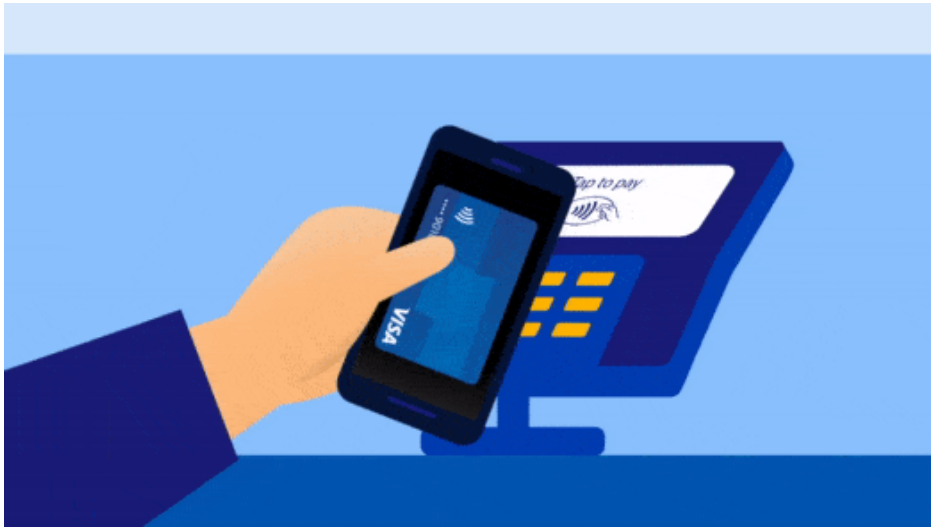
IST 707 PROJECT: BANK CHURNERS



IST 707 APPLIED MACHINE LEARNING

RENJINI RAJAN AND AMANDA NORWOOD

TABLE OF CONTENTS

INTRODUCTION

The idea of banking began a long time ago in 1,800 B.C. in Babylon as a method of lending to people. However, lending was not limited to money, it also included banks lending out seeds before harvesting season began. Once harvesting season was over, farmers would pay back their seed loan from the harvest. In terms of money, this also allowed people to take out loans, deposit money for safekeeping, and exchange currency.

The core idea of banking was to have a safe method of storing money, gold, and silver. In today's time, banking has evolved upon the original idea and expanded their business to credit cards, home lending, checking accounts, savings accounts, auto loans, and much more. The banking industry has evolved to a highly competitive industry as consumers have a vast selection of banks and products to choose from.

A study by the Federal Reserve Bank of San Francisco found that in 2021, credit cards were used to make 28% of all payments. This is the highest level it’s been since the study began in 2016, indicating that credit cards are gaining ground compared to cash or other forms of payment. The percentage of payments made by credit cards is larger for households with higher income: it jumps to 34% for households earning $100,000 to $149,999 and 44% for those earning over $150,000. Another study by Experian suggests that, on average, people have three different credit card accounts.

These studies and statistics show that credit card industry is a lot more competitive than we initially thought. Retaining existing customers becomes equally

important as gaining new customers. Banks must continuously assess their customer base to predict their needs and when they are about to churn-- meaning leave their services. In the same scenario, banks also want to understand what it takes to retain their current customer base while attracting new customers with the same attributes. The goal of modern banks is to retain their customers and limit the number of customers that leave their services for another bank.

As part of the regulatory requirements like KYC(know your customer) banks do collect detailed information from its customers. This includes their demographic information also. With the growth of business and data analytics this information can be used to understand customer behaviors and can become a primary source to retaining its customers. Banks can study the change in customer behavior and spending trends to improve its customer satisfaction and can be used to attract new customers.

The following study uses demographic data from a bank to understand the reason for its customer attrition. The study includes various analytical models to carefully evaluate the data and understand key aspects of customer behavior which is a clear indication that they are going to stop using the current services. This can be used by the marketing team to reach out to the customer with more personalized recommendations and to improve their products and services in general.

ANALYSIS AND MODELS

The analysis is trying to address the concerns of a bank manager who noticed that more and more customers are leaving their credit card services and the manager is concerned with the amount of people leaving. Bank collects and stores information about its existing and past customers and the data includes personal, demographic and

the account details of the products associated with each customer. By analyzing the data provided by the manager to proactively advise on their business practices to retain their current customers, new products for their customer base, and the potential attributes that cause customers to leave their credit card services. Analysis will include using a variety of visualization and machine learning methods and then comparing the results to make a recommendation to the marketing team.

ABOUT THE DATA

The data set was pulled from Kaggle, and it contains 10,000 rows of data with each row representing a customer. The layout and a brief description as shown below:

Column Name	Description
CLIENTNUM	Client number. Unique identifier for the customer holding the account
Attrition_Flag	Internal event (customer activity) variable - if the account is closed then 1 else 0
Customer_Age	Demographic variable - Customer's Age in Years

Gender	Demographic variable - M=Male, F=Female
Dependent_count	Demographic variable - Number of dependents
Education_Level	Demographic variable - Educational Qualification of the account holder (example: high school, college graduate, etc.)
Marital_Status	Demographic variable - Married, Single, Divorced, Unknown
Income_Category	Demographic variable - Annual Income Category of the account holder (< $40K, $40K - 60K, $60K - $80K, $80K-$120K, > $120K, Unknown)
Card_Category	Product Variable - Type of Card (Blue, Silver, Gold, Platinum)

Months_on_book	Period of relationship with bank
Total_Relationship_Count	Total no. of products held by the customer
Months_Inactive_12_mon	No. of months inactive in the last 12 months
Contacts_Count_12_mon	No. of Contacts in the last 12 months
Credit_Limit	Credit Limit on the Credit Card
Total_Revolving_Bal	Total Revolving Balance on the Credit Card
Avg_Open_To_Buy	Open to Buy Credit Line (Average of last 12 months)
Total_Amt_Chng_Q4_Q1	Change in Transaction Amount (Q4 over Q1)
Total_Trans_Amt	Total Transaction Amount (Last 12 months)
Total_Trans_Ct	Total Transaction Count (Last 12 months)

Total_Ct_Chng_Q4_Q1	Change in Transaction Count (Q4 over Q1)
Avg_Utilization_Ratio	Average Card Utilization Ratio

R PACKAGES USED

For the analysis purposes we have used several r packages as shown below:

```{r include = FALSE, message = FALSE, warning = FALSE}
#install.packages('dplyr')
#install.packages('ggplot2')
#install.packages('data.table')
#install.packages('treemap')
#install.packages('DataExplorer')
#install.packages('highcharter')
#install.packages('ggcorrplot')
#install.packages('viridisLite')
#install.packages('rpart')
#install.packages('rpart.plot')
#install.packages('rattle')
#install.packages('caret')
#install.packages('janitor')
#install.packages('tcltk')
#install.packages('party')
#install.packages("randomForest")
#install.packages("e1071")
#install.packages("FactoMineR")
#install.packages('cluster')
#install.packages('factoextra')
```

```
library(randomForest)
library(dplyr)
library(ggplot2)
library(highcharter)
library(data.table)
library(treemap)
library(DataExplorer)
library(viridisLite)
library(ggcorrplot)
library(arules)
library(class)
library(BayesFactor)
library(arulesViz)
library(rpart)
library(rpart.plot)
library(rattle)
library(janitor)
library(caret)
library(e1071)
library(e1071)
library(caTools)
library(arulesViz)
library(party)
##library(tcltk)
library(FactoMineR)
library(cluster)
library(factoextra)
```

LOADING THE DATA

```{r}
#ccdata = read.csv('/Users/fruitisushi/Documents/Grad Classes/IST 707 - Applied Machine Learning/Project/BankChurners.csv')
ccdata= read.csv("/Users/renjinirajan/Desktop/Masters/IST 707- Applied Machine Learning/BankChurners.csv")
ccdata_cor = ccdata
```

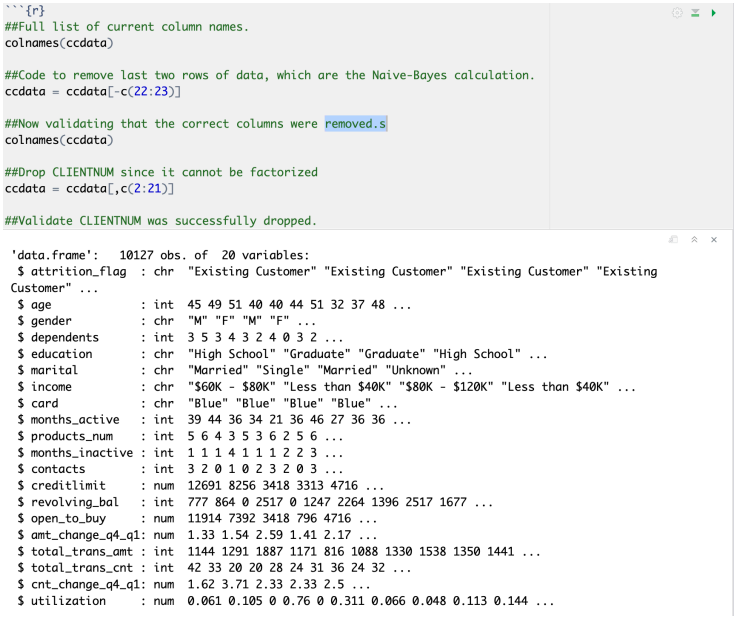
CLEANUP

Before running models on the data, the following steps were performed:

- Rename the column to a more user-friendly format.

```
##Rename columns

setnames(ccdata,
         old = c('Attrition_Flag',
                 'Customer_Age',
                 'Gender',
                 'Dependent_count',
                 'Education_Level',
                 'Marital_Status',
                 'Income_Category',
                 'Card_Category',
                 'Months_on_book',
                 'Total_Relationship_Count',
                 'Months_Inactive_12_mon',
                 'Contacts_Count_12_mon',
                 'Credit_Limit',
                 'Total_Revolving_Bal',
                 'Avg_Open_To_Buy',
                 'Total_Amt_Chng_Q4_Q1',
                 'Total_Trans_Amt',
                 'Total_Trans_Ct',
                 'Total_Ct_Chng_Q4_Q1',
                 'Avg_Utilization_Ratio'),
         new = c('attrition_flag',
         'age',
         'gender',
         'dependents',
         'education',
         'marital',
         'income',
         'card',
         'months_active',
         'products_num',
         'months_inactive',
         'contacts',
         'creditlimit',
         'revolving_bal',
         'open_to_buy',
         'amt_change_q4_q1',
         'total_trans_amt',
         'total_trans_cnt',
         'cnt_change_q4_q1',
         'utilization'
         ))
```

- Identify variables to remove because the data is bad or not useful.
  - We have successfully dropped the CLIENTNUM and the two columns that contain Naive-Bayes analysis. Again, those variables could not be factorized or were not relevant to our analysis and thus dropped from the data set. We also validated the correct columns were dropped and confirmed we still have zero complete columns.



```r
##Full list of current column names.
colnames(ccdata)

##Code to remove last two rows of data, which are the Naive-Bayes calculation.
ccdata = ccdata[-c(22:23)]

##Now validating that the correct columns were removed.s
colnames(ccdata)

##Drop CLIENTNUM since it cannot be factorized
ccdata = ccdata[,c(2:21)]

##Validate CLIENTNUM was successfully dropped.
```

```
'data.frame':   10127 obs. of  20 variables:
 $ attrition_flag  : chr  "Existing Customer" "Existing Customer" "Existing Customer" "Existing
Customer" ...
 $ age             : int  45 49 51 40 40 44 51 32 37 48 ...
 $ gender          : chr  "M" "F" "M" "F" ...
 $ dependents      : int  3 5 3 4 3 2 4 0 3 2 ...
 $ education       : chr  "High School" "Graduate" "Graduate" "High School" ...
 $ marital         : chr  "Married" "Single" "Married" "Unknown" ...
 $ income          : chr  "$60K - $80K" "Less than $40K" "$80K - $120K" "Less than $40K" ...
 $ card            : chr  "Blue" "Blue" "Blue" "Blue" ...
 $ months_active   : int  39 44 36 34 21 36 46 27 36 36 ...
 $ products_num    : int  5 6 4 3 5 3 6 2 5 6 ...
 $ months_inactive : int  1 1 1 4 1 1 1 2 2 3 ...
 $ contacts        : int  3 2 0 1 0 2 3 2 0 3 ...
 $ creditlimit     : num  12691 8256 3418 3313 4716 ...
 $ revolving_bal   : int  777 864 0 2517 0 1247 2264 1396 2517 1677 ...
 $ open_to_buy     : num  11914 7392 3418 796 4716 ...
 $ amt_change_q4_q1: num  1.33 1.54 2.59 1.41 2.17 ...
 $ total_trans_amt : int  1144 1291 1887 1171 816 1088 1330 1538 1350 1441 ...
 $ total_trans_cnt : int  42 33 20 20 28 24 31 36 24 32 ...
 $ cnt_change_q4_q1: num  1.62 3.71 2.33 2.33 2.5 ...
 $ utilization     : num  0.061 0.105 0 0.76 0 0.311 0.066 0.048 0.113 0.144 ...
```

- Removed rows with NA's.
- Set appropriate types for each column (e.g., factor, numeric).
- Visualize the variables to give a sense of where to focus analysis.
- Look for associations/correlations between variables.
- Perform data transformations for each method, such as creating transactions before using ARM and converting values to numbers before using k-means.

FACTORIZATION

1: attrition_flag

1/20: attrition_flag

This variables lets us know whether the customer is current or past.

```r
unique(ccdata$attrition_flag)
#There are two unique fields
ccdata$attrition_flag = factor(ccdata$attrition_flag)
#We should see a return factor of 2
str(ccdata$attrition_flag)
#2 levels are returned
```

```
 [1] Existing Customer Attrited Customer
 Levels: Attrited Customer Existing Customer
```

2: gender

```
#### 2/20: gender
The gender is defined as Male or Female.
```{r}
unique(ccdata$gender)
#There are two unique fields
ccdata$gender = factor(ccdata$gender)
#We should see a return factor of 2
str(ccdata$gender)
#2 levels are returned
```
```

```
[1] M F
Levels: F M
```

3: education

```
#### 3/20: education
Education level of the customer.
```{r}
unique(ccdata$education)
#There are 7 unique fields
ccdata$education = factor(ccdata$education)
#We should see a return factor of 7
str(ccdata$education)
#7 levels are returned
```
```

```
[1] High School   Graduate      Uneducated    Unknown       College       Post-Graduate
[7] Doctorate
Levels: College Doctorate Graduate High School Post-Graduate Uneducated Unknown
```

4: marital status

```
#### 4/20: marital status
This is the customers marital status.
```{r}
unique(ccdata$marital)
#There are 4 unique fields
ccdata$marital = factor(ccdata$marital)
#We should see a return factor of 4
str(ccdata$marital)
#4 levels are returned
```
```

```
[1] Married  Single   Unknown  Divorced
Levels: Divorced Married Single Unknown
```

5: income

```
#### 5/20: income
This variable shows the customers income level.
```{r}
unique(ccdata$income)
#There are 6 unique fields
ccdata$income = factor(ccdata$income)
#We should see a return factor of 6
str(ccdata$income)
#6 levels are returned
```
```

```
[1] $60K - $80K    Less than $40K $80K - $120K   $40K - $60K    $120K +        Unknown
Levels: $120K + $40K - $60K $60K - $80K $80K - $120K Less than $40K Unknown
```

6: card

```
#### 6/20: card
This field shows the type of credit card the customer is using.
```{r}
unique(ccdata$card)
#There are 4 unique fields
ccdata$card = factor(ccdata$card)
#We should see a return factor of 4
str(ccdata$card)
#4 levels are returned
```
```

```
[1] Blue     Gold     Silver   Platinum
Levels: Blue Gold Platinum Silver
```

7: dependents

```
#### 7/20: dependents
This shows the number of dependents the customer has reported.
```{r}
unique(ccdata$dependents)
#There are 6 unique fields
ccdata$dependents = factor(ccdata$dependents)
#We should see a return factor of 6
str(ccdata$dependents)
#6 levels are returned
```

```
[1] 5 6 4 3 2 1
 Factor w/ 6 levels "1","2","3","4",..: 5 6 4 3 5 3 6 2 5 6 ...
```

8: products_num
```

9/20: contacts
This field shows the number of times the bank has contacted the customer.

```{r}
unique(ccdata$contacts)
#There are 7 unique fields
ccdata$contacts = factor(ccdata$contacts)
#We should see a return factor of 7
str(ccdata$contacts)
#7 levels are returned
```

```
[1] 3 2 0 1 4 5 6
Levels: 0 1 2 3 4 5 6
```

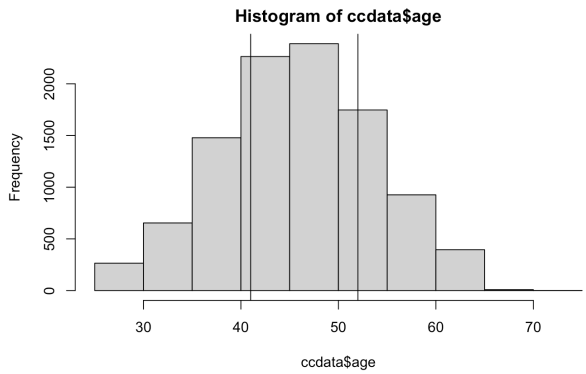
10: months_inactive

10/20: months_inactive
This field shows the number of months the customer has been inactive with the bank.

```{r}
unique(ccdata$months_inactive)
#There are 7 unique fields
ccdata$months_inactive = factor(ccdata$months_inactive)
#We should see a return factor of 7
str(ccdata$months_inactive)
#7 levels are returned
```

```
[1] 1 4 2 3 6 0 5
Levels: 0 1 2 3 4 5 6
```

11: age



```
#### 11/20: age
This is the customers age.

