Bank Churners

IST 707 - Final Project

Renjini Rajan & Amanda Norwood

2023-03-05

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 also 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.

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 amount of customers that leave their services for another bank.

Motivation

Our team members both work in the banking industry and we are interested in analyzing the bank data set as practice before we implement potential data analysis methods at our jobs. This is great practice for a smaller data set as well since our banks' contain millions of rows of data.

Problem Statement

Syracuse Bank (SB) noticed that more and more customers are leaving their credit card services and the manager is concerned with the amount of people leaving. SB 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. Our goal is to analyze 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

The Data

The data set was pulled from Kaggle and it contains 10,000 rows of data with each row representing a customer.

Load Data & Libraries

We will load the data and begin our initial analysis and description of the data.

```
ccdata = read.csv('/Users/fruitisushi/Documents/Grad Classes/IST 707 - Applied Machine L
earning/Project/BankChurners.csv')
#ccdata= read.csv("/Users/renjinirajan/Desktop/Masters/IST 707- Applied Machine Learnin
g/BankChurners.csv")
ccdata_cor = ccdata
```

Initial Data Exploration and Cleaning

```
str(ccdata)
```

```
## 'data.frame':    10127 obs. of  23 variables:
##  $ CLIENTNUM
: int  768805383 818770008 713982108 769911858 709106358 713061558 810347208 818906208 7
10930508 719661558 ...
##  $ Attrition_Flag
: chr  "Existing Customer" "Existing Customer" "Existing Customer" "Existing Customer"
...
##  $ Customer_Age
: int  45 49 51 40 40 44 51 32 37 48 ...
##  $ Gender
: chr  "M" "F" "M" "F" ...
##  $ Dependent_count
: int  3 5 3 4 3 2 4 0 3 2 ...
##  $ Education_Level
: chr  "High School" "Graduate" "Graduate" "High School" ...
##  $ Marital_Status
: chr  "Married" "Single" "Married" "Unknown" ...
##  $ Income_Category
: chr  "$60K - $80K" "Less than $40K" "$80K - $120K" "Less than $40K" ...
##  $ Card_Category
: chr  "Blue" "Blue" "Blue" "Blue" ...
##  $ Months_on_book
: int  39 44 36 34 21 36 46 27 36 36 ...
##  $ Total_Relationship_Count
: int  5 6 4 3 5 3 6 2 5 6 ...
##  $ Months_Inactive_12_mon
: int  1 1 1 4 1 1 1 2 2 3 ...
##  $ Contacts_Count_12_mon
: int  3 2 0 1 0 2 3 2 0 3 ...
##  $ Credit_Limit
: num  12691 8256 3418 3313 4716 ...
##  $ Total_Revolving_Bal
: int  777 864 0 2517 0 1247 2264 1396 2517 1677 ...
##  $ Avg_Open_To_Buy
: num  11914 7392 3418 796 4716 ...
##  $ Total_Amt_Chng_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_Ct
: int  42 33 20 20 28 24 31 36 24 32 ...
##  $ Total_Ct_Chng_Q4_Q1
: num  1.62 3.71 2.33 2.33 2.5 ...
##  $ Avg_Utilization_Ratio
: num  0.061 0.105 0 0.76 0 0.311 0.066 0.048 0.113 0.144 ...
##  $ Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependen
t_count_Education_Level_Months_Inactive_12_mon_1: num  9.34e-05 5.69e-05 2.11e-05 1.34e-
04 2.17e-05 ...
##  $ Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependen
t_count_Education_Level_Months_Inactive_12_mon_2: num  1 1 1 1 1 ...
```

```
dim(ccdata)
```

```
## [1] 10127    23
```

The initial data set shows 10,127 rows and 23 columns. The data structure is a mix of integers, characters, and numerical data.

Now, we will check for incomplete rows.

```
(sum(is.na(ccdata)))
```

```
## [1] 0
```

There are 0 rows with incomplete data.

Next, we will rename the columns to be more code friendly by shortening some of the variable names and changing it to all lower case.

```
##Checking column names
colnames(ccdata)
```

```
##  [1] "CLIENTNUM"
##  [2] "Attrition_Flag"
##  [3] "Customer_Age"
##  [4] "Gender"
##  [5] "Dependent_count"
##  [6] "Education_Level"
##  [7] "Marital_Status"
##  [8] "Income_Category"
##  [9] "Card_Category"
## [10] "Months_on_book"
## [11] "Total_Relationship_Count"
## [12] "Months_Inactive_12_mon"
## [13] "Contacts_Count_12_mon"
## [14] "Credit_Limit"
## [15] "Total_Revolving_Bal"
## [16] "Avg_Open_To_Buy"
## [17] "Total_Amt_Chng_Q4_Q1"
## [18] "Total_Trans_Amt"
## [19] "Total_Trans_Ct"
## [20] "Total_Ct_Chng_Q4_Q1"
## [21] "Avg_Utilization_Ratio"
## [22] "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_1"
## [23] "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_2"
```

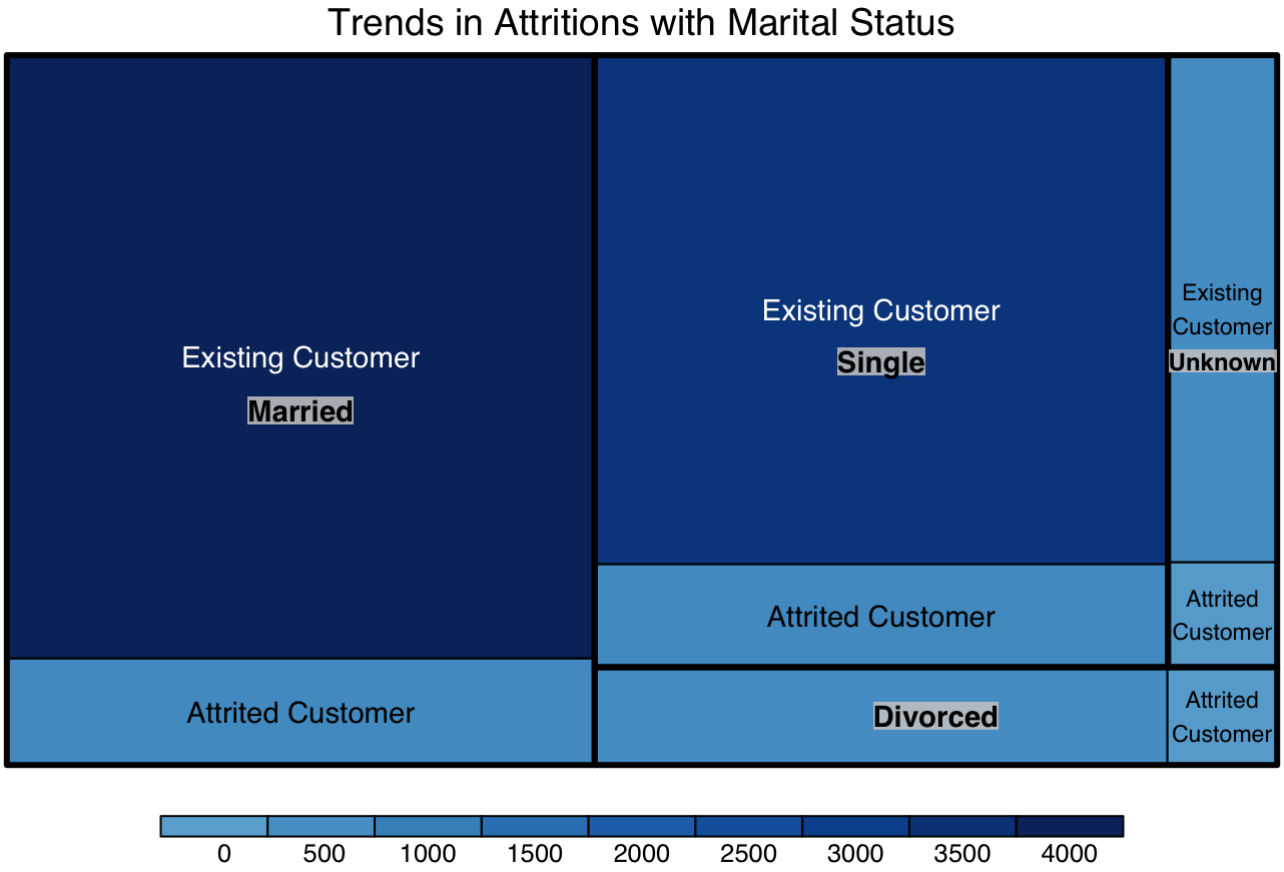
```
##Rename columns

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'
         ))
```

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

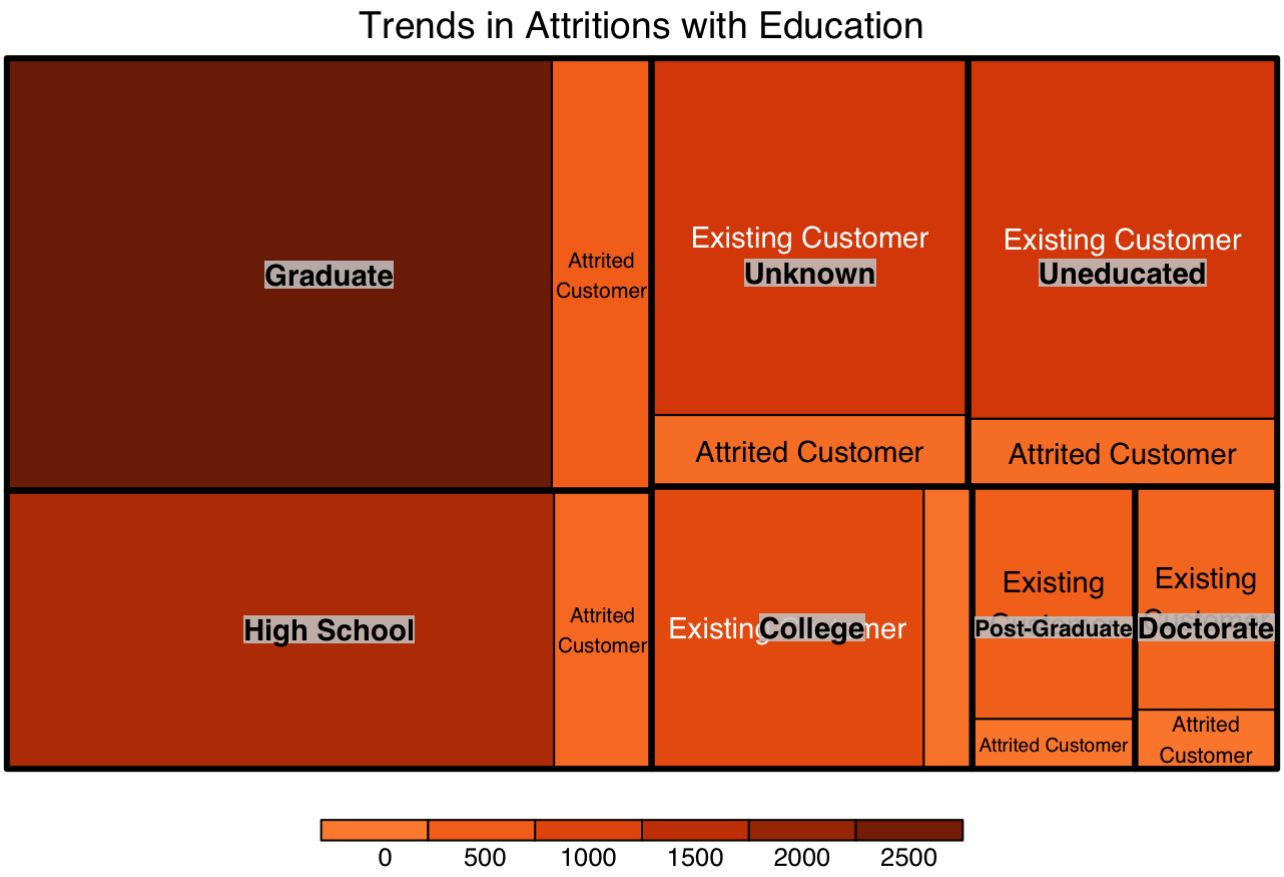
a. Trends in attrition with Marital Status and Education. Based on the trend shown below in genral our dataset contains a lesser precentage of Attried Customers and customers who are single has a visually higher attrition rate.

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



b. Trends in Attritions with Education Based on the visualization the attrition trend seems to be same across various education level from unknown to Doctorate. This could be an indication that education might not be an influencing factor for customer attrition. We will check for this as we progress through the analysis using various algorithms and techniques.

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



We must check which variables are relevant to our analysis.

```
summary(ccdata)
```

```
##    CLIENTNUM         attrition_flag          age           gender
##  Min.   :708082083   Length:10127       Min.   :26.00   Length:10127
##  1st Qu.:713036770   Class :character   1st Qu.:41.00   Class :character
##  Median :717926358   Mode  :character   Median :46.00   Mode  :character
##  Mean   :739177606                      Mean   :46.33
##  3rd Qu.:773143533                      3rd Qu.:52.00
##  Max.   :828343083                      Max.   :73.00
##    dependents     education           marital             income
##  Min.   :0.000   Length:10127       Length:10127       Length:10127
##  1st Qu.:1.000   Class :character   Class :character   Class :character
##  Median :2.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2.346
##  3rd Qu.:3.000
##  Max.   :5.000
##      card           months_active    products_num   months_inactive
##  Length:10127       Min.   :13.00   Min.   :1.000   Min.   :0.000
##  Class :character   1st Qu.:31.00   1st Qu.:3.000   1st Qu.:2.000
##  Mode  :character   Median :36.00   Median :4.000   Median :2.000
##                     Mean   :35.93   Mean   :3.813   Mean   :2.341
##                     3rd Qu.:40.00   3rd Qu.:5.000   3rd Qu.:3.000
##                     Max.   :56.00   Max.   :6.000   Max.   :6.000
##     contacts      creditlimit    revolving_bal   open_to_buy
##  Min.   :0.000   Min.   : 1438   Min.   :   0   Min.   :    3
##  1st Qu.:2.000   1st Qu.: 2555   1st Qu.: 359   1st Qu.: 1324
##  Median :2.000   Median : 4549   Median :1276   Median : 3474
##  Mean   :2.455   Mean   : 8632   Mean   :1163   Mean   : 7469
##  3rd Qu.:3.000   3rd Qu.:11068   3rd Qu.:1784   3rd Qu.: 9859
##  Max.   :6.000   Max.   :34516   Max.   :2517   Max.   :34516
##  amt_change_q4_q1 total_trans_amt total_trans_cnt  cnt_change_q4_q1
##  Min.   :0.0000   Min.   :  510   Min.   : 10.00   Min.   :0.0000
##  1st Qu.:0.6310   1st Qu.: 2156   1st Qu.: 45.00   1st Qu.:0.5820
##  Median :0.7360   Median : 3899   Median : 67.00   Median :0.7020
##  Mean   :0.7599   Mean   : 4404   Mean   : 64.86   Mean   :0.7122
##  3rd Qu.:0.8590   3rd Qu.: 4741   3rd Qu.: 81.00   3rd Qu.:0.8180
##  Max.   :3.3970   Max.   :18484   Max.   :139.00   Max.   :3.7140
##   utilization
##  Min.   :0.0000
##  1st Qu.:0.0230
##  Median :0.1760
##  Mean   :0.2749
##  3rd Qu.:0.5030
##  Max.   :0.9990
##  Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_
count_Education_Level_Months_Inactive_12_mon_1
##  Min.   :0.0000077
##  1st Qu.:0.0000990
##  Median :0.0001815
##  Mean   :0.1599975
##  3rd Qu.:0.0003373
##  Max.   :0.9995800
##  Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_
count_Education_Level_Months_Inactive_12_mon_2
```

```
##  Min.   :0.00042
##  1st Qu.:0.99966
##  Median :0.99982
##  Mean   :0.84000
##  3rd Qu.:0.99990
##  Max.   :0.99999
```

CLIENTNUM and the Naives_Bayes_Classifier are irrelevant to our analysis because CLIENTNUM is the customers unique identifier and cannot be factorized. The Naive_Bayes calculation cannot be factorized as well.

First, we can to make sure we have the original list of column names to ensure we are removing the correct columns.

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

```
##  [1] "CLIENTNUM"
##  [2] "attrition_flag"
##  [3] "age"
##  [4] "gender"
##  [5] "dependents"
##  [6] "education"
##  [7] "marital"
##  [8] "income"
##  [9] "card"
## [10] "months_active"
## [11] "products_num"
## [12] "months_inactive"
## [13] "contacts"
## [14] "creditlimit"
## [15] "revolving_bal"
## [16] "open_to_buy"
## [17] "amt_change_q4_q1"
## [18] "total_trans_amt"
## [19] "total_trans_cnt"
## [20] "cnt_change_q4_q1"
## [21] "utilization"
## [22] "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_1"
## [23] "Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depen
dent_count_Education_Level_Months_Inactive_12_mon_2"
```

```
##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)
```

```
##  [1] "CLIENTNUM"        "attrition_flag"   "age"              "gender"
##  [5] "dependents"       "education"        "marital"          "income"
##  [9] "card"             "months_active"    "products_num"     "months_inactive"
## [13] "contacts"         "creditlimit"      "revolving_bal"    "open_to_buy"
## [17] "amt_change_q4_q1" "total_trans_amt"  "total_trans_cnt"  "cnt_change_q4_q1"
## [21] "utilization"
```

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

##Validate CLIENTNUM was successfully dropped.
colnames(ccdata)
```

```
##  [1] "attrition_flag"   "age"              "gender"           "dependents"
##  [5] "education"        "marital"          "income"           "card"
##  [9] "months_active"    "products_num"     "months_inactive"  "contacts"
## [13] "creditlimit"      "revolving_bal"    "open_to_buy"      "amt_change_q4_q1"
## [17] "total_trans_amt"  "total_trans_cnt"  "cnt_change_q4_q1" "utilization"
```

```
##Verifying we still have 0 complete rows
(sum(is.na(ccdata)))
```

```
## [1] 0
```

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.

```
##Validate column names were changed correctly
str(ccdata)
```

```
## '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 $4
0K" ...
##  $ 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 ...
```

Exploratory Data Analysis & Factorization

```
dim(ccdata)
```

```
## [1] 10127    20
```

Factorization

After have cleaned the data set, we will explore the each of the variables and factorize them. We have 20 variables to factorize.

We will explore the character type variables first and find their number of unique variables within the column to factorize them.

1/20: attrition_flag

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

```
unique(ccdata$attrition_flag)
```

```
## [1] "Existing Customer" "Attrited Customer"
```

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

```
##  Factor w/ 2 levels "Attrited Customer",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```
#2 levels are returned
```

2/20: gender

The gender is defined as Male or Female.

```
unique(ccdata$gender)
```

```
## [1] "M" "F"
```

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

```
##  Factor w/ 2 levels "F","M": 2 1 2 1 2 2 2 2 2 2 ...
```

```
#2 levels are returned
```

3/20: education

Education level of the customer.

```
unique(ccdata$education)
```

```
## [1] "High School"   "Graduate"      "Uneducated"    "Unknown"
## [5] "College"       "Post-Graduate" "Doctorate"
```

```
#There are 7 unique fields
ccdata$education = factor(ccdata$education)
#We should see a return factor of 7
str(ccdata$education)
```

```
##  Factor w/ 7 levels "College","Doctorate",..: 4 3 3 4 6 3 7 4 6 3 ...
```

```
#7 levels are returned
```

4/20: marital status

This is the customers marital status.

```
unique(ccdata$marital)
```

```
## [1] "Married"  "Single"   "Unknown"  "Divorced"
```

```
#There are 4 unique fields
ccdata$marital = factor(ccdata$marital)
#We should see a return factor of 4
str(ccdata$marital)
```

```
##  Factor w/ 4 levels "Divorced","Married",..: 2 3 2 4 2 2 2 4 3 3 ...
```

```
#4 levels are returned
```

5/20: income

This variable shows the customers income level.

```
unique(ccdata$income)
```

```
## [1] "$60K - $80K"    "Less than $40K" "$80K - $120K"   "$40K - $60K"
## [5] "$120K +"        "Unknown"
```

```
#There are 6 unique fields
ccdata$income = factor(ccdata$income)
#We should see a return factor of 6
str(ccdata$income)
```

```
##  Factor w/ 6 levels "$120K +","$40K - $60K",..: 3 5 4 5 3 2 1 3 3 4 ...
```

```
#6 levels are returned
```

6/20: card

This field shows the type of credit card the customer is using.

```
unique(ccdata$card)
```

```
## [1] "Blue"     "Gold"     "Silver"   "Platinum"
```

```
#There are 4 unique fields
ccdata$card = factor(ccdata$card)
#We should see a return factor of 4
str(ccdata$card)
```

```
##  Factor w/ 4 levels "Blue","Gold",..: 1 1 1 1 1 1 2 4 1 1 ...
```

```
#4 levels are returned
```

7/20: dependents

This shows the number of dependents the customer has reported.

```
unique(ccdata$dependents)
```

```
## [1] 3 5 4 2 0 1
```

```
#There are 6 unique fields
ccdata$dependents = factor(ccdata$dependents)
#We should see a return factor of 6
str(ccdata$dependents)
```

```
##  Factor w/ 6 levels "0","1","2","3",..: 4 6 4 5 4 3 5 1 4 3 ...
```

```
#6 levels are returned
```

8/20: products_num

```
unique(ccdata$products_num)
```

```
## [1] 5 6 4 3 2 1
```

```
#There are 6 unique fields
ccdata$products_num = factor(ccdata$products_num)
#We should see a return factor of 6
str(ccdata$products_num)
```

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

```
#6 levels are returned
```

9/20: contacts

This field shows the number of times the bank has contacted the customer.

```
unique(ccdata$contacts)
```

```
## [1] 3 2 0 1 4 5 6
```

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

```
##  Factor w/ 7 levels "0","1","2","3",..: 4 3 1 2 1 3 4 3 1 4 ...
```

```
#7 levels are returned
```

10/20: months_inactive

This field shows the number of months the customer has been inactive with the bank.

```
unique(ccdata$months_inactive)
```

```
## [1] 1 4 2 3 6 0 5
```

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

```
##  Factor w/ 7 levels "0","1","2","3",..: 2 2 2 5 2 2 2 3 3 4 ...
```

```
#7 levels are returned
```

11/20: age

This is the customers age.

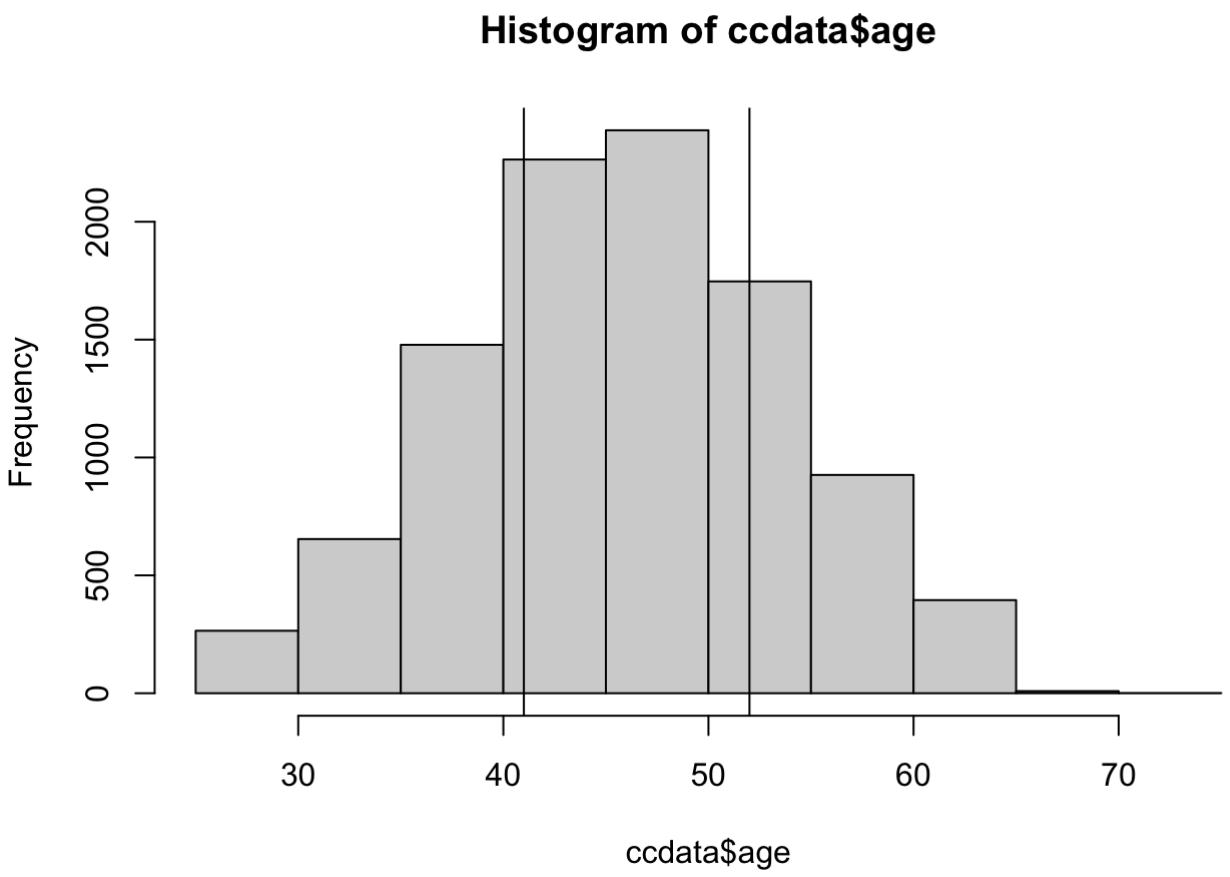
```
#We can see the customer ages can age from 26 to 73
min(ccdata$age)
```

```
## [1] 26
```

```
max(ccdata$age)
```

```
## [1] 73
```

```
#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','s
eventies'))

#Verify it is factorized
str(ccdata$age)
```

```
##  Factor w/ 6 levels "twenties","thirties",..: 3 3 4 2 2 3 4 2 2 3 ...
```

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

```
## [1] 0
```

12/20: months active

This field quantifies the number of months the customer has been on the books with the bank.

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

```
## [1] 13
```

```
max(ccdata$months_active)
```

```
## [1] 56
```

```
#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','sixt
ies','seventies'))

#Verify it was factorized
str(ccdata$months_active)
```

```
##  Factor w/ 7 levels "tens","twenties",..: 3 4 3 3 2 3 4 2 3 3 ...
```

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

```
## [1] 0
```

13/20: credit_limit

This field shows the credit limit of the customer in dollar amounts.

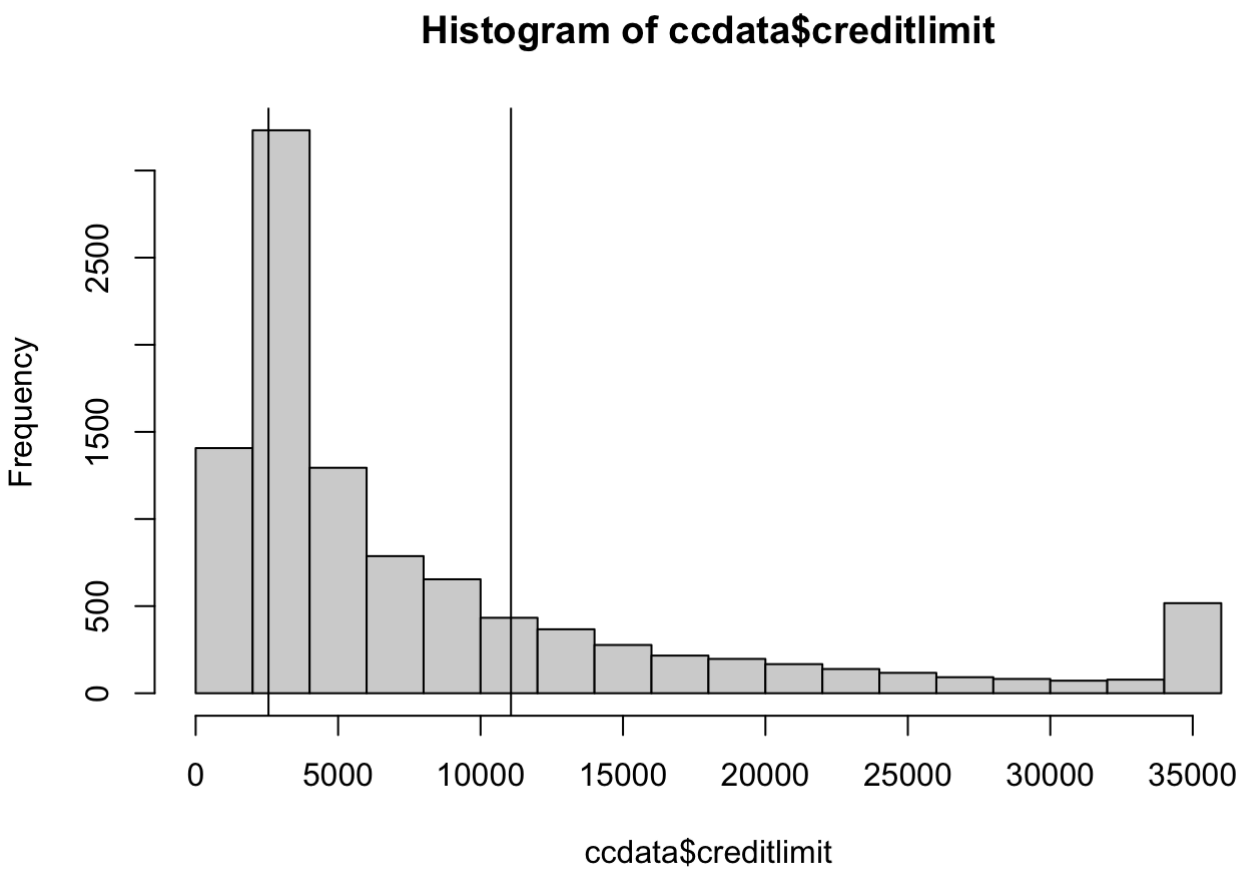
```
#Credit limit ranges from $1,438 to $34,516
min(ccdata$creditlimit)
```

