Data Analysis for Business - Midterm Project

Standard Deviated Group

Group members:

- Edoardo Cocciò - 282401
- Alexandra Tabarani - 282091
- Simone Filosofi - 284531
- Marta Torella - 284091

EDA

The banking industry operates in a highly competitive environment, where customer retention is crucial for sustained business success. In this context, our project aims to delve into a dataset containing detailed information about bank customers to predict the likelihood of an account closure.

To begin our exploratory data analysis (EDA), we imported the necessary libraries and we loaded the dataset titled `bank_accounts_train.csv` into the R environment.

```
data <- read.csv("bank_accounts_train.csv", header = T, stringsAsFactors = T)
data.test <- read.csv("bank_accounts_test.csv", header =T, stringsAsFactors = T)
```

We first check the structure of the data to get an overall feeling of the respective types and values.

```
str(data)
```

```
## 'data.frame':    7597 obs. of  21 variables:
##  $ CLIENTNUM               : int  721038408 711968358 719174433 712841883 715980258 708394908 769152
##  $ Customer_Age            : int  38 55 36 50 36 44 60 37 55 60 ...
##  $ Gender                  : Factor w/ 2 levels "F","M": 1 1 1 2 1 2 1 1 1 2 ...
##  $ Dependent_count         : int  2 4 1 3 1 0 2 3 1 1 ...
##  $ Education_Level         : Factor w/ 7 levels "College","Doctorate",..: 4 3 7 1 2 4 4 4 7 4 ...
##  $ Marital_Status          : Factor w/ 4 levels "Divorced","Married",..: 2 3 3 3 3 3 3 2 2 2 ...
##  $ Card_Category           : Factor w/ 4 levels "Blue","Gold",..: 1 1 1 4 1 1 1 1 1 1 ...
##  $ Months_on_book          : int  27 44 24 36 25 33 54 19 43 55 ...
##  $ Total_Relationship_Count: int  4 1 2 3 1 5 1 5 2 5 ...
##  $ Months_Inactive_12_mon  : int  1 2 1 3 2 1 6 1 1 1 ...
##  $ Contacts_Count_12_mon   : int  2 1 2 2 3 3 2 3 2 4 ...
##  $ Credit_Limit            : int  1830 2638 1735 34516 2835 2337 8141 4138 3391 2661 ...
##  $ Total_Revolving_Bal     : int  0 1423 0 2517 2511 1442 1811 0 1696 1466 ...
##  $ Avg_Open_To_Buy         : int  1830 1215 1735 31999 324 895 6330 4138 1695 1195 ...
##  $ Total_Amt_Chng_Q4_Q1    : int  736 551 74 735 731 659 1043 1168 537 631 ...
##  $ Total_Trans_Amt         : int  1741 4153 2467 7155 4811 3945 5043 2420 4291 1354 ...
##  $ Total_Trans_Ct          : int  43 77 35 66 77 65 53 50 77 25 ...
```

```
##  $ Total_Ct_Chng_Q4_Q1     : int  654 925 346 65 925 548 432 613 711 316 ...
##  $ Avg_Utilization_Ratio   : int  0 539 0 73 886 617 222 0 5 551 ...
##  $ Income                  : num  169.8 87 140.3 90.2 101.6 ...
##  $ Closed_Account          : int  0 0 1 1 0 0 1 0 0 0 ...
```

We see that there are 7597 rows and we have 21 features in our data. These features include customer demographic information, account characteristics, and transaction history. Notably, we have factors such as Gender, Education Level, Marital Status, and Card Category, as well as a range of numerical features like Customer Age, Total Relationship Count, Credit Limit, and various transaction-related variables.

```
summary(data)
```

```
##    CLIENTNUM          Customer_Age   Gender   Dependent_count
##  Min.   :708082083   Min.   :26.00   F:4037   Min.   :0.000
##  1st Qu.:713030958   1st Qu.:41.00   M:3560   1st Qu.:1.000
##  Median :717908208   Median :46.00            Median :2.000
##  Mean   :738866129   Mean   :46.27            Mean   :2.354
##  3rd Qu.:772601283   3rd Qu.:52.00            3rd Qu.:3.000
##  Max.   :828343083   Max.   :73.00            Max.   :5.000
##
##       Education_Level  Marital_Status  Card_Category  Months_on_book
##  College      : 748   Divorced: 555   Blue    :7080   Min.   :13.00
##  Doctorate    : 331   Married :3540   Gold    :  88   1st Qu.:31.00
##  Graduate     :2384   Single  :2937   Platinum:  16   Median :36.00
##  High School  :1506   Unknown : 565   Silver  : 413   Mean   :35.86
##  Post-Graduate: 401                                   3rd Qu.:40.00
##  Uneducated   :1100                                   Max.   :56.00
##  Unknown      :1127
##  Total_Relationship_Count Months_Inactive_12_mon Contacts_Count_12_mon
##  Min.   :1.000            Min.   :0.000          Min.   :0.000
##  1st Qu.:3.000            1st Qu.:2.000          1st Qu.:2.000
##  Median :4.000            Median :2.000          Median :3.000
##  Mean   :3.808            Mean   :2.332          Mean   :2.469
##  3rd Qu.:5.000            3rd Qu.:3.000          3rd Qu.:3.000
##  Max.   :6.000            Max.   :6.000          Max.   :6.000
##
##   Credit_Limit   Total_Revolving_Bal Avg_Open_To_Buy Total_Amt_Chng_Q4_Q1
##  Min.   : 1439   Min.   :   0        Min.   :   14   Min.   :   0.0
##  1st Qu.: 2765   1st Qu.: 289        1st Qu.: 1462   1st Qu.: 579.0
##  Median : 5346   Median :1265        Median : 4199   Median : 715.0
##  Mean   : 9253   Mean   :1155        Mean   : 7972   Mean   : 691.7
##  3rd Qu.:13477   3rd Qu.:1776        3rd Qu.:11423   3rd Qu.: 846.0
##  Max.   :34516   Max.   :2517        Max.   :34516   Max.   :3355.0
##
##  Total_Trans_Amt Total_Trans_Ct   Total_Ct_Chng_Q4_Q1 Avg_Utilization_Ratio
##  Min.   :  510   Min.   : 10.00   Min.   :   0.0      Min.   :  0
##  1st Qu.: 2143   1st Qu.: 45.00   1st Qu.: 444.0      1st Qu.:  3
##  Median : 3895   Median : 67.00   Median : 656.0      Median :132
##  Mean   : 4404   Mean   : 64.76   Mean   : 593.7      Mean   :249
##  3rd Qu.: 4742   3rd Qu.: 81.00   3rd Qu.: 786.0      3rd Qu.:463
##  Max.   :18484   Max.   :139.00   Max.   :3714.0      Max.   :994
##
##      Income          Closed_Account
```

```
##  Min.   :  0.01425   Min.   :0.0000
##  1st Qu.: 53.86291   1st Qu.:0.0000
##  Median : 79.61163   Median :0.0000
##  Mean   : 91.00471   Mean   :0.1602
##  3rd Qu.:121.90419   3rd Qu.:0.0000
##  Max.   :199.94190   Max.   :1.0000
##
```

Using the `summary()` function, we obtained a snapshot of the central tendency and dispersion of the numerical features and the distribution of the categorical variables:

- **Client Number (`CLIENTNUM`)**: Appears to be a unique identifier for each customer with a wide range.
- **Customer Age**: Age ranges from 26 to 73, with a median of 46, indicating a middle-aged customer base.
- **Gender**: More female (F:4037) customers than male (M:3560), though not by a wide margin.
- **Dependent Count**: Ranges from 0 to 5, with most customers having up to 3 dependents.
- **Education Level**: A diverse educational background is evident, with a notable number of customers with unknown education levels (1127).
- **Marital Status**: Majority are married (3540), followed by single (297) and divorced (555), with some unknowns (565).
- **Card Category**: Overwhelmingly, customers have a Blue card (7080), with far fewer Silver (413), Gold (88), or Platinum (16) cards.
- **Months on Book**: Customers have been with the bank for a period ranging from 13 to 56 months.
- **Total Relationship Count**: Customers have between 1 to 6 products with the bank.
- **Months Inactive 12 mon**: A variable indicating customer engagement, varying between 0 to 6 months.
- **Contacts Count 12 mon**: Reflects customer interaction with the bank, ranging from 0 to 6 in the last year.
- **Credit Limit**: Shows a wide range from 1439 to 34516, highlighting the variety in customers' credit situations.
- **Total Revolving Bal**: Ranges from 0 to 2517, suggesting varying usage of available credit.
- **Avg Open To Buy**: Considerable variation is observed, suggesting differences in the unused credit line.
- **Total Amt Chng Q4 Q1**: Indicates the change in transaction amount over time.
- **Total Trans Amt** and **Total Trans Ct**: High variation in both total transaction amount and count, essential for understanding customer activity.
- **Total Ct Chng Q4 Q1**: Some customers show significant changes in their transaction count over time.
- **Avg Utilization Ratio**: Spans from 0 to 0.994, which will be significant in understanding how customers are utilizing their credit.
- **Income**: Ranges widely from 0.01425 to 199.94190 (in thousands of dollars), indicating a diverse economic background among customers.
- **Closed Account**: A binary variable where most accounts are open (0), with a proportion closed (1).

The summary statistics provide a comprehensive overview of the dataset, offering insights into the distribution of variables and the characteristics of bank customers.

Upon examining the structure of our dataset, we observed that `CLIENTNUM` serves solely as an identifier and can thus be safely removed from further analysis. Additionally, we converted the 'Closed_Account' variable to a factor to facilitate subsequent modeling and visualization.

```
data <- within(data, rm("CLIENTNUM"))
data$Closed_Account <- as.factor(data$Closed_Account)
```

A significant number of categorical variables are marked as "Unknown", which we will need to decide how to handle—whether to treat as missing and impute or remove.

```
sapply(data, function(x) any(x %in% "Unknown"))
```

```
##              Customer_Age                    Gender          Dependent_count
##                     FALSE                     FALSE                     FALSE
##           Education_Level            Marital_Status             Card_Category
##                      TRUE                      TRUE                     FALSE
##            Months_on_book  Total_Relationship_Count    Months_Inactive_12_mon
##                     FALSE                     FALSE                     FALSE
##     Contacts_Count_12_mon              Credit_Limit       Total_Revolving_Bal
##                     FALSE                     FALSE                     FALSE
##           Avg_Open_To_Buy      Total_Amt_Chng_Q4_Q1           Total_Trans_Amt
##                     FALSE                     FALSE                     FALSE
##            Total_Trans_Ct       Total_Ct_Chng_Q4_Q1     Avg_Utilization_Ratio
##                     FALSE                     FALSE                     FALSE
##                    Income            Closed_Account
##                     FALSE                     FALSE
```

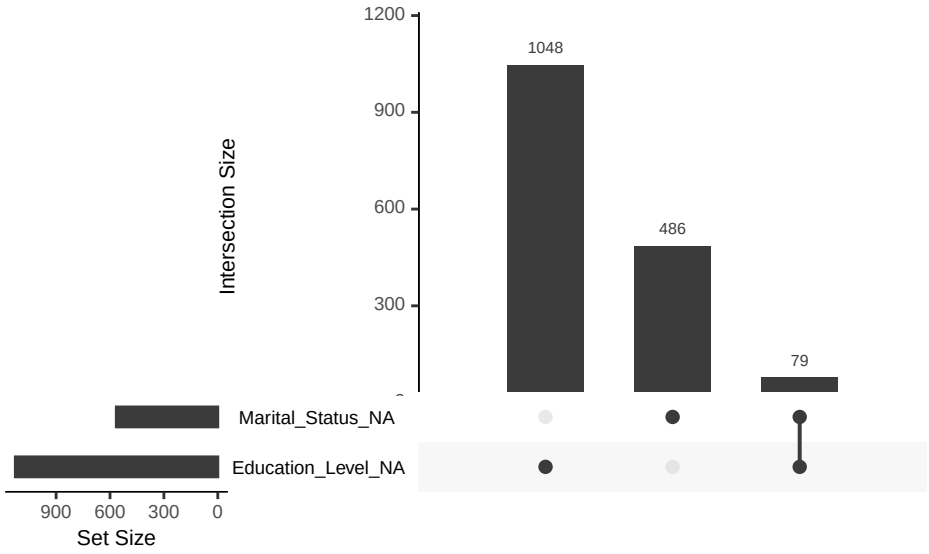
We see that the columns `Education_Level` and `Marital_Status` both contain unknown values, which we will replace using the appropriate datatype `NA`. This conversion allows us to treat these entries appropriately during analysis, ensuring that our statistical summaries and models handle them as missing rather than as a category of their own.

Handling NA Values

```
data$Education_Level[data$Education_Level == "Unknown"] <- NA
data$Marital_Status[data$Marital_Status == "Unknown"] <- NA

sum(is.na(data))
```

```
## [1] 1692
```

```
naniar::gg_miss_upset(data)
```



Thanks to this handy visualization we can see that `Marital Status` contains 1048 missing values and `Education_Level` 486. In 79 rows both of those columns will contain missing values. Moreover, we can see that our dataset contains 1613 missing values in total. The problem is that those two values, as previously inspected, are both part of categorical variables and we decided to avoid imputing them using the most frequent value as we risk to introduce bias in our model. An alternative approach that we tried without success was to build a model to predict the missing values, but the results were not satisfactory. Therefore, we decided to **remove the missing rows** from our dataset.

```
data <- na.omit(data)
sum(apply(data, 1, anyNA))
```

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

We'll continue by checking if there are any duplicates in this dataset and eventually remove them to ensure data integrity and accuracy of our analysis.

```
all_duplicates <- duplicated(data) | duplicated(data, fromLast = TRUE)
data[all_duplicates, ]
```

```
##      Customer_Age Gender Dependent_count Education_Level Marital_Status
## 1              38      F               2     High School        Married
## 2              55      F               4        Graduate         Single
## 7596           38      F               2     High School        Married
## 7597           55      F               4        Graduate         Single
##      Card_Category Months_on_book Total_Relationship_Count
## 1             Blue             27                        4
## 2             Blue             44                        1
## 7596          Blue             27                        4
## 7597          Blue             44                        1
##      Months_Inactive_12_mon Contacts_Count_12_mon Credit_Limit
## 1                         1                     2         1830
## 2                         2                     1         2638
## 7596                      1                     2         1830
## 7597                      2                     1         2638
##      Total_Revolving_Bal Avg_Open_To_Buy Total_Amt_Chng_Q4_Q1 Total_Trans_Amt
## 1                      0            1830                  736            1741
## 2                   1423            1215                  551            4153
## 7596                   0            1830                  736            1741
## 7597                1423            1215                  551            4153
##      Total_Trans_Ct Total_Ct_Chng_Q4_Q1 Avg_Utilization_Ratio   Income
## 1                43                 654                     0 169.7797
## 2                77                 925                   539  87.0297
## 7596             43                 654                     0 169.7797
## 7597             77                 925                   539  87.0297
##      Closed_Account
## 1                0
## 2                0
## 7596             0
## 7597             0
```

Upon inspection, we found that the two latter rows (7596 and 7597) are duplicates of the first and the second lines, respectively. Therefore, we'll proceed by eliminating them from the dataset:

```
data <- data[-c(7596, 7597), ]
```

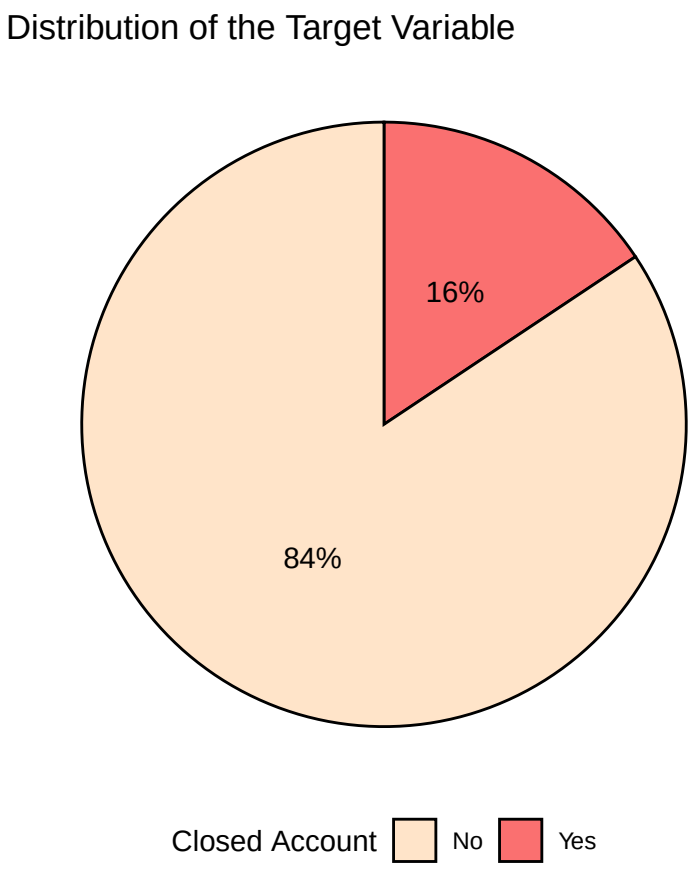
We can now initiate our analysis by dividing the dataset into continuous and categorical variables. This segmentation facilitates a focused examination of the distinct characteristics inherent in our data. Following this initial step, we delve into a comprehensive evaluation of the dataset's unique features. The subsequent code will facilitate the calculation of the count of distinct values for each feature, providing valuable insights into the breadth and diversity of our dataset.

```
continuous_vars <- data[, sapply(data, is.numeric)]
categorical_vars <- data[, !names(data) %in% names(continuous_vars)]
```

This step ensures that our dataset contains only unique observations, which is essential for maintaining the reliability of our analytical processes.

Let's now start analyzing and visualizing patterns and relationships among our variables. Understanding these dynamics can offer us insights into the behavior of account holders and the factors influencing their decisions to continue or discontinue their services.

As a starting point, we will examine the proportion of accounts that have been closed within our dataset. In this context, an account marked with a '0' signifies that it remains open and active, whereas a '1' indicates an account that has been closed.