```{r}
#We can see the customer ages can age from 26 to 73
min(ccdata$age)
max(ccdata$age)

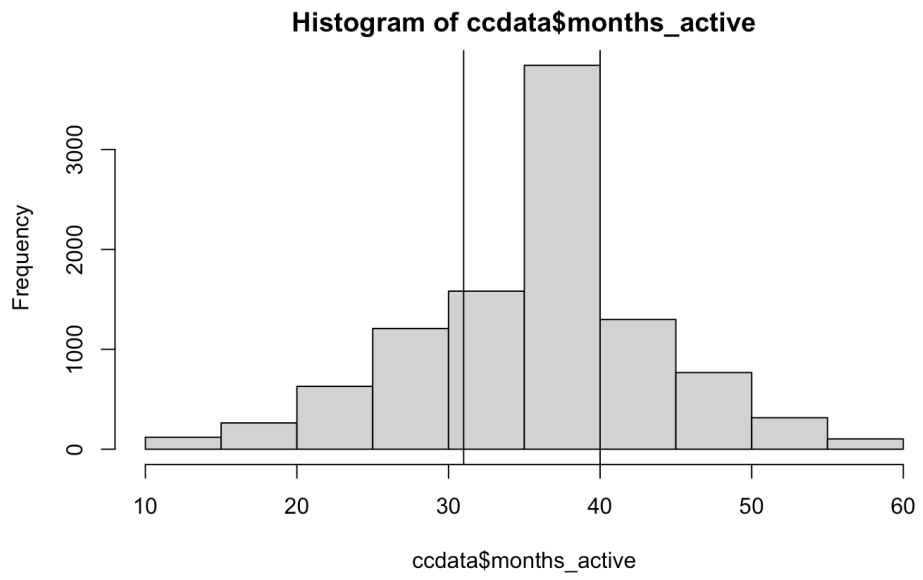
#The histogram shows the the core of the customer ages are between 41 and 52.
hist(ccdata$age)
abline(v=quantile(ccdata$age,c(0.25)), col="black")
abline(v=quantile(ccdata$age,c(0.75)), col="black")

#Factorize age
ccdata$age = cut(ccdata$age, breaks = c(20,30,40,50,60,70,Inf),
                         labels=c('twenties','thirties','fourties','fifties','sixties','seventies'))

#Verify it is factorized
str(ccdata$age)

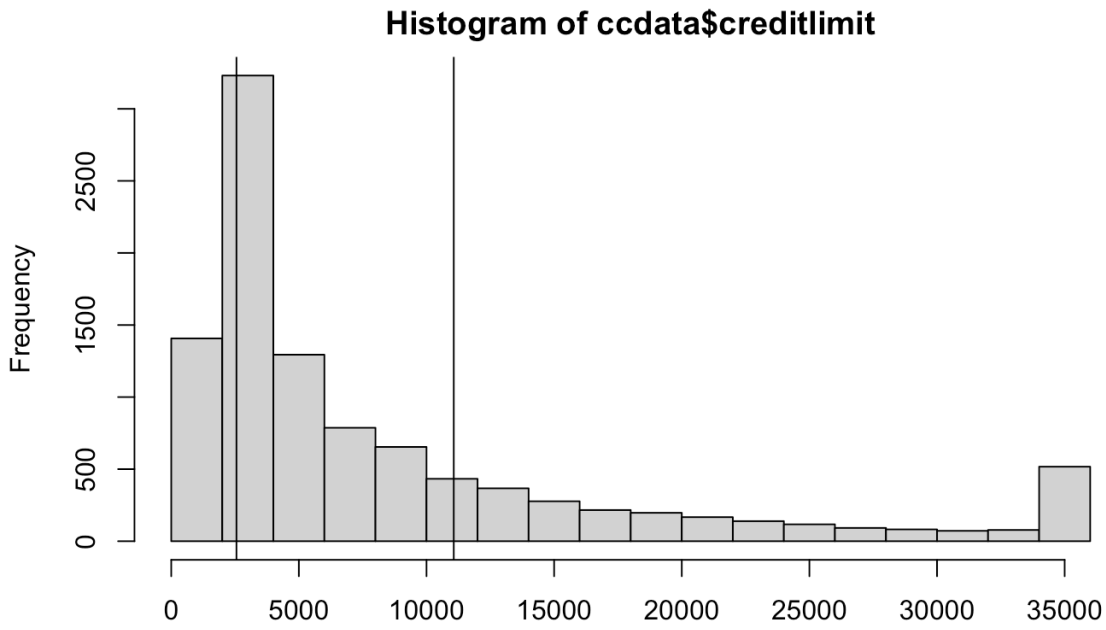
#Verify there are no NA's
(sum(is.na(ccdata)))
```
```

12: months active



```r
#Months active range between 13 and 56.
min(ccdata$months_active)
max(ccdata$months_active)

hist(ccdata$months_active)
abline(v=quantile(ccdata$months_active,c(0.25)), col="black")
abline(v=quantile(ccdata$months_active,c(0.75)), col="black")


#Factorize months_active
ccdata$months_active = cut(ccdata$months_active, breaks = c(10,20,30,40,50,60,70,Inf),
                           labels=c('tens','twenties','thirties','fourties','fifties','sixties','seventies'))

#Verify it was factorized
str(ccdata$months_active)

#Verify no NA data
(sum(is.na(ccdata)))
```

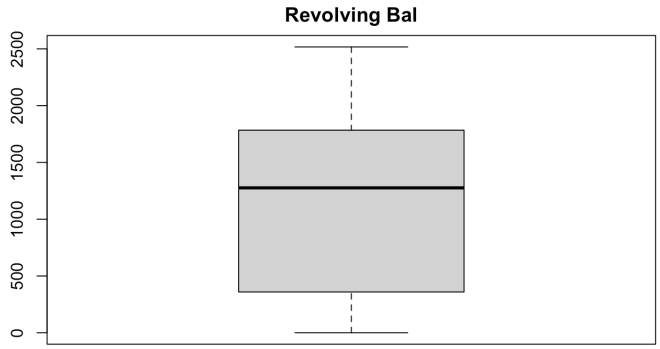
13: credit_limit



13/20: credit_limit
This field shows the credit limit of the customer in dollar amounts.

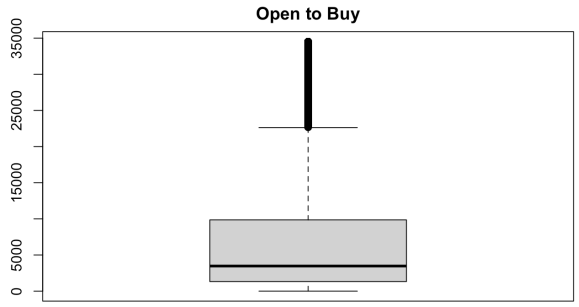
```{r}
#Credit limit ranges from $1,438 to $34,516
min(ccdata$creditlimit)
max(ccdata$creditlimit)

#Credit limit has a wide range with 50% of its customers falling between roughly $2,600 and $11,000
hist(ccdata$creditlimit)
abline(v=quantile(ccdata$creditlimit,c(0.25)), col="black")
abline(v=quantile(ccdata$creditlimit,c(0.75)), col="black")

#Factorize into 4 bins
min_ccl = min(ccdata$creditlimit) - 1
max_ccl = max(ccdata$creditlimit) + 1
bins_ccl = 4
width_ccl=(max_ccl-min_ccl)/bins_ccl;

ccdata$creditlimit = cut(ccdata$creditlimit, breaks=seq(min_ccl, max_ccl, width_ccl),
                      labels=c('Low','Medium','High','Max'))

#Verify
str(ccdata$creditlimit)

#Verify no NA data
(sum(is.na(ccdata)))
```

14: revolving_bal



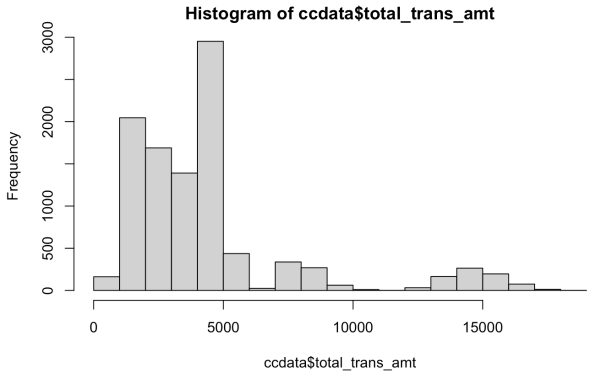

```{r}
#The balance ranges from $0 to $2,517
min(ccdata$revolving_bal)
max(ccdata$revolving_bal)

boxplot(ccdata$revolving_bal,main='Revolving Bal')

#We will break down the data by every 500
min_rb <- min(ccdata$revolving_bal) - 1
max_rb <- max(ccdata$revolving_bal) + 1
bins_rb = 6
width_rb=(max_rb-min_rb)/bins_rb;

ccdata$revolving_bal = cut(ccdata$revolving_bal, breaks=seq(min_rb, max_rb, width_rb),
                       labels=c('0','500s','1000s','1500s','2000s','2500s'))

#Verify factorization
str(ccdata$revolving_bal)

#Verify no NAs
(sum(is.na(ccdata$revolving_bal)))
```

15: open_to_buy




```{r}
boxplot(ccdata$open_to_buy,main='Open to Buy')

min_otb <- min(ccdata$open_to_buy) - 1
max_otb <- max(ccdata$open_to_buy) + 1
bins_otb = 6
width_otb=(max_otb-min_otb)/bins_otb;

#Factorize
ccdata$open_to_buy = cut(ccdata$open_to_buy, breaks=seq(min_otb, max_otb, width_otb),
                      labels=c('0','500s','1000s','1500s','2000s','2500s'))

#Verify
str(ccdata$open_to_buy)
#Verify no NAs
(sum(is.na(ccdata$open_to_buy)))
```

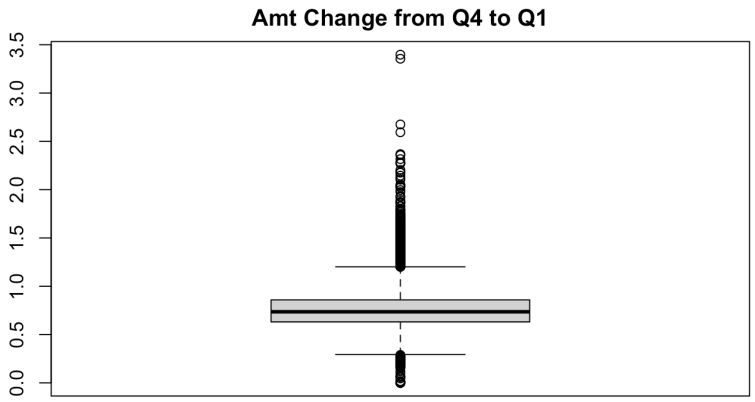
16: total_trans_amt



16/20: total_trans_amt
This is the total dollar amount of transactions in the last 12 months.

```{r}
hist(ccdata$total_trans_amt)
min_tta <- min(ccdata$total_trans_amt) - 1
max_tta <- max(ccdata$total_trans_amt) + 1
bins_tta = 3
width_tta=(max_tta-min_tta)/bins_tta

#Factorize
ccdata$total_trans_amt = cut(ccdata$total_trans_amt, breaks=seq(min_tta, max_tta, width_tta),
                      labels=c('Low','Medium','High'))
#Verify
str(ccdata$total_trans_amt)
#Verify no NAs
(sum(is.na(ccdata$total_trans_amt)))
```

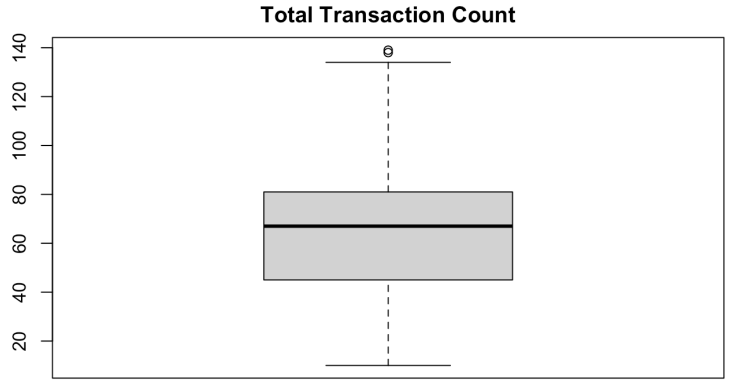
17: amt_change_q4_q1



17/20: amt_change_q4_q1
This is the change of dollar amount between Q4 over Q1.

```{r}
boxplot(ccdata$amt_change_q4_q1,main='Amt Change from Q4 to Q1')
min_amtchange <- min(ccdata$amt_change_q4_q1) -.01
max_amtchange <- max(ccdata$amt_change_q4_q1) + .01
bins_amtchange = 3
width_amtchange=(max_amtchange-min_amtchange)/bins_amtchange
#Factorization
ccdata$amt_change_q4_q1  = cut(ccdata$amt_change_q4_q1, breaks=seq(min_amtchange, max_amtchange,
width_amtchange),
                    labels=c('Low','Medium','High'))
#Verify factorization
str(ccdata$amt_change_q4_q1)

#Verify no NA's
sum(is.na(ccdata$amt_change_q4_q1))
```

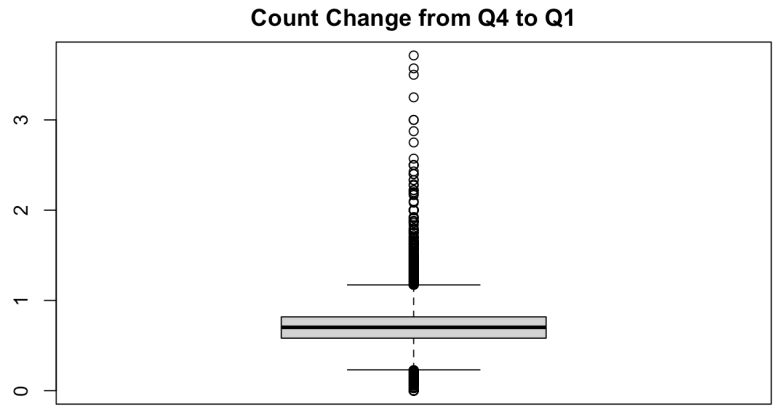
18: total_trans_cnt



18/20: total_trans_cnt
This is the total number of tranaactions in the last 12 months.

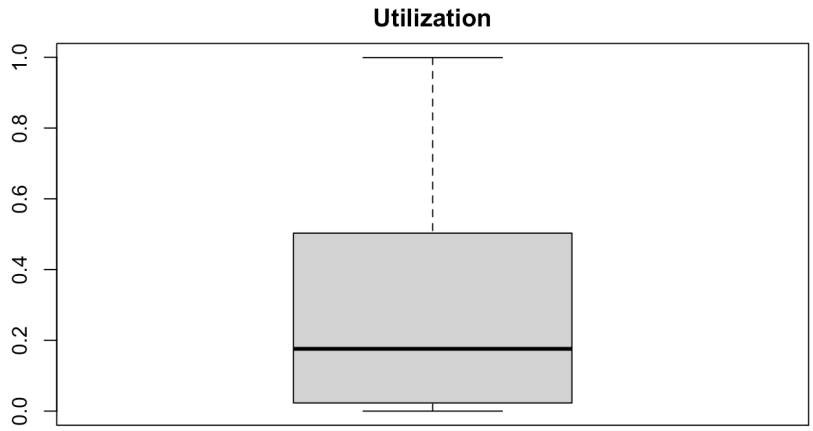
```{r}
boxplot(ccdata$total_trans_cnt,main='Total Transaction Count')
#Factorization,'
ccdata$total_trans_cnt = cut(ccdata$total_trans_cnt, breaks = c(0,20,40,60,80,100,120,140,Inf),
                       labels=c('Under 20','20s','40s','60s','80s','100s','120s','140s'))
#Verify
str(ccdata$total_trans_cnt)
#Verify no NA's
sum(is.na(ccdata$total_trans_cnt))
```

19: cnt_change_q4_q1




```
#### 19/20: cnt_change_q4_q1
This is the change of transaction count between Q4 over Q1.
```{r}
boxplot(ccdata$cnt_change_q4_q1,main='Count Change from Q4 to Q1')
min_cntchange = min(ccdata$cnt_change_q4_q1) -.01
max_cntchange = max(ccdata$cnt_change_q4_q1) + .01
bins_cntchange = 3
width_cntchange=(max_cntchange-min_cntchange)/bins_cntchange;
#Factorization
ccdata$cnt_change_q4_q1 = cut(ccdata$cnt_change_q4_q1, breaks=seq(min_cntchange, max_cntchange,
width_cntchange),
                      labels=c('Low','Medium','High'))
str(ccdata$cnt_change_q4_q1)
#Verify no NA's
sum(is.na(ccdata$cnt_change_q4_q1))
```
```

20: utilization



```
#### 20/20: utilization
```

```{r}
boxplot(ccdata$utilization,main='Utilization')
min_utilization <- min(ccdata$utilization) -.01
max_utilization <- max(ccdata$utilization) + .01
bins_utilization = 3
width_utilization=(max_utilization-min_utilization)/bins_utilization;
#Factorize
ccdata$utilization  = cut(ccdata$utilization, breaks=seq(min_utilization, max_utilization,
width_utilization),
                      labels=c('Low','Medium','High'))
#Verify
str(ccdata$utilization)
#Verify no NA's
sum(is.na(ccdata$utilization))
```

INITIAL EXPLORATORY DATA ANALYSIS

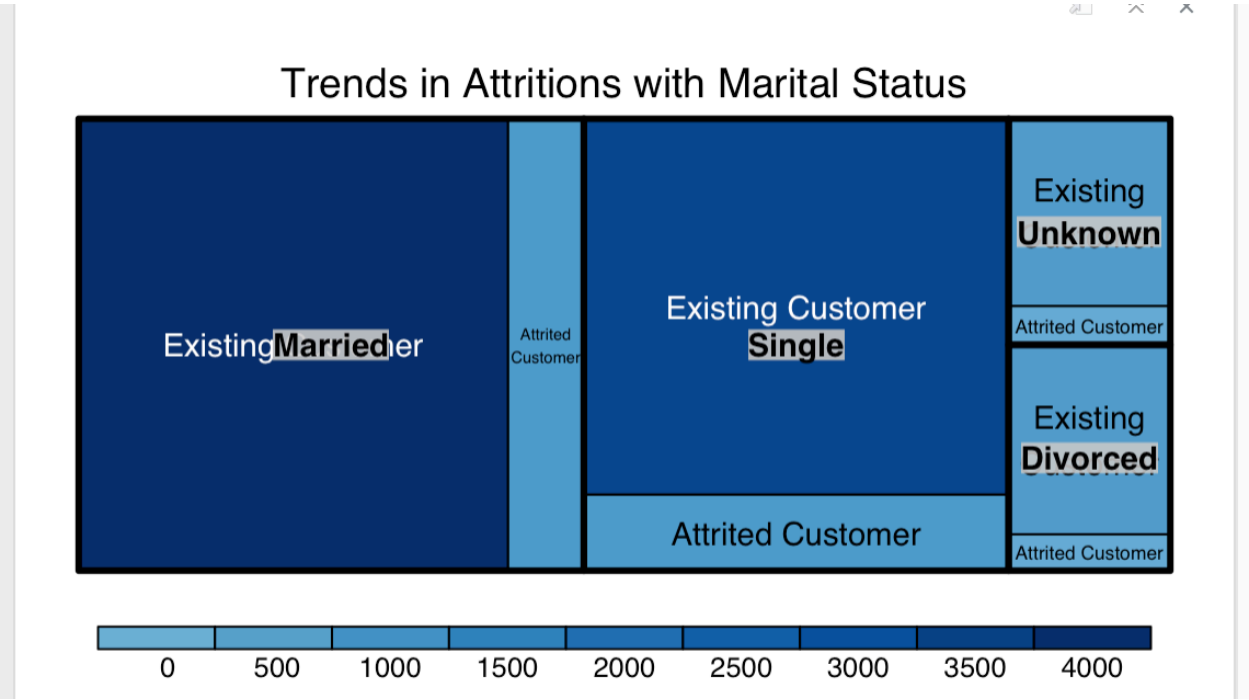
TREEMAP TRENDS

Treemap displays hierarchical data as a set of nested rectangles. Each group is represented by a rectangle, which area is proportional to its value.

Attrition Trend is explored with some key data fields like Marital Status and Education.

a) Trends in attrition with Marital Status and Education:

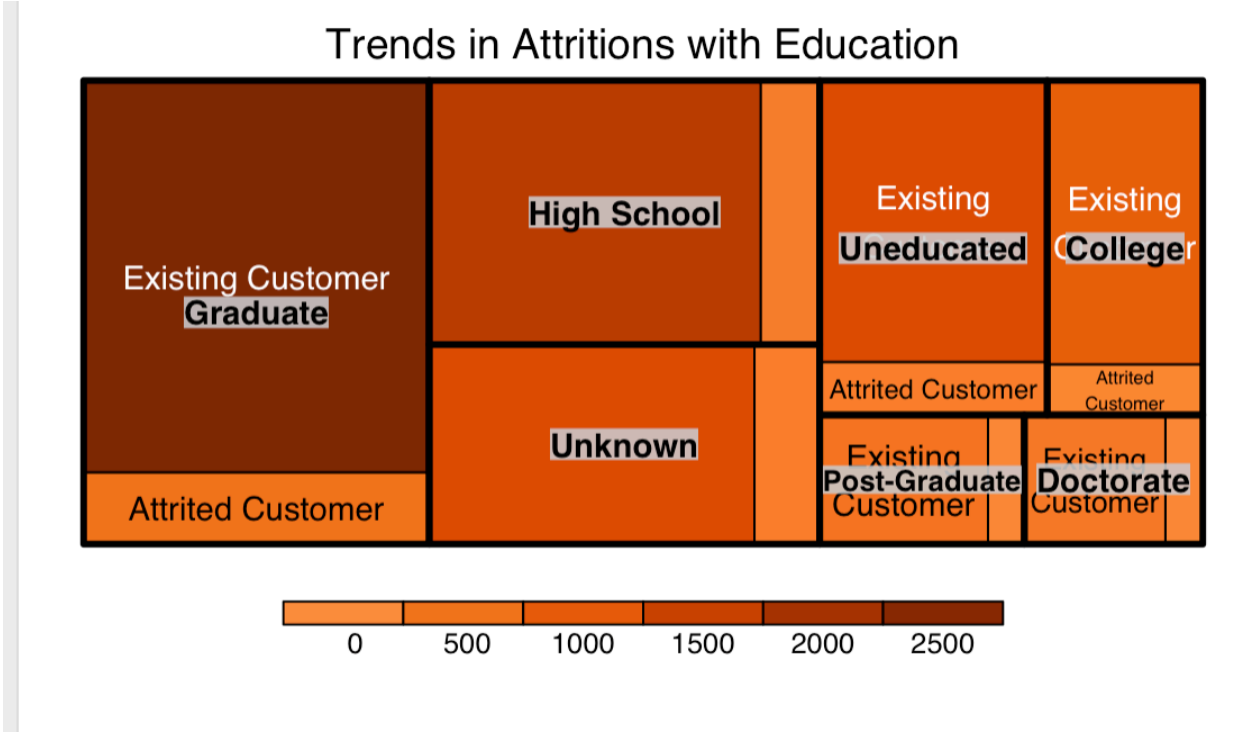
```{r}
treemap(ccdata,
        index=c("marital","attrition_flag"),
        vSize = "CLIENTNUM",
        type="value",
        palette = "Blues",
        title="Trends in Attritions with Marital Status",
        fontsize.title = 14
        )

```



Based on the trend shown above in general our dataset contains a lesser percentage of Attrited Customers and customers who are single have a visually higher attrition rate.

b) Trends in Attritions with Education:

```
treemap(ccdata,
        index=c("education","attrition_flag"),
        vSize = "CLIENTNUM",
        type="value",
        palette = "Oranges",
        title="Trends in Attritions with Education",
        fontsize.title = 14
        )
```



CORRELATION MATRIX

The correlation matrix below converted all the factors in the data set and set them as numeric. Then the p-values were derived from the correlation matrix. The visual below is showing only the lower half of the correlation matrix since the top half would be repeated values. Also, insignificant p-values or correlations were left as blank.