```
## [1] 1438.3
```

```
max(ccdata$creditlimit)
```

```
## [1] 34516
```

```
#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)
```

```
##  Factor w/ 4 levels "Low","Medium",..: 2 1 1 1 1 1 4 4 3 2 ...
```

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

```
## [1] 0
```

14/20: revolving_bal

This field shows the revolving balance on the customers credit card in dollar amount.

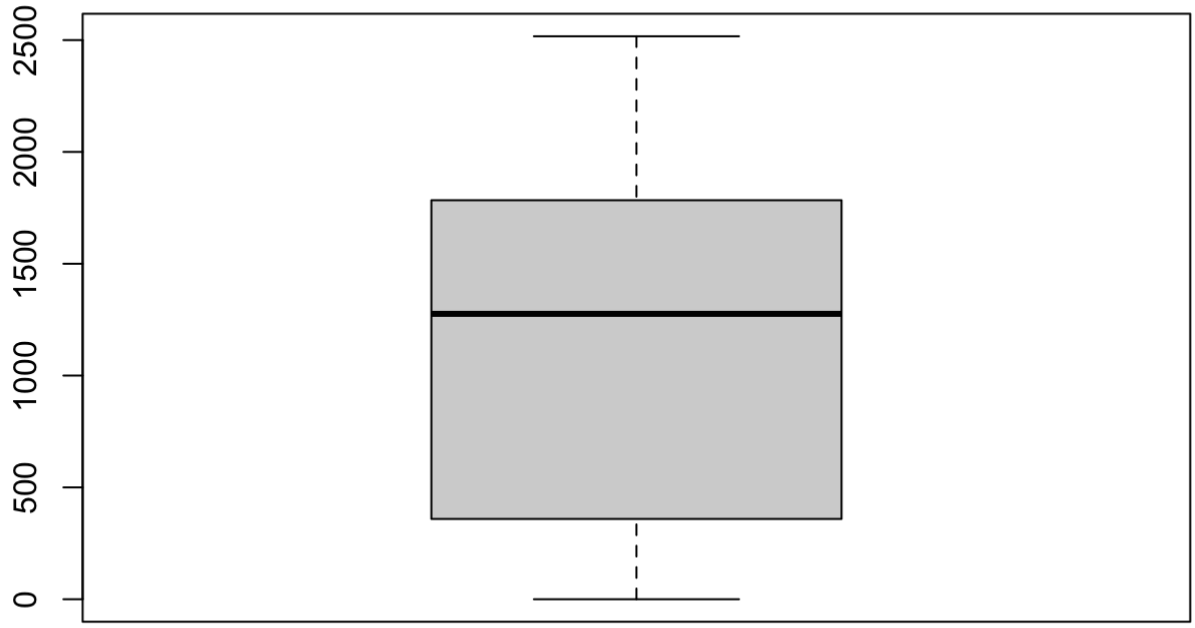
```
#The balance ranges from $0 to $2,517
min(ccdata$revolving_bal)
```

```
## [1] 0
```

```
max(ccdata$revolving_bal)
```

```
## [1] 2517
```

```
boxplot(ccdata$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)
```

```
##  Factor w/ 6 levels "0","500s","1000s",..: 2 3 1 6 1 3 6 4 6 4 ...
```

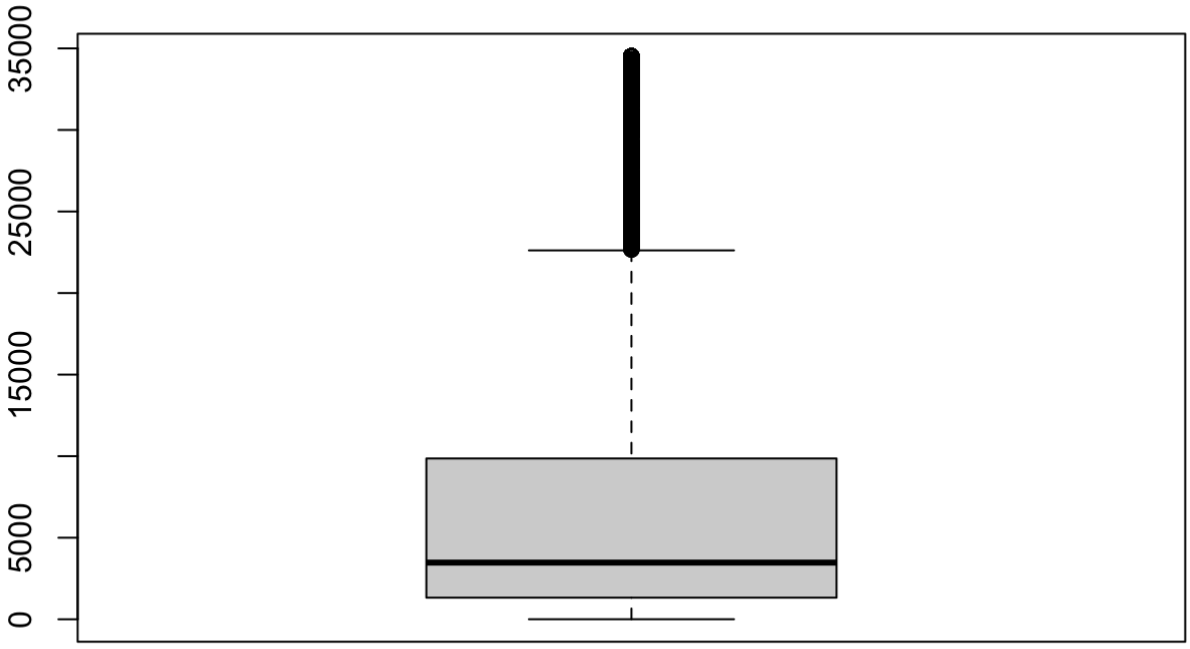
```
#Verify no NAs
(sum(is.na(ccdata$revolving_bal)))
```

```
## [1] 0
```

15/20: open_to_buy

This is the average credit line of the past 12 months.

```
boxplot(ccdata$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)
```

```
##  Factor w/ 6 levels "0","500s","1000s",..: 3 2 1 1 1 1 6 5 4 2 ...
```

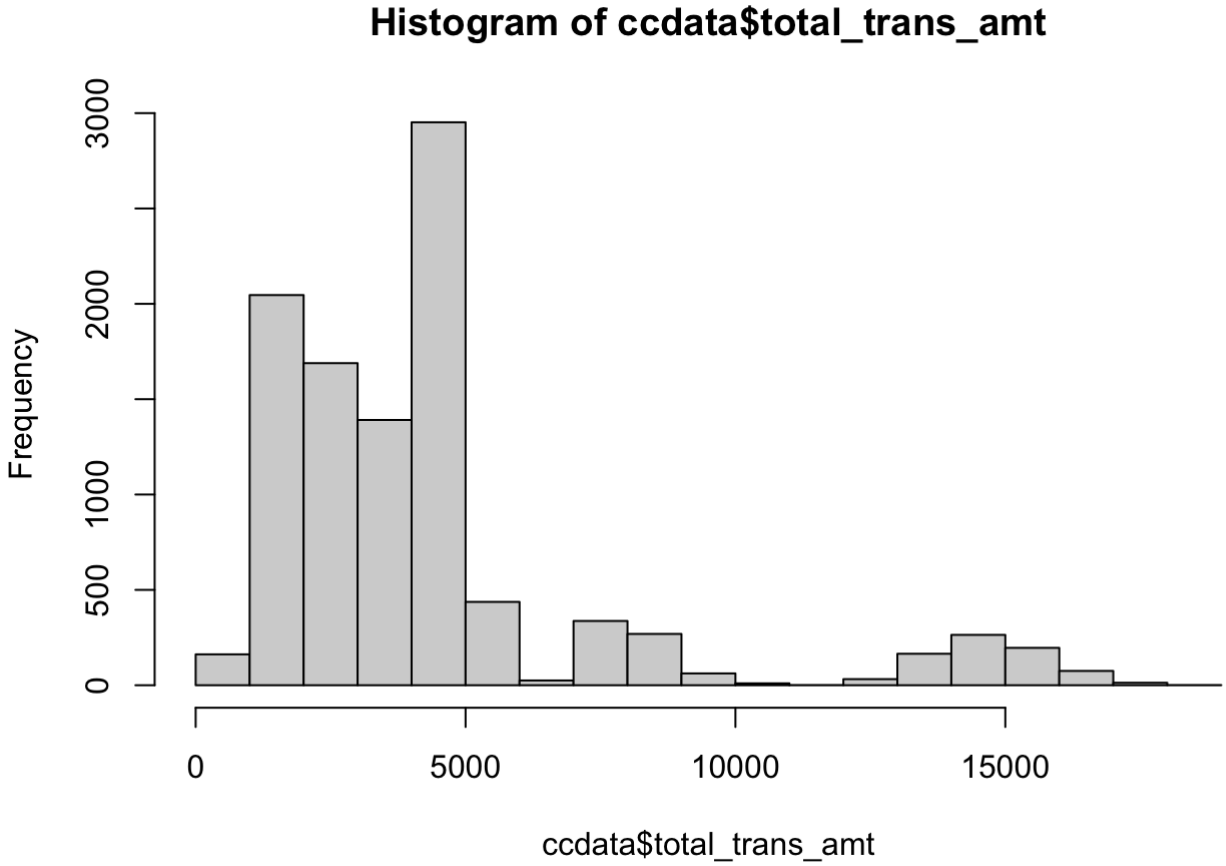
```
#Verify no NAs
(sum(is.na(ccdata$open_to_buy)))
```

```
## [1] 0
```

16/20: total_trans_amt

This is the total dollar amount of transactions in the last 12 months.

```
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)
```

```
##  Factor w/ 3 levels "Low","Medium",..: 1 1 1 1 1 1 1 1 1 1 ...
```

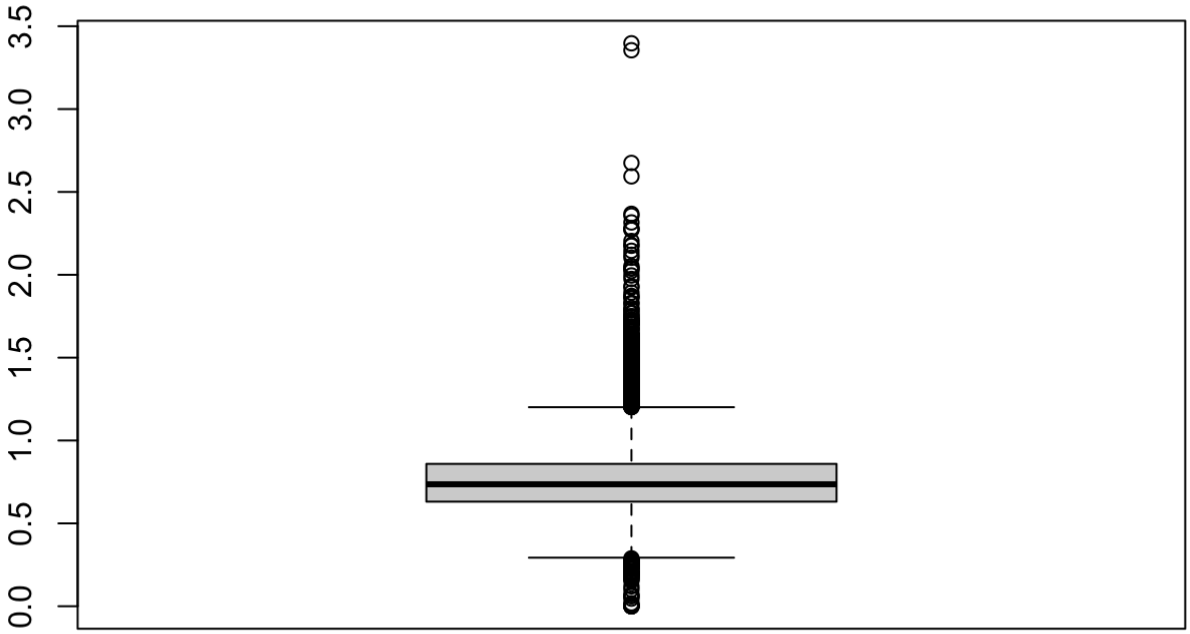
```
#Verify no NAs
(sum(is.na(ccdata$total_trans_amt)))
```

```
## [1] 0
```

17/20: amt_change_q4_q1

This is the change of dollar amount between Q4 over Q1.

```
boxplot(ccdata$amt_change_q4_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_am
tchange, width_amtchange),
                     labels=c('Low','Medium','High'))
#Verify factorization
str(ccdata$amt_change_q4_q1)
```

```
##  Factor w/ 3 levels "Low","Medium",..: 2 2 3 2 2 2 2 2 3 2 ...
```

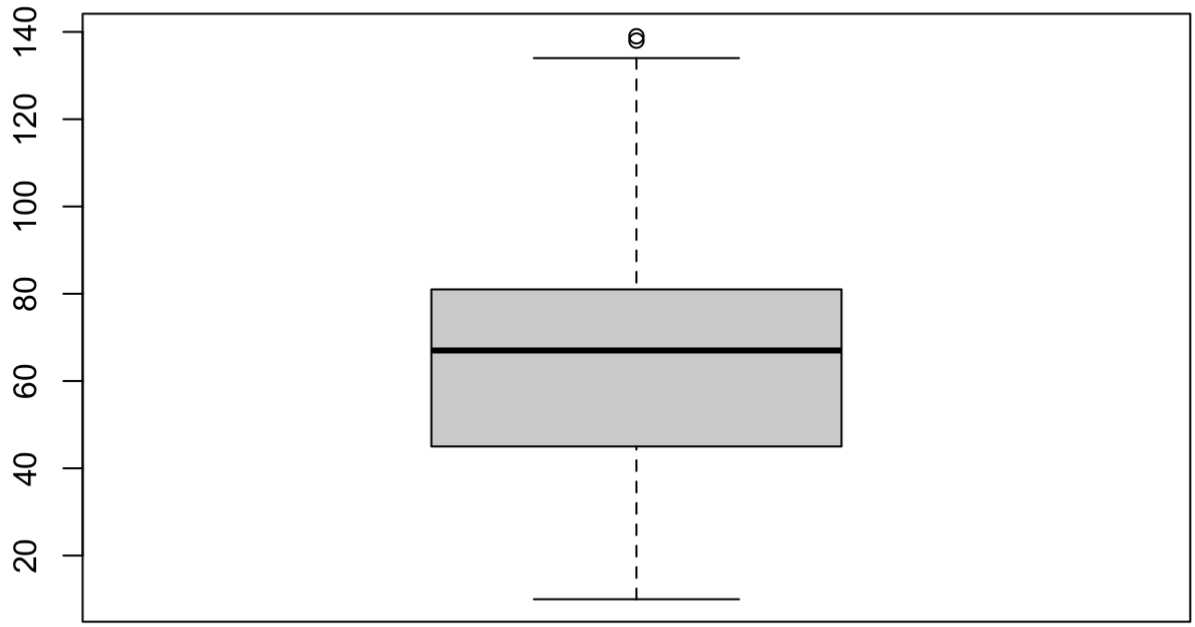
```
#Verify no NA's
sum(is.na(ccdata$amt_change_q4_q1))
```

```
## [1] 0
```

18/20: total_trans_cnt

This is the total number of tranaactions in the last 12 months.

```
boxplot(ccdata$total_trans_cnt)
```




```
#Factorization
ccdata$total_trans_cnt = cut(ccdata$total_trans_cnt, breaks = c(0,20,40,60,80,100,120,14
0,Inf),
                        labels=c('Under 20','20s','40s','60s','80s','100s','120s','140
s'))
#Verify
str(ccdata$total_trans_cnt)
```

```
##  Factor w/ 8 levels "Under 20","20s",..: 3 2 1 1 2 2 2 2 2 2 ...
```

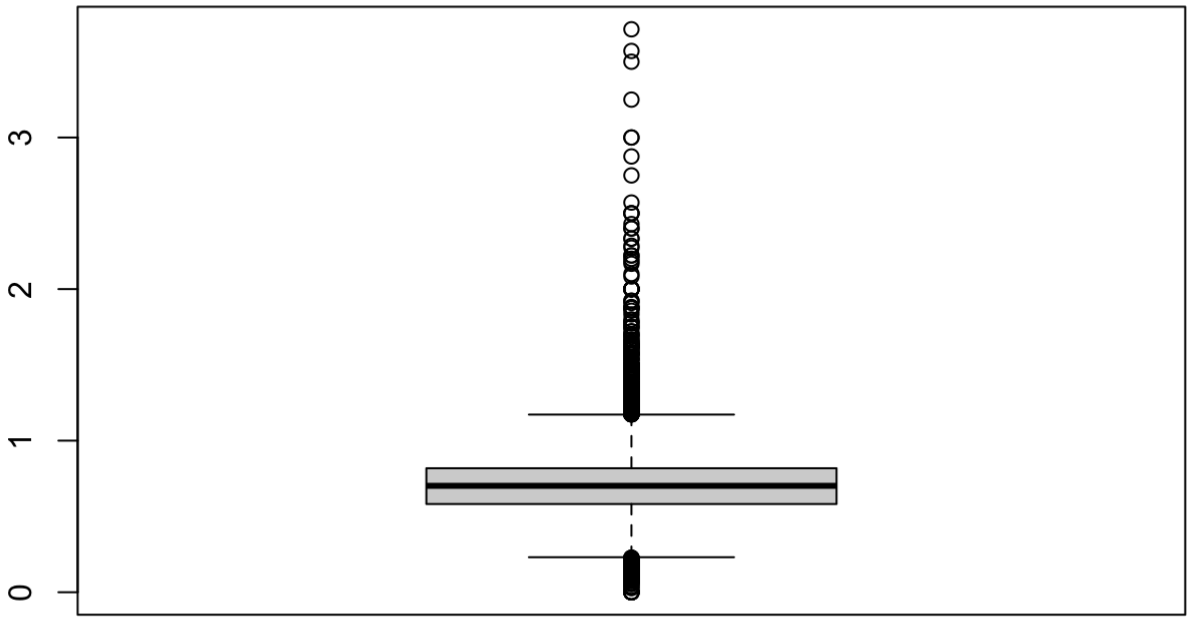
```
#Verify no NA's
sum(is.na(ccdata$total_trans_cnt))
```

```
## [1] 0
```

19/20: cnt_change_q4_q1

This is the change of transaction count between Q4 over Q1.

```
boxplot(ccdata$cnt_change_q4_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_cnt
change, width_cntchange),
                      labels=c('Low','Medium','High'))
str(ccdata$cnt_change_q4_q1)
```

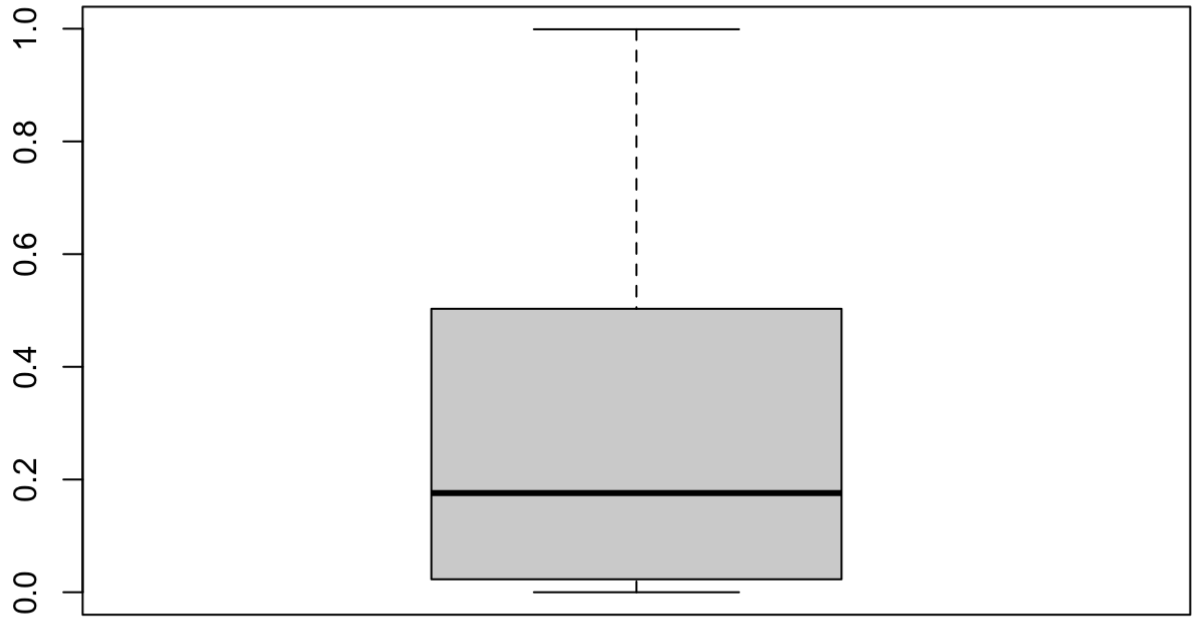
```
##  Factor w/ 3 levels "Low","Medium",..: 2 3 2 2 3 1 1 1 1 1 ...
```

```
#Verify no NA's
sum(is.na(ccdata$cnt_change_q4_q1))
```

```
## [1] 0
```

20/20: utilization

```
boxplot(ccdata$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_utilizatio
n, width_utilization),
                      labels=c('Low','Medium','High'))
#Verify
str(ccdata$utilization)
```

```
##  Factor w/ 3 levels "Low","Medium",..: 1 1 1 3 1 1 1 1 1 1 ...
```

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

```
## [1] 0
```

One final check to verify all the variables are factorized.

```
str(ccdata)
```

```
## '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 ...
##  $ total_trans_cnt : Factor w/ 8 levels "Under 20","20s",..: 3 2 1 1 2 2 2 2 2 2 ...
##  $ cnt_change_q4_q1: Factor w/ 3 levels "Low","Medium",..: 2 3 2 2 3 1 1 1 1 1 ...
##  $ utilization     : Factor w/ 3 levels "Low","Medium",..: 1 1 1 3 1 1 1 1 1 1 ...
```

Exploratory Data Analysis

Correlation Matrix

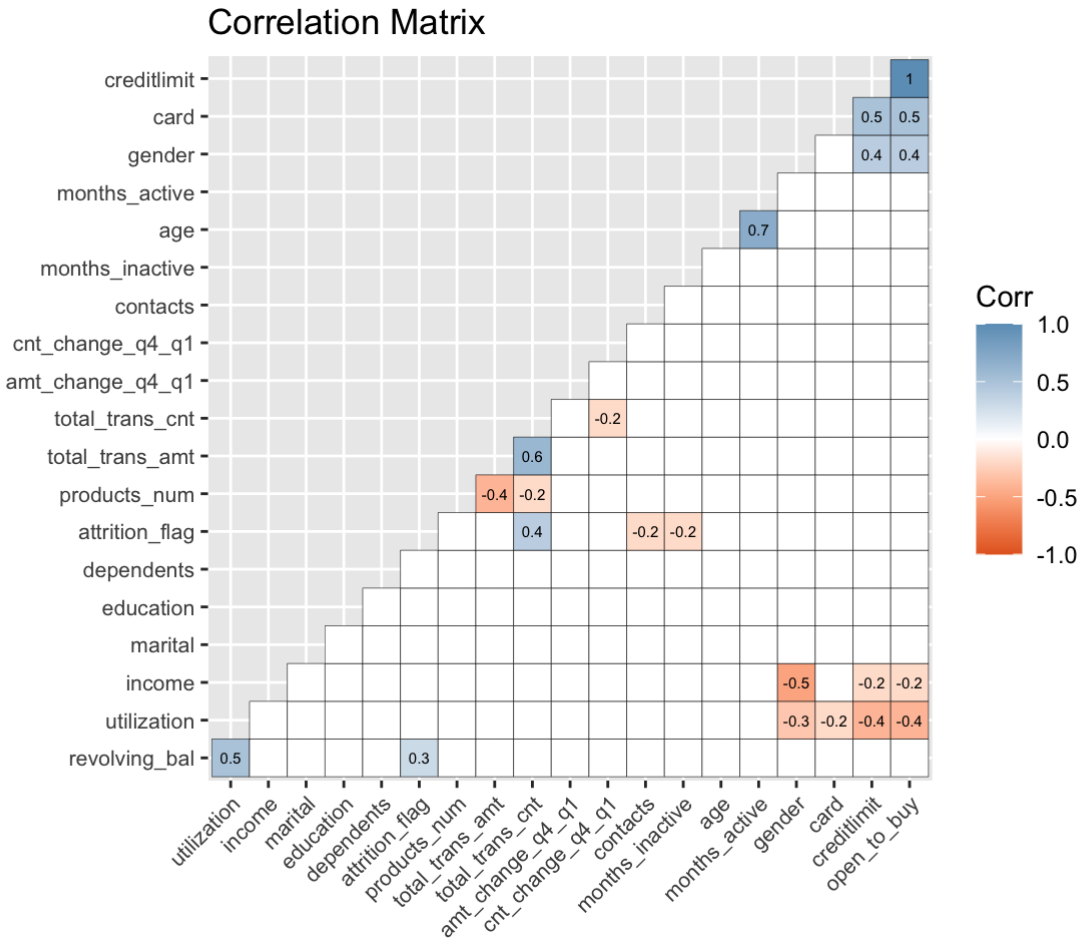
The correlation matrix below converted all the factors in the data set and set them as numeric. Then the p-values were derived from the correlation matrix. The visual below is showing only the lower half of the correlation matrix since the top half would be repeated values. Also, insignificant p-values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