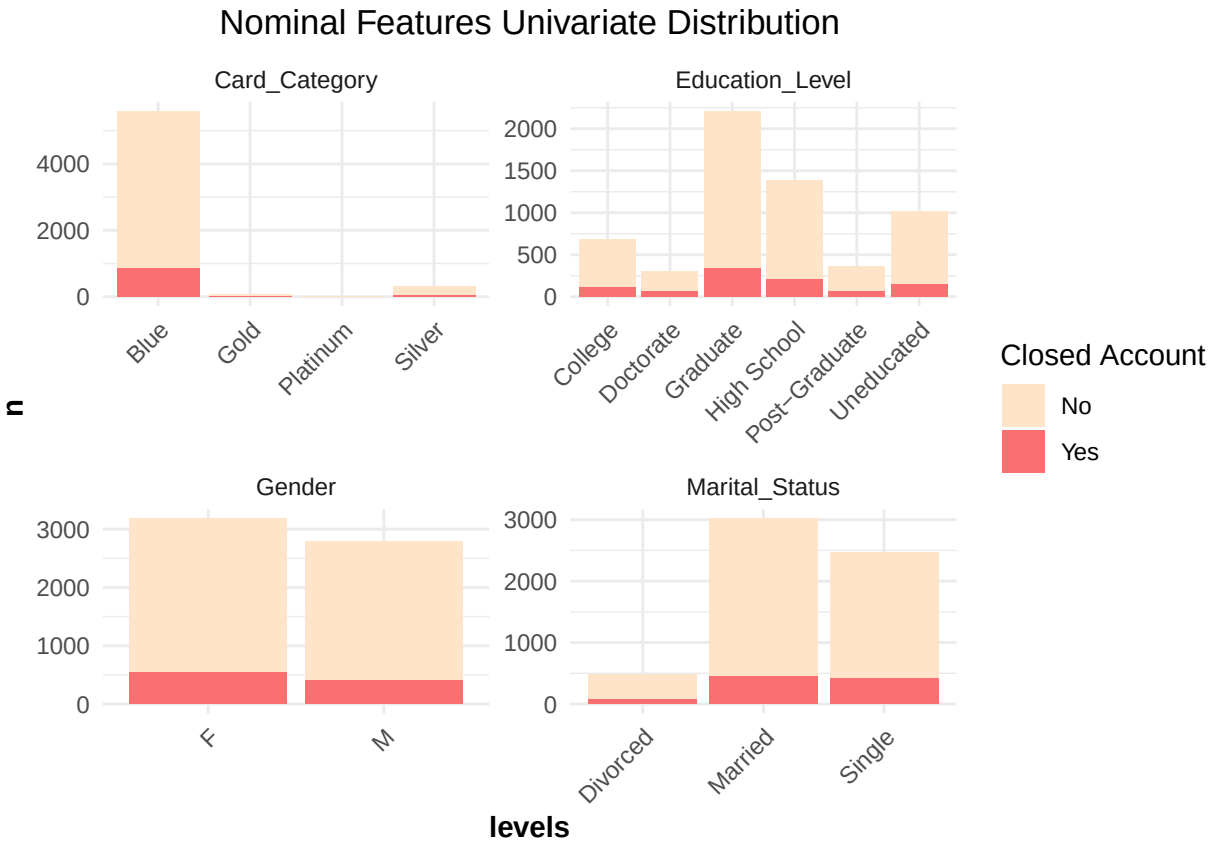


The barplot illustrates the distribution of closed versus active accounts, revealing an unbalanced proportion where a significant majority, 83.98%, of accounts remain open (denoted as '0'), while a smaller fraction, 16.02%, are marked as closed (denoted as '1'). This imbalance in the dataset underscores a lower churn rate and effectively emphasizes the disparity in account status, with most customers maintaining active accounts.

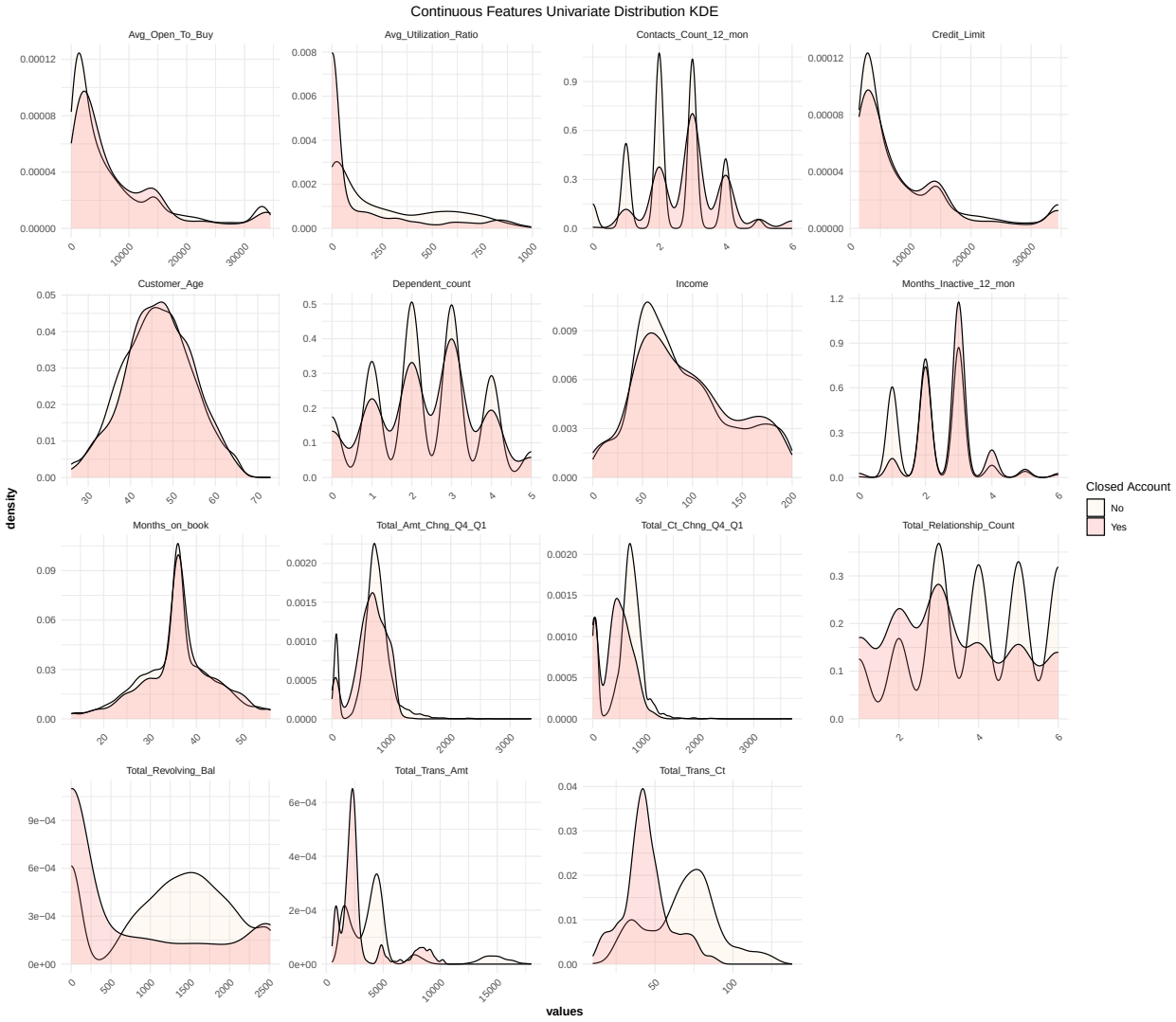
Univariate Analysis

Now is the time to conduct some univariate analysis between our variables and our target variable `closed_account` to assess their individual distributions and explore any potential patterns or trends that may exist, a crucial step in our analysis to improve customer retention strategies.

```
## `summarise()` has grouped output by 'Gender', 'Education_Level',
## 'Marital_Status', 'Card_Category'. You can override using the `.groups`
## argument.
## Scale for fill is already present. Adding another scale for fill, which will
## replace the existing scale.
```



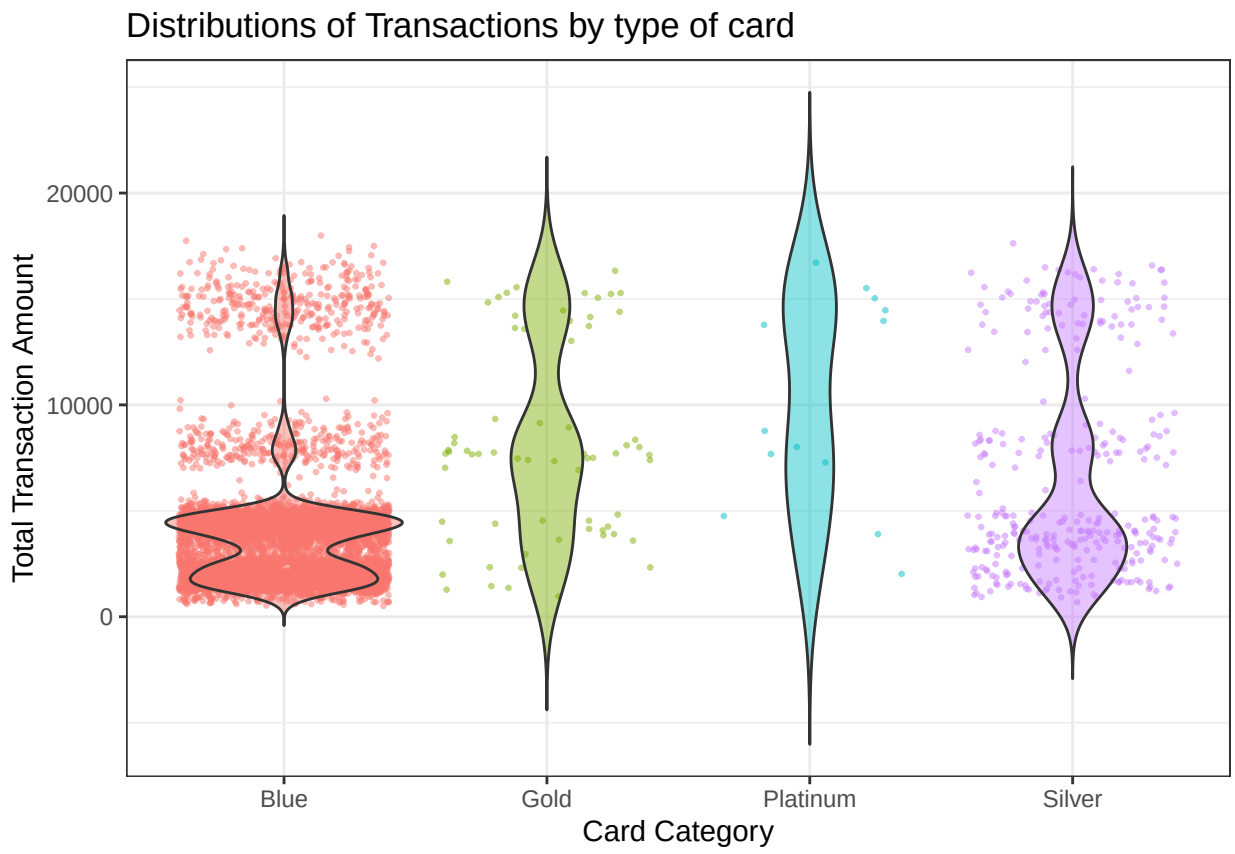
We start by analyzing the distribution of categorical variables in relation to the target variable 'Closed_Account'. The bar plots provide insights into the distribution of each category within the features, highlighting the proportion of closed and active accounts. This first univariate analysis conducted with sets of barplots shows that most accounts are associated with the 'Blue' card category and the 'Graduate' education level. The distribution of closed accounts is fairly uniform across different levels of card categories, education, gender, and marital status, with no immediately apparent trend indicating a strong association between these categorical features and the likelihood of account closure. We now proceed with the univariate analysis of continuous variables, where we'll use instead density plots to visualize the distribution of numerical features in relation to the target variable 'Closed_Account'.



The density plots show that the distributions of continuous variables for accounts that were closed and those that were not closed often overlap, suggesting no stark contrast between the two groups. However, lower values in `Credit_Limit`, `Total_Revolving_Bal`, `Total_Trans_Amt`, and `Total_Trans_Ct` are more common among closed accounts, which could imply that accounts with lower credit activity and balances are more likely to be closed. We can see that among those distributions there's a tendency to be **skewed to the right**, which could indicate that most of the data is concentrated on the lower values of the variables. We can also notice a well defined normal distribution in the `Customer_Age` variable.

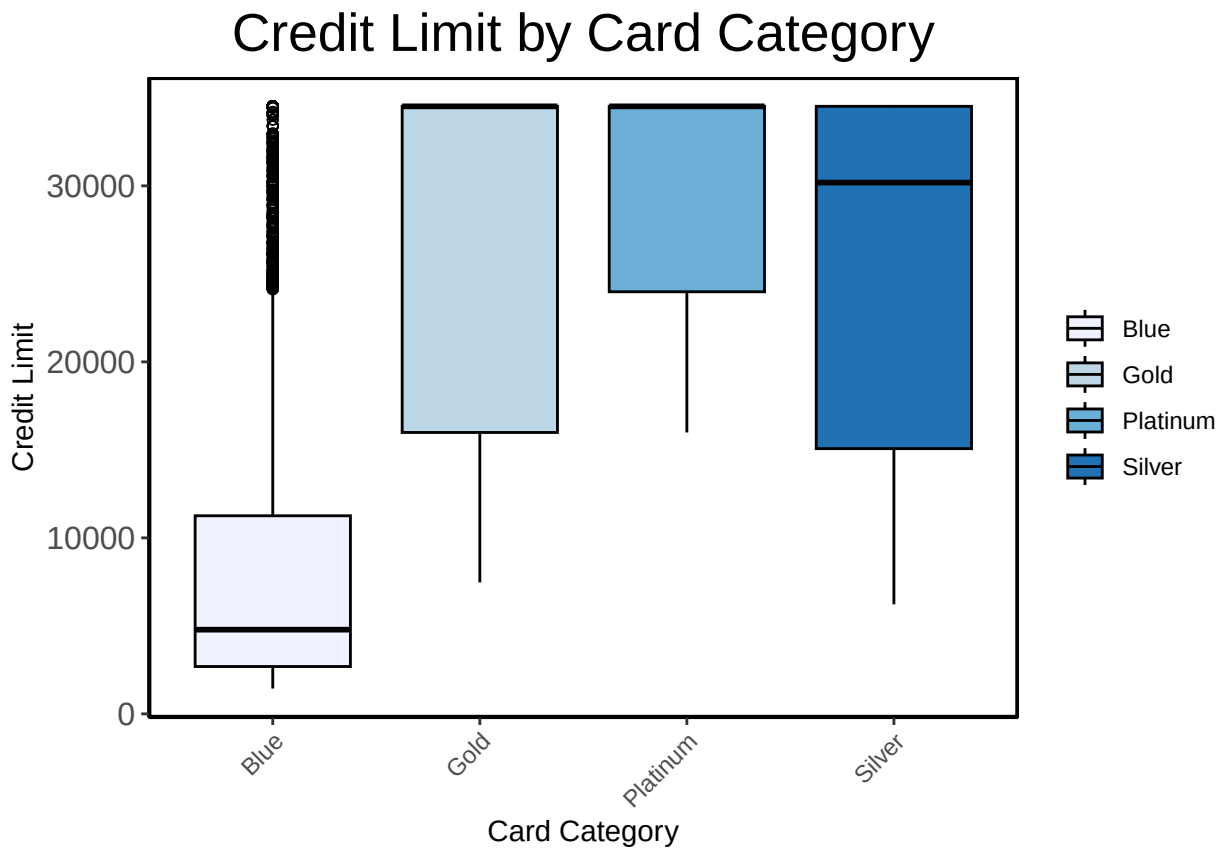
Bivariate analysis

We will now proceed with a bivariate analysis to explore the relationships between pairs of variables and their impact on account closure. We selected the features we believe could be most relevant to predicting account closure and will examine their interactions to identify potential patterns and correlations.

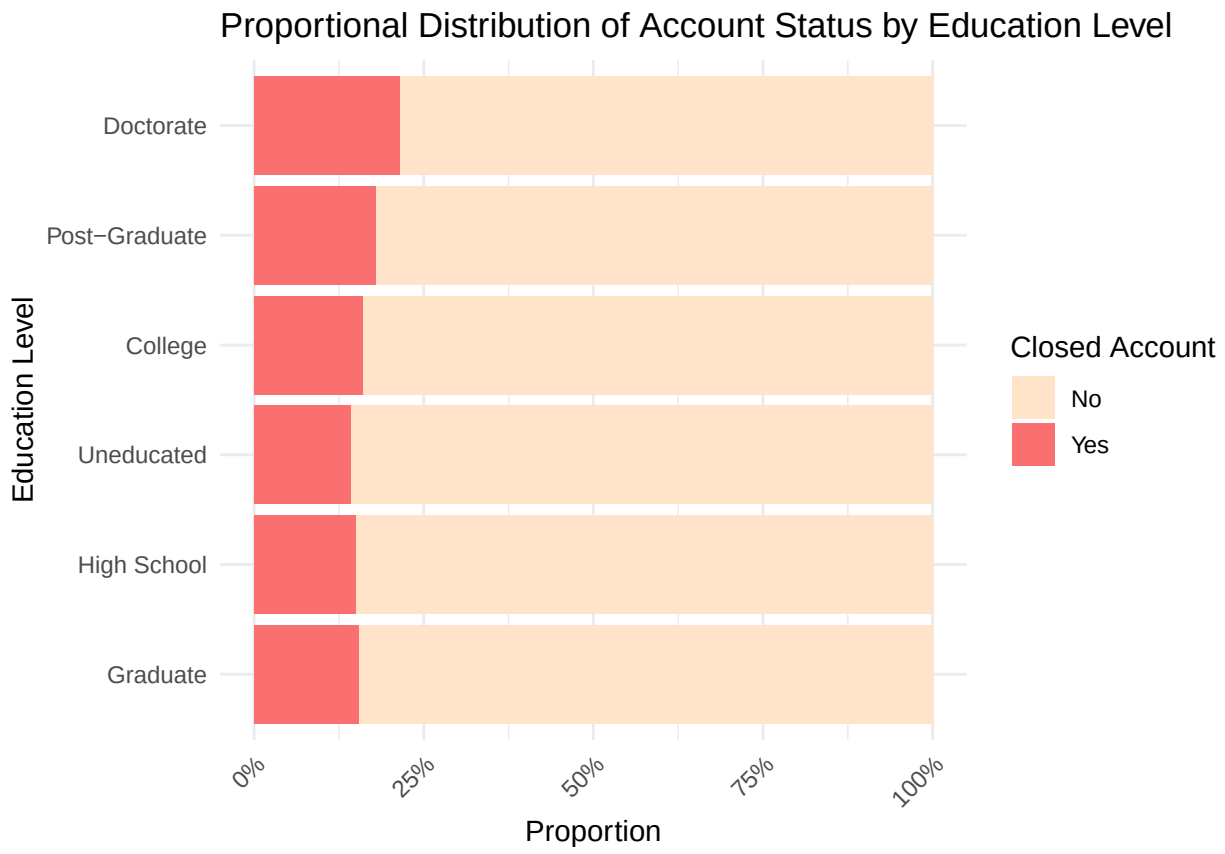


The first bivariate analysis examines the relationship between `Card_Category`, `Avg_Utilization_Ratio`, and `Total_Trans_Amt`; the plot aim is to provide a visual summary of how transaction amounts are distributed across different card categories, showing differences in variability and transaction behavior among cardholders of different categories. The Violin Plot shows the distribution of the `Total_Trans_Amt` for each `Card_Category` and its width at different amounts indicates the density of data points there; the Jitter Plot instead shows individual data points. The plot reveals that 'Platinum' and 'Silver' cardholders have a wider range of transaction amounts, indicating greater variability in their spending. 'Blue' cardholders tend to have smaller transactions more frequently. The 'Gold' category has fewer large transactions compared to 'Platinum'.