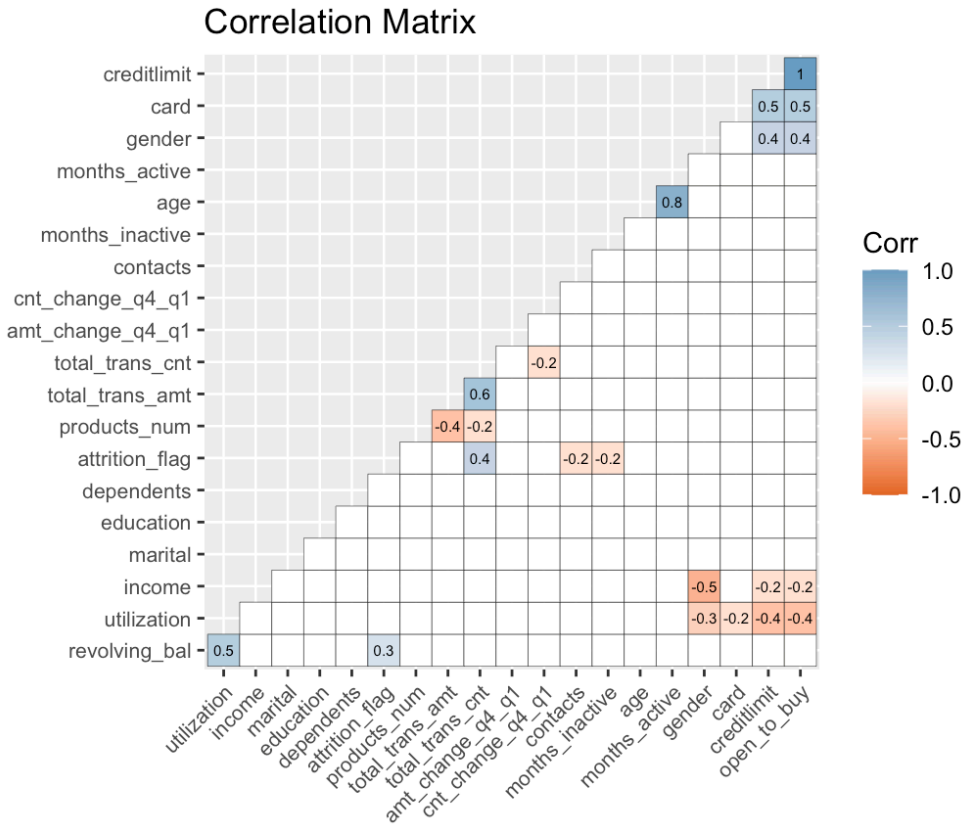
A positive correlation is shown in blue and negative correlations are colored in red. The number displayed in the center of the boxes are the correlation p-values. The strongest positive correlations are:

- age & months active: this makes sense considering the older you become, the longer you are retained at the bank.
- credit limit & the amount open to buy: the higher the customer's credit limit, the more they have open to buy.

- total transaction amount & total transaction count: seemingly, transaction amount and count go hand in hand.

The strongest negative correlations are:

- gender & income: this may be irrelevant as the correlation p-value is -0.5 and there are only two potential genders.
- -pen to buy & utilization: the higher the utilization, the less the customer has open to buy.
- credit limit & utilization: the higher the utilization, the less their credit limit may be.
- number of products & total transaction amount

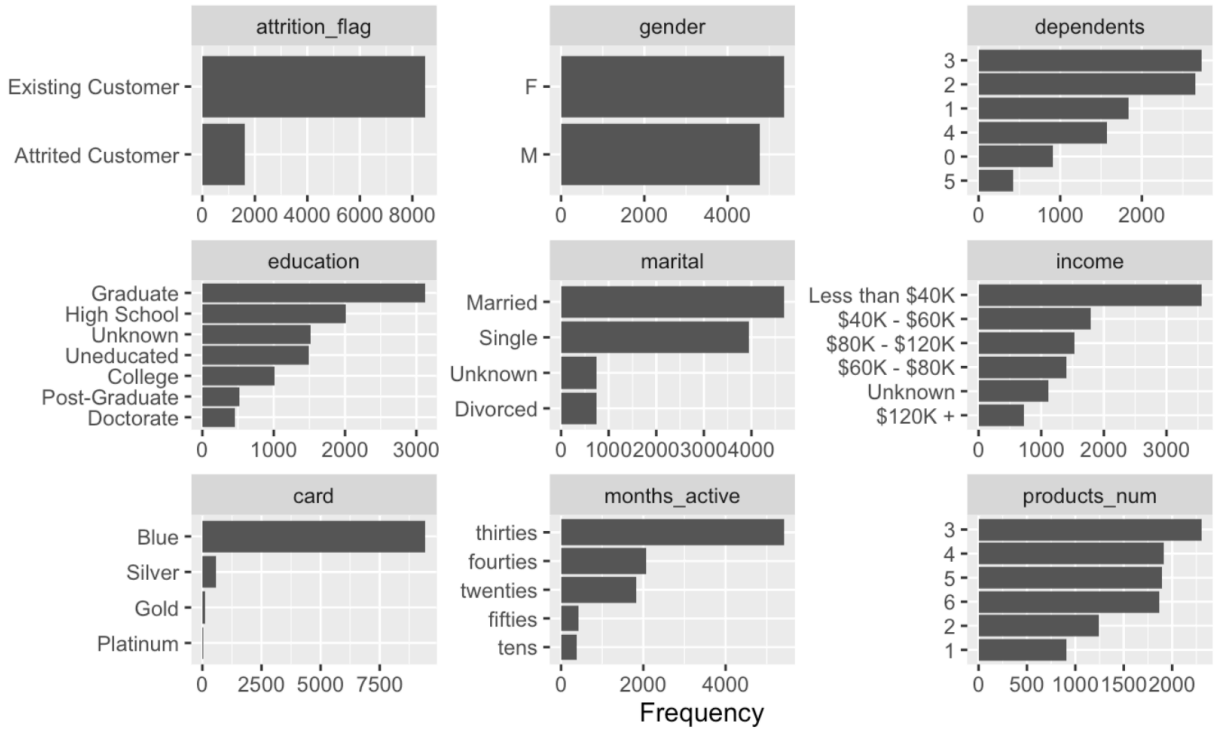


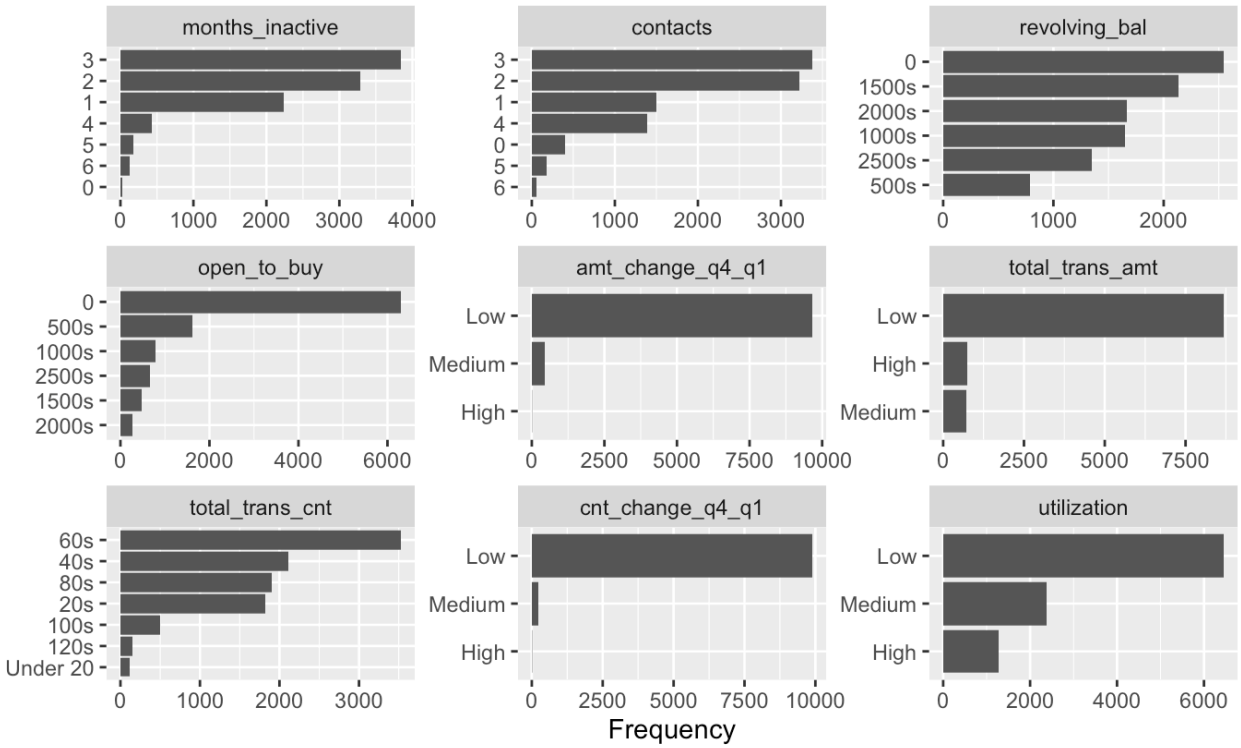
```r
corr = lapply(ccdata,as.numeric)
corr = data.frame(corr)
corrmatrix = round(cor(corr), 1)
p.mat = cor_pmat(corrmatrix)

ggcorrplot(corrmatrix, hc.order = TRUE, type = 'lower',
           outline.col = 'black',
           ggtheme = ggplot2::theme_gray,
           lab = TRUE,
           title= 'Correlation Matrix',
           show.legend =TRUE,
           lab_size = 2,
           tl.cex = 8,
           p.mat=p.mat,insig='blank',
           colors=c("#E4672G", "white", "#6D9EC1"))
```

DATA PLOT

After the data cleaning, the data is plotted to see the frequency and dispersion of data.

We can see most of our data set contains current customers.





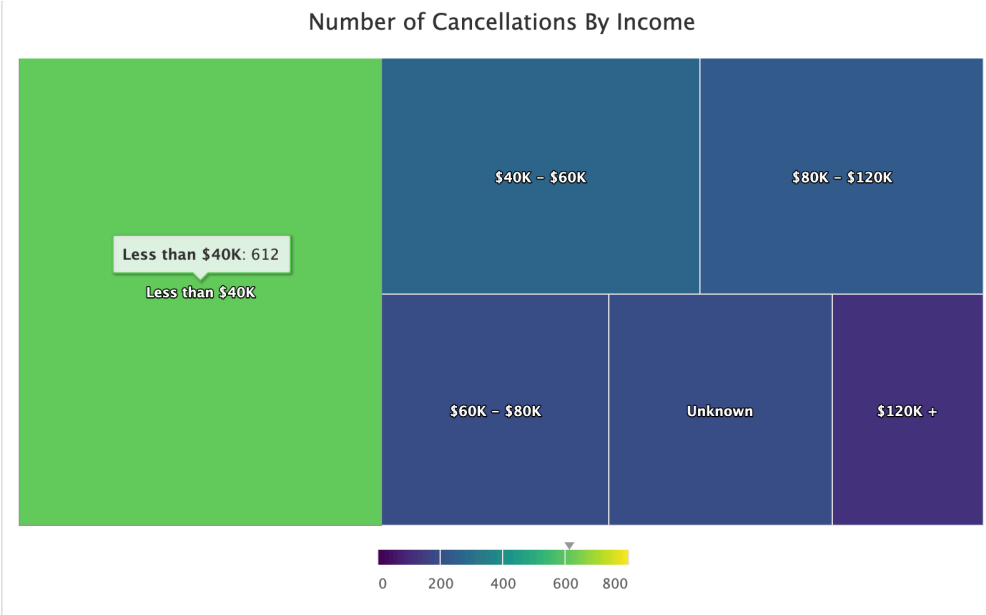
ATTRITION FLAG EXPLORATION

Looks like attrited customers have a total transaction amount cut off at around $11,000.


```{r}
hchart(
  ccdata_cor,
  "scatter",
  hcaes(x = creditlimit, y = total_trans_amt, group = attrition_flag)
)
```

CANCELLATIONS BY ATTRIBUTES

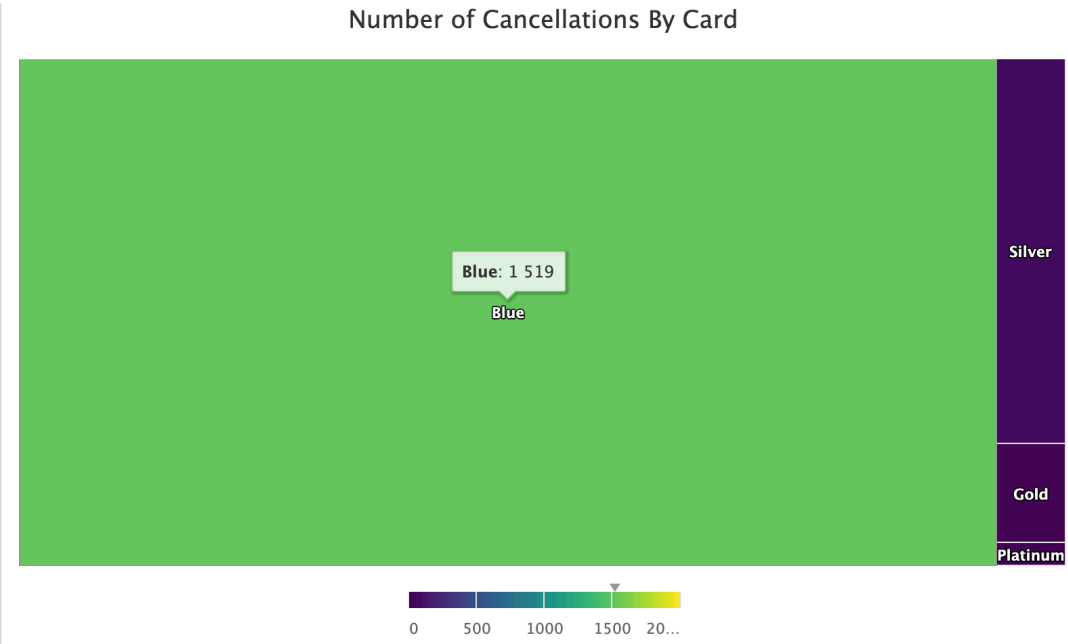
We can see a quick overview of attrited customers and their income levels.




```r
ccdata %>%
  filter(attrition_flag == "Attrited Customer") %>%
  count(income) %>%
  hchart("treemap", hcaes(x = income, value = n, color = n)) %>%
  hc_colorAxis(stops = color_stops(colors = viridis(11)))  %>%
  hc_title(text = "Number of Cancellations By Income", align = "center")
```

CANCELLATIONS BY INCOME

Blue cards are the most common card cancelled.




```r
ccdata %>%
  filter(attrition_flag == "Attrited Customer") %>%
  count(card) %>%
  hchart("treemap", hcaes(x = card, value = n, color = n)) %>%
  hc_colorAxis(stops = color_stops(colors = viridis(11)))  %>%
  hc_title(text = "Number of Cancellations By Card", align = "center")
```

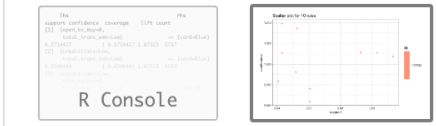
APRIORI

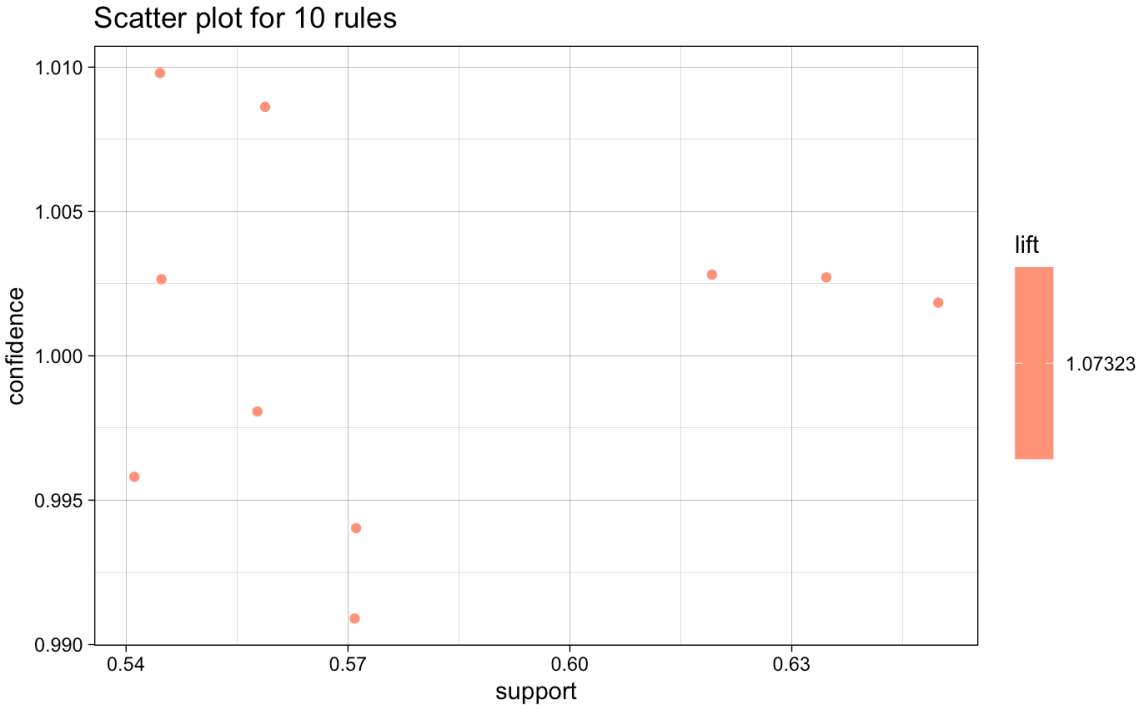
APRIORI Algorithm is used to generate frequent itemset from a given set of transactions. According to APRIORI Principle: 'If an itemset is frequent, then all of its subsets must also be frequent'. Algorithm based on this principle helps to support based pruning to systematically control the exponential growth of candidate itemset. Apriori are measured by support, confidence, and lift.

BLUE CARD

The confidence is set to at least 90% and support is set to at least 50%. The top ten rules are sorted by confidence. With the scatterplot between lift and support, we can see how the data is dispersed. At 90% confidence and 60% support, there are 4 data points at 100% confidence. the lowest confidence is roughly around 98%. The support is between 50-65%.

