Data Analysis for Business Project

Michele Turco

2024-04-09

Introduction

The dataset contains information about bank customers. The goal is to predict whether a customer will close his credit card account, in that case the bank may try to propose more convenient services and prevent the client from leaving the bank.

Setup

In order to run all the chunks below, the following libraries need to be imported. However, for the sake of simplicity, the code is not displayed in the generated pdf file.

```
library(ggplot2)
library(corrplot)
library(class)
library(pROC)
library(RColorBrewer)
```

Data Import

Correctly importing data is crucial, and setting colClasses = "character" during the import process can prevent R from automatically converting data types. This approach ensures that all information, such as leading zeros, is preserved exactly as it appears in the source file. For instance, when importing a CSV file without losing leading zeros or unintentionally converting data:

```
Data = read.csv2("./Dataset/bank_accounts_train.csv",
                 header = T,
                 sep = ",",
                 colClasses = "character")
```

Data Preprocessing

In first place, we drop the first line since it is a pure identifier and it has no impact on the goal of the analysis. Then, we describe the variables present in the dataset.

```
Data$CLIENTNUM = NULL
str(Data)
```

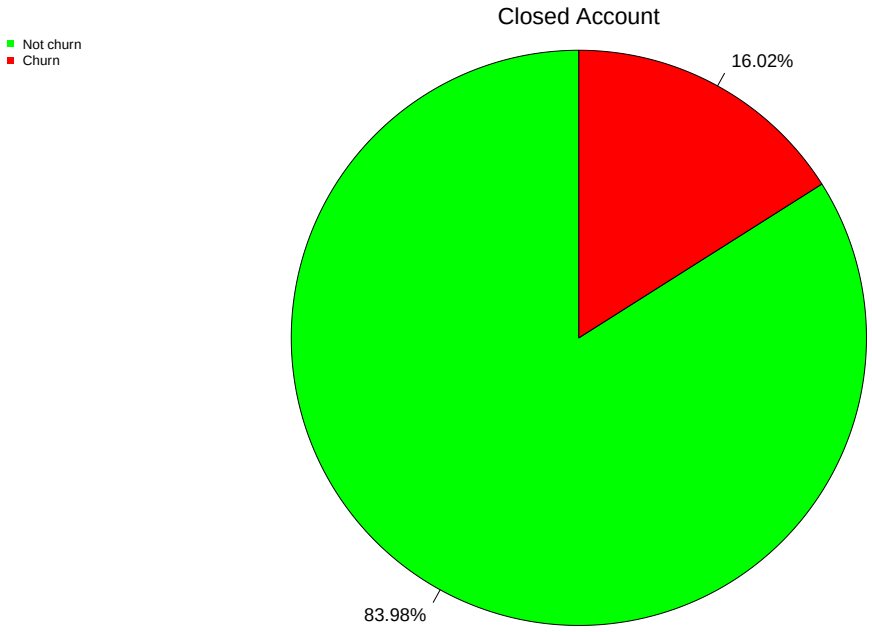
```
## 'data.frame':    7597 obs. of  20 variables:
##  $ Customer_Age            : chr  "38" "55" "36" "50" ...
##  $ Gender                  : chr  "F" "F" "F" "M" ...
##  $ Dependent_count         : chr  "2" "4" "1" "3" ...
##  $ Education_Level         : chr  "High School" "Graduate" "Unknown" "College" ...
##  $ Marital_Status          : chr  "Married" "Single" "Single" "Single" ...
##  $ Card_Category           : chr  "Blue" "Blue" "Blue" "Silver" ...
##  $ Months_on_book          : chr  "27" "44" "24" "36" ...
##  $ Total_Relationship_Count: chr  "4" "1" "2" "3" ...
##  $ Months_Inactive_12_mon  : chr  "1" "2" "1" "3" ...
##  $ Contacts_Count_12_mon   : chr  "2" "1" "2" "2" ...
##  $ Credit_Limit            : chr  "1830" "2638" "1735" "34516" ...
##  $ Total_Revolving_Bal     : chr  "0" "1423" "0" "2517" ...
##  $ Avg_Open_To_Buy         : chr  "1830" "1215" "1735" "31999" ...
##  $ Total_Amt_Chng_Q4_Q1    : chr  "736" "551" "74" "735" ...
##  $ Total_Trans_Amt         : chr  "1741" "4153" "2467" "7155" ...
##  $ Total_Trans_Ct          : chr  "43" "77" "35" "66" ...
##  $ Total_Ct_Chng_Q4_Q1     : chr  "654" "925" "346" "65" ...
##  $ Avg_Utilization_Ratio   : chr  "0" "539" "0" "73" ...
##  $ Income                  : chr  "169.77966777049005" "87.02969695441425" "140.2794818393886" "90.1
##  $ Closed_Account          : chr  "0" "0" "1" "1" ...
```

