Predicting Credit Card Customer Churn with Data in R

Janhvi Goje

Introduction

Purpose

Our purpose with this project is to **predict if a customer is going to close their credit card account**, so that we can use this information to maximise customer retention by introducing more convenient services in order to convince the customer against leaving the bank.

Data Description

First, let's start off by looking at all the data that is provided to us and how we can use them in our project.

- **Bank Account Data**: This dataset provides a comprehensive view of customer interactions, account details, and financial behaviors over a period, which are invaluable for predicting customer churn, specifically in terms of closing credit card accounts.
- **Purchase Data**: This dataset offers a rich set of quantitative variables reflecting customers' annual expenditures across various product categories. This data will be analysed to identify homogeneous groups of customers with similar purchasing behaviors.

Part 1 - Classification

Let's import our `bank_accounts_train.csv` dataset.

```r
options(warn = -1)
bank_accounts <- read.csv("/Users/janvigoje/Desktop/bank_accounts_train.csv", colClasses = "character")
head(bank_accounts)
```

```
##   CLIENTNUM Customer_Age Gender Dependent_count Education_Level Marital_Status
## 1 721038408           38      F               2     High School        Married
## 2 711968358           55      F               4        Graduate         Single
## 3 719174433           36      F               1         Unknown         Single
## 4 712841883           50      M               3         College         Single
## 5 715980258           36      F               1       Doctorate         Single
## 6 708394908           44      M               0     High School         Single
##   Card_Category Months_on_book Total_Relationship_Count Months_Inactive_12_mon
## 1          Blue             27                        4                      1
## 2          Blue             44                        1                      2
## 3          Blue             24                        2                      1
## 4        Silver             36                        3                      3
## 5          Blue             25                        1                      2
## 6          Blue             33                        5                      1
##   Contacts_Count_12_mon Credit_Limit Total_Revolving_Bal Avg_Open_To_Buy
## 1                     2         1830                   0            1830
## 2                     1         2638                1423            1215
## 3                       2         1735                   0             1735
## 4                       2        34516                2517            31999
## 5                       3         2835                2511              324
## 6                       3         2337                1442              895
##   Total_Amt_Chng_Q4_Q1 Total_Trans_Amt Total_Trans_Ct Total_Ct_Chng_Q4_Q1
## 1                  736            1741             43                 654
## 2                  551            4153             77                 925
## 3                   74            2467             35                 346
## 4                  735            7155             66                  65
## 5                  731            4811             77                 925
## 6                  659            3945             65                 548
##   Avg_Utilization_Ratio            Income Closed_Account
## 1                     0 169.77966777049005              0
## 2                   539  87.02969695441425              0
## 3                     0  140.2794818393886              1
## 4                    73  90.16357845626771              1
## 5                   886 101.55167812481524              0
## 6                   617 46.926825870759785              0
```

Data Cleaning (Q1)

We are informed that the categorical features have some missing values denoted by ”Unknown”. Instinctively, we can simply remove these rows; but this might lead to loss of information. Therefore, we replace Unknown with ‘NA’.

```
cols <- colnames(bank_accounts)
Data <- bank_accounts                       # Duplicate data frame
Data[Data == "Unknown"] <- NA                  # Replace particular value with NA
```

Next, we turn our categorical variables into factors.

```
categ = c("Education_Level", "Marital_Status", "Card_Category","Gender","Closed_Account")
for(i in 1:ncol(Data)){
  if (cols[i] %in% categ){
    Data[,i] = as.factor(Data[,i])
  }
}
```

Next, we turn the strings that are numerical, into numeric.

```
for(i in 1:ncol(Data)){
  if (!(cols[i] %in% categ)){
    Data[,i] = as.numeric(Data[,i])
  }
}
```

```
head(Data)
```

```
##   CLIENTNUM Customer_Age Gender Dependent_count Education_Level Marital_Status
## 1 721038408           38      F               2     High School        Married
## 2 711968358           55      F               4        Graduate         Single
## 3 719174433           36      F               1            <NA>         Single
## 4 712841883           50      M               3         College         Single
## 5 715980258           36      F               1       Doctorate         Single
## 6 708394908           44      M               0     High School         Single
##   Card_Category Months_on_book Total_Relationship_Count Months_Inactive_12_mon
## 1          Blue             27                        4                      1
## 2         Blue           44                      1                      2
## 3         Blue           24                      2                      1
## 4       Silver           36                      3                      3
## 5         Blue           25                      1                      2
## 6         Blue           33                      5                      1
##   Contacts_Count_12_mon Credit_Limit Total_Revolving_Bal Avg_Open_To_Buy
## 1                     2         1830                   0            1830
## 2                     1         2638                1423            1215
## 3                     2         1735                   0            1735
## 4                     2        34516                2517           31999
## 5                     3         2835                2511             324
## 6                     3         2337                1442             895
##   Total_Amt_Chng_Q4_Q1 Total_Trans_Amt Total_Trans_Ct Total_Ct_Chng_Q4_Q1
## 1                  736            1741             43                 654
## 2                  551            4153             77                 925
## 3                   74            2467             35                 346
## 4                  735            7155             66                  65
## 5                  731            4811             77                 925
## 6                  659            3945             65                 548
##   Avg_Utilization_Ratio    Income Closed_Account
## 1                     0 169.77967              0
## 2                   539  87.02970              0
## 3                     0 140.27948              1
## 4                    73  90.16358              1
## 5                   886 101.55168              0
## 6                   617  46.92683              0
```

Exploratory Data Analysis (Q2)

Let's start with our EDA, looking at the columns we have available.

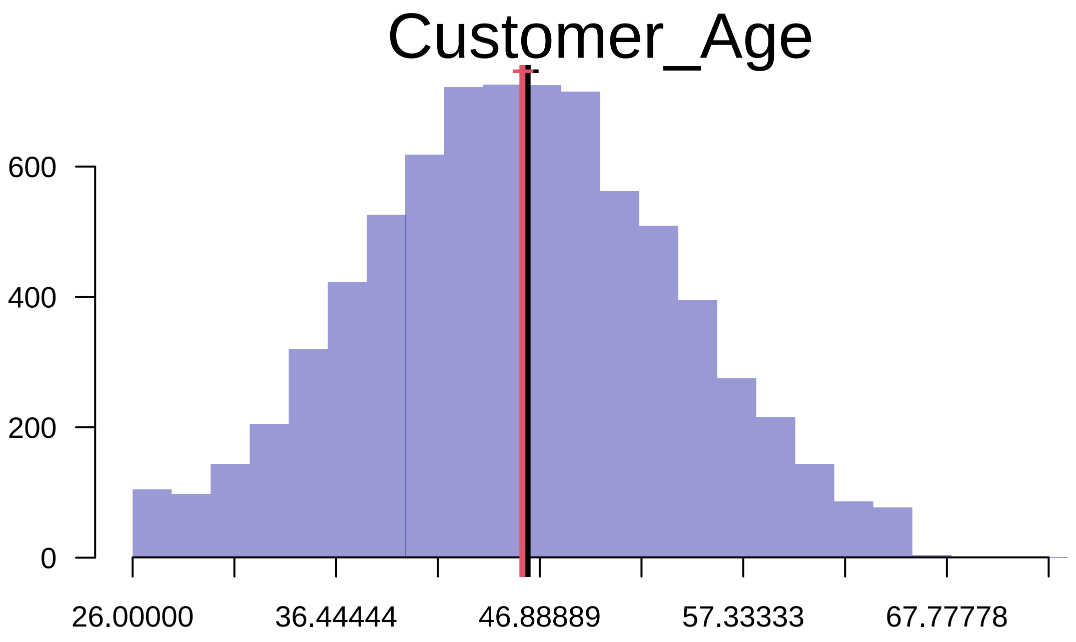
```r
colnames(Data)
```

```
##  [1] "CLIENTNUM"                "Customer_Age"
##  [3] "Gender"                   "Dependent_count"
##  [5] "Education_Level"          "Marital_Status"
##  [7] "Card_Category"            "Months_on_book"
##  [9] "Total_Relationship_Count" "Months_Inactive_12_mon"
## [11] "Contacts_Count_12_mon"    "Credit_Limit"
## [13] "Total_Revolving_Bal"      "Avg_Open_To_Buy"
## [15] "Total_Amt_Chng_Q4_Q1"     "Total_Trans_Amt"
## [17] "Total_Trans_Ct"           "Total_Ct_Chng_Q4_Q1"
## [19] "Avg_Utilization_Ratio"    "Income"
## [21] "Closed_Account"
```

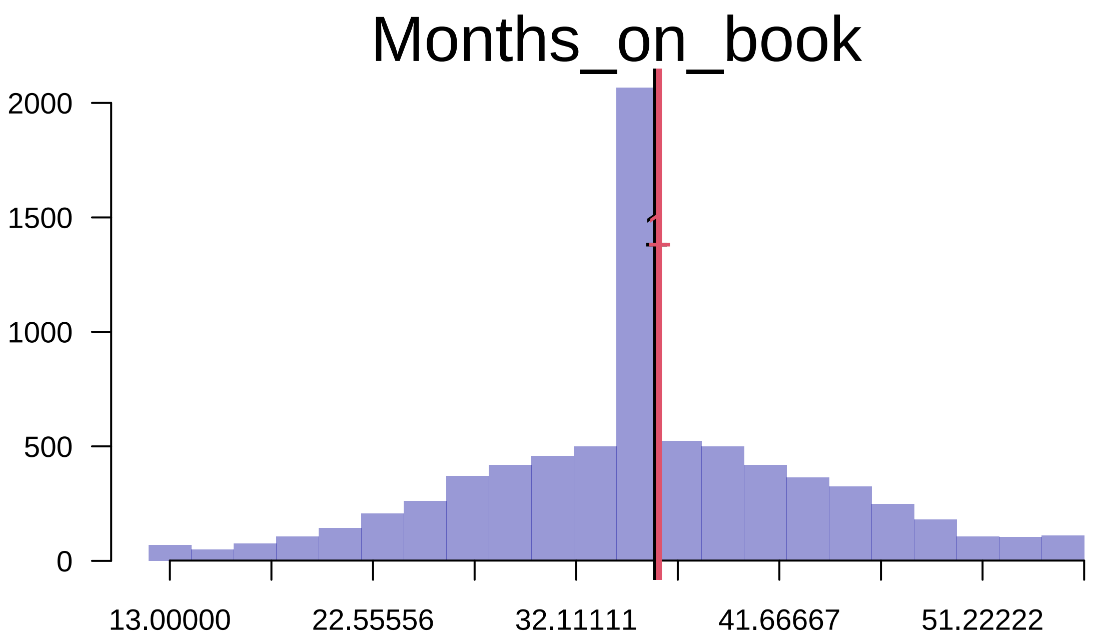
Next, let's make some visualizations. We start by making histograms for all the continuous variables. Note that here, the red line is the median and the black line is the mean.

```r
contcols <- c("Customer_Age", "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",
                     "Income", "Months_on_book")

for (col in colnames(Data)) {
  if (col %in% contcols) {
    print(" ")
    hist(Data[[col]],
         main = paste("Histogram of", col),
         xlab = col,
         breaks = 20,
         freq = TRUE,
         border = 0,
         col = rgb(0, 0, 0.6, 0.4),
         ann = FALSE,
         axes = FALSE)

    axis(1, cex.axis = 0.9, at = seq(min(Data[[col]], na.rm = TRUE), max(Data[[col]], na.rm = TRUE), len
    axis(2, las = 2, cex.axis = 0.9)

    mtext(col, side = 3, cex = 2)

    abline(v = mean(Data[[col]], na.rm = TRUE), lwd = 3)
    text(mean(Data[[col]], na.rm = TRUE), max(0.7 * max(table(cut(Data[[col]], breaks = 20)), na.rm = TR

    abline(v = median(Data[[col]], na.rm = TRUE), lwd = 3, col = 2)
    text(median(Data[[col]], na.rm = TRUE), max(0.7 * max(table(cut(Data[[col]], breaks = 20)), na.rm =
  }
}
```

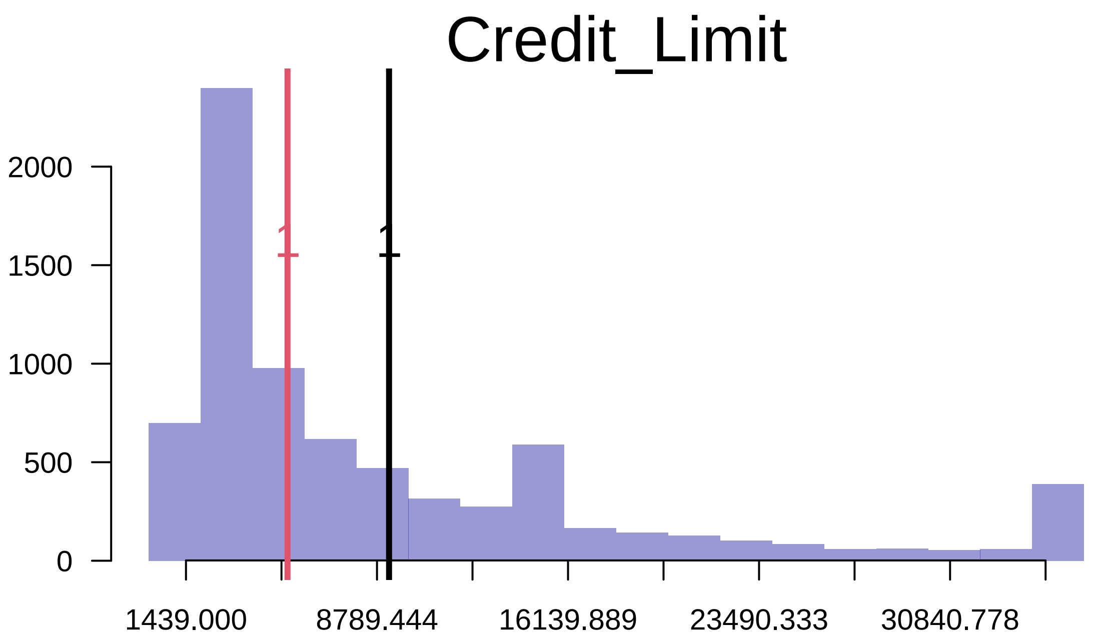
```
## [1] " "
```



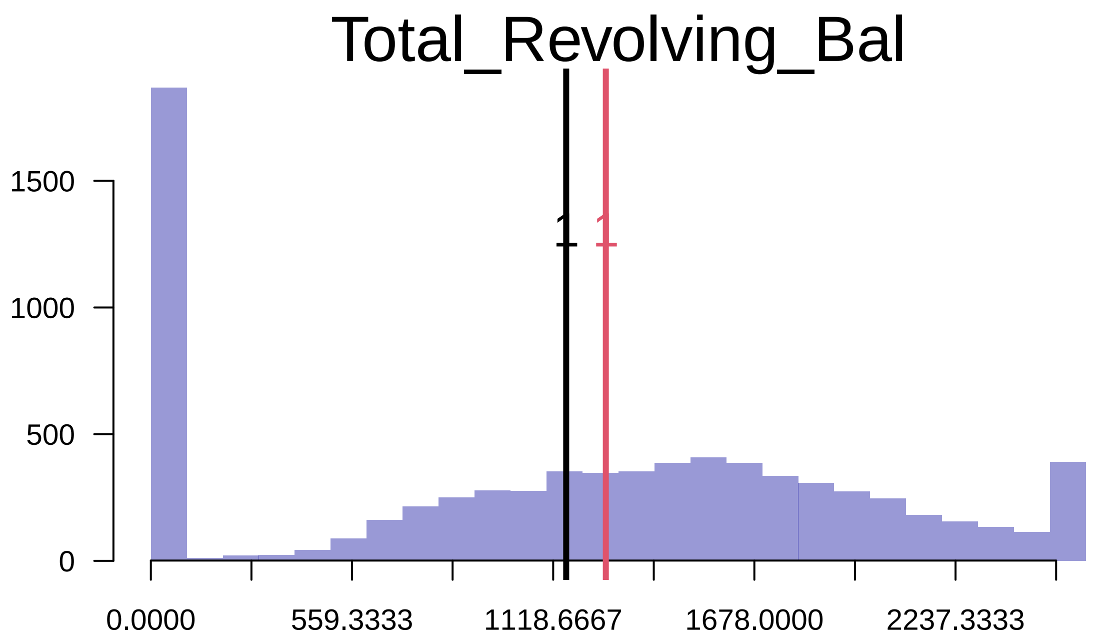
```
## [1] " "
```


Months_on_book
2000
1500
1000
500
0
13.00000
22.55556
32.11111
41.66667
51.22222

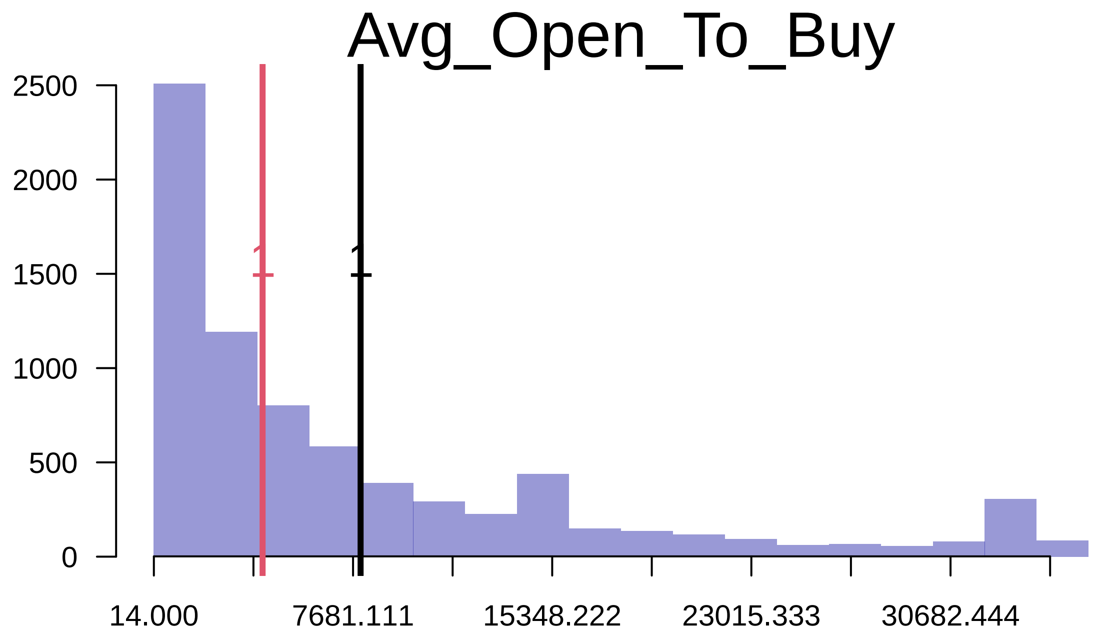
```
## [1] " "
```


Credit_Limit
2000
1500
1000
500
0
1439.000
8789.444
16139.889
23490.333
30840.778

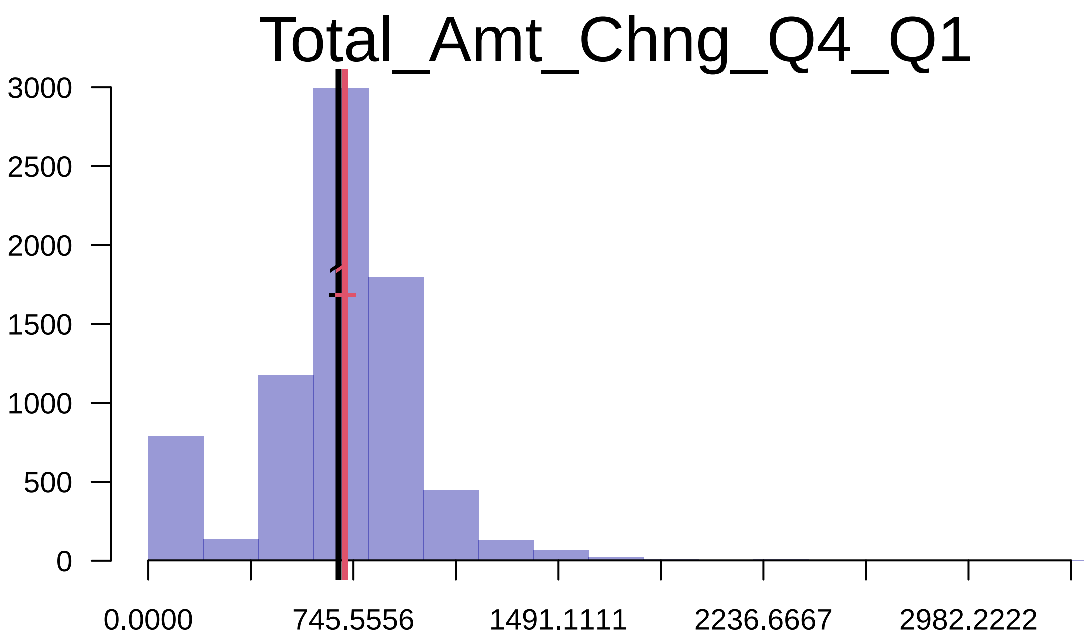
```
## [1] " "
```


Total_Revolving_Bal
1500
1000
500
0
1
1
0.0000
559.3333
1118.6667
1678.0000
2237.3333

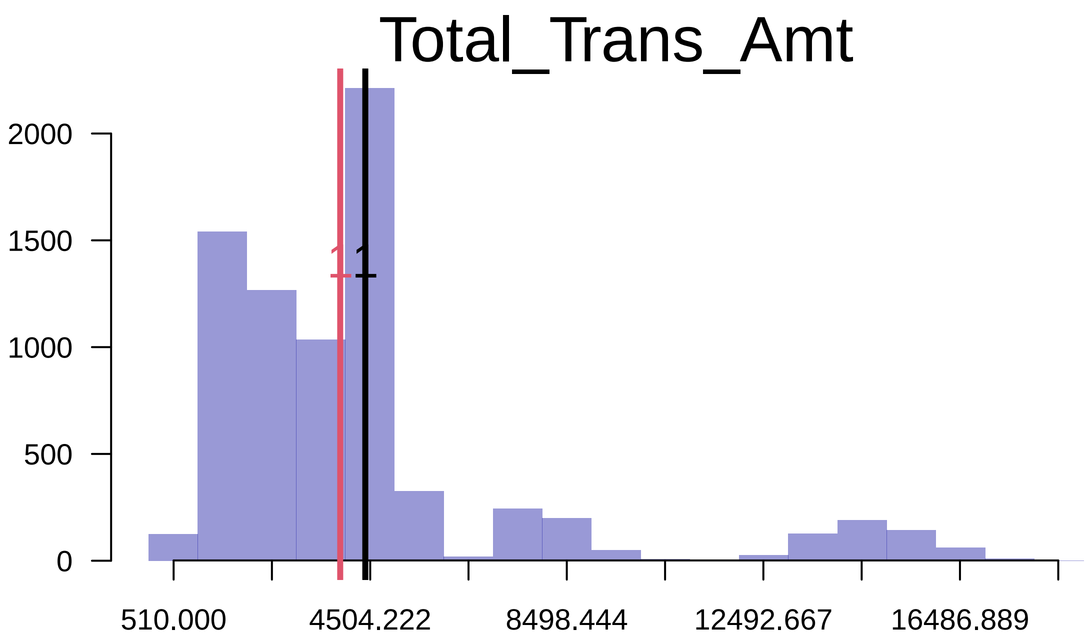
```
## [1] " "
```


Avg_Open_To_Buy
2500
2000
1500
1000
500
0
1
1
14.000
7681.111
15348.222
23015.333
30682.444

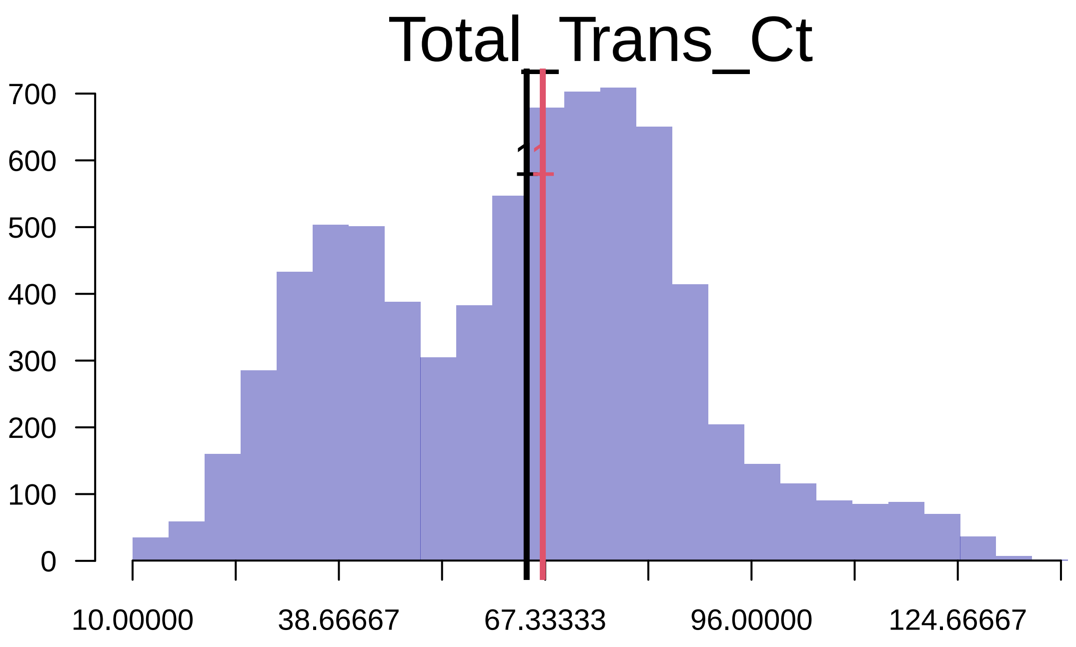
```
## [1] " "
```


Total_Amt_Chng_Q4_Q1
3000
2500
2000
1500
1000
500
0
0.0000
745.5556
1491.1111
2236.6667
2982.2222