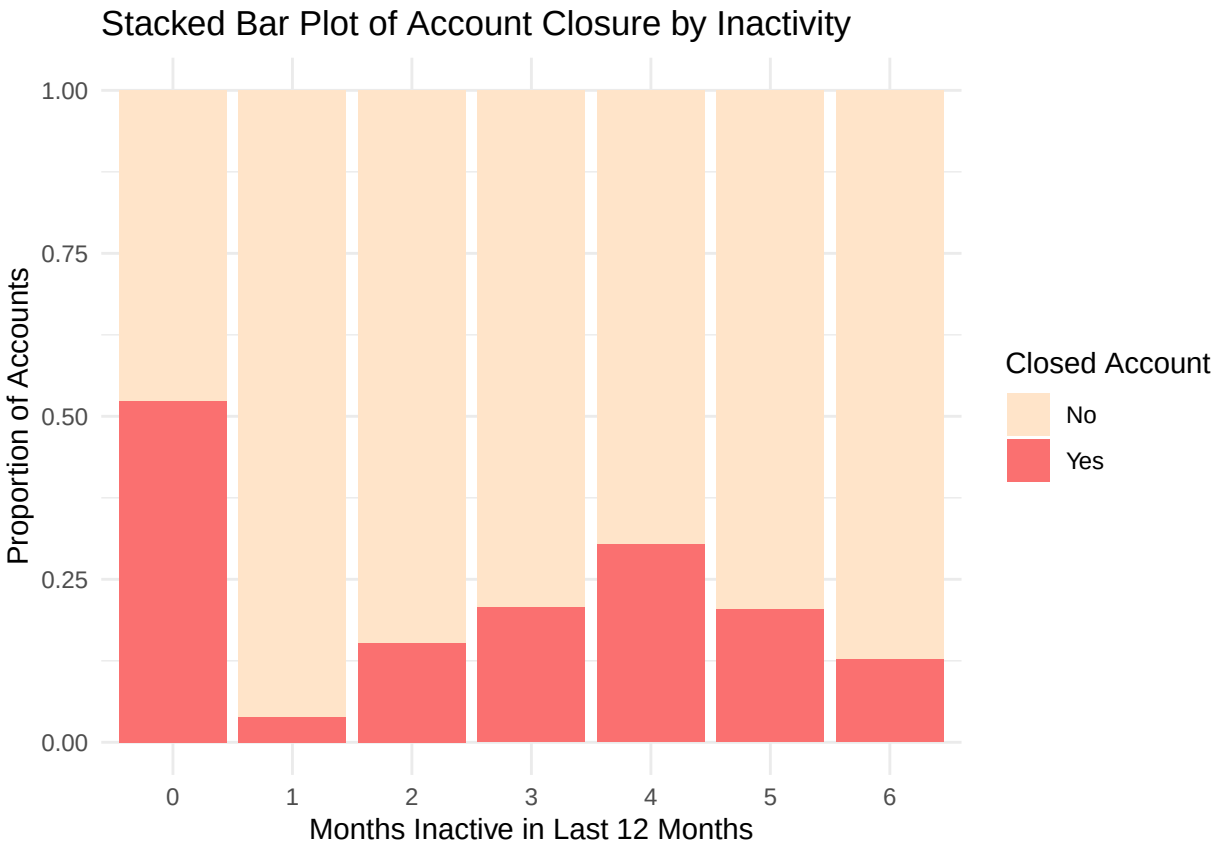
```
## Warning: The ‘size‘ argument of ‘element_rect()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



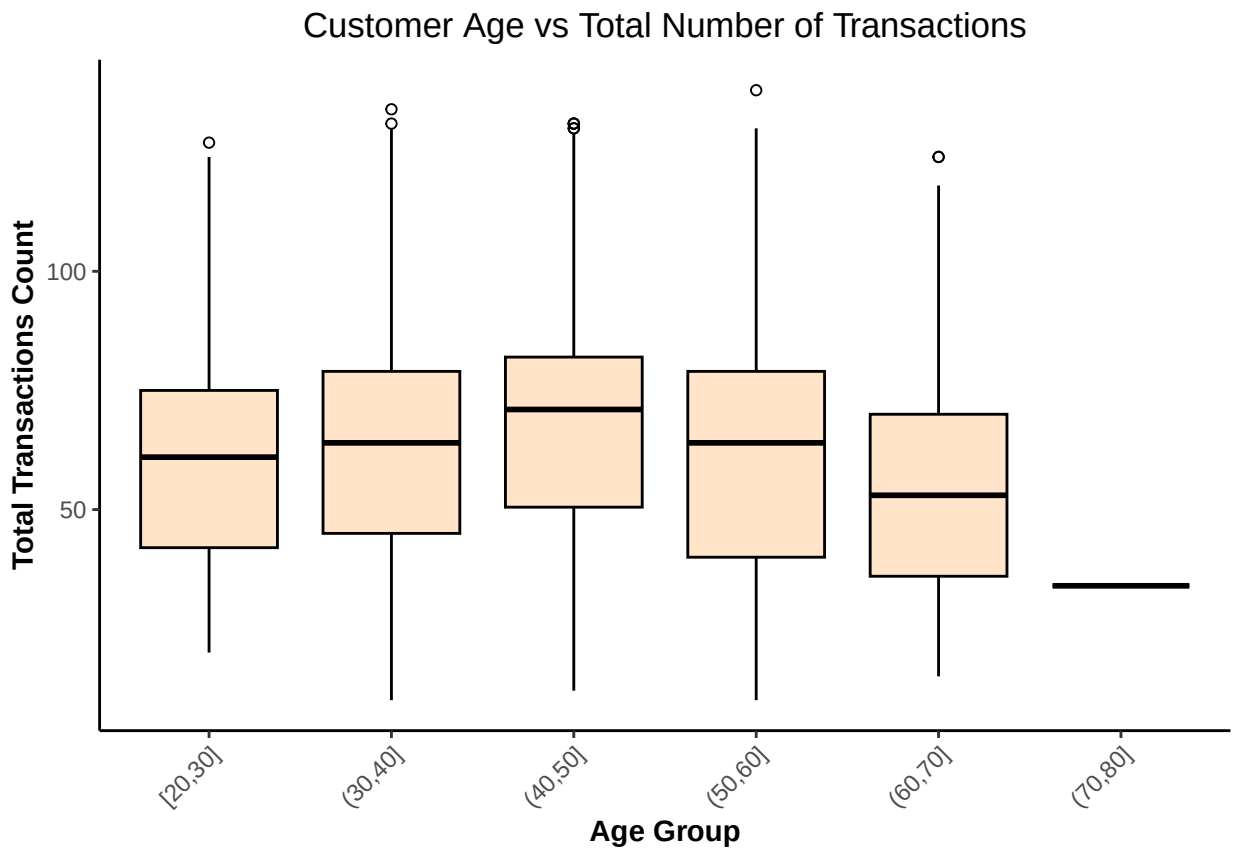
The second bivariate analysis explores the relationship between `Card_Category` and `Credit_Limit` to understand how credit limits are distributed across different card categories. The boxplot provides a visual summary of the distribution of credit limits for each card category, highlighting the median, quartiles, and potential outliers. The box plot shows that 'Platinum' and 'Silver' cards have higher median credit limits than 'Blue' and 'Gold', with 'Gold' and 'Platinum' showing a greater range in limits. 'Blue' cards have a few extreme outliers with very high credit limits.



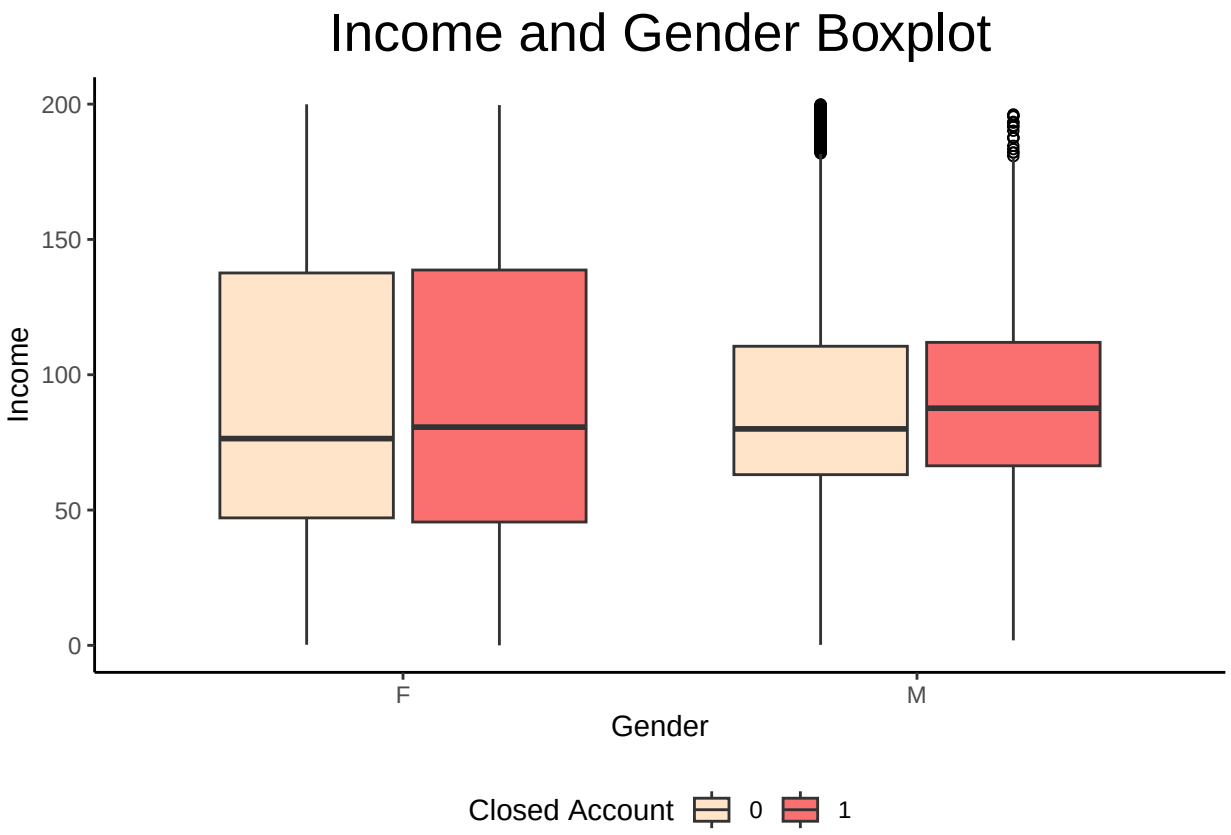
Here, we proceed with a bivariate analysis to explore the relationship between `Education_Level` and `Closed_Account`. The bar plot shows the proportional distribution of account status by education level, highlighting the proportion of closed and active accounts for each education category. The plot suggests that there might be a relationship between education level and the likelihood of an account being closed, with some level showing a higher or lower proportion of closures: indeed, the education levels 'Doctorate' and 'Post-Graduate' show a slightly higher proportion of closed accounts compared to other levels.



This bivariate analysis examines the relationship between `Months_Inactive_12_mon` and `Closed_Account`: this stacked bar chart provides a visual representation of the relationship between account inactivity (measured in months within the last 12 months) and account closure status (where '0' signifies an open account and '1' signifies a closed account). The bars represent the proportion of accounts that have been inactive for a certain number of months. The chart suggests a trend where accounts with zero and four months of inactivity are more likely to be closed, while accounts with one, two, three and six months of inactivity are more likely to remain open. This could indicate that customers who are either very active or with moderate activity levels are more likely to close their accounts, while those customers who are more inactive are more likely to keep their accounts open.

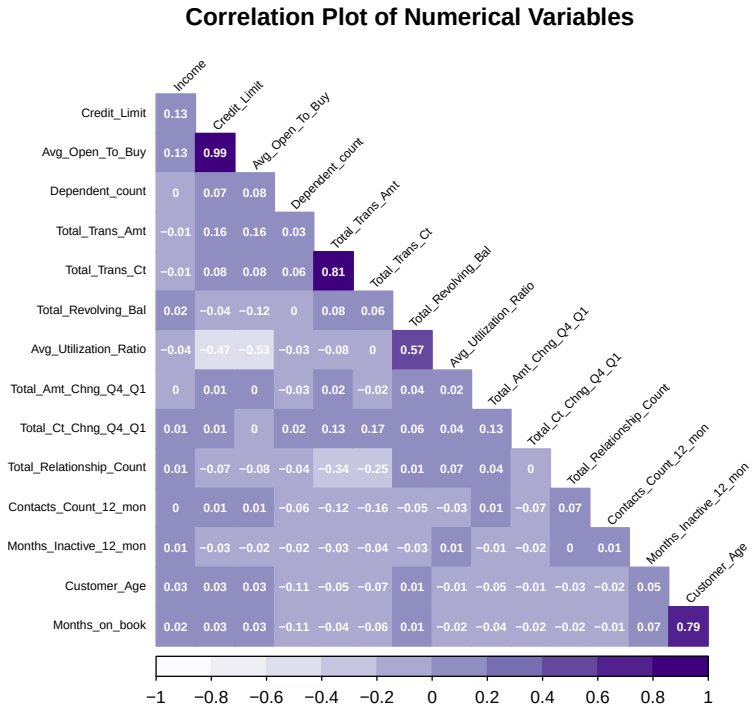


With the following bivariate analysis we explore the relationship between `Customer_Age` and `Total_Trans_Ct` to understand how the total number of transactions varies across different age groups. The box plot provides a visual summary of the distribution of the total number of transactions for each age group, highlighting the median, quartiles, and potential outliers. The box plot shows that the number of transactions is relatively consistent across different age groups, and this suggests that the total number of transactions is not significantly influenced by age.



The last bivariate analysis explores the relationship between `Income`, `Gender`, and `Closed_Account`. The boxplot visualizes the distribution of income across genders, with an additional differentiation between accounts that are closed ('Yes') and those that are not ('No'). The boxplot analysis indicates that the median income and overall income distribution are similar for both genders regardless of whether the accounts are open or closed. Outliers suggest some individuals have significantly higher incomes in both categories. There's no visible difference in income related to account status. This suggests that income alone may not be a strong predictor of account closure.

Correlation



Finally, we present a correlation matrix which displays the correlation coefficients between pairs of numerical variables in the dataset. The color intensity and the number within each square represent the strength and direction of the correlation: the closer the number is to 1, the stronger the positive correlation, while the closer the number is to -1, the stronger the negative correlation. A number close to 0 indicates little to no linear relationship. The correlation matrix reveals several key relationships between variables: `Total_Trans_Amt` and `Total_Trans_Ct` show a strong positive correlation, indicating that transaction counts and amounts tend to increase together. `Avg_Utilization_Ratio` and `Total_Revolving_Bal` have a moderate positive correlation, suggesting that higher revolving balances are generally associated with higher utilization ratios. There's also a noticeable positive correlation between `Months_Inactive_12_mon` and `Closed_Account`. `Customer_Age` and `Months_on_book` are highly correlated, likely reflecting that older customers have been with the bank longer.

Logistic Regression

Implementing the Logistic Regression Model In the forthcoming section of our report, we are set to employ a Logistic Regression Model to analyze how an individual's income and their gender may affect the probability that they will close a bank account. This statistical method will enable us to predict the odds of account closure based on these variables. We will specifically look at whether income influences the closure decision differently for men and women. To make sense of the results, we will interpret the model's coefficients, which quantify the relationship between the predictors (income and gender) and the likelihood of account closure.

```
set.seed(1)
model <- glm(Closed_Account ~ Income*Gender, data = data, family = binomial(link="logit"))
summary(model)
```

```
##
## Call:
```

```
## glm(formula = Closed_Account ~ Income * Gender, family = binomial(link = "logit"),
##     data = data)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.6018179  0.0905192 -17.696  < 2e-16 ***
## Income          0.0001220  0.0008468   0.144  0.88545
## GenderM        -0.4374483  0.1649679  -2.652  0.00801 **
## Income:GenderM  0.0024540  0.0015916   1.542  0.12310
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5190.3  on 5983  degrees of freedom
## Residual deviance: 5178.1  on 5980  degrees of freedom
## AIC: 5186.1
##
## Number of Fisher Scoring iterations: 4
```

```
income_range <- with(data, seq(min(Income, na.rm = TRUE), max(Income, na.rm = TRUE), length.out = 100))
newdata <- expand.grid(Income = income_range, Gender = levels(data$Gender))
newdata$Probability <- predict(model, newdata = newdata, type = "response")
```

The script begins by fitting a logistic regression model, with the intention of analyzing the interaction between income and gender in relation to the likelihood of a bank account being closed. The `glm` function specifies the model where `Closed_Account` is the dependent variable we're interested in, and `Income` and `Gender` are the independent variables, including their interaction term (Income*Gender). The 'family' argument set to 'binomial' with the 'logit' link function indicates that we are modeling a binary outcome (account being closed or not) using logistic regression.

Next, we generate predictions from our model. We create a range of income values that span from the minimum to the maximum income observed in our dataset, ensuring we cover the entire spectrum of our data. This range consists of 100 equally spaced values.

We then set up a new data frame called `newdata`, which combines these income values with the different levels of gender found in our original dataset. This is achieved through the `expand.grid` function, which effectively creates all possible combinations of income and gender that we want to use for our predictions.

Finally, we employ the 'predict' function to estimate the probabilities of account closure across our range of incomes for each gender within `newdata`.

From the coefficients, we understand that the intercept term (-1.6018179) represents the estimated log-odds of account closure when income is zero and the individual is female. The coefficient for income suggests that, holding gender constant, a one-unit increase in income is associated with a negligible change in the log-odds of account closure. For gender, the coefficient (-0.44483) indicates that being male is associated with a decrease in the log-odds of account closure compared to being female. However, the interaction term (Income:GenderM) has a coefficient (0.0024540) that is not statistically significant (p value = 0.1231), suggesting that the effect of income on account closure does not differ significantly between genders. These findings imply that *while gender plays a significant role in account closure probability, income alone does not have a substantial impact once gender is taken into account.*

```
ggplot(newdata, aes(x = Income, y = Probability, color = Gender)) +
  geom_line() +
  labs(title = "Effect of Income and Gender on the Probability of Closing an Account",
```

```
      x = "Income",
      y = "Probability of Account Closure",
      color = "Gender") +
scale_y_continuous(labels = scales::percent_format()) +
theme_minimal() +
theme(
  plot.title = element_text(hjust = 0.5),
  axis.text.x = element_text(angle = 45, hjust = 1),
  axis.title.x = element_text(face = "bold"),
  axis.title.y = element_text(face = "bold"),
)
```

The red line representing females remains consistently horizontal across all income levels, which implies that the propensity for females to close an account does not fluctuate with changes in income. In contrast, the blue line for males ascends steadily, suggesting an increasing probability of males closing an account as their income rises.

Addressing Unbalanced Data The dataset is unbalanced, with a significantly higher proportion of open accounts compared to closed accounts. This imbalance can lead to biased model predictions, as the model may be more inclined to predict open accounts due to their higher representation in the dataset. In this section we have explicitly decided to address this issue by undersampling the majority class (open accounts) to balance the dataset. This approach involves randomly selecting a subset of open accounts to match the number of closed accounts, ensuring that both classes are equally represented in the data.

```r
set.seed(1)

majority_indices <- which(data$Closed_Account == 0)
minority_count <- sum(data$Closed_Account == 1)

sampled_indices <- sample(majority_indices, size = minority_count, replace = FALSE)
balanced_data <- data[c(sampled_indices, which(data$Closed_Account == 1)), ]
balanced_model <- glm(Closed_Account ~ Income*Gender, data = balanced_data, family = binomial(link="log
```

Now we can compare the confusion matrices of the unbalanced and balanced models to evaluate the impact of balancing the dataset on the model's predictive performance.