From the output, we infer that we have 15 numerical variables, 4 categorical variables and our target variable "Closed_Account". Next, we will convert the numerical variables to numeric types and the categorical variables to factors to ensure they are appropriately processed during our analysis."

```
variables = colnames(Data) # With this line we have a vector of our variables
categorical_variables <- c("Gender", "Education_Level", "Marital_Status", "Card_Category") # We select
target_variable = "Closed_Account"
numerical_variables <- setdiff(variables, c(categorical_variables, target_variable))
for (var in numerical_variables) {
  Data[[var]] = as.numeric(Data[[var]])
}

for (var in categorical_variables) {
  Data[[var]] = as.factor(Data[[var]])
}

Data$Closed_Account = as.factor(Data$Closed_Account)
```

EDA

Target Variable Distribution

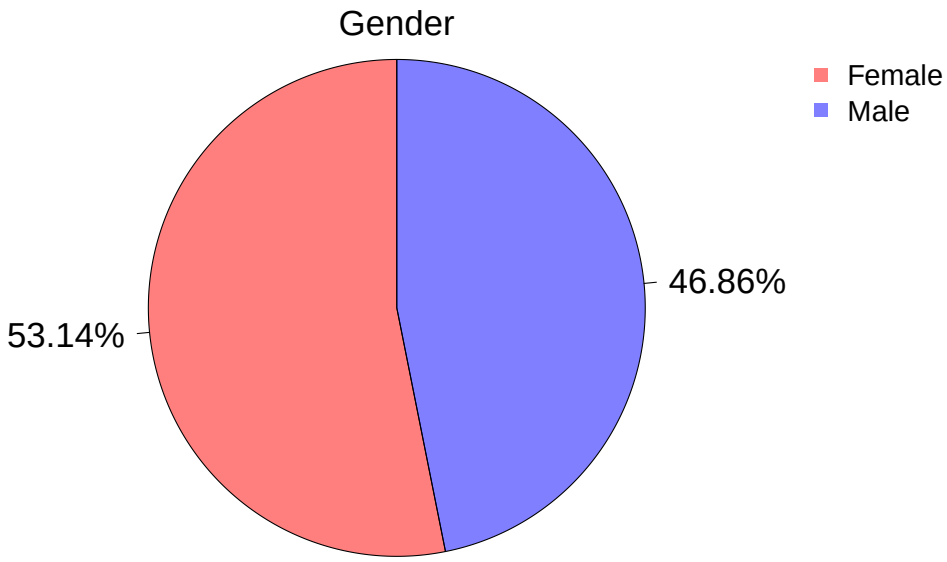


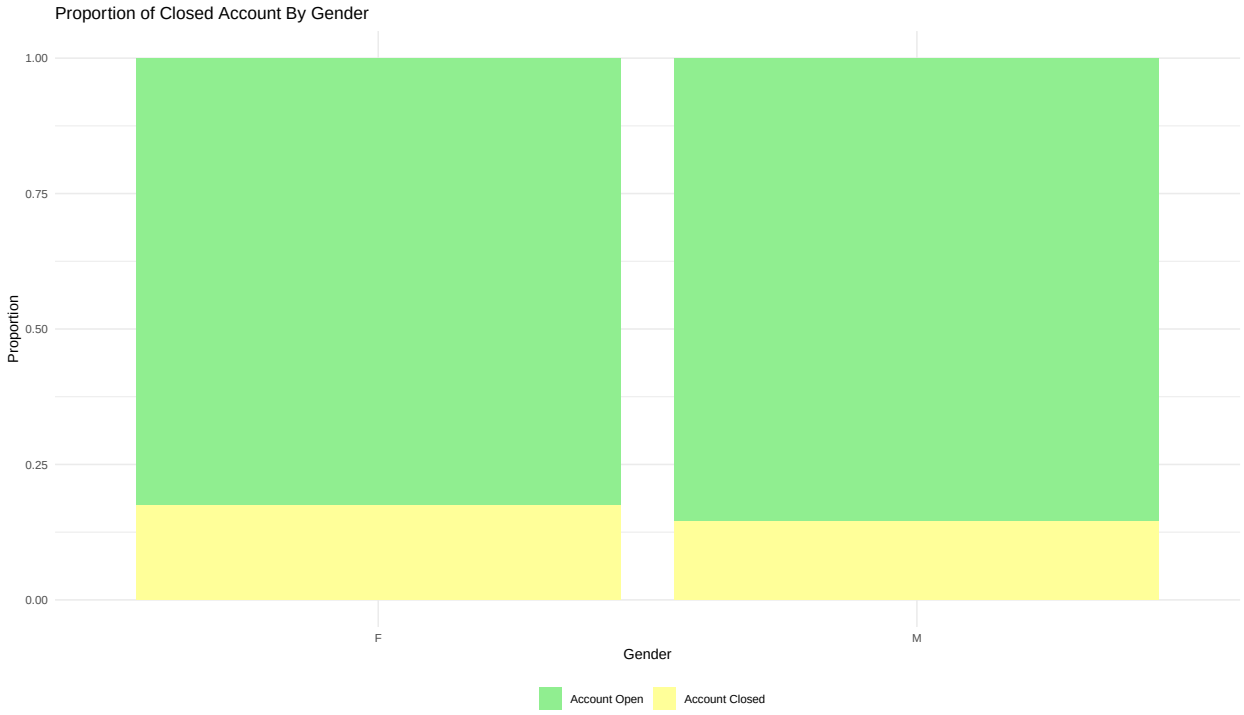
The pie chart illustrates the distribution of the binary categorical target variable 'Closed Account', which represents churn status. Two segments are visible: 'Not churn', constituting a majority with approximately 84% of the accounts, and 'Churn', representing about 16%. This significant imbalance indicates that far fewer customers have closed their accounts compared to those who haven't.

In a real-world context, this kind of target variable is critical for businesses to understand customer retention. 'Churn' signifies customers who have ended their relationship with the company, while 'Not churn' indicates customers who continue to engage with the company's services or products.