```
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("#E46726", "white", "#6D9EC1"))
```

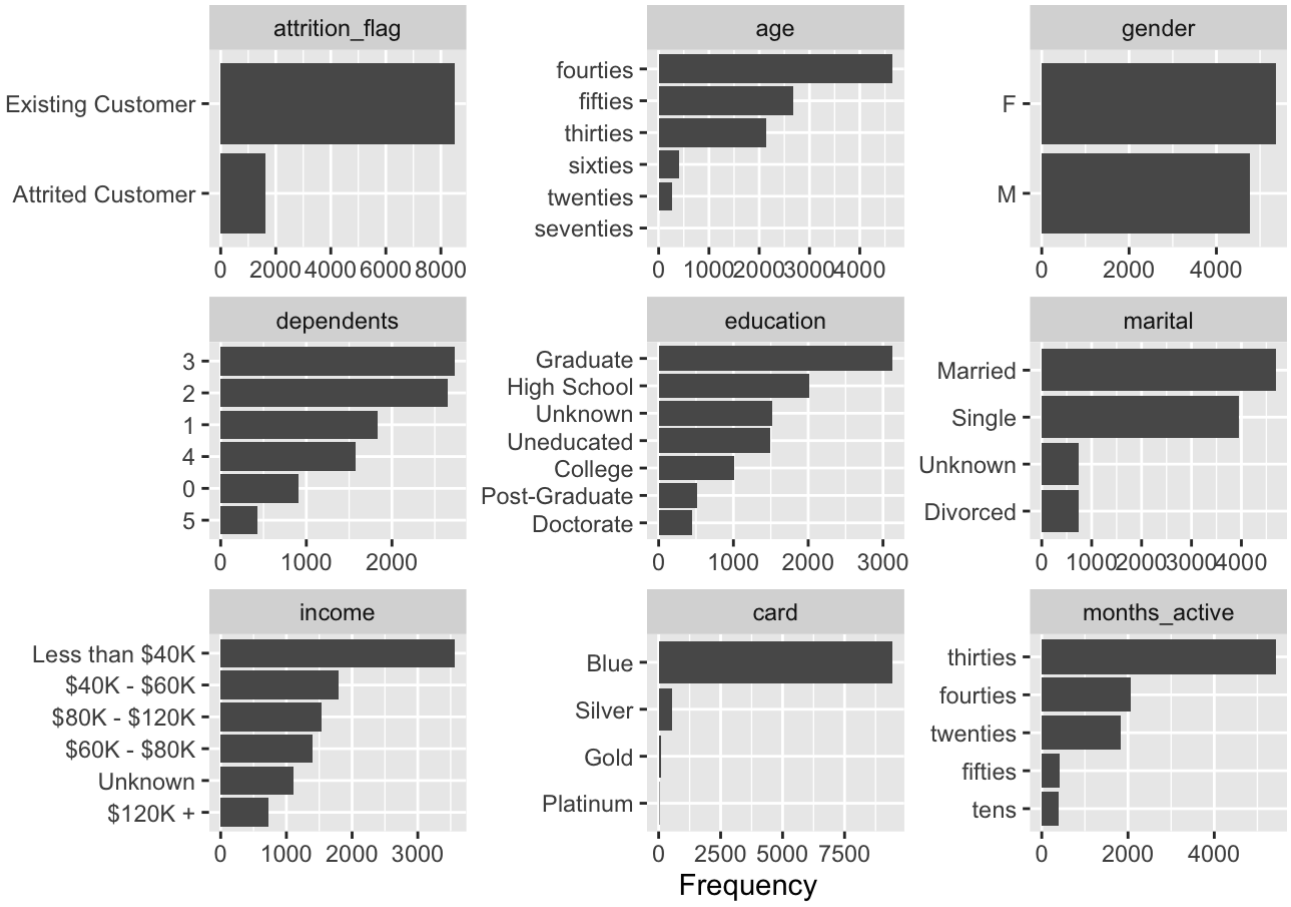


Data Plot

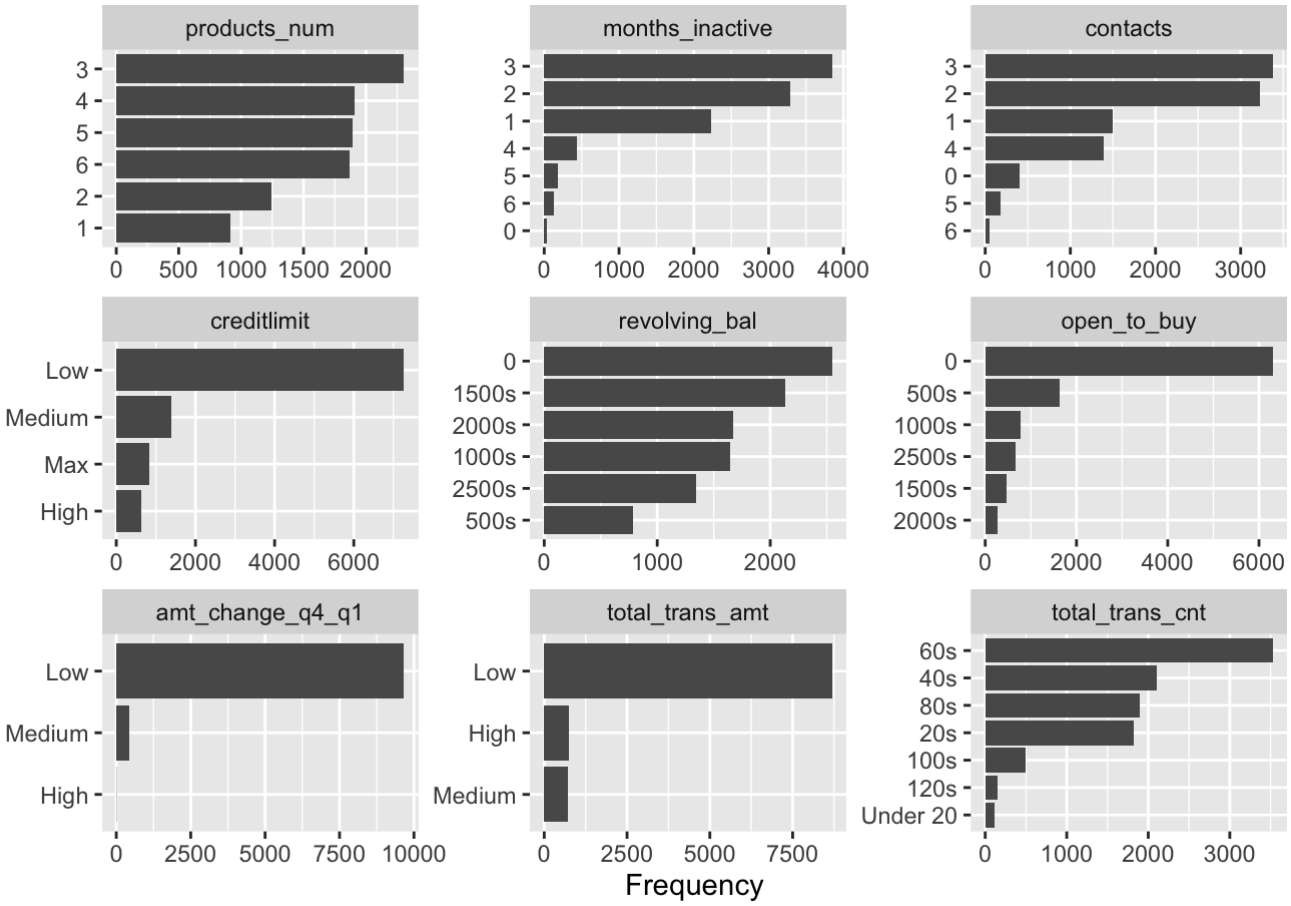
After the data cleaning, the data is plotted to see the frequency and dispersion of data.

We can see the majority of our data set contains current customers.

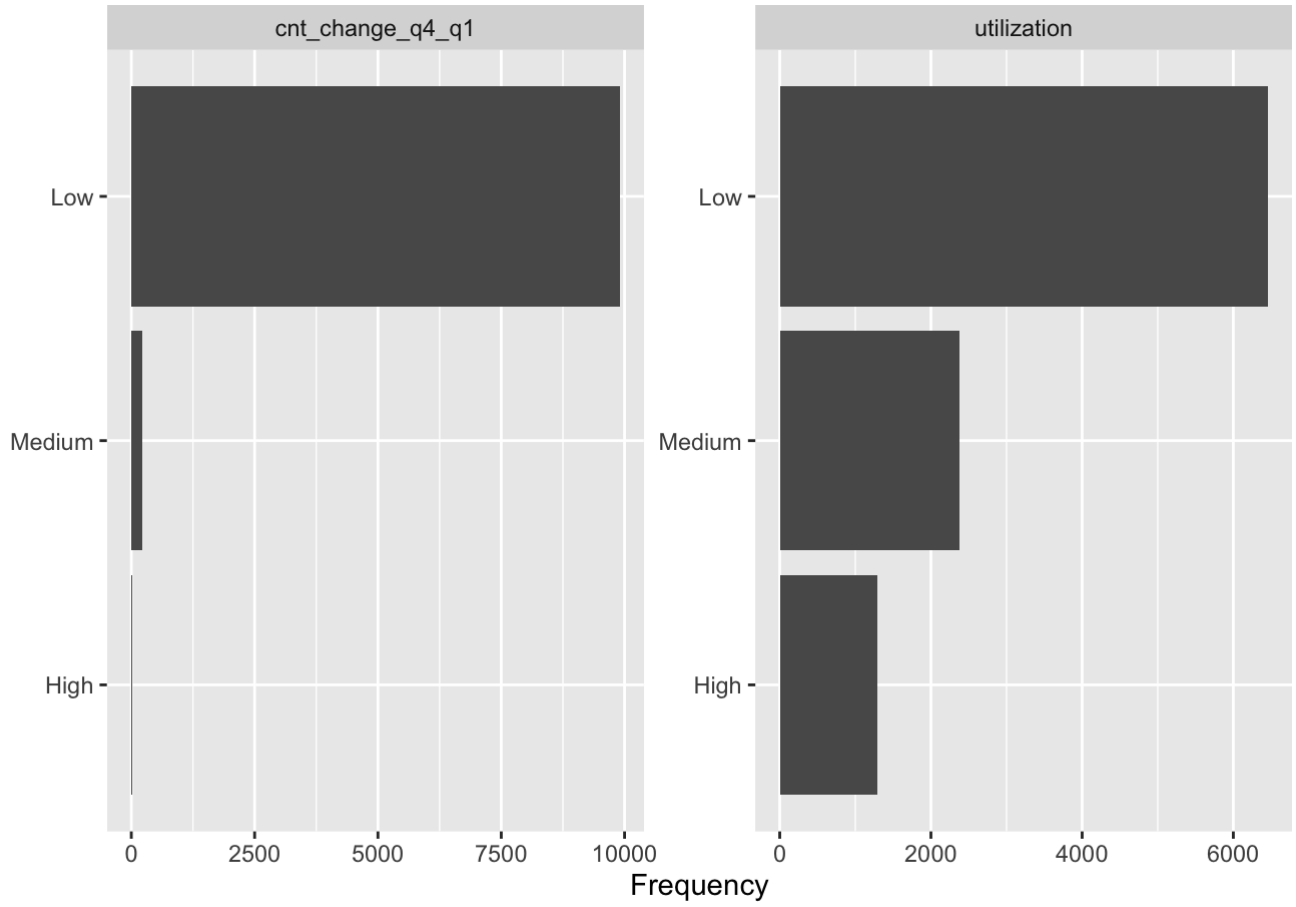
```
plot_bar(ccdata)
```


attrition_flag
Existing Customer
Attrited Customer
0 2000 4000 6000 8000
age
fourties
fifties
thirties
sixties
twenties
seventies
0 1000 2000 3000 4000
gender
F
M
0 2000 4000
dependents
3
2
1
4
0
5
0 1000 2000
education
Graduate
High School
Unknown
Uneducated
College
Post-Graduate
Doctorate
0 1000 2000 3000
marital
Married
Single
Unknown
Divorced
0 1000 2000 3000 4000
income
Less than $40K
$40K - $60K
$80K - $120K
$60K - $80K
Unknown
$120K +
0 1000 2000 3000
card
Blue
Silver
Gold
Platinum
0 2500 5000 7500
months_active
thirties
fourties
twenties
fifties
tens
0 2000 4000
Frequency

Page 1


products_num
3
4
5
6
2
1
0 500 1000 1500 2000
months_inactive
3
2
1
4
5
6
0
0 1000 2000 3000 4000
contacts
3
2
1
4
0
5
6
0 1000 2000 3000
creditlimit
Low
Medium
Max
High
0 2000 4000 6000
revolving_bal
0
1500s
2000s
1000s
2500s
500s
0 1000 2000
open_to_buy
0
500s
1000s
2500s
1500s
2000s
0 2000 4000 6000
amt_change_q4_q1
Low
Medium
High
0 2500 5000 7500 10000
total_trans_amt
Low
High
Medium
0 2500 5000 7500
total_trans_cnt
60s
40s
80s
20s
100s
120s
Under 20
0 1000 2000 3000
Frequency

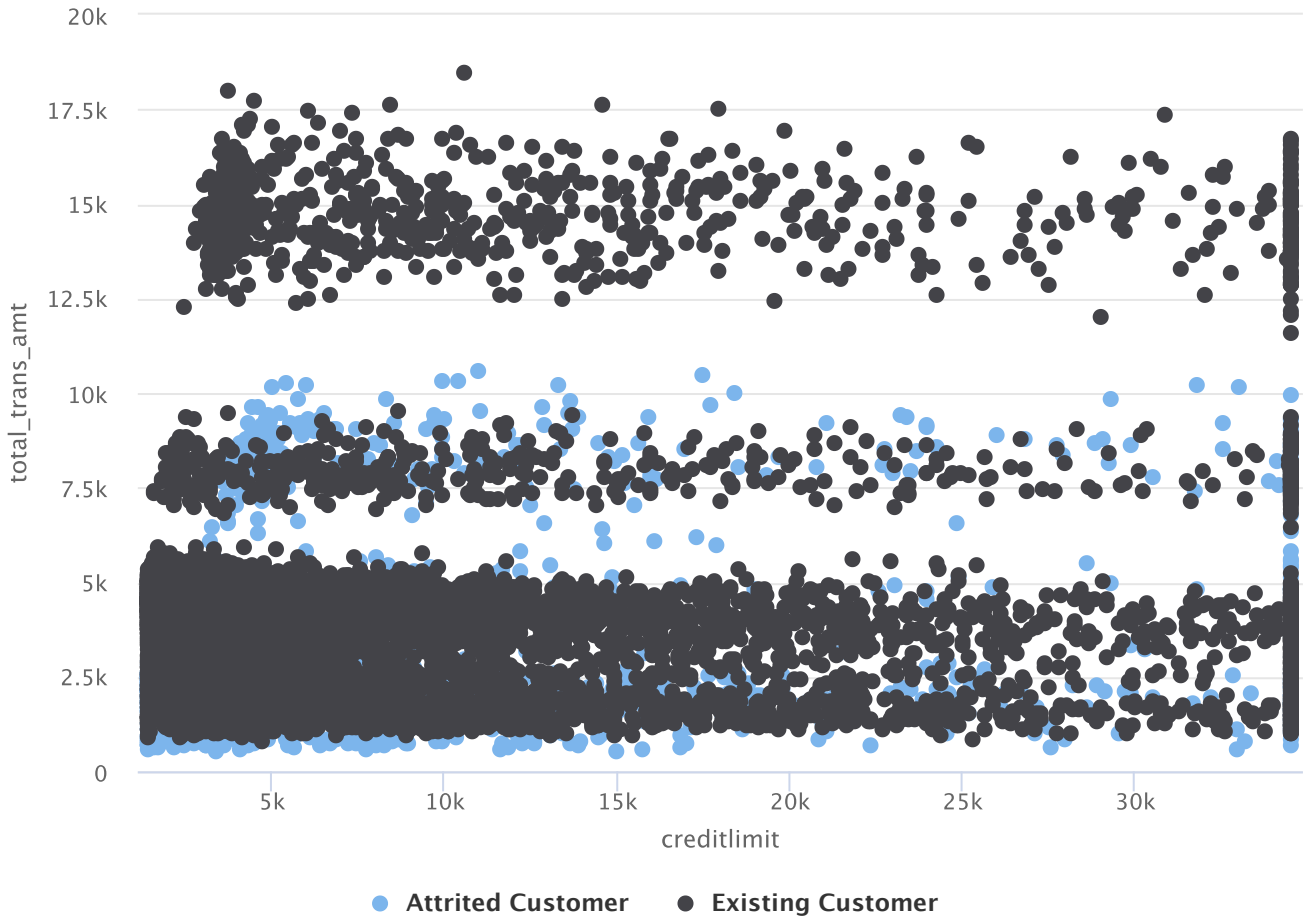
Page 2



Attrition Flag Exploration

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

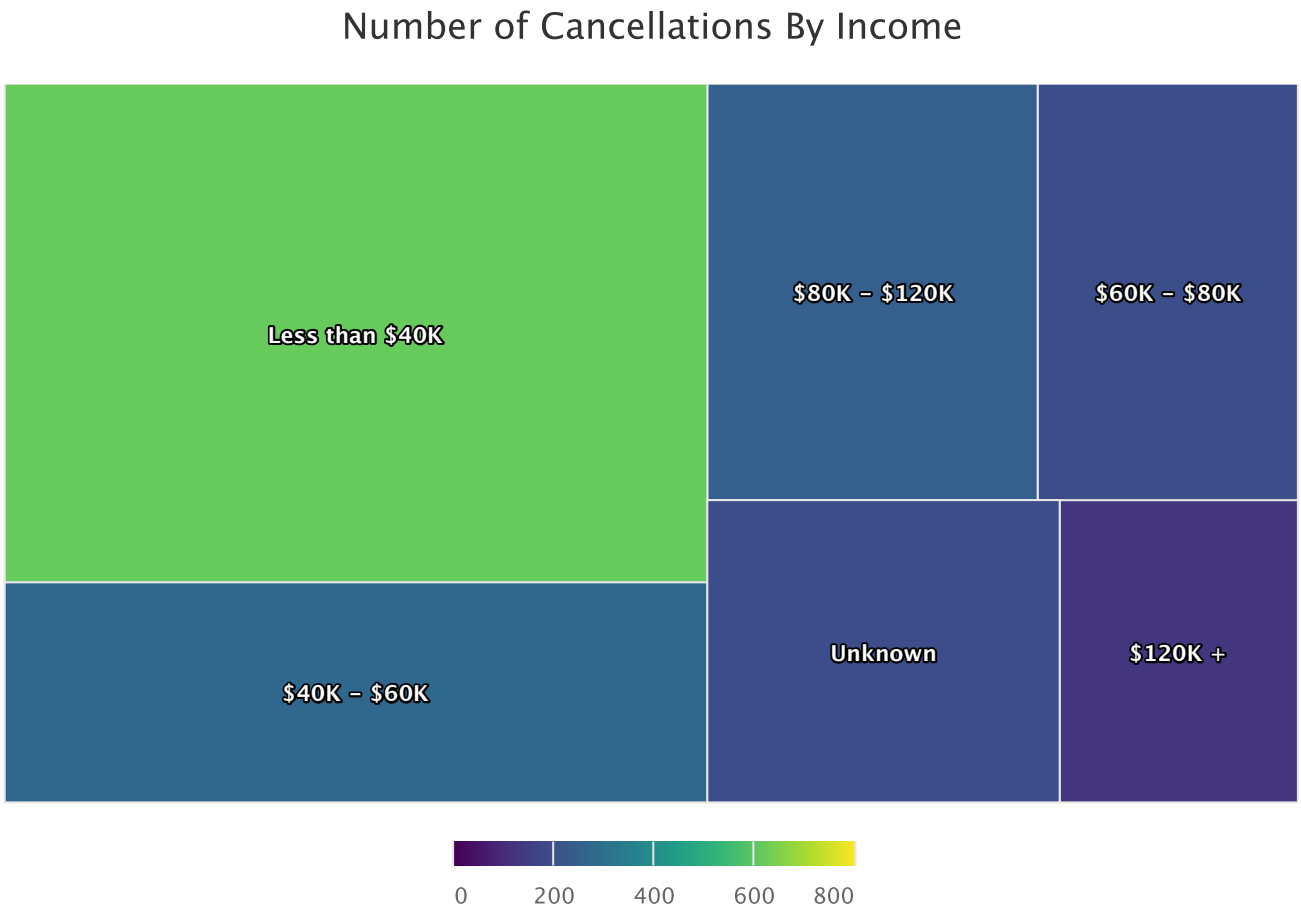
```
hchart(
  ccdata_cor,
  "scatter",
  hcaes(x = creditlimit, y = total_trans_amt, group = attrition_flag)
)
```



Cancellations by Attributes

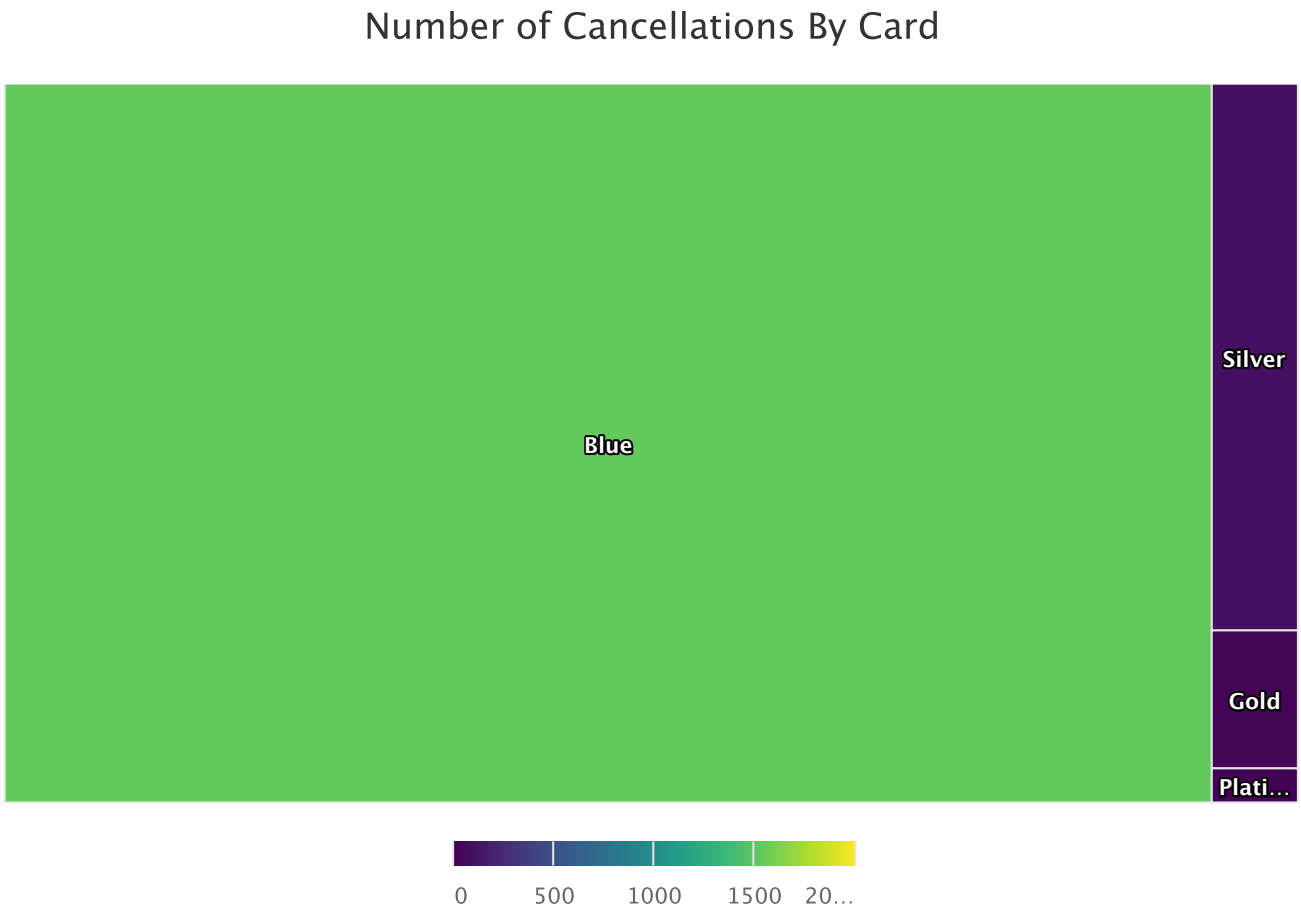
We can see a quick overview of attrited customers and their income levels.

```
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")
```



Blue cards is the most common card cancelled.

```
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")
```



Data Analysis Methods

1. Association Rule Mining 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.

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.

Card Type

Through the exploratory data analysis, it is noted that the blue card is the most common card among all cards offered at the bank. Using apriori method will analyze the customer base that uses the blue card.

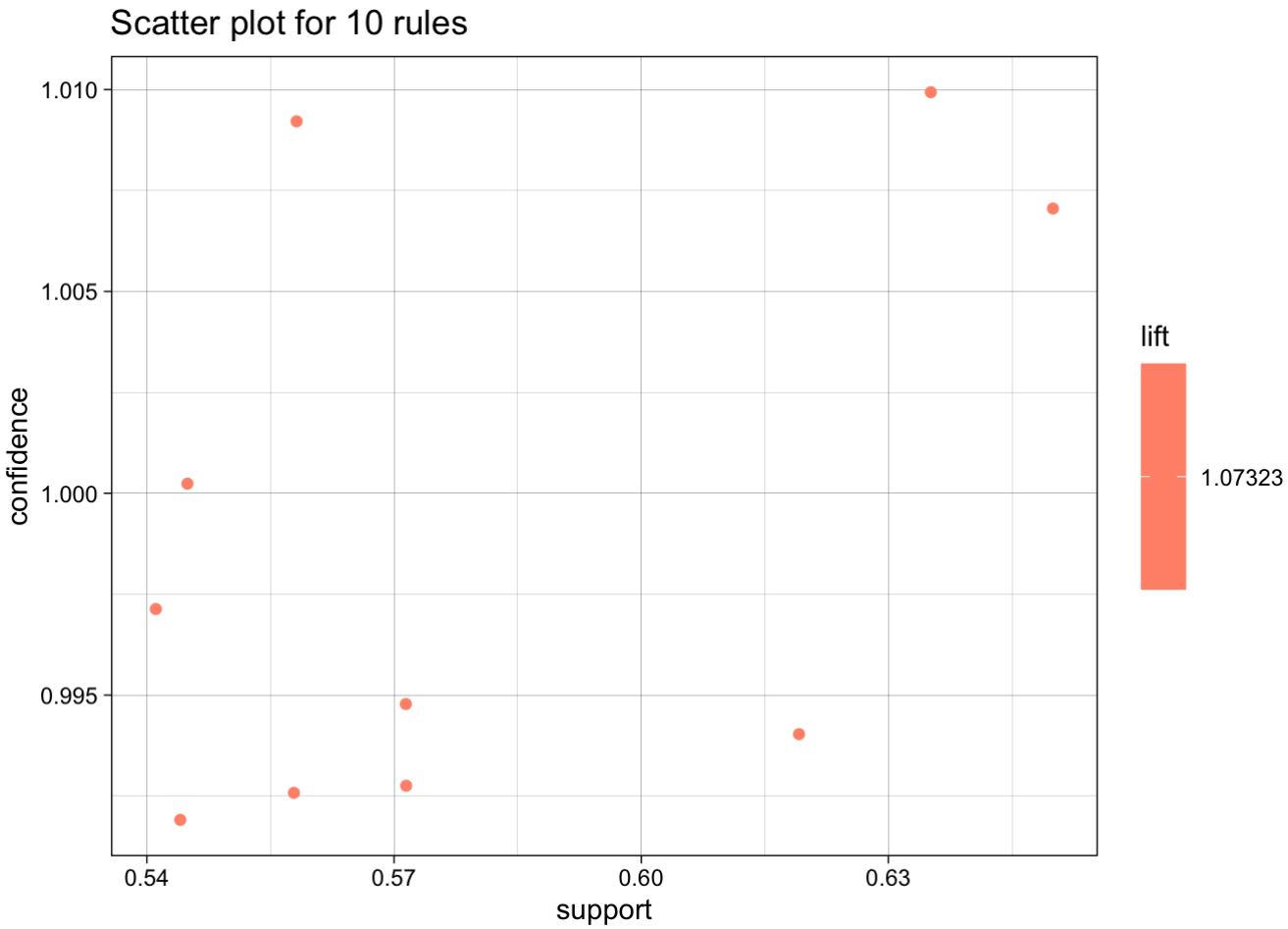
```
bluecard = apriori(data=ccdata, parameter=list(supp=0.5,conf = 0.9),
               appearance = list(default="lhs",rhs="card=Blue"),
               control = list(verbose=F))
bluecard = sort(bluecard, decreasing=TRUE,by="confidence")
inspect(bluecard[1:10])
```

```
##      lhs                                   rhs           support confidence  coverag
e    lift count
## [1]  {open_to_buy=0,
##       total_trans_amt=Low}              => {card=Blue} 0.5714427          1 0.571442
7 1.07323  5787
## [2]  {creditlimit=Low,
##       total_trans_amt=Low}              => {card=Blue} 0.6500444          1 0.650044
4 1.07323  6583
## [3]  {creditlimit=Low,
##       open_to_buy=0,
##       total_trans_amt=Low}              => {card=Blue} 0.5714427          1 0.571442
7 1.07323  5787
## [4]  {open_to_buy=0,
##       amt_change_q4_q1=Low,
##       total_trans_amt=Low}              => {card=Blue} 0.5448800          1 0.544880
0 1.07323  5518
## [5]  {open_to_buy=0,
##       total_trans_amt=Low,
##       cnt_change_q4_q1=Low}             => {card=Blue} 0.5582107          1 0.558210
7 1.07323  5653
## [6]  {attrition_flag=Existing Customer,
##       creditlimit=Low,
##       total_trans_amt=Low}              => {card=Blue} 0.5412264          1 0.541226
4 1.07323  5481
## [7]  {creditlimit=Low,
##       amt_change_q4_q1=Low,
##       total_trans_amt=Low}              => {card=Blue} 0.6185445          1 0.618544
5 1.07323  6264
## [8]  {creditlimit=Low,
##       total_trans_amt=Low,
##       cnt_change_q4_q1=Low}             => {card=Blue} 0.6344426          1 0.634442
6 1.07323  6425
## [9]  {creditlimit=Low,
##       open_to_buy=0,
##       amt_change_q4_q1=Low,
##       total_trans_amt=Low}              => {card=Blue} 0.5448800          1 0.544880
0 1.07323  5518
## [10] {creditlimit=Low,
##       open_to_buy=0,
##       total_trans_amt=Low,
##       cnt_change_q4_q1=Low}             => {card=Blue} 0.5582107          1 0.558210
7 1.07323  5653
```

```
plot(bluecard[1:10], method='scatterplot',interactive=FALSE )
```

```
## Warning in plot.rules(bluecard[1:10], method = "scatterplot", interactive
## = FALSE): The parameter interactive is deprecated. Use engine='interactive'
## instead.
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



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 are between 50-65%.

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

```
## Warning in plot.rules(bluecard[1:10], method = "graph", interactive = FALSE):
## The parameter interactive is deprecated. Use engine='interactive' instead.
```

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 there credit limit is already low. In this scenario, they are also likely to be an existing customer with a blue card.

