm_behav_pltt_R + one_cm_behav_pltt_R)

spe_cm_behav_pl_test

sqrt(sen_cm_behav_pl_test*spe_cm_behav_pl_test)


## Average Profit/Loss ##

N65_behav_pl_test <-
 (confus.matrix_behav_pl_test[1,1]+confus.matrix_behav_pl_test[1,2])/sum(confus.matrix_behav_pl_test)

N45_behav_pl_test <-
 (confus.matrix_behav_pl_test[2,1]+confus.matrix_behav_pl_test[2,2])/sum(confus.matrix_behav_pl_test)

N35_behav_pl_test <-
 (confus.matrix_behav_pl_test[3,1]+confus.matrix_behav_pl_test[3,2])/sum(confus.matrix_behav_pl_test)

N25_behav_pl_test <-
 (confus.matrix_behav_pl_test[4,1]+confus.matrix_behav_pl_test[4,2])/sum(confus.matrix_behav_pl_test)

N15_behav_pl_test <-
 (confus.matrix_behav_pl_test[5,1]+confus.matrix_behav_pl_test[5,2])/sum(confus.matrix_behav_pl_test)

P0_behav_pl_test <-
 (confus.matrix_behav_pl_test[6,1]+confus.matrix_behav_pl_test[6,2])/sum(confus.matrix_behav_pl_test)

P3_behav_pl_test <-
 (confus.matrix_behav_pl_test[7,1]+confus.matrix_behav_pl_test[7,2])/sum(confus.matrix_behav_pl_test)


0.03*P3_behav_pl_test + 0*P0_behav_pl_test -0.15*N15_behav_pl_test - 0.25*N25_behav_pl_test -
 0.35*N35_behav_pl_test -0.45*N45_behav_pl_test -0.65*N65_behav_pl_test


################################################

# Computing and plotting ROC curve

################################################
```

```
res.roc_behav_pl_test <- roc(CS_Test$PNL_All_Max_Bucket, pred_card_behav_pl_test)

rocobj_pl_test <- plot.roc(res.roc_behav_pl_test, print.auc = TRUE)

rocobj_pl_test

coords(rocobj_pl_test, x="best", input="threshold", best.method="youden") # Same than last line


#### Classification tree

printcp(cart.model_behav_pl)

plotcp(cart.model_behav_pl)

prunetree_cart.model_behav_pl_test <- prune(cart.model_behav_pl, cp =
 cart.model_behav_pl$cptable[which.min(cart.model_behav_pl$cptable[,"xerror"]),"CP"])

prunetree_pred_behav_pl_test <- predict(prunetree_cart.model_behav_pl_test, newdata=CS_Test,
 type="vector")

table(real=CS_Test$PNL_All_Max_Bucket, predict=prunetree_pred_behav_pl_test)


prunetree_confus.matrix_behav_pl_test <- table(real=CS_Test$PNL_All_Max_Bucket,
 predict=prunetree_pred_behav_pl_test)

sum(diag(prunetree_confus.matrix_behav_pl_test))/sum(prunetree_confus.matrix_behav_pl_test)


### K-fold Cross-Validation_overfitting

train_control_behav_pl_test <- trainControl(method="cv", number=10)

train_control.model_behav_pl_test <- train(PNL_All_Max_Bucketd ~ Spending_Limit + Avg_Payments +
 Avg_Balances + Py_Amt_Cash + Ytd_Amt_Cash + Current_Balance + Attrition_Flag + Writeoff_Flag +
 Account_Status_Bucket + Py_Max_Bucket_Flag + Ytd_Max_Bucket_Flag, data=CS_Test,
 method="rpart", trControl=train_control_app_zc)