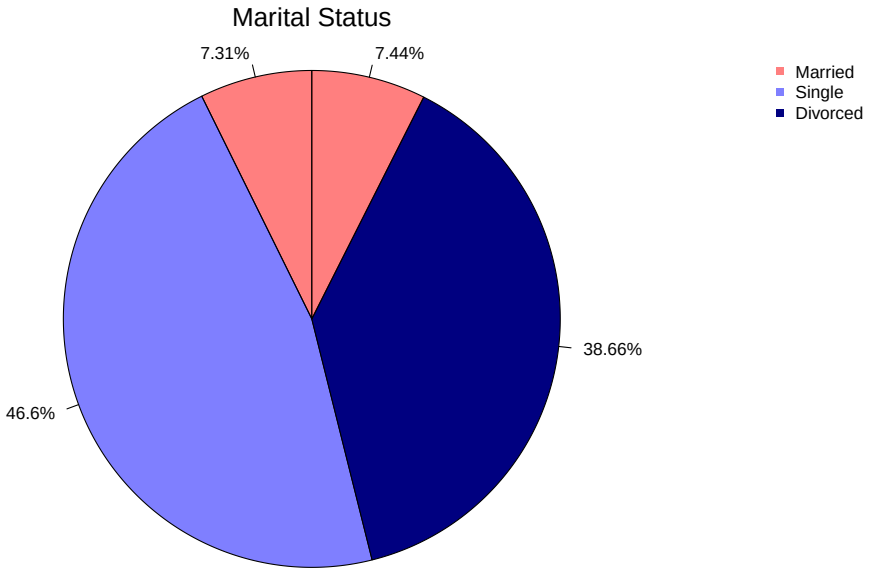
Categorical Variables exploration