Existing Customer

Apriori to explore the support and confidence of Attrited Customer.

```
existingcust<-apriori(data=ccdata, parameter=list(supp=0.03,conf = 0.9),
               appearance = list(default="lhs",rhs="attrition_flag=Existing Customer"),
               control = list(verbose=F))
existingcust<-sort(existingcust, decreasing=TRUE,by="lift")
inspect(existingcust[1:10])
```

```
##      lhs                          rhs                                    support confide
nce   coverage     lift count
## [1]  {total_trans_cnt=100s}  => {attrition_flag=Existing Customer} 0.04947171
1 0.04947171 1.191412   501
## [2]  {total_trans_amt=High}  => {attrition_flag=Existing Customer} 0.07277575
1 0.07277575 1.191412   737
## [3]  {total_trans_amt=High,
##       total_trans_cnt=100s}  => {attrition_flag=Existing Customer} 0.04246075
1 0.04246075 1.191412   430
## [4]  {gender=M,
##       total_trans_cnt=100s}  => {attrition_flag=Existing Customer} 0.03011751
1 0.03011751 1.191412   305
## [5]  {total_trans_cnt=100s,
##       utilization=Low}       => {attrition_flag=Existing Customer} 0.03910339
1 0.03910339 1.191412   396
## [6]  {card=Blue,
##       total_trans_cnt=100s}  => {attrition_flag=Existing Customer} 0.03831342
1 0.03831342 1.191412   388
## [7]  {amt_change_q4_q1=Low,
##       total_trans_cnt=100s}  => {attrition_flag=Existing Customer} 0.04917547
1 0.04917547 1.191412   498
## [8]  {total_trans_cnt=100s,
##       cnt_change_q4_q1=Low}  => {attrition_flag=Existing Customer} 0.04947171
1 0.04947171 1.191412   501
## [9]  {age=fourties,
##       total_trans_amt=High}  => {attrition_flag=Existing Customer} 0.03288239
1 0.03288239 1.191412   333
## [10] {marital=Married,
##       total_trans_amt=High}  => {attrition_flag=Existing Customer} 0.03347487
1 0.03347487 1.191412   339
```

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 customer 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.

```
#visualization of rules
plot(existingcust[1:5],method="graph",engine='interactive',shading="lift")
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)),
## stdout = TRUE): running command ''/usr/bin/otool' -L
## '/Library/Frameworks/R.framework/Resources/library/tcltk/libs//tcltk.so'' had
## status 1
```

Attrited Customer

Apriori to explore the support and confidence of Attrited Customer.

```
attritedcust<-apriori(data=ccdata, parameter=list(supp=0.002,conf = 0.9),
               appearance = list(default="lhs",rhs="attrition_flag=Attrited Customer"),
               control = list(verbose=F))
attritedcust<-sort(attritedcust, decreasing=TRUE,by="support")
inspect(attritedcust[1:10])
```

```
##      lhs                          rhs                                    support confide
nce   coverage     lift count
## [1]  {gender=F,
##       months_inactive=3,
##       revolving_bal=0,
##       open_to_buy=0,
##       amt_change_q4_q1=Low,
##       total_trans_cnt=40s}   => {attrition_flag=Attrited Customer} 0.01412067  0.9108
280 0.01550311 5.669303   143
## [2]  {gender=F,
##       months_inactive=3,
##       revolving_bal=0,
##       open_to_buy=0,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low}  => {attrition_flag=Attrited Customer} 0.01412067  0.9050
633 0.01560186 5.633421   143
## [3]  {products_num=2,
##       amt_change_q4_q1=Low,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low,
##       utilization=Low}       => {attrition_flag=Attrited Customer} 0.01165202  0.9007
634 0.01293572 5.606657   118
## [4]  {products_num=1,
##       total_trans_cnt=40s}   => {attrition_flag=Attrited Customer} 0.01086205  0.9016
393 0.01204700 5.612109   110
## [5]  {card=Blue,
##       products_num=2,
##       amt_change_q4_q1=Low,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low,
##       utilization=Low}       => {attrition_flag=Attrited Customer} 0.01086205  0.9016
393 0.01204700 5.612109   110
## [6]  {products_num=1,
##       amt_change_q4_q1=Low,
##       total_trans_cnt=40s}   => {attrition_flag=Attrited Customer} 0.01076331  0.9008
264 0.01194826 5.607049   109
## [7]  {products_num=1,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low}  => {attrition_flag=Attrited Customer} 0.01076331  0.9008
264 0.01194826 5.607049   109
## [8]  {products_num=1,
##       amt_change_q4_q1=Low,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low}  => {attrition_flag=Attrited Customer} 0.01066456  0.9000
000 0.01184951 5.601905   108
## [9]  {products_num=2,
##       creditlimit=Low,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low}  => {attrition_flag=Attrited Customer} 0.01056581  0.9067
797 0.01165202 5.644104   107
## [10] {card=Blue,
##       products_num=2,
```

```
##       creditlimit=Low,
##       total_trans_cnt=40s,
##       cnt_change_q4_q1=Low}  => {attrition_flag=Attrited Customer} 0.01056581  0.9067
797 0.01165202 5.644104   107
```

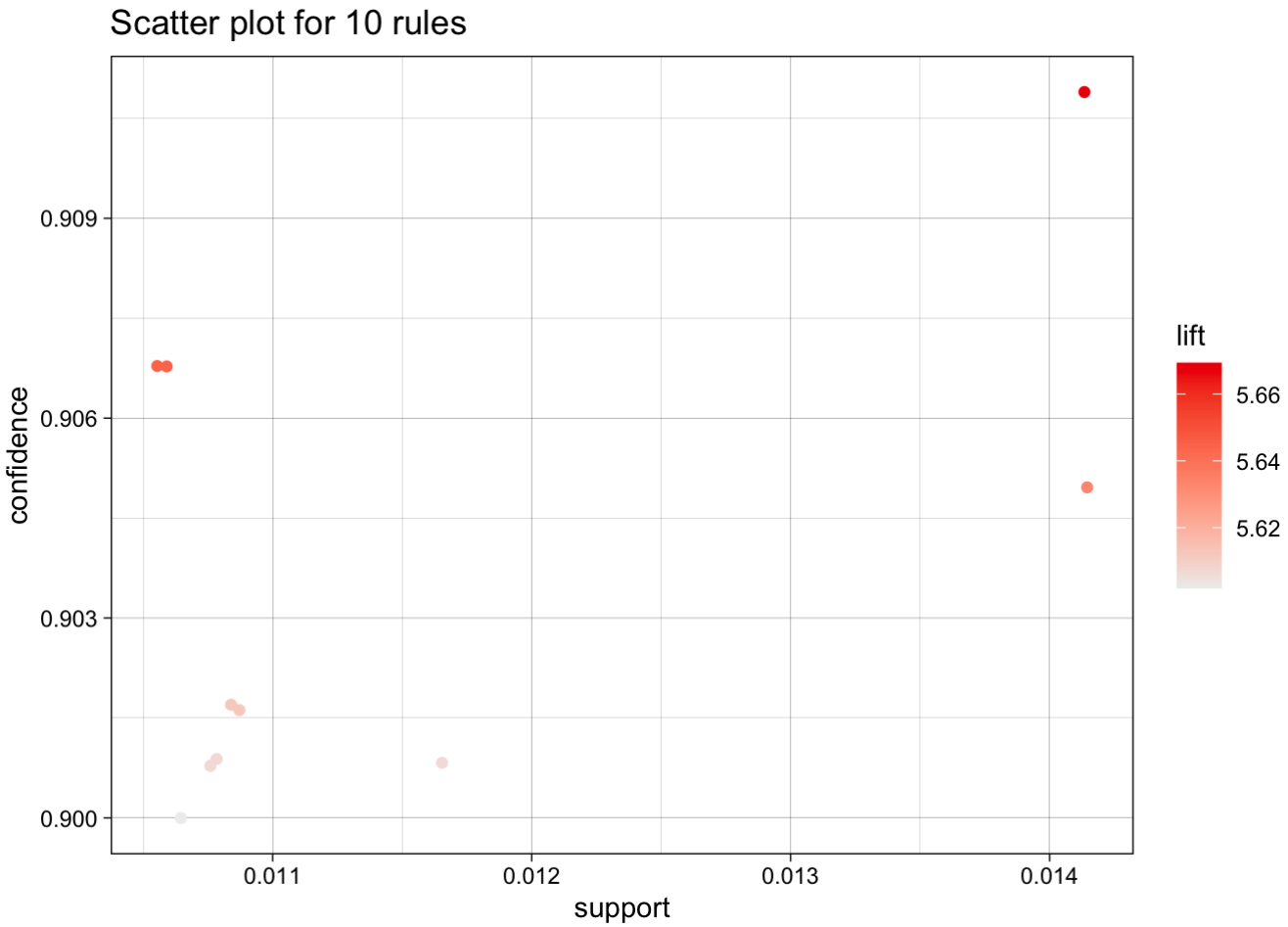
For the top 10 rules on Attrited Customers: a) Customers with lesser contacts b) Customers with transaction counts on the lower side(forties) c)Customers relatively new (active months in the thirties) d)Inactive Customers

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

```
#Visualizing the rules
plot(attritedcust[1:5],method="graph",engine='interactive',shading="lift")
```

```
plot(attritedcust[1:10], method='scatterplot')
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



As we apply additional techniques on the data, we will explore further on these criterias.

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.

```
ccdata_dt<-ccdata
```

#split the data 70-30%

```
split1<- sample(c(rep(0, 0.7 * nrow(ccdata_dt)), rep(1, 0.3 * nrow(ccdata_dt))))
```

```
table(split1)
```

```
## split1
##    0    1
## 7088 3038
```

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

```
str(train)
```

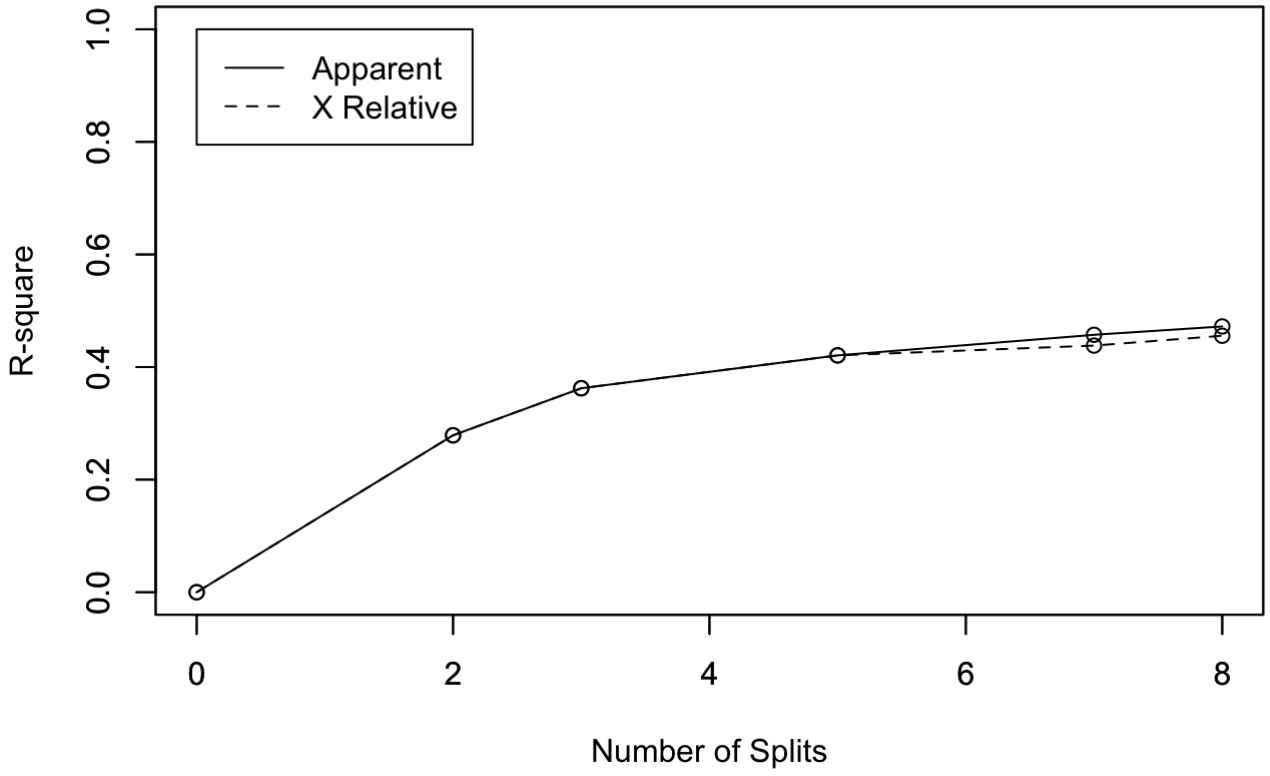
```
## 'data.frame':    7089 obs. of  20 variables:
##  $ attrition_flag  : Factor w/ 2 levels "Attrited Customer",..: 2 2 2 2 2 2 2 2 2 2
...
##  $ age             : Factor w/ 6 levels "twenties","thirties",..: 3 3 4 2 2 4 2 3 5 4
...
##  $ gender          : Factor w/ 2 levels "F","M": 2 1 2 1 2 2 2 2 2 2 ...
##  $ dependents      : Factor w/ 6 levels "0","1","2","3",..: 4 6 4 5 4 5 1 6 2 2 ...
##  $ education       : Factor w/ 7 levels "College","Doctorate",..: 4 3 3 4 6 7 4 6 7 1
...
##  $ marital         : Factor w/ 4 levels "Divorced","Married",..: 2 3 2 4 2 2 4 4 2 3
...
##  $ income          : Factor w/ 6 levels "$120K +","$40K - $60K",..: 3 5 4 5 3 1 3 1 2
4 ...
##  $ card            : Factor w/ 4 levels "Blue","Gold",..: 1 1 1 1 1 2 4 1 1 1 ...
##  $ months_active   : Factor w/ 7 levels "tens","twenties",..: 3 4 3 3 2 4 2 3 5 3 ...
##  $ products_num    : Factor w/ 6 levels "1","2","3","4",..: 5 6 4 3 5 6 2 5 6 3 ...
##  $ months_inactive : Factor w/ 7 levels "0","1","2","3",..: 2 2 2 5 2 2 3 4 3 7 ...
##  $ contacts        : Factor w/ 7 levels "0","1","2","3",..: 4 3 1 2 1 4 3 3 4 1 ...
##  $ creditlimit     : Factor w/ 4 levels "Low","Medium",..: 2 1 1 1 1 4 4 1 1 2 ...
##  $ revolving_bal   : Factor w/ 6 levels "0","500s","1000s",..: 2 3 1 6 1 6 4 4 4 1
...
##  $ open_to_buy     : Factor w/ 6 levels "0","500s","1000s",..: 3 2 1 1 1 6 5 1 2 3
...
##  $ amt_change_q4_q1: Factor w/ 3 levels "Low","Medium",..: 2 2 3 2 2 2 2 1 2 3 ...
##  $ total_trans_amt : Factor w/ 3 levels "Low","Medium",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ total_trans_cnt : Factor w/ 8 levels "Under 20","20s",..: 3 2 1 1 2 2 2 3 2 1 ...
##  $ cnt_change_q4_q1: Factor w/ 3 levels "Low","Medium",..: 2 3 2 2 3 1 1 1 2 3 ...
##  $ utilization     : Factor w/ 3 levels "Low","Medium",..: 1 1 1 3 1 1 1 1 1 1 ...
```

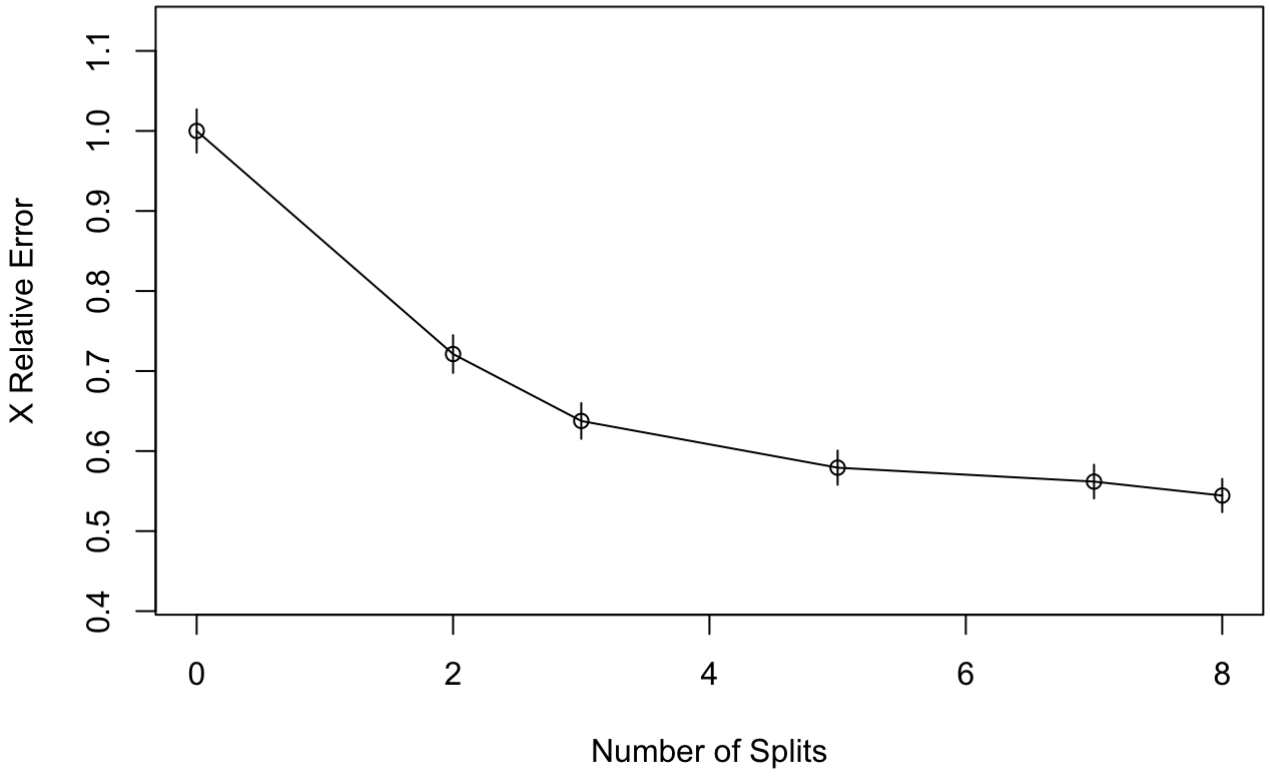
#Unpruned Tree testtime = rpart(attrition_flag~., data = train, method = 'class', control = rpart.control(cp=0,minsplit=3,maxdepth=7))

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

rsq.rpart(tree_model1)
```

```
##
## Classification tree:
## rpart(formula = attrition_flag ~ ., data = train, method = "class",
##     model = T)
##
## Variables actually used in tree construction:
## [1] months_inactive products_num    revolving_bal   total_trans_amt
## [5] total_trans_cnt
##
## Root node error: 1148/7089 = 0.16194
##
## n= 7089
##
##         CP nsplit rel error  xerror     xstd
## 1 0.139373      0   1.00000 1.00000 0.027019
## 2 0.083624      2   0.72125 0.72125 0.023556
## 3 0.029181      3   0.63763 0.63763 0.022318
## 4 0.018293      5   0.57927 0.57927 0.021384
## 5 0.014808      7   0.54268 0.56185 0.021092
## 6 0.010000      8   0.52787 0.54443 0.020795
```

```
## Warning in rsq.rpart(tree_model1): may not be applicable for this method
```


Apparent
X Relative
R-square
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
Number of Splits


X Relative Error
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
0
2
4
6
8
Number of Splits

```
summary(tree_model1)
```

```
## Call:
## rpart(formula = attrition_flag ~ ., data = train, method = "class",
##     model = T)
##   n= 7089
##
##           CP nsplit rel error    xerror       xstd
## 1 0.13937282      0 1.0000000 1.0000000 0.02701882
## 2 0.08362369      2 0.7212544 0.7212544 0.02355606
## 3 0.02918118      3 0.6376307 0.6376307 0.02231759
## 4 0.01829268      5 0.5792683 0.5792683 0.02138352
## 5 0.01480836      7 0.5426829 0.5618467 0.02109227
## 6 0.01000000      8 0.5278746 0.5444251 0.02079487
##
## Variable importance
##  total_trans_cnt    revolving_bal     products_num  total_trans_amt
##               37               29               17                7
##         contacts  months_inactive amt_change_q4_q1 cnt_change_q4_q1
##                3                2                2                1
##              age
##                1
##
## Node number 1: 7089 observations,    complexity param=0.1393728
##   predicted class=Existing Customer  expected loss=0.161941  P(node) =1
##     class counts:  1148  5941
##    probabilities: 0.162 0.838
##   left son=2 (2788 obs) right son=3 (4301 obs)
##   Primary splits:
##       total_trans_cnt splits as  LLLRRRR-, improve=322.80430, (0 missing)
##       revolving_bal   splits as  LRRRRR,   improve=228.42260, (0 missing)
##       months_inactive splits as  LRLLLLR,  improve= 56.87742, (0 missing)
##       contacts        splits as  RRRRRRL,  improve= 55.08579, (0 missing)
##       utilization     splits as  LRR,      improve= 45.72563, (0 missing)
##   Surrogate splits:
##       amt_change_q4_q1 splits as  RLL,     agree=0.633, adj=0.067, (0 split)
##       cnt_change_q4_q1 splits as  RLL,     agree=0.624, adj=0.044, (0 split)
##       contacts         splits as  RRRRLLL, agree=0.623, adj=0.041, (0 split)
##       age              splits as  RRRRLL,  agree=0.610, adj=0.009, (0 split)
##       months_active    splits as  RRRRL--, agree=0.607, adj=0.002, (0 split)
##
## Node number 2: 2788 observations,    complexity param=0.1393728
##   predicted class=Existing Customer  expected loss=0.3493544  P(node) =0.3932854
##     class counts:   974  1814
##    probabilities: 0.349 0.651
##   left son=4 (836 obs) right son=5 (1952 obs)
##   Primary splits:
##       revolving_bal    splits as  LRRRRR,  improve=279.37380, (0 missing)
##       products_num     splits as  LLRRRR,  improve=237.04880, (0 missing)
##       months_inactive  splits as  LRLLLLL, improve= 79.49083, (0 missing)
##       amt_change_q4_q1 splits as  LRR,     improve= 62.20589, (0 missing)
##       gender           splits as  LR,      improve= 49.78275, (0 missing)
##   Surrogate splits:
##       products_num    splits as  LLRRRR,  agree=0.706, adj=0.020, (0 split)
```

```
##       contacts        splits as  RRRRRRL, agree=0.704, adj=0.012, (0 split)
##       months_inactive splits as  LRRRRRR, agree=0.702, adj=0.006, (0 split)
##       total_trans_amt splits as  RL-,     agree=0.701, adj=0.002, (0 split)
##       card            splits as  RRLR,    agree=0.701, adj=0.001, (0 split)
##
## Node number 3: 4301 observations,    complexity param=0.02918118
##   predicted class=Existing Customer  expected loss=0.04045571  P(node) =0.6067146
##     class counts:   174  4127
##    probabilities: 0.040 0.960
##   left son=6 (494 obs) right son=7 (3807 obs)
##   Primary splits:
##       total_trans_amt splits as  RLR,      improve=68.095000, (0 missing)
##       contacts        splits as  RRRRRRL,  improve=16.607810, (0 missing)
##       revolving_bal   splits as  LRRRRR,   improve=12.175930, (0 missing)
##       utilization     splits as  LRR,      improve= 5.495250, (0 missing)
##       total_trans_cnt splits as  ---LRRR-, improve= 4.892582, (0 missing)
##   Surrogate splits:
##       contacts splits as  RRRRRRL, agree=0.887, adj=0.014, (0 split)
##
## Node number 4: 836 observations,    complexity param=0.01829268
##   predicted class=Attrited Customer  expected loss=0.3086124  P(node) =0.1179292
##     class counts:   578   258
##    probabilities: 0.691 0.309
##   left son=8 (195 obs) right son=9 (641 obs)
##   Primary splits:
##       products_num     splits as  LLRRRR,  improve=39.26562, (0 missing)
##       months_inactive  splits as  LRLLLLL, improve=31.23272, (0 missing)
##       amt_change_q4_q1 splits as  LRR,     improve=18.27158, (0 missing)
##       contacts         splits as  RLLLLLL, improve=15.50774, (0 missing)
##       gender           splits as  LR,      improve=12.35208, (0 missing)
##   Surrogate splits:
##       total_trans_amt splits as  RL-,     agree=0.775, adj=0.036, (0 split)
##       card            splits as  RLLR,    agree=0.769, adj=0.010, (0 split)
##       months_inactive splits as  RRRRRLR, agree=0.768, adj=0.005, (0 split)
##
## Node number 5: 1952 observations,    complexity param=0.08362369
##   predicted class=Existing Customer  expected loss=0.2028689  P(node) =0.2753562
##     class counts:   396  1556
##    probabilities: 0.203 0.797
##   left son=10 (178 obs) right son=11 (1774 obs)
##   Primary splits:
##       products_num     splits as  LLRRRR,  improve=125.84230, (0 missing)
##       revolving_bal    splits as  -LRRRL,  improve= 33.72273, (0 missing)
##       months_inactive  splits as  LRLLLLL, improve= 25.24145, (0 missing)
##       amt_change_q4_q1 splits as  LRR,     improve= 19.31011, (0 missing)
##       gender           splits as  LR,      improve= 17.79912, (0 missing)
##   Surrogate splits:
##       total_trans_amt splits as  RL-, agree=0.911, adj=0.028, (0 split)
##
## Node number 6: 494 observations,    complexity param=0.02918118
##   predicted class=Existing Customer  expected loss=0.2874494  P(node) =0.06968543
##     class counts:   142   352
```

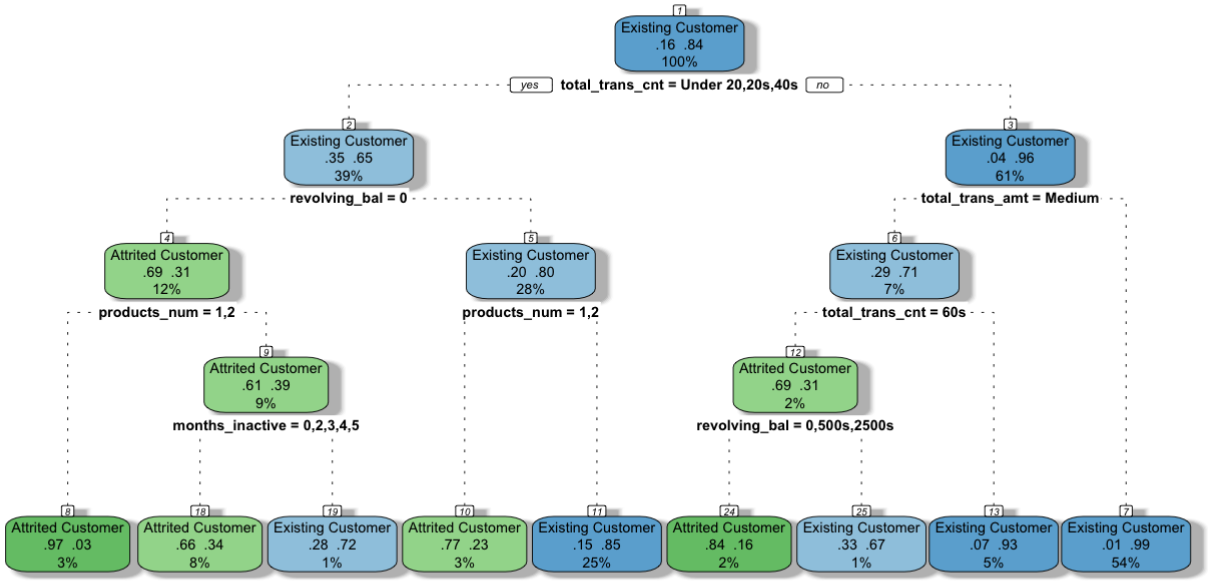
```
##    probabilities: 0.287 0.713
##   left son=12 (173 obs) right son=13 (321 obs)
##   Primary splits:
##       total_trans_cnt splits as  ---LRRR-, improve=87.85394, (0 missing)
##       products_num    splits as  RRRLLL,   improve=65.62303, (0 missing)
##       revolving_bal   splits as  LLRRRR,   improve=60.64026, (0 missing)
##       contacts        splits as  RRRRLLL,  improve=40.61383, (0 missing)
##       age             splits as  LLRLR-,   improve=12.73284, (0 missing)
##   Surrogate splits:
##       revolving_bal splits as  LLRRRR,  agree=0.737, adj=0.249, (0 split)
##       products_num  splits as  RRRLLL,  agree=0.735, adj=0.243, (0 split)
##       contacts      splits as  RRRRLLL, agree=0.709, adj=0.168, (0 split)
##       age           splits as  LRRRR-,  agree=0.668, adj=0.052, (0 split)
##       months_active splits as  LRRRR--, agree=0.668, adj=0.052, (0 split)
##
## Node number 7: 3807 observations
##   predicted class=Existing Customer  expected loss=0.008405569  P(node) =0.5370292
##     class counts:    32  3775
##    probabilities: 0.008 0.992
##
## Node number 8: 195 observations
##   predicted class=Attrited Customer  expected loss=0.03076923  P(node) =0.02750741
##     class counts:   189     6
##    probabilities: 0.969 0.031
##
## Node number 9: 641 observations,    complexity param=0.01829268
##   predicted class=Attrited Customer  expected loss=0.3931357  P(node) =0.09042178
##     class counts:   389   252
##    probabilities: 0.607 0.393
##   left son=18 (547 obs) right son=19 (94 obs)
##   Primary splits:
##       months_inactive  splits as  LRLLLLR, improve=24.03051, (0 missing)
##       gender           splits as  LR,      improve=16.48928, (0 missing)
##       amt_change_q4_q1 splits as  LRR,     improve=13.66781, (0 missing)
##       contacts         splits as  RRRLLLL, improve=11.73991, (0 missing)
##       products_num     splits as  --LRLR,  improve=11.26652, (0 missing)
##   Surrogate splits:
##       amt_change_q4_q1 splits as  LLR, agree=0.856, adj=0.021, (0 split)
##       cnt_change_q4_q1 splits as  LLR, agree=0.856, adj=0.021, (0 split)
##
## Node number 10: 178 observations
##   predicted class=Attrited Customer  expected loss=0.2303371  P(node) =0.02510932
##     class counts:   137    41
##    probabilities: 0.770 0.230
##
## Node number 11: 1774 observations
##   predicted class=Existing Customer  expected loss=0.1459977  P(node) =0.2502469
##     class counts:   259  1515
##    probabilities: 0.146 0.854
##
## Node number 12: 173 observations,    complexity param=0.01480836
##   predicted class=Attrited Customer  expected loss=0.3063584  P(node) =0.02440401
```

```
##     class counts:   120    53
##    probabilities: 0.694 0.306
##   left son=24 (124 obs) right son=25 (49 obs)
##   Primary splits:
##       revolving_bal splits as  LLRRRL,  improve=18.426600, (0 missing)
##       products_num  splits as  RRRLLL,  improve=13.955760, (0 missing)
##       contacts      splits as  RRLLLLL, improve=13.150150, (0 missing)
##       utilization   splits as  LRR,     improve= 6.344562, (0 missing)
##       age           splits as  LLRLR-,  improve= 6.141065, (0 missing)
##   Surrogate splits:
##       contacts        splits as  RRLLLLL, agree=0.740, adj=0.082, (0 split)
##       utilization     splits as  LLR,     agree=0.734, adj=0.061, (0 split)
##       months_inactive splits as  LLLLLR-, agree=0.723, adj=0.020, (0 split)
##
## Node number 13: 321 observations
##   predicted class=Existing Customer  expected loss=0.06853583  P(node) =0.04528142
##     class counts:    22   299
##    probabilities: 0.069 0.931
##
## Node number 18: 547 observations
##   predicted class=Attrited Customer  expected loss=0.3363803  P(node) =0.0771618
##     class counts:   363   184
##    probabilities: 0.664 0.336
##
## Node number 19: 94 observations
##   predicted class=Existing Customer  expected loss=0.2765957  P(node) =0.01325998
##     class counts:    26    68
##    probabilities: 0.277 0.723
##
## Node number 24: 124 observations
##   predicted class=Attrited Customer  expected loss=0.1612903  P(node) =0.01749189
##     class counts:   104    20
##    probabilities: 0.839 0.161
##
## Node number 25: 49 observations
##   predicted class=Existing Customer  expected loss=0.3265306  P(node) =0.006912117
##     class counts:    16    33
##    probabilities: 0.327 0.673
```

Variable importance: cnt_change_q4_q1 revolving_bal total_trans_cnt total_trans_amt