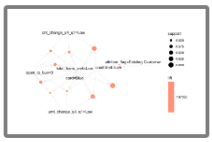
```{r}
bluecard = apriori(data=ccdata, parameter=list(supp=0.5,conf = 0.9),
               appearance = list(default="lhs",rhs="card=Blue"),
               control = list(verbose=F))
bluecard = sort(bluecard, decreasing=TRUE,by="confidence")
inspect(bluecard[1:10])
plot(bluecard[1:10], method='scatterplot',interactive=FALSE )
```

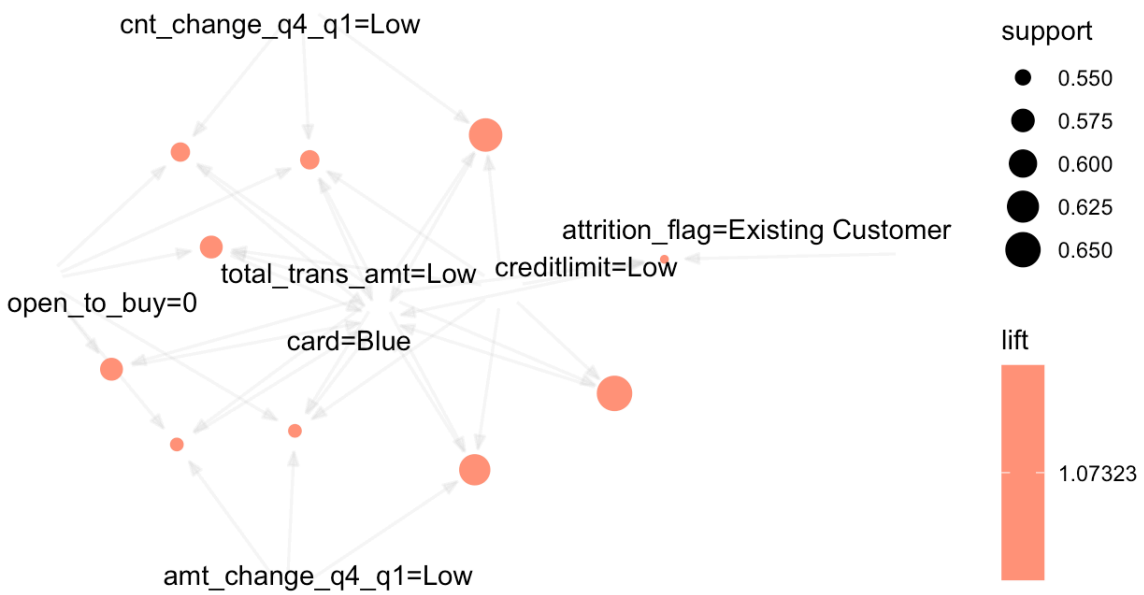




Now plotting the data, we can see the major factors of blue card customers. Customers are associated with a low credit limit with results in an open_to_buy of 0. This goes hand in hand since the customers' ability to make more purchases is close to 0 if their credit limit is already low. In this scenario, they are also likely to be an existing customer with a blue card.

```{r}
plot(bluecard[1:10], method='graph',interactive=FALSE )
```





EXISTING CUSTOMER

For all the top 10 rules for existing customers:

a) have a lift value > 1. This shows the variables present in the rules are positively correlated.

b) Confidence =1. This shows the high reliability of the rule.

c) Support = Support between 20-40% considered as good and all the rules are above that minimal threshold.

Some Additional Notes:

Based on the rules a customers tend to stay with the bank when

a) Transaction Amounts are high

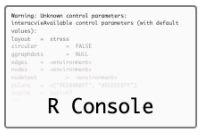
b) Transactions counts are higher like in the hundreds

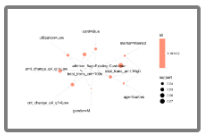
c) Combination of both a and b

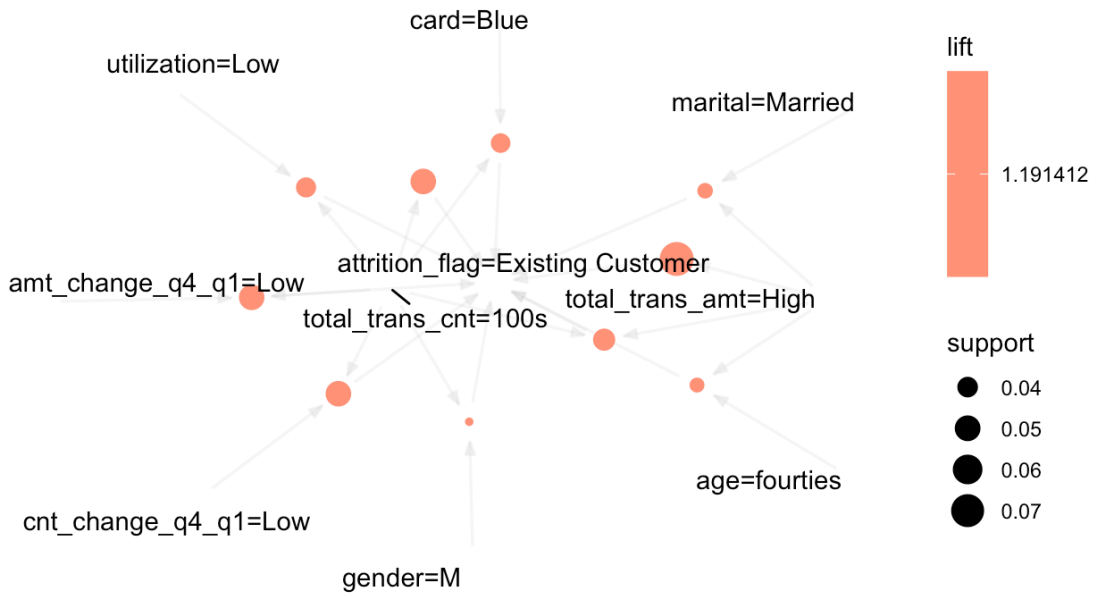
d) 'Male' customer with high transaction count

e) Customer age is in the forties.

```r
#visualization of rules
plot(existingcust[1:10],method="graph",interacvie=TRUE,shading="lift")
```







PAST CUSTOMER

For the top 10 rules on Attrited Customers:

a) Customers with lesser contacts

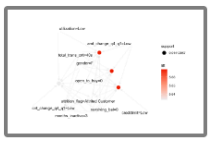
b) Customers with transaction counts on the lower side(forties)

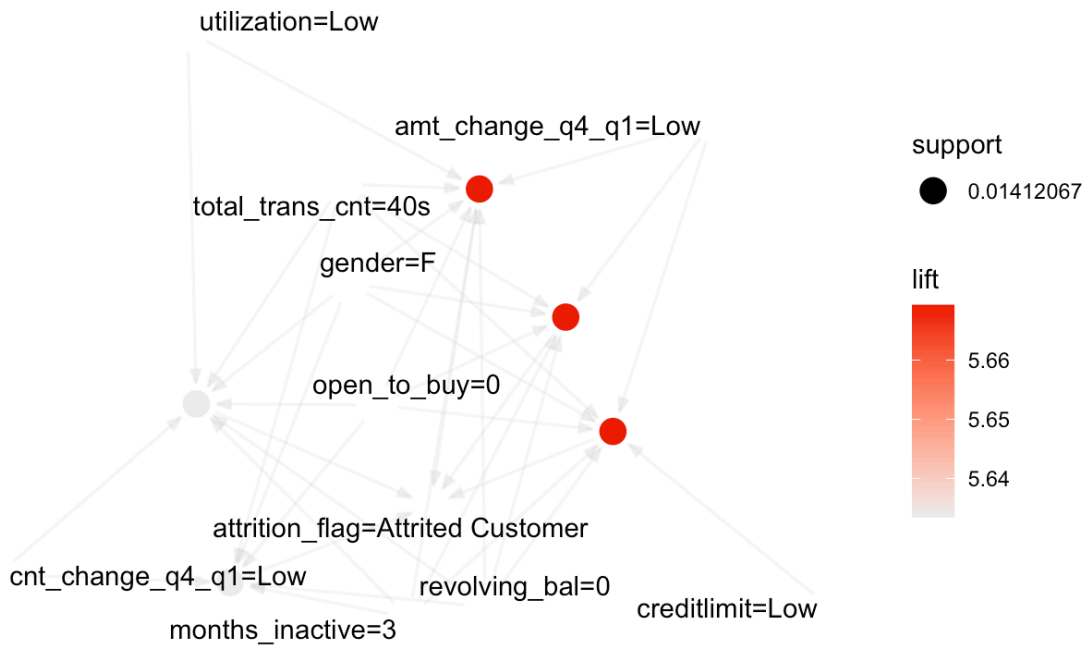
c)Customers relatively new (active months in the thirties)

d)Inactive Customers

These are agreeing to the common behavior of a customer when he/she want to stop using a card.

```{r}
#Visualizing the rules
plot(attritedcust[1:5],method="graph",interactive=FALSE,shading="lift")
```





MODELS

UNSUPERVISED MODELS:

ASSOCIATION RULE MINING USING APRIORI:

Association Rule Mining is the process of discovering useful and valuable rules from large amounts of data in various applications. This process can help organizations extract information from their data and identify patterns, relationships, and dependencies that can be used to make informed decisions, improve business processes, and optimize operations. Some of the key terms used in ARM analysis are briefly explained below:

- Support: Measures the frequency of the rule in the data. Higher support value indicates that the rule is more frequent and therefore more relevant.

Typical range for support is 20-40%.

- Confidence: Confidence is an estimate of conditional probability of transactions from the left-hand side of the rule when transactions on the right- hand side of the rule happens . It measures the reliability of the rule and higher values indicates that the rule is more likely to be true.

A confidence value of 0.9 is considered as good.

- Lift: Measures the correlation of associated items in a rule.

Lift Value of >1 is considered positive correlation and <1 implies negative correlation. Value of 1 suggests that the items under consideration are independent of each other. The above terms and the range mentioned here will decide the outcome of ARM analysis.

K-MEANS CLUSTERING ANALYSIS:

Loading and pre-processing of Data:

```r
kmeans_ccdata<-ccdata
str(kmeans_ccdata)
```

```r
#Pre-processing data - Converting factors to Numeric
kmeans_ccdata$age<-as.numeric(kmeans_ccdata$age)
kmeans_ccdata$gender<-as.numeric(kmeans_ccdata$gender)
kmeans_ccdata$dependents<-as.numeric(kmeans_ccdata$dependents)
kmeans_ccdata$education<-as.numeric(kmeans_ccdata$education)
kmeans_ccdata$marital<-as.numeric(kmeans_ccdata$marital)
kmeans_ccdata$income<-as.numeric(kmeans_ccdata$income)
kmeans_ccdata$card<-as.numeric(kmeans_ccdata$card)
kmeans_ccdata$months_active<-as.numeric(kmeans_ccdata$months_active)
kmeans_ccdata$products_num<-as.numeric(kmeans_ccdata$products_num)
kmeans_ccdata$months_inactive<-as.numeric(kmeans_ccdata$months_inactive)
kmeans_ccdata$contacts<-as.numeric(kmeans_ccdata$contacts)
kmeans_ccdata$creditlimit<-as.numeric(kmeans_ccdata$creditlimit)
kmeans_ccdata$revolving_bal<-as.numeric(kmeans_ccdata$revolving_bal)
kmeans_ccdata$open_to_buy<-as.numeric(kmeans_ccdata$open_to_buy)
kmeans_ccdata$amt_change_q4_q1<-as.numeric(kmeans_ccdata$amt_change_q4_q1)
kmeans_ccdata$total_trans_amt<-as.numeric(kmeans_ccdata$total_trans_amt)
kmeans_ccdata$total_trans_cnt<-as.numeric(kmeans_ccdata$total_trans_cnt)
kmeans_ccdata$cnt_change_q4_q1<-as.numeric(kmeans_ccdata$cnt_change_q4_q1)
kmeans_ccdata$utilization<-as.numeric(kmeans_ccdata$utilization)
```

Adding the unique identifier Client Number back to the data to keep it as row names.

#Adding Client Number back to the data and keeping it as row name

```{r}
kmeans_ccdata<-cbind(ccdata_cor$CLIENTNUM,kmeans_ccdata)
```

```{r}
head(kmeans_ccdata)
```

Description: df [6 × 21]

	ccdata_cor$CLIENTNUM <int>	attrition_flag <fctr>	age <dbl>	gender <dbl>	dependents <dbl>
1	768805383	Existing Customer	3	2	4
2	818770008	Existing Customer	3	1	6
3	713982108	Existing Customer	4	2	4
4	769911858	Existing Customer	2	1	5
5	709106358	Existing Customer	2	2	4
6	713061558	Existing Customer	3	2	3

6 rows | 1-6 of 21 columns

Adding Client Number as Row Name

```{r}
rownames(kmeans_ccdata)<-kmeans_ccdata[,1]
kmeans_ccdata[,1]<-NULL
```

Adding Client Number as Row Name

```{r}
rownames(kmeans_ccdata)<-kmeans_ccdata[,1]
kmeans_ccdata[,1]<-NULL
```

```{r}
head(kmeans_ccdata)
```

Description: df [6 × 20]

	attrition_flag <fctr>	age <dbl>	gender <dbl>	dependents <dbl>	education <dbl>
768805383	Existing Customer	3	2	4	4
818770008	Existing Customer	3	1	6	3
713982108	Existing Customer	4	2	4	3
769911858	Existing Customer	2	1	5	4
709106358	Existing Customer	2	2	4	6
713061558	Existing Customer	3	2	3	3

6 rows | 1-6 of 20 columns

```
#taking a backup and removing attrition_flag
```{r}
kmeans_ccdata_bkup<-kmeans_ccdata
```

```{r}
kmeans_ccdata$attrition_flag<-NULL
```
```

Attrition Flag has two possible values. In the first clustering attempt, k value is supplied with 2.

```
# Executing k-means with k=2
```{r}
set.seed(50)
kcluster<-kmeans(kmeans_ccdata,2)
kmeans_ccdata$kcluster<-as.factor(kcluster$cluster)
```
```

Exploring the cluster details:

```{r}
str(kcluster)
kcluster$centers
```

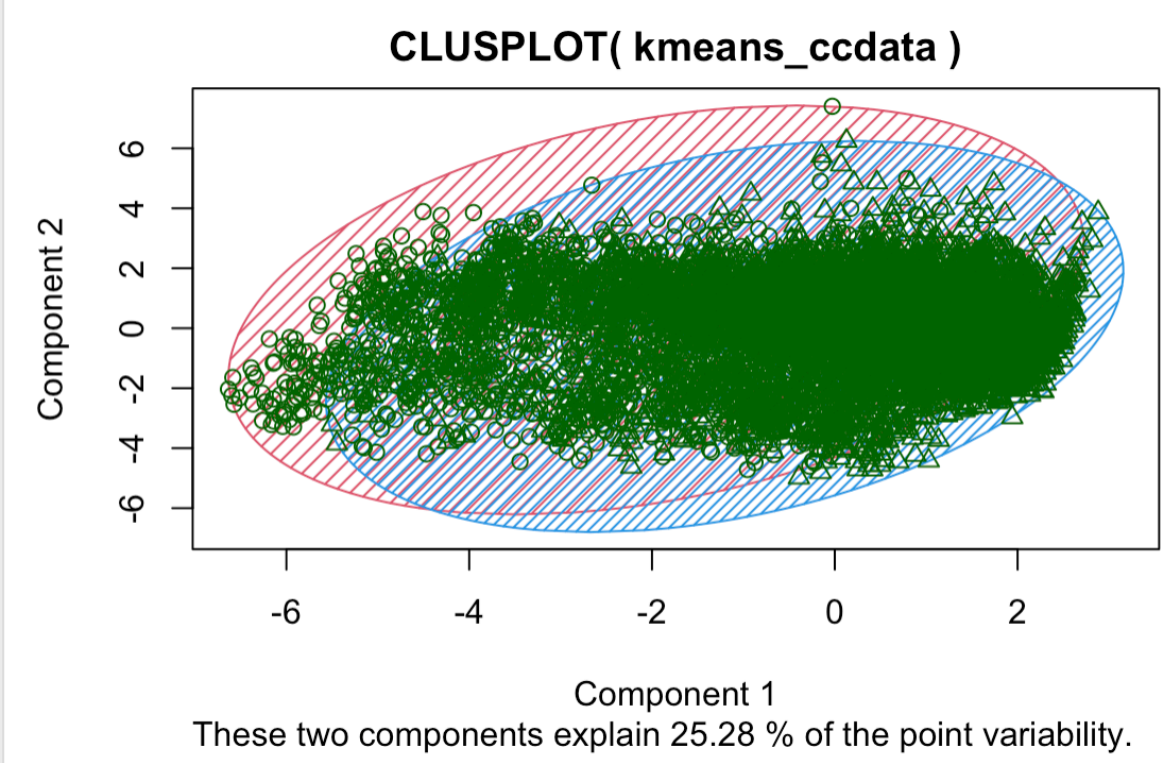
```
List of 9
 $ cluster     : Named int [1:10127] 1 2 2 2 1 2 1 1 1 2 ...
  ..- attr(*, "names")= chr [1:10127] "768805383" "818770008" "713982108"
"769911858" ...
 $ centers     : num [1:2, 1:19] 3.08 3.08 1.54 1.42 3.41 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "1" "2"
  .. ..$ : chr [1:19] "age" "gender" "dependents" "education" ...
 $ totss       : num 234060
 $ withinss    : num [1:2] 96906 112051
 $ tot.withinss: num 208956
 $ betweenss   : num 25104
 $ size        : int [1:2] 4144 5983
 $ iter        : int 1
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
       age   gender dependents education  marital   income     card
1 3.077220 1.537403   3.412162  5.870656 2.484073 3.711149 1.307432
2 3.083904 1.424870   3.300518  2.867792 2.449106 3.969748 1.091426
  months_active products_num months_inactive contacts creditlimit
1      3.023407     3.748069        3.333012 3.473697    1.830598
2      3.033595     3.857262        3.346816 3.442587    1.285308
  revolving_bal open_to_buy amt_change_q4_q1 total_trans_amt
1      3.370415    2.402510         1.044884        1.277268
2      3.350159    1.540532         1.047301        1.171987
  total_trans_cnt cnt_change_q4_q1 utilization
1        3.791506         1.024373    1.405647
2        3.683604         1.023400    1.547217
```

From the above, it can be noted that the two clusters were of the size 4144 and 5983 respectively. The cluster centers were across the variables age, gender, dependents, and education.

Plotting k-Clusters:

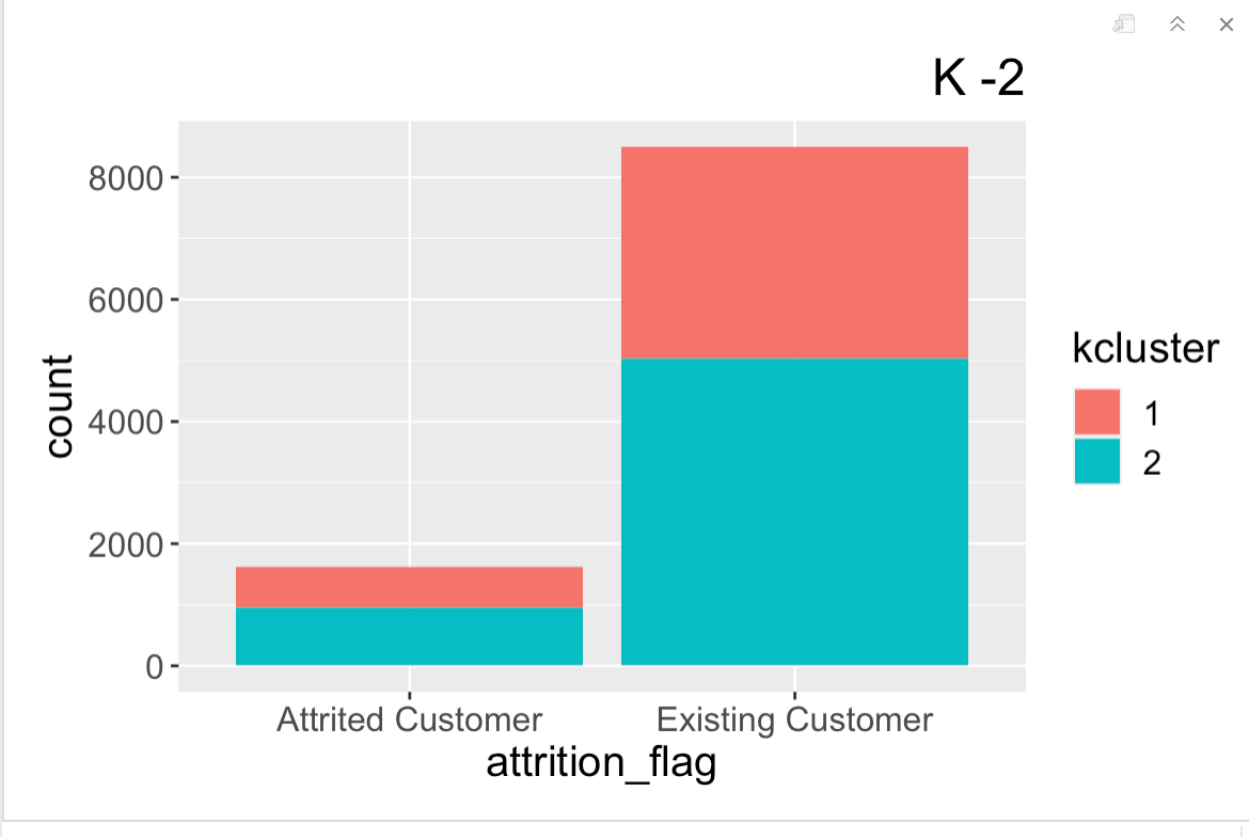
```r
kmeans_ccdata1<-kmeans_ccdata_bkup
kmeans_ccdata1$kcluster<-as.factor(kcluster$cluster)
```

```r
clusplot(kmeans_ccdata,kmeans_ccdata$kcluster,
         color = TRUE,labels=0,lines = 0, shade = TRUE)
```



Only 25% of the variability is explained here.