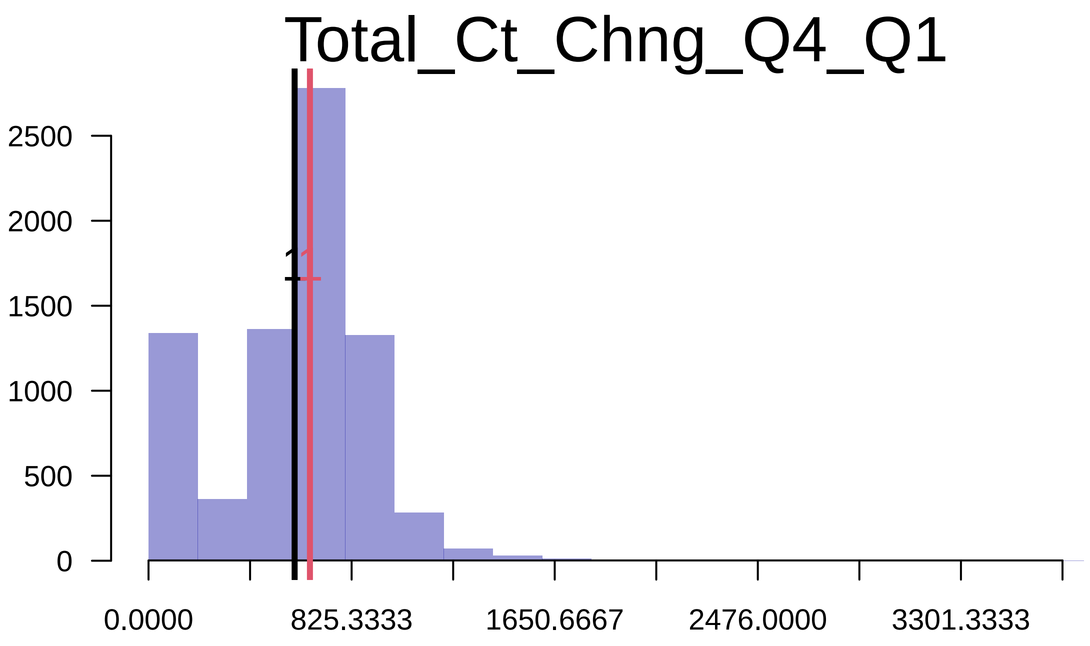
```
## [1] " "
```


Total_Trans_Amt
2000
1500
1000
500
0
510.000
4504.222
8498.444
12492.667
16486.889

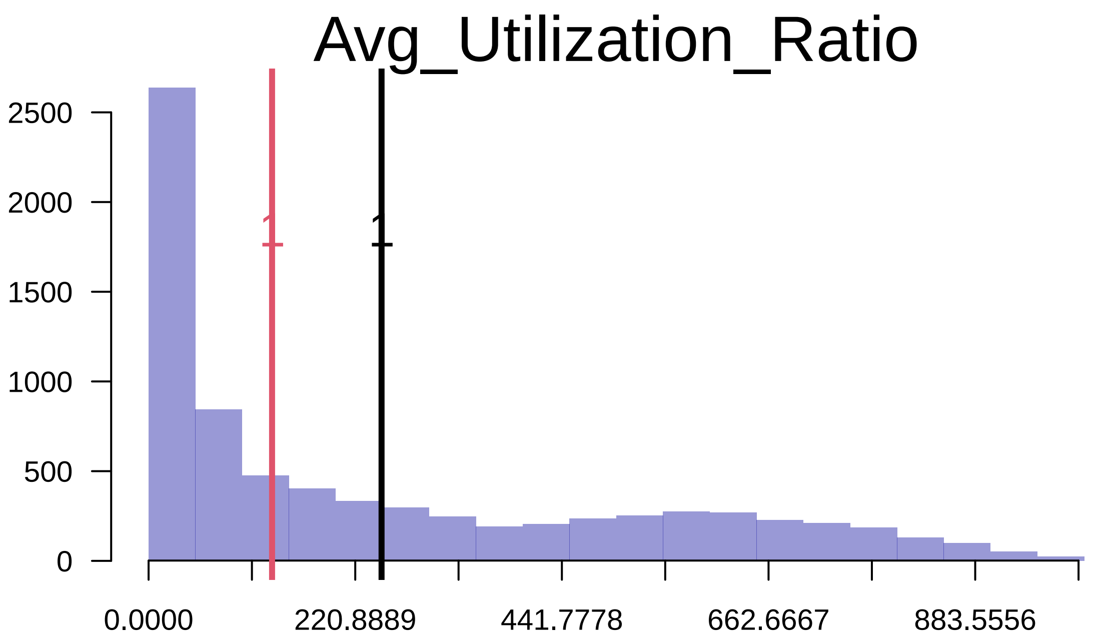
```
## [1] " "
```


Total_Trans_Ct
700
600
500
400
300
200
100
0
10.00000
38.66667
67.33333
96.00000
124.66667

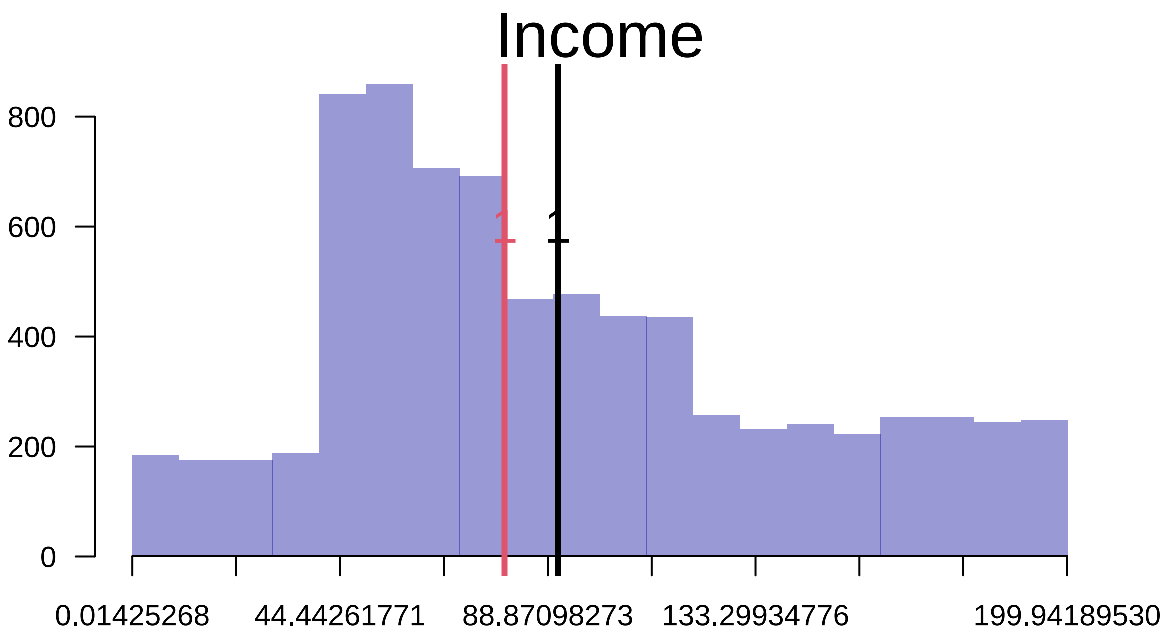
```
## [1] " "
```


Total_Ct_Chng_Q4_Q1
2500
2000
1500
1000
500
0
0.0000
825.3333
1650.6667
2476.0000
3301.3333

```
## [1] " "
```



```
## [1] " "
```



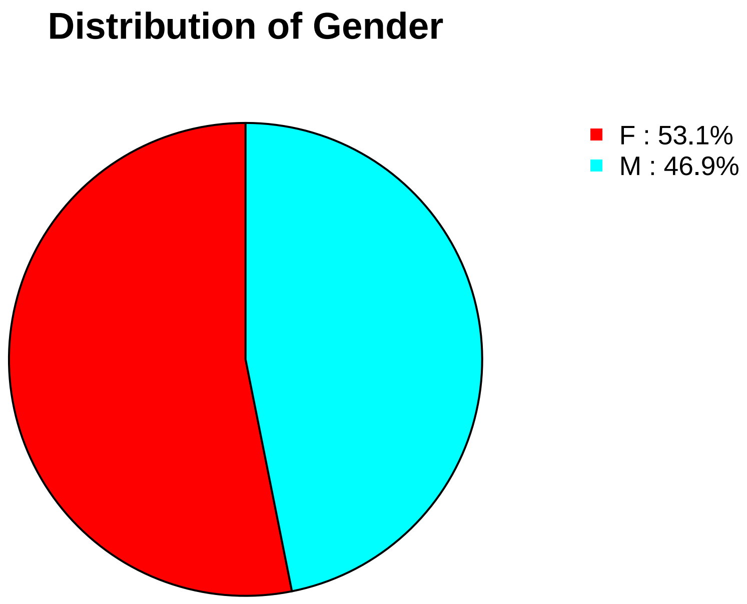
The histograms for ‘Total_Revolving_Bal’, ‘Avg_Open_To_Buy’, ‘Total_Trans_Amt’, ‘Avg_Utilization_Ratio’, and ‘Income’ exhibit right-skewed distributions, indicating that most customers have lower balances, credit available, transaction amounts, utilization ratios, and incomes, with fewer customers having higher values in these areas. In contrast, ‘Total_Trans_Ct’ shows a roughly normal distribution, suggesting consistent transaction behavior across customers. The ‘Total_Amt_Chng_Q4_Q1’ histogram also appears symmetrical, centering around the mean, which indicates uniform changes in transaction amounts over time.

Let’s look at our categorical variables now. A pie chart is ideal for categories like marital status and gender with low number of modalities.

```
pivars <- c("Gender", "Education_Level", "Marital_Status", "Card_Category", "Closed_Account")
```

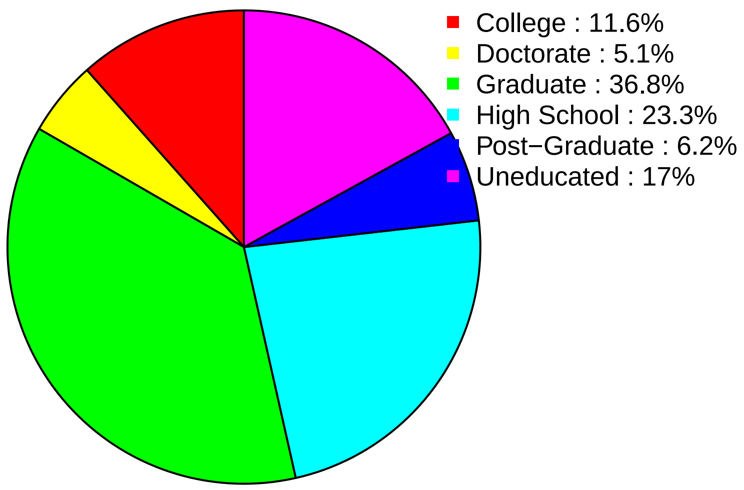
```r
for (var in pivars) {
  print(" ")
  tab <- table(Data[[var]])

  pct <- round((tab / sum(tab)) * 100, 1)

  labels <- paste(pct, "%", sep = "")
  legend_labels <- paste(names(tab), ":", labels)

  pie(x = tab,
      labels = NA,
      main = paste("Distribution of", var),
      col = rainbow(length(tab)),
      cex = 1.2,
      init.angle = 90,
      radius = 1)

  legend("topright",
         legend = legend_labels,
         pch = 15,
         col = rainbow(length(tab)),
         cex = 0.8,
         bty = "n")
}
```

```
## [1] " "
```



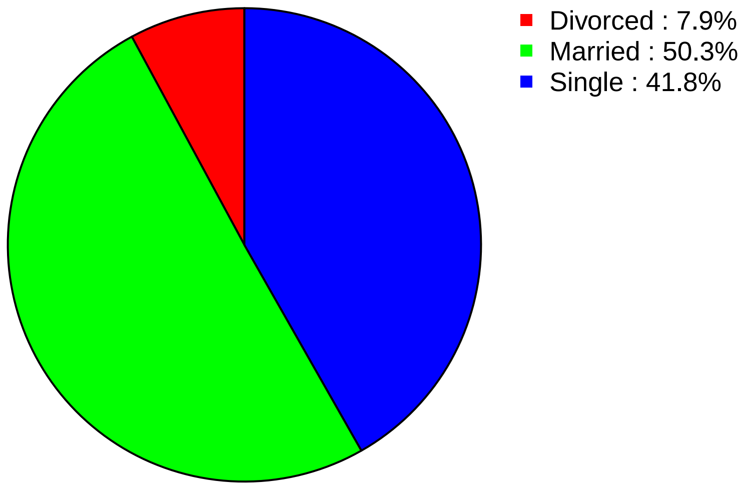
```
## [1] " "
```

Distribution of Education_Level



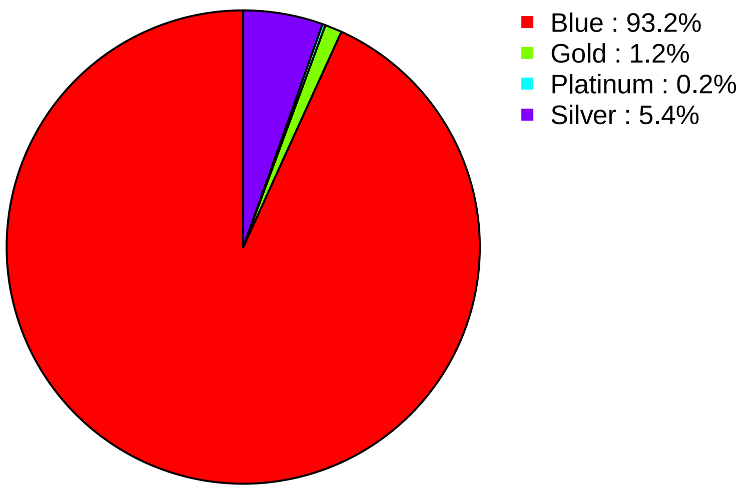
```
## [1] " "
```

Distribution of Marital_Status



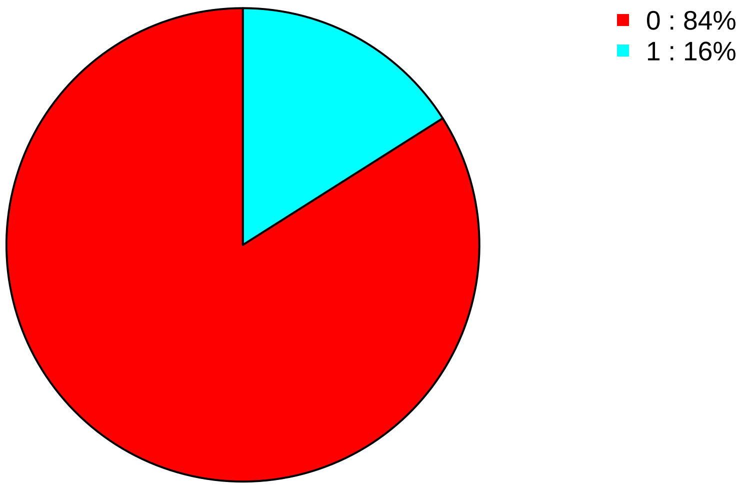
```
## [1] " "
```

Distribution of Card_Category



```
## [1] " "
```

Distribution of Closed_Account

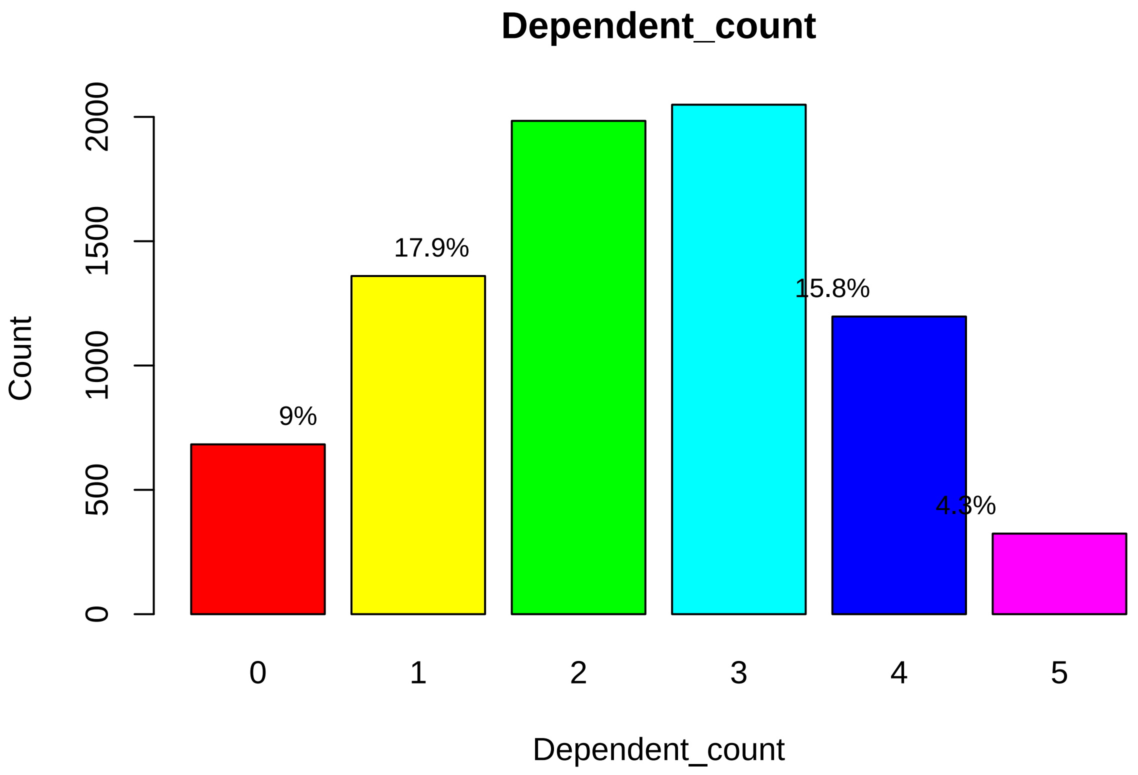


The ‘Distribution of Gender’ chart shows a nearly even split between male and female, with females slightly more prevalent. The ‘Distribution of Education_Level’ indicates a diverse educational background among the individuals, with the majority holding a graduate education. The ‘Distribution of Marital_Status’ reveals that a significant proportion of individuals are married. The ‘Distribution of Card_Category’ shows that most individuals have a Blue card. Lastly, the ‘Distribution of Closed_Account’ indicates that the majority of accounts are active.

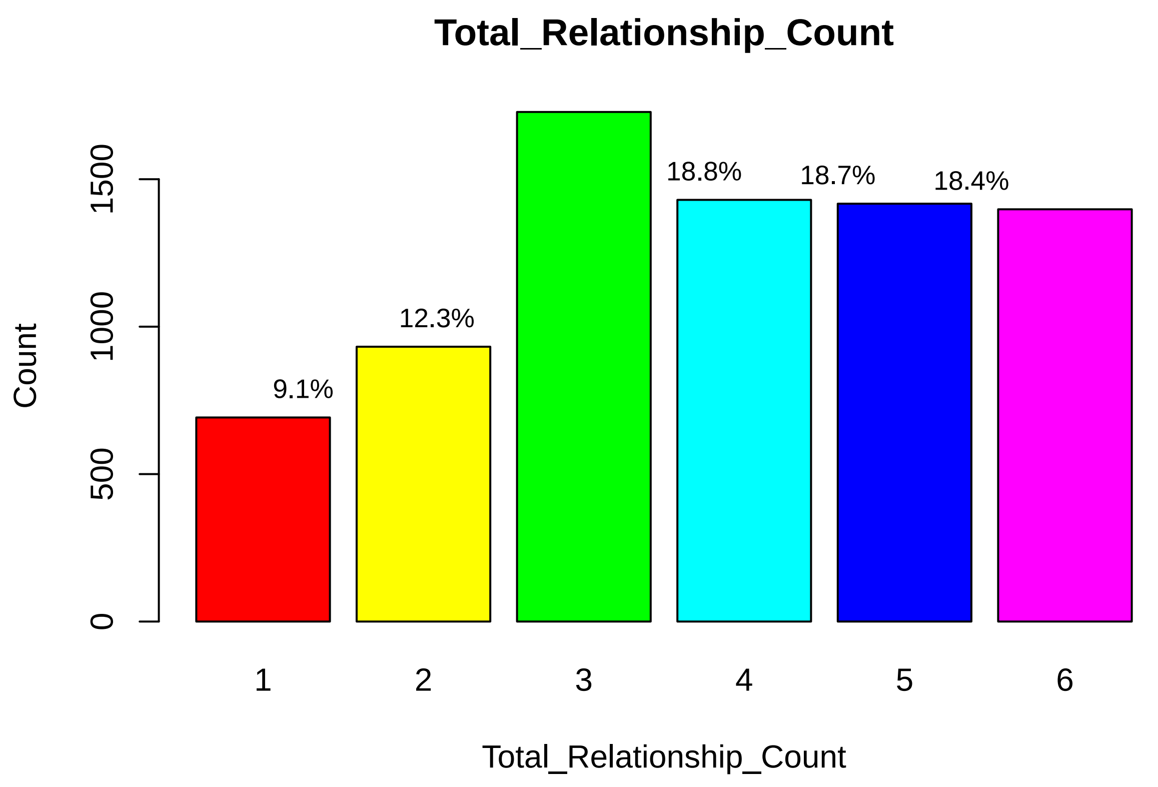
Now, let’s visualise columns with high modalities with a Bar Plot.

```
barvars <- c("Dependent_count", "Total_Relationship_Count",
             "Months_Inactive_12_mon", "Contacts_Count_12_mon")
for (var in barvars) {
  print(" ")
  var_table <- table(Data[[var]])
  percentages <- round((var_table / sum(var_table)) * 100, 1)

  barplot(var_table,
          col = rainbow(length(var_table)), # Use rainbow colors for differentiation
          main = var, # Use variable name as the main title
          ylim = c(0, max(var_table) + 5), # Adjust y limit for space for labels
          xlab = var, # Label x-axis with variable name
          ylab = "Count") # Label y-axis as "Count"

  text(x = seq_along(var_table), y = var_table + 1, labels = paste(percentages, "%", sep=""), pos = 3, 

}
```