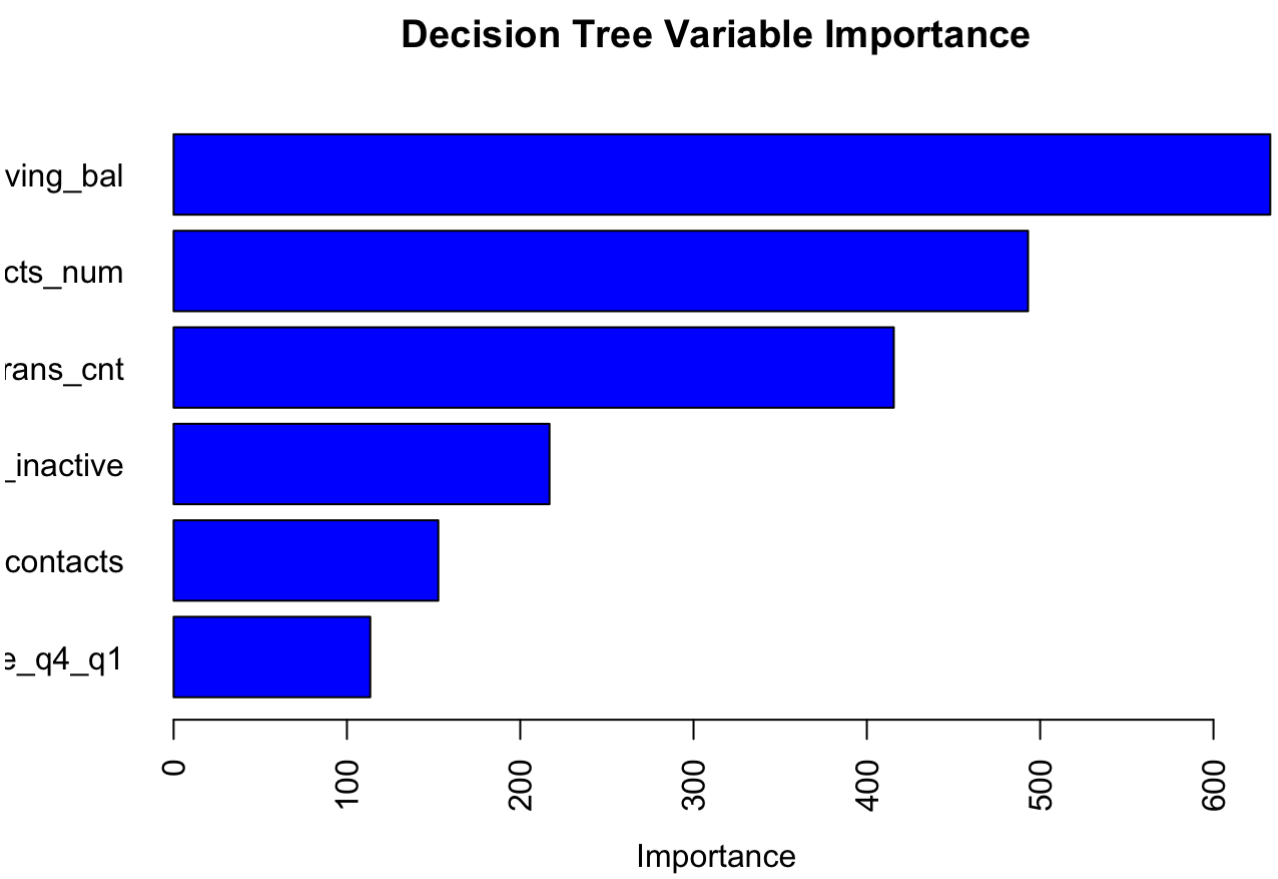
These are in line with Apriori.

```
fancyRpartPlot(tree_model1)
```



Rattle 2023-Mar-23 23:02:09 fruitisushi

```
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 using tree_model1

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

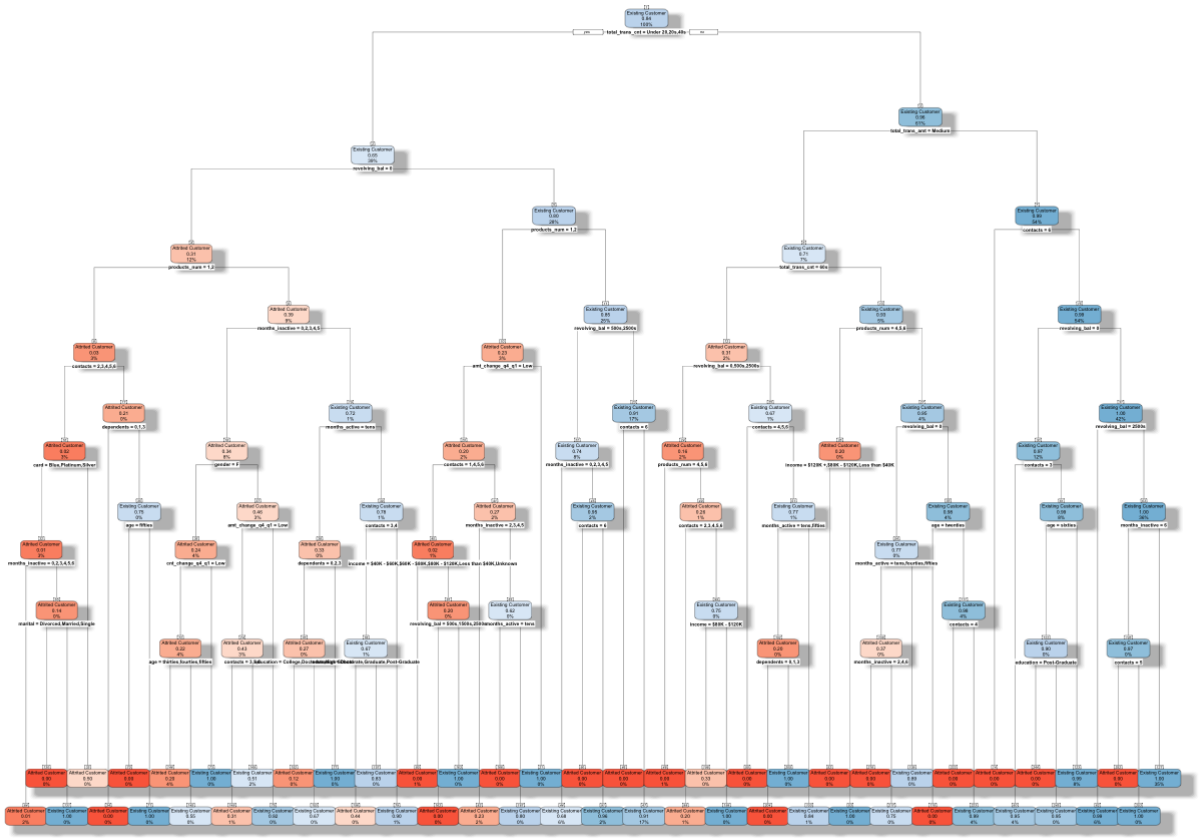
```
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               330                96
##   Existing Customer               149              2463
```

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

```
## Accuracy
##    91.94
```

```
testtime = rpart(attrition_flag~., data = train, method = 'class', control = rpart.contr
ol(cp=0,minsplit=3,maxdepth=7))
rpart.plot(testtime, box.palette="RdBu", shadow.col="gray", nn=TRUE)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```
#identify best cp value to use
#best <- testtime$cptable[which.min(testtime$cptable[,"xerror"]),"CP"]
#produce a pruned tree based on the best cp value
#pruned_tree <- prune(testtime, cp=best)
#plot the pruned tree
#prp(pruned_tree)

testpred<- predict(object=testtime,test[-1],type="class")
testpred.cm <- confusionMatrix(testpred, test$attrition_flag)
testpred.cm$table
```

```
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               304                73
##   Existing Customer               175              2486
```

```
test.acc <- round(testpred.cm$overall[1]*100,2)
test.acc
```

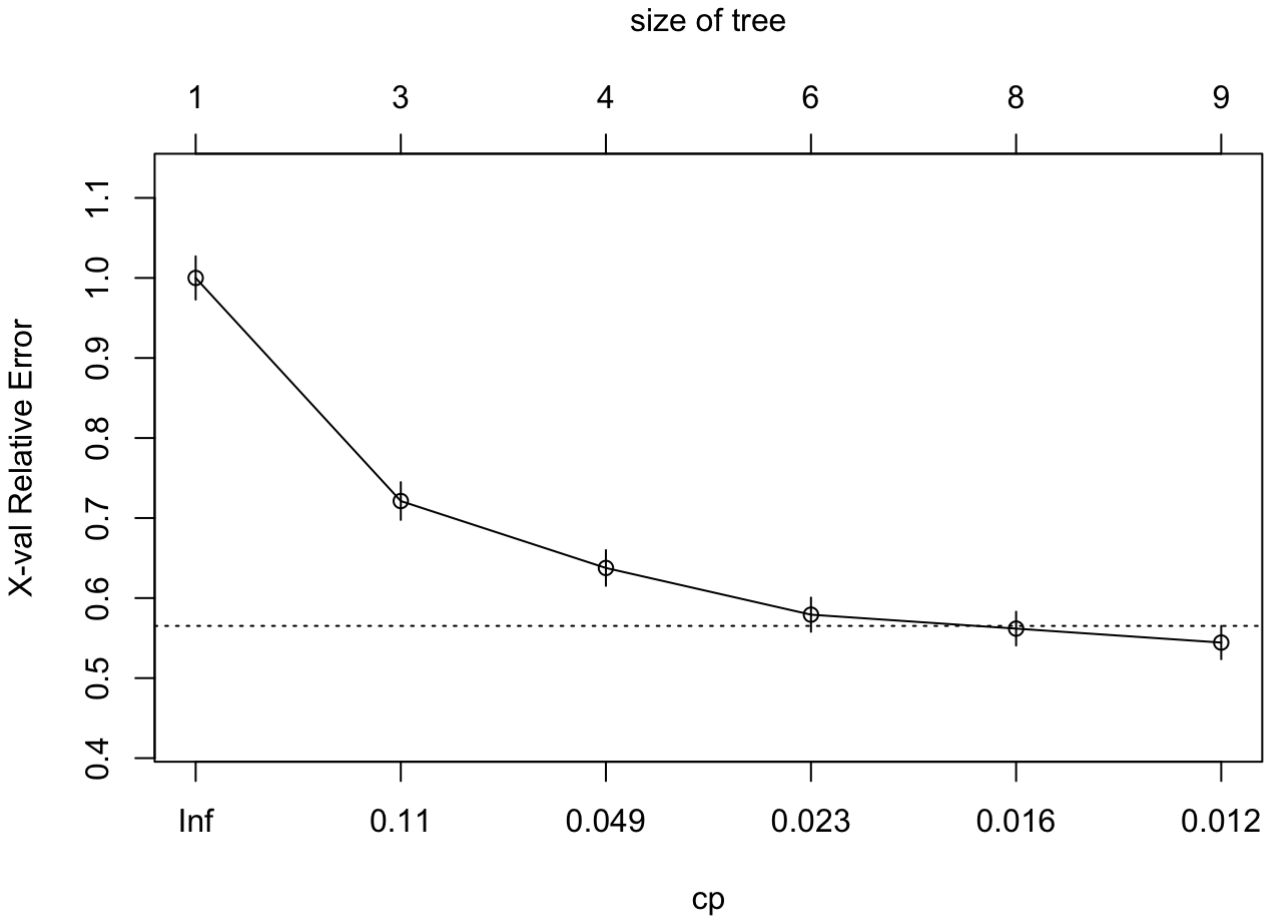
```
## Accuracy
##    91.84
```

#Pruned Tree and accuracy

```
printcp(tree_model1)
```

```
##
## Classification tree:
## rpart(formula = attrition_flag ~ ., data = train, method = "class",
##     model = T)
##
## Variables actually used in tree construction:
## [1] months_inactive products_num    revolving_bal   total_trans_amt
## [5] total_trans_cnt
##
## Root node error: 1148/7089 = 0.16194
##
## n= 7089
##
##         CP nsplit rel error  xerror     xstd
## 1 0.139373      0   1.00000 1.00000 0.027019
## 2 0.083624      2   0.72125 0.72125 0.023556
## 3 0.029181      3   0.63763 0.63763 0.022318
## 4 0.018293      5   0.57927 0.57927 0.021384
## 5 0.014808      7   0.54268 0.56185 0.021092
## 6 0.010000      8   0.52787 0.54443 0.020795
```

```
plotcp(tree_model1)
```



#choose a cp(complexity parameter) with lowest x-error value

```
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
```

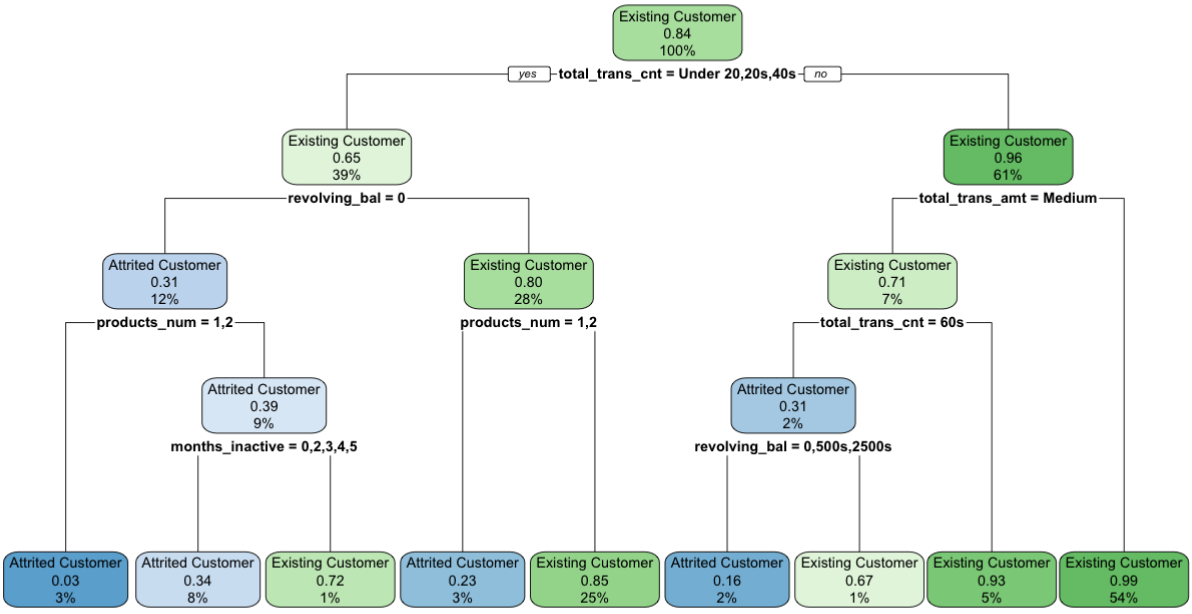
```
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               330                96
##   Existing Customer               149              2463
```

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

```
## Accuracy
##    91.94
```

```
#summary(tree_model2)
```

```
rpart.plot(tree_model2)
```



Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

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 in order 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.

In the current scenario, we have a high accuracy rate, but it is still useful to check for any overfitting.

Cross validation

```
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)
}
```

#Error

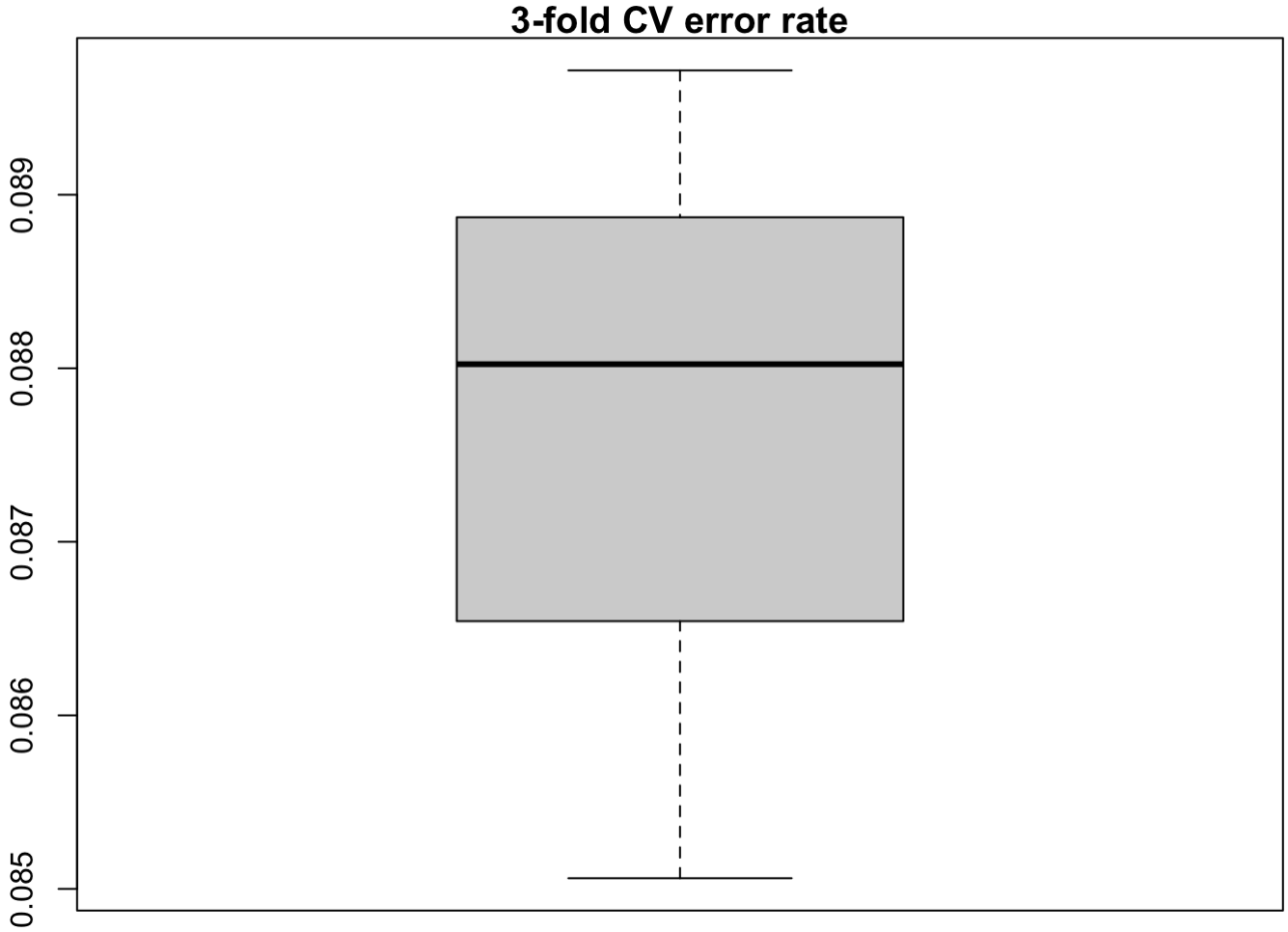
```
all.err_tree
```

```
##           [,1]
## err 0.08506136
## err 0.08802370
## err 0.08971646
```

Error percentage is approx. 8%

Plotting the error

```
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")
```




```
fit = rpart(attrition_flag~., data = train, method = 'class',
            control = rpart.control(cp=0,minsplit=3,maxdepth=7))

#rsq.rpart(fit)
#summary(fit)
#fancyRpartPlot(fit)
##rpart.plot(fit, box.palette="RdBu", shadow.col="gray", nn=TRUE)

fitpred<- predict(object=fit,test[-1],type="class")
fitpred.cm <- confusionMatrix(fitpred, test$attrition_flag)
fitpred.cm$table
```

```
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               304                73
##   Existing Customer               175              2486
```

```
fit.acc <- round(fitpred.cm$overall[1]*100,2)
fit.acc
```

```
## Accuracy
##    91.84
```

```
set.seed(10)
fit2 <- prune(fit, cp = 0.001)
pred2<- predict(object=fit2,test[-1],type="class")
fit2.cm <- confusionMatrix(pred2, test$attrition_flag)
fit2.cm$table
```

```
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               303                61
##   Existing Customer               176              2498
```

```
fit2.acc <- round(fit2.cm$overall[1]*100,2)
fit2.acc
```

```
## Accuracy
##     92.2
```

Naive Bayes

This will take the train dataset and run through naiveBayes algorithm.

```
trainnb = naiveBayes(as.factor(attrition_flag) ~., data = train)
```

```
set.seed(999)
trainnb_pred = predict(trainnb, train, type='class')
confusionMatrix(trainnb_pred, as.factor(train$attrition_flag))
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               695               192
##   Existing Customer               453              5749
##
##                Accuracy : 0.909
##                  95% CI : (0.9021, 0.9156)
##     No Information Rate : 0.8381
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6309
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.60540
##             Specificity : 0.96768
##          Pos Pred Value : 0.78354
##          Neg Pred Value : 0.92696
##              Prevalence : 0.16194
##          Detection Rate : 0.09804
##    Detection Prevalence : 0.12512
##       Balanced Accuracy : 0.78654
##
##        'Positive' Class : Attrited Customer
##
```

Miscalculation rate:

```
mean(trainnb_pred != train$attrition_flag)
```

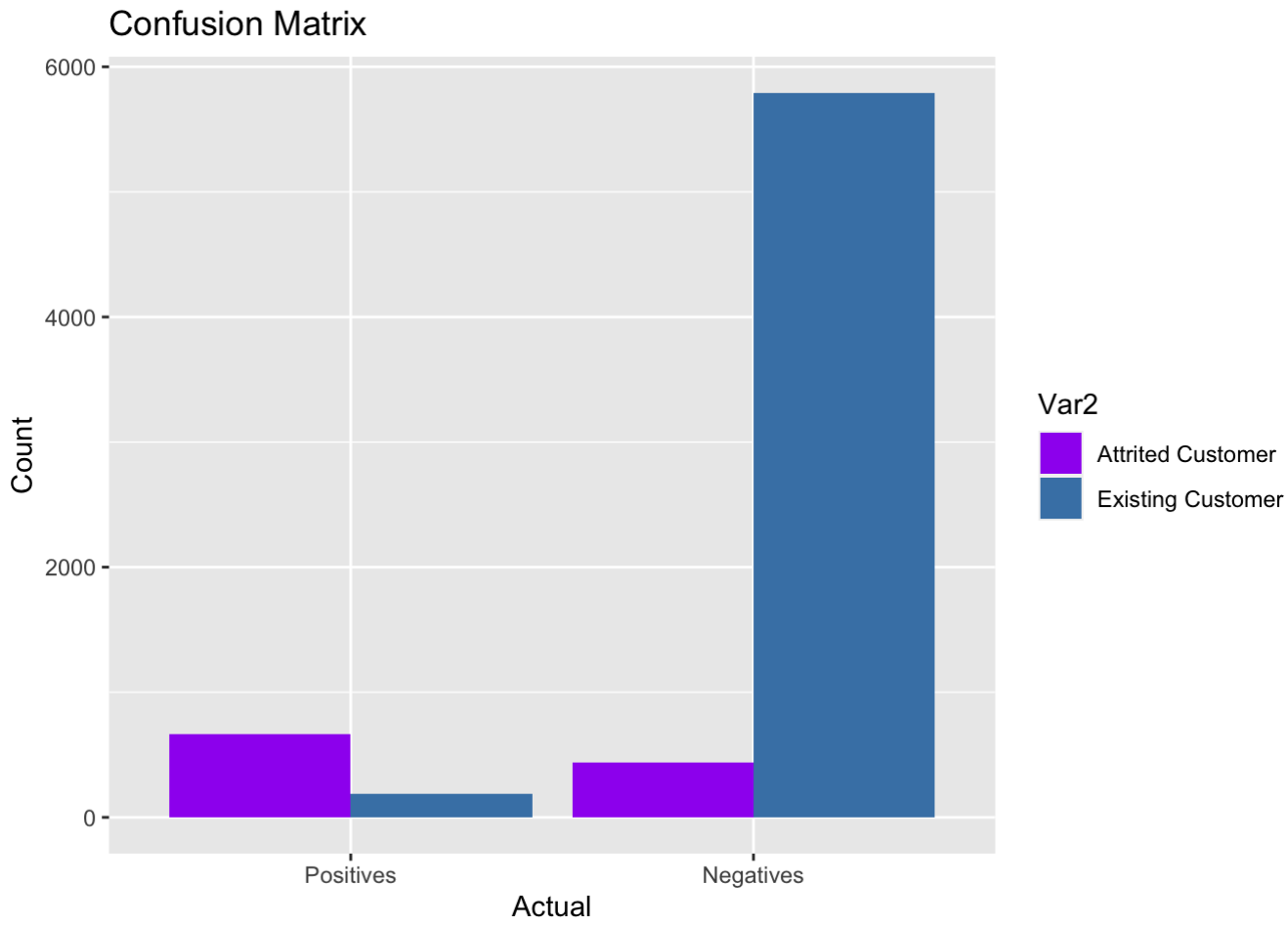
```
## [1] 0.09098603
```

```
set.seed(999)
trainnb_pred = predict(trainnb, train, type='class')
cmnb=confusionMatrix(trainnb_pred, as.factor(train$attrition_flag))
cmnb
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               695               192
##   Existing Customer               453              5749
##
##                Accuracy : 0.909
##                  95% CI : (0.9021, 0.9156)
##     No Information Rate : 0.8381
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6309
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.60540
##             Specificity : 0.96768
##          Pos Pred Value : 0.78354
##          Neg Pred Value : 0.92696
##              Prevalence : 0.16194
##          Detection Rate : 0.09804
##    Detection Prevalence : 0.12512
##       Balanced Accuracy : 0.78654
##
##        'Positive' Class : Attrited Customer
##
```

```
# Create a matrix with the confusion matrix counts
conf_mat <- matrix(c(668, 186, 442, 5792), nrow = 2, byrow = TRUE,
                   dimnames = list(c("Positives", "Negatives"),
                                   c("Attrited Customer", "Existing Customer")))

# Create a bar plot for the confusion matrix
ggplot(melt(conf_mat), aes(x = Var1, y = value, fill = Var2)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Actual", y = "Count") +
  scale_fill_manual(values = c("Attrited Customer" = "purple", "Existing Customer" = "st
eelblue")) +
  ggtitle("Confusion Matrix")
```

```
## Warning in melt(conf_mat): The melt generic in data.table has been passed a
## matrix and will attempt to redirect to the relevant reshape2 method; please note
## that reshape2 is deprecated, and this redirection is now deprecated as well.
## To continue using melt methods from reshape2 while both libraries are attached,
## e.g. melt.list, you can prepend the namespace like reshape2::melt(conf_mat). In
## the next version, this warning will become an error.
```



Miscalculation rate:

```
mean(trainnb_pred != train$attrition_flag)
```

```
## [1] 0.09098603
```

This first Bayes prediction test has an predicting ability accuracy rate of 91%. This is high but not quite high enough for an accurate model.

Now it is time to test the prediction model and apply it to the test set:

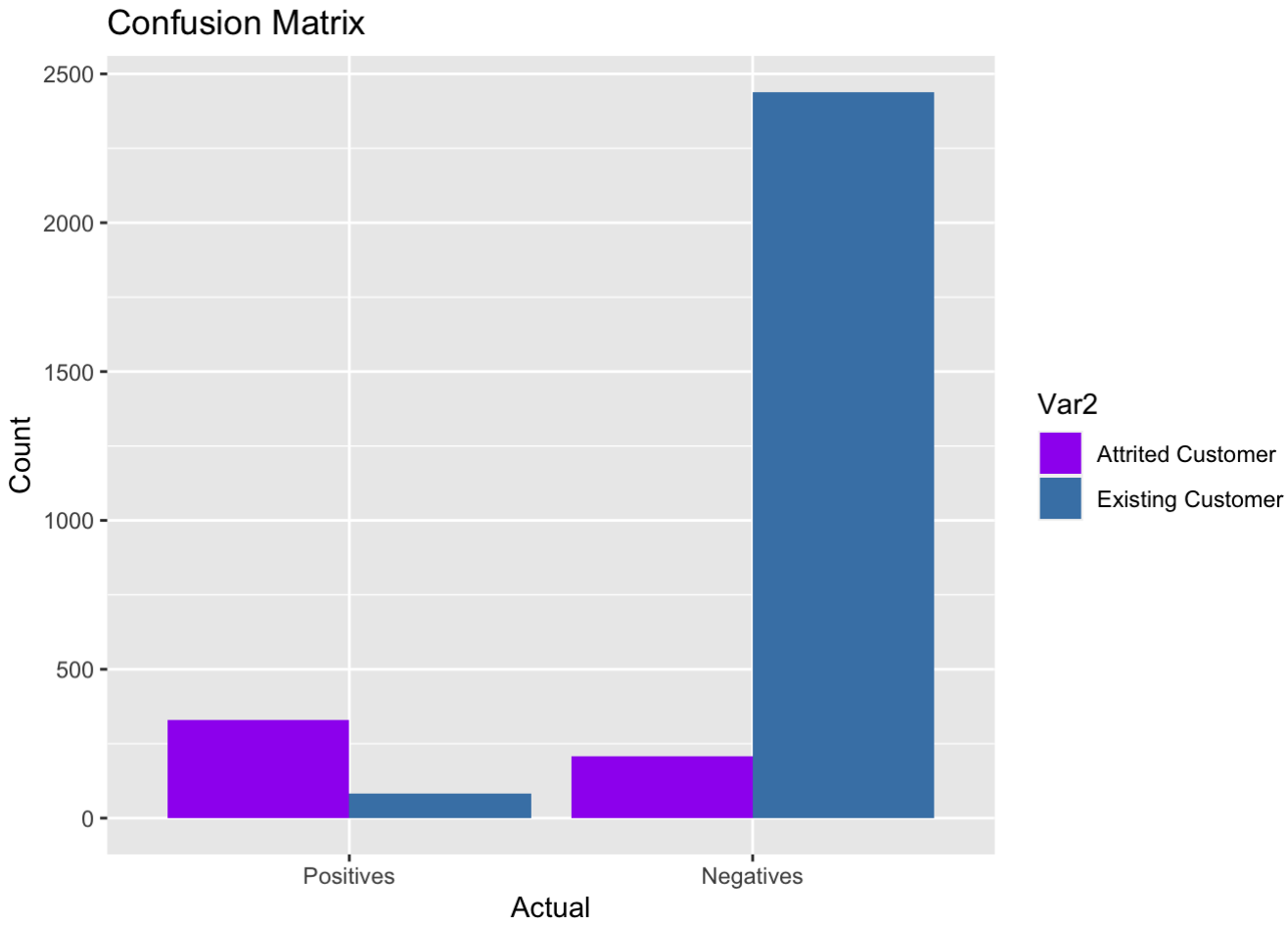
```
set.seed(999)
test$predicted = predict(trainnb,test)
test$actual = test$attrition_flag
test_df = data.frame(test$actual,test$predicted)
confusionMatrix(factor(test$predicted),
                factor(test$actual))
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               293                82
##   Existing Customer               186              2477
##
##                Accuracy : 0.9118
##                  95% CI : (0.9011, 0.9216)
##     No Information Rate : 0.8423
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6357
##
##  Mcnemar's Test P-Value : 3.14e-10
##
##             Sensitivity : 0.61169
##             Specificity : 0.96796
##          Pos Pred Value : 0.78133
##          Neg Pred Value : 0.93015
##              Prevalence : 0.15767
##          Detection Rate : 0.09645
##    Detection Prevalence : 0.12344
##       Balanced Accuracy : 0.78982
##
##        'Positive' Class : Attrited Customer
##
```

The accuracy rate is 91.3% after applying the training data to the test set. The train and test set are 70/30% split. 90% is high however, it may not be high enough to accurately predict whether a customer will leave or stay.