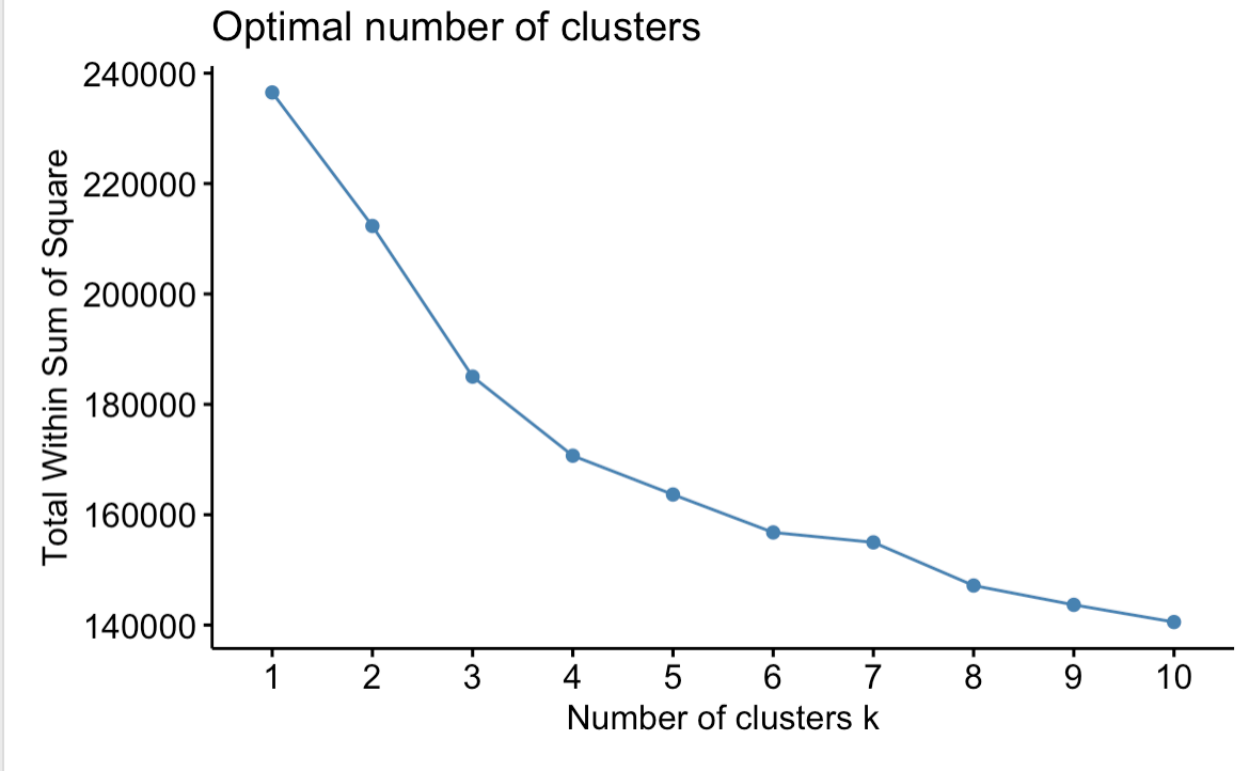
```r
ggplot(data=kmeans_ccdata1,aes(x=attrition_flag,fill=kcluster))+
         geom_bar(stat = "count")+
                  labs(title="K -2")+
                  theme(plot.title=element_text(hjust = 1),
                        text=element_text(size=15))
```



Elbow and Silhouette Methods were used to identify an optimal value for k.

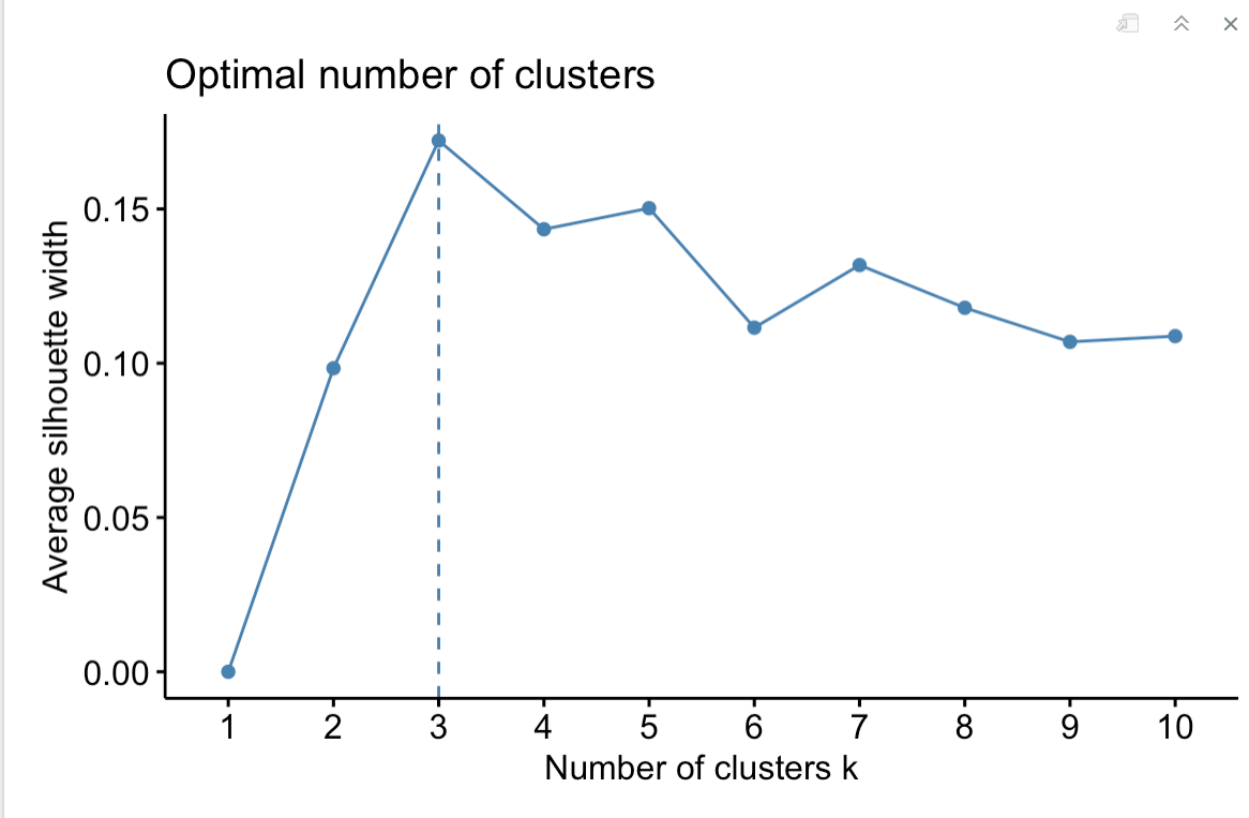
Elbow Method

```{r}
set.seed(123)

fviz_nbclust(kmeans_ccdata, kmeans, method = "wss")
```



Silhoutte Method

```{r}
set.seed(123)
fviz_nbclust(kmeans_ccdata, kmeans, method = "silhouette")
```



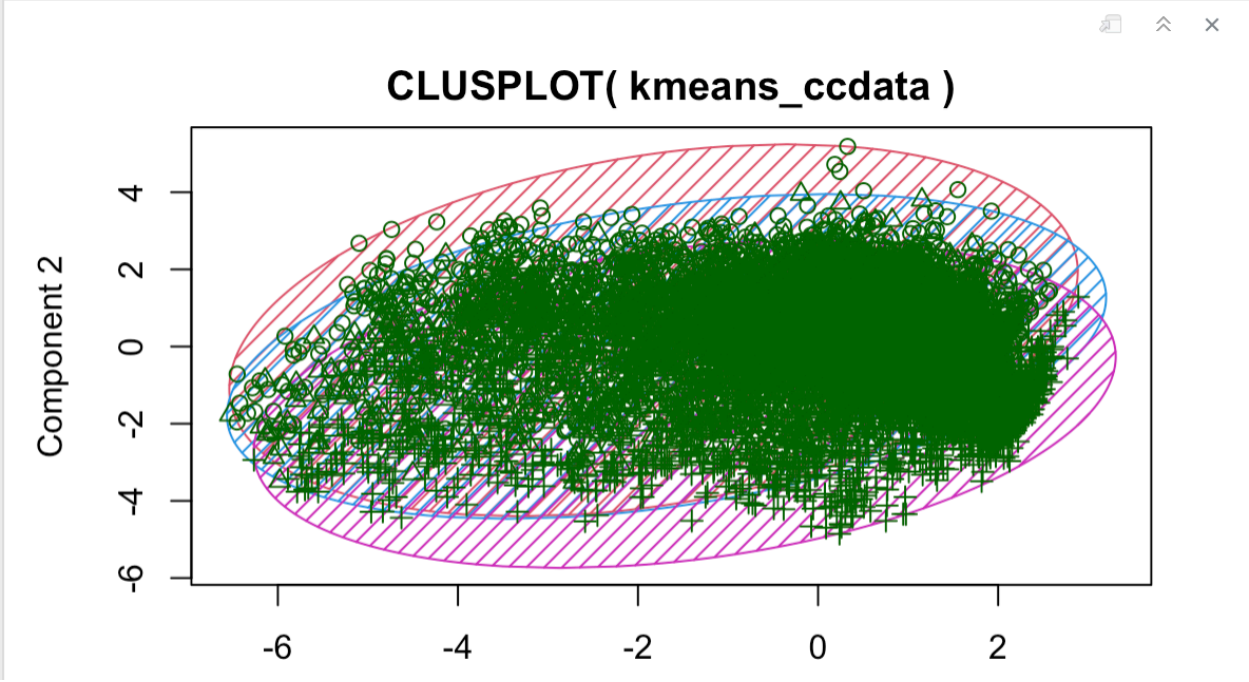
Both methods suggest an optimal k value of 3.

Next step is to re-run the model with k=3.

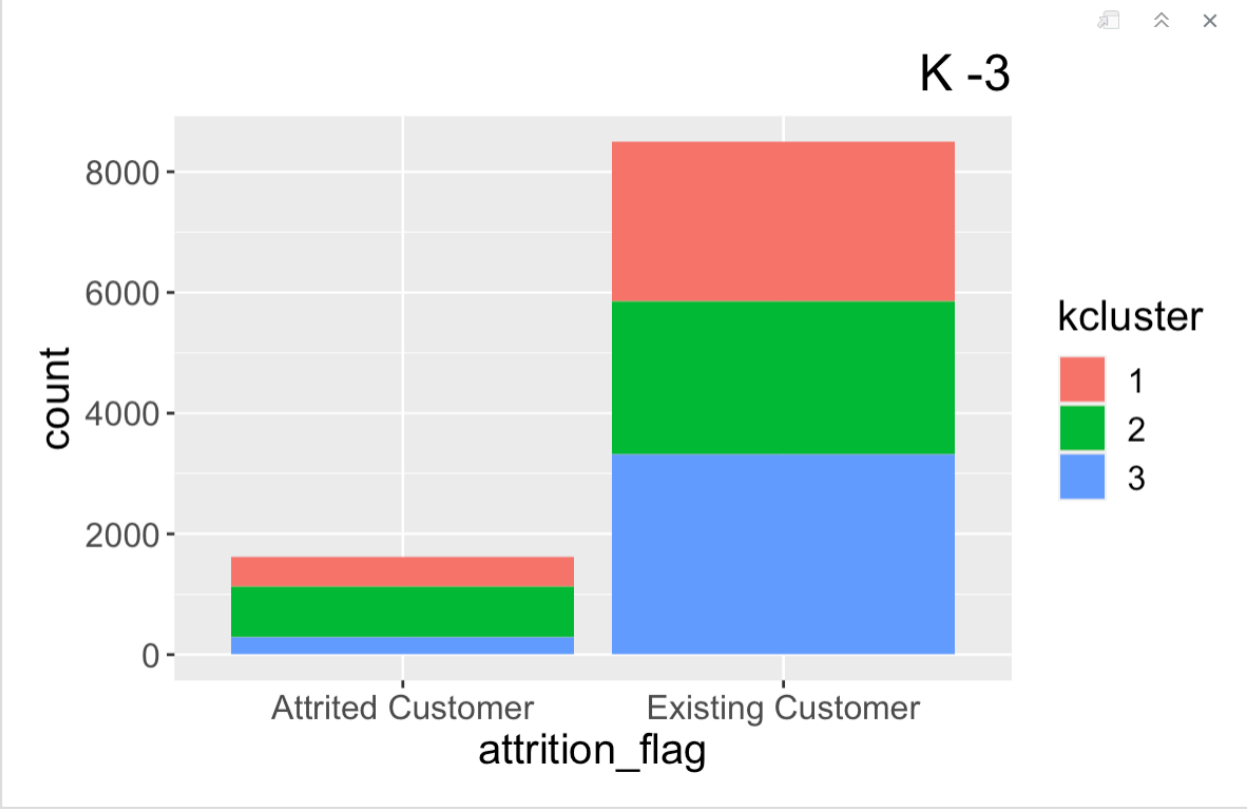
```r
set.seed(50)
kcluster<-kmeans(kmeans_ccdata,3)
kmeans_ccdata$kcluster<-as.factor(kcluster$cluster)
```

```r
kmeans_ccdata1<-kmeans_ccdata_bkup
kmeans_ccdata1$kcluster<-as.factor(kcluster$cluster)
```

```r
clusplot(kmeans_ccdata,kmeans_ccdata$kcluster,
         color = TRUE,labels=0,lines = 0, shade = TRUE)
```



```r
ggplot(data=kmeans_ccdata1,aes(x=attrition_flag,fill=kcluster))+
        geom_bar(stat = "count")+
                labs(title="K -3")+
                theme(plot.title=element_text(hjust = 1),
                      text=element_text(size=15))
```



The plot above shows the distribution of clusters, and it looks like variables influencing attrition data is equally spread between the clusters.

HIERARCHICAL AGGLOMERATIVE CLUSTERING (HAC) ANALYSIS

HAC approach is a collection of closely related clustering techniques that produces hierarchical clustering. HAC Clustering starts with points as individual clusters and at

each step the closest pair of clusters merge. The distance between two points is measured using a variety of techniques such as:

- Euclidean Distance
- Minkowski Distance
- Manhattan
- Binary
- Maximum
- Canberra

Loading and pre-processing of Data:

```{r}
ccdata_hac<-kmeans_ccdata_bkup
str(ccdata_hac)
```

```
'data.frame':   10127 obs. of  20 variables:
 $ attrition_flag  : Factor w/ 2 levels "Attrited Customer",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ age             : num  3 3 4 2 2 3 4 2 2 3 ...
 $ gender          : num  2 1 2 1 2 2 2 2 2 2 ...
 $ dependents      : num  4 6 4 5 4 3 5 1 4 3 ...
 $ education       : num  4 3 3 4 6 3 7 4 6 3 ...
 $ marital         : num  2 3 2 4 2 2 2 4 3 3 ...
 $ income          : num  3 5 4 5 3 2 1 3 3 4 ...
 $ card            : num  1 1 1 1 1 1 2 4 1 1 ...
 $ months_active   : num  3 4 3 3 2 3 4 2 3 3 ...
 $ products_num    : num  5 6 4 3 5 3 6 2 5 6 ...
 $ months_inactive : num  2 2 2 5 2 2 2 3 3 4 ...
 $ contacts        : num  4 3 1 2 1 3 4 3 1 4 ...
 $ creditlimit     : num  2 1 1 1 1 1 4 4 3 2 ...
 $ revolving_bal   : num  2 3 1 6 1 3 6 4 6 4 ...
 $ open_to_buy     : num  3 2 1 1 1 1 6 5 4 2 ...
 $ amt_change_q4_q1: num  2 2 3 2 2 2 2 2 3 2 ...
 $ total_trans_amt : num  1 1 1 1 1 1 1 1 1 1 ...
 $ total_trans_cnt : num  3 2 1 1 2 2 2 2 2 2 ...
 $ cnt_change_q4_q1: num  2 3 2 2 3 1 1 1 1 1 ...
```

Removing attrition flag from analysis:

```r
ccdata_hac1<-ccdata_hac[,c(2:20)]
str(ccdata_hac1)
```

```
'data.frame':   10127 obs. of  19 variables:
 $ age             : num  3 3 4 2 2 3 4 2 2 3 ...
 $ gender          : num  2 1 2 1 2 2 2 2 2 2 ...
 $ dependents      : num  4 6 4 5 4 3 5 1 4 3 ...
 $ education       : num  4 3 3 4 6 3 7 4 6 3 ...
 $ marital         : num  2 3 2 4 2 2 2 4 3 3 ...
 $ income          : num  3 5 4 5 3 2 1 3 3 4 ...
 $ card            : num  1 1 1 1 1 1 2 4 1 1 ...
 $ months_active   : num  3 4 3 3 2 3 4 2 3 3 ...
 $ products_num    : num  5 6 4 3 5 3 6 2 5 6 ...
 $ months_inactive : num  2 2 2 5 2 2 2 3 3 4 ...
 $ contacts        : num  4 3 1 2 1 3 4 3 1 4 ...
 $ creditlimit     : num  2 1 1 1 1 1 4 4 3 2 ...
 $ revolving_bal   : num  2 3 1 6 1 3 6 4 6 4 ...
 $ open_to_buy     : num  3 2 1 1 1 1 6 5 4 2 ...
 $ amt_change_q4_q1: num  2 2 3 2 2 2 2 2 3 2 ...
 $ total_trans_amt : num  1 1 1 1 1 1 1 1 1 1 ...
 $ total_trans_cnt : num  3 2 1 1 2 2 2 2 2 2 ...
 $ cnt_change_q4_q1: num  2 3 2 2 3 1 1 1 1 1 ...
 $ utilization     : num  1 1 1 3 1 1 1 1 1 1 ...
```

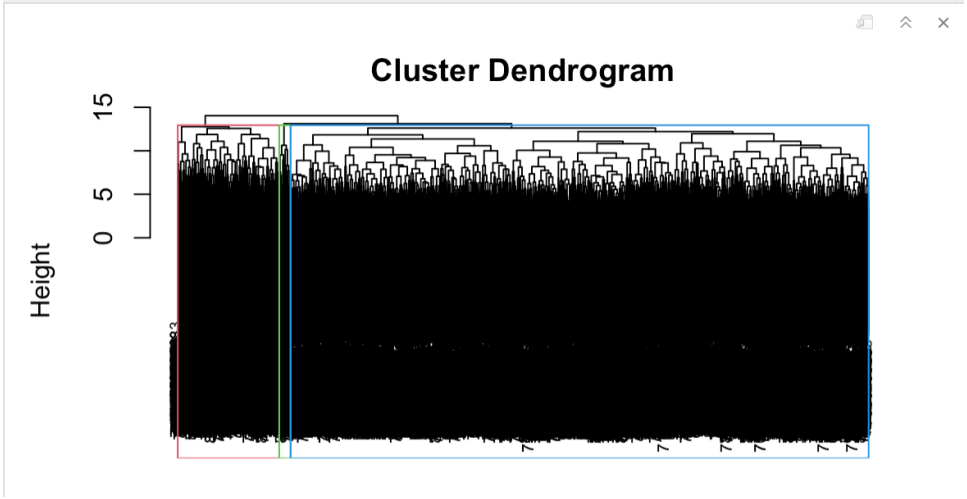
Chunk 152 R Markdown

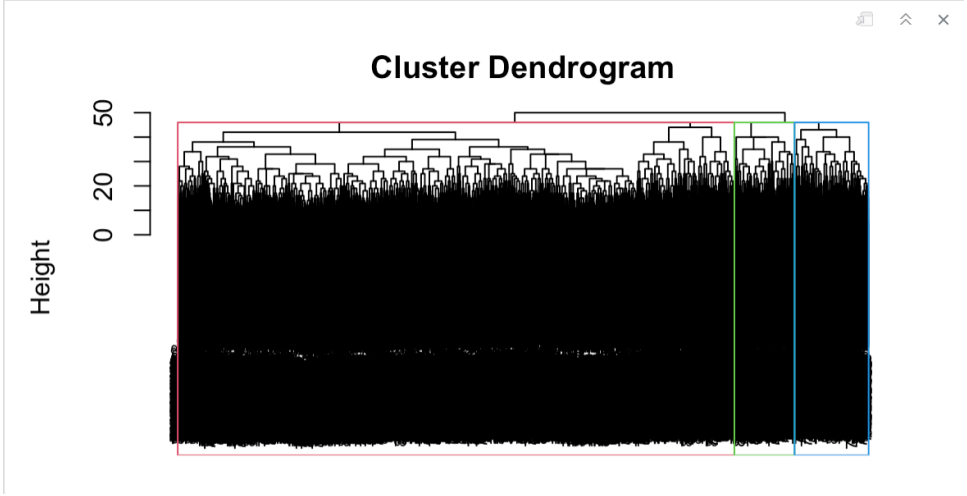
Calculating various distances:

```r
#calculating various distances:
```

```r
distance_eucl<-dist(ccdata_hac1,method='euclidean')
distance_man<-dist(ccdata_hac1,method='manhattan')
distance_mink<-dist(ccdata_hac1,method='minkowski')
distance_can<-dist(ccdata_hac1,method='canberra')
distance_max<-dist(ccdata_hac1,method='maximum')
distance_bin<-dist(ccdata_hac1,method='binary')
```