```r
# Unbalanced Confusion Matrix
predicted_Y <- ifelse(predict(model, type = "response") > 0.5, 1, 0)
actual_Y <- data$Closed_Account
True.positive <- sum(predicted_Y[actual_Y == 1] == 1)
True.negative <- sum(predicted_Y[actual_Y == 0] == 0)
False.positive <- sum(predicted_Y[actual_Y == 0] == 1)
False.negative <- sum(predicted_Y[actual_Y == 1] == 0)
Confusion.Matrix <- matrix(c(True.positive, False.negative,
                             False.positive, True.negative),
                           nrow = 2, byrow = TRUE)
rownames(Confusion.Matrix) <- c("Actual Positive", "Actual Negative")
colnames(Confusion.Matrix) <- c("Predicted Positive", "Predicted Negative")

# Balanced confusion matrix
predicted_Y_balanced <- ifelse(predict(balanced_model, type = "response") > 0.5, 1, 0)
actual_Y_balanced <- balanced_data$Closed_Account
True.positive_balanced <- sum(predicted_Y_balanced[actual_Y_balanced == 1] == 1)
True.negative_balanced <- sum(predicted_Y_balanced[actual_Y_balanced == 0] == 0)
False.positive_balanced <- sum(predicted_Y_balanced[actual_Y_balanced == 0] == 1)
False.negative_balanced <- sum(predicted_Y_balanced[actual_Y_balanced == 1] == 0)

Confusion.Matrix_balanced <- matrix(c(True.positive_balanced, False.negative_balanced,
                                      False.positive_balanced, True.negative_balanced),
                                    nrow = 2, byrow = TRUE)
rownames(Confusion.Matrix_balanced) <- c("Actual Positive", "Actual Negative")
colnames(Confusion.Matrix_balanced) <- c("Predicted Positive", "Predicted Negative")

Confusion.Matrix
```

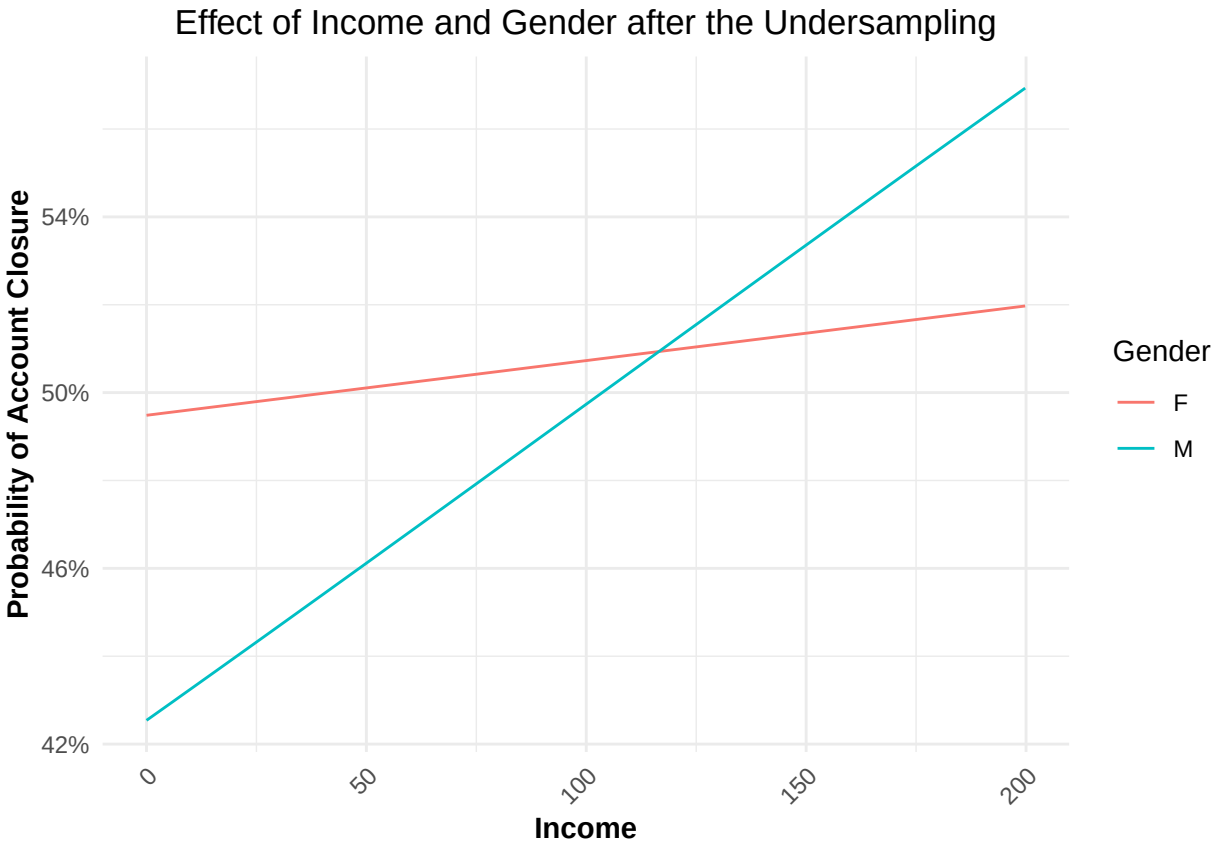
```
##                 Predicted Positive Predicted Negative
## Actual Positive                  0                936
## Actual Negative                  0               5048
```

```r
Confusion.Matrix_balanced
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                569                367
## Actual Negative                545                391
```

As shown in the confusion matrices, the balanced model has a higher true positive rate and a lower false positive rate compared to the unbalanced model. This indicates that balancing the dataset has improved

the model's ability to correctly predict closed accounts, reducing the bias towards open accounts that was present in the unbalanced model.



The unbalanced dataset plot suggests a relatively flat effect of income on account closure probability for females and a moderate effect for males, with a narrow overall probability range (12%-18%). In contrast, **the balanced dataset plot reveals a clearer and steeper increase in closure probability for both genders as income rises**, with males showing a higher likelihood than females across the income spectrum, and a wider probability range (42%-54%). This suggests that balancing the dataset might uncover stronger and more differentiated relationships between the variables.

Main conclusions To conclude the analysis of account closure probabilities in relation to income and gender, the logistic model provides insightful revelations. The baseline propensity for account closure is statistically significant for females when other variables are not considered, with the log-odds of account closure for females at the intercept being -1.548. While income appears to have a negligible impact on account closure probability for females, it is notable that **being male is associated with lower odds of closing an account but it is more sensitive to income levels**. The interaction between income and being male does not significantly alter the account closure probability, indicating that the influence of income on account closure is consistent across genders.

The visual analysis confirms these findings, showing a flatter probability line for females across all income levels, whereas for males, there is a discernible positive slope, suggesting an increasing probability of account closure as income rises. These insights suggest that gender plays a more significant role in the likelihood of account closure than income alone. The lack of a significant interaction effect between income and gender indicates that income's influence on account closure decisions is uniform across both genders.

Upon examination of the regression results, it becomes apparent that income does not exert a differential impact on the probability of account closure between males and females. This is evidenced by the non-significant interaction coefficient (Income:GenderM) obtained from the model. A significant coefficient for this interaction term would suggest a variance in the relationship between income and account closure likelihood across genders. However, given the lack of statistical significance, we infer that the effect of income on account closure is homogeneous for both genders when other variables are controlled. This indicates that while there may be distinct overall probabilities for males and females regarding account closures, with males generally less inclined to close their accounts, the influence of income on this decision-making process does not significantly diverge between the genders. Therefore, based on the analyzed data, income appears to be a uniform factor in predicting account closure probabilities across male and female account holders within the scope of this study.

K-NN Model

We now proceed with the implementation of the K-Nearest Neighbors (K-NN) algorithm to predict the likelihood of account closure; focusing on the continuous predictors Total Trans Amt (Total Transaction Amount) and Total Trans Ct (Total Transaction Count) we aim to determine the optimal number of neighbors (k) that maximizes the predictive performance of our model.

Splitting the dataset We start by splitting the dataset into training and validation sets to evaluate the performance of our K-NN model. We select two features (`Total_Trans_Amt` and `Total_Trans_Ct`) from both training and validation datasets (`data.train` and `data.val`) for the feature variables (`train.x` and `val.x`). We also extracts the last column from these datasets (Closed_Account) as the target variable (`train.y` and `val.y`) for both training and validation, respectively.

```
set.seed(1)

ids.train <- sample(1:nrow(data), size = 0.75 * nrow(data), replace = F)
data.train <- data[ids.train,]
data.val <- data[-ids.train,]

train.x <- select(data.train, c("Total_Trans_Amt", "Total_Trans_Ct"))
train.y <- data.train[,length(data.train)]
val.x <- select(data.val, c("Total_Trans_Amt", "Total_Trans_Ct"))
val.y <- data.val[,length(data.val)]
```

Here, we proceed by **standardizing the data** to ensure that all features are on the same scale, which is essential for the K-NN algorithm to function effectively: moreover, we carry out the standardization process on both the training and validation datasets separately to maintain consistency in the scaling of our features and to **avoid data leakage**.

```
train.x.scaled <- scale(train.x)
train.mean <- apply(train.x, MARGIN = 2, FUN = mean)
train.sd <- apply(train.x, MARGIN = 2, FUN = sd)

val.x.scaled <- scale(val.x, center = train.mean, scale = train.sd)
```

Implementing the K-NN Model Here we implement a procedure to evaluate the performance of a k-Nearest Neighbors (K-NN) classifier across a range of k values, from 1 to 50, on both training and validation datasets. We start by creating a sequence of k values from 1 to 50; each of these values will be used to train a kNN model to find the optimal number of neighbors. For each iteration, the function: - trains a K-NN

model on the scaled training dataset (train.x.scaled) and makes predictions on both the scaled training data and the scaled validation data (val.x.scaled) using the current k value;
- calculates the misclassification rate for both training and validation predictions by comparing predicted labels to actual labels; - computes the probability of positive class predictions (prob.yes.train and prob.yes.val) based on the predicted probabilities attached to the kNN predictions; - calculates the Area Under the Curve (AUC) for both training and validation sets using these probabilities. AUC is a performance metric for binary classification models; - determines the number of true positives, true negatives, false positives, and false negatives for both training and validation datasets, which are essential for calculating sensitivity, specificity, and accuracy; - calculates sensitivity (true positive rate), specificity (true negative rate), and accuracy for both training and validation predictions; - finally, it aggregates these metrics into a vector results for each k value.

```
k.grid <- 1:50
set.seed(1)
k.metrics <- sapply(k.grid, function(k) {
  knn.fit.train <- knn(train = train.x.scaled, test = train.x.scaled, cl = train.y, k = k, prob = T)
  knn.fit.val <- knn(train = train.x.scaled, test = val.x.scaled, cl = train.y, k = k, prob = T)

  misclass.train <- 1 - mean(train.y == knn.fit.train)
  misclass.val <- 1 - mean(val.y == knn.fit.val)

  prob.yes.train <- ifelse(knn.fit.train == "1", attr(knn.fit.train, "prob"), 1 - attr(knn.fit.train, "
  prob.yes.val <- ifelse(knn.fit.val == "1", attr(knn.fit.val, "prob"), 1 - attr(knn.fit.val, "prob"))

  auc.train <- auc(train.y == "1", prob.yes.train)
  auc.val <- auc(val.y == "1", prob.yes.val)

  true.positives.train <- sum(knn.fit.train[which(train.y == 1)] == "1")
  true.negatives.train <- sum(knn.fit.train[which(train.y == 0)] == "0")
  false.positives.train <- sum(knn.fit.train[which(train.y == 0)] == "1")
  false.negatives.train <- sum(knn.fit.train[which(train.y == 1)] == "0")

  sensitivity.train <- true.positives.train / (true.positives.train + false.negatives.train)
  specificity.train <- true.negatives.train / (true.negatives.train + false.positives.train)
  accuracy.train <- (true.positives.train + true.negatives.train) / length(knn.fit.train)

  true.positives.val <- sum(knn.fit.val[which(val.y == 1)] == "1")
  true.negatives.val <- sum(knn.fit.val[which(val.y == 0)] == "0")
  false.positives.val <- sum(knn.fit.val[which(val.y == 0)] == "1")
  false.negatives.val <- sum(knn.fit.val[which(val.y == 1)] == "0")

  sensitivity.val <- true.positives.val / (true.positives.val + false.negatives.val)
  specificity.val <- true.negatives.val / (true.negatives.val + false.positives.val)
  accuracy.val <- (true.positives.val + true.negatives.val) / length(knn.fit.val)


  results <- c(misclass.train, misclass.val, auc.train, auc.val, sensitivity.train, sensitivity.val, sp

  return(results)
})
```

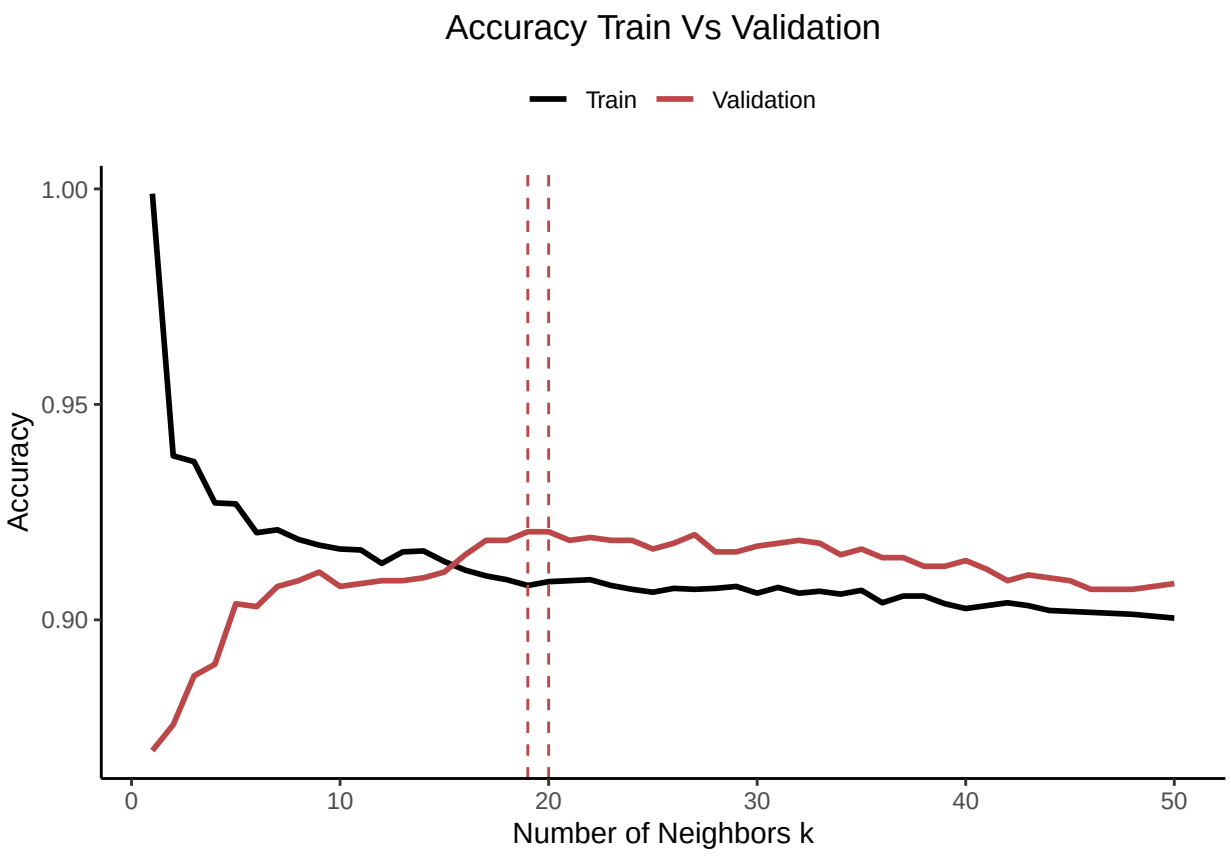
Here we transform the K-NN model evaluation results into a data frame where each row corresponds to a different k value (number of neighbors) and label the columns with descriptive names for each performance metric, such as **misclassification rates**, **AUC scores**, **sensitivity**, **specificity**, and **accuracy** for both

training and validation datasets. This structure makes it easier to analyze the impact of varying k values on model performance.