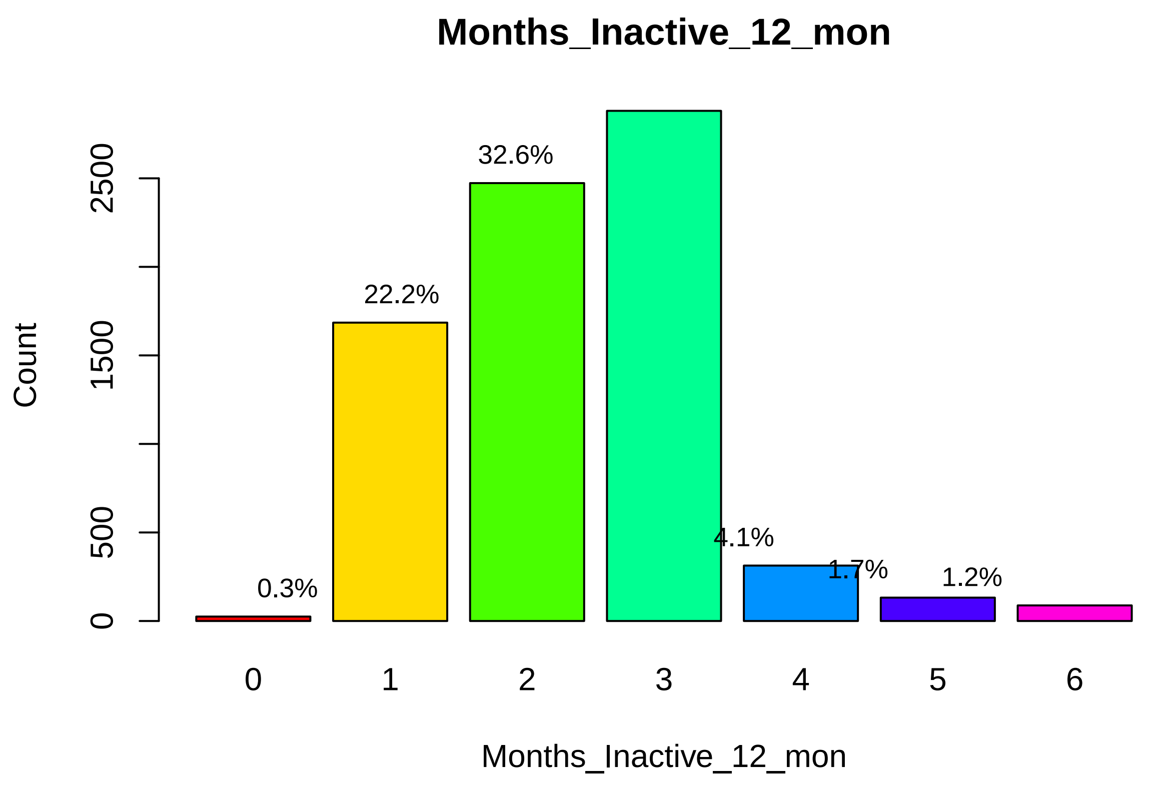
```
## [1] " "
```



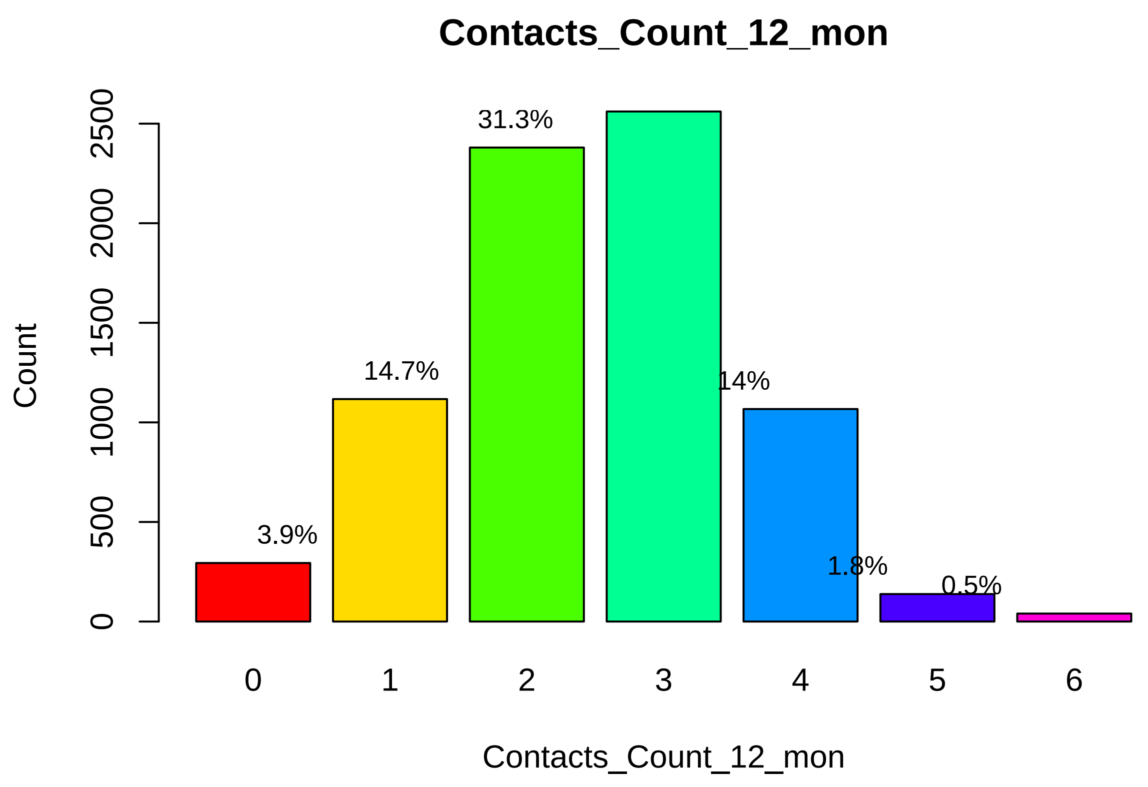
```
## [1] " "
```


Total_Relationship_Count
Count
0
500
1000
1500
9.1%
12.3%
18.8%
18.7%
18.4%
1
2
3
4
5
6
Total_Relationship_Count

```
## [1] " "
```


Months_Inactive_12_mon
Count
0
500
1500
2500
0.3%
22.2%
32.6%
4.1%
1.7%
1.2%
0
1
2
3
4
5
6
Months_Inactive_12_mon

```
## [1] " "
```



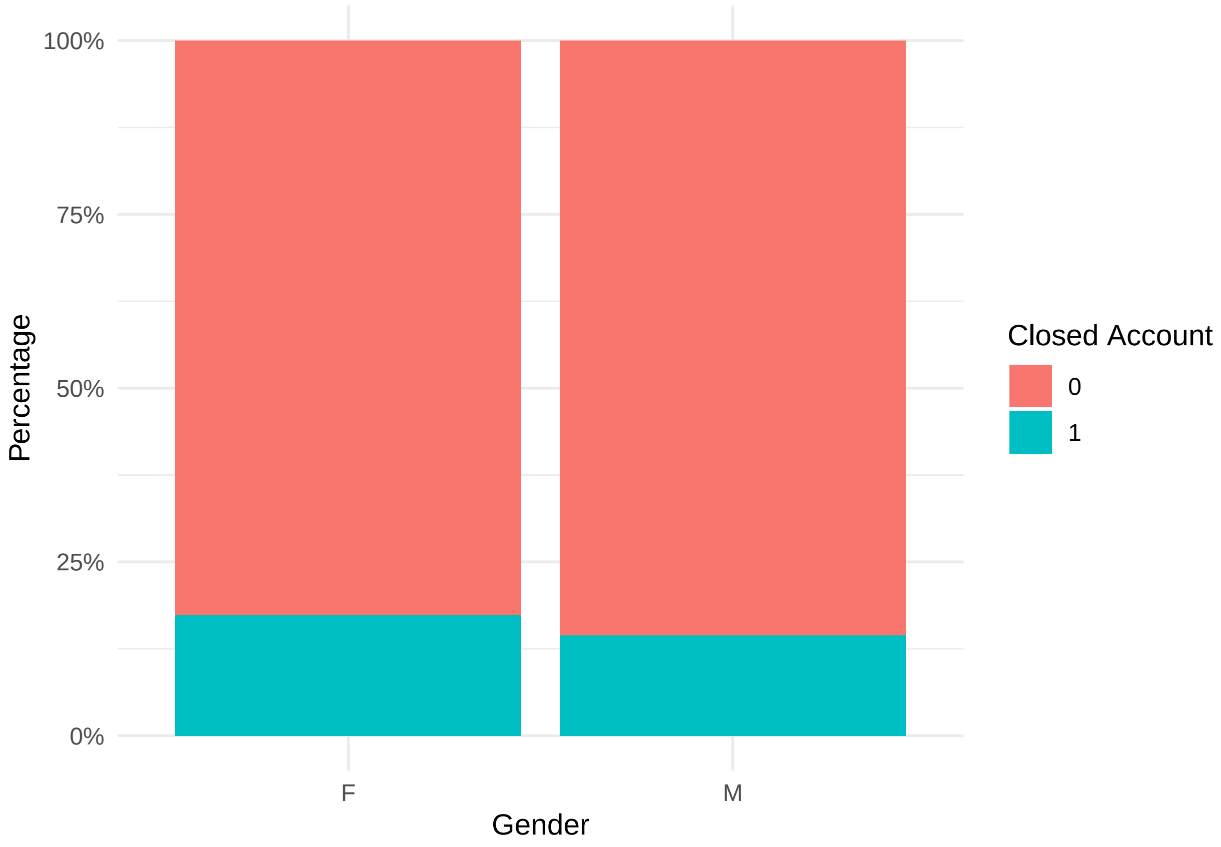
These visualisation are great to give us a general overlook of our data. We can make some interesting observations:

Dependent_count: The chart indicates that a considerable number of customers have two or three dependents. Total_Relationship_Count: Customers are spread fairly evenly across different relationship levels with the bank. Months_Inactive_12_mon: There is a significant concentration of customers who have been inactive for three months within the last year. Contacts_Count_12_mon: The distribution suggests that contacting customers two or three times in the past 12 months is most common

Bivariate Analysis

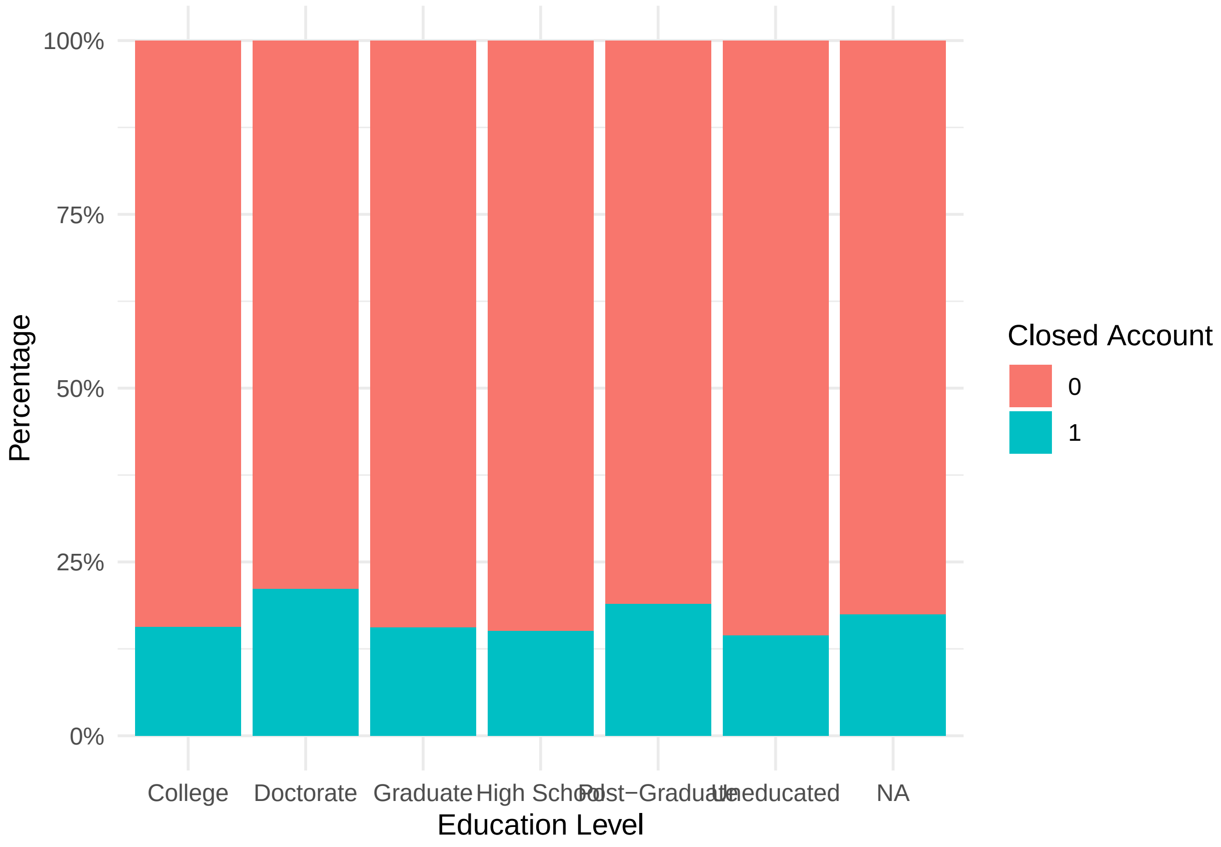
Now, let's move on to looking at relevant relationships between variables. Let's look at a few relevant categorical relationships with a closed account. Closed account remains to be the centre of our focus since we keep our purpose (defined in the introduction) in mind.

```r
library(ggplot2)
ggplot(Data, aes(x = Gender, fill = factor(Closed_Account))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = "Gender", y = "Percentage", fill = "Closed Account") +
  theme_minimal()
```



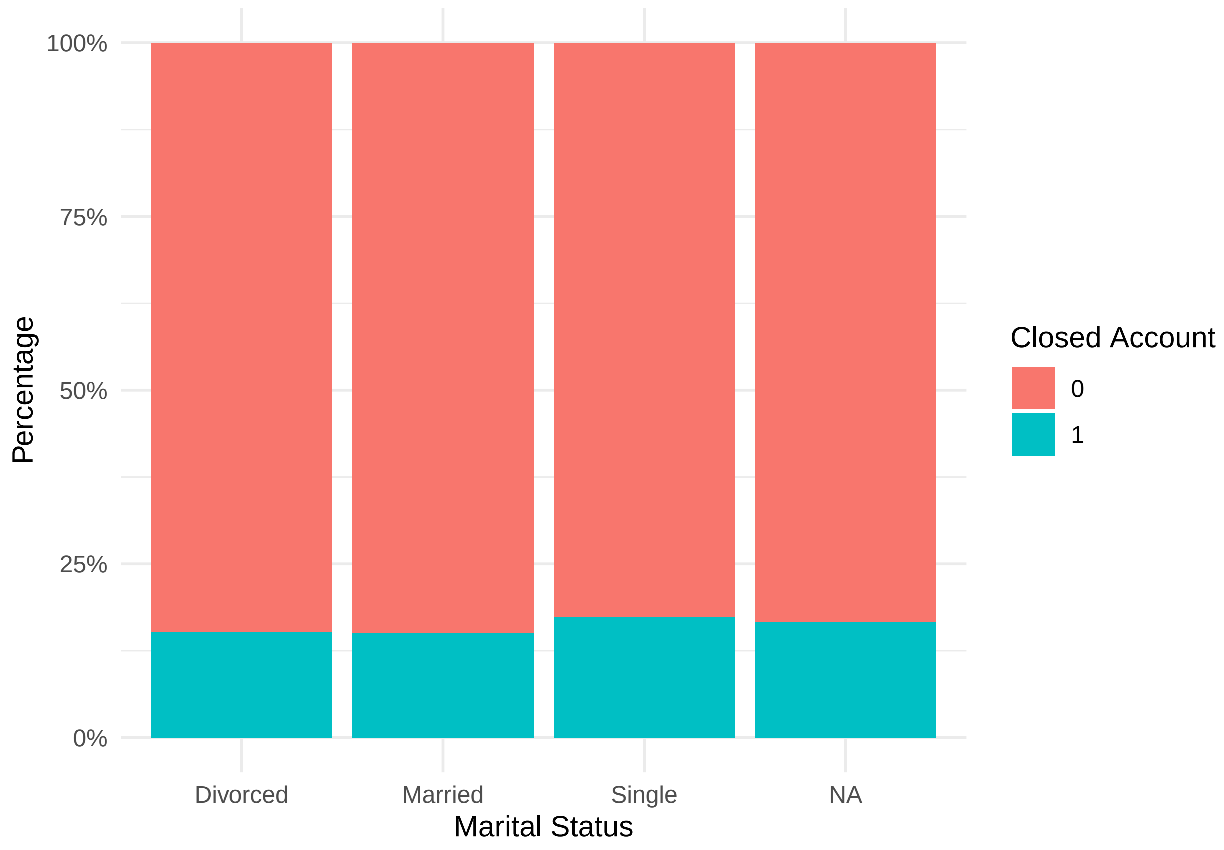
The chart shows a similar small proportion of closed accounts for both genders, with the vast majority of accounts remaining active for both females and males.

```r
ggplot(Data, aes(x = Education_Level, fill = factor(Closed_Account))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = "Education Level", y = "Percentage", fill = "Closed Account") +
  theme_minimal()
```



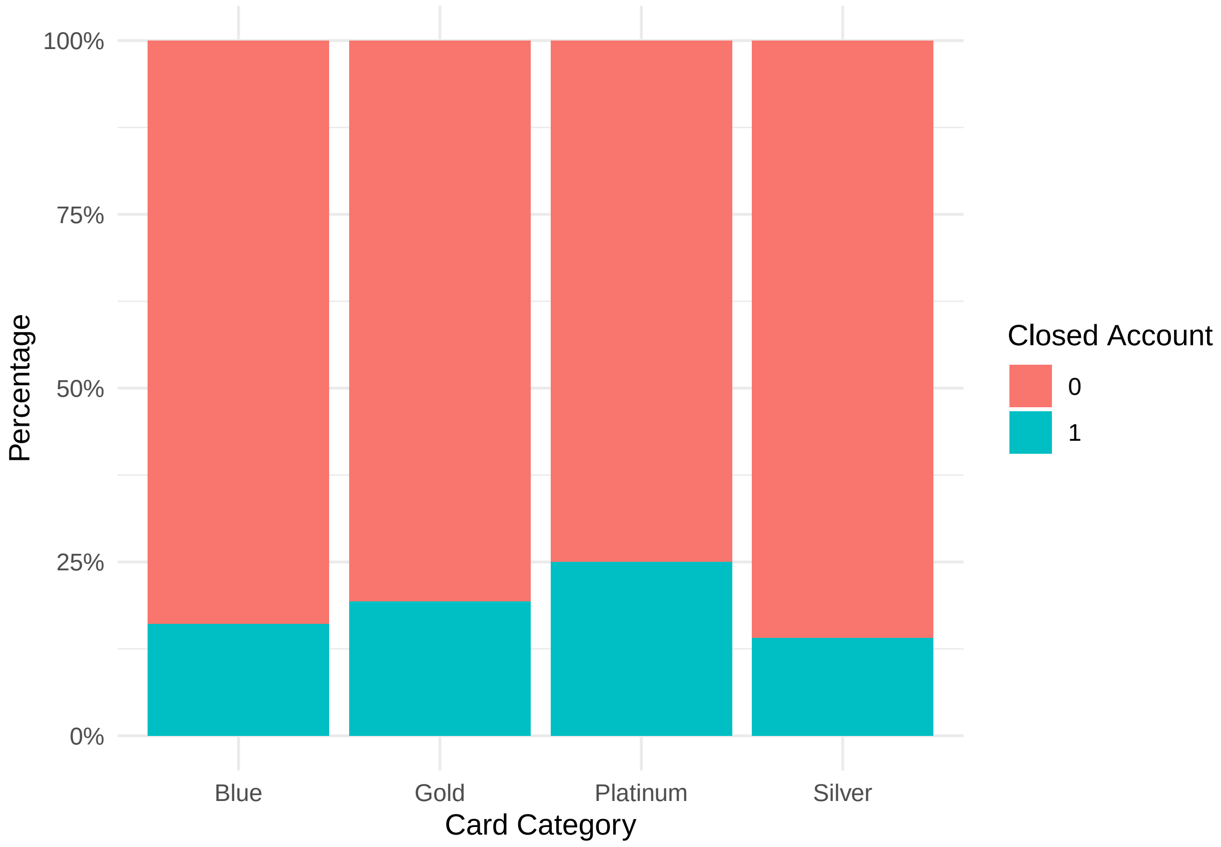
Across different education levels, the majority of accounts are active, with closure rates being comparatively low.

```r
ggplot(Data, aes(x = Marital_Status, fill = factor(Closed_Account))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = "Marital Status", y = "Percentage", fill = "Closed Account") +
  theme_minimal()
```



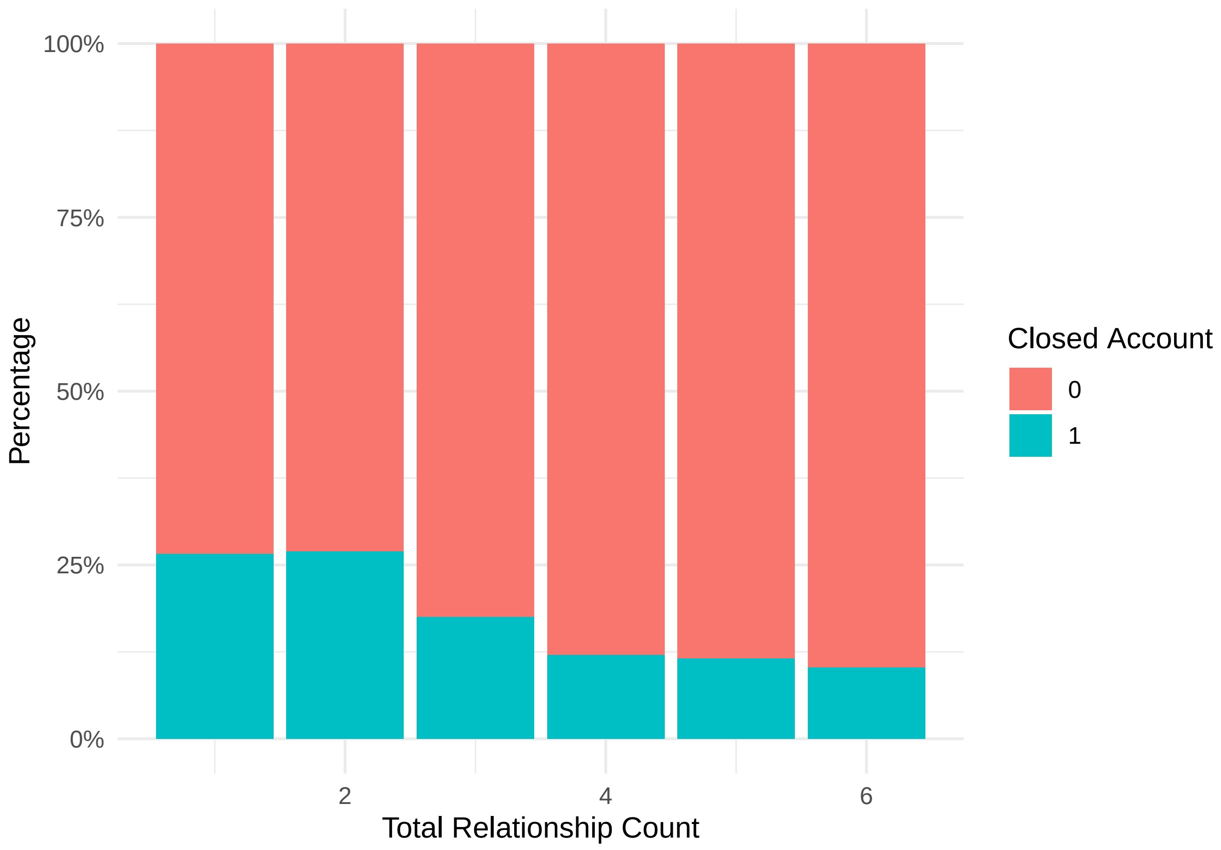
Active accounts predominate across all marital statuses, with closed accounts making up a smaller portion.

```
ggplot(Data, aes(x = Card_Category, fill = factor(Closed_Account))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = "Card Category", y = "Percentage", fill = "Closed Account") +
  theme_minimal()
```



The blue card category shows the most activity with very few closures, consistent across other card types.

```r
ggplot(Data, aes(x = Total_Relationship_Count, fill = factor(Closed_Account))) +
  geom_bar(position = "fill") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = "Total Relationship Count", y = "Percentage", fill = "Closed Account") +
  theme_minimal()
```



Most customers maintain active accounts across all levels of product relationships with the bank.

Let's perform some hypothesis testing to rigidify our interpretations.

```
# Ensure the Data$Closed_Account is a factor for accurate Chi-square testing
Data$Closed_Account <- factor(Data$Closed_Account)

# Gender vs. Closed_Account
# H0: There is no association between Gender and Closed_Account status.
# Ha: There is an association between Gender and Closed_Account status.
gender_table <- table(Data$Gender, Data$Closed_Account)
chisq.test(gender_table)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  gender_table
## X-squared = 12.231, df = 1, p-value = 0.0004699
```

With a p-value of 0.0004699, there is strong evidence against the null hypothesis. This suggests there is an association between gender and whether an account is closed.

```
# Education_Level vs. Closed_Account
# H0: There is no association between Education_Level and Closed_Account status.
# Ha: There is an association between Education_Level and Closed_Account status.
education_level_table <- table(Data$Education_Level, Data$Closed_Account)
chisq.test(education_level_table)
```

```
##
```

```
##  Pearson's Chi-squared test
##
## data:  education_level_table
## X-squared = 12.339, df = 5, p-value = 0.03043
```

The p-value of 0.03043 indicates that there is a statistically significant association between education level and closed account status, although the evidence is not as strong as for gender.

```r
# Marital_Status vs. Closed_Account
# HO: There is no association between Marital_Status and Closed_Account status.
# Ha: There is an association between Marital_Status and Closed_Account status.
marital_status_table <- table(Data$Marital_Status, Data$Closed_Account)
chisq.test(marital_status_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  marital_status_table
## X-squared = 6.6217, df = 2, p-value = 0.03648
```

The p-value of 0.03648 suggests a statistically significant association between marital status and closed account status, but this significance is marginal and could be sensitive to sample size and distribution.

```r
# Card_Category vs. Closed_Account
# HO: There is no association between Card_Category and Closed_Account status.
# Ha: There is an association between Card_Category and Closed_Account status.
card_category_table <- table(Data$Card_Category, Data$Closed_Account)
chisq.test(card_category_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  card_category_table
## X-squared = 2.8848, df = 3, p-value = 0.4097
```

With a p-value of 0.4097, there is insufficient evidence to suggest an association between card category and closed account status, and we fail to reject the null hypothesis.