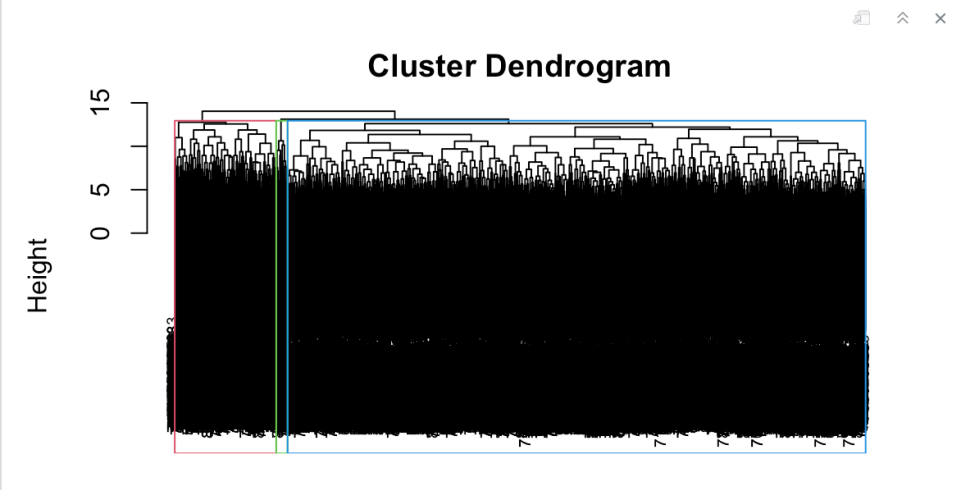
HAC with Euclidean Distance with k=3 (same value from k-means analysis):

```
#hac with euclidean distance
```{r}
hac_eucl=hclust(distance_eucl,method="complete")
plot(hac_eucl,cex=0.6,hang=1)
rect.hclust(hac_eucl,k=3,border=2:5)
```
```



HAC with Manhattan Distance with k=3 (same value from k-means analysis):

```
```{r}
hac_man=hclust(distance_man,method="complete")
plot(hac_man,cex=0.6,hang=1)
rect.hclust(hac_man,k=3,border=2:5)
```
```



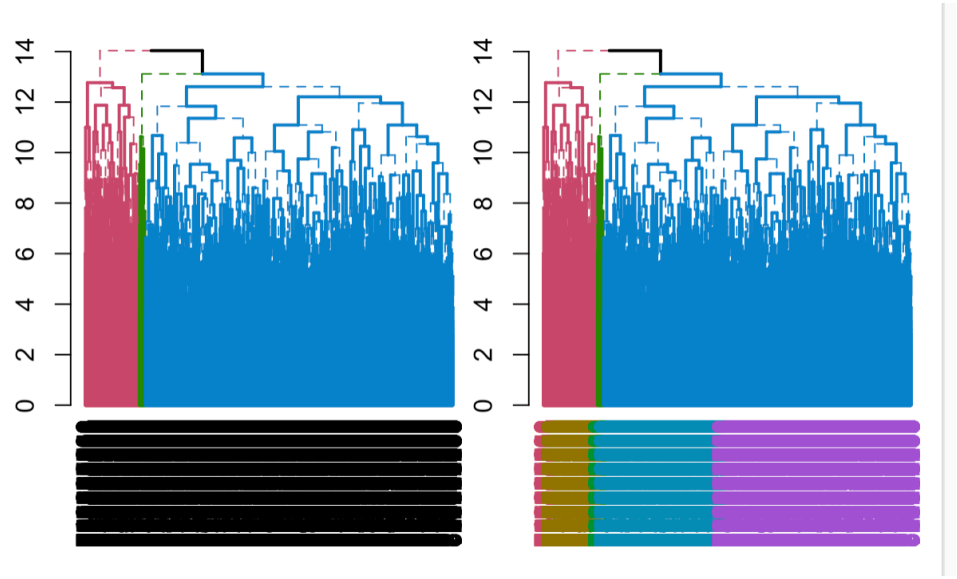
HAC with Minkowski Distance with k=3 (same value from k-means analysis):

```r
hac_mink=hclust(distance_mink,method="complete")
plot(hac_mink,cex=0.6,hang=1)
rect.hclust(hac_mink,k=3,border=2:5)
```

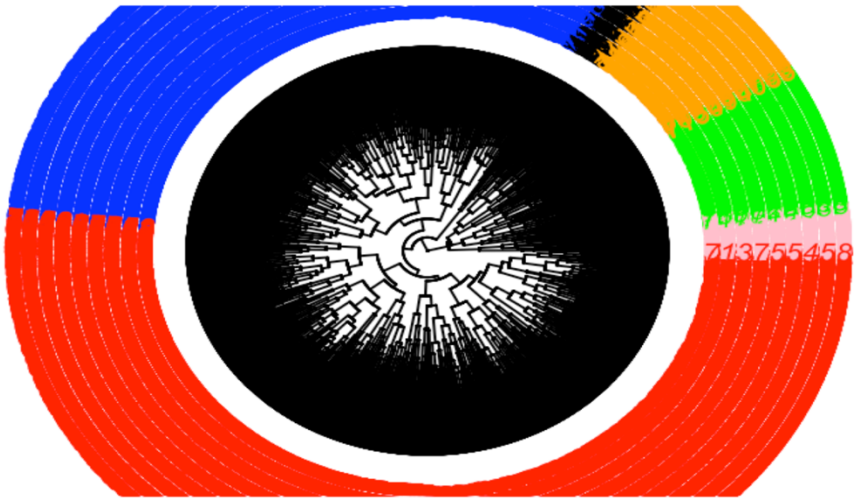
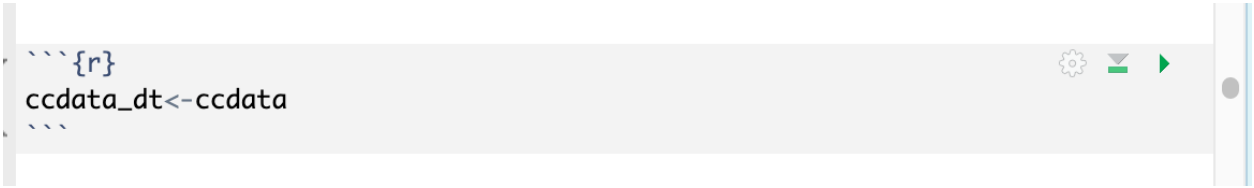


Exploring Dendrogram:

```r
dend <- as.dendrogram(hac_mink)

library(dendextend)
par(mfrow = c(1,2), mar = c(5,2,1,0))
dend <- dend %>%
         color_branches(k = 3) %>%
         set("branches_lwd", c(2,1,2)) %>%
         set("branches_lty", c(1,2,1))

plot(dend)

dend <- color_labels(dend, k = 5)
# The same as:
# labels_colors(dend)  <- get_leaves_branches_col(dend)
plot(dend)
```



Fan Dendrogram:

```r
```{r}
#install.packages("ape")
library("ape")
#plot(as.phylo(hac_mink), type = "fan")
# Cut the dendrogram into 4 clusters
colors = c("red", "blue", "green", "black","orange","pink")
clus4 = cutree(hac_mink, 6)
plot(as.phylo(hac_mink),type="fan", tip.color = colors[clus4],
     label.offset = 1, cex = 0.7)
```
```



The above analysis also points to the fact the clusters are influenced by multiple variables.

SUPERVISED MODELS

DECISION TREE ANALYSIS

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

Decision Trees can be built unpruned or pruned. Pruning is a data compression technique in machine learning and search algorithms that reduces the size of decision trees by removing sections of the tree that are non-critical and redundant to classify instances. Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

Loading the dataset for decision tree analysis.



For the analysis train and test datasets needs to be created. A split of 70-30% is applied to create the train and test datasets.

```
#split the data 70-30%
```{r}
split1<- sample(c(rep(0, 0.7 * nrow(ccdata_dt)), rep(1, 0.3 *
nrow(ccdata_dt))))
```

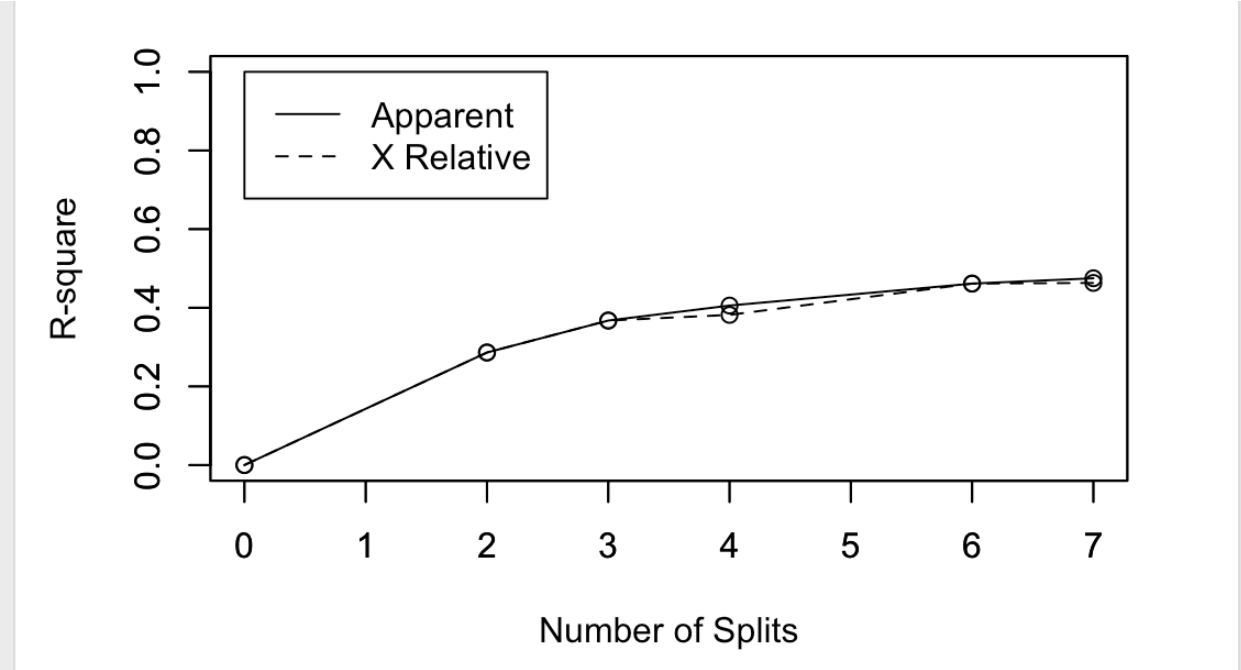
```{r}
table(split1)
```

split1
   0   1
7088 3038

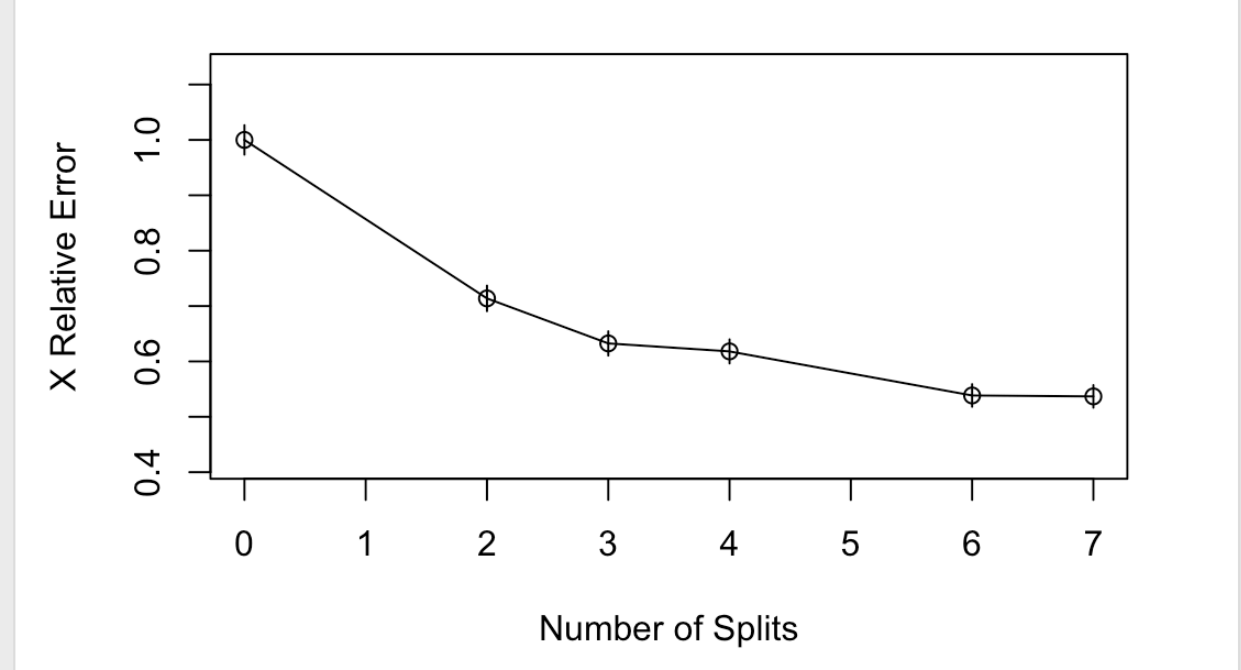
```{r}
train <- ccdata_dt[split1 == 0, ]
test <- ccdata_dt[split1== 1, ]
```
```

UNPRUNED DECISION TREE

First attempt is to train a decision tree as unpruned. The train outcome will give an insight to the complexity of the data.

```
```{r}
tree_model1 <- rpart( attrition_flag ~ . , data = train
, method = 'class'
#, control = rpart.control(minbucket = 1, minsplit=1, cp=-1)
, model = T
)

rsq.rpart(tree_model1)
```
```

R-Square Plot:



Relative Error plot:



The downward trending of Relative Error plot indicates the complexity of the tree and makes it a good candidate for pruning.

Based on the Decision Tree Summary:

```r
summary(tree_model1)
```

```
Call:
rpart(formula = attrition_flag ~ ., data = train, method = "class",
    model = T)
  n= 7089

          CP nsplit rel error    xerror       xstd
1 0.14216973      0 1.0000000 1.0000000 0.02708924
2 0.08573928      2 0.7156605 0.7156605 0.02353457
3 0.03499563      3 0.6299213 0.6299213 0.02225170
4 0.03018373      4 0.5949256 0.6185477 0.02207239
5 0.01487314      6 0.5345582 0.5450569 0.02085562
6 0.01000000      7 0.5196850 0.5301837 0.02059613

Variable importance
 total_trans_cnt    revolving_bal     products_num  total_trans_amt
              37               31               14                6
 months_inactive         contacts amt_change_q4_q1 cnt_change_q4_q1
               4                3                2                1
             age    months_active
               1                1
```

The variables of Importance are:

a) Total Transactions Count

b) Revolving Balance

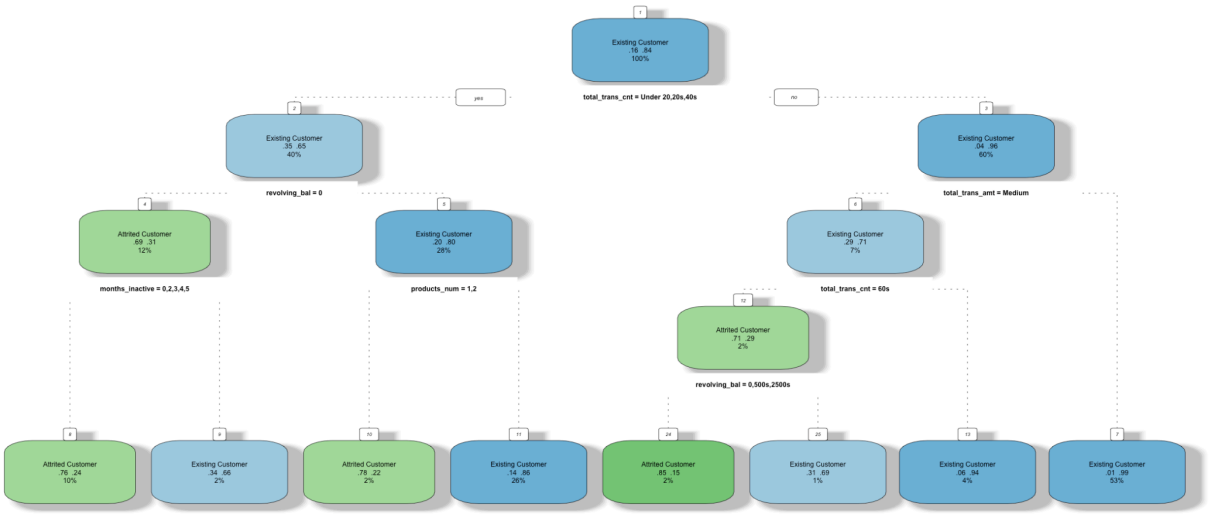
c) Number of Products

d) Total Transactions Amount

e) Inactive Months

f) Contacts

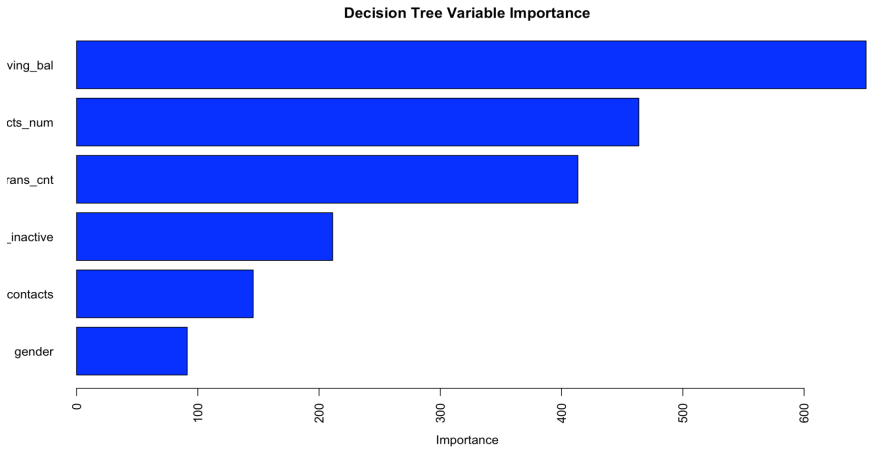
A plot of Decision Tree is as shown below:



Rattle 2023-Mar-19 19:13:39 renjinirajan

A frequency plot of Variables of Importance is as shown below:

```{r}
set.seed(500)
imp <- varImp(tree_model1, scale = TRUE) # most important variables
imp$pixel <- rownames(imp)
imp <- imp[order(imp$Overall),]
imp.df <- as.data.frame(tail(imp))
barplot(imp.df$Overall,
 main = "Decision Tree Variable Importance",
 xlab = "Importance",
 names.arg = imp.df$pixel,
 las = 2,
 col = "blue",
 horiz = TRUE)
```



Prediction of Test Data with the Unpruned Model

```{r}
set.seed(500)
pred<- predict(object=tree_model1,test[-1],type="class")
tree_model1.cm <- confusionMatrix(pred, test$attrition_flag)
tree_model1.cm$table
```

```
                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               326               114
  Existing Customer               158              2440
```

Based on the above confusion Matrix the Unpruned Tree predicted accurately for 326 loss of customers and 2440 of existing customers accurately.

This has led to an accuracy of 91.97% as shown below.

```{r}
tree_model1.acc <- round(tree_model1.cm$overall[1]*100,2)
tree_model1.acc
```

```
Accuracy
   91.97
```

PRUNED DECISION TREE

A detailed look at the complexity parameter(cp) will help to identify the optimal value that can be used for pruning.

```r
printcp(tree_model1)
```

```
Classification tree:
rpart(formula = attrition_flag ~ ., data = train, method = "class",
    model = T)

Variables actually used in tree construction:
[1] contacts        months_inactive products_num    revolving_bal
[5] total_trans_amt total_trans_cnt

Root node error: 1181/7089 = 0.1666

n= 7089

        CP nsplit rel error  xerror     xstd
1 0.143099      0   1.00000 1.00000 0.026565
2 0.081287      2   0.71380 0.71380 0.023077
3 0.038103      3   0.63251 0.63251 0.021889
4 0.027942      4   0.59441 0.61812 0.021668
5 0.013548      6   0.53853 0.53853 0.020374
6 0.010000      7   0.52498 0.53683 0.020345
```

```r
```

The CP value with lowest x-error value is identified as the optimal value for pruning.