```
k.metrics <- data.frame(t(k.metrics))
colnames(k.metrics) <- c("Misclassification_Train", "Misclassification_Val", "AUC_Train", "AUC_Val", "S
```

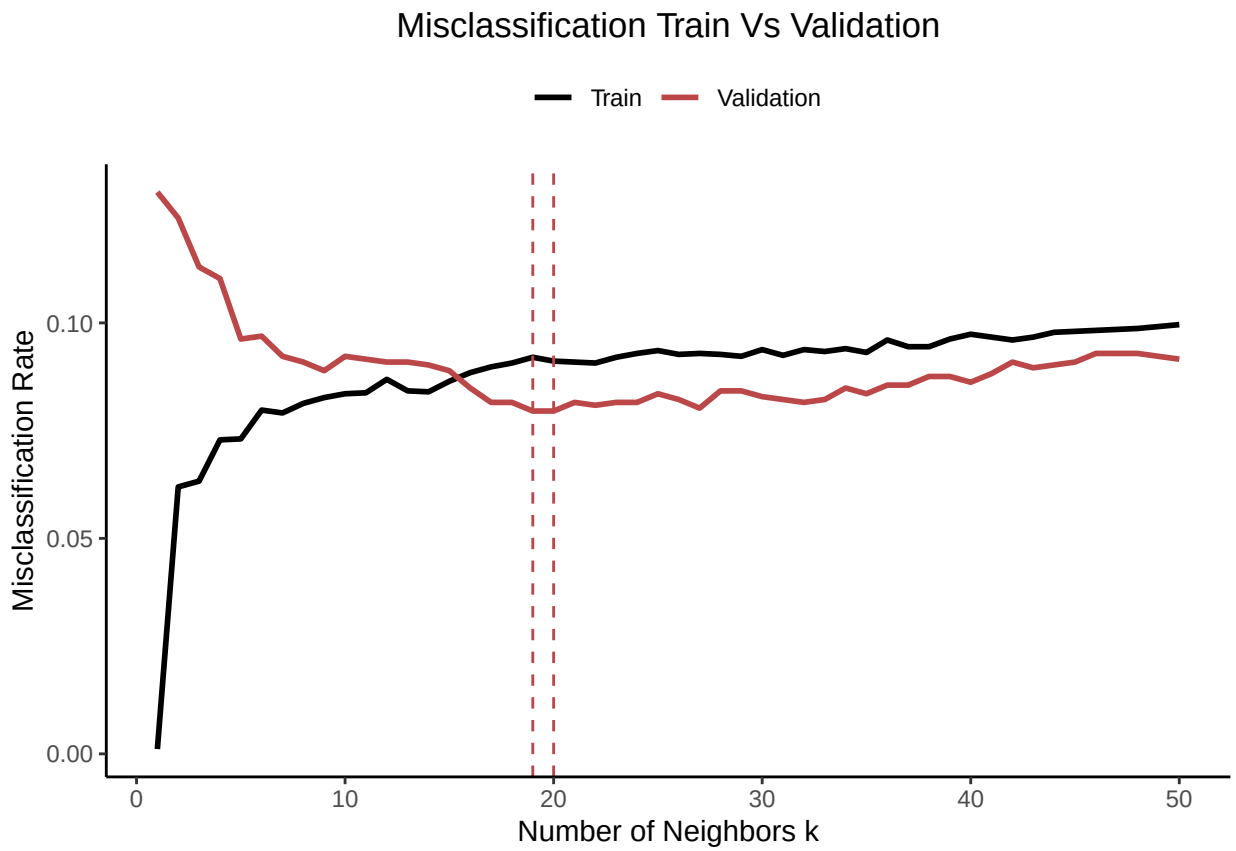
Comparing Performances We now proceed to create several plots to visualize the performance of our K-NN model across different k values by comparing the performances obtained in the train and validation sets; these plots will help us identify the optimal number of neighbors that maximize the model's predictive power.

Performance Plot using Accuracy



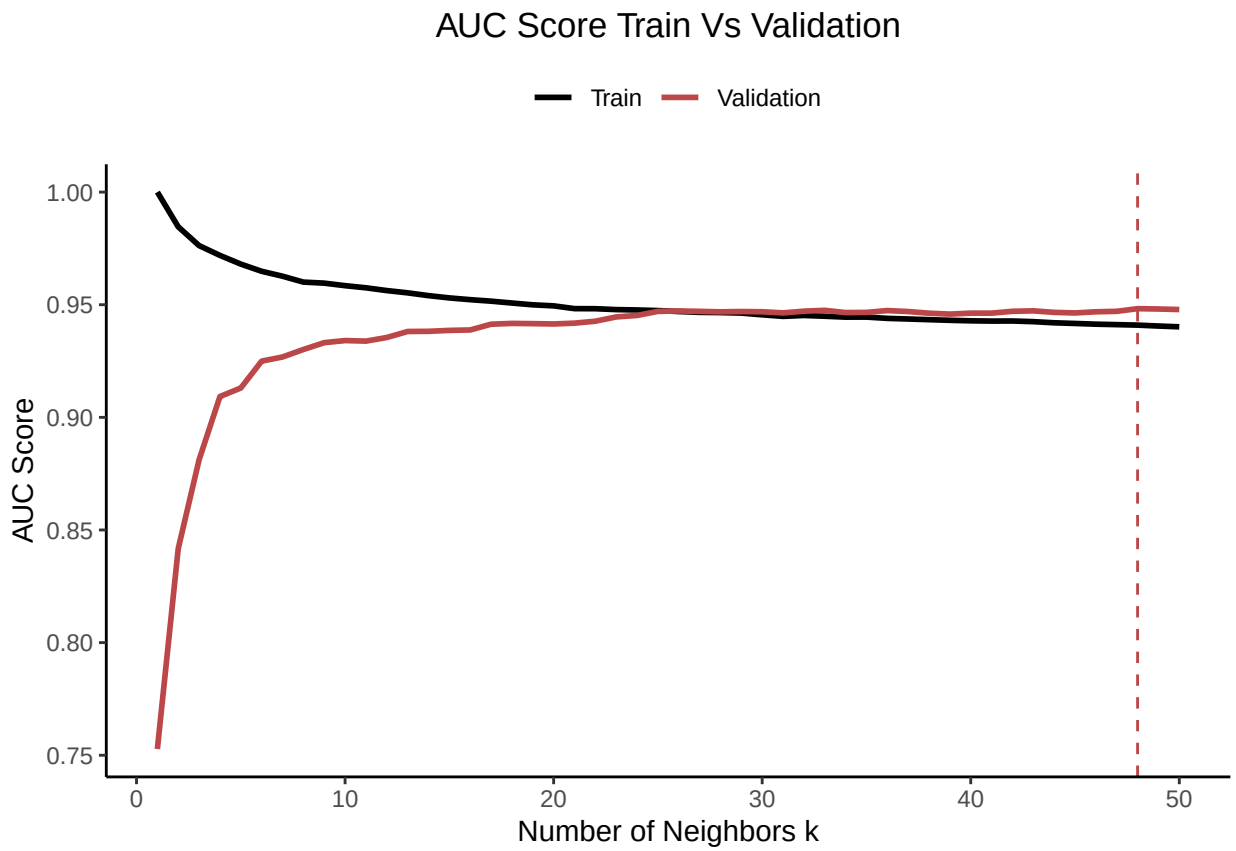
The above graph shows the performance of our model with different numbers of neighbors by evaluating their accuracy. The training accuracy starts high at lower k values and generally decreases as k increases, which is typical since a very low k can lead to overfitting, where the model captures noise in the training data. The validation accuracy, while lower and more variable, suggests how well the model generalizes to unseen data. The optimal k value, indicated by the dashed vertical line around **k=18** and **k=20** is where the validation accuracy peaks, suggesting it's the best compromise between underfitting and overfitting for this model.

Performance Plot using Misclassification Rate



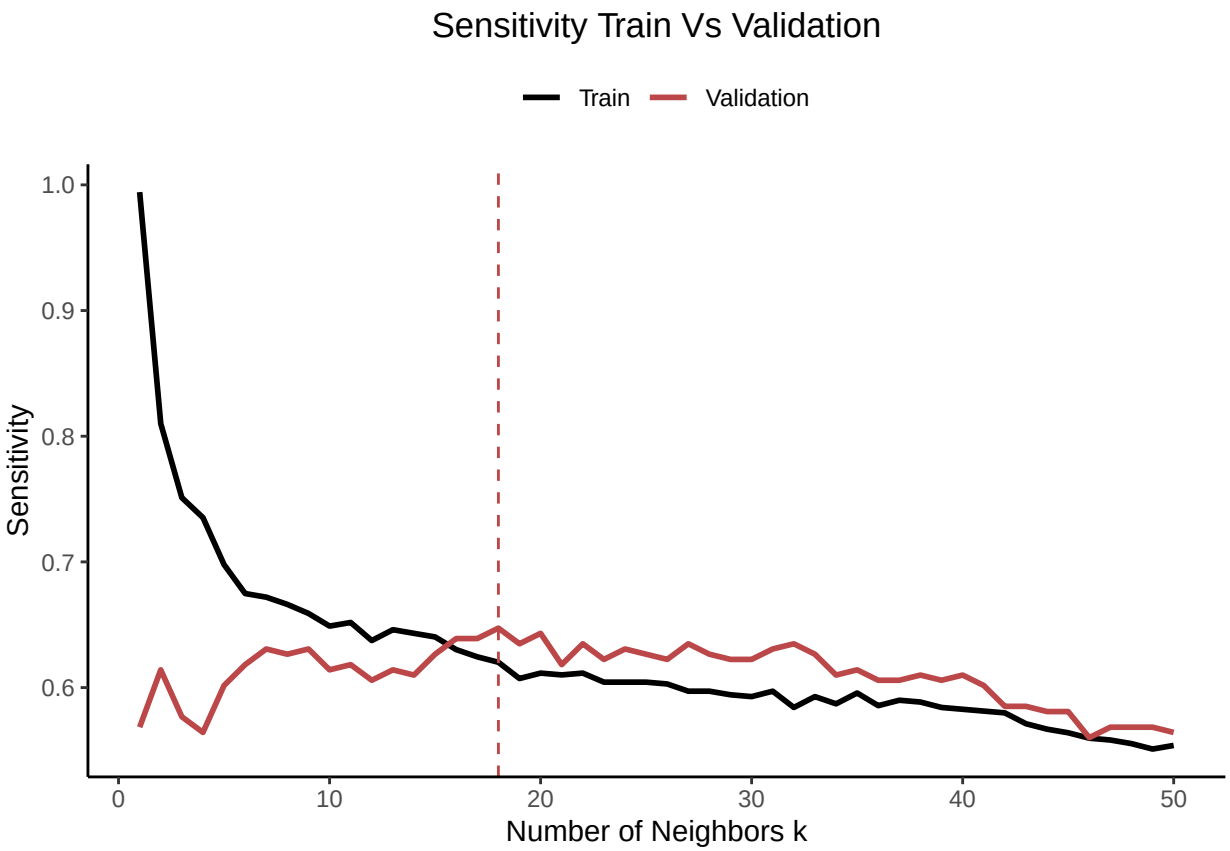
The code above plots misclassification rates for our KNN model on the training data (in black) and validation data (in red) over a range of k values (number of neighbors). The y-axis indicates the misclassification rate, which is lower near the start, especially for the training data, and gradually increases with k. The x-axis represents the number of neighbors k. The training misclassification rate starts very low, suggesting potential overfitting with a very small number of neighbors, but then stabilizes. The validation misclassification rate decreases initially, then levels off and runs parallel to the training misclassification rate, suggesting an optimal balance between bias and variance, again around **k=18** and **k=20**.

Performance Plot using AUC



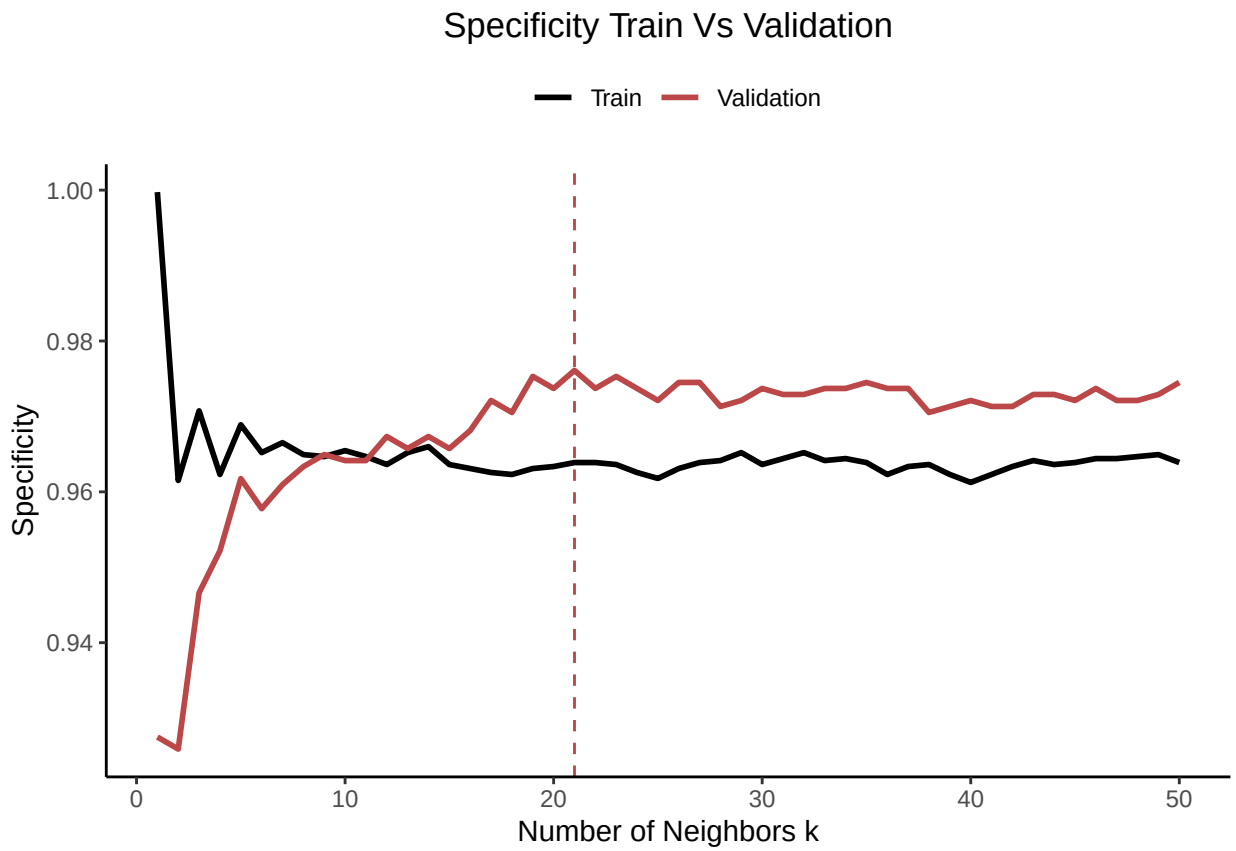
This graph displays the Area Under the Curve (AUC) scores for both training (in black) and validation (in red) datasets of our model, plotted over a range of k values. The y-axis represents the AUC score, and the x-axis denotes the number of neighbors k. The training AUC starts very high, near 1.00, indicating overfit on the training data. It shows a gentle decline as k increases, which is common as a larger number of neighbors typically smoothens the decision boundary and may decrease the model's ability to fit the training data closely. The validation AUC increases sharply at first, then levels off around k=10, suggesting that the model's ability to generalize improves with a few neighbors and stabilizes or slightly declines with more neighbors. The dashed vertical line suggests an **optimal k value of about 48** based on the validation AUC score, where the model likely strikes a balance between underfitting and overfitting. The closeness of the two lines towards the right side of the graph indicates that the model, with higher k values, has a similar performance on both training and validation data, which could be indicative of a more robust model with better generalization capabilities.

Performance Plot using Sensitivity



Here, the above code plots the sensitivity (true positive rate) of our KNN model for both the training set (black line) and the validation set (red line) across different k values. The sensitivity (the ability of the model to correctly identify the positive cases out of all actual positive cases) is highest for the training set at lower k values and decreases with more neighbors. The validation sensitivity also decreases initially and then levels off, with a slight fluctuation as k increases. Again, the vertical dashed line marks the k value with the highest sensitivity on the validation set, suggesting a value of **k=18** as the best from a sensitivity standpoint.

Performance Plot using Specificity



Lastly, the above graph displays specificity (the true negative rate) for our KNN model on the training set (black line) and validation set (red line) across varying k values. Both lines start with a bit of fluctuation; the training set specificity is initially high and stabilizes as k increases, while the validation set specificity improves and then roughly parallels the training specificity. In this case we have that a value of $\mathbf{k = 21}$ maximizes specificity. Overall, the model maintains high specificity across most k values, particularly in the validation set, suggesting it's consistently good at correctly identifying negative cases (open accounts).

Confusion Matrix

Now, we'll create a function to generate the confusion matrix for the KNN model with the optimal k value for each performance metric we previously plotted (accuracy, misclassification rate, AUC, sensitivity, and specificity) to evaluate the model's predictive performance in terms of true positives, true negatives, false positives, and false negatives. Remember that true negatives are the number of correctly identified negative cases (0 or open accounts), true positives are the number of correctly identified positive cases (1 or closed accounts), false negatives are the number of actual positive cases incorrectly identified as negative, and false positives are the number of actual negative cases incorrectly identified as positive.

```
set.seed(1)
confusion.matrix.k <- function(k) {
  knn.fit <- knn(train = train.x.scaled, test = val.x.scaled, cl = train.y, k = k, prob = T)

  true.positives <- sum(knn.fit[which(val.y == 1)] == "1")
  true.negatives <- sum(knn.fit[which(val.y == 0)] == "0")
  false.positives <- sum(knn.fit[which(val.y == 0)] == "1")
  false.negatives <- sum(knn.fit[which(val.y == 1)] == "0")
```

```
  cm <- matrix(c(true.positives, false.positives,
                 false.negatives, true.negatives), nrow = 2, ncol = 2)
  row.names(cm) = c("Actual Positive", "Actual Negative")
  colnames(cm) = c("Predicted Positive", "Predicted Negative")
  return(cm)
}
```

```
confusion.matrix.k(best.k.acc)
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                149                 92
## Actual Negative                 30               1225
```

```
confusion.matrix.k(best.k.misc)
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                137                104
## Actual Negative                 90               1165
```

```
confusion.matrix.k(best.k.auc)
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                137                104
## Actual Negative                 36               1219
```

```
confusion.matrix.k(best.k.sens)
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                153                 88
## Actual Negative                 41               1214
```

```
confusion.matrix.k(best.k.spec)
```

```
##                 Predicted Positive Predicted Negative
## Actual Positive                150                 91
## Actual Negative                 31               1224
```

Interpretation and choice of the best k

The varying numbers in these matrices reflect how the model's predictions shift when optimizing for different performance metrics. A higher number of true positives and true negatives indicate better performance, but the trade-offs between false positives and false negatives can significantly affect the choice of k, depending on the cost or impact of each type of error in the specific application context. Notice that since we are dealing with an unbalanced dataset, as shown in the EDA, with a higher number of open accounts than closed accounts, the model will tend to predict more open accounts than closed accounts. This is reflected in the confusion matrices, where the number of true negatives is much higher than the number of true positives.

- For the model tuned for *accuracy* (`best.k.acc`), we see a balanced trade-off between true positives and true negatives against the false negatives and false positives.

- When tuned for *misclassification rate* (`best.k.misc`), there's an increase in false negatives and false positives, indicating a different balance in the type of errors made.
- The *AUC*-optimized model (`best.k.auc`) shows a similar error distribution to the accuracy-optimized one.
- Optimizing for *sensitivity* (`best.k.sens`), the model slightly improves true positives but maintains a comparable count of false positives.
- Lastly, the *specificity*-tuned model (`best.k.spec`) demonstrates similar scores to the accuracy.

In conclusion, the choice of the best k value depends on our goal and the relative importance of different types of errors. For a balanced approach, the model optimized for accuracy provides the best trade-off between true positives and true negatives, with a relatively low number of false positives and false negatives. However, since in our case the bank is interested in predicting accounts that will close (and we'll see later with the cost matrix that closing accounts determine higher losses), we would suggest to maximize predictions of clients that are effectively leaving in order to act preventively and offer them an incentive to stay. Therefore, we would recommend the **model optimized for sensitivity**, which is designed to maximize true positives, with a k value of **18**.

Best Model

Feature Selection

VIF Here we will select which features to include in our model. We will start by examining the variance inflation factor to detect multicollinearity among the features. The VIF is a measure of how much the variance of the estimated regression coefficients is increased due to multicollinearity in the model. Its formula is

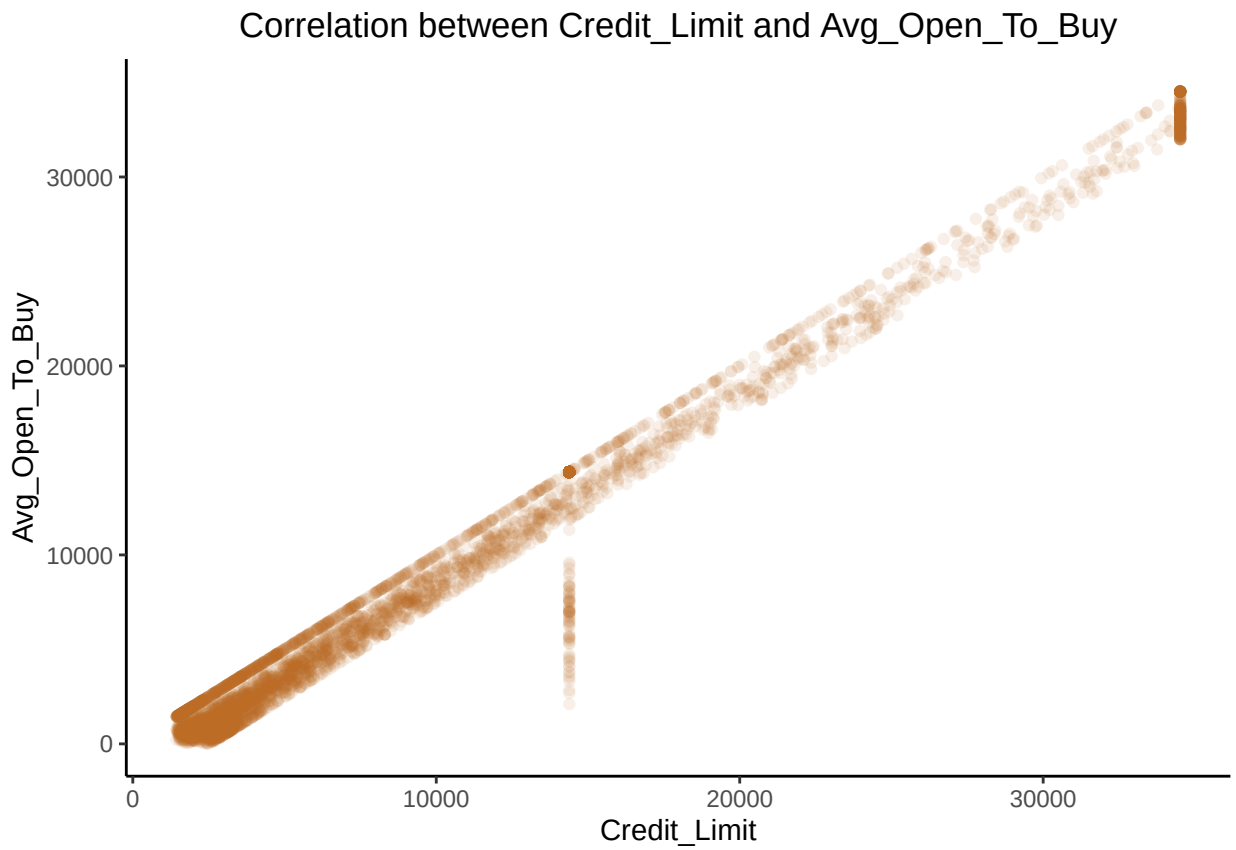
$$VIF_i = \frac{1}{1 - R_i^2}$$

A VIF value greater than 5 indicates a problematic level of multicollinearity. To use the VIF function we'll quickly fit a logistic regression model including all the covariates in our dataframe.