train_control.model_behav_pl_test
```

12. References

Maklin, C. (2019). Random Forest In R. [online] Medium. Available at: https://towardsdatascience.com/random-forest-in-r-f66adf80ec9.

Wikipedia. (2020). Random forest. [online] Available at: https://en.wikipedia.org/wiki/Random_forest#/media/File:Random_forest_diagram_complete.png.

www.rdocumentation.org. (n.d.). randomForest function | R Documentation. [online] Available at: https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest.

Analytics Vidhya. (2021). Sensitivity, Specificity and Accuracy - Decoding the Relationship. [online] Available at: https://www.analyticsvidhya.com/blog/2021/06/classification-problem-relation-between-sensitivity-specificity-and-accuracy/ [Accessed 23 Sep. 2021].

Verma, S. (2021). Logistic Regression From Scratch in Python. [online] Medium. Available at: https://towardsdatascience.com/logistic-regression-from-scratch-in-python-ec66603592e2 [Accessed 23 Sep. 2021].

www.sciencedirect.com. (n.d.). Logistic Regression - an overview | ScienceDirect Topics. [online] Available at: https://www.sciencedirect.com/topics/computer-science/logistic-regression.

https://www.facebook.com/kdnuggets (2020). Decision Tree Algorithm, Explained - KDnuggets. [online] KDnuggets. Available at: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

Kdnuggets.com. (2019). A Beginner's Guide to Linear Regression in Python with Scikit-Learn. [online] Available at: https://www.kdnuggets.com/2019/03/beginners-guide-linear-regression-python-scikit-learn.html

Abdou, H. & Pointon, J. (2011). *Credit scoring, statistical techniques and evaluation criteria: a review of the literature*. Intelligent Systems in Accounting, Finance & Management, 18 (2-3), pp. 59-88.

Abdou, H., Pointon, J., El Masry, A. (2008). *Neural nets versus conventional techniques in credit scoring in Egyptian banking*. Expert Systems with Applications 35 (3): 12751292.

Abdou, H.A. and Pointon, J. (2011). *CREDIT SCORING, STATISTICAL TECHNIQUES AND EVALUATION CRITERIA: A REVIEW OF THE LITERATURE*. Intelligent Systems in Accounting, Finance and Management, 18(2-3), pp.59–88.

Amari, A. (2002). *The credit evaluation process and the role of credit scoring: A case study of Qatar*. Ph.D. Thesis, University College Dublin.

Ala'raj, M. and Abbod, M.F. (2016). *Classifiers consensus system approach for credit scoring*. Knowledge-Based Systems, 104, pp.89–105.

Anderson, R. (2007). *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*. New York: Oxford University Press.

Baesens, B., Gestel, T. V., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J. (2003). Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring. Journal of the Operational Research Society 54 (6): 627-635.

Basel Committee. (2010). *Basel III: A global regulatory framework for more resilient banks and banking systems.* Basel Committee on Banking Supervision, Basel.

Bastos, J. (2008). *Credit scoring with boosted decision trees*. [online] mpra.ub.uni-muenchen.de. Available at: https://mpra.ub.uni-muenchen.de/8156/ [Accessed 21 Sep. 2021].

Bensic, M., Sarlija, N. and Zekic-Susac, M. (2005). *Modelling small-business credit scoring by using logistic regression, neural networks and decision trees.* Intelligent Systems in Accounting, Finance and Management, 13(3), pp.133–150.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* wadsworth. Belmont, CA.

Chandler, G. G., Coffman, J. Y. 1979. *A comparative analysis of empirical vs. judgemental credit evaluation*. The Journal of Retail Banking 1 (2): 15-26.

Credit scoring Case study in data analytics. Deloitte (2016). [online] Available at: https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Financial-Services/gx-be-aers-fsi-credit-scoring.pdf.

Crook, J. N. (1996). *Credit scoring: An overview. Working paper series No. 96/13*, British Association, Festival of Science. University of Birmingham, The University of Edinburgh.

Crook, J., Edelman D., Thomas, L. (2007). *Recent developments in consumer credit risk assessment.* European Journal of Operational Research 183 (3): 1447-1465.