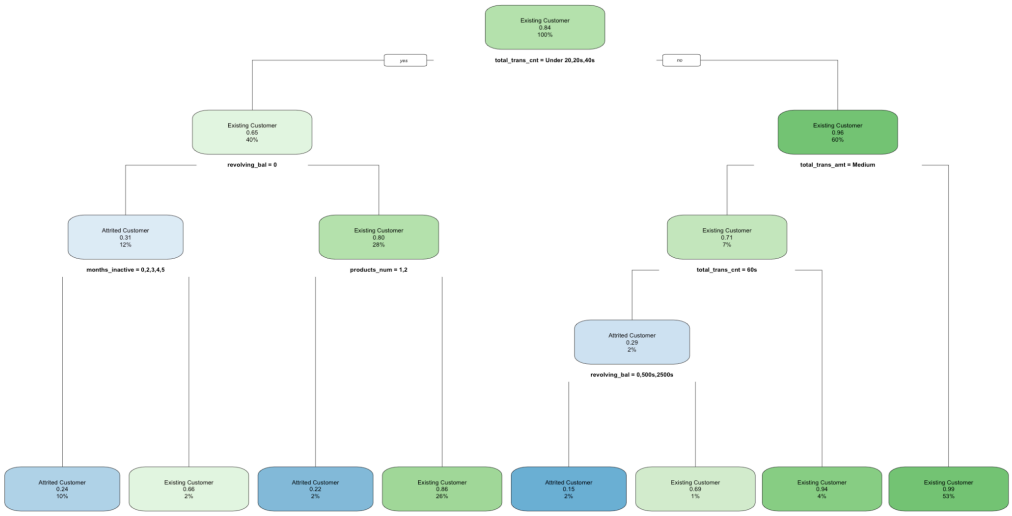
The Decision Tree is further pruned with a CP value of ‘0.01000’ as shown below:

```
#choose a cp(complexity parameter) with lowest x-error value
```{r}
set.seed(1000)
tree_model2 <- prune(tree_model1, cp = 0.010000)
pred<- predict(object=tree_model2,test[-1],type="class")
tree_model2.cm <- confusionMatrix(pred, test$attrition_flag)
tree_model2.cm$table
```
```

```
                    Reference
Prediction           Attrited Customer Existing Customer
  Attrited Customer                326                114
  Existing Customer                158               2440
```

The confusion matrix shows a slight improvement and has an accuracy of 91.97 as shown below:

```
```{r}
tree_model2.acc <- round(tree_model2.cm$overall[1]*100,2)
tree_model2.acc

```
```

```
Accuracy
   91.97
```

Plotting Pruned Tree:



Checking for overfitting:

Decision trees are a nonparametric machine learning algorithm that is very flexible and is subject to overfitting training data. This problem can be addressed by pruning a tree after it has learned to remove some of the detail it has picked up.

The most popular resampling technique is k-fold cross validation. It allows you to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data.

Cross Validation with K =3

```{r}
n <- nrow(train)
K <- 3
size <- n%/%K
set.seed(100)
rand_value <- runif(n)
rank <- rank(rand_value)
block <- (rank-1)%/%size+1
block <- as.factor(block)
all.err_tree<- numeric(0)
for (k in 1:K) {
 # learn the model on all individuals except the k block
 model.1<- rpart(attrition_flag~., data=train[block!=k,], method="class")
 # apply the model to the block number k
 pred.1<- predict(model.1, newdata=train[block==k,], type="class")
 # confusion matrix
  mc<- table(train$attrition_flag[block==k],pred.1)
 # error rate
 err<- 1.0 - (mc[1,1]+mc[2,2])/sum(mc)
 # keep
 all.err_tree<- rbind(all.err_tree,err)
}
```

The approx. error percentage is as shown below:

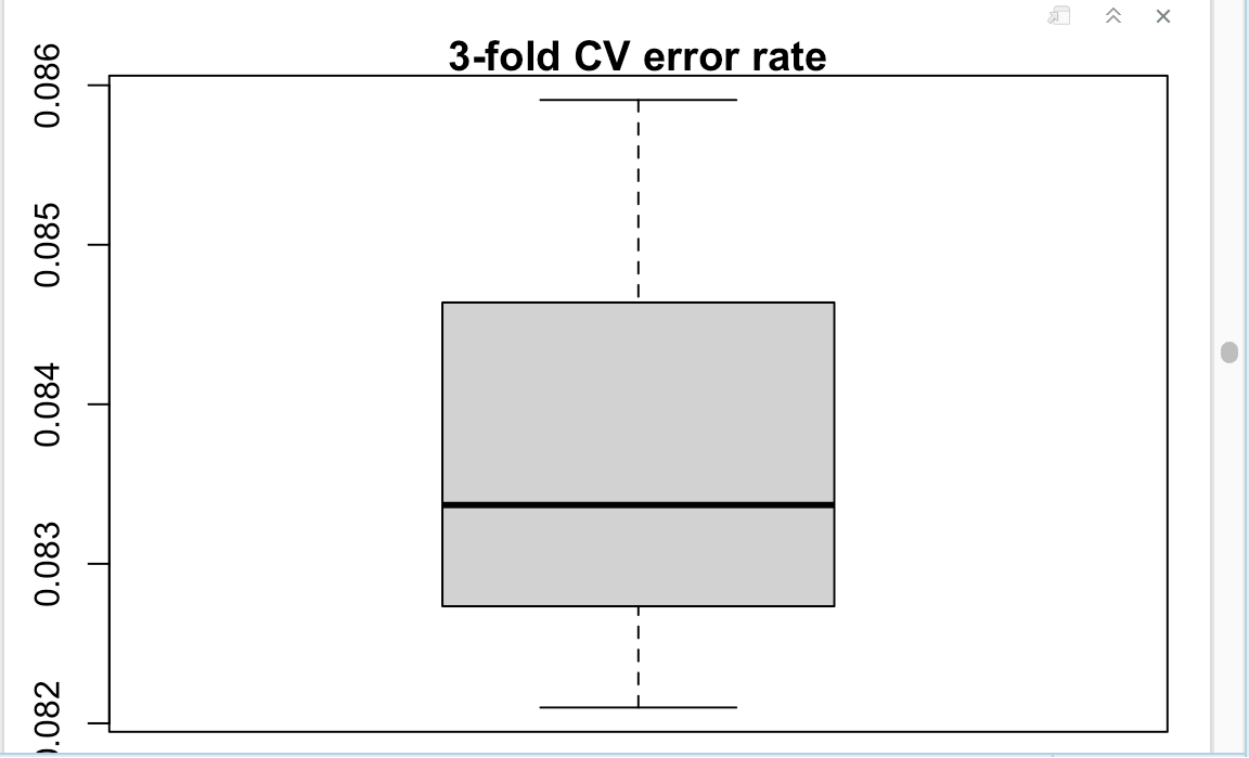
```{r}
all.err_tree
```

```
          [,1]
err 0.08336860
err 0.08590774
err 0.08209903
```

```
Error percentage is approx. 8%
```

Plotting k-fold Error Rate:

```r
par(mar=c(.5,2,1,.5))
k.fold.error <-as.data.frame(all.err_tree)
boxplot(k.fold.error, main="3-fold CV error rate")
```



Cross validation check resulted in an approx. error rate of 8% which is indicates the model is not overfitted and accurate for further analysis.

SUPPORT VECTOR MACHINES(SVM) ANALYSIS

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression, and outlier detection. The advantages of support vector machines are: Effective in high dimensional spaces. SVM model with less supporting vectors implies faster processing.

Loading data for SVM processing. 70-30 is the split ration for Train and Test Datasets.

```r
set.seed(123)
ccdata_svm<-ccdata
split1<- sample(c(rep(0, 0.7 * nrow(ccdata_svm)), rep(1, 0.3 *
nrow(ccdata_svm))))
```

```r
trainsvm <- ccdata_svm[split1 == 0, ]

testsvm <- ccdata_svm[split1== 1, ]
```

SVM Model supports different types of kernels for its processing. SVM uses kernels to find an optimal boundary for its outputs.

As a rule of thumb, SVM analysis can start with Linear Kernel.

SVM WITH KERNEL= LINEAR

With Kernel -Linear

```r
set.seed(345)
svm_linear = svm(formula =attrition_flag ~ .,
                 data = trainsvm,
                 type = 'C-classification',
                 kernel = 'linear')
```

The detail of linear kernel is as shown below:

```r
print(svm_linear)
```

```
Call:
svm(formula = attrition_flag ~ ., data = trainsvm, type = "C-
classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  1423
```

Number of supporting vectors is 1423.

Prediction and confusion matrix using Linear Kernel:

```r
svmlinear_pred = predict(svm_linear, newdata = testsvm)
```

```r
cm = table(testsvm$attrition_flag, svmlinear_pred)
cm
```

```
                    svmlinear_pred
                     Attrited Customer Existing Customer
  Attrited Customer               346               153
  Existing Customer                89              2450
```

```
confusionMatrix(testsvm$attrition_flag, svmlinear_pred)
```

```
Confusion Matrix and Statistics

                   Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               346               153
  Existing Customer                89              2450

               Accuracy : 0.9203
                 95% CI : (0.9101, 0.9297)
    No Information Rate : 0.8568
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6941

 Mcnemar's Test P-Value : 5.126e-05

            Sensitivity : 0.7954
            Specificity : 0.9412
         Pos Pred Value : 0.6934
         Neg Pred Value : 0.9649
             Prevalence : 0.1432
         Detection Rate : 0.1139
   Detection Prevalence : 0.1643
      Balanced Accuracy : 0.8683
```

No information rate is Naive classifier which needs to be exceeded to prove that model is significant.

The accuracy is 92.03% and Naïve classifier is 85.68%, hence the model is significant.

SVM WITH KERNEL = POLYNOMIAL

```
#### Using Polynomial
```{r}
set.seed(423)
svm_poly = svm(formula =attrition_flag ~ .,
                 data = trainsvm,
                 type = 'C-classification',
                 kernel = 'polynomial')
```
```

Calculating the number of supporting vectors:

```
```{r}
print(svm_poly)
```
```

```
Call:
svm(formula = attrition_flag ~ ., data = trainsvm, type = "C-
classification",
    kernel = "polynomial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  1
     degree:  3
     coef.0:  0

Number of Support Vectors:  2468
```

The number of supporting vectors is 2468 and higher than the Linear kernel. This is an indication of an inferior kernel type compared to linear.

Calculation of Prediction and Accuracy:

```r
svmpoly_pred = predict(svm_poly, newdata = testsvm)
```

```r
confusionMatrix(testsvm$attrition_flag, svmpoly_pred)
```

```
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer                 0                499
  Existing Customer                 0               2539

               Accuracy : 0.8357
                 95% CI : (0.8221, 0.8488)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity :     NA
            Specificity : 0.8357
         Pos Pred Value :     NA
         Neg Pred Value :     NA
             Prevalence : 0.0000
         Detection Rate : 0.0000
   Detection Prevalence : 0.1643
```

Accuracy for Polynomial Kernel is 83.57%.

The Naïve Classifier is 100% which is more than the accuracy. Hence SVM with polynomial kernel is not significant and can be eliminated from further considerations.

SVM WITH KERNEL = RADIAL

```
# kernel="radial"
```{r}
set.seed(123)
svm_rad = svm(formula =attrition_flag ~ .,
                  data = trainsvm,
                  type = 'C-classification',
                  kernel = 'radial')
```
```

Calculating the number of supporting vectors:

```
```{r}
print(svm_rad)
```
```

```
Call:
svm(formula = attrition_flag ~ ., data = trainsvm, type = "C-classification",
    kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  2120
```

The Number of support vectors are 2120. This is lower than polynomial and should yield a better accuracy and significance.

Calculation of Prediction and Accuracy:

```
```{r}
svmrad_pred = predict(svm_rad, newdata = testsvm)
```
```

```r
confusionMatrix(testsvm$attrition_flag, svmrad_pred)
```

```
Confusion Matrix and Statistics

                   Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               247               252
  Existing Customer                45              2494

               Accuracy : 0.9022
                 95% CI : (0.8911, 0.9126)
    No Information Rate : 0.9039
    P-Value [Acc > NIR] : 0.6353

                  Kappa : 0.5727

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.84589
            Specificity : 0.90823
         Pos Pred Value : 0.49499
         Neg Pred Value : 0.98228
             Prevalence : 0.09612
         Detection Rate : 0.08130
   Detection Prevalence : 0.16425
      Balanced Accuracy : 0.87706

       'Positive' Class : Attrited Customer
```

The accuracy of the model is 90.22%.

Naïve Classifier (Non-Information Rate) is almost at the same level of 90.39%. This makes the model less significant.

SVM WITH KERNEL = SIGMOID

```r
set.seed(565)
svm_sig = svm(formula =attrition_flag ~ .,
                  data = trainsvm,
                  type = 'C-classification',
                  kernel = 'sigmoid')
```

Calculating the number of supporting vectors:

```r
print(svm_sig)
```

```
Call:
svm(formula = attrition_flag ~ ., data = trainsvm, type = "C-classification",
    kernel = "sigmoid")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  sigmoid
       cost:  1
     coef.0:  0

Number of Support Vectors:  2242
```

The number of supporting vectors is 2242 which is higher than Linear, indicating a more complex model.

Calculation of Prediction and Accuracy:

```r
svmsig_pred = predict(svm_sig, newdata = testsvm)
```

```r
confusionMatrix(testsvm$attrition_flag, svmsig_pred)
```

```
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               183               316
  Existing Customer                22              2517

               Accuracy : 0.8887
                 95% CI : (0.877, 0.8997)
    No Information Rate : 0.9325
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4691

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.89268
            Specificity : 0.88846
         Pos Pred Value : 0.36673
         Neg Pred Value : 0.99134
             Prevalence : 0.06748
         Detection Rate : 0.06024
   Detection Prevalence : 0.16425
      Balanced Accuracy : 0.89057

       'Positive' Class : Attrited Customer
```

The accuracy is 88.87%

No-Information Rate is 93.25% which makes the model not significant.

Finding Optimal Cost Factor for Linear Model:

Cost Factor (c) is the penalty factor. If it is too large, there is a high penalty for non-separable points and can increase the number of supporting vectors and overfit. If it is too small, there are chances of underfitting the model.

```r
set.seed(4321)
tune.model.1<-tune(svm,attrition_flag~.,data=trainsvm,
                   kernel="linear",
                   ranges=list(cost=c(0.01,.1,1,10,100,1000)))
tune1.best.performance<-round(tune.model.1$best.performance,3)
```

```r
tune1.best.performance
```

```
[1] 0.08
```

Recalculating accuracy:

```r
set.seed(50)
svm_linear1 = svm(formula =attrition_flag ~ .,
                  data = trainsvm,
                  type = 'C-classification',cost=0.08,
                  kernel = 'linear')
```

```r
svmlinear_pred1 = predict(svm_linear1, newdata = testsvm)
```

```r
confusionMatrix(testsvm$attrition_flag, svmlinear_pred1)
```

```
Confusion Matrix and Statistics

                   Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               313               186
  Existing Customer                81              2458

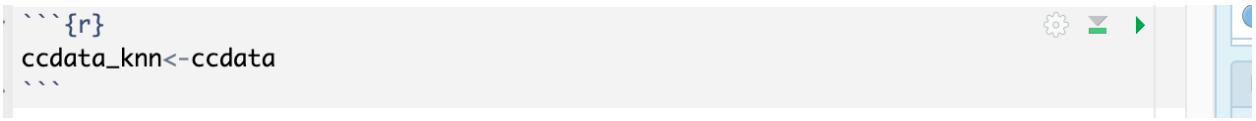
               Accuracy : 0.9121
```

Accuracy has slightly gone down. This could be because as an optimal value of C could have avoided some previous misclassification.

KNN(K-NEAREST NEIGHBOR) ANALYSIS

kNN which stands for K Nearest Neighbor, is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

Loading data for kNN processing.



Pre-Processing for kNN:

For all previous models the data was formatted to be categorical factors. Now it needs to be converted to numeric for kNN.

```
str(ccdata_knn)
```

```
'data.frame':   10127 obs. of  20 variables:
 $ attrition_flag  : Factor w/ 2 levels "Attrited Customer",..: 2 2 2 2 2 2 2 2
2 2 ...
 $ age             : Factor w/ 6 levels "twenties","thirties",..: 3 3 4 2 2 3 4
2 2 3 ...
 $ gender          : Factor w/ 2 levels "F","M": 2 1 2 1 2 2 2 2 2 2 ...
 $ dependents      : Factor w/ 6 levels "0","1","2","3",..: 4 6 4 5 4 3 5 1 4 3
...
 $ education       : Factor w/ 7 levels "College","Doctorate",..: 4 3 3 4 6 3 7
4 6 3 ...
 $ marital         : Factor w/ 4 levels "Divorced","Married",..: 2 3 2 4 2 2 2
4 3 3 ...
 $ income          : Factor w/ 6 levels "$120K +","$40K - $60K",..: 3 5 4 5 3 2
1 3 3 4 ...
 $ card            : Factor w/ 4 levels "Blue","Gold",..: 1 1 1 1 1 1 2 4 1 1
...
 $ months_active   : Factor w/ 7 levels "tens","twenties",..: 3 4 3 3 2 3 4 2 3
3 ...
 $ products_num    : Factor w/ 6 levels "1","2","3","4",..: 5 6 4 3 5 3 6 2 5 6
...
 $ months_inactive : Factor w/ 7 levels "0","1","2","3",..: 2 2 2 5 2 2 2 3 3 4
...
 $ contacts        : Factor w/ 7 levels "0","1","2","3",..: 4 3 1 2 1 3 4 3 1 4
...
 $ creditlimit     : Factor w/ 4 levels "Low","Medium",..: 2 1 1 1 1 1 4 4 3 2
...
 $ revolving_bal   : Factor w/ 6 levels "0","500s","1000s",..: 2 3 1 6 1 3 6 4
6 4 ...
 $ open_to_buy     : Factor w/ 6 levels "0","500s","1000s",..: 3 2 1 1 1 1 6 5
4 2 ...
 $ amt_change_q4_q1: Factor w/ 3 levels "Low","Medium",..: 2 2 3 2 2 2 2 2 3 2
...
 $ total_trans_amt : Factor w/ 3 levels "Low","Medium",..: 1 1 1 1 1 1 1 1 1 1
```

Converting factors to numeric

```
#### Pre-Processing of data: converting factor to numeric values
```{r}
ccdata_knn$attrition_flag<-as.numeric(ccdata_knn$attrition_flag)
ccdata_knn$age<- as.numeric((ccdata_knn$age))
ccdata_knn$gender<- as.numeric(ccdata_knn$gender)
ccdata_knn$dependents<-as.numeric(ccdata_knn$dependents)
ccdata_knn$education<-as.numeric(ccdata_knn$education)
ccdata_knn$marital<-as.numeric(ccdata_knn$marital)
ccdata_knn$income<-as.numeric(ccdata_knn$income)
ccdata_knn$card<-as.numeric(ccdata_knn$card)
ccdata_knn$months_active<-as.numeric(ccdata_knn$months_active)
ccdata_knn$products_num<-as.numeric(ccdata_knn$products_num)
ccdata_knn$months_inactive<-as.numeric(ccdata_knn$months_inactive)
ccdata_knn$contacts<-as.numeric(ccdata_knn$contacts)
ccdata_knn$creditlimit<-as.numeric(ccdata_knn$creditlimit)
ccdata_knn$revolving_bal<-as.numeric(ccdata_knn$revolving_bal)
ccdata_knn$open_to_buy<- as.numeric(ccdata_knn$open_to_buy)
ccdata_knn$amt_change_q4_q1<-as.numeric(ccdata_knn$amt_change_q4_q1)
ccdata_knn$total_trans_amt<-as.numeric(ccdata_knn$total_trans_amt)
ccdata_knn$total_trans_cnt<-as.numeric(ccdata_knn$total_trans_cnt)
ccdata_knn$cnt_change_q4_q1<-as.numeric(ccdata_knn$cnt_change_q4_q1)
ccdata_knn$utilization<-as.numeric(ccdata_knn$utilization)
```
```

Exploring the new structure:

```
```{r}
str(ccdata_knn)
```

'data.frame':   10127 obs. of 20 variables:
 $ attrition_flag : num 2 2 2 2 2 2 2 2 2 2 ...
 $ age   : num 3 3 4 2 2 3 4 2 2 3 ...
 $ gender   : num 2 1 2 1 2 2 2 2 2 2 ...
 $ dependents   : num 4 6 4 5 4 3 5 1 4 3 ...
 $ education   : num 4 3 3 4 6 3 7 4 6 3 ...
 $ marital   : num 2 3 2 4 2 2 2 4 3 3 ...
 $ income   : num 3 5 4 5 3 2 1 3 3 4 ...
 $ card   : num 1 1 1 1 1 1 2 4 1 1 ...
 $ months_active   : num 3 4 3 3 2 3 4 2 3 3 ...
 $ products_num   : num 5 6 4 3 5 3 6 2 5 6 ...
 $ months_inactive : num 2 2 2 5 2 2 2 3 3 4 ...
 $ contacts   : num 4 3 1 2 1 3 4 3 1 4 ...
 $ creditlimit   : num 2 1 1 1 1 1 4 4 3 2 ...
 $ revolving_bal   : num 2 3 1 6 1 3 6 4 6 4 ...
 $ open_to_buy   : num 3 2 1 1 1 1 6 5 4 2 ...
 $ amt_change_q4_q1: num 2 2 3 2 2 2 2 2 3 2 ...
 $ total_trans_amt : num 1 1 1 1 1 1 1 1 1 1 ...
 $ total_trans_cnt : num 3 2 1 1 2 2 2 2 2 2 ...
 $ cnt_change_q4_q1: num 2 3 2 2 3 1 1 1 1 1 ...
 $ utilization   : num 1 1 1 3 1 1 1 1 1 1 ...
```

```r
head(ccdata_knn)
```

Description: df [6 × 20]

total_trans_amt <dbl>	total_trans_cnt <dbl>	cnt_change_q4_q1 <dbl>	utilization <dbl>
1	3	2	1
1	2	3	1
1	1	2	1
1	1	2	3
1	2	3	1
1	2	1	1

6 rows | 18–21 of 20 columns

Initial exploration implies that the data is already in the normalized range and no further pre-processing is required.