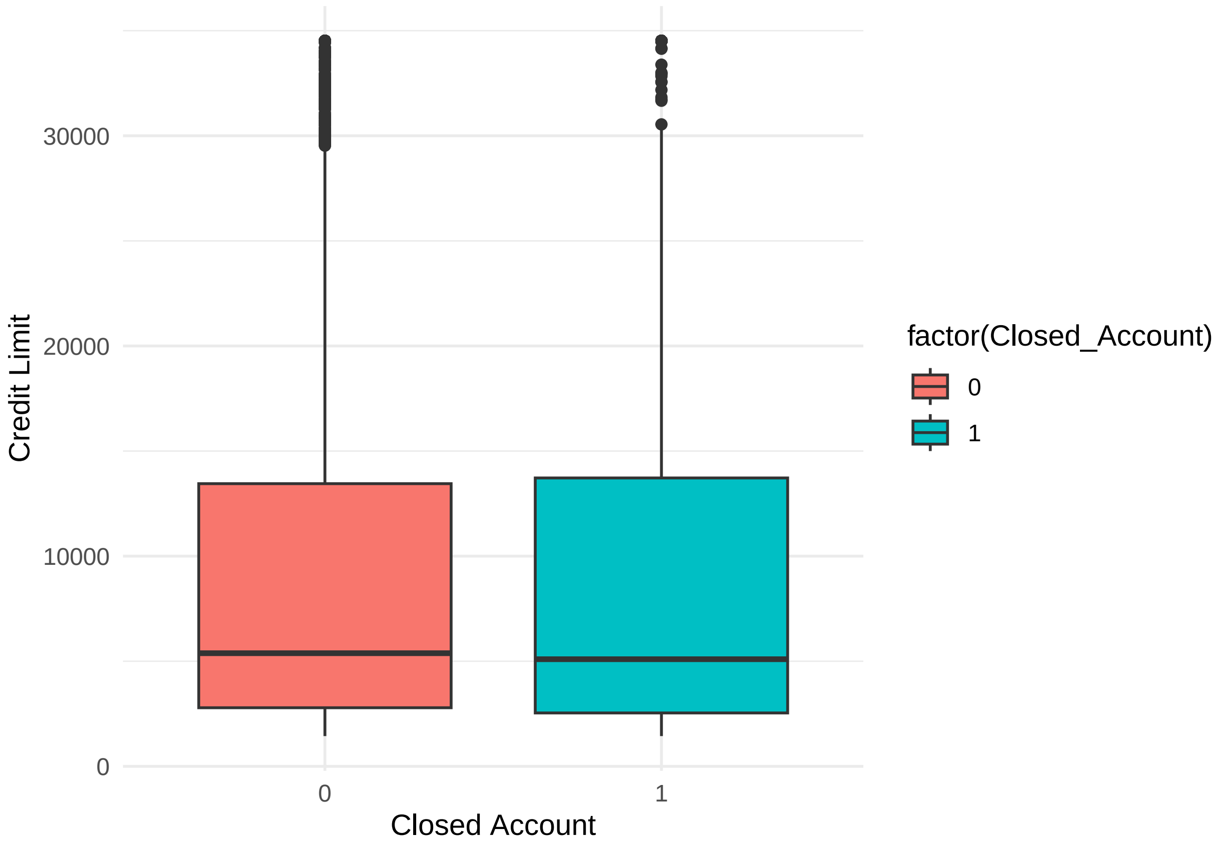
```r
# Total_Relationship_Count vs. Closed_Account
# HO: There is no association between Total_Relationship_Count and Closed_Account status.
# Ha: There is an association between Total_Relationship_Count and Closed_Account status.
total_relationship_count_table <- table(Data$Total_Relationship_Count, Data$Closed_Account)
chisq.test(total_relationship_count_table)
```

```
##
##  Pearson's Chi-squared test
##
## data:  total_relationship_count_table
## X-squared = 215.5, df = 5, p-value < 2.2e-16
```

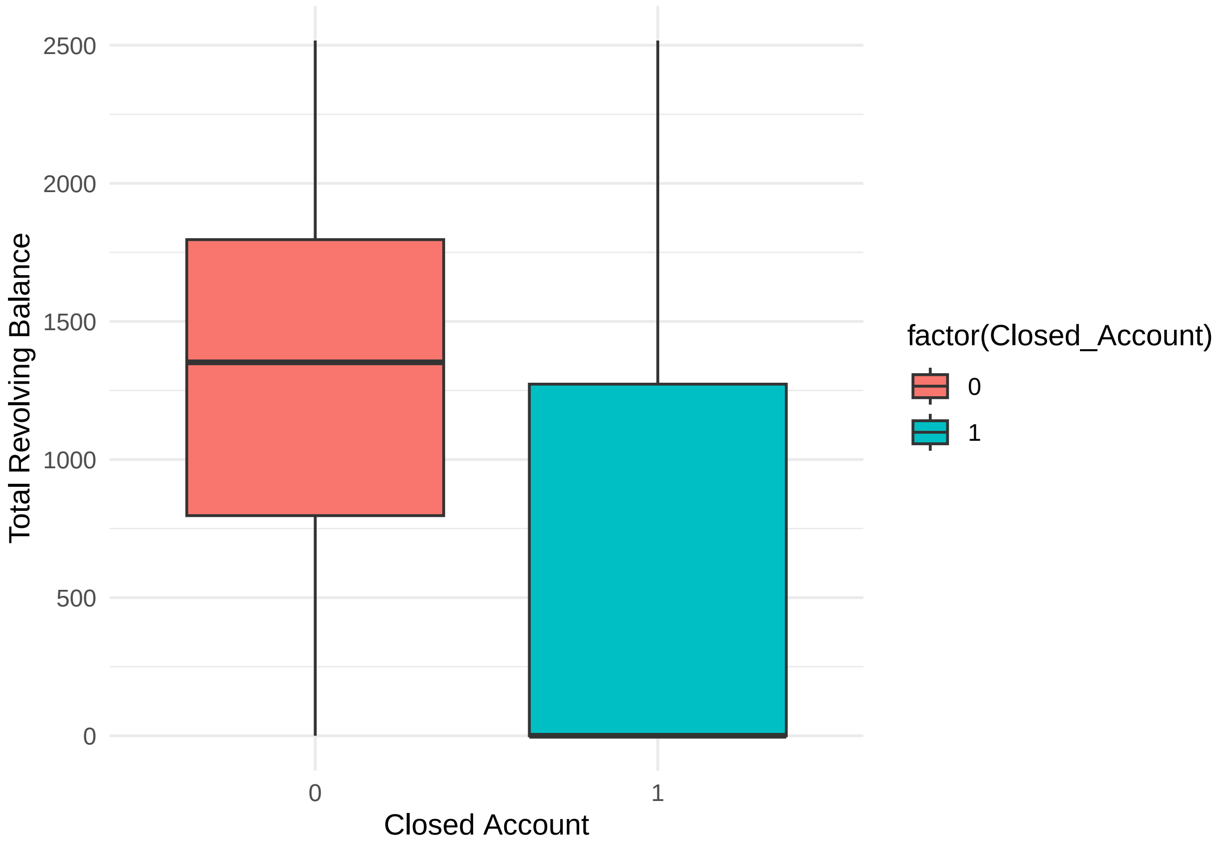
The very small p-value (< 2.2e-16) indicates a very strong association between the total relationship count and closed account status, strongly rejecting the null hypothesis.

Now, let's do a boxplot to understand the relationship between Closed__Account and the relevant continuous variables.

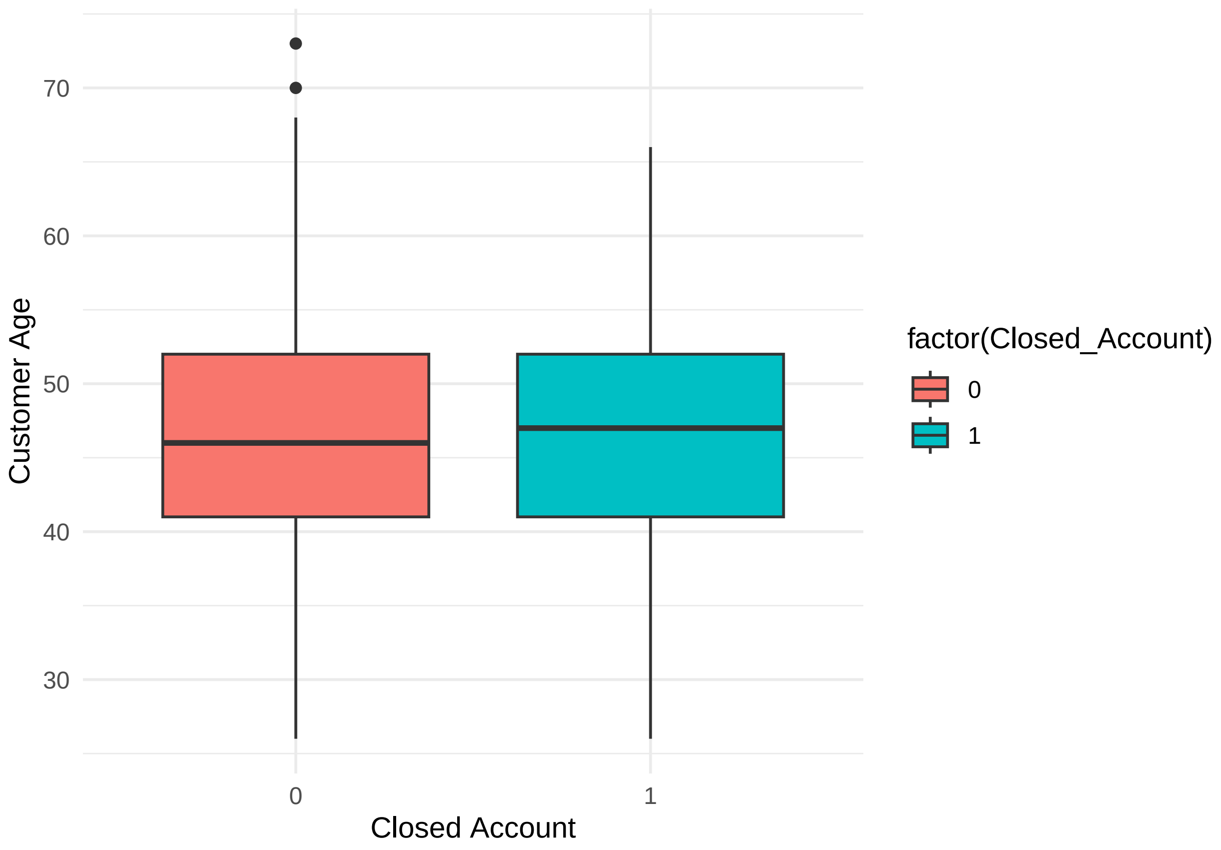
```r
ggplot(Data, aes(x=factor(Closed_Account), y=Credit_Limit, fill=factor(Closed_Account))) +
  geom_boxplot() +
  labs(x="Closed Account", y="Credit Limit") +
  theme_minimal()
```



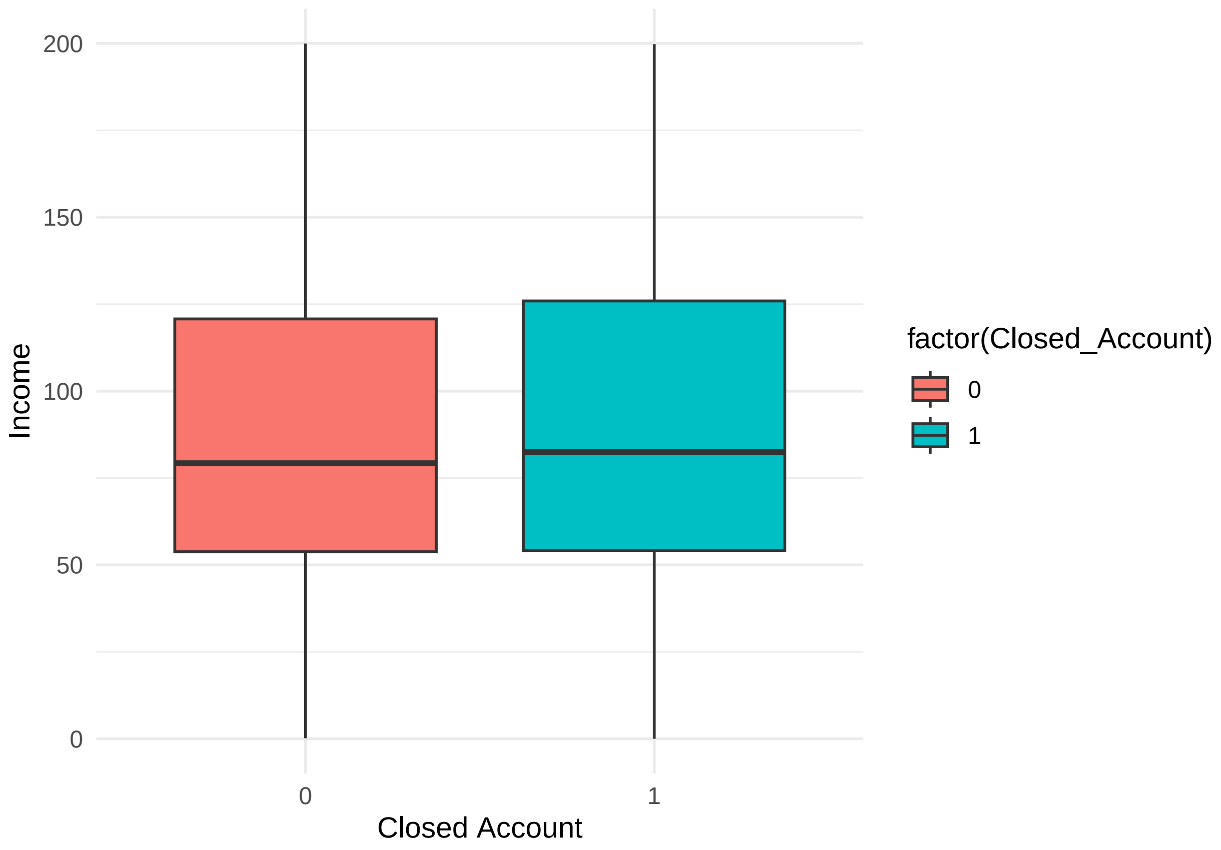
```
ggplot(Data, aes(x=factor(Closed_Account), y=Total_Revolving_Bal, fill=factor(Closed_Account))) +
  geom_boxplot() +
  labs(x="Closed Account", y="Total Revolving Balance") +
  theme_minimal()
```



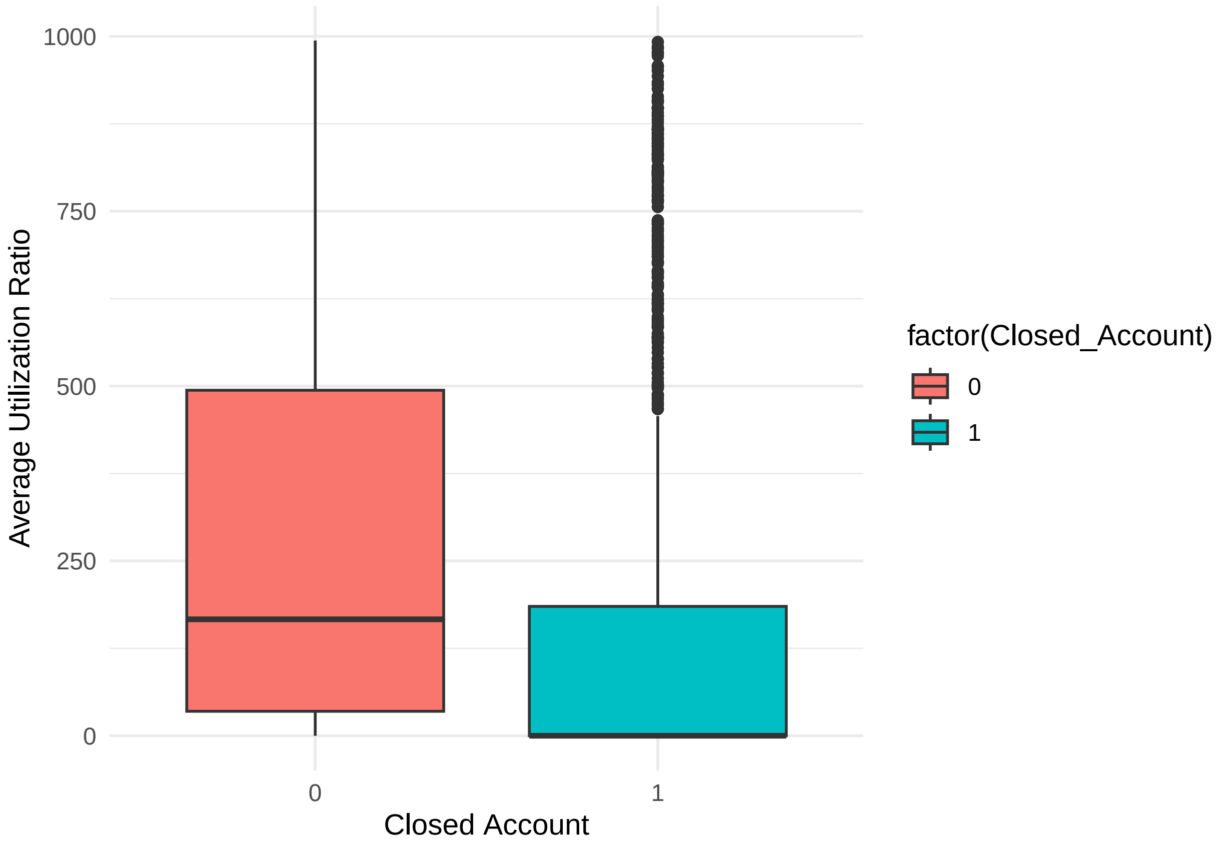
```
ggplot(Data, aes(x=factor(Closed_Account), y=Customer_Age, fill=factor(Closed_Account))) +
  geom_boxplot() +
  labs(x="Closed Account", y="Customer Age") +
  theme_minimal()
```



```
ggplot(Data, aes(x=factor(Closed_Account), y=Income, fill=factor(Closed_Account))) +
  geom_boxplot() +
  labs(x="Closed Account", y="Income") +
  theme_minimal()
```



```r
ggplot(Data, aes(x=factor(Closed_Account), y=Avg_Utilization_Ratio, fill=factor(Closed_Account))) +
  geom_boxplot() +
  labs(x="Closed Account", y="Average Utilization Ratio") +
  theme_minimal()
```



The boxplots reveal that active accounts generally have higher credit limits and revolving balances than closed accounts, with a broader range in both metrics. Customer age and income levels are comparable for both active and closed accounts, suggesting that account status does not vary significantly with these demographics. However, active accounts exhibit a higher median credit utilization ratio, with a wider spread indicating greater variability in how much credit is used relative to the limit, compared to closed accounts which show a more condensed utilization pattern.

Now, let's perform t-test to determine further the impact of these variables on the Closed_Account variable.

```
t_test_credit_limit <- t.test(Data$Credit_Limit ~ Data$Closed_Account)
print(t_test_credit_limit)
```

```
##
##  Welch Two Sample t-test
##
## data:  Data$Credit_Limit by Data$Closed_Account
## t = 1.121, df = 1727.1, p-value = 0.2625
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -234.7302  860.9402
## sample estimates:
## mean in group 0 mean in group 1
##        9303.600        8990.495
```

With a p-value of 0.2625, there is not enough evidence to conclude that there is a significant difference in credit limits between active and closed accounts. The means of both groups are similar, and the confidence interval includes zero, further suggesting no significant difference.

```r
t_test_total_revolving_balance <- t.test(Data$Total_Revolving_Bal ~ Data$Closed_Account)
print(t_test_total_revolving_balance)
```

```
##
##  Welch Two Sample t-test
##
## data:  Data$Total_Revolving_Bal by Data$Closed_Account
## t = 20.969, df = 1545.5, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  531.8715 641.6442
## sample estimates:
## mean in group 0 mean in group 1
##       1248.7842        662.0263
```

The p-value is less than 2.2e-16, which is extremely small, indicating a highly significant difference in the total revolving balances between active and closed accounts. The average revolving balance is significantly higher for active accounts than for closed ones.

```r
t_test_income <- t.test(Data$Income ~ Data$Closed_Account)
print(t_test_income)
```

```
##
##  Welch Two Sample t-test
##
## data:  Data$Income by Data$Closed_Account
## t = -0.62389, df = 1689, p-value = 0.5328
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -4.010509  2.074841
## sample estimates:
## mean in group 0 mean in group 1
##        90.84967        91.81750
```

The p-value of 0.5328 suggests that there is no significant difference in income between active and closed accounts. The means are nearly equal, and the confidence interval includes zero.

```r
t_test_avg_utilization_ratio <- t.test(Data$Avg_Utilization_Ratio ~ Data$Closed_Account)
print(t_test_avg_utilization_ratio)
```

```
##
##  Welch Two Sample t-test
##
## data:  Data$Avg_Utilization_Ratio by Data$Closed_Account
## t = 15.046, df = 1774.8, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  105.3873 136.9820
## sample estimates:
## mean in group 0 mean in group 1
##        268.4418        147.2572
```

A p-value less than 2.2e-16 signifies a highly significant difference in average utilization ratios between the two groups. Active accounts have a substantially higher average utilization ratio compared to closed accounts.

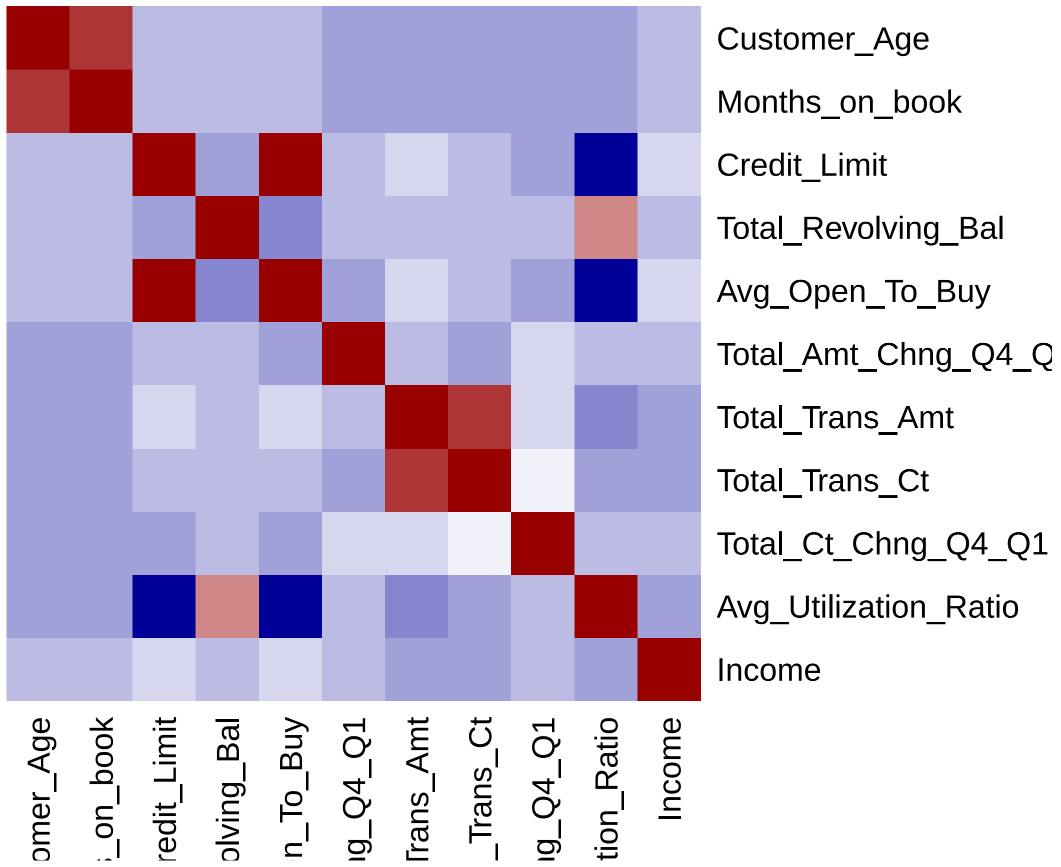
```r
t_test_customer_age <- t.test(Data$Customer_Age ~ Data$Closed_Account)
print(t_test_customer_age)
```

```
##
##  Welch Two Sample t-test
##
## data:  Data$Customer_Age by Data$Closed_Account
## t = -2.252, df = 1758, p-value = 0.02444
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -1.02951386 -0.07103968
## sample estimates:
## mean in group 0 mean in group 1
##        46.17774        46.72802
```

With a p-value of 0.02444, there is a statistically significant difference in age between active and closed accounts, although the actual difference in mean ages is quite small.

Finally, lets look into the relationship between certain continuous variables to further gain insight into some interesting patterns that we might find interesting.

```
Data_num <- Data[,names(Data) %in% contcols]
Cor.Matrix = cor(Data_num)

col<- colorRampPalette(c(rgb(0, 0,.6), "white", rgb(.6,0,0)))(20)
heatmap(x = Cor.Matrix, col = col, symm = T,Rowv = NA)
```



```
correlated_var <- matrix(nrow=1, ncol=3)
for(i in 2:ncol(Data_num)){
  if (is.numeric(Data_num[1,i])) {
    for (j in i:ncol(Data_num)) {
      if (i != j & (Cor.Matrix[i,j] > 0.70 | Cor.Matrix[i,j] < -0.70)){
        correlated_var <- rbind(correlated_var, c(colnames(Data_num)[i],colnames(Data_num)[j],Cor.Matri
      }
    }
  }
}
```

Customer_Age and Months_on_book might have a strong positive correlation, which is logical since older customers are likely to have been with the bank for longer periods.Credit_Limit seems to have a strong positive correlation with Avg_Open_To_Buy, which makes sense as the open to buy is typically a function of the credit limit minus the current balance.Total_Trans_Amt and Total_Trans_Ct show a strong positive correlation, suggesting that customers with more transactions tend to also have higher total transaction amounts.Avg_Utilization_Ratio seems to have a strong negative correlation with Avg_Open_To_Buy. This would align with the idea that higher utilization ratios (proportion of credit limit used) often mean less credit is available to use. Between credit_limit and Avg_Utilization_Ratio there is a negative relationship. Customers with higher credit limits tend to use a smaller proportion of their available credit.