```
# Create a matrix with the confusion matrix counts
conf_mat2 <- matrix(c(331, 83, 206, 2439), nrow = 2, byrow = TRUE,
                    dimnames = list(c("Positives", "Negatives"),
                                    c("Attrited Customer", "Existing Customer")))


# Create a bar plot for the confusion matrix
ggplot(melt(conf_mat2), aes(x = Var1, y = value, fill = Var2)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Actual", y = "Count") +
  scale_fill_manual(values = c("Attrited Customer" = "purple", "Existing Customer" = "st
eelblue")) +
  ggtitle("Confusion Matrix")
```

```
## Warning in melt(conf_mat2): The melt generic in data.table has been passed a
## matrix and will attempt to redirect to the relevant reshape2 method; please note
## that reshape2 is deprecated, and this redirection is now deprecated as well.
## To continue using melt methods from reshape2 while both libraries are attached,
## e.g. melt.list, you can prepend the namespace like reshape2::melt(conf_mat2). In
## the next version, this warning will become an error.
```



SVM

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are: Effective in high dimensional spaces.

SVM model with less supporting vectors implies faster processing.

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

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

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

With Kernel -Linear

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

```
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
```

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

```
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
##
##        'Positive' Class : Attrited Customer
##
```

SVM Linear has support Vectors 1423 and Accuracy of 92.03

```
table(predicted = svmlinear_pred, true = testsvm$attrition_flag)
```

```
##                    true
## predicted           Attrited Customer Existing Customer
##   Attrited Customer               346                89
##   Existing Customer               153              2450
```

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

Using Polynomial

```
set.seed(423)
svm_poly = svm(formula =attrition_flag ~ .,
                 data = trainsvm,
                 type = 'C-classification',
                 kernel = 'polynomial')
```

```
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
```

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

```
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
##       Balanced Accuracy :     NA
##
##        'Positive' Class : Attrited Customer
##
```

SVM Polynomial has 2468 supporting vectors and accuracy of 83.57% # kernel="radial"

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

```
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
```

```
svmrad_pred = predict(svm_rad, newdata = testsvm)
```

```
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
##
```

SVM Radial has 2120 Supporting Vectors with 90.22 accuracy

Using sigmoid

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

```
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
```

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

```
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
##
```

finding an optimal cost factor and applying it to the lowest accurate model

```
#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)
```

```
#tune1.best.performance
```

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

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

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

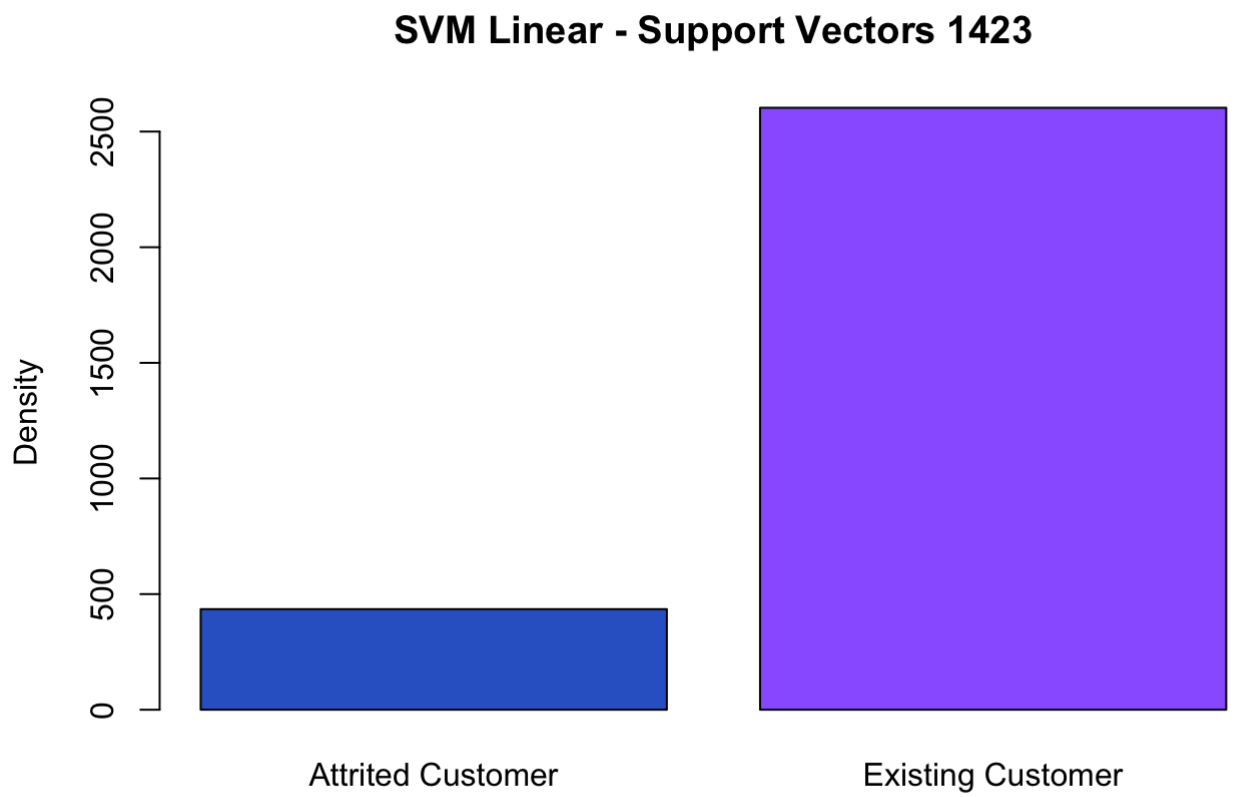
```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer               313               186
##   Existing Customer                81              2458
##
##                Accuracy : 0.9121
##                  95% CI : (0.9015, 0.9219)
##     No Information Rate : 0.8703
##     P-Value [Acc > NIR] : 3.258e-13
##
##                   Kappa : 0.6503
##
##  Mcnemar's Test P-Value : 1.957e-10
##
##             Sensitivity : 0.7944
##             Specificity : 0.9297
##          Pos Pred Value : 0.6273
##          Neg Pred Value : 0.9681
##              Prevalence : 0.1297
##          Detection Rate : 0.1030
##    Detection Prevalence : 0.1643
##       Balanced Accuracy : 0.8620
##
##        'Positive' Class : Attrited Customer
##
```

```
print(svm_linear1)
```

```
##
## Call:
## svm(formula = attrition_flag ~ ., data = trainsvm, type = "C-classification",
##     cost = 0.08, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.08
##
## Number of Support Vectors:  1792
```

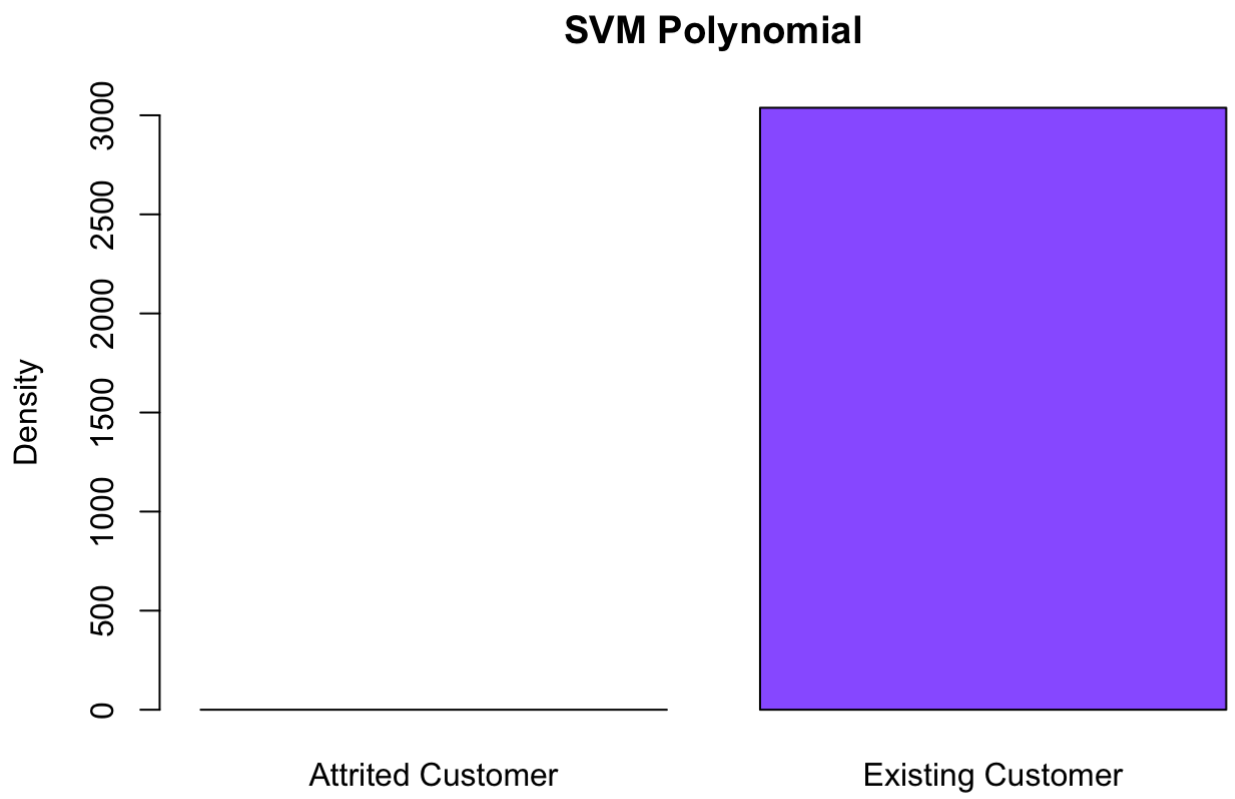
Plotting SVM Predictions

```
plot(svmlinear_pred, ylab='Density',main = 'SVM Linear - Support Vectors 1423',col=c('#3
366CC','#9966FF'))
```

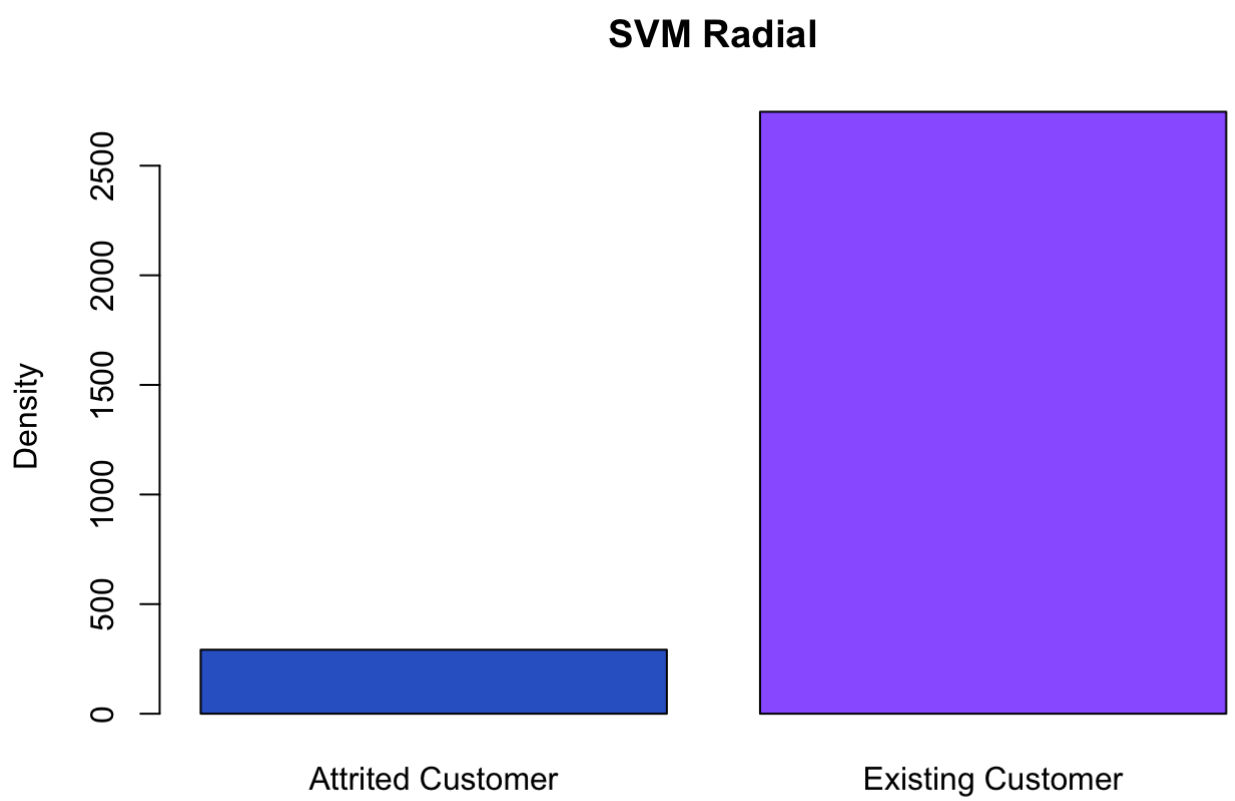



```
svmlinear_df = as.list(svmlinear_pred)
```

```
plot(svmpoly_pred, ylab='Density',main = 'SVM Polynomial',col=c('#3366CC','#9966FF'))
```




```
plot(svmrad_pred, ylab='Density',main = 'SVM Radial',col=c('#3366CC','#9966FF'))
```




```
plot(svmsig_pred, ylab='Density',main = 'SVM Sigmoid')
```

Conclusion SVM: SVM Linear has support Vectors 1423 and Accuracy of 93.03% SVM Polynomial has 2468 supporting vectors with Accuracy of 83.57% SVM Radial has 2120 Supporting Vectors with Accuracy of 90.22% SVM Sigmoid had 2242 supporting vectors with Accuracy of 88.87%

The conclusion is SVM Linear Model has the highest accuracy and fastest classification of testing points.

kNN

KNN which stand 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. It is one of the most simple Machine learning algorithms and it can be easily implemented for a varied set of problems. It is mainly based on feature similarity. KNN checks how similar a data point is to its neighbor and classifies the data point into the class it is most similar to.

```
ccdata_knn<-ccdata
```

```
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 ...
##  $ total_trans_cnt : Factor w/ 8 levels "Under 20","20s",..: 3 2 1 1 2 2 2 2 2 2 ...
##  $ cnt_change_q4_q1: Factor w/ 3 levels "Low","Medium",..: 2 3 2 2 3 1 1 1 1 1 ...
##  $ utilization     : Factor w/ 3 levels "Low","Medium",..: 1 1 1 3 1 1 1 1 1 1 ...
```

Pre-Processing of data: converting factor to numeric values

```
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)
```

```
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 ...
```

```
head(ccdata_knn)
```

```
##   attrition_flag age gender dependents education marital income card
## 1              2   3      2          4         4       2      3    1
## 2              2   3      1          6         3       3      5    1
## 3              2   4      2          4         3       2      4    1
## 4              2   2      1          5         4       4      5    1
## 5              2   2      2          4         6       2      3    1
## 6              2   3      2          3         3       2      2    1
##   months_active products_num months_inactive contacts creditlimit revolving_bal
## 1             3            5               2        4           2             2
## 2             4            6               2        3           1             3
## 3             3            4               2        1           1             1
## 4             3            3               5        2           1             6
## 5             2            5               2        1           1             1
## 6             3            3               2        3           1             3
##   open_to_buy amt_change_q4_q1 total_trans_amt total_trans_cnt cnt_change_q4_q1
## 1           3                2               1               3                2
## 2           2                2               1               2                3
## 3           1                3               1               1                2
## 4           1                2               1               1                2
## 5           1                2               1               2                3
## 6           1                2               1               2                1
##   utilization
## 1           1
## 2           1
## 3           1
## 4           3
## 5           1
## 6           1
```

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

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

```
dim(trainknn)
```

```
## [1] 7088   20
```

```
dim(testknn)
```

```
## [1] 3039   20
```

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

```
## [1] 7088
```

Sq.rt of 7088 is 84.2 . So starting with K=84 and 85

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

```
#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
```

```
## [1] 89.66765
```

```
ACC.85
```

```
## [1] 89.63475
```

```
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
##
```

```
confusionMatrix(table(knn.85 ,testknn_labels))
```

```
## 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
##
```

Both are at accuracy 89.63.

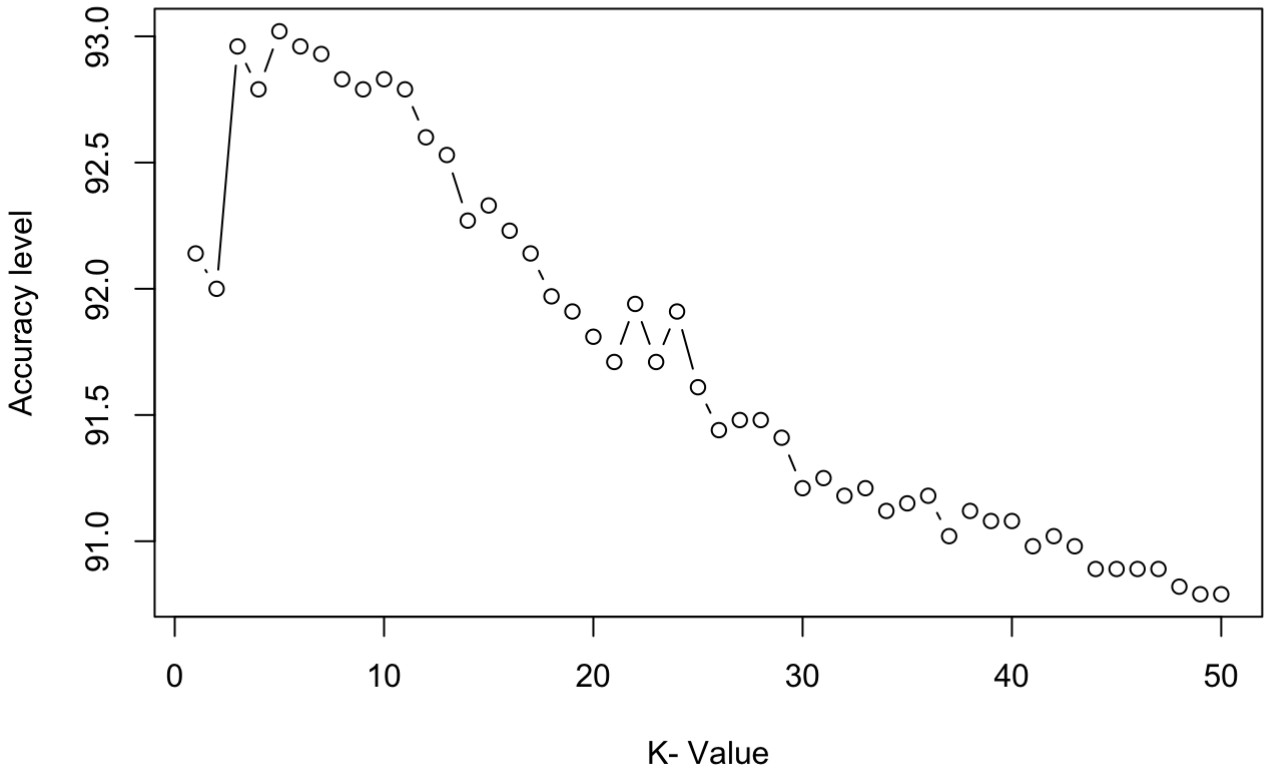
Looping to calculate accuracy when k=1 through 50.

```
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],'
')
 }
```

```
## 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
```

Most accurate model is when k=5 at 93.02%

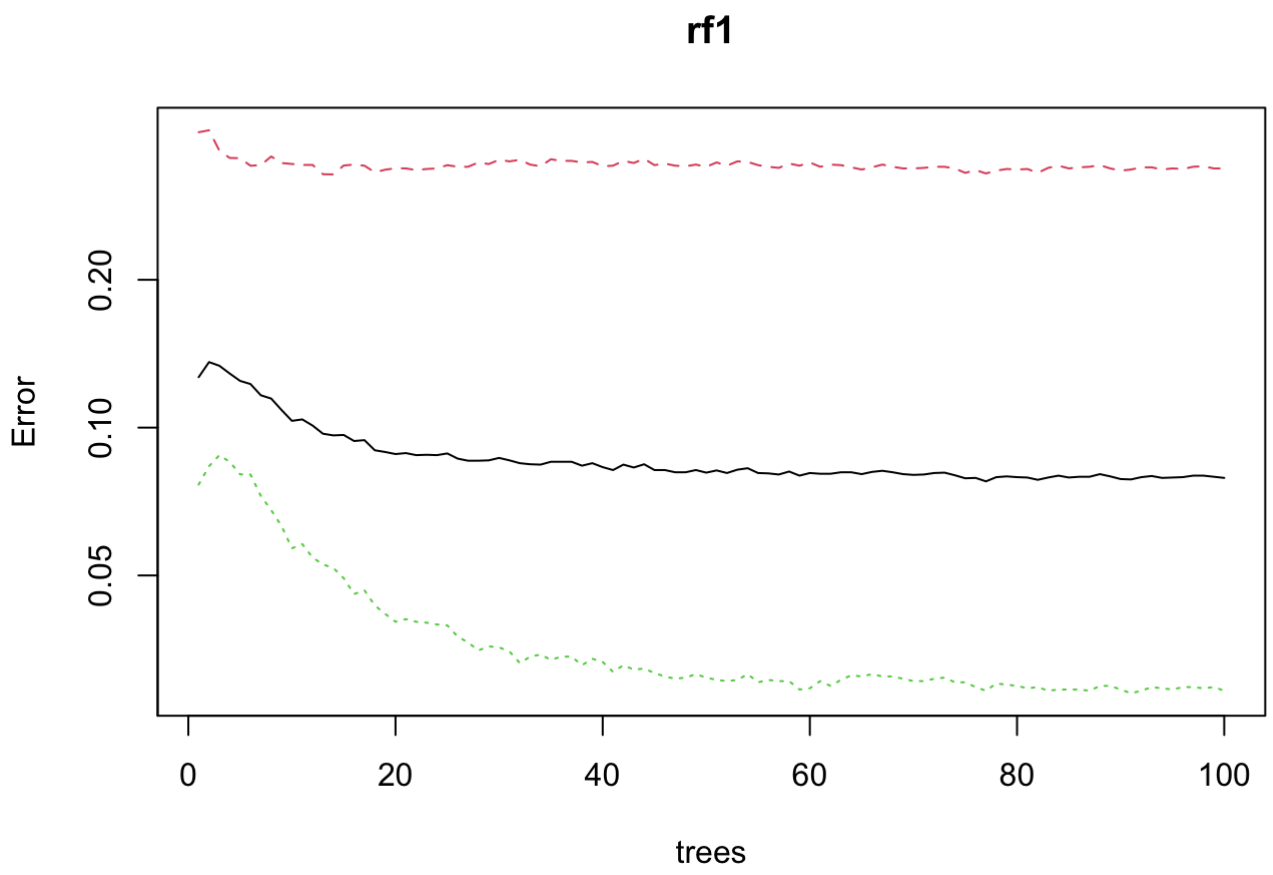
```
#Accuracy plot
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```



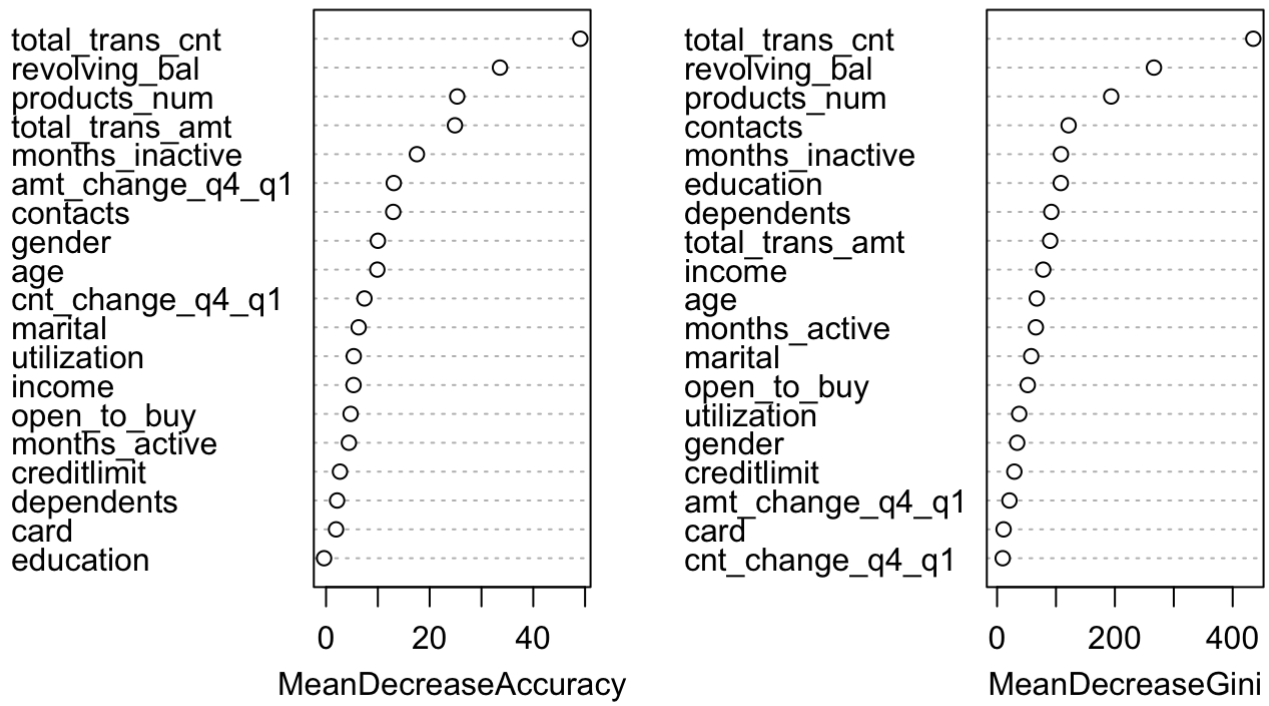
Random Forest

Finding Optimal Number of Trees

```
rf1 <- randomForest(attrition_flag ~ ., data=train,
                          ntree=100, keep.forest=FALSE,
                           importance=TRUE)
plot(rf1, log="y")
```




```
varImpPlot(rf1)
```

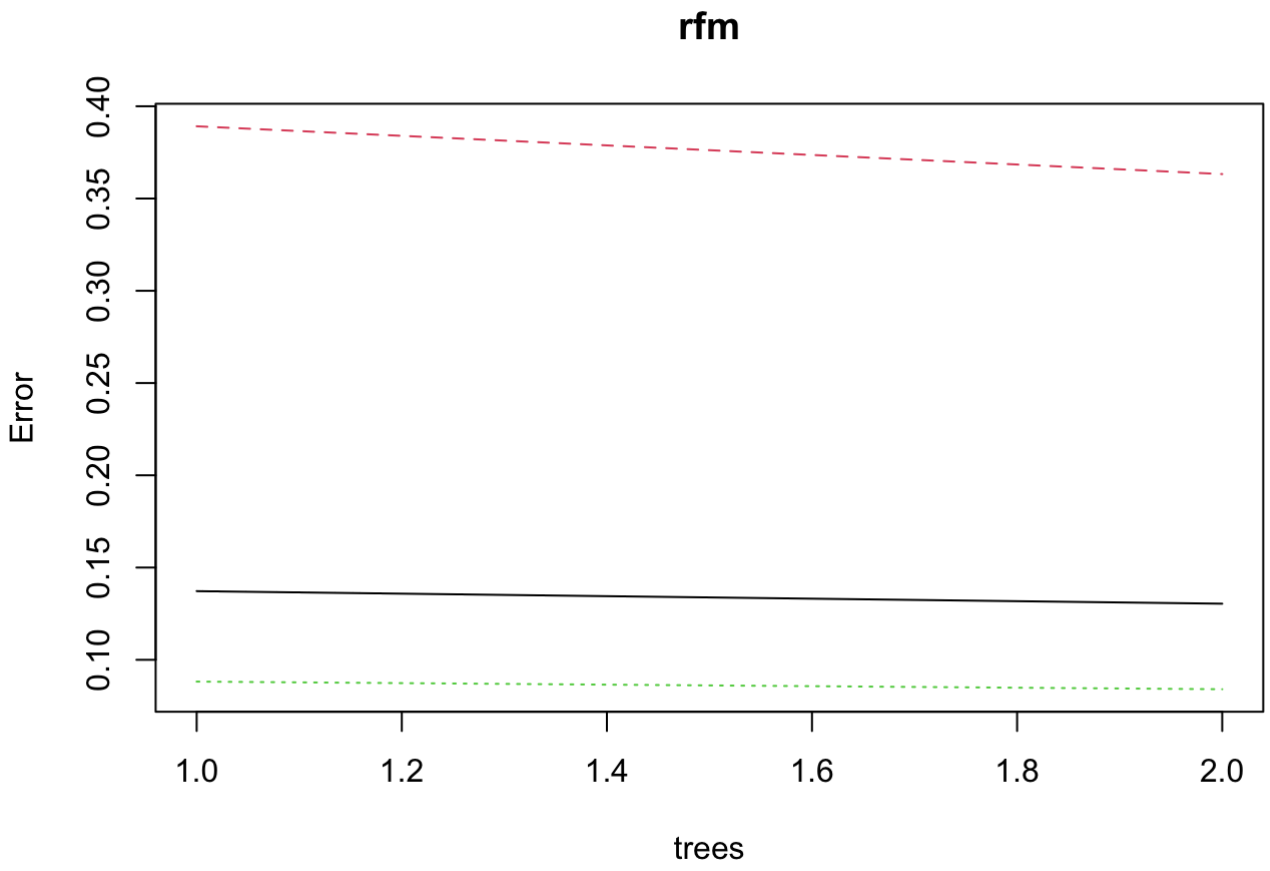


Testing with 2 Trees

```
set.seed(1111)
rfm <- randomForest(attrition_flag~., data=train, ntree=2)
print(rfm)
```

```
##
## Call:
##  randomForest(formula = attrition_flag ~ ., data = train, ntree = 2)
##                Type of random forest: classification
##                      Number of trees: 2
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 13.04%
## Confusion matrix:
##                   Attrited Customer Existing Customer class.error
## Attrited Customer               454               259  0.36325386
## Existing Customer               301              3280  0.08405473
```

```
plot(rfm)
```




```
#apply the prediction and save it as it's own data
predRF <- predict(rfm, train, type=c("class"))
#save the prediction and append it into the original train dataset
train$pred = predRF

##create a new dataframe with the actual and predicted
actual = train$attrition_flag
pred = train$pred

df_rmf = (data.frame(actual,pred))
##calculate the confusion matrix
confusionMatrix(df_rmf$actual,df_rmf$pred)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer              1006               142
##   Existing Customer               246              5695
##
##                Accuracy : 0.9453
##                  95% CI : (0.9397, 0.9504)
##     No Information Rate : 0.8234
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8055
##
##  Mcnemar's Test P-Value : 1.704e-07
##
##             Sensitivity : 0.8035
##             Specificity : 0.9757
##          Pos Pred Value : 0.8763
##          Neg Pred Value : 0.9586
##              Prevalence : 0.1766
##          Detection Rate : 0.1419
##    Detection Prevalence : 0.1619
##       Balanced Accuracy : 0.8896
##
##        'Positive' Class : Attrited Customer
##
```

This is quite high but we can potentially increase the accuracy by optimizing the number of trees used.