Fernandez Vidal, M. and Barbon, F. (2019). *CREDIT SCORING IN FINANCIAL INCLUSION How to use advanced analytics to build credit-scoring models that increase access.* [online] Available at: https://www.cgap.org/sites/default/files/publications/2019_07_Technical_Guide_CreditScore.pdf.

Freund, R.J., Wilson, W.J. and Sa, P. (2006). *Regression analysis.* Oxford: Academic.

Galindo J, Tamayo P. (2000). C*redit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modelling Applications.* Computational Economics 15: 107-143.

Grosan, C., & Abraham, A. (2011). *Intelligent systems Springer*.

Gup, B. E., Kolari, J. W. (2005). *Commercial Banking: The management of risk*. Alabama: John Wiley & Sons, Inc.

Hand, D.J. and Henley, W.E. (1997) *Statistical Classification Methods in Consumer Credit Scoring: A Review*. Journal of Royal Statistical Society, 160, 523-541.

KARIMI, A. (2014). *Evaluation of the Credit Risk with Statistical analysis*. International Journal of Academic Research in Accounting, Finance and Management Sciences, 4(3).

Kaseke, C. & Murairwa, S. (2019). *Analysis of creditworthiness determinant factors in Zimbabwe.* Accepted for publication in Amity Journal of Finance (AJF), 4(1). ISSN: 2455- 9741 (Print), ISSN: 2456-1568 (Online).

Lee, T., Chen, I. (2005). *A Two-Stage Hybrid Credit Scoring Model Using Artificial Neural Networks and Multivariate Adaptive Regression Splines*. Expert Systems with Applications 28 (4): 743-752.

Lee, T., Chiu, C. Lu, C., Chen, I. (2002). *Credit Scoring Using the Hybrid Neural Discriminant Technique.* Expert Systems with Applications 23 (3): 245-254.

Lee, T., Chiu, C., Chou, Y., & Lu, C. (2006). *Mining the customer credit using classification and regression tree and multivariate adaptive regression splines*. Computational Statistics & Data Analysis, 50(4), 1113-1130.

Lessmann, S., Baesens, B., Seow, H.-V. and Thomas, L.C. (2015). *Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. European Journal of Operational Research*, 247(1), pp.124–136.

Lessmann, S., Baesens, B., Seow, H., & Thomas, L. C. (2015). *Benchmarking state-of-the-art classification algorithms for credit-scoring: An update of research*. European Journal of Operational Research.

Li, X.-L. and Zhong, Y. (2012). A*n Overview of Personal Credit Scoring: Techniques and Future Work*. International Journal of Intelligence Science, 02(04), pp.181–189.

Liberati, C. and Camillo, F. (2018). *Personal values and credit scoring: new insights in the financial prediction*. Journal of the Operational Research Society, 69(12), pp.1994–2005.

Liu, Y. (2001). *New issues in credit-scoring application.* Work Report no, 16

Mansouri, S. and Dastoori, M. (2013). *Credit Scoring Model for Iranian Banking Customers and Forecasting Creditworthiness of Borrowers*. International Business Research, 6(10).

Nisbet, R., Elder, J. and Miner, G. (2009) *Handbook of Statistical Analysis and Data Mining Applications.* Academic Press, p.864.

Ong, C., Huang, J., Tzeng, G. (2005). *Building Credit Scoring Models Using Genetic Programming*. Expert Systems with Applications 29 (1): 41-47.

Samreen, A. and Zaidi, F. (2012). *Design and Development of Credit Scoring Model for the Commercial banks of Pakistan: Forecasting Creditworthiness of Individual Borrowers*. International Journal of Business and Social Science, [online] 3(17). Available at: https://ijbssnet.com/journals/Vol_3_No_17_September_2012/17.pdf.

Thomas, L. C. (2000). A *survey of credit and behavioural scoring: Forecasting financial risk of lending to consumers*. International Journal of Forecasting, 16(2), 149-172.

W, D., Lemeshow, S. and Sturdivant, R.X. (2013). *Applied logistic regression*. [online] New York, Etc.: John Wiley And Sons, Cop. Available at: https://www.wiley.com/en-us/Applied+Logistic+Regression%2C+3rd+Edition-p-9780470582473.

Zekic-Susac, M., Sarlija, N., & Bensic, M. (2004). *Small business credit-scoring: A comparison of logistic regression, neural network, and decision tree models.* Information Technology Interfaces, 2004. 26th International Conference on, 265-270.

https://www.facebook.com/jason.brownlee.39 (2018). *What is a Confusion Matrix in Machine Learning*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/confusion-matrix-machine-learning/.