Logistic Regression (Q3)

```
X1 <- Data$Gender
X2 <- Data$Income
Y <- as.numeric(Data$Closed_Account) - 1

mod = glm(Y~X1+X2, family = binomial(link = "logit"))
summary(mod)
```

```
##
## Call:
## glm(formula = Y ~ X1 + X2, family = binomial(link = "logit"))
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5952639  0.0709750 -22.476  < 2e-16 ***
## X1M         -0.2232886  0.0632205  -3.532 0.000413 ***
## X2           0.0004261  0.0006311   0.675 0.499513
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6685.3  on 7596  degrees of freedom
## Residual deviance: 6672.3  on 7594  degrees of freedom
## AIC: 6678.3
##
## Number of Fisher Scoring iterations: 4
```

According to the output we received, we notice that gender is significant while income isn't. We can further analyse output by taking the exponentials and questioning their odds.

```
exp(coefficients(mod)[1])#odds of closing account for F with avg income
```

```
## (Intercept)
##    0.202855
```

```r
exp((coefficients(mod)[1]) + coefficients(mod)[2]) #odds of closing account for M with avg income
```

```
## (Intercept)
##   0.1622605
```

```r
coefficients(mod)[3] #odds ratio of closing when income increases by 1 std deviation
```

```
##           X2
## 0.0004261444
```

```r
coefficients(mod)[3]/4 #maximal prob increase rate when income goes up
```

```
##           X2
## 0.0001065361
```

By looking at the exponentials, we see that the odds of closing, based on an income variation, are fairly low. We can add an interaction term to better interpret the effect of income.

```r
# Null Hypothesis (H0): The effect of income on the probability of account closure is the same for male

# Alternative Hypothesis (H1): The effect of income on the probability of account closure differs betwe

mod2 = glm(Y~X1*X2, family = binomial(link = "logit"))
summary(mod2)
```

```
##
## Call:
## glm(formula = Y ~ X1 * X2, family = binomial(link = "logit"))
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5474767  0.0787725 -19.645  < 2e-16 ***
## X1M         -0.3990676  0.1445507  -2.761  0.00577 **
## X2          -0.0001008  0.0007426  -0.136  0.89207
## X1M:X2       0.0019037  0.0014016   1.358  0.17440
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6685.3  on 7596  degrees of freedom
## Residual deviance: 6670.5  on 7593  degrees of freedom
## AIC: 6678.5
##
## Number of Fisher Scoring iterations: 4
```

We notice that the Interaction term isn't statistically significant. Let's further interpret the coefficients.

```r
exp(coefficients(mod2)[3])#odds of closing when F and income varies by sd
```

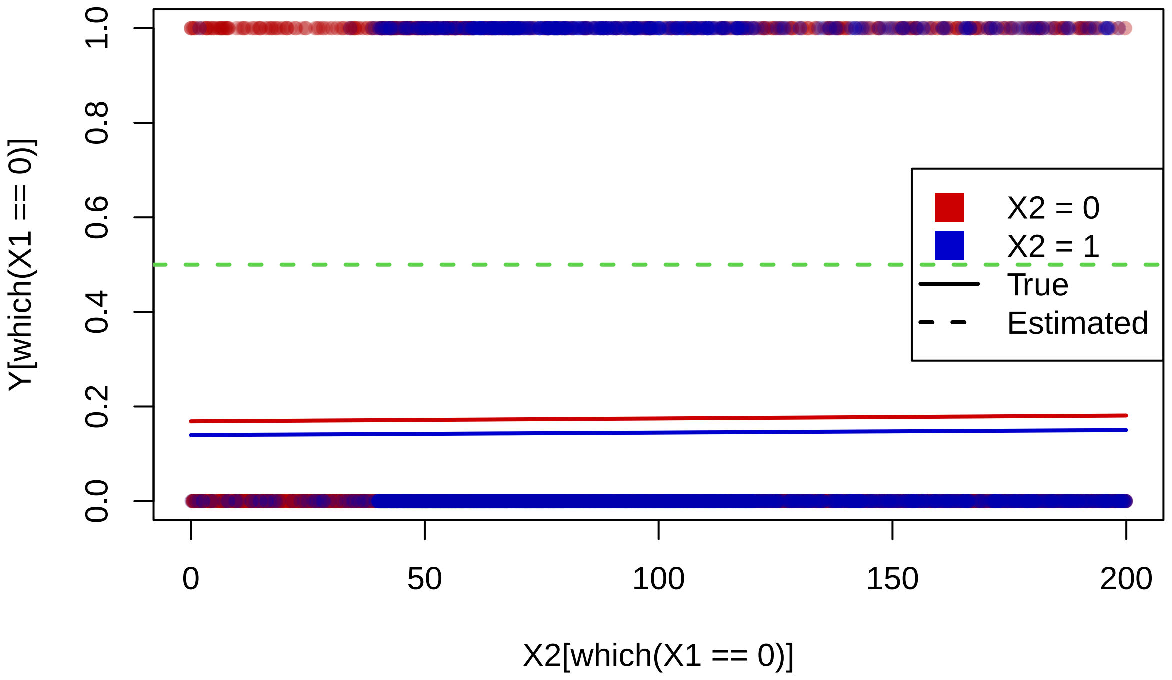
```
##        X2
## 0.9998992
```

```r
exp(coefficients(mod2)[3] + coefficients(mod2)[4]) #odds of closing when M and income varies by sd
```

```
##       X2
## 1.001805
```

We get numbers *very close to 1*. This mean that there is Income does have an effect, but since the *interaction term isn't significant*, we don't reject the null hypothesis. We now plot without the interaction term.

```
X1 <- as.numeric(X1) -1
plot(X2[which(X1 == 0)], Y[which(X1 == 0)], col = rgb(.7,0,0,.2), pch = 16)
points(X2[which(X1 == 1)], Y[which(X1 == 1)], col = rgb(0,0,0.7,.2), pch = 16)

f.true0 = function(x) exp(coefficients(mod)[1] + coefficients(mod)[3]*x)/ (1 + exp(coefficients(mod)[1]
f.true1 = function(x) exp(coefficients(mod)[1] + coefficients(mod)[3]*x + coefficients(mod)[2])/ (1 + e

curve(f.true0, add=T, lwd = 2, col = rgb (.8,0,0))
curve(f.true1, add=T, lwd = 2, col = rgb (0,0,0.8))

abline(h = 0.5, col = 3, lwd = 2, lty = 2)

legend("right", c("X2 = 0", "X2 = 1", "True", "Estimated"),
       col = c(rgb (0.8,0,0),rgb (0,0,0.8), 1, 1), pch = c(15,15,NA,NA), pt.cex = c(2,2,NA,NA),
       lty = c(NA,NA, 1,2), lwd = c(NA,NA, 2,2))
```



We notice a very flat estimation. We also notice low probabilities leading to t never predicting a 1. This might be caused by the insignificance of 'Income' variable.

The Confusion Matrix is as follows.

```
predicted.Y = (predict(mod, type = "response")>.5)*1

True.positive = sum(predicted.Y[which(Y == 1)] == 1)
True.negative = sum(predicted.Y[which(Y == 0)] == 0)
False.positive = sum(predicted.Y[which(Y == 0)] == 1)
False.negative = sum(predicted.Y[which(Y == 1)] == 0)

Confusion.Matrix = matrix(c(True.positive,
                            False.positive,
                            False.negative,
                            True.negative),
                          nrow = 2,ncol = 2)

row.names(Confusion.Matrix) = c("Actual Positive", "Actual Negative")
colnames(Confusion.Matrix) = c("Predicted Positive", "Predicted Negative")

Confusion.Matrix
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                  0               1217
## Actual Negative                  0               6380
```

```
(Sensitivity = True.positive/(True.positive + False.negative))
```

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

```
(Specificity = True.negative/(True.negative + False.positive))
```

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

```
(Accuracy = (True.positive + True.negative) / nrow(Data))
```

```
## [1] 0.8398052
```

All predictions were categorized as negative (Predicted Negative). This suggests that the model predicted everyone as not having a closed account. As for the sensitivity, it didn't detect any of the accounts that were actually closed. The specificity of the model is 1. This means that the model correctly identified all actual negative cases. The accuracy of the model is approximately 83.98%. While this may seem high, it is misleading in this context because the model predicted negative for all cases.

k-NN (Q4)

Next, let's move on to performing k-NN. We first split the training data (and the indices) into a training set and a validation set.

```
set.seed(404)
indices <- sample(nrow(Data))

trainIndices <- indices[1:round(0.75 * nrow(Data))]
validationIndices <- indices[(round(0.75 * nrow(Data)) + 1):nrow(Data)]

d_train <- Data[trainIndices, ]
d_val <- Data[validationIndices, ]

x_train <- d_train[c('Total_Trans_Amt', 'Total_Trans_Ct')]
y_train <- d_train$Closed_Account
x_val <- d_val[c('Total_Trans_Amt', 'Total_Trans_Ct')]
y_val <- d_val$Closed_Account



# Grid of values for k
k_grid <- 1:80

train_m <- apply(x_train,
                 MARGIN = 2,
                 FUN = mean)
```

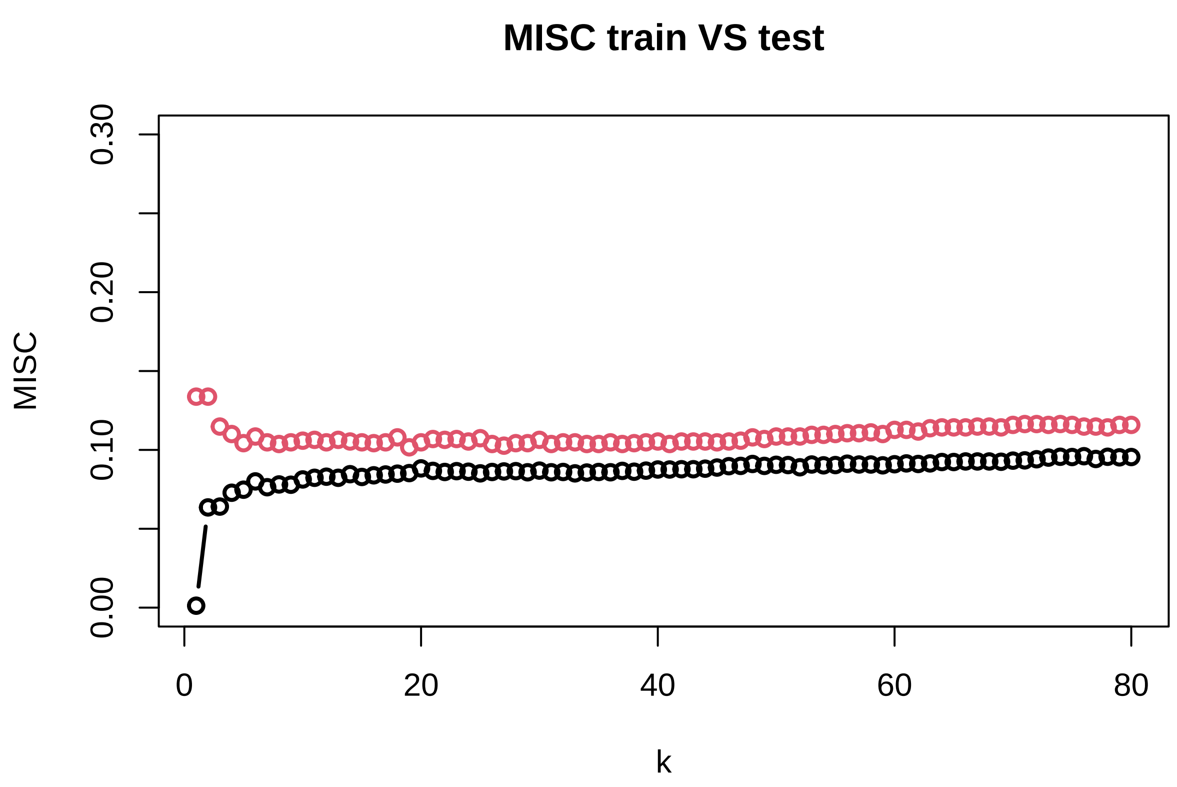
```r
train_s <- apply(x_train,
                 MARGIN = 2,
                 FUN = sd)
x_train <- scale(x_train, center = TRUE, scale = TRUE)

#scale validation using training metrics
x_val_sc <- scale(x_val,
                  center = train_m,
                  scale = train_s)
```

Our first step is to calculate *the best number of neighbours k*. We do this by setting up the k-nearest neighbors (KNN) algorithm to a range of 'k' values to find the one that performs the best in terms of misclassification rate and AUC.

```r
library(class)
suppressWarnings(suppressMessages({
      k_results <- sapply(k_grid,
                          FUN = function (kk) {

                            # Model fitting for the training data. It classifies each point in the trai
                            # using the k-nearest points in the training set itself.
                            kk_fit_train <- knn(train = x_train,
                                                test = x_train,
                                                cl = as.factor(y_train), # Convert the response variabl
                                                k = kk, # The current value of 'k' in the grid.
                                                prob = TRUE) # Request probability estimates for the wi

                            # Model fitting for the validation data. It classifies each point in the va
                            # using the k-nearest points in the training set.
                            kk_fit_val <- knn(train = x_train,
                                              test = x_val_sc,
                                              cl = as.factor(y_train),
                                              k = kk,
                                              prob = TRUE)

                            # Calculate the misclassification error on the training set:
                            # the proportion of incorrect predictions.
                            kk_misc_in <- 1 - mean(y_train == kk_fit_train)

                            # Calculate the misclassification error on the validation set.
                            kk_misc_out <- 1 - mean(y_val == kk_fit_val)

                            # Calculate the AUC for the training set.
                            # The ifelse() function adjusts the probability based on the predicted clas
                            # it represents the probability of the positive class "1".
                            kk_probyes_in <- ifelse(kk_fit_train == "1", attr(kk_fit_train, "prob"), 1 

                            # Calculate the AUC for the validation set.
                            kk_probyes_out <- ifelse(kk_fit_val == "1", attr(kk_fit_val, "prob"), 1 - a

                            # Use pROC library to compute AUC on the training set.
                            kk_auc_in <- pROC::auc(y_train == 1, kk_probyes_in)

                            # Use pROC library to compute AUC on the validation set.
```

```
                              kk_auc_out <- pROC::auc(y_val == 1, kk_probyes_out)

                              # Return a vector containing the misclassification error and AUC for both t
                              # and validation sets for the current value of 'k'.
                              return(c(kk_misc_in,
                                       kk_misc_out,
                                       kk_auc_in,
                                       kk_auc_out))
                            })
     }))
```

```
# Evaluation metrics on the grid
misc_in <- k_results[1, ]
misc_out <- k_results[2, ]
auc_in <- k_results[3, ]
auc_out <- k_results[4, ]

# Plot the performance (MISC)
plot(k_grid,
     misc_in,
     type = "b",
     lwd = 2,
     ylim = c(0, 0.3),
     main = "MISC train VS test",
     ylab = "MISC",
     xlab = 'k')
lines(k_grid,
      misc_out,
      type = "b",
      lwd = 2,
      col = 2,
      lty = 1)
```



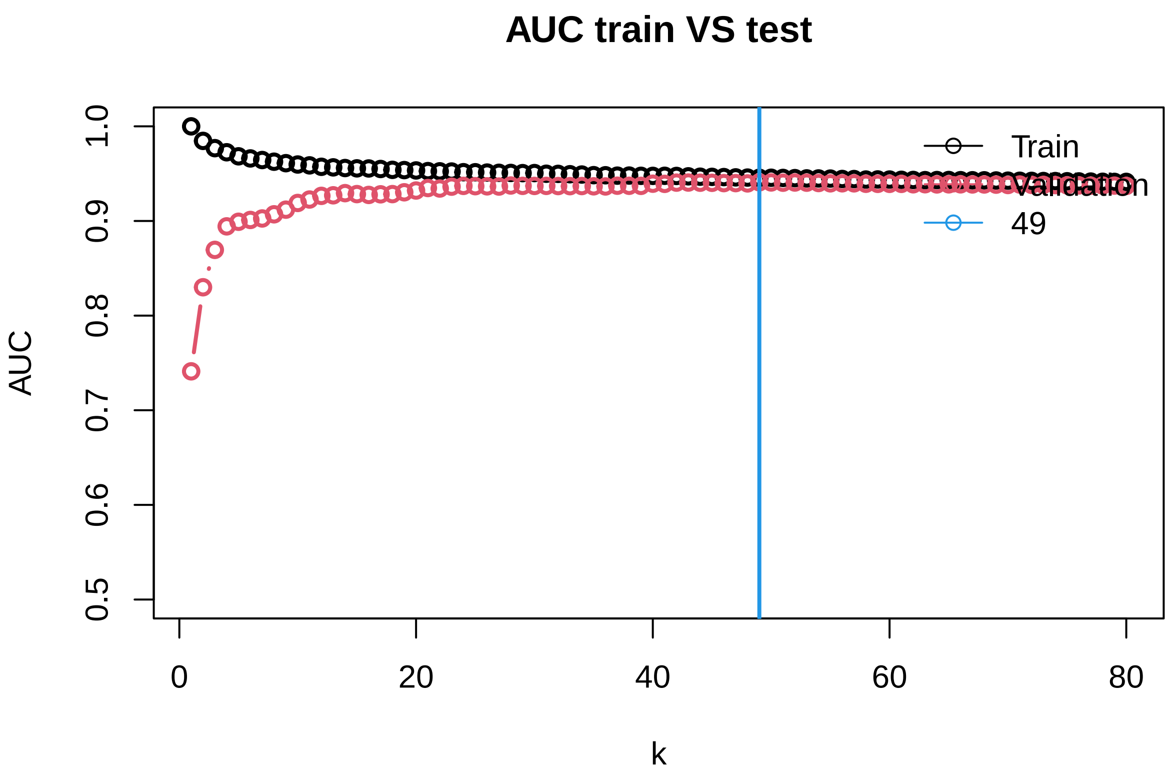
As the number of neighbors increases, the KNN model's ability to generalize improves, up to a point. After reaching an optimal k value, further increases in k do not significantly change the misclassification rate, indicating that the model has stabilized. The optimal k value for this dataset is where the test misclassification rate is the lowest and before it plateaus.

```
#MISC Performance doesn't converge
k_best_misc <- min(misc_out)
k_best_misc
```

```
## [1] 0.1016324
```

```
# Plot the performance (AUC)
plot(k_grid,
     auc_in,
     type = "b",
     lwd = 2,
     ylim = c(0.5, 1),
     main = "AUC train VS test",
     ylab = "AUC",
     xlab = 'k')
lines(k_grid,
      auc_out,
      type = "b",
      lwd = 2,
      col = 2,
      lty = 1)
# Best k using AUC : in this case train and test converge
k_best_auc <- k_grid[which.max(auc_out)]
abline(v = k_best_auc,
       col = 4,
       lwd = 2)
```

```r
legend("topright",
       c("Train", "Validation",k_best_auc),
       col = c(1,2,4),
       lty = c(1),
       bty = "n",
       pch = 21)
```



```r
best_auc <- max(auc_out)
c(k_best_auc, best_auc)
```

```
## [1] 49.0000000  0.9408681
```

```r
#Since were gonna base our model choice on AUC metric, we choose k_best_auc as best K for knn
```

The optimal number of neighbors (k) for the KNN model, based on the Area Under the ROC Curve (AUC) metric, is 49. This model achieves the best AUC value of approximately 0.941, indicating a very high ability to distinguish between the classes.

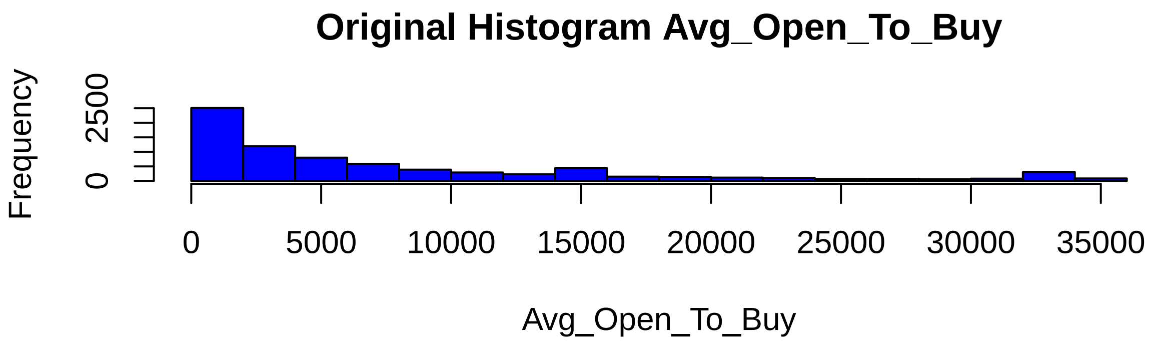
Best Model (Q5)

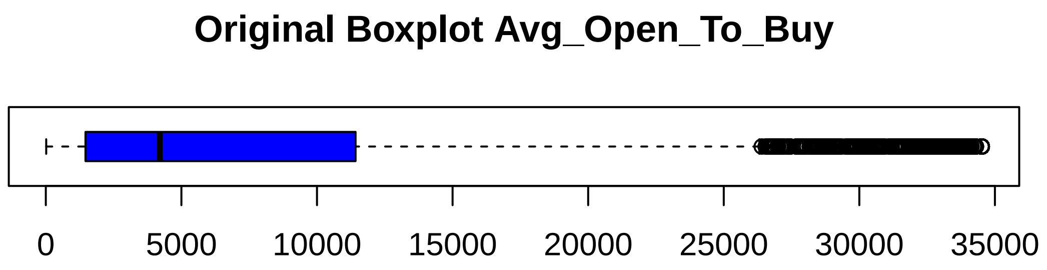
First, let's start off with Outlier Removal so that our models are not too affected by extreme values.

```r
library(skimr)
skimmed <- skim(Data)
```

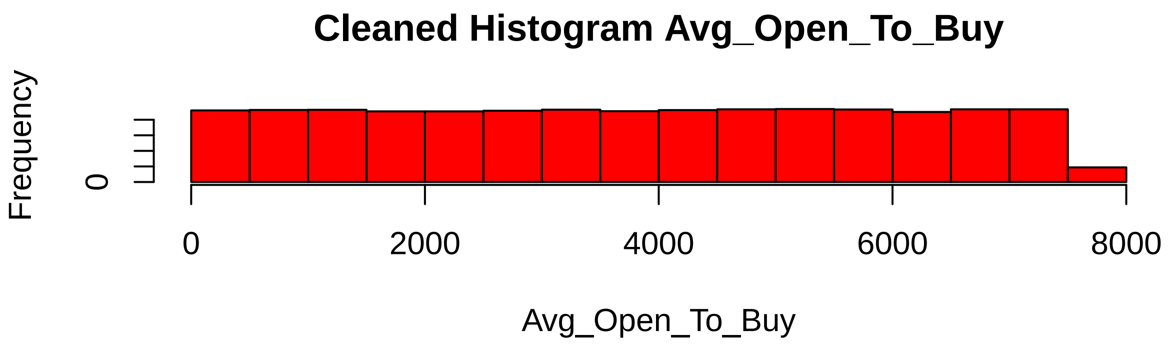
From skimmed, it seems like some variables might be subjected to outliers. We are going to try and remove outliers basing ourselves on the plots and quantiles. We start by determining the threshold using the IQR approach as a function remove outliers.