Like in the previous model's data is split 70-30% for Train and Test purposes.

```r
set.seed(123)
split <- sample(1:nrow(ccdata_knn),size=nrow(ccdata_knn)*0.7,replace = FALSE)
#random selection of 70% data.

trainknn <- ccdata_knn[split,] # 70% training data
testknn <- ccdata_knn[-split,] # remaining 30% test data
```

```r
dim(trainknn)
dim(testknn)
```

```
[1] 7088   20
[1] 3039   20
```

```r
trainknn_labels <- ccdata_knn[split,1]
testknn_labels <-ccdata_knn[-split,1]
NROW(trainknn_labels)

```

```
[1] 7088
```

kNN needs user to input k value for classification. To start the analysis , supplying a k value which is the square root of number of rows of Train dataset.

√7088 = 84.19

The initial k values are 84 and 85.

```r
knn.84 <- knn(train=trainknn, test=testknn, cl=trainknn_labels, k=84)
knn.85 <- knn(train=trainknn, test=testknn, cl=trainknn_labels, k=85)

```

Calculating Confusion Matrices and Accuracy:

```r
#Calculate the proportion of correct classification for k = 84,85
ACC.84 <- 100 * sum(testknn_labels == knn.84)/NROW(testknn_labels)
ACC.85<- 100 * sum(testknn_labels == knn.85)/NROW(testknn_labels)

ACC.84
ACC.85
```

```
[1] 89.66765
[1] 89.63475
```

```r
confusionMatrix(table(knn.84 ,testknn_labels))
```

```
Confusion Matrix and Statistics

      testknn_labels
knn.84    1    2
     1  170    6
     2  308 2555

               Accuracy : 0.8967
                 95% CI : (0.8853, 0.9073)
    No Information Rate : 0.8427
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4755

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

            Sensitivity : 0.35565
            Specificity : 0.99766
         Pos Pred Value : 0.96591
         Neg Pred Value : 0.89242
             Prevalence : 0.15729
         Detection Rate : 0.05594
   Detection Prevalence : 0.05791
      Balanced Accuracy : 0.67665

       'Positive' Class : 1
```

```
Confusion Matrix and Statistics

      testknn_labels
knn.85    1    2
     1  169    6
     2  309 2555

               Accuracy : 0.8963
                 95% CI : (0.885, 0.907)
    No Information Rate : 0.8427
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4732

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

            Sensitivity : 0.35356
            Specificity : 0.99766
         Pos Pred Value : 0.96571
         Neg Pred Value : 0.89211
             Prevalence : 0.15729
         Detection Rate : 0.05561
   Detection Prevalence : 0.05758
      Balanced Accuracy : 0.67561

       'Positive' Class : 1
```

In both scenarios, accuracy is around 89.6%.

This is not a high accuracy.

For a better understanding, different k values are supplied (k=1 thru 50)

Looping to calculate accuracy when k=1 through 50.

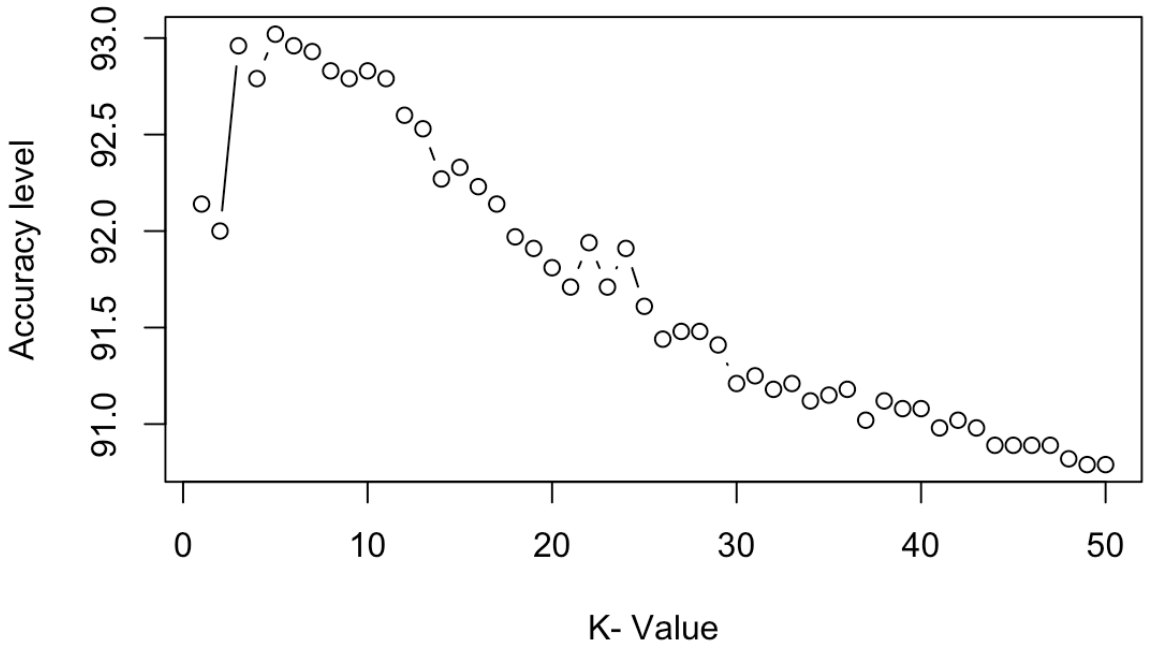
```{r}
i=1
k.optm=1
for (i in 1:50){
 knn.mod <- knn(train=trainknn, test=testknn, cl=trainknn_labels, k=i)
 k.optm[i] <- round(100 * sum(testknn_labels == knn.mod)/NROW(testknn_labels),2)
 k=i
 cat(k,'=',k.optm[i],'
')
 }

```

And the results are as shown below:

```
1 = 92.14
2 = 92
3 = 92.96
4 = 92.79
5 = 93.02
6 = 92.96
7 = 92.93
8 = 92.83
9 = 92.79
10 = 92.83
11 = 92.79
12 = 92.6
13 = 92.53
14 = 92.27
15 = 92.33
16 = 92.23
17 = 92.14
18 = 91.97
19 = 91.91
20 = 91.81
21 = 91.71
22 = 91.94
23 = 91.71
24 = 91.91
25 = 91.61
26 = 91.44
27 = 91.48
28 = 91.48
29 = 91.41
30 = 91.21
31 = 91.25
32 = 91.18
33 = 91.21
34 = 91.12
35 = 91.15
36 = 91.18
37 = 91.02
38 = 91.12
39 = 91.08
40 = 91.08
41 = 90.98
42 = 91.02
43 = 90.98
44 = 90.89
45 = 90.89
46 = 90.89
47 = 90.89
48 = 90.82
49 = 90.79
50 = 90.79
```

Plotting accuracy with various k-values:

```r
#Accuracy plot
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```



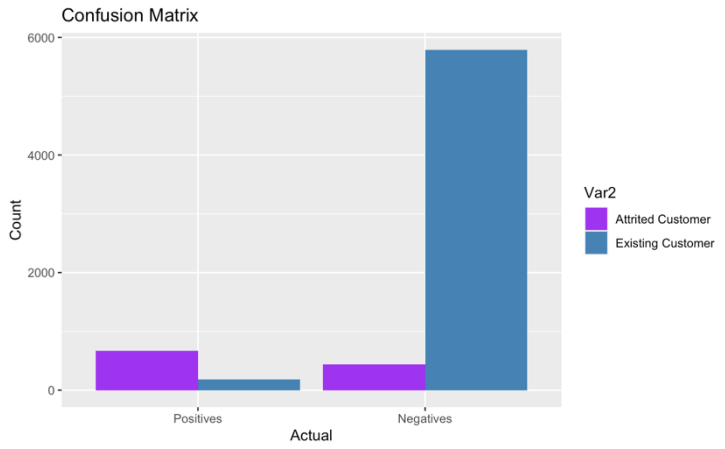
From the above the highest accuracy of 93.02 was with k=5

NAÏVE BAYES

Naïve Bayes is another machine learning algorithm that is used for classification and prediction tasks. It predicts the probability of an event occurring based on prior data or knowledge.

TRAIN DATA

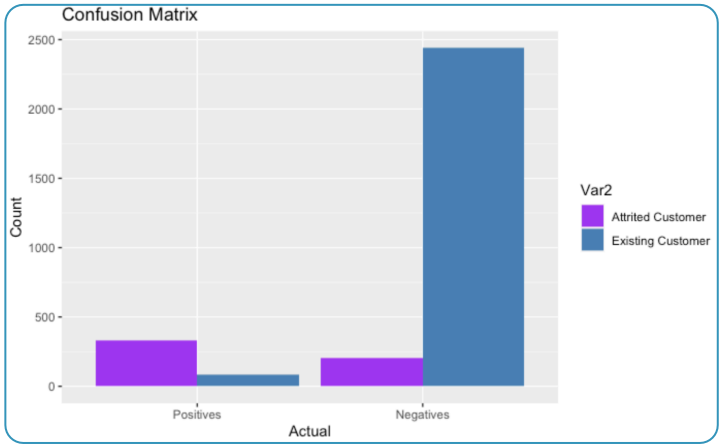
```
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               682              179
  Existing Customer               443             5785

               Accuracy : 0.9123
                 95% CI : (0.9054, 0.9187)
    No Information Rate : 0.8413
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6368

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

            Sensitivity : 0.60622
            Specificity : 0.96999
         Pos Pred Value : 0.79210
         Neg Pred Value : 0.92887
             Prevalence : 0.15870
         Detection Rate : 0.09621
   Detection Prevalence : 0.12146
      Balanced Accuracy : 0.78810

       'Positive' Class : Attrited Customer
```



Accuracy in the context of a Naive Bayes confusion matrix refers to the proportion of correctly classified instances out of all instances. It is a measure of the overall performance of the classifier in predicting whether a customer will churn or not. and it is at 91.12%. The positive class are the past customers. The true positive for past customers is the 682 number where the model correctly predicts a customer will leave and they do. The false positive is 179, where it predicts a customer will leave but they did not. The false negative is 443 where the model predicts a customer will not leave but they do. Lastly, the true negatives where existing customers were correctly classified was 5785, where a customer is predicted to stay, and they do.

TRAIN DATA

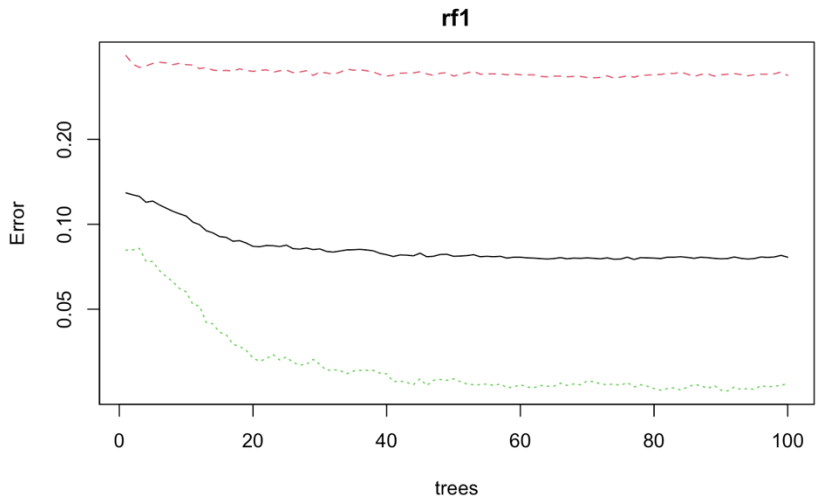
```
Confusion Matrix and Statistics

                  Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer               311                83
  Existing Customer               206              2439

               Accuracy : 0.9049
                 95% CI : (0.8939, 0.9151)
    No Information Rate : 0.8299
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.628

 Mcnemar's Test P-Value : 7.153e-13

            Sensitivity : 0.6015
            Specificity : 0.9671
         Pos Pred Value : 0.7893
         Neg Pred Value : 0.9221
             Prevalence : 0.1701
         Detection Rate : 0.1023
   Detection Prevalence : 0.1296
      Balanced Accuracy : 0.7843

       'Positive' Class : Attrited Customer
```



Naive Bayes test set calculated an accuracy rate of 90.49%, which makes the miscalculation rate at 9.51%. This is not as favorable as the training set.

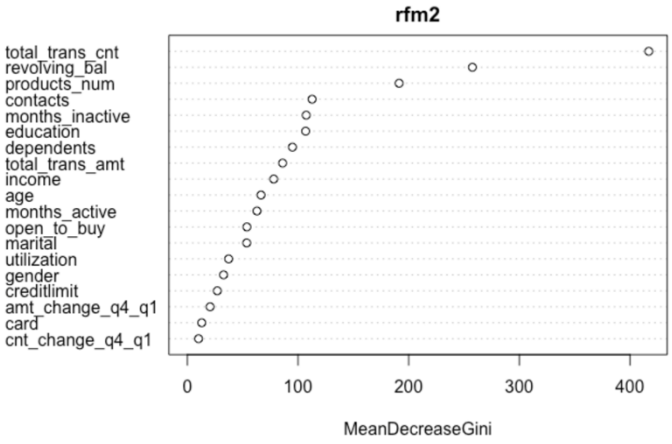
RANDOM FOREST

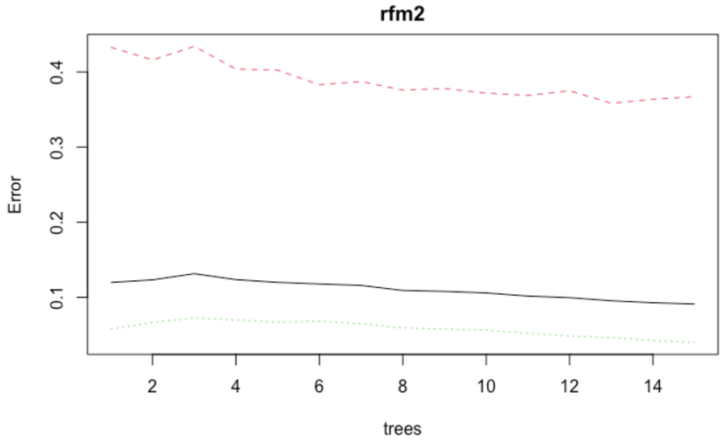
Random forest is another machine learning algorithm that is used for classification. It uses decision trees, and each tree is trained to make a prediction based on the subset of the input data.



The plot on the left shows the Error and the Number of Trees. We can easily notice that how the Error is dropping as we keep on adding more and more trees and average them. We settled for 15 trees as our optimal number.

At 15 trees, we plotted the variance importance plot and the decision tree error plot.

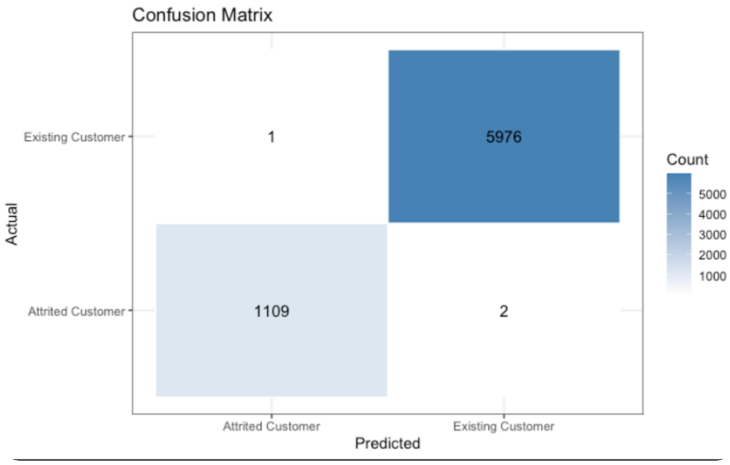




With the variance importance plot helped us identify the most important variables that contributed to predicting the model within the random forest method and understand how it contributes to the model's accuracy. This can be used to improve the model's performance by focusing on the most important variables in the future. Looking at the variance importance plot, it falls in line with our apriori results, which is a good sign.

Now looking at the confusion matrix, the random forest model has been our best and most accurate model at 99.96%. There is 1 false positive and 2 false negatives.

```
Confusion Matrix and Statistics

                 Reference
Prediction          Attrited Customer Existing Customer
  Attrited Customer              1109                1
  Existing Customer                 2             5976

               Accuracy : 0.9996
                 95% CI : (0.9988, 0.9999)
    No Information Rate : 0.8433
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9984

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9982
            Specificity : 0.9998
         Pos Pred Value : 0.9991
         Neg Pred Value : 0.9997
             Prevalence : 0.1567
         Detection Rate : 0.1565
   Detection Prevalence : 0.1566
      Balanced Accuracy : 0.9990

       'Positive' Class : Attrited Customer
```



RESULTS COMPARISON

Model	Accuracy %	Comments
Decision Tree	91.97	Unpruned
Decision Tree	91.97	Pruned
Naïve Bayes	90.49	
SVM Linear	92.03	
SVM Polynomial	83.57	
SVM Radial	90.22	
SVM Sigmoid	88.87	
kNN	93.02	K=5
Random Forest	99.96	Trees = 15

The results show that the Random Forest method showed the highest accuracy at 99.96% in predicting whether a customer will leave or stay with the credit card service at 15 decision trees. The next highest came in at 93.02% accuracy with kNN at 5. For the banks data set, it is recommended to use the Random Forest method as the machine learning algorithm due to its high accuracy.

CONCLUSION

Based on the various model analysis and accuracy comparison, Random Forest came out as the best model. The recommendation to bank's marketing team should be to

continually collect the data and monitor for any changes to the critical data points identified during the analysis such as

a) Total Transaction Count

b) Revolving Balance

c) Number of contacts

d) New customers

A declining trend in any of the above should trigger a customer check-in process to ensure customer satisfaction and the bank can work on any issues or feedback from the customer. The banks marketing team can take the guidance and revise the products and services to encourage customers to stay back as well as invite new customers to enroll in the services.

In addition, new customers with low number of contacts seem to leave the credit card services. The marketing team or credit card department can find ways to expand rewards and incentives to retain the customers. For example, credit card customers can have special incentives for other products at the bank such a bonus when opening a savings account or a wider selection of rewards if the credit card has a reward point system.

Credit and debit analytics allow banks to identify the top merchants based on the number of transactions and the value of the transactions. Note shifts in spending habits, and align rewards programs based on where your customers are shopping and what they are buying. Such banking analytics provides the means to enhance the customer experience, identify opportunities for revenue growth, and remain competitive in this increasingly volatile market.

A careful analysis should consider additional analysis of customers other debts, credit scores etc. It should also include additional external factor that can influence a customer's decision such as inflation, pandemic, war, extreme weather etc. These additional data will be helpful in making informed decisions.

Ensuring Customer Satisfaction is an ongoing process and communication plays a key role in that. Reaching out to its customers for feedback and enquiring about their needs should be a top priority for the marketing team. As mentioned in the beginning, the credit card industry is highly competitive and understanding market and competitors are a key to success. Product lines and services offered should always be competitive and relevant for the current economy.