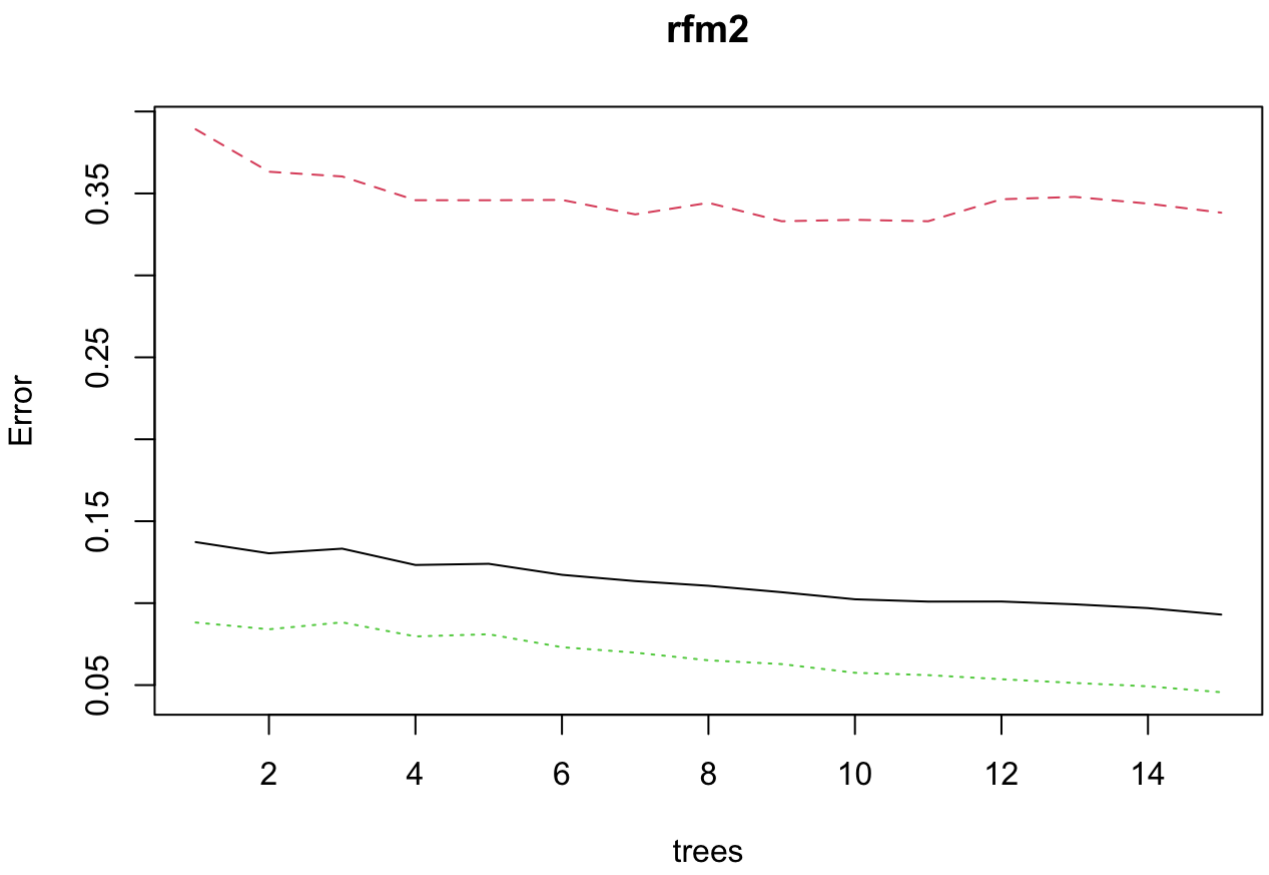
The table above allows us to see the percentages of our prediction model with 2 trees.

Testing with 15 Trees

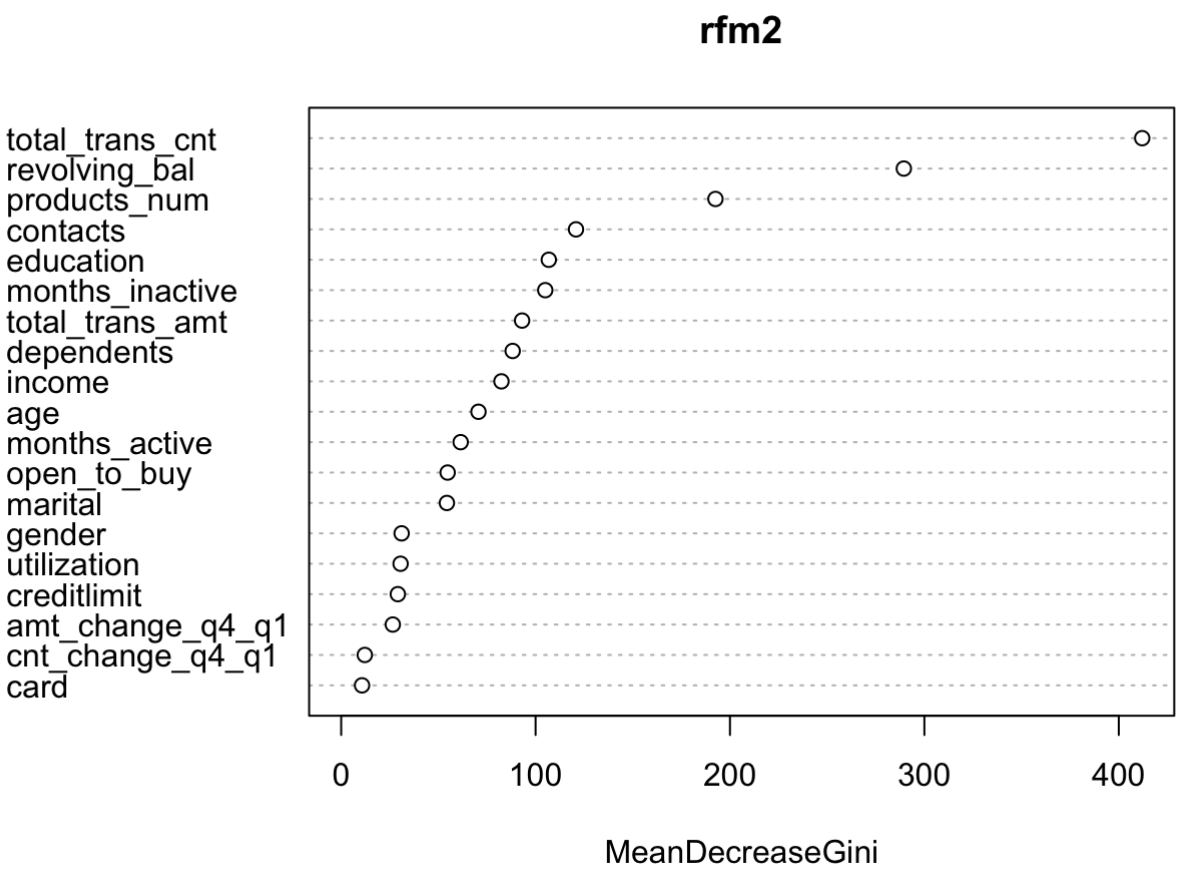
```
set.seed(1111)
rftrain = train[-c(21:22)]
rfm2 <- randomForest(attrition_flag~., data=rftrain, ntree=15)
print(rfm2)
```

```
##
## Call:
##  randomForest(formula = attrition_flag ~ ., data = rftrain, ntree = 15)
##                Type of random forest: classification
##                      Number of trees: 15
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 9.3%
## Confusion matrix:
##                    Attrited Customer Existing Customer class.error
## Attrited Customer               759               388  0.33827376
## Existing Customer               271              5666  0.04564595
```

```
plot(rfm2)
```




```
varImpPlot(rfm2)
```




```
predRF2 <- predict(rfm2, train, type=c("class"))
train$pred2 = predRF2
pred2 = train$pred2
df_rmf = (data.frame(actual,pred2))
confusionMatrix(df_rmf$actual,df_rmf$pred2)
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Attrited Customer Existing Customer
##   Attrited Customer              1123                25
##   Existing Customer                10              5931
##
##                Accuracy : 0.9951
##                  95% CI : (0.9931, 0.9966)
##     No Information Rate : 0.8402
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9817
##
##  Mcnemar's Test P-Value : 0.01796
##
##             Sensitivity : 0.9912
##             Specificity : 0.9958
##          Pos Pred Value : 0.9782
##          Neg Pred Value : 0.9983
##              Prevalence : 0.1598
##          Detection Rate : 0.1584
##    Detection Prevalence : 0.1619
##       Balanced Accuracy : 0.9935
##
##        'Positive' Class : Attrited Customer
##
```

This plot 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.

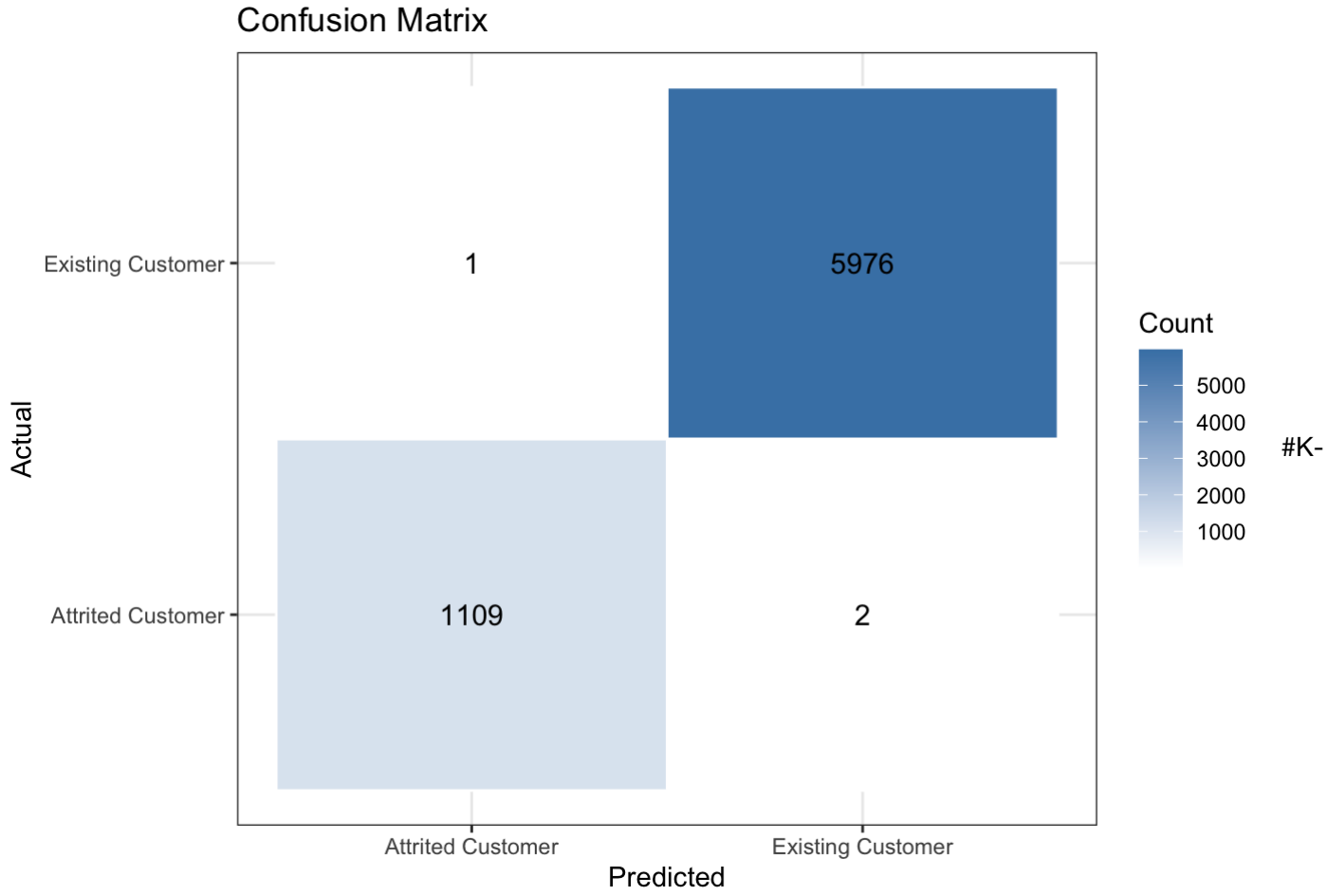
With 15 trees, the accuracy rate is 99%, which is currently our most accurate data model. Based on the confusion matrix, the random forest predicted 3 incorrectly.

```
# recreate confusion matrix numbers
conf_mat <- matrix(c(1109, 2, 1, 5976), nrow = 2, byrow = TRUE)
colnames(conf_mat) <- c("Attrited Customer", "Existing Customer")
rownames(conf_mat) <- c("Attrited Customer", "Existing Customer")
# Convert confusion matrix to dataframe
conf_df <- as.data.frame.matrix(conf_mat)
conf_df$actual <- rownames(conf_df)
# Melt dataframe for plotting
conf_melt <- melt(conf_df, id.vars = "actual")
```

```
## Warning in melt(conf_df, id.vars = "actual"): The melt generic in data.table
## has been passed a data.frame and will attempt to redirect to the relevant
## reshape2 method; please note that reshape2 is deprecated, and this redirection
## is now deprecated as well. To continue using melt methods from reshape2 while
## both libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(conf_df). In the next version, this warning will become an error.
```

```
# Plot heatmap with counts
ggplot(data = conf_melt, aes(x = variable, y = actual, fill = value)) +
  geom_tile(color = "white", size = 0.5) +
  scale_fill_gradient(low = "white", high = "steelblue", name = "Count") +
  geom_text(aes(label = value), color = "black", size = 4) +
  theme_bw() +
  labs(title = "Confusion Matrix", x = "Predicted", y = "Actual")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Means Clustering

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

```
## '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 ...
##  $ total_trans_cnt : Factor w/ 8 levels "Under 20","20s",..: 3 2 1 1 2 2 2 2 2 2 ...
##  $ cnt_change_q4_q1: Factor w/ 3 levels "Low","Medium",..: 2 3 2 2 3 1 1 1 1 1 ...
##  $ utilization     : Factor w/ 3 levels "Low","Medium",..: 1 1 1 3 1 1 1 1 1 1 ...
```

#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)
```

```
#kmeans_ccdata$attrition_flag<-as.numeric(kmeans_ccdata$attrition_flag)
```

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

```
kmeans_ccdata<-cbind(ccdata_cor$CLIENTNUM,kmeans_ccdata)
```

```
head(kmeans_ccdata)
```

```
##   ccdata_cor$CLIENTNUM    attrition_flag age gender dependents education
## 1            768805383 Existing Customer   3      2          4         4
## 2            818770008 Existing Customer   3      1          6         3
## 3            713982108 Existing Customer   4      2          4         3
## 4            769911858 Existing Customer   2      1          5         4
## 5            709106358 Existing Customer   2      2          4         6
## 6            713061558 Existing Customer   3      2          3         3
##   marital income card months_active products_num months_inactive contacts
## 1       2      3    1             3            5                2        4
## 2       3      5    1             4            6                2        3
## 3       2      4    1             3            4                2        1
## 4       4      5    1             3            3                5        2
## 5       2      3    1             2            5                2        1
## 6       2      2    1             3            3                2        3
##   creditlimit revolving_bal open_to_buy amt_change_q4_q1 total_trans_amt
## 1           2             2           3                2               1
## 2           1             3           2                2               1
## 3           1             1           1                3               1
## 4           1             6           1                2               1
## 5           1             1           1                2               1
## 6           1             3           1                2               1
##   total_trans_cnt cnt_change_q4_q1 utilization
## 1               3                2           1
## 2               2                3           1
## 3               1                2           1
## 4               1                2           3
## 5               2                3           1
## 6               2                1           1
```

Adding Client Number as Row Name

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

```
head(kmeans_ccdata)
```

```
##              attrition_flag age gender dependents education marital income card
## 768805383 Existing Customer   3      2          4         4       2      3    1
## 818770008 Existing Customer   3      1          6         3       3      5    1
## 713982108 Existing Customer   4      2          4         3       2      4    1
## 769911858 Existing Customer   2      1          5         4       4      5    1
## 709106358 Existing Customer   2      2          4         6       2      3    1
## 713061558 Existing Customer   3      2          3         3       2      2    1
##           months_active products_num months_inactive contacts creditlimit
## 768805383             3            5                2        4           2
## 818770008             4            6                2        3           1
## 713982108             3            4                2        1           1
## 769911858             3            3                5        2           1
## 709106358             2            5                2        1           1
## 713061558             3            3                2        3           1
##           revolving_bal open_to_buy amt_change_q4_q1 total_trans_amt
## 768805383             2           3                2               1
## 818770008             3           2                2               1
## 713982108             1           1                3               1
## 769911858             6           1                2               1
## 709106358             1           1                2               1
## 713061558             3           1                2               1
##           total_trans_cnt cnt_change_q4_q1 utilization
## 768805383               3                2           1
## 818770008               2                3           1
## 713982108               1                2           1
## 769911858               1                2           3
## 709106358               2                3           1
## 713061558               2                1           1
```

#taking a backup and removing attrition_flag

```
kmeans_ccdata_bkup<-kmeans_ccdata
```

```
kmeans_ccdata$attrition_flag<-NULL
```

Executing k-means with k=2

```
set.seed(50)
kcluster<-kmeans(kmeans_ccdata,2)
kmeans_ccdata$kcluster<-as.factor(kcluster$cluster)
```

```
str(kcluster)
```

```
## 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"
```

```
kcluster$centers
```

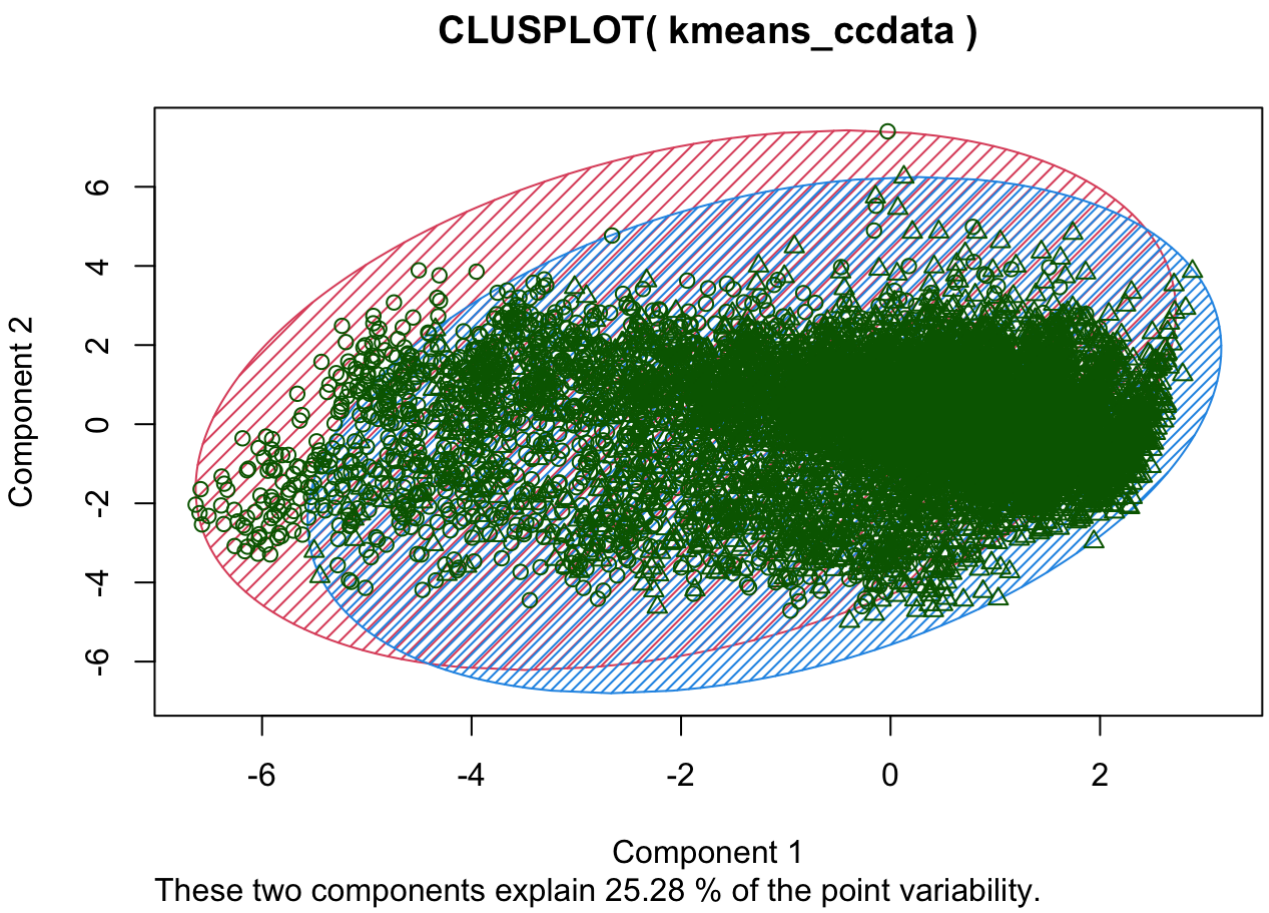
```
##        age   gender dependents education  marital   income     card
## 1 3.077220 1.537403   3.412162  5.870656 2.484073 3.711149 1.307432
## 2 3.083904 1.424870   3.300518  2.867792 2.449106 3.969748 1.091426
##   months_active products_num months_inactive contacts creditlimit revolving_bal
## 1      3.023407     3.748069        3.333012 3.473697    1.830598      3.370415
## 2      3.033595     3.857262        3.346816 3.442587    1.285308      3.350159
##   open_to_buy amt_change_q4_q1 total_trans_amt total_trans_cnt cnt_change_q4_q1
## 1    2.402510         1.044884        1.277268        3.791506         1.024373
## 2    1.540532         1.047301        1.171987        3.683604         1.023400
##   utilization
## 1    1.405647
## 2    1.547217
```

```
#kcluster
```

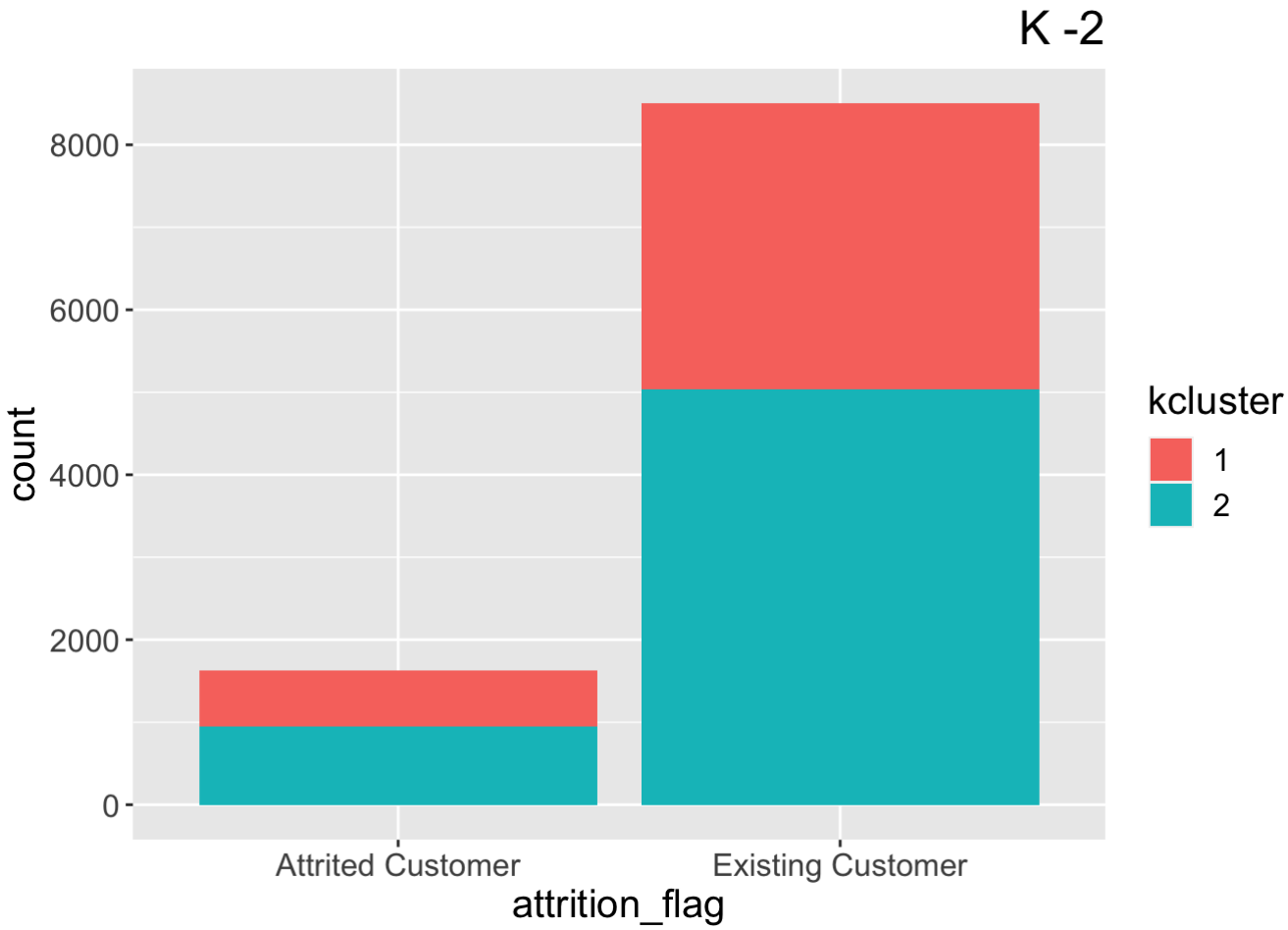
#plotting the clusters

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

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

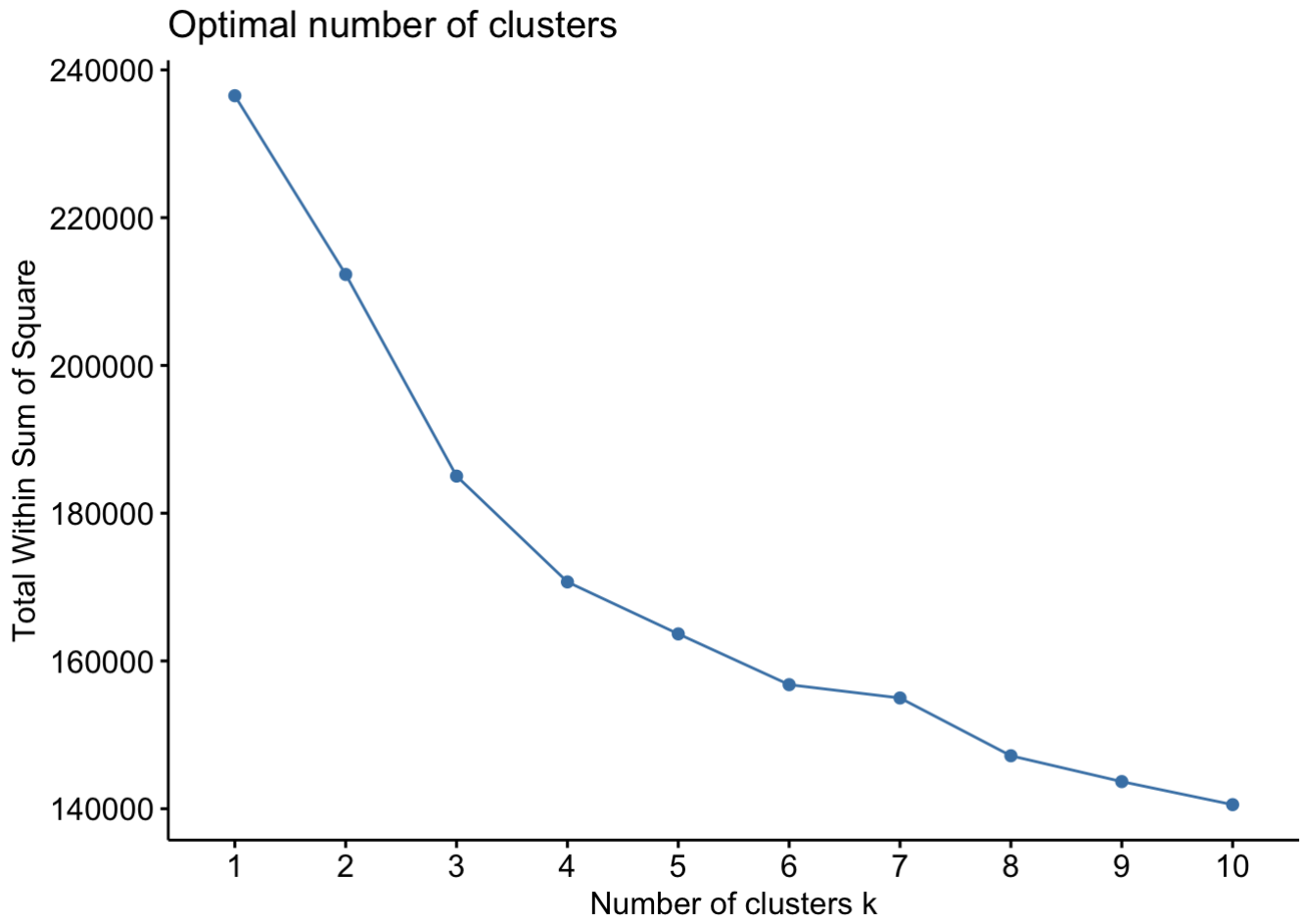



```
ggplot(data=kmeans_ccdata1,aes(x=attrition_flag,fill=kcluster))+
         geom_bar(stat = "count")+
                  labs(title="K -2")+
                  theme(plot.title=element_text(hjust = 1),
                        text=element_text(size=15))
```