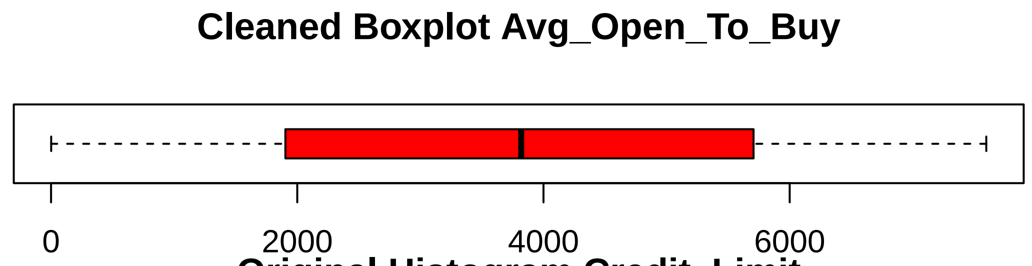
```r
remove_outliers_indices <- function(data, column_name) {
  IQR_value <- IQR(data[[column_name]], na.rm = TRUE)
  Q1 <- quantile(data[[column_name]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[column_name]], 0.75, na.rm = TRUE)

  lower_bound <- Q1 - 1.5 * IQR_value
  upper_bound <- Q3 + 1.5 * IQR_value

  cat("Thresholds for", column_name, "\n",
      "Lower bound:", lower_bound, "\n",
      "Upper bound:", upper_bound, "\n\n")

  return(which(data[[column_name]] >= lower_bound & data[[column_name]] <= upper_bound))
}

variables <- c("Avg_Open_To_Buy", "Credit_Limit", "Total_Amt_Chng_Q4_Q1", "Total_Trans_Amt", "Total_Ct_

par(mfrow = c(2, 1)) # Set up the plotting area to have 2 rows and 1 column

for (var in variables) {
  hist(Data[[var]], main = paste("Original Histogram", var), xlab = var, col = 'blue')
  boxplot(Data[[var]], main = paste("Original Boxplot", var), horizontal = TRUE, col = 'blue')

  cleaned_data <- remove_outliers_indices(Data, var)

  hist(cleaned_data, main = paste("Cleaned Histogram", var), xlab = var, col = 'red')
  boxplot(cleaned_data, main = paste("Cleaned Boxplot", var), horizontal = TRUE, col = 'red')
}
```

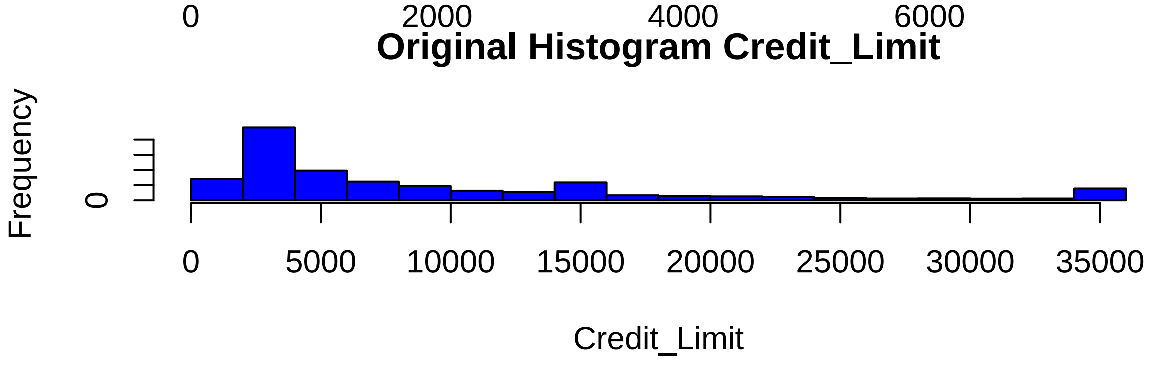


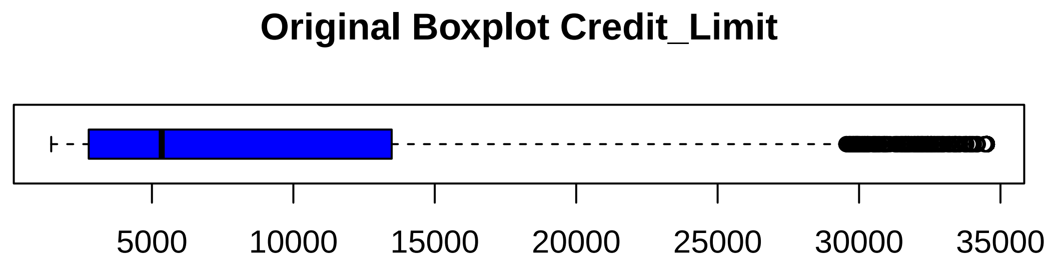


```
## Thresholds for Avg_Open_To_Buy
##  Lower bound: -13479.5
##  Upper bound: 26364.5
```

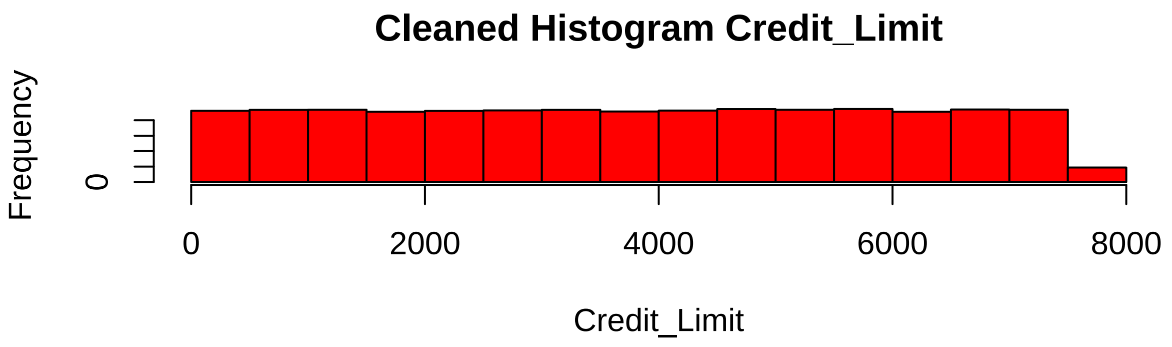


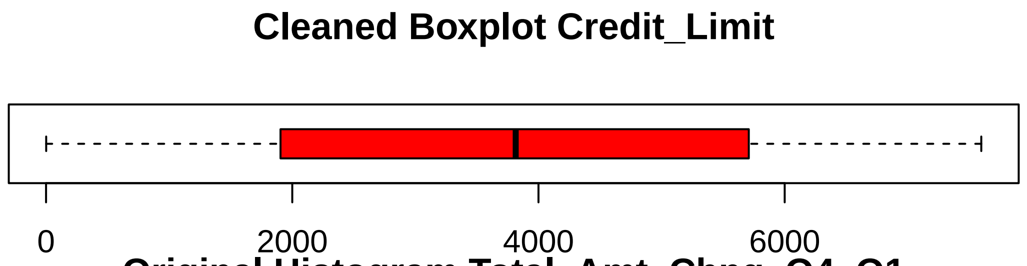


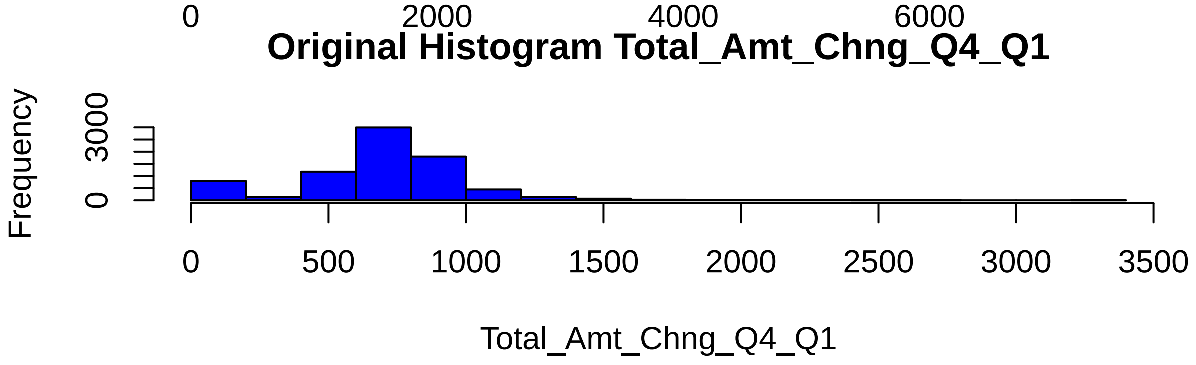


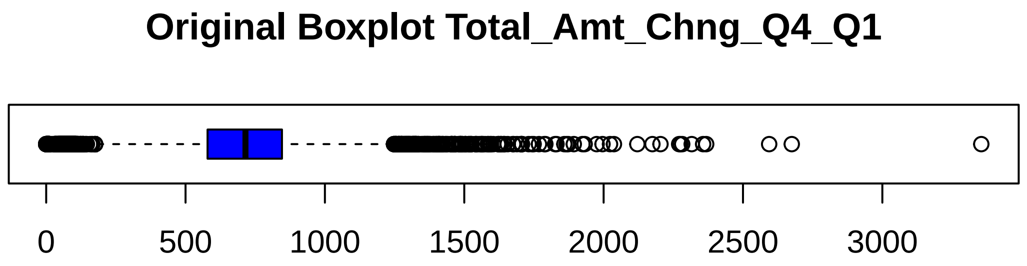


```
## Thresholds for Credit_Limit
##  Lower bound: -13303
##  Upper bound: 29545
```

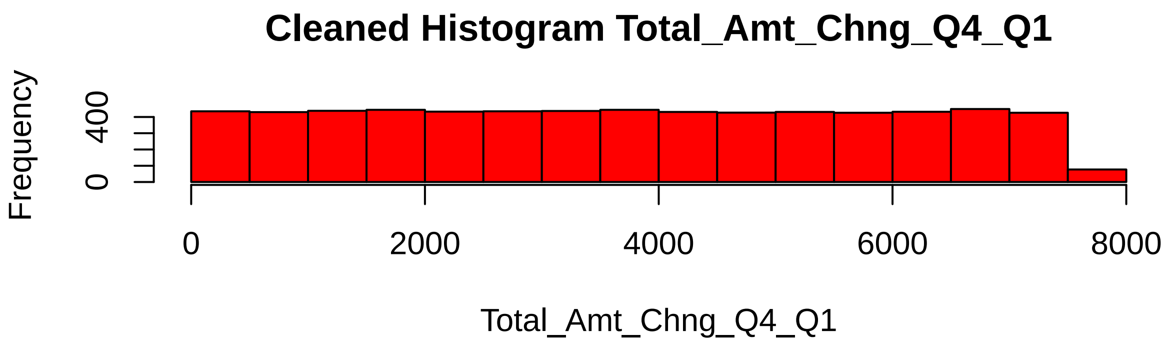


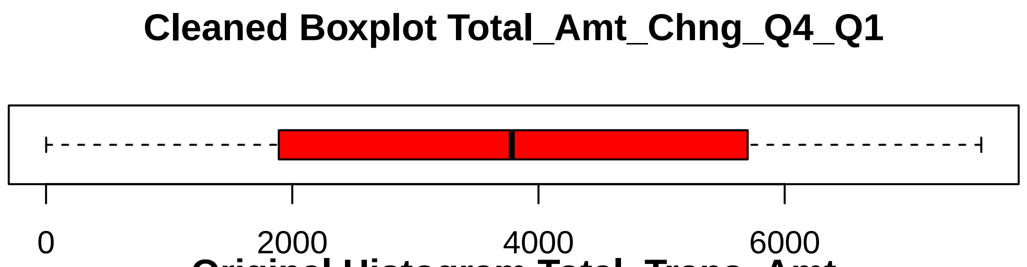


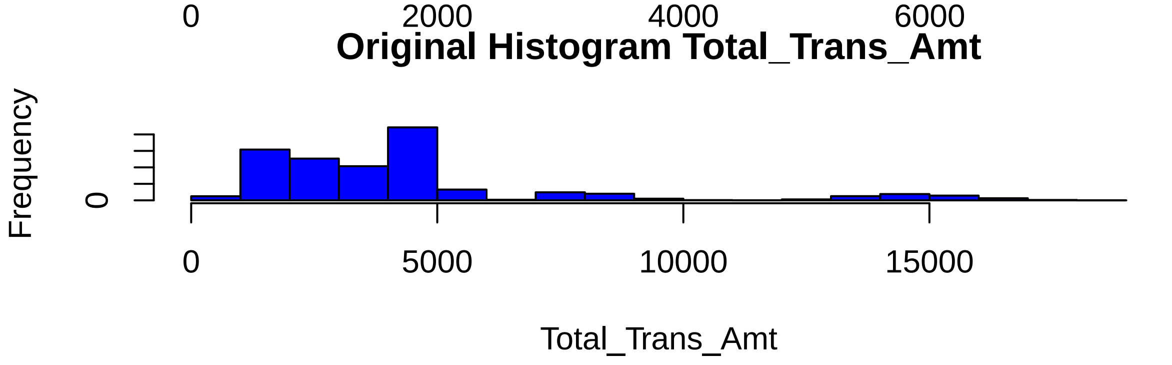


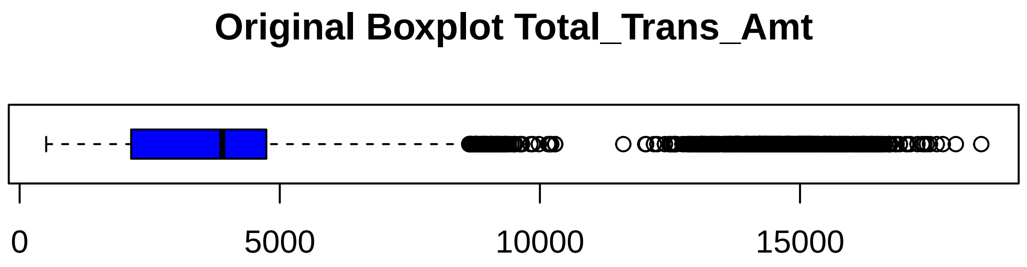


```
## Thresholds for Total_Amt_Chng_Q4_Q1
##  Lower bound: 178.5
##  Upper bound: 1246.5
```

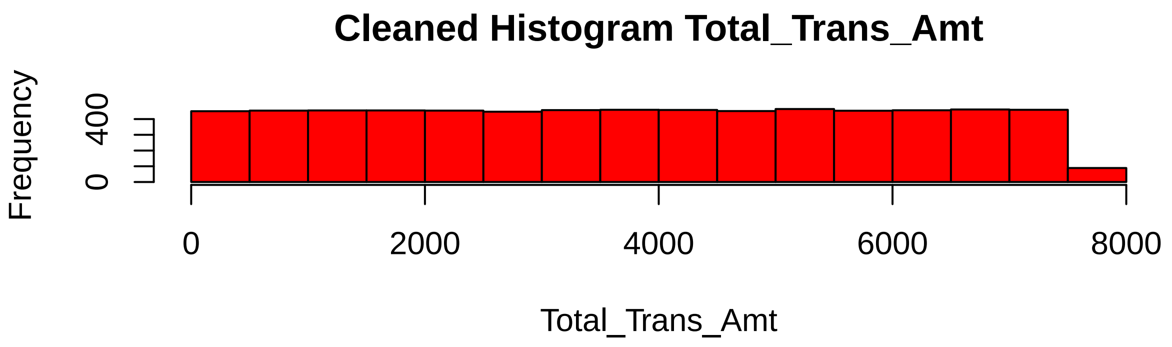


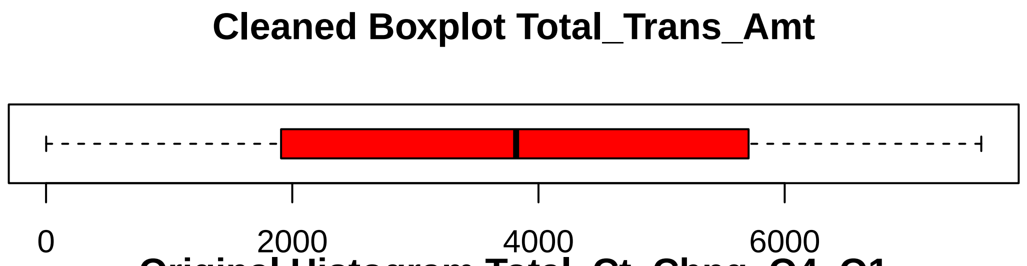


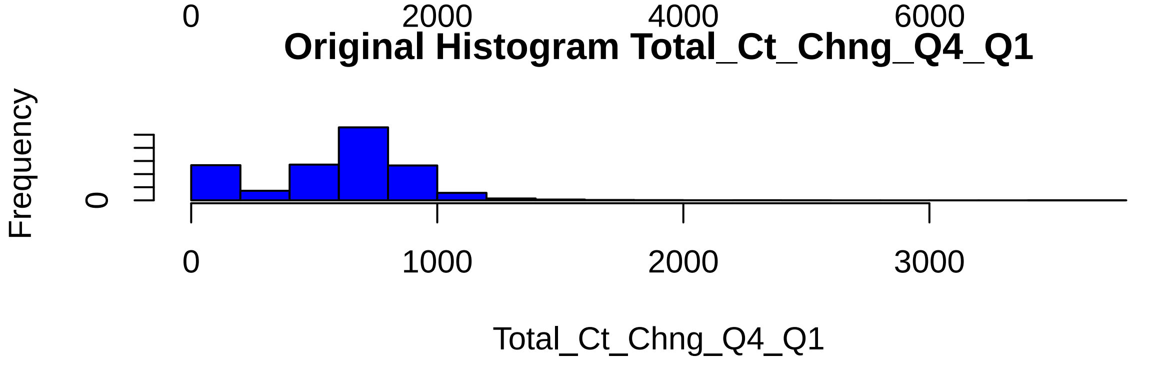


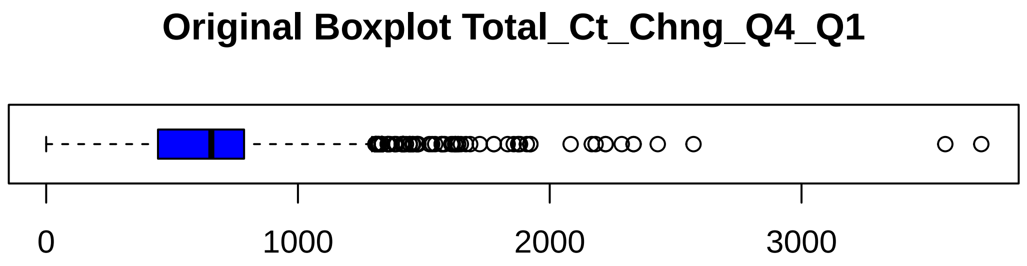


```
## Thresholds for Total_Trans_Amt
##  Lower bound: -1755.5
##  Upper bound: 8640.5
```

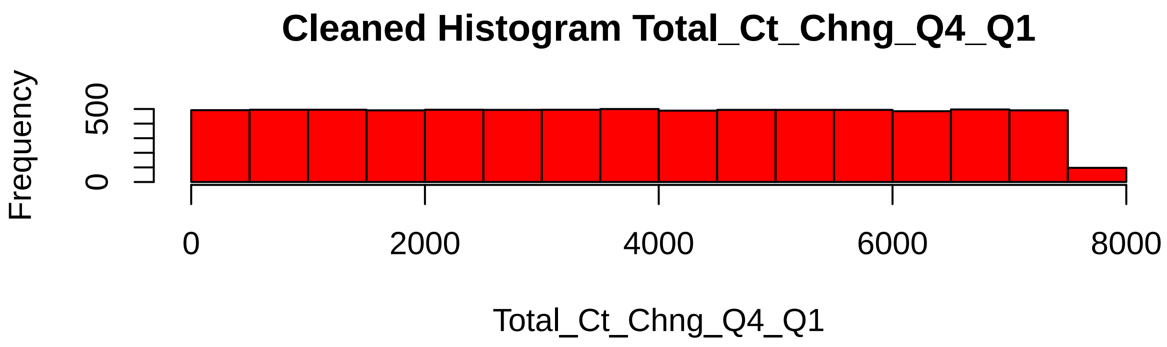


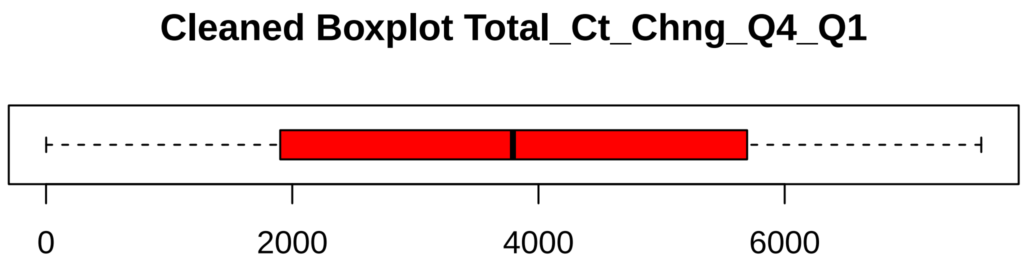






```
## Thresholds for Total_Ct_Chng_Q4_Q1
##  Lower bound: -69
##  Upper bound: 1299
```





As seen above, we first found the intervals in which the outliers take place. We have successfully identified them, and then removed them. In the plots, we can observe the difference of the distribution of the data before and after the operation of the removal of the outliers.

Now, let's move on to Feature Engineering, Looking at the data set, we notice that the variable Credit_Limit is the sum of the 2 variables Avg_Open_To_Buy and Total_Revolving_Bal and since from the correlation matrix Avg_Open_To_Buy is highly correlated with multiple variables we can extract a new variable Tot_Rev_Bal_ratio that explains the percentage of Credit_Limit represented by Total_Revolving_Bal and Avg_Open_To_Buy is explained by the difference between 1 and the ratio. We also extract a new variable representing the average transaction amount for each customer, it can be a good indicator of the possibility of closing the account

```
Data$Tot_Rev_Bal_ratio <- as.numeric(Data$Total_Revolving_Bal / Data$Credit_Limit)
Data$Avg_Trans_Amt <-as.numeric(Data$Total_Trans_Amt / Data$Total_Trans_Ct)
Closed_Account <- as.factor(Data$Closed_Account)

head(Data)
```

```
##   CLIENTNUM Customer_Age Gender Dependent_count Education_Level Marital_Status
## 1 721038408           38      F               2     High School        Married
## 2 711968358           55      F               4        Graduate         Single
## 3 719174433           36      F               1            <NA>         Single
## 4 712841883           50      M               3         College         Single
## 5 715980258           36      F               1       Doctorate         Single
## 6 708394908           44      M               0     High School         Single
##   Card_Category Months_on_book Total_Relationship_Count Months_Inactive_12_mon
## 1          Blue             27                        4                      1
## 2          Blue             44                        1                      2
## 3          Blue             24                        2                      1
## 4        Silver             36                        3                      3
## 5          Blue             25                        1                      2
## 6          Blue             33                        5                      1
##   Contacts_Count_12_mon Credit_Limit Total_Revolving_Bal Avg_Open_To_Buy
## 1                     2         1830                   0            1830
## 2                     1         2638                1423            1215
## 3                     2         1735                   0            1735
## 4                     2        34516                2517           31999
## 5                     3         2835                2511             324
## 6                     3         2337                1442             895
##   Total_Amt_Chng_Q4_Q1 Total_Trans_Amt Total_Trans_Ct Total_Ct_Chng_Q4_Q1
## 1                  736            1741             43                 654
## 2                  551            4153             77                 925
## 3                   74            2467             35                 346
## 4                  735            7155             66                  65
## 5                  731            4811             77                 925
## 6                  659            3945             65                 548
##   Avg_Utilization_Ratio    Income Closed_Account Tot_Rev_Bal_ratio
## 1                     0 169.77967              0         0.0000000
## 2                   539  87.02970              0         0.5394238
## 3                     0 140.27948              1         0.0000000
## 4                    73  90.16358              1         0.0729227
## 5                   886 101.55168              0         0.8857143
## 6                   617  46.92683              0         0.6170304
##   Avg_Trans_Amt
## 1      40.48837
## 2      53.93506
## 3      70.48571
## 4     108.40909
## 5      62.48052
## 6      60.69231
```

```
summary(Data$Tot_Rev_Bal_ratio)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.02055 0.16092 0.26401 0.48162 0.99393
```

```
summary(Data$Avg_Trans_Amt)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.14   47.60   55.89   62.70   65.47  190.19
```

Next, we remove 'CLIENTNUM' and 'Avg_Open_To_Buy' as we don't need it anymore.

```
Data <- Data[, !names(Data) %in% c("CLIENTNUM", "Avg_Open_To_Buy", "Closed_Account")]
Data <- cbind(Data, Closed_Account)
```

Let's move on to making the validation set now.

```
library(caret)
```

```
## Loading required package: lattice
```

```
trainrows <- createDataPartition(Data$Closed_Account, p = 0.8, list = FALSE)
d_train <- Data[trainrows,]
d_val <- Data[-trainrows,]

#Scaling
for(i in 1:ncol(d_val)){
  if (is.numeric(d_val[1,i])){
    d_val[,i] = scale(d_val[,i], center = mean(d_train[,i]), scale = sd(d_train[,i]))
  }
```

```r
}
```

We now move on to KNN-Imputation for imputing missing values.

```r
# We exclude the dependent variable from this process
# We also split the data into predictors and the target variable
Closed_Account <- as.factor(d_train$Closed_Account)
d_train <- d_train[,!names(d_train) %in% c("Closed_Account")]

# Processing the data
missingtraindata_model <- preProcess(d_train, method='knnImpute',k = k_best_auc)
missingtraindata_model
```

```
## Created from 4802 samples and 20 variables
##
## Pre-processing:
##   - centered (16)
##   - ignored (4)
##   - 49 nearest neighbor imputation (16)
##   - scaled (16)
```

```r
# Use the imputation model to predict the values of missing data points
library(RANN)
d_train <- predict(missingtraindata_model, newdata = d_train)
d_train <- cbind(d_train, Closed_Account)

anyNA(d_train)
```

```
## [1] TRUE
```

```r
anyNA(d_val)
```

```
## [1] TRUE
```

```r
d_train <- na.omit(d_train)
d_val <- na.omit(d_val)

# Check if there is any NA value n our data set
anyNA(d_train)
```

```
## [1] FALSE
```

```r
anyNA(d_val)
```

```
## [1] FALSE
```

Now, let's encode our categorical variables for the training set and validation set.

```r
Closed_Account <- as.factor(d_train$Closed_Account)
d_train <- d_train[,!names(d_train) %in% c("Closed_Account")]
dmy <- dummyVars(" ~ .", data = d_train)
d_train <- data.frame(predict(dmy, newdata = d_train))
d_train <- cbind(d_train, Closed_Account)

Closed_Account <- as.factor(d_val$Closed_Account)
d_val <- d_val[,!names(d_val) %in% c("Closed_Account")]
dmy <- dummyVars(" ~ .", data = d_val)
d_val <- data.frame(predict(dmy, newdata = d_val))
d_val <- cbind(d_val, Closed_Account)

head(d_val)
```

```
##    Customer_Age Gender.F Gender.M Dependent_count Education_Level.College
## 2     1.0797478        1        0       1.2593712                       0
## 10    1.6992335        0        1      -1.0374256                       0
## 13    0.4602622        0        1      -0.2718267                       0
## 19    0.7080564        0        1      -0.2718267                       0
## 35   -2.3893719        1        0      -1.0374256                       1
## 40    0.2124679        1        0       0.4937723                       1
##    Education_Level.Doctorate Education_Level.Graduate
## 2                          0                        1
## 10                         0                        0
## 13                         0                        1
## 19                         0                        1
## 35                         0                        0
## 40                         0                        0
##    Education_Level.High.School Education_Level.Post.Graduate
## 2                            0                             0
## 10                           1                             0
## 13                           0                             0
## 19                           0                             0
## 35                           0                             0
## 40                           0                             0
##    Education_Level.Uneducated Marital_Status.Divorced Marital_Status.Married
## 2                           0                       0                      0
## 10                          0                       0                      1
## 13                          0                       0                      1
## 19                          0                       0                      1
## 35                          0                       1                      0
## 40                          0                       0                      0
##    Marital_Status.Single Card_Category.Blue Card_Category.Gold
## 2                      1                  1                  0
## 10                     0                  1                  0
## 13                     0                  1                  0
## 19                     0                  1                  0
## 35                     0                  1                  0
## 40                     1                  1                  0
##    Card_Category.Platinum Card_Category.Silver Months_on_book
## 2                       0                    0     1.01328377
## 10                      0                    0     2.38158832
## 13                      0                    0     0.14254451
## 19                      0                    0     0.01815319
## 35                      0                    0    -2.59406458
## 40                      0                    0     0.14254451
##    Total_Relationship_Count Months_Inactive_12_mon Contacts_Count_12_mon
## 2                -1.8031458             -0.3276028            -1.3254433
## 10                0.7630358             -1.3251829             1.3845963
## 13               -0.5200550             -1.3251829             0.4812498
## 19                0.1214904              0.6699773            -0.4220968
## 35                0.7630358              0.6699773            -0.4220968
## 40                1.4045812              0.6699773            -1.3254433
##    Credit_Limit Total_Revolving_Bal Total_Amt_Chng_Q4_Q1 Total_Trans_Amt
## 2    -0.7307041           0.3277345           -0.4730690     -0.07349159
## 10   -0.7281427           0.3807830           -0.2063408     -0.89526934
## 13   -0.6763583          -1.4278010            0.4371410     -0.88528704
## 19   -0.7053130           0.8939964            0.3671249      0.13114546
## 35   -0.7832680          -0.3298203            1.1039616     -0.24671376
## 40    0.5772694          -1.4278010            0.2804382      0.18546089
##    Total_Trans_Ct Total_Ct_Chng_Q4_Q1 Avg_Utilization_Ratio      Income
## 2       0.5178548           1.0182658             1.0668642 -0.07857715
## 10     -1.6803713          -0.8616154             1.1111518  1.49188031
## 13     -1.0039940          -0.4047641            -0.9223867  0.30519058
## 19      0.8137699           0.7126694             1.5023588  0.28951205
## 35      0.2219398           0.9164003             0.5944632 -0.02054768
## 40      0.4755812           0.5305462            -0.9223867 -0.71739739
##    Tot_Rev_Bal_ratio Avg_Trans_Amt Closed_Account
## 2          1.0092925    -0.3332391              0
## 10         1.0515792    -0.3246969              0
## 13        -0.9747590    -1.0958596              0
## 19         1.4405143    -0.1888135              0
## 35         0.5365539    -0.4485001              0
## 40        -0.9747590     0.1344384              0
```

Next, We perform Feature Selection and remove any unimportant features from our dataset using Random Forests.