```
logit.full <- glm(data = data.train, data.train$Closed_Account~., family = "binomial")
vif(logit.full)
```

```
##                              GVIF Df GVIF^(1/(2*Df))
## Customer_Age             3.058736  1        1.748924
## Gender                   1.234565  1        1.111110
## Dependent_count          1.060401  1        1.029758
## Education_Level          1.057564  5        1.005613
## Marital_Status           1.100312  2        1.024186
## Card_Category            1.469014  3        1.066197
## Months_on_book           3.017461  1        1.737084
## Total_Relationship_Count 1.190832  1        1.091252
## Months_Inactive_12_mon   1.136269  6        1.010703
## Contacts_Count_12_mon    1.042755  1        1.021154
## Credit_Limit           133.395916  1       11.549715
## Total_Revolving_Bal      2.634537  1        1.623126
## Avg_Open_To_Buy        140.238922  1       11.842252
## Total_Amt_Chng_Q4_Q1     1.032829  1        1.016282
## Total_Trans_Amt          4.024723  1        2.006171
## Total_Trans_Ct           4.233597  1        2.057571
## Total_Ct_Chng_Q4_Q1      1.064141  1        1.031572
## Avg_Utilization_Ratio    2.466466  1        1.570499
## Income                   1.054661  1        1.026967
```

```r
ggplot(data.train, aes(x = Credit_Limit, y = Avg_Open_To_Buy)) +
  geom_point(alpha = 0.1, col = "#bc6c25") +
  theme_classic() +
  labs(title = "Correlation between Credit_Limit and Avg_Open_To_Buy") +
  theme(legend.position = "top", plot.title = element_text(hjust = .5))
```



We notice a critical level of multicollinearity, specifically between the `Credit_Limit` and `Avg_Open_To_Buy` variables, with an exceptionally high Variance Inflation Factor (VIF). This issue is graphically represented with a scatter plot that illustrates the near-perfect linear relationship between the two variables, with the points densely clustered along a diagonal line stretching from the origin upwards. In response to this, we removed the `Avg_Open_To_Buy` variable from our dataset, as it was almost an exact mirror of the `Credit_Limit` variable, evidenced by their correlation coefficient of 0.99. The plot itself serves as a visual confirmation of this redundancy, with every increase in `Credit_Limit` paralleled by an increase in `Avg_Open_To_Buy`. Since the two were basically the same variable, **we decided to remove `Avg_Open_To_Buy` to avoid multicollinearity issues in our model**.

```r
data.train$Avg_Open_To_Buy = NULL
data.val$Avg_Open_To_Buy = NULL
```

Stepwise Selection Next we will choose the best features to include in our model using stepwise selection. This method is used to select the best subset of features by iteratively adding or removing variables based on AIC and BIC criteria. We will use the `step` function to perform forward, backward and bidirectional selection, and then compare the results to choose the best model.

```r
# Stepwise model selection using Akaike Information Criterion
logit.full <- glm(data = data.train, Closed_Account~., family = "binomial") # baseline

logit.fw.ak <- step(glm(Closed_Account ~ 1, family = "binomial", data = data.train), scope = formula(log
logit.bw.ak <- step(logit.full, direction = "backward")
logit.bi.ak <- step(logit.full, direction = "both")

# Stepwise model selection using Bayesian Information Criterion
logit.fw.bay <- step(glm(Closed_Account ~ 1, family = "binomial", data = data.train), scope = formula(l
logit.bw.bay <- step(logit.full, direction = "backward", k = log(nrow(data.train)))
logit.bi.bay <- step(logit.full, direction = "both", k = log(nrow(data.train)))
```

In both cases we get the same variables selected. We will now compare the models selected by the Akaike and Bayesian Information Criteria with the full model performing an anova test to check if there's a significant difference between them. We start from the null hypothesis of model equivalence:

```r
anova(logit.bi.ak, logit.full, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Closed_Account ~ Gender + Dependent_count + Marital_Status +
##     Card_Category + Total_Relationship_Count + Months_Inactive_12_mon +
##     Contacts_Count_12_mon + Credit_Limit + Total_Revolving_Bal +
##     Total_Amt_Chng_Q4_Q1 + Total_Trans_Amt + Total_Trans_Ct +
##     Total_Ct_Chng_Q4_Q1 + Avg_Utilization_Ratio + Income
## Model 2: Closed_Account ~ Customer_Age + Gender + Dependent_count + Education_Level +
##     Marital_Status + Card_Category + Months_on_book + Total_Relationship_Count +
##     Months_Inactive_12_mon + Contacts_Count_12_mon + Credit_Limit +
##     Total_Revolving_Bal + Total_Amt_Chng_Q4_Q1 + Total_Trans_Amt +
##     Total_Trans_Ct + Total_Ct_Chng_Q4_Q1 + Avg_Utilization_Ratio +
##     Income
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      4464     2113.7
## 2      4457     2104.9  7   8.7983   0.2675
```

```r
anova(logit.full, logit.bi.bay, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Closed_Account ~ Customer_Age + Gender + Dependent_count + Education_Level +
##     Marital_Status + Card_Category + Months_on_book + Total_Relationship_Count +
##     Months_Inactive_12_mon + Contacts_Count_12_mon + Credit_Limit +
##     Total_Revolving_Bal + Total_Amt_Chng_Q4_Q1 + Total_Trans_Amt +
##     Total_Trans_Ct + Total_Ct_Chng_Q4_Q1 + Avg_Utilization_Ratio +
##     Income
## Model 2: Closed_Account ~ Gender + Dependent_count + Marital_Status +
##     Total_Relationship_Count + Months_Inactive_12_mon + Contacts_Count_12_mon +
##     Total_Revolving_Bal + Total_Trans_Amt + Total_Trans_Ct +
##     Total_Ct_Chng_Q4_Q1
##   Resid. Df Resid. Dev  Df Deviance Pr(>Chi)
## 1      4457     2104.9
## 2      4471     2136.6 -14  -31.669 0.004463 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results of the first test, by looking at the p-values, we fail to reject the null hypothesis of model equivalence, while in the second test we reject it. Based on the ANOVA results, Model 2 (the model selected according to BIC) we have evidence that excluding the additional variables does not significantly worsen the model's ability to predict closed accounts. Therefore, considering the trade-off between model complexity and explanatory power, the BIC model might be preferable in this case.

Principal Component Analysis We now proceed with the implementation of Principal Component Analysis (PCA) to reduce the dimensionality of our dataset and identify the most important components that explain the variance in the data. This technique transforms the original features into a set of linearly uncorrelated variables called principal components, which are ordered by the amount of variance they explain. The first step consist in selecting and standardizing the continuous variables in our dataset to ensure that all features are on the same scale, which is essential for PCA to function effectively. We will then fit a PCA model to our training data and extract the principal components. To fit the model we will use the `princomp` function, which performs PCA on the input data and returns the principal components.

```
data.continuous <- data.train[, sapply(data.train, is.numeric)]
data.continuous <- data.continuous[, !colnames(data.continuous) %in% c("Closed_Account")]
data.continuous.scaled <- scale(data.continuous)
pca <- princomp(data.continuous.scaled, cor = T, fix_sign = F)
pca$loadings
```

```
##
## Loadings:
##                          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
## Customer_Age              0.200  0.531  0.392
## Dependent_count          -0.141        -0.156  0.106 -0.304  0.594 -0.637
## Months_on_book            0.201  0.529  0.389                       -0.107
## Total_Relationship_Count  0.321 -0.154        -0.257         0.126  0.181
## Contacts_Count_12_mon     0.183        -0.136                -0.694 -0.532
## Credit_Limit             -0.184  0.328 -0.278 -0.299 -0.336        -0.145
## Total_Revolving_Bal             -0.294  0.470 -0.153 -0.379        -0.191
## Total_Amt_Chng_Q4_Q1                          -0.638  0.332        -0.265
## Total_Trans_Amt          -0.594         0.167                -0.182
## Total_Trans_Ct           -0.580         0.182
## Total_Ct_Chng_Q4_Q1      -0.160         0.122 -0.532  0.270  0.247
## Avg_Utilization_Ratio     0.116 -0.428  0.517        -0.117
## Income                                        -0.333 -0.669 -0.129  0.346
##                          Comp.8 Comp.9 Comp.10 Comp.11 Comp.12 Comp.13
## Customer_Age                                            -0.701  -0.102
## Dependent_count          -0.141  0.172 -0.193
## Months_on_book                         -0.100           0.699
## Total_Relationship_Count -0.327 -0.523 -0.598
## Contacts_Count_12_mon    -0.406  0.112
## Credit_Limit              0.131 -0.498  0.299   0.444
## Total_Revolving_Bal       0.145 -0.335  0.225  -0.529          -0.124
## Total_Amt_Chng_Q4_Q1      0.541  0.180 -0.280
## Total_Trans_Amt                        -0.231  -0.123  -0.103   0.699
## Total_Trans_Ct           -0.161        -0.337   0.132          -0.665
## Total_Ct_Chng_Q4_Q1      -0.585  0.116  0.419
## Avg_Utilization_Ratio                          0.698           0.150
```

```
## Income                         0.513 -0.165
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8 Comp.9
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.077  0.077  0.077  0.077  0.077  0.077  0.077  0.077  0.077
## Cumulative Var  0.077  0.154  0.231  0.308  0.385  0.462  0.538  0.615  0.692
##                Comp.10 Comp.11 Comp.12 Comp.13
## SS loadings      1.000   1.000   1.000   1.000
## Proportion Var   0.077   0.077   0.077   0.077
## Cumulative Var   0.769   0.846   0.923   1.000
```

```
data.val.continuous <- data.val[, sapply(data.val, is.numeric)]
data.val.continuous <- data.val.continuous[, !colnames(data.val.continuous) %in% c("Closed_Account")]
data.val.continuous.scaled <- scale(data.val.continuous)
pca.val <- predict(pca, data.val.continuous.scaled)
pca.val <- pca.val[, 1:10]
```

```
pca.var <- pca$sdev^2
pca.var.percent <- pca.var / sum(pca.var)
cum.pca.var.percent <- cumsum(pca.var.percent)
pca.explained <- data.frame(
  Component = 1:length(pca.var.percent),
  Variance = pca.var.percent,
  CumulativeVariance = cum.pca.var.percent
)
pca.explained
```

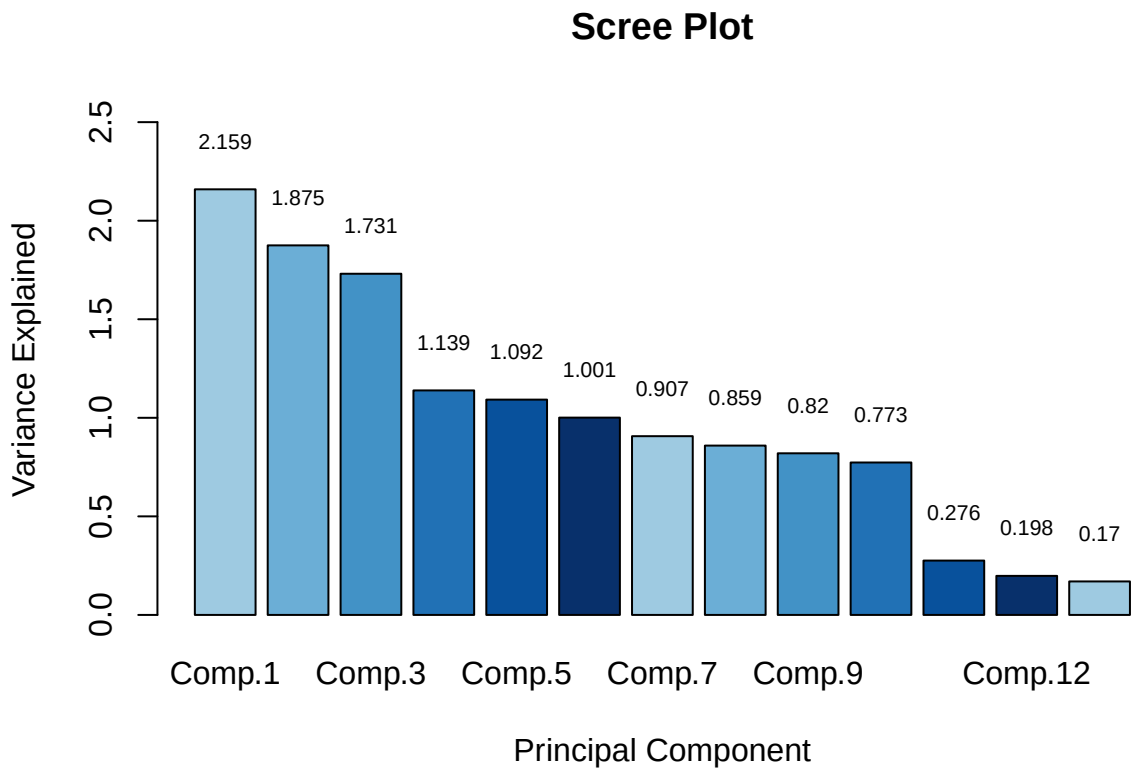
```
##         Component   Variance CumulativeVariance
## Comp.1          1 0.16609115          0.1660912
## Comp.2          2 0.14421151          0.3103027
## Comp.3          3 0.13314143          0.4434441
## Comp.4          4 0.08761495          0.5310590
## Comp.5          5 0.08402545          0.6150845
## Comp.6          6 0.07700269          0.6920872
## Comp.7          7 0.06975407          0.7618413
## Comp.8          8 0.06608779          0.8279290
## Comp.9          9 0.06307574          0.8910048
## Comp.10        10 0.05948372          0.9504885
## Comp.11        11 0.02121155          0.9717001
## Comp.12        12 0.01523100          0.9869310
## Comp.13        13 0.01306895          1.0000000
```

We display the results of our principal components analysis by showing the loadings of each variable on the principal components and the summary of the PCA model. The loadings indicate the correlation between the variables and the principal components, with higher absolute values indicating a stronger relationship. The summary provides information on the proportion of variance explained by each principal component, the cumulative proportion of variance explained, and the eigenvalues of each component. We also created a data frame to print out the explained variance information ordered by their portion of variance explained. In this PCA analysis, the data's complexity has been reduced to 14 principal components, with the first component capturing the largest variance portion (18.5%) and subsequent components explaining progressively less. The cumulative variance covered by the components suggests that, for instance, the first 9 to 10 components may be sufficient for most analytical purposes, encompassing around 90% of the data's variability. High positive or negative loadings indicate a strong association with the component; for example, `Customer_Age`

largely influences `Comp.12`, possibly suggesting a demographic trend, while `Total_Revolving_Bal` significantly impacts `Comp.6`, hinting at financial behaviors.

In order to decide the number of components to keep in our model, we respectively consider the variance explained by each component, the percentage of variance explained, and the cumulative percentage of variance explained; we then create a scree plot to display the variance each component contributes and a line plot to show the cumulative variance explained by the components, marking significant thresholds. Moreover, we create a biplot for a graphical summary of the PCA.

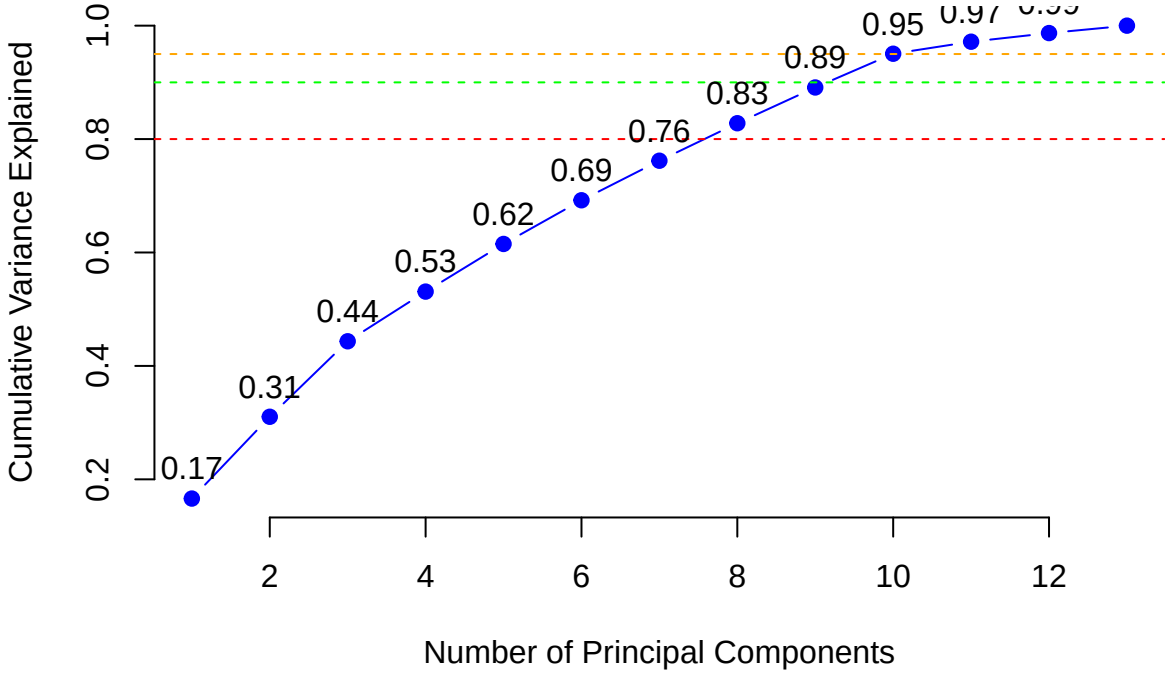
```
blue_palette <- brewer.pal(9, "Blues")[4:9]
scree.plot <- barplot(pca.var, main="Scree Plot", xlab="Principal Component", ylab="Variance Explained"
                      col=blue_palette, ylim=c(0, max(pca.var) * 1.2))
text(x=scree.plot, y=pca.var + max(pca.var) * 0.05, labels=round(pca.var, 3), pos=3, cex=.7)
```