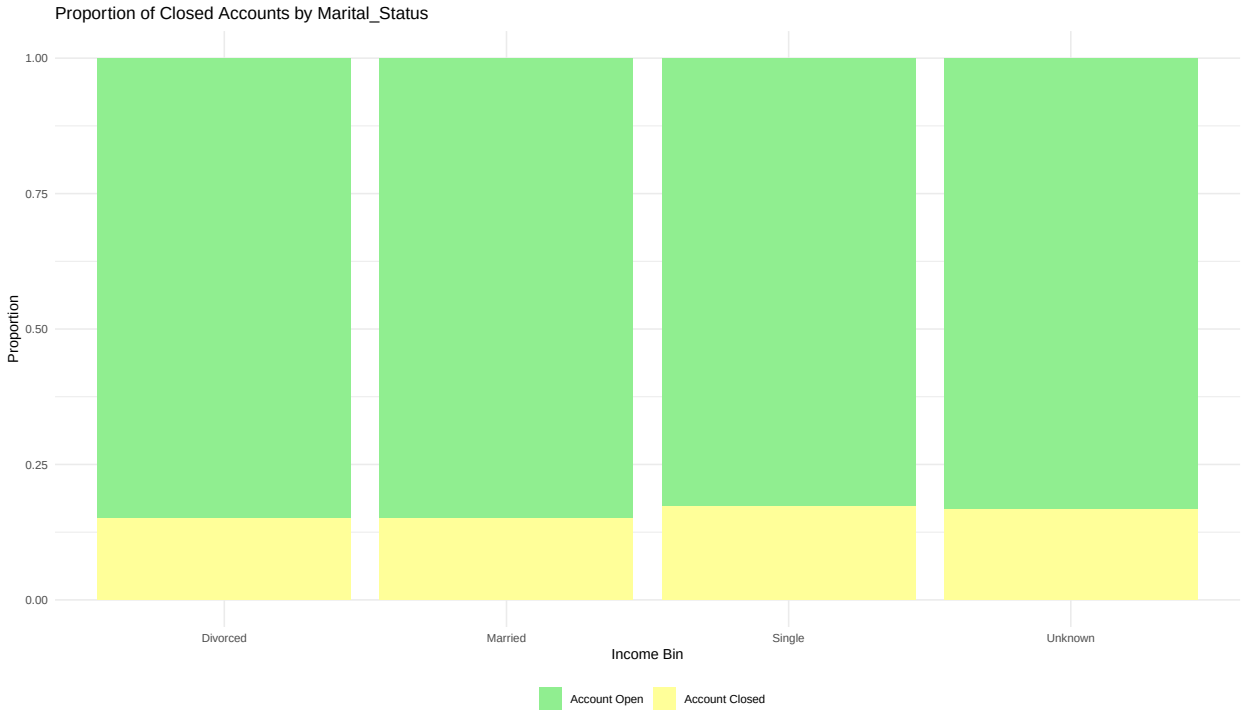
Gender



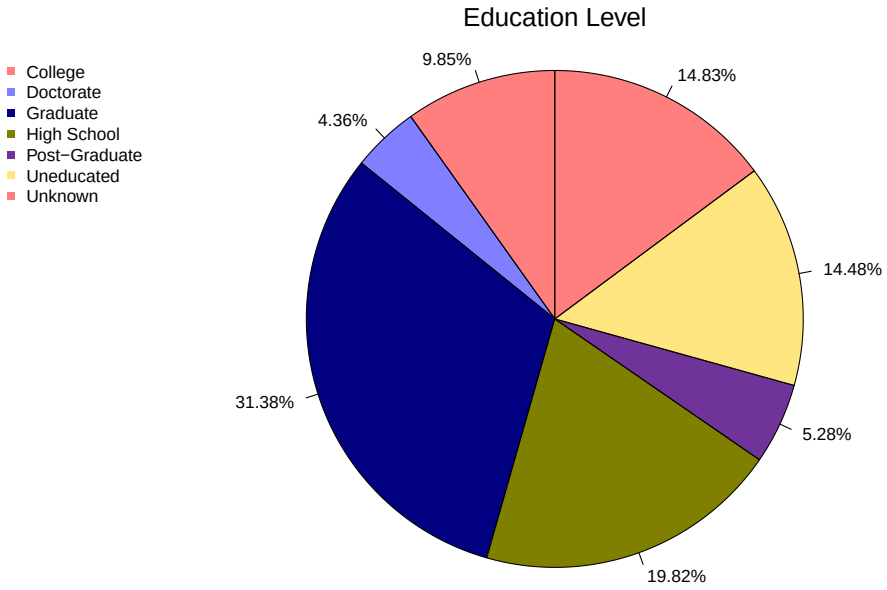