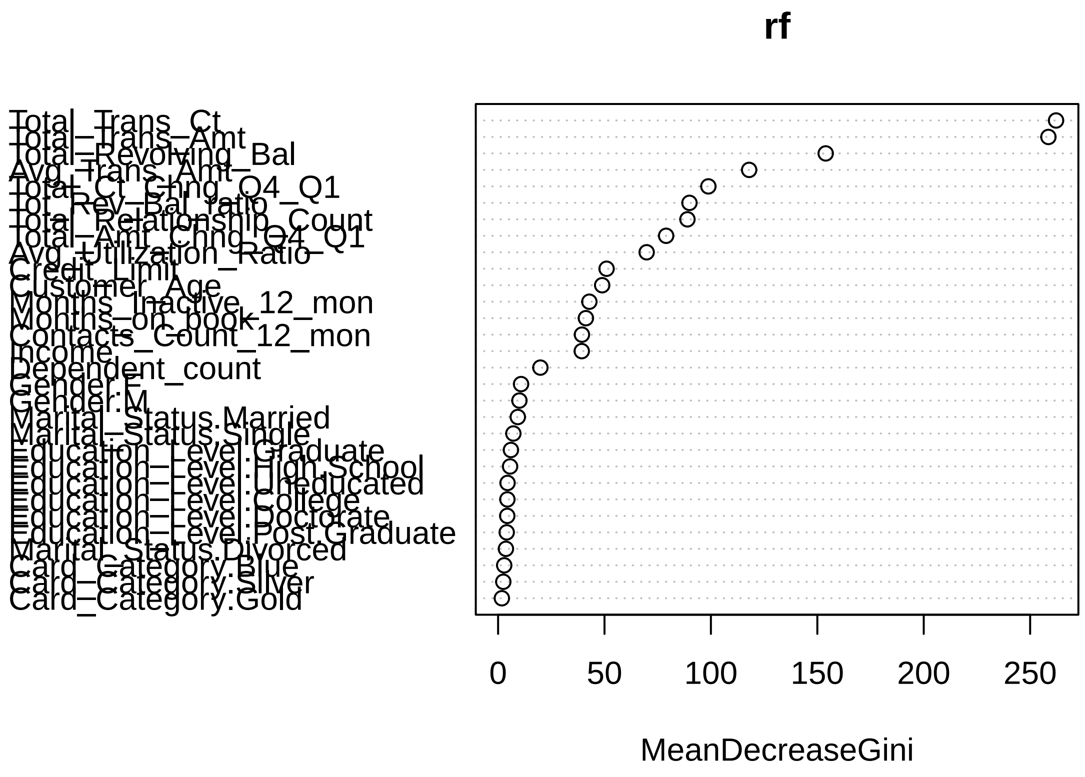
```r
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
Data <- rbind(d_train, d_val)
rf <- randomForest(Data$Closed_Account ~ ., data=Data)


# Importance of features according to the model
round(importance(rf), 2)
```

```
##                                 MeanDecreaseGini
## Customer_Age                               48.91
## Gender.F                                   10.80
## Gender.M                                   10.01
## Dependent_count                            19.87
## Education_Level.College                     4.37
## Education_Level.Doctorate                   4.30
## Education_Level.Graduate                    6.03
## Education_Level.High.School                 5.64
## Education_Level.Post.Graduate               4.05
## Education_Level.Uneducated                  4.44
## Marital_Status.Divorced                     3.68
## Marital_Status.Married                      9.26
## Marital_Status.Single                      7.17
## Card_Category.Blue                         2.88
## Card_Category.Gold                         1.77
## Card_Category.Platinum                     0.72
## Card_Category.Silver                       2.45
## Months_on_book                            41.25
## Total_Relationship_Count                  88.97
## Months_Inactive_12_mon                    42.88
## Contacts_Count_12_mon                     39.39
## Credit_Limit                              50.97
## Total_Revolving_Bal                      153.94
## Total_Amt_Chng_Q4_Q1                      78.96
## Total_Trans_Amt                          258.58
## Total_Trans_Ct                           262.18
## Total_Ct_Chng_Q4_Q1                       98.81
## Avg_Utilization_Ratio                     69.81
## Income                                    39.32
## Tot_Rev_Bal_ratio                         89.92
## Avg_Trans_Amt                            117.97
```

```r
varImpPlot(rf)
```



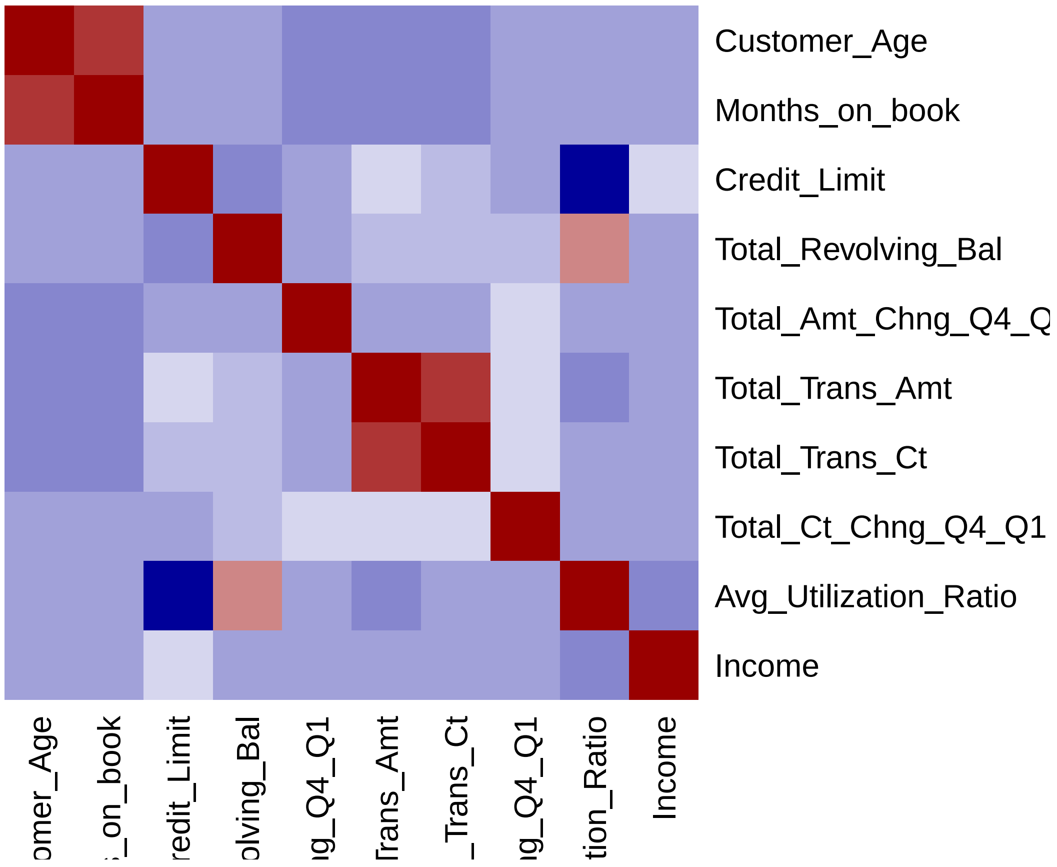
Here the plot shows 'Total_Trans_Ct' and 'Total_Trans_Amt' as the most influential variables in predicting the target, with the highest Mean Decrease in Gini, indicating they are important for the model.

```r
# Unimportant features that should be removed
disc <- c("Card_Category.Platinum", "Card_Category.Gold", "Card_Category.Silver", "Marital_Status.Divor
          "Education_Level.Graduate", "Education_Level.Doctorate", "Education_Level.High.School", "Educa
          "Education_Level.Uneducated", "Card_Category.Blue", "Gender.M", "Gender.F", "Dependent_count"

# Removing unimportant features
```

```r
Data <- Data[,!names(Data) %in% disc]
d_train <- d_train[,!names(d_train) %in% disc]
d_val <- d_val[,!names(d_val) %in% disc]

#Training and validation sets
x_train <- d_train[, !(names(d_train) %in% "Closed_Account")]
y_train <- d_train$Closed_Account

x_val <- d_val[, !(names(d_val) %in% "Closed_Account")]
y_val <- d_val$Closed_Account
```

Lets now look at the variables that are highly correlated with each other in our dataset.

```r
Data_num <- Data[,names(Data) %in% contcols]
Cor.Matrix = cor(Data_num)

col<- colorRampPalette(c(rgb(0, 0,.6), "white", rgb(.6,0,0)))(20)
heatmap(x = Cor.Matrix, col = col, symm = T,Rowv = NA)
```



```r
correlated_var <- matrix(nrow=1, ncol=3)
for(i in 2:ncol(Data_num)){
  if (is.numeric(Data_num[1,i])) {
    for (j in i:ncol(Data_num)) {
      if (i != j & (Cor.Matrix[i,j] > 0.70 | Cor.Matrix[i,j] < -0.70)){
        correlated_var <- rbind(correlated_var, c(colnames(Data_num)[i],colnames(Data_num)[j],Cor.Matri
      }
    }
  }
}
```

After appyling Random Forests we can observe the strong positive correlation between 'Total_Trans_Ct' and 'Total_Amt_Chng_Q4_Q1' which suggests that customers with a higher number of transactions also tend to have significant changes in transaction amounts between Q4 and Q1. Additionally, 'Credit_Limit' and 'Avg_Utilization_Ratio' exhibit a notable negative correlation, indicating that customers with higher credit limits tend to use a smaller percentage of their available credit.

We remove one of the variables that are highly correlated to another as they carry similar information and thereby, minimise noise for our models.

```
disc <- c("Customer_Age","Avg_Open_To_Buy")

# Removing unimportant features
Data <- Data[,!names(Data) %in% disc]
d_train <- d_train[,!names(d_train) %in% disc]
d_val <- d_val[,!names(d_val) %in% disc]

x_train <- d_train[, !(names(d_train) %in% "Closed_Account")]
y_train <- d_train$Closed_Account

x_val <- d_val[, !(names(d_val) %in% "Closed_Account")]
y_val <- d_val$Closed_Account
```

Now, let's move on to determining models. We take into consideration 3 models that we think will be most suitable.

1 - Logistic Regression: This model is inherently interpretable. It's particularly useful if you need to explain the reasoning behind predictions.It's also computationally efficient and can serve as a baseline model.

```
#Logistic Regression Model
logit_fit1 <- glm(d_train$Closed_Account ~ .,
                  family = "binomial",
                  data = d_train)
#Base threshold
tt <- 0.5

pred_logit <- as.factor(ifelse(logit_fit1$fitted.values > tt, 1, 0))

# Confusion matrix for the training set
confusionMatrix(table(pred_logit,d_train$Closed_Account))
```

```
## Confusion Matrix and Statistics
##
##
## pred_logit    0    1
##          0 3914  331
##          1  136  421
##
##                Accuracy : 0.9027
##                  95% CI : (0.894, 0.911)
##     No Information Rate : 0.8434
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5884
```

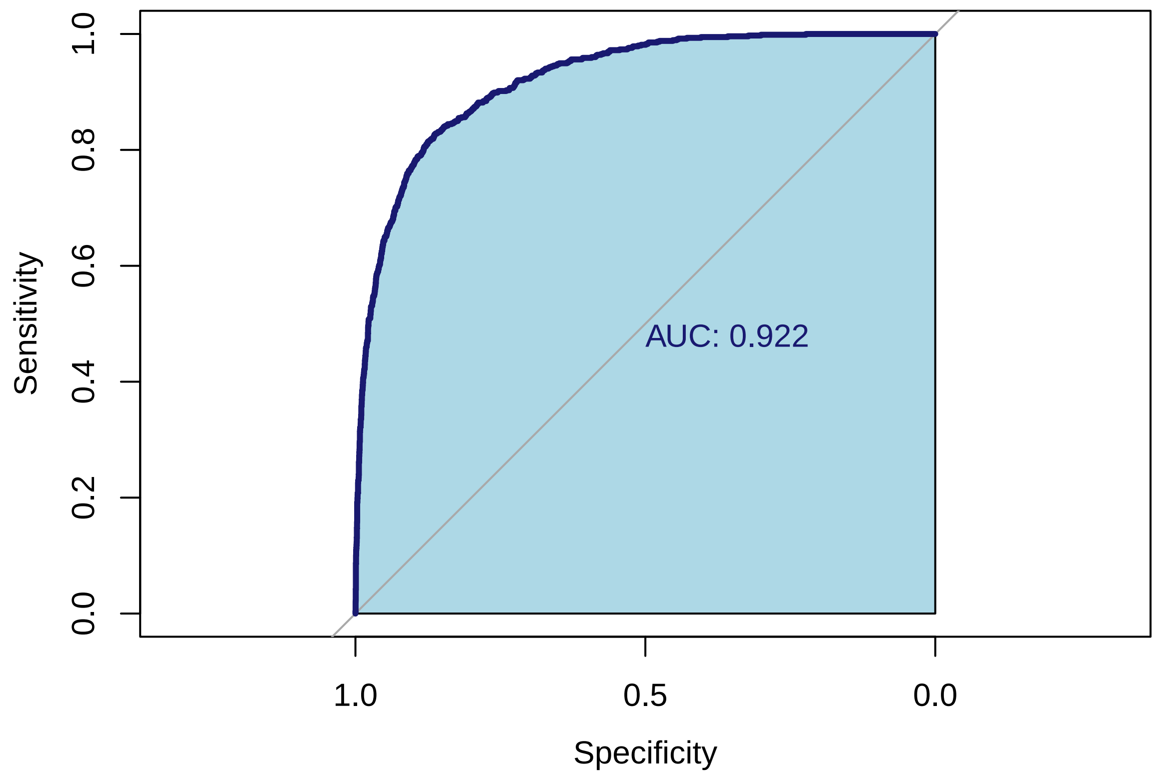
```
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9664
##             Specificity : 0.5598
##          Pos Pred Value : 0.9220
##          Neg Pred Value : 0.7558
##              Prevalence : 0.8434
##          Detection Rate : 0.8151
##    Detection Prevalence : 0.8840
##       Balanced Accuracy : 0.7631
##
##        'Positive' Class : 0
##
```

The model correctly predicted the majority of customers who did not close their accounts (3894 true negatives) but was less successful in correctly identifying those who did (418 true positives). It incorrectly predicted 327 customers would close their accounts when they did not (false positives) and failed to identify 137 customers who closed their accounts (false negatives).

```r
# ROC curve for the training set
roc_logit <- pROC::roc(d_train$Closed_Account,
                       logit_fit1$fitted.values,
                       plot = TRUE,
                       col = "midnightblue",
                       lwd = 3,
                       auc.polygon = T,
                       auc.polygon.col = "lightblue",
                       print.auc = T)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
# AUC in the training set
roc_logit$auc
```

```
## Area under the curve: 0.9221
```

The ROC curve for the training set shows an AUC of 0.922, which is excellent, demonstrating the model's good discriminative ability between the two classes.

```
## Validation set ##

prob_out_logit <- predict(logit_fit1,
                          newdata = d_val,
                          type = "response")
pred_out_logit <- as.factor(ifelse(prob_out_logit > tt, 1, 0))

# Confusion matrix for the validation set
confusionMatrix(table(pred_out_logit,d_val$Closed_Account))
```

```
## Confusion Matrix and Statistics
##
##
## pred_out_logit   0   1
##              0 961  81
##              1  37 103
##
##                Accuracy : 0.9002
##                  95% CI : (0.8816, 0.9167)
##     No Information Rate : 0.8443
##     P-Value [Acc > NIR] : 1.423e-08
##
##                   Kappa : 0.5792
```

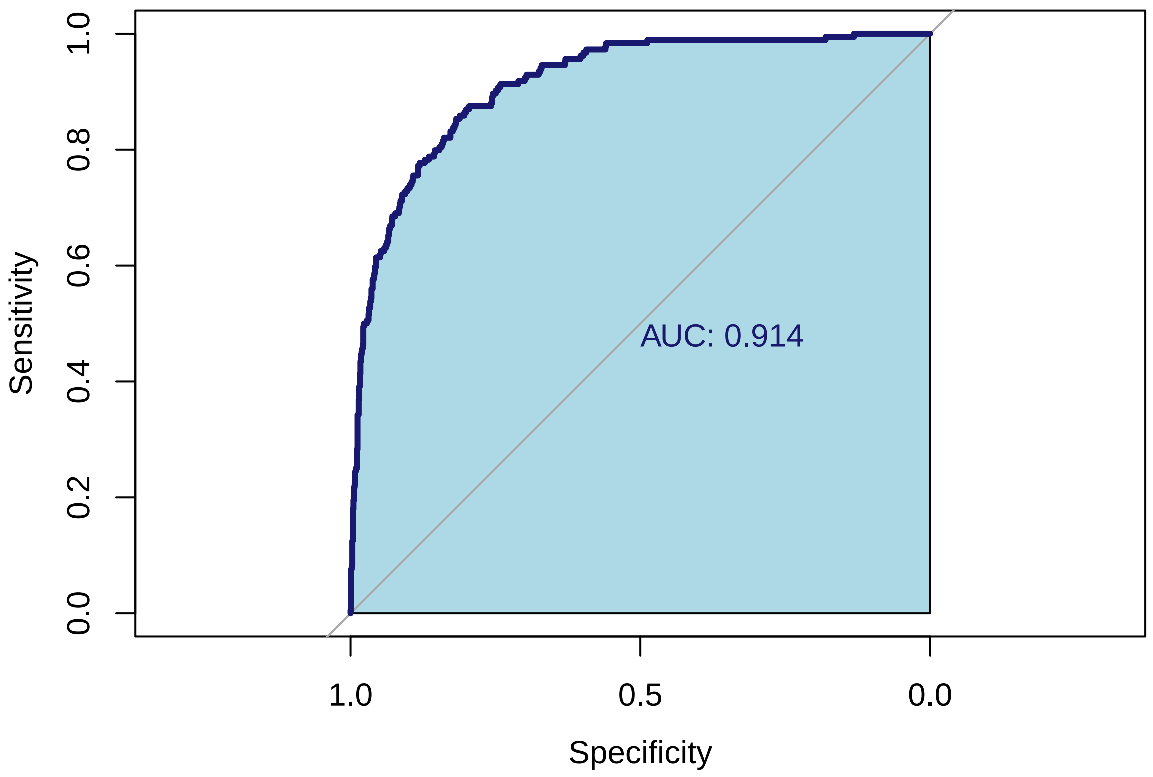
```
##
##  Mcnemar's Test P-Value : 7.543e-05
##
##             Sensitivity : 0.9629
##             Specificity : 0.5598
##          Pos Pred Value : 0.9223
##          Neg Pred Value : 0.7357
##              Prevalence : 0.8443
##          Detection Rate : 0.8130
##    Detection Prevalence : 0.8816
##       Balanced Accuracy : 0.7614
##
##        'Positive' Class : 0
##
```

On the validation set, the model showed strong performance in predicting customers who retained their accounts (978 true negatives) and had a fair success rate in predicting account closures (110 true positives). It incorrectly predicted 81 as closures (false positives) and missed 39 actual closures (false negatives).

```r
# ROC curve for the validation set
roc_logit_val <- pROC::roc(d_val$Closed_Account,
                           prob_out_logit,
                           plot = TRUE,
                           col = "midnightblue",
                           lwd = 3,
                           auc.polygon = T,
                           auc.polygon.col = "lightblue",
                           print.auc = T)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```r
# AUC in the training set
roc_logit_val$auc
```

```
## Area under the curve: 0.9139
```

The AUC for the validation set is 0.916, very close to the training set's AUC, suggesting that the model's performance is stable across both sets.

Overall, the Logistic Regression model performs very well on both the training and validation sets, with high accuracy and sensitivity, and an excellent AUC indicating strong discriminative power.

2 - Random Forests: This ensemble model can handle non-linear relationships and interactions between variables without the need for manual feature engineering. It's robust to outliers and can handle unbalanced datasets well.

```r
#Random Forest
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(ROCR)

# Making the random forest model
rf <- randomForest(d_train$Closed_Account ~ ., data=d_train)

# Vector with the predicted probabilities on the training set
```

```r
rf_prob = predict(rf,d_val[-ncol(d_val)],type="prob")

#Predictions on the validation set
val.predictions = predict(rf, newdata=d_val[-ncol(d_val)])

#Confusion matrix for he validation set
confusionMatrix(table(val.predictions,d_val$Closed_Account))
```

```
## Confusion Matrix and Statistics
##
##
## val.predictions   0   1
##               0 979  39
##               1  19 145
##
##                Accuracy : 0.9509
##                  95% CI : (0.937, 0.9625)
##     No Information Rate : 0.8443
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8047
##
##  Mcnemar's Test P-Value : 0.0126
##
##             Sensitivity : 0.9810
##             Specificity : 0.7880
##          Pos Pred Value : 0.9617
##          Neg Pred Value : 0.8841
##              Prevalence : 0.8443
##          Detection Rate : 0.8283
##    Detection Prevalence : 0.8613
##       Balanced Accuracy : 0.8845
##
##        'Positive' Class : 0
##
```

Here we can obsereve that the sensitivity is extremely high at 98.03%, showing the model is adept at identifying customers who will not close their accounts, while specificity is decent at 78.01%, indicating room for improvement in correctly identifying customers who will close their accounts. The high kappa score of 0.7978 reflects strong agreement between the model's predictions and the actual values.

```r
# ROC/AUC on training data
roc.train <- roc(d_train$Closed_Account, rf$votes[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc(roc.train)
```

```
## Area under the curve: 0.985
```

The Area Under the Curve (AUC) for the training set is 0.9858, which signifies an excellent ability of the Random Forest model to differentiate between customers who will close their accounts and those who will not.

```r
# ROC/AUC on test data
roc.val <- roc(d_val$Closed_Account, rf_prob[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
auc(roc.val)
```

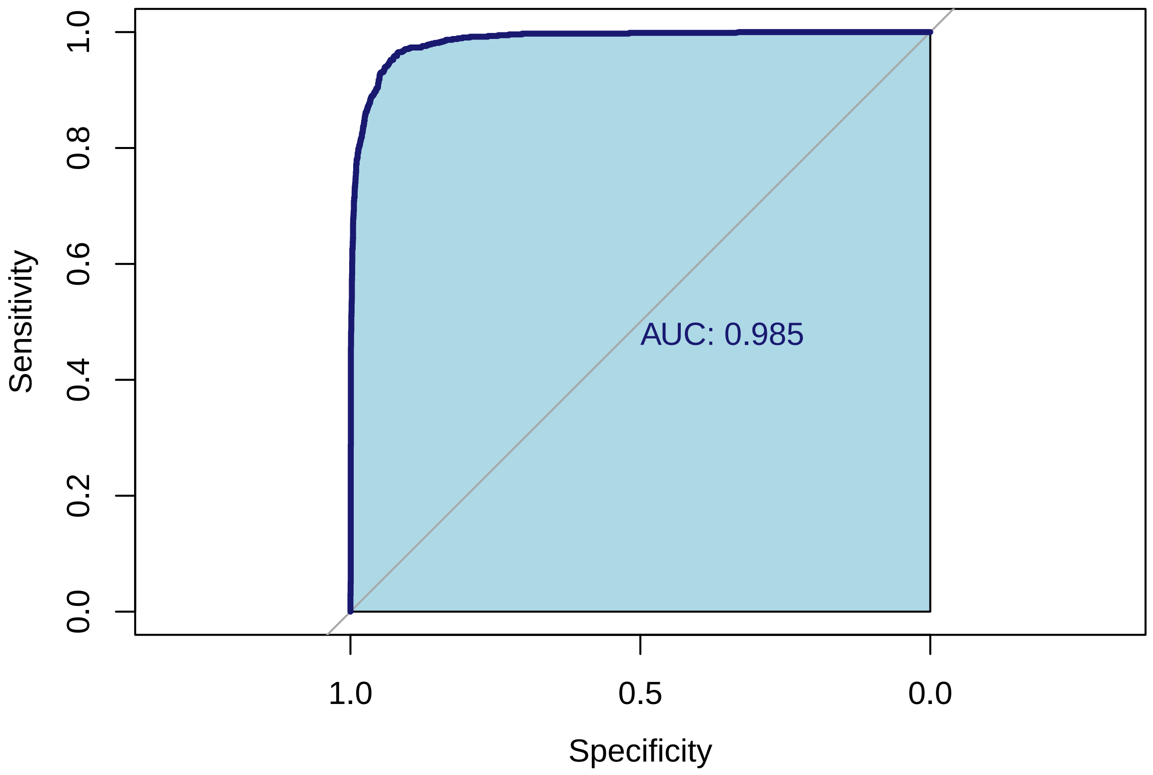
```
## Area under the curve: 0.9846
```

On the test data, the AUC is 0.9831, indicating that the model's performance is highly consistent and generalizes well to unseen data.

```r
# ROC curves
roc_rf <- pROC::roc(d_train$Closed_Account,
                    rf$votes[,2],
                    plot = TRUE,
                    col = "midnightblue",
                    lwd = 3,
                    auc.polygon = T,
                    auc.polygon.col = "lightblue",
                    print.auc = T)
```