The first bar, representing Component 1 (Comp.1), is the tallest, indicating that it explains the most variance within our dataset at approximately 2.595 units. Each subsequent component explains less variance, with the values labeled on top of each bar. By the time we reach Component 14 (the last one), the variance explained is close to zero, at 0.005 units. The pattern of the bars shows a rapid decrease from Comp.1 to Comp.5, after which the decrease in variance explained slows down, suggesting that the first few components capture most of the important information. In PCA, we look for a point where the decrease in variance explained becomes more gradual, known as the "elbow," to determine the number of components to keep; in our graph, the elbow appears to occur around Comp.5 or Comp.6, suggesting that subsequent components contribute relatively little additional information.

```
plot(cum.pca.var.percent, type='b', xlab="Number of Principal Components", ylab="Cumulative Variance Exp
     main="Cumulative Scree Plot", pch=19, frame=FALSE, col="blue")
```

```r
abline(h=0.8, col="red", lty=2) # Adding a line at 80% cumulative variance explained
abline(h=0.9, col="green", lty=2) # Adding a line at 90% cumulative variance explained
abline(h=0.95, col="orange", lty=2) # Adding a line at 95% cumulative variance explained
text(x=1:length(cum.pca.var.percent), y=cum.pca.var.percent, labels=round(cum.pca.var.percent, 2), pos=3
```



Furthermore, from the above plot, it appears that around 10 components are required to explain 90% of the variance, which might be considered sufficient for our analysis, ensuring that most of the information in the original data is retained while reducing dimensionality. Indeed, we proceed by reducing the number of components to 10 and fitting a logistic regression model using these components to predict the likelihood of account closure.

```r
pca.data <- predict(pca, data.continuous.scaled)
pca.data <- pca.data[, 1:10]
logit.pca <- glm(Closed_Account ~ ., data = data.frame(pca.data, Closed_Account = data.train$Closed_Acc
summary(logit.pca)
```

```
##
## Call:
## glm(formula = Closed_Account ~ ., family = "binomial", data = data.frame(pca.data,
##     Closed_Account = data.train$Closed_Account))
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.51581    0.07029 -35.793  < 2e-16 ***
## Comp.1       0.69030    0.04772  14.465  < 2e-16 ***
```

```
## Comp.2       0.31277    0.03821   8.184 2.73e-16 ***
## Comp.3      -0.64211    0.04159 -15.438  < 2e-16 ***
## Comp.4       0.45404    0.04727   9.605  < 2e-16 ***
## Comp.5       0.07361    0.04657   1.581    0.114
## Comp.6      -0.20450    0.05244  -3.900 9.64e-05 ***
## Comp.7      -0.34707    0.05297  -6.553 5.65e-11 ***
## Comp.8       0.26995    0.05284   5.109 3.24e-07 ***
## Comp.9       0.85177    0.05824  14.626  < 2e-16 ***
## Comp.10      0.52474    0.05784   9.073  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3869.0  on 4487  degrees of freedom
## Residual deviance: 2691.4  on 4477  degrees of freedom
## AIC: 2713.4
##
## Number of Fisher Scoring iterations: 6
```

Comparison and Choice of the best model

For the final step of this section, we compare the models obtained through the different feature selection methods (VIF, Stepwise Selection, PCA) to determine the best model to predict account closure.

```
thresh <- 0.5

pred.full.raw <- predict(logit.full, newdata = val.x, type = "response")
pred.full <- ifelse(pred.full.raw > thresh, 1, 0)
pred.bic.raw <- predict(logit.bi.bay, newdata = val.x, type = "response")
pred.bic <- ifelse(pred.bic.raw > thresh, 1, 0)
pred.aic.raw <- predict(logit.bi.ak, newdata = val.x, type = "response")
pred.aic <- ifelse(pred.aic.raw > thresh, 1, 0)
pred.pca.raw <- predict(logit.pca, newdata = data.frame(pca.val), type = "response")
pred.pca <- ifelse(pred.pca.raw > thresh, 1, 0)
cm.full <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(pred.full))
cm.bic <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(pred.bic))
cm.aic <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(pred.aic))
cm.pca <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(pred.pca))

cm.full$table
```

```
##            Reference
## Prediction    0    1
##          0 1217   38
##          1  105  136
```

```
cm.bic$table
```

```
##            Reference
## Prediction    0    1
##          0 1213   42
##          1  106  135
```

```
cm.aic$table
```

```
##            Reference
## Prediction    0    1
##          0 1214   41
##          1  104  137
```

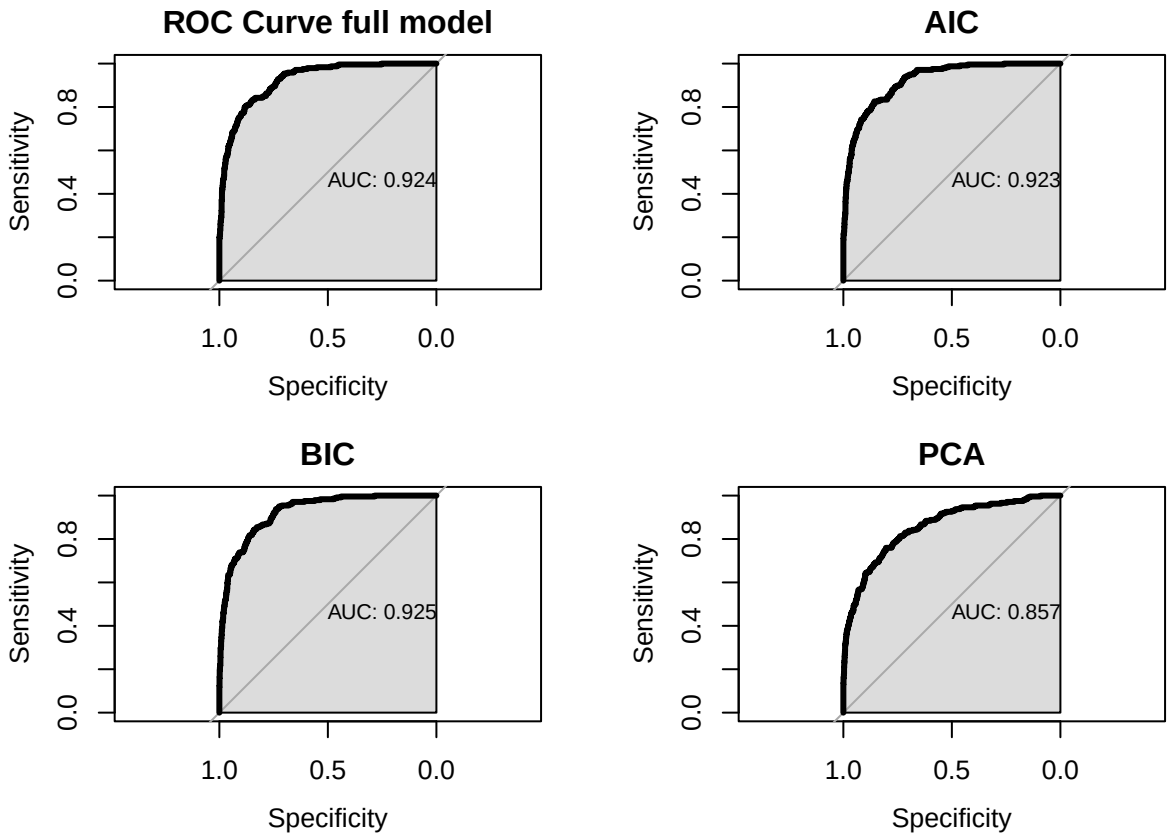
```
cm.pca$table
```

```
##            Reference
## Prediction    0    1
##          0 1222   33
##          1  140  101
```

By analyzing the above confusion matrices applied to our four tuned models we can see that:
- **Full model**: the model has a relatively high number of true negatives and true positives, indicating good classification performance for both classes. However, there are more false negatives than false positives, which suggests a higher cost of missing the positive class.
- **BIC model**: similar to the first model, this one also performs well in identifying the negative class but has a slightly higher number of false positives and one less true positive compared to the first matrix.
- **AIC model**: this model shows a slight improvement over the second model, with one fewer false positive and one fewer false negative, which means it has better accuracy for both predicting negatives and capturing positives.
- **PCA model**: the model associated with this matrix is better at predicting the negative class (fewest false positives) but worse at predicting the positive class (most false negatives).

In summary, all models are relatively consistent in predicting the negative class but vary slightly in their ability to capture the positive class; this is no surprise since as we previously highlight the dataset is unbalanced, with more open accounts than closed accounts.

Again, we proceed to create ROC curves for the full, AIC, BIC, and PCA models; each illustrate the trade-off between sensitivity (true positive rate) and specificity (1 - false positive rate) across different thresholds. Moreover, the AUC (Area Under the Curve) provides a single measure of overall model performance, with higher values indicating better discrimination ability.



As shown by the above plots, the full model has an AUC of 0.924, which suggests a very good predictive performance. It's closely followed by the AIC model with an AUC of 0.923 and the BIC model with an AUC of 0.925. These three models perform almost identically in terms of their ability to distinguish between the positive and negative classes, and their AUC values indicate high accuracy. In contrast, the PCA model exhibits a lower AUC of 0.857. While this still represents a good level of predictive ability, it is notably less effective than the full, AIC, and BIC models. This might be due to the PCA model's reduced feature set, which, although simplifying the model, may have omitted some important predictive information contained in the full data.

Finally, we compare the performances of the full, AIC, BIC, and PCA models in terms of accuracy, sensitivity, specificity, positive predictive value, negative predictive value, precision, recall, F1 score, and AUC. This comparison will help us determine which model is the best at predicting account closure based on the chosen performance metrics.

```
comparison.mat <- rbind(c(cm.full$overall["Accuracy"], cm.full$byClass[1:7], roc.full$auc),
                        c(cm.aic$overall["Accuracy"], cm.aic$byClass[1:7], roc.aic$auc),
                        c(cm.bic$overall["Accuracy"], cm.bic$byClass[1:7], roc.bic$auc),
                        c(cm.pca$overall["Accuracy"], cm.pca$byClass[1:7], roc.pca$auc))
rownames(comparison.mat) <- c("logistic.full", "logistic.aic", "logistic.bic", "logistic.pca")
colnames(comparison.mat) <- c("Accuracy", "Sensitivity", "Specificity", "Positive Predicted Value", "Neg
comparison.mat
```

```
##                Accuracy Sensitivity Specificity Positive Predicted Value
## logistic.full 0.9044118   0.9205749   0.7816092                 0.9697211
## logistic.aic  0.9030749   0.9210926   0.7696629                 0.9673307
## logistic.bic  0.9010695   0.9196361   0.7627119                 0.9665339
```

```
## logistic.pca  0.8843583   0.8972100   0.7537313                0.9737052
##                Negative Predicted Value Precision    Recall        F1       AUC
## logistic.full                0.5643154 0.9697211 0.9205749 0.9445091 0.9241672
## logistic.aic                 0.5684647 0.9673307 0.9210926 0.9436455 0.9234002
## logistic.bic                 0.5601660 0.9665339 0.9196361 0.9425019 0.9245772
## logistic.pca                 0.4190871 0.9737052 0.8972100 0.9338938 0.8574730
```

The performance metrics indicate that the full, AIC, and BIC logistic regression models are all highly effective, with the full model exhibiting a slight edge in accuracy and sensitivity, meaning it's marginally better at correctly identifying true positives. Specificity is highest for the full model, suggesting better true negative identification. Precision across all models is impressively high, indicating a strong likelihood that predicted positives are correct. While the negative predicted value is lower across the models, it doesn't vary much between them. The F1 scores are comparable, suggesting a balanced precision-recall trade-off. The AIC and BIC models show a trivial advantage in AUC, implying a marginally better discriminative ability between positive and negative classes. The PCA model, while still effective, lags behind the others in most metrics, indicating that the reduced feature set may have compromised its predictive power. This suggests that while the full model is better at prediction, the AIC and BIC models offer a better balance in discriminating between the positive and negative classes.

Cross Validation

We will now perform a 10 fold cross validation to measure our models' predictive performance on unseen validation sets. We will report the mean error obtained by each model and see which one overall performs best. This time we will investigate the ROC score.

```
set.seed(1)
train.control <- trainControl(method = "cv", number = 10, classProbs = T, summaryFunction = twoClassSumm
levels(data.train$Closed_Account) <- c("No","Yes")

cvlogit.full <- train(Closed_Account ~ ., data = data.train, method = "glm", family = "binomial", trCon
cvlogit.bic <- train(formula(logit.bi.bay), data = data.train, method = "glm", family = "binomial", trC
cvlogit.aic <- train(formula(logit.bi.ak), data = data.train, method = "glm", family = "binomial", trCo
cvlogit.pca <- train(formula(logit.pca), data = data.frame(pca.data, Closed_Account = data.train$Closed_

cv.matrix <- rbind(cvlogit.full$results$ROC, cvlogit.bic$results$ROC, cvlogit.aic$results$ROC, cvlogit.p
colnames(cv.matrix) <- "Mean ROC Score after 10fold cv"
rownames(cv.matrix) <- c("cvlogit.full", "cvlogit.bay", "cvlogit.aic", "cvlogit.pca")
```

```
cv.matrix
```

```
##              Mean ROC Score after 10fold cv
## cvlogit.full                       0.9189755
## cvlogit.bay                        0.9172043
## cvlogit.aic                        0.9183903
## cvlogit.pca                        0.8446584
```

The mean ROC scores from 10-fold cross-validation suggest that the logistic regression models cvlogit.full, cvlogit.bay, and cvlogit.ak perform quite similarly and very well, with ROC scores above 0.91 indicating high model accuracy. In contrast, cvlogit.pca, which likely uses principal component analysis for dimensionality reduction, shows a noticeably lower mean ROC score, suggesting it may not capture the variance as effectively as the other models in this context. We will exclude a priori the full model because its high score is most likely due to overfitting, among the remaining models, we will select the one having the highest score which is the one selected using the AIC criterion, which as we will see in the following analysis it will also be the one maximizing the gain. We'll now compute the predicted probabilities using this model.

Predictions on data.test using aic model

```
data.test <- data.test[!apply(data.test == "Unknown", 1, any),]

data.test$Months_Inactive_12_mon <- factor(data.test$Months_Inactive_12_mon)

predicted_probabilities_test <- predict(logit.bi.ak, newdata = data.test, type = "response")
predictions_df <- data.frame(CLIENTNUM = data.test$CLIENTNUM, Predicted_Closure = predicted_probabiliti
write.csv(predictions_df, "predicted_account_closures_prob.csv", row.names = FALSE)
```

Cost matrix

In the context of our problem, the cost of misclassification is not uniform. Predicting that an account will close when it actually remains open (FP) may result in a missed opportunity to retain a customer, leading to potential revenue loss of 30. On the other hand, predicting that an account will remain open when it actually closes (FN) may result in a loss of revenue from the account of 50. To account for these different costs, we will create a cost matrix to evaluate the performance of our models based on the financial implications of their predictions.

```
cost_of_TP <- 0
cost_of_FP <- 30  # Cost of an unnecessary offer due to a false alarm (False Positive)
cost_of_FN <- 50  # Cost when we fail to retain a customer who closes the account (False Negative)
cost_of_TN <- 0

cost_mat <- matrix(c(cost_of_TN, cost_of_FP, cost_of_FN, cost_of_TP),
                   ncol = 2,
                   byrow = TRUE,
                   dimnames = list(c("Actual Stay", "Actual Close"),
                                   c("Predicted Stay", "Predicted Close")))
cost_mat
```

```
##              Predicted Stay Predicted Close
## Actual Stay               0              30
## Actual Close             50               0
```

Threshold Selection

To evaluate the models' performance in terms of the cost matrix, we have to pick a threshold, compute the confusion matrix, multiply it by the cost matrix, and finally sum all the entries (to get the total cost).

Treshold = 0.5 First of all we proceed with the classic threshold 0.5 and compute the total cost for each model on the validation set.

```
total_cost_full <- sum(cm.full$table * cost_mat)
total_cost_aic <- sum(cm.aic$table * cost_mat)
total_cost_bic <- sum(cm.bic$table * cost_mat)
total_cost_pca <- sum(cm.pca$table * cost_mat)

cat("Cost Full Model: ", total_cost_full, "\nCost AIC Model: ", total_cost_aic, "\nCost BIC Model: ", t
```

```
## Cost Full Model:  6390
## Cost AIC Model:  6430
## Cost BIC Model:  6560
## Cost PCA Model:  7990
```

The lowest cost is obtained by the full logistic regression model.This suggests that the full logistic regression model is the most cost-effective in terms of minimizing the financial impact of misclassification errors.

Treshold = 0.35 Now we will try to find the optimal threshold for each model that minimizes the total cost. We will do this by computing the total_cost of each model for a threshold equal to 0.35 and selecting the one that minimizes the cost.

```
thresh <- 0.35
predict.Close.raw <- predict(logit.full, newdata = data.val[1:length(data.val) - 1], type = "response")
predict.Close <- ifelse(predict.Close.raw > thresh, 1, 0)
cm.full <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(predict.Close))
total_cost_full1 <- sum(cm.full$table * cost_mat)
total_cost_full1
```

```
## [1] 6200
```

```
predict.Close.raw <- predict(logit.bi.ak, newdata = data.val[1:length(data.val) - 1], type = "response")
predict.Close <- ifelse(predict.Close.raw > thresh, 1, 0)
cm.aic <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(predict.Close))
total_cost_aic1 <- sum(cm.aic$table * cost_mat)
total_cost_aic1
```

```
## [1] 6070
```

```
predict.Close.raw <- predict(logit.bi.bay, newdata = data.val[1:length(data.val) - 1], type = "response")
predict.Close <- ifelse(predict.Close.raw > thresh, 1, 0)
cm.bic <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(predict.Close))
total_cost_bic1 <- sum(cm.bic$table * cost_mat)
total_cost_bic1
```

```
## [1] 6140
```

```
predict.Close.raw <- predict(logit.pca, newdata = data.frame(pca.val), type = "response")
predict.Close <- ifelse(predict.Close.raw > thresh, 1, 0)
cm.pca <- confusionMatrix(as.factor(data.val$Closed_Account), as.factor(predict.Close))
total_cost_pca1 <- sum(cm.pca$table * cost_mat)
total_cost_pca1
```

```
## [1] 8100
```

Comparison between the two thresholds Now that we computed the total cost for each model with two different thresholds, we can compare the results and choose the best model.

```
total_costs_05 <- c(
  logistic.full = total_cost_full,
  logistic.aic = total_cost_aic,
  logistic.bic = total_cost_bic,
  logistic.pca = total_cost_pca
)
total_costs_035 <- c(
  logistic.full = total_cost_full1,
  logistic.aic = total_cost_aic1,
  logistic.bic = total_cost_bic1,
  logistic.pca = total_cost_pca1
)

comparison_df <- data.frame(
  `Cost 0.5` = total_costs_05,
  `Cost 0.35` = total_costs_035
)
comparison_df
```

```
##               Cost.0.5 Cost.0.35
## logistic.full     6390      6200
## logistic.aic      6430      6070
## logistic.bic      6560      6140
## logistic.pca      7990      8100
```