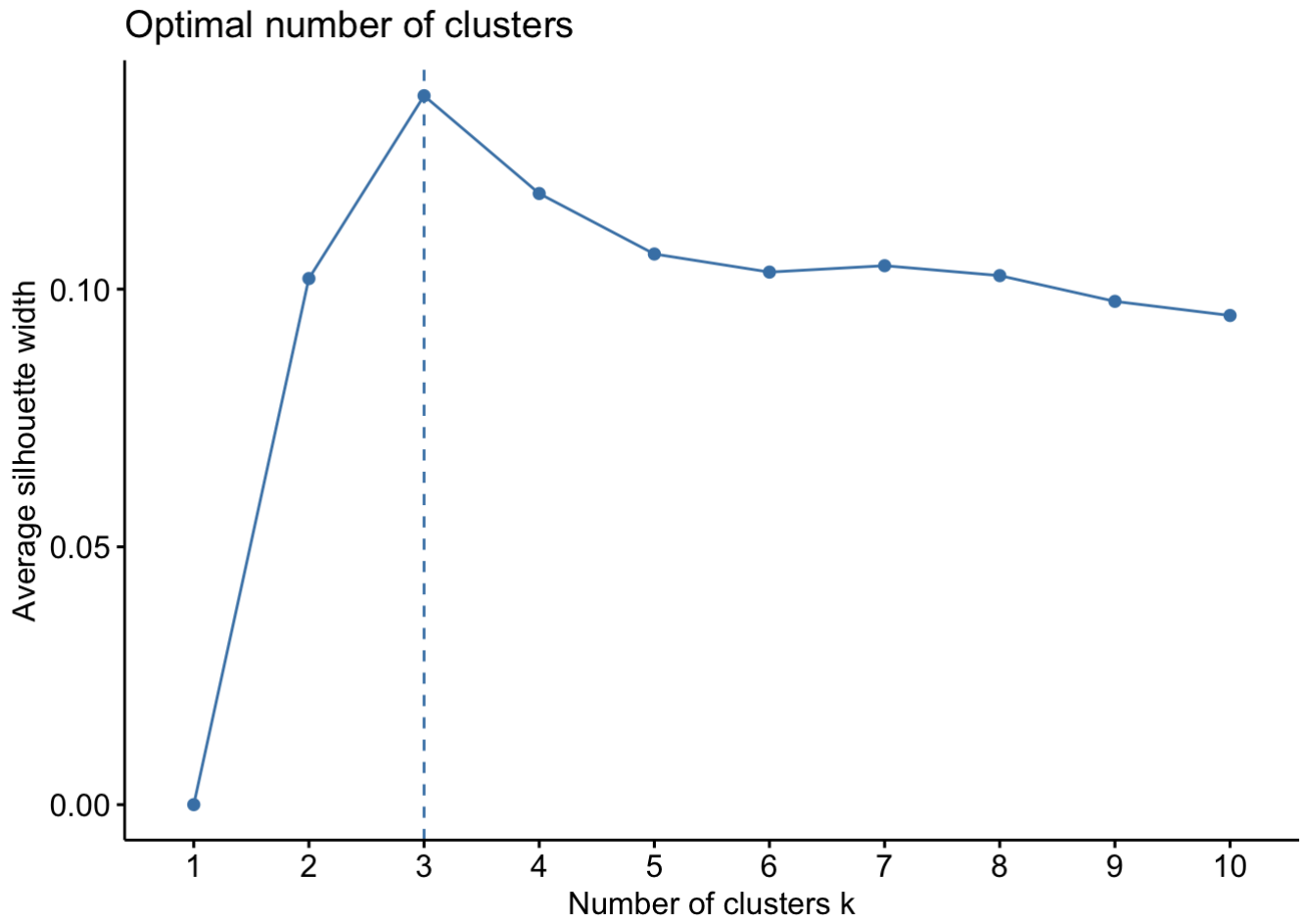
#identifying an optimal value for k-clusters Elbow Method

```
set.seed(123)

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



Silhoutte Method

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

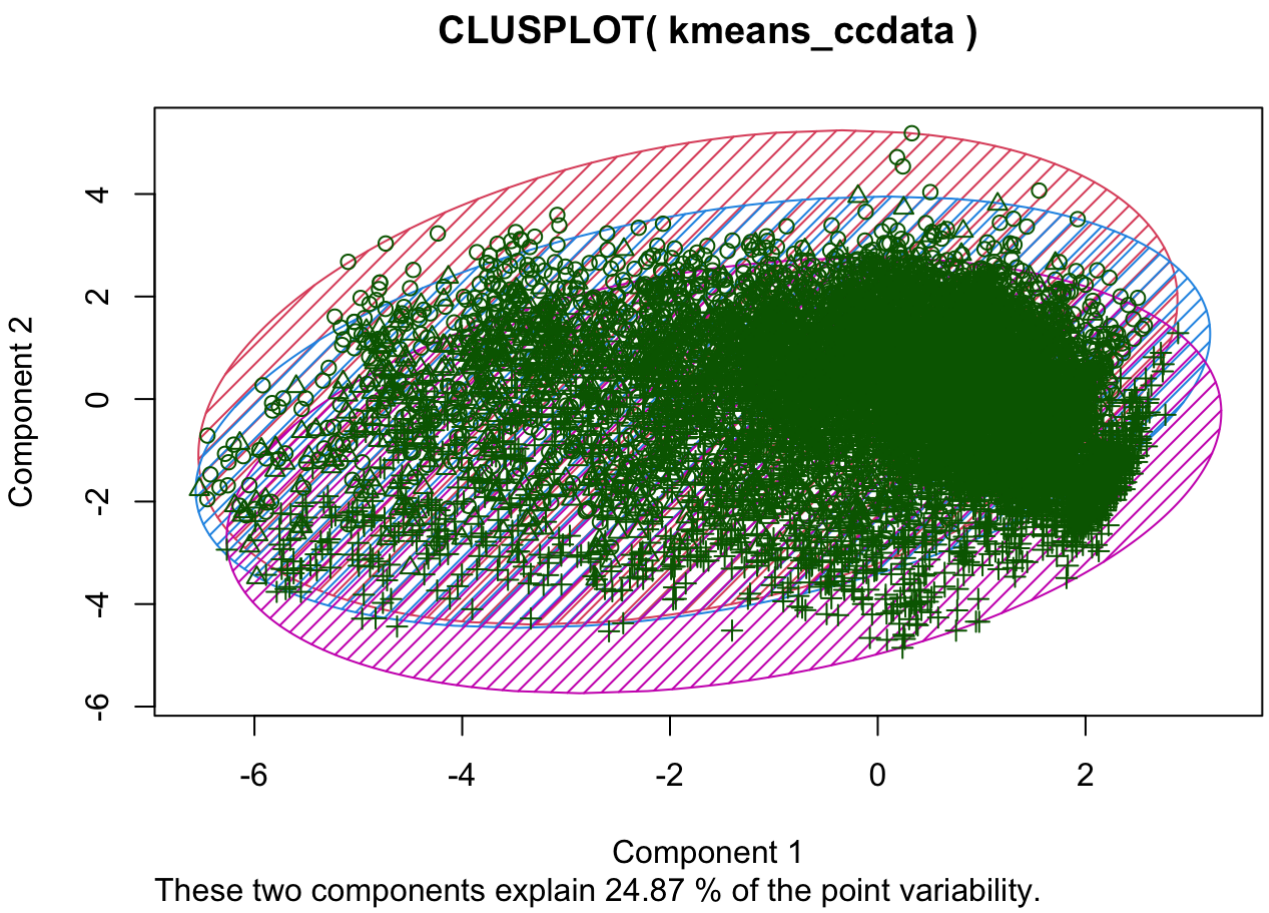


Both Method suggests that k=3 is an ideal choice Rerun with k=3

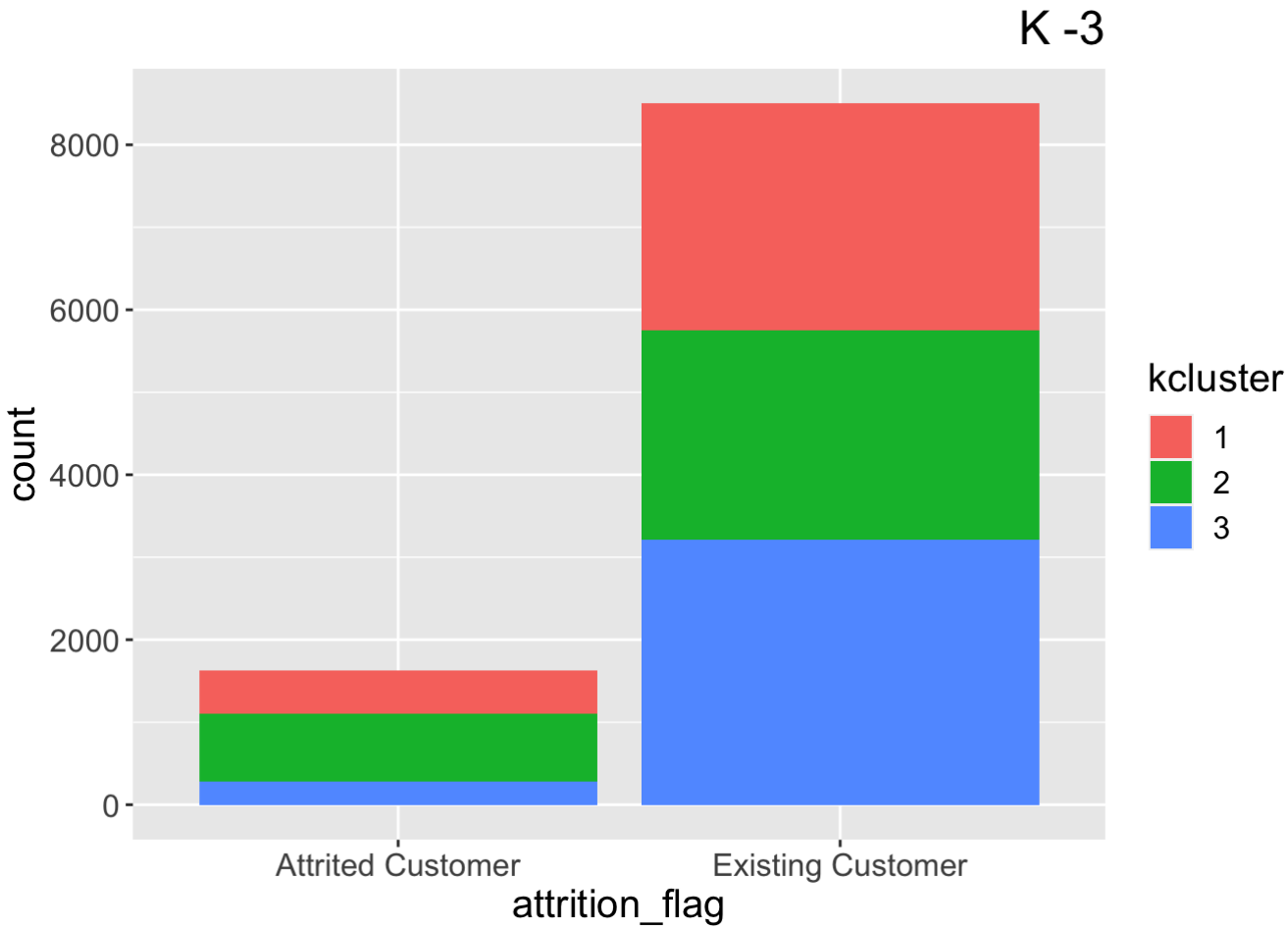
```
set.seed(50)
kcluster<-kmeans(kmeans_ccdata,3)
kmeans_ccdata$kcluster<-as.factor(kcluster$cluster)
```

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

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




```
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))
```



#HAC

```
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 ...
##  $ utilization     : num  1 1 1 3 1 1 1 1 1 1 ...
```

```
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 ...
```

#calculating various distances:

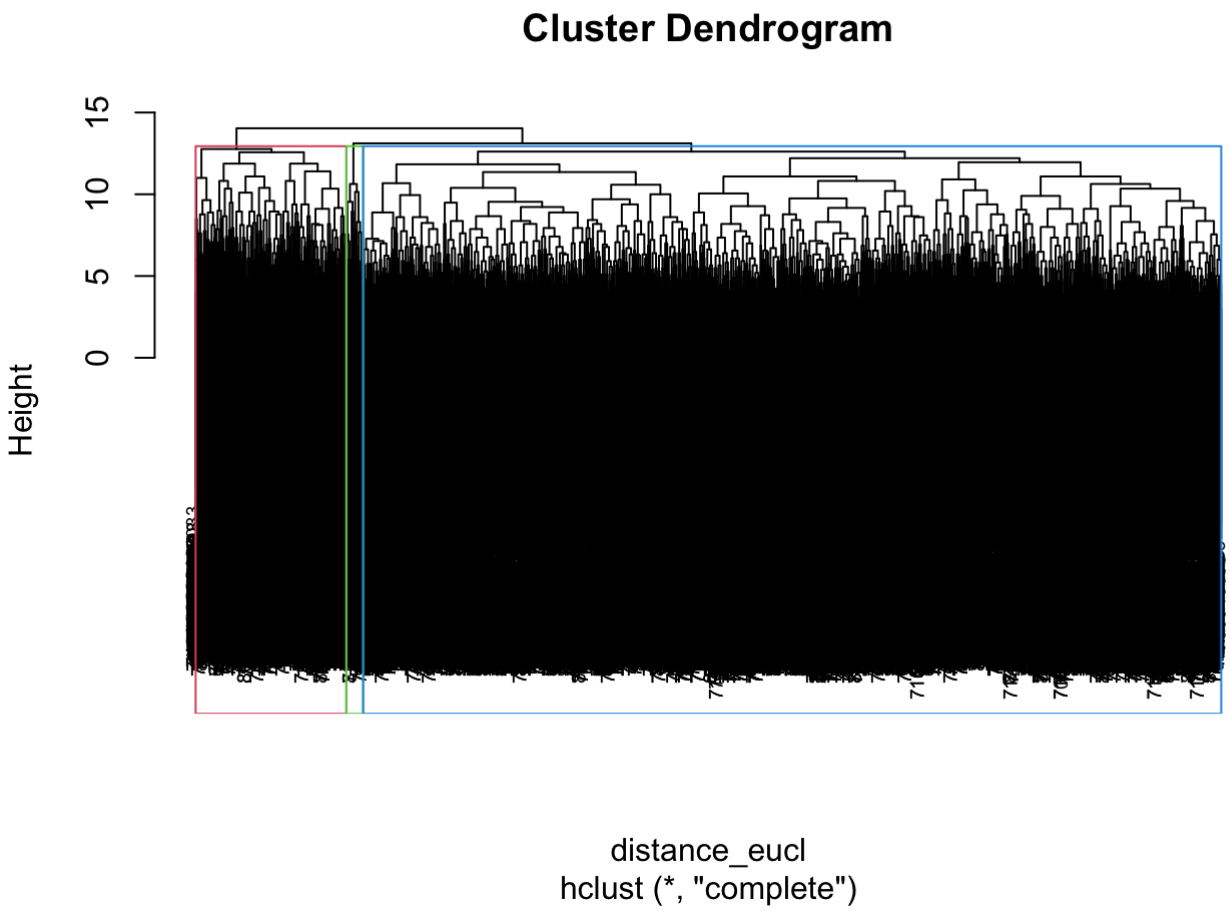
```
distance_eucl<-dist(ccdata_hac1,method='euclidean')
distance_man<-dist(ccdata_hac1,method='manhattan')
distance_mink<-dist(ccdata_hac1,method='minkowski')
distance_can<-dist(ccdata_hac1,method='canberra')
distance_max<-dist(ccdata_hac1,method='maximum')
distance_bin<-dist(ccdata_hac1,method='binary')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

```
head(ccdata_hac1)
```

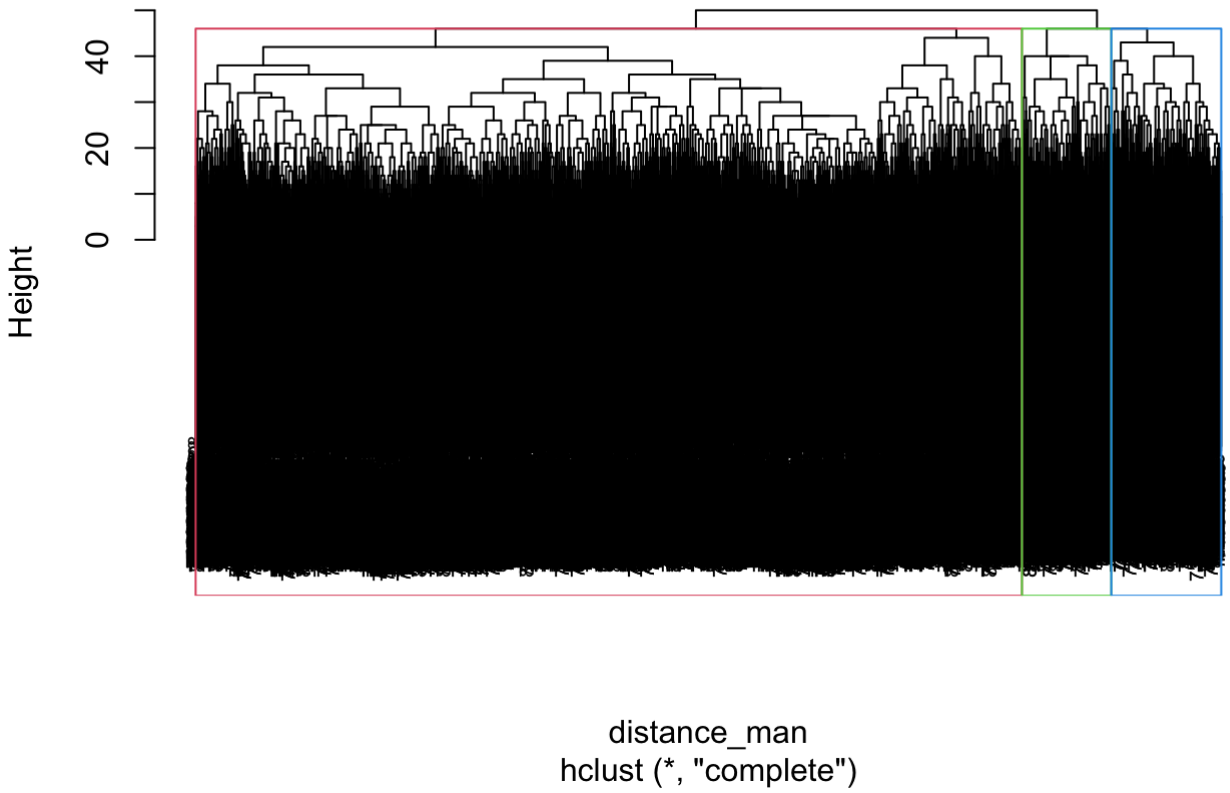
```
##           age gender dependents education marital income card months_active
## 768805383   3      2          4         4       2      3    1             3
## 818770008   3      1          6         3       3      5    1             4
## 713982108   4      2          4         3       2      4    1             3
## 769911858   2      1          5         4       4      5    1             3
## 709106358   2      2          4         6       2      3    1             2
## 713061558   3      2          3         3       2      2    1             3
##           products_num months_inactive contacts creditlimit revolving_bal
## 768805383            5               2        4           2             2
## 818770008            6               2        3           1             3
## 713982108            4               2        1           1             1
## 769911858            3               5        2           1             6
## 709106358            5               2        1           1             1
## 713061558            3               2        3           1             3
##           open_to_buy amt_change_q4_q1 total_trans_amt total_trans_cnt
## 768805383           3                2               1               3
## 818770008           2                2               1               2
## 713982108           1                3               1               1
## 769911858           1                2               1               1
## 709106358           1                2               1               2
## 713061558           1                2               1               2
##           cnt_change_q4_q1 utilization
## 768805383                2           1
## 818770008                3           1
## 713982108                2           1
## 769911858                2           3
## 709106358                3           1
## 713061558                1           1
```

#hac with euclidean distance

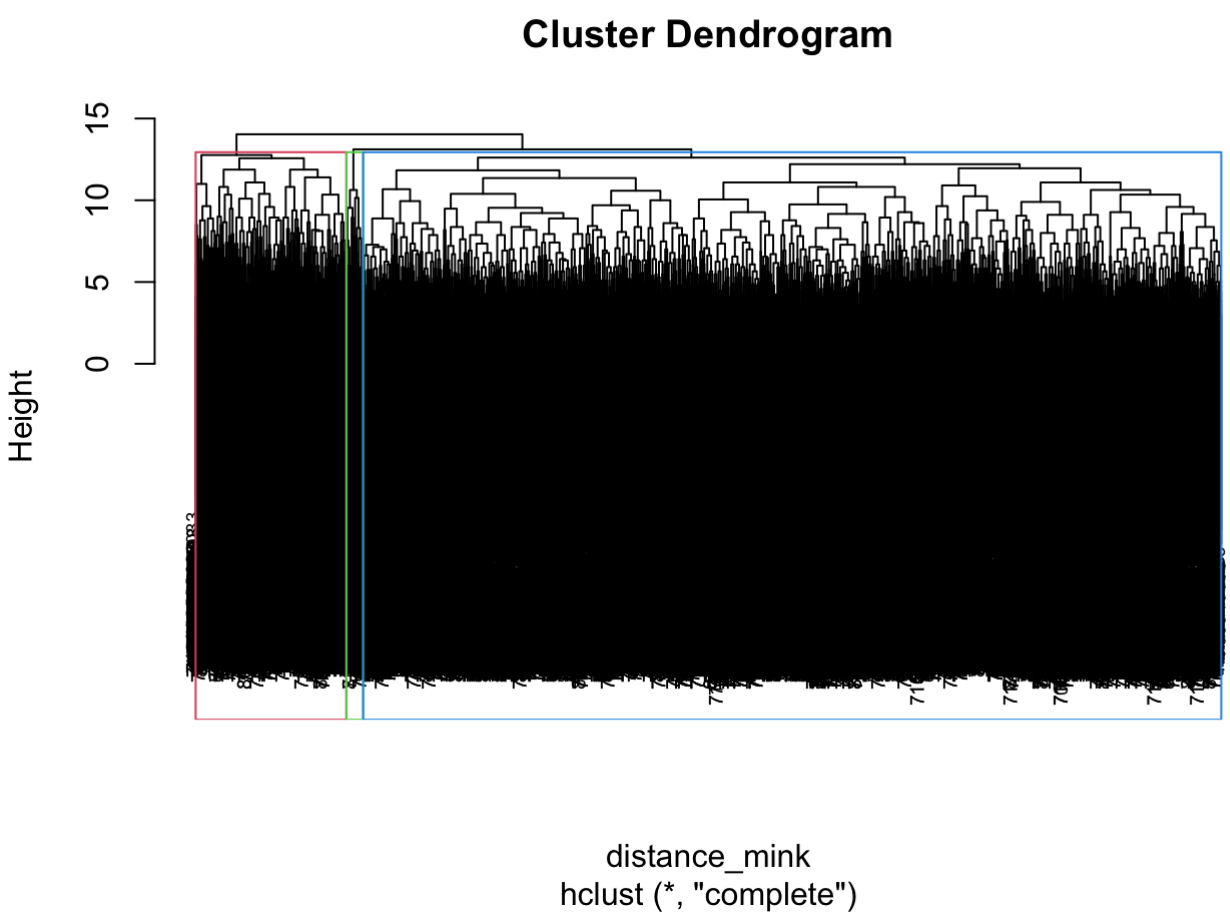
```
hac_eucl=hclust(distance_eucl,method="complete")
plot(hac_eucl,cex=0.6,hang=1)
rect.hclust(hac_eucl,k=3,border=2:5)
```




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




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




```
#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)
```

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

library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.16.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendext
end/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

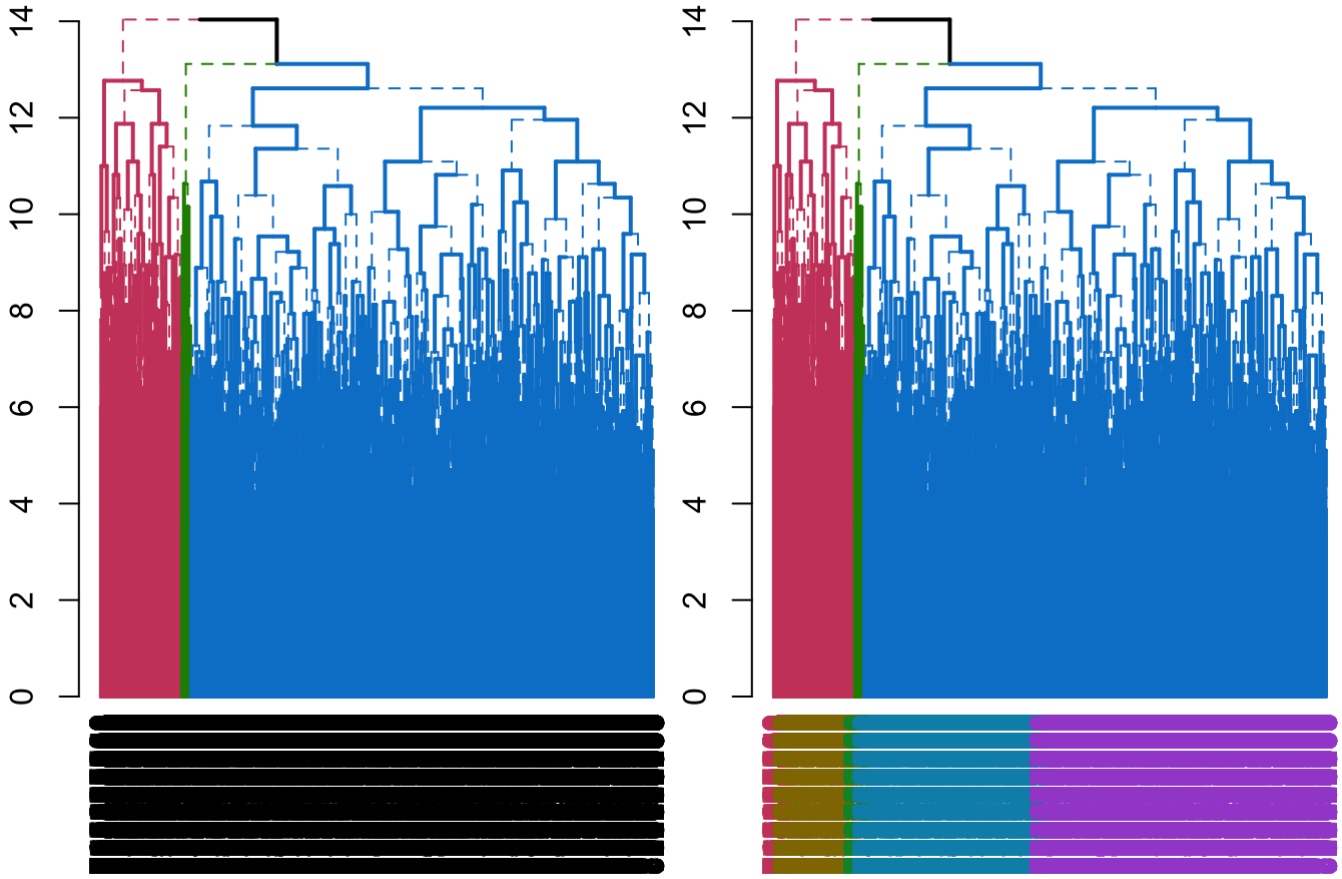
```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:rpart':
##
##     prune
```

```
## The following object is masked from 'package:data.table':
##
##     set
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```
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)
```


0
2
4
6
8
10
12
14
0
2
4
6
8
10
12
14