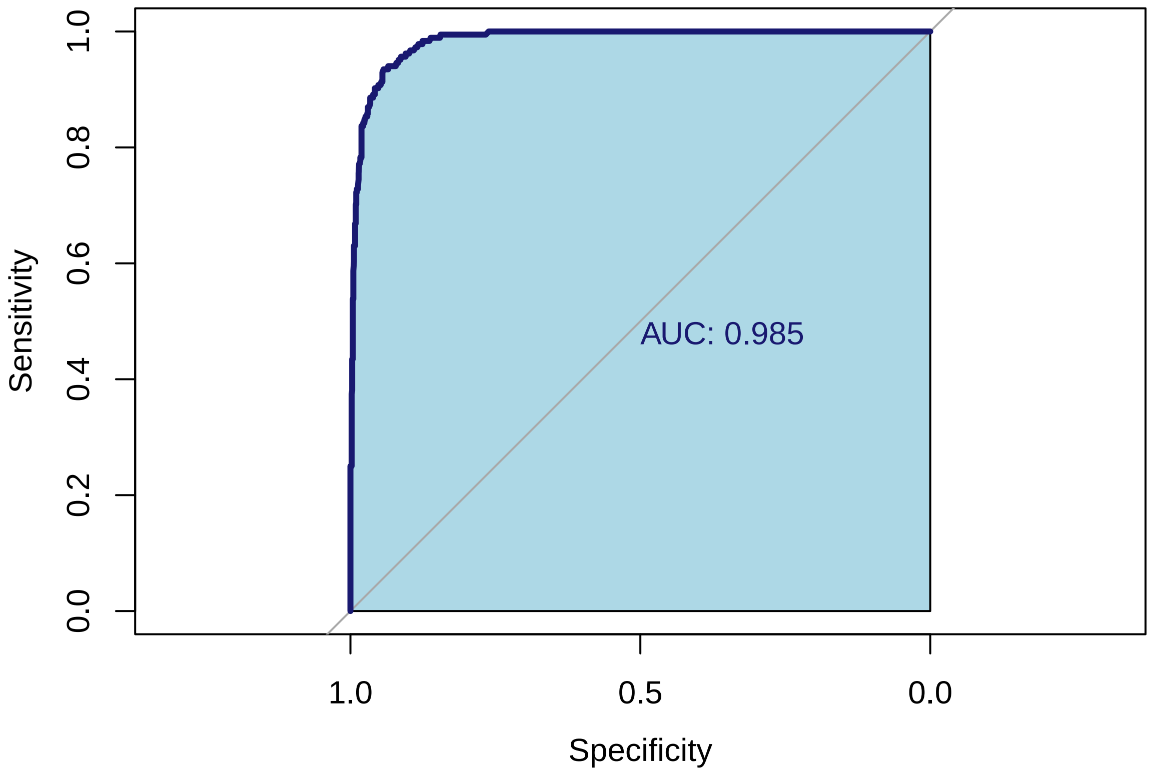
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```r
roc_rf_val <- pROC::roc(d_val$Closed_Account,
                        rf_prob[,2],
                        plot = TRUE,
                        col = "midnightblue",
                        lwd = 3,
                        auc.polygon = T,
                        auc.polygon.col = "lightblue",
                        print.auc = T)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

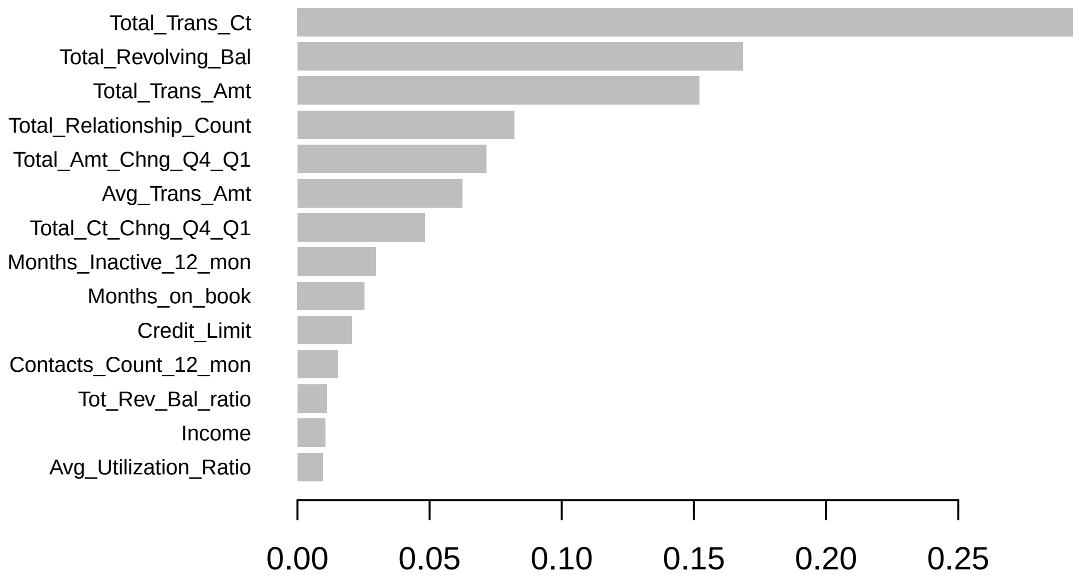


3 - XGBoost : It is a powerful gradient boosting framework known for its performance on structured or tabular data and because of its ability to optimize for speed and model performance. It can automatically handle missing data, and it's efficient with large datasets.

```
library(xgboost)
library(caTools)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:xgboost':
##
##     slice

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(cvms)
library(caret)

# Transforming the training and validation sets into xgbMatrices (xgboost works with xgbMatrices instea
xgb_train <- xgb.DMatrix(data = as.matrix(x_train), label = as.numeric(as.character(y_train)))
xgb_val <- xgb.DMatrix(data = as.matrix(x_val), label = as.numeric(as.character(y_val)))

# List of chosen parameters for our model
xgb_params <- list(
  booster = "gbtree",
  eta = 0.01,
  max_depth = 8,
  gamma = 4,
  subsample = 0.75,
  colsample_bytree = 1,
  objective = "binary:logistic",
  eval_metric = "auc"
)

# Making the xgboost model
xgb_model <- xgb.train(
  params = xgb_params,
  data = xgb_train,
  nrounds = 5000,
  verbose = 1
)

# Variable importance
importance_matrix <- xgb.importance(
  feature_names = colnames(xgb_train),
  model = xgb_model
)

xgb.plot.importance(importance_matrix)
```



Here we can observe that 'Total_Trans_Ct' and 'Total_Revolving_Bal' are the most influential features, having the highest gain and strong coverage, suggesting their pivotal role in predicting account closure.

```
# Predicted values for the validation set
xgb_preds <- predict(xgb_model, xgb_val)

# ROC/AUC on training set
train_pred_xgb <- predict(xgb_model, xgb_train)
roc.train <- roc(d_train$Closed_Account, train_pred_xgb)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
auc(roc.train)
```

```
## Area under the curve: 0.9993
```

```
# ROC/AUC on validation set
roc.val <- roc(d_val$Closed_Account, xgb_preds)
```

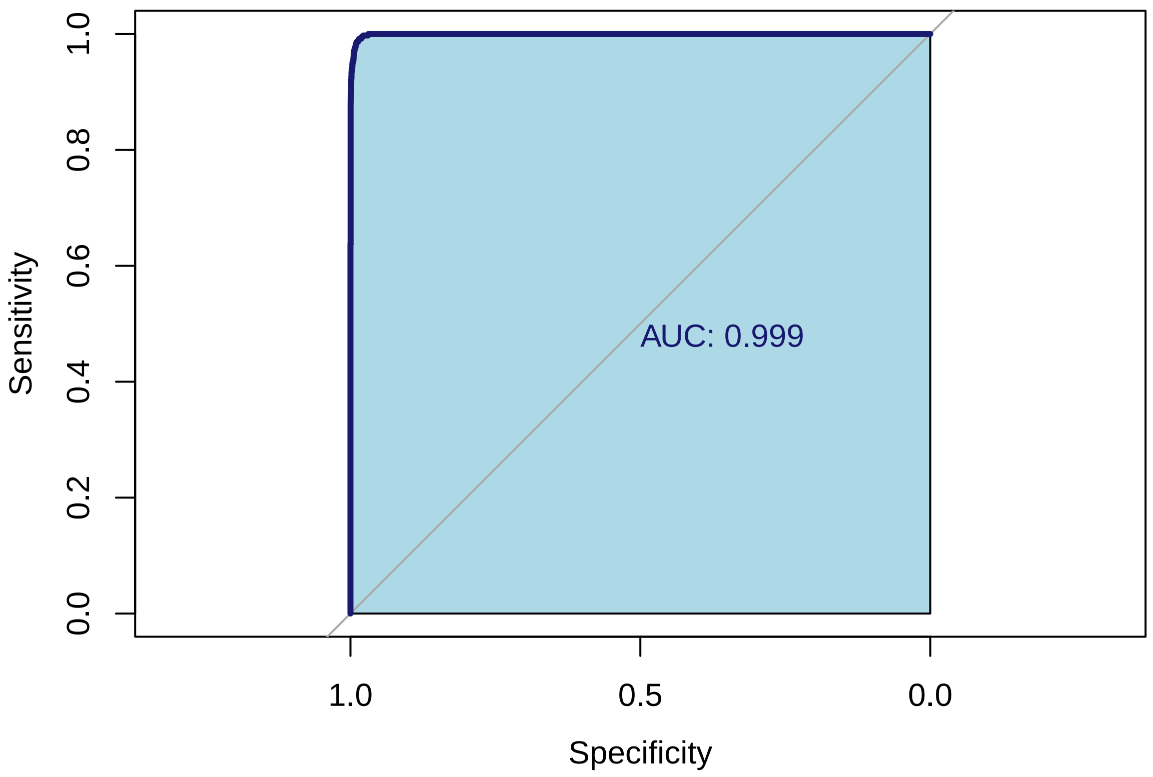
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
auc(roc.val)
```

```
## Area under the curve: 0.9899
```

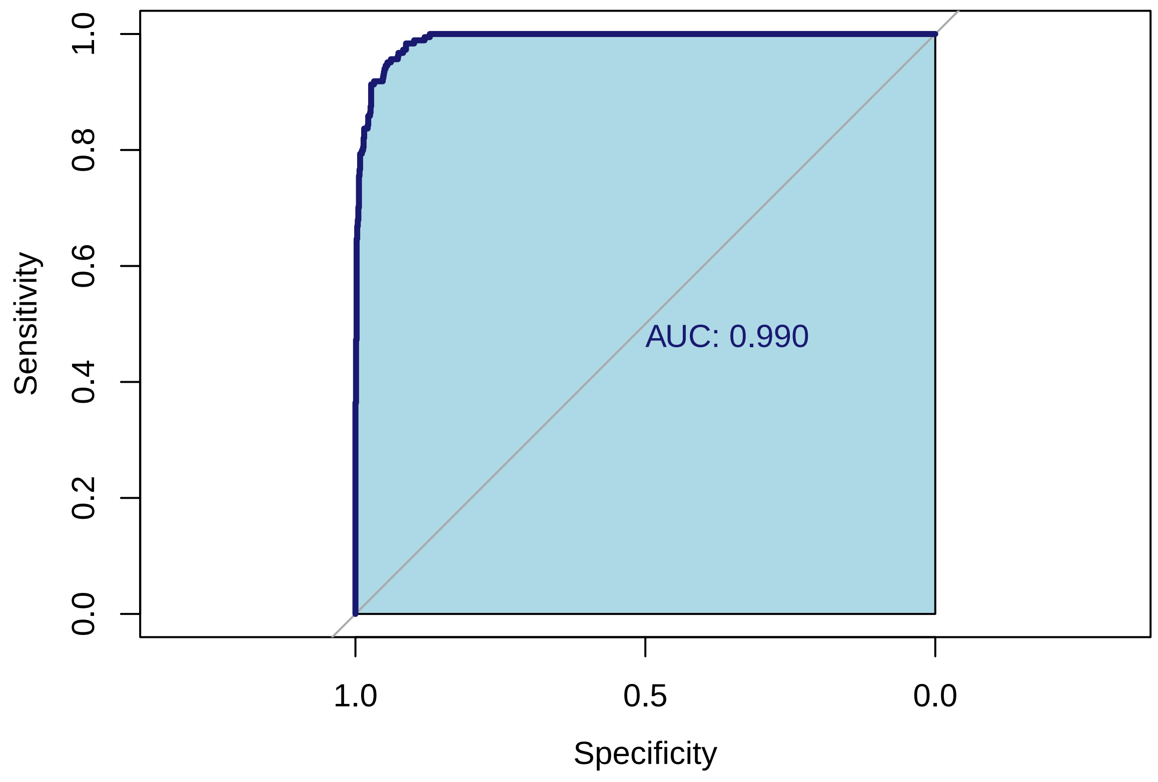
```
# ROC curve for the training set
roc_xgb <- pROC::roc(d_train$Closed_Account,
                    train_pred_xgb,
                    plot = TRUE,
                    col = "midnightblue",
                    lwd = 3,
                    auc.polygon = T,
                    auc.polygon.col = "lightblue",
                    print.auc = T)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```
# ROC curve for the validation set
roc_xgb_val <- pROC::roc(d_val$Closed_Account,
                         xgb_preds,
                         plot = TRUE,
                         col = "midnightblue",
                         lwd = 3,
                         auc.polygon = T,
                         auc.polygon.col = "lightblue",
                         print.auc = T)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```r
auc(roc_xgb_val)
```

```
## Area under the curve: 0.9899
```

```r
# Transform predicted probabilities in a 0 1 variable
xgb_preds <-  as.numeric(xgb_preds > 0.5)

# Confusion matrix for the validation set
cm <- confusionMatrix(as.factor(xgb_preds), d_val$Closed_Account)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 976  27
##          1  22 157
##
##                Accuracy : 0.9585
##                  95% CI : (0.9456, 0.9692)
##     No Information Rate : 0.8443
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8405
##
##  Mcnemar's Test P-Value : 0.5677
##
##             Sensitivity : 0.9780
##             Specificity : 0.8533
##          Pos Pred Value : 0.9731
##          Neg Pred Value : 0.8771
##              Prevalence : 0.8443
##          Detection Rate : 0.8257
##    Detection Prevalence : 0.8486
##       Balanced Accuracy : 0.9156
##
##        'Positive' Class : 0
##
```

Using a combination of models allows for a more comprehensive understanding of the dataset. Ensemble methods or model stacking could further improve prediction performance by leveraging the strengths of each model.

We notice that the AUC for validation set is the highest for XgBoost of 0.990 (and 0.999 for training set!). Additionally, XgBoost also has the highest accuracy of approximately 0.96. Therefore, *XgBoost is our best model*.

Now, we apply the XgBoost model on the test set. We also make all the adjustments we made to the original validation set to ensure that our model works in the same manner as the training and validation set, ensuring appropriate and maximal performance.

```r
# Training set
training_data <- rbind(x_train, x_val)

y_train <- as.data.frame(y_train)
colnames(y_train)[1] <- "Closed_Account"
y_val <- as.data.frame(y_val)
colnames(y_val)[1] <- "Closed_Account"

y_training <- as.matrix(rbind(y_train, y_val))

# Importing test set
test_data = read.csv("/Users/janvigoje/Desktop/bank_accounts_test (1).csv",
                     sep = ",",
                     dec = ".",
                     header = T,
                     colClasses = "character")


test_data[test_data == "Unknown"] <- NA            # Replace particular value with NA
anyNA(test_data)
```

```
## [1] TRUE
```

```r
test_data <- na.omit(test_data)
anyNA(test_data)
```

```
## [1] FALSE
```

```r
cols <- colnames(test_data)
categ = c("Education_Level", "Marital_Status", "Card_Category","Gender")

# Turning categorical variables into factors
for(i in 1:ncol(test_data)){
  if (cols[i] %in% categ){
    test_data[,i] = as.factor(test_data[,i])
  }
}

# Turning numerical strings to numerical types
for(i in 1:ncol(test_data)){
  if (!(cols[i] %in% categ)){
    test_data[,i] = as.numeric(test_data[,i])
  }
}

test_data <- test_data[,!names(test_data) %in% c("Avg_Open_To_Buy", "CLIENTNUM")]
```

```
# Encoding categorical variables
dmy <- dummyVars(" ~ .", data = test_data)
test_data <- test_data[,names(test_data) %in% colnames(training_data)]

# Adding the extracted variables also to the test set
Tot_Rev_Bal_ratio <- as.numeric(as.character(test_data$Total_Revolving_Bal)) / as.numeric(as.character(t
Avg_Trans_Amt <- as.numeric(as.character(test_data$Total_Trans_Amt)) / as.numeric(as.character(test_data

test_data <- cbind(test_data, Tot_Rev_Bal_ratio)
test_data <- cbind(test_data, Avg_Trans_Amt)


test_data <- as.data.frame(test_data)

head(test_data)
```

```
##    Months_on_book Total_Relationship_Count Months_Inactive_12_mon
## 1              39                        5                      1
## 4              36                        3                      6
## 5              36                        6                      2
## 6              37                        6                      1
## 7              33                        4                      2
## 10             36                        4                      1
##    Contacts_Count_12_mon Credit_Limit Total_Revolving_Bal Total_Amt_Chng_Q4_Q1
## 1                      3        12691                 777                 1335
## 4                      0        11751                   0                 3397
## 5                      3        30367                2362                 1708
## 6                      2        14470                1157                  966
## 7                      1         4470                 680                 1608
## 10                     2         8923                2517                 1726
##    Total_Trans_Amt Total_Trans_Ct Total_Ct_Chng_Q4_Q1 Avg_Utilization_Ratio
## 1             1144             42                1625                    61
## 4             1539             17                 325                     0
## 5             1671             27                 929                    78
## 6             1207             21                 909                     8
## 7              931             18                1571                   152
## 10            1589             24                1667                   282
##       Income Tot_Rev_Bal_ratio Avg_Trans_Amt
## 1   77.51107        0.06122449      27.23810
## 4  119.59708        0.00000000      90.52941
## 5  106.73974        0.07778180      61.88889
## 6  149.17876        0.07995853      57.47619
## 7   48.59461        0.15212528      51.72222
## 10  74.02609        0.28208002      66.20833
```

```
set.seed(606)

# The model
xgb_train <- xgb.DMatrix(data = as.matrix(training_data), label = as.numeric(as.character(y_training)))
xgb_test <- xgb.DMatrix(data = as.matrix(test_data))

# List of parameters of our model
xgb_params <- list(
  booster = "gbtree",
  eta = 0.01,
  max_depth = 8,
  gamma = 4,
  subsample = 0.75,
  colsample_bytree = 1,
  objective = "binary:logistic",
  eval_metric = "auc"
)

xgb_model <- xgb.train(
  params = xgb_params,
  data = xgb_train,
  nrounds = 5000,
  verbose = 1
)

xgb_preds <- predict(xgb_model, xgb_test)
write.csv(xgb_preds, "myprob.csv", row.names = F)
```

Cost Matrix (Q6)

To get the cost matrix we consider all the possible cases:

1. Prediction: 0, Outcome: 0 In this case, the customer doesn't close his account without offering him highly competitive interest rates, so the gain is 50, which will be represented by -50 in the matrix since it's a cost matrix, so costs[1,1] = -50
2. Prediction: 0, Outcome: 1 In this case, the customer closes his account since we didn't offer him highly competitive interest rates, so the loss is 50, therefore costs[1,2] = 50
3. In both cases in which we predict 1, the bank offers to the customer highly competitive interest rates, by doing so the bank loses 20 but we are sure that the customer doesn't close his account so the bank gains 50. Overall the bank gains 20 in both cases. Therefore costs[2,1] = costs[2,2] = -20

```
#Cost matrix

costs <- matrix(c(-50, -20, 50, -20), 2)
colnames(costs) = rownames(costs) = levels(Data$Closed_Account)
costs
```

```
##     0   1
## 0 -50  50
## 1 -20 -20
```

The only case in which the bank loses is when we predict that a client won't close his account but he actually does in reality (false negative). Therefore, to minimize costs, we should find the threshold such that the expectation of the loss function. Since the only loss we get is when we have a false negative, and we should predict 0 when the outcome is 0 to make more profits.

A good way to minimize the expectation of the loss function is to always have the cost of predicting 0 smaller than the cost of predicting 1 costs[1,1].(1-p) + costs[1,2].p < costs[2,1].(1-p) + costs[2,2].p where p is the probability of the client to close the account.

```
threshold <- (costs[2,1] - costs[1,1]) / (costs[1,2] - costs[1,1] + costs[2,1] - costs[2,2])
threshold
```

```
## [1] 0.3
```

By solving this inequality, we get p < 0.3. Therefore, the optimal threshold to minimize costs is t = 0.3

```
default_t <- 0.5 # Threshold used in the model
opt_t <- threshold  # Optimal threshold for cost minimization

# We will compare between the two on the validation set since we don't have the actual outcomes of the
xgb_preds_default_t <-  as.numeric(predict(xgb_model, xgb_val) > default_t) # Predictions using the def
xgb_preds_opt_t <-  as.numeric(predict(xgb_model, xgb_val) > opt_t) # Predictions using the optimal thr

# Prediction costs using the default threshold
conf_matrix_default_t <- confusionMatrix(as.factor(xgb_preds_default_t), d_val$Closed_Account)
conf_matrix_default_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 993   6
##          1   5 178
##
##                Accuracy : 0.9907
##                  95% CI : (0.9834, 0.9953)
##     No Information Rate : 0.8443
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9645
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9950
##             Specificity : 0.9674
##          Pos Pred Value : 0.9940
##          Neg Pred Value : 0.9727
##              Prevalence : 0.8443
##          Detection Rate : 0.8401
##    Detection Prevalence : 0.8452
##       Balanced Accuracy : 0.9812
##
##        'Positive' Class : 0
##
```

```
costs_default_t <- conf_matrix_default_t$table[1,1]*costs[1,1] + conf_matrix_default_t$table[1,2]*costs
  conf_matrix_default_t$table[2,2]*costs[2,2]
costs_default_t
```

```
## [1] -53010
```

```
# Prediction costs using the optimal threshold
conf_matrix_opt_t <- confusionMatrix(as.factor(xgb_preds_opt_t), d_val$Closed_Account)
conf_matrix_opt_t
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 985   2
##          1  13 182
##
##                Accuracy : 0.9873
##                  95% CI : (0.9792, 0.9929)
##     No Information Rate : 0.8443
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9529
##
##  Mcnemar's Test P-Value : 0.009823
##
##             Sensitivity : 0.9870
##             Specificity : 0.9891
##          Pos Pred Value : 0.9980
##          Neg Pred Value : 0.9333
##              Prevalence : 0.8443
##          Detection Rate : 0.8333
##    Detection Prevalence : 0.8350
##       Balanced Accuracy : 0.9881
##
##        'Positive' Class : 0
##
```

```
costs_opt_t <- conf_matrix_opt_t$table[1,1]*costs[1,1] + conf_matrix_opt_t$table[1,2]*costs[1,2] + conf
  conf_matrix_opt_t$table[2,2]*costs[2,2]
costs_opt_t
```

```
## [1] -53050
```

```
# Difference in costs
diff <- costs_opt_t - costs_default_t
diff
```

```
## [1] -40
```

We notice that using the optimal threshold we get a lower cost. Using the optimal threshold, the model is a bit less accurate, we also noticed from the confusion matrix that we have less false negatives without penalizing the prediction of true negatives that generate the most profits, which is what we needed to minimize the costs.

Comparing the two thresholds, the optimal threshold (0.3) resulted in a better financial outcome, with a cost of -54070 compared to -53610 for the default threshold (0.5). This is evidenced by the reduction in false negatives, which are the most costly misclassification in this scenario. Although there may be a slight decrease in overall accuracy, the optimal threshold better aligns with the financial objectives by effectively reducing the most costly type of error and not significantly impacting the correct predictions of non-closure, which are the most profitable outcomes. The difference in cost (-460) indicates a substantial saving by using the optimal threshold. This illustrates the importance of considering business costs and outcomes when determining the best operational point for a classification model, rather than relying solely on traditional metrics like accuracy.

End