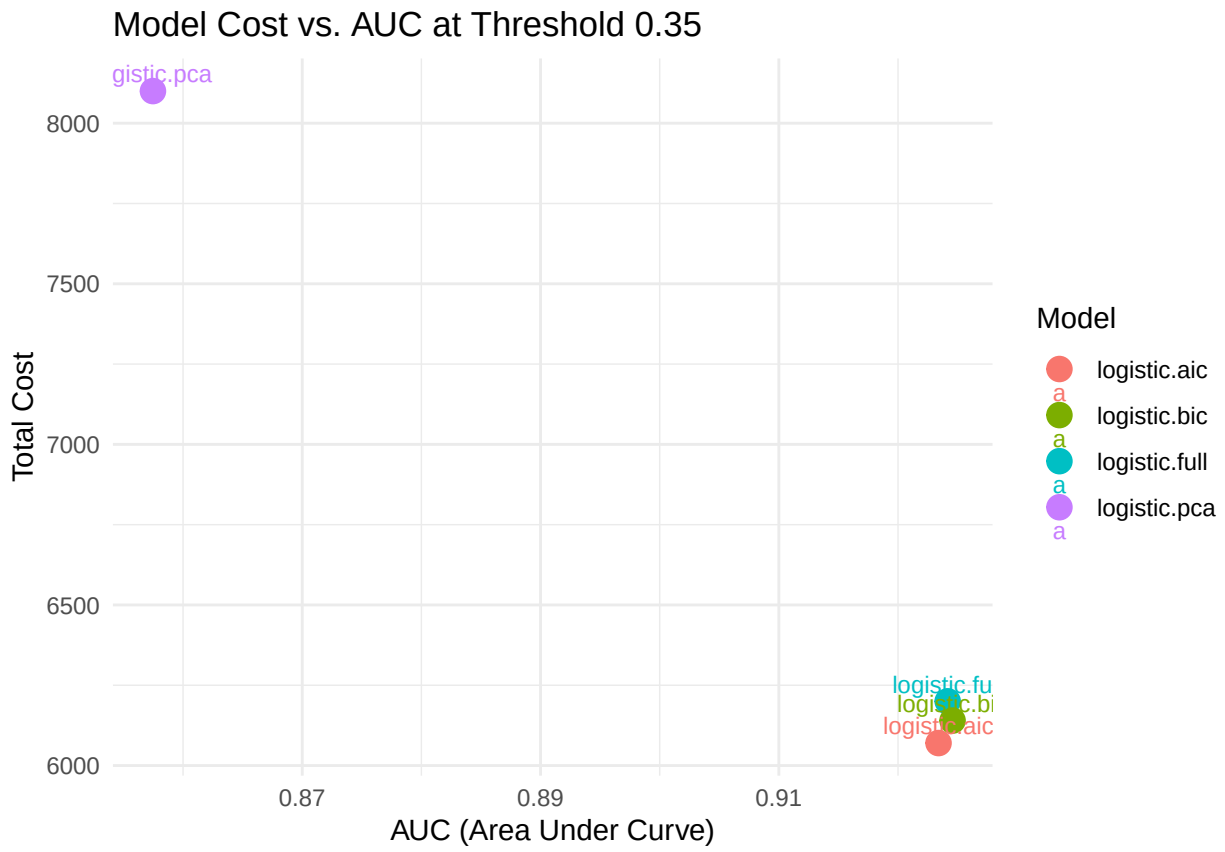
```
comparison.mat <- cbind(comparison.mat, comparison_df)

comparison.mat
```

```
##                Accuracy Sensitivity Specificity Positive Predicted Value
## logistic.full 0.9044118   0.9205749   0.7816092                0.9697211
## logistic.aic  0.9030749   0.9210926   0.7696629                0.9673307
## logistic.bic  0.9010695   0.9196361   0.7627119                0.9665339
## logistic.pca  0.8843583   0.8972100   0.7537313                0.9737052
##               Negative Predicted Value Precision    Recall        F1       AUC
## logistic.full                0.5643154 0.9697211 0.9205749 0.9445091 0.9241672
## logistic.aic                 0.5684647 0.9673307 0.9210926 0.9436455 0.9234002
## logistic.bic                 0.5601660 0.9665339 0.9196361 0.9425019 0.9245772
## logistic.pca                 0.4190871 0.9737052 0.8972100 0.9338938 0.8574730
##               Cost.0.5 Cost.0.35
## logistic.full     6390      6200
## logistic.aic      6430      6070
## logistic.bic      6560      6140
## logistic.pca      7990      8100
```

From a sensitivity point of view we can notice that lowering the threshold from 0.5 to 0.35 generally reduces the total cost for all models. This suggests that using a lower threshold leads to more predictions of account closures, which may help in capturing more true positives (correctly predicted closures) at the expense of potentially increasing false positives (incorrectly predicted closures).

Concerning the models' performance in terms of the cost matrix, **with a lower threshold the aic model is the most cost-effective.**

Visualization Finally, we will visualize the total cost of each model at the 0.35 threshold and compare it with the AUC value for each model. This will provide a visual representation of the trade-off between model cost and predictive performance, helping to identify the best model based on the chosen metrics.



The first noticeable aspect of the graph is the clustering of points representing the full logistic, the AIC, and the BIC models in the bottom right, all with slightly higher AUC values above 0.92, while the PCA model is positioned elsewhere: in the top left. This highlights a significantly higher total cost and a lower AUC (0.85) compared to the other three models.

From a performance standpoint, this indicates that **the full logistic, AIC, and BIC models exhibit better overall performance**, as evidenced by their clustering in the bottom right of the graph with slightly higher AUC values above 0.92. Conversely, the PCA model, positioned in the top left, demonstrates inferior performance, characterized by a significantly higher total cost and a lower AUC of 0.85 compared to the other three models.

Threshold selection to maximize profit

To select the threshold that maximize the profit we will use the following function. It computes the total financial impact by evaluating the performance of a binary classification model based on given predictions and actual labels, considering gains and losses associated with true positives, false positives, true negatives, and false negatives.

```
calculate_financial_impact <- function(threshold, predictions, actual) {
  pred_labels <- ifelse(predictions > threshold, 1, 0)
```

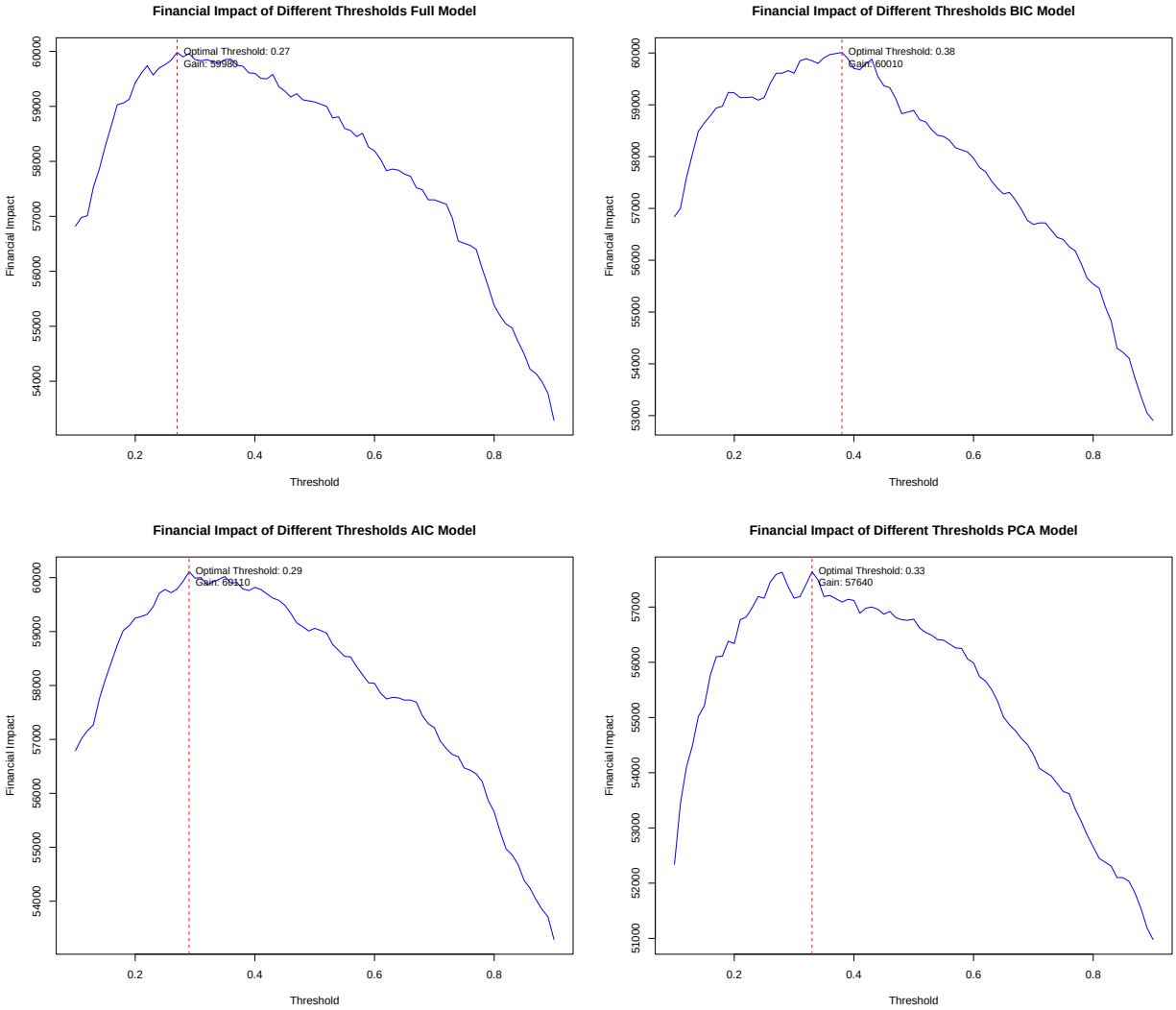
```
  TP <- sum(pred_labels == 1 & actual == 1)
  FP <- sum(pred_labels == 1 & actual == 0)
  TN <- sum(pred_labels == 0 & actual == 0)
  FN <- sum(pred_labels == 0 & actual == 1)


  gain_from_TP_FP <- (TP + FP) * 20
  gain_from_TN <- TN * 50
  loss_from_FN <- FN * -50

  total_gain <- gain_from_TP_FP + gain_from_TN + loss_from_FN
  return(total_gain)
}
```

What we are going to do now is to perform threshold selection to maximize profit by iteratively evaluating the financial impact at different thresholds for each model's predictions. Then, we determine the optimal threshold that yields the highest financial gain and assess model performance using ROC curves and AUC values. Finally, we conduct k-fold cross-validation to further validate the model's performance.

For each model we will calculate the financial impact at different thresholds, identify the optimal threshold that maximizes profit, and evaluate the performance based on the chosen threshold:

```
set.seed(1)
thresholds <- seq(0.1, 0.9, by = 0.01)
par(mfrow = c(2,2))

financial_impacts <- sapply(thresholds, function(thresh) calculate_financial_impact(thresh, pred.full.ra
optimal_threshold_full <- thresholds[which.max(financial_impacts)]
max_gain <- max(financial_impacts)
plot(thresholds, financial_impacts, type = 'l', col = 'blue', xlab = "Threshold", ylab = "Financial Impa
     main = "Financial Impact of Different Thresholds Full Model")
abline(v = optimal_threshold_full, col = 'red', lty = 2)
text(optimal_threshold_full, max_gain, paste("\n\nOptimal Threshold:", round(optimal_threshold_full, 2)

financial_impacts <- sapply(thresholds, function(thresh) calculate_financial_impact(thresh, pred.bic.rau
optimal_threshold_bic <- thresholds[which.max(financial_impacts)]
max_gain <- max(financial_impacts)
plot(thresholds, financial_impacts, type = 'l', col = 'blue', xlab = "Threshold", ylab = "Financial Impa
     main = "Financial Impact of Different Thresholds BIC Model")
abline(v = optimal_threshold_bic, col = 'red', lty = 2)
text(optimal_threshold_bic, max_gain, paste("\n\nOptimal Threshold:", round(optimal_threshold_bic, 2),

financial_impacts <- sapply(thresholds, function(thresh) calculate_financial_impact(thresh, pred.aic.rau
optimal_threshold_aic <- thresholds[which.max(financial_impacts)]
max_gain <- max(financial_impacts)
plot(thresholds, financial_impacts, type = 'l', col = 'blue', xlab = "Threshold", ylab = "Financial Impa
     main = "Financial Impact of Different Thresholds AIC Model")
abline(v = optimal_threshold_aic, col = 'red', lty = 2)
text(optimal_threshold_aic, max_gain, paste("\n\nOptimal Threshold:", round(optimal_threshold_aic, 2),

financial_impacts <- sapply(thresholds, function(thresh) calculate_financial_impact(thresh, pred.pca.rau
optimal_threshold <- thresholds[which.max(financial_impacts)]
max_gain <- max(financial_impacts)
```

```
plot(thresholds, financial_impacts, type = 'l', col = 'blue', xlab = "Threshold", ylab = "Financial Impa
     main = "Financial Impact of Different Thresholds PCA Model")
abline(v = optimal_threshold, col = 'red', lty = 2)
text(optimal_threshold, max_gain, paste("\n\nOptimal Threshold:", round(optimal_threshold, 2), "\nGain:
```



Some insights from the analysis of the **Full model**: - **The optimal threshold for the logit.full model is 0.27**. - The model achieves an accuracy of 88.85%, indicating the proportion of correctly classified instances among all instances. - Sensitivity (true positive rate) measures the proportion of actual positive cases that the model correctly identifies. It is 91.48% in this case. - PPV represents the proportion of instances predicted as positive that are truly positive. Here it is 95.12%: a really high percentage. - Balanced accuracy provides a more reliable measure when there is an imbalance in the dataset between positive and negative cases. It is 83.08% in this evaluation.

Some insights from the analysis of the **BIC model**: - **The optimal threshold for maximizing profit is 0.38**. - The model achieves an accuracy of 89.05%, with a 95% confidence interval between 88.24% and 89.83%. - The sensitivity, or true positive rate, is 92.02%, indicating the model's ability to correctly identify true positives. - The positive predictive value is high at 94.85%, indicating that when the model predicts an account closure, it is correct 94.85% of the time. - The balanced accuracy, considering both sensitivity and

specificity, is 82.55%, reflecting a relatively good overall performance of the model in distinguishing between positive and negative cases.

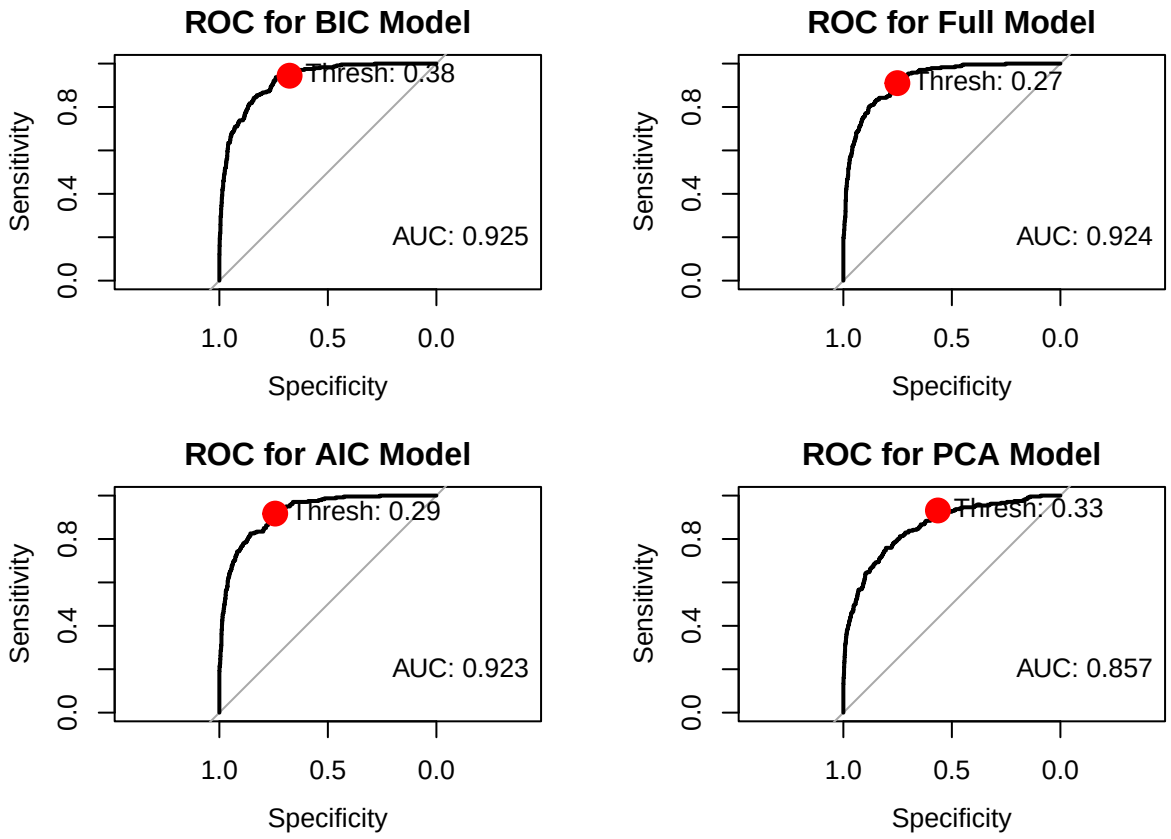
Some insights from the analysis of the **AIC model**: - **For the this model, an optimal threshold of 0.29**. - High positive predictive value (PPV) of 95.50% indicates a strong probability of correctly identifying positive cases. - Sensitivity of 90.49% demonstrates the model's effectiveness in accurately detecting positive instances.

Some insights from the analysis of the **PCA model**: - **The optimal threshold for maximizing financial gain with the PCA model is 0.33**. - The model achieves an accuracy of 86.46%, indicating its overall effectiveness in classifying instances correctly. - Sensitivity (true positive rate) stands at 91.22%, suggesting a strong ability to identify true positives, albeit with a specificity (true negative rate) of 60.79%, indicating a lower capability in identifying true negatives. - The positive predictive value (PPV) is notably high at 92.62%, denoting a high correctness rate when predicting account closures. - The balanced accuracy, considering both sensitivity and specificity, is 76.01%, indicating a fair overall performance in distinguishing between positive and negative cases.

ROC and AUC comparison

At this point we create a function called `plot_auc` that will help us compare the performance of different models by generating ROC curves and calculating their AUC values. This function will take predictions and actual labels as inputs and plot the ROC curve, annotating it with the AUC value. Additionally, if we specify an optimal threshold (that we found in the previous steps), it will mark that point on the curve.

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

## Setting direction: controls < cases

## Area under the curve: 0.9246

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

## Area under the curve: 0.9242

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

## Area under the curve: 0.9234

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




```
## Area under the curve: 0.8575
```

We've generated ROC curves for the BIC, Full, AIC, and PCA models. Here are the AUC values for each model: - **BIC Model**: AUC = **0.925** - **Full Model**: AUC = **0.924** - **AIC Model**: AUC = **0.923** - **PCA Model**: AUC = **0.857**

Which one maximizes gain?

```
gains <- c(
  logistic.full = calculate_financial_impact(optimal_threshold_full, pred.full.raw, data.val$Closed_Acc
  logistic.aic = calculate_financial_impact(optimal_threshold_aic, pred.aic.raw, data.val$Closed_Accoun
  logistic.bic = calculate_financial_impact(optimal_threshold_bic, pred.bic.raw, data.val$Closed_Accoun
  logistic.pca = calculate_financial_impact(optimal_threshold, pred.pca.raw, data.val$Closed_Account)
)
max_gain_model <- names(gains)[which.max(gains)]
cat("The model with the maximum gain is:", max_gain_model, "with a gain of", gains[max_gain_model], "\n
```

```
## The model with the maximum gain is: logistic.aic with a gain of 60110
```

These lines of code compute the financial impact for each model using the optimal threshold previously determined. They then compare the gains from each model and identifies the model with the highest gain. The result we obtained at the end of the analysis is that the **logistic.aic model** is the one that maximizes the gain.

‘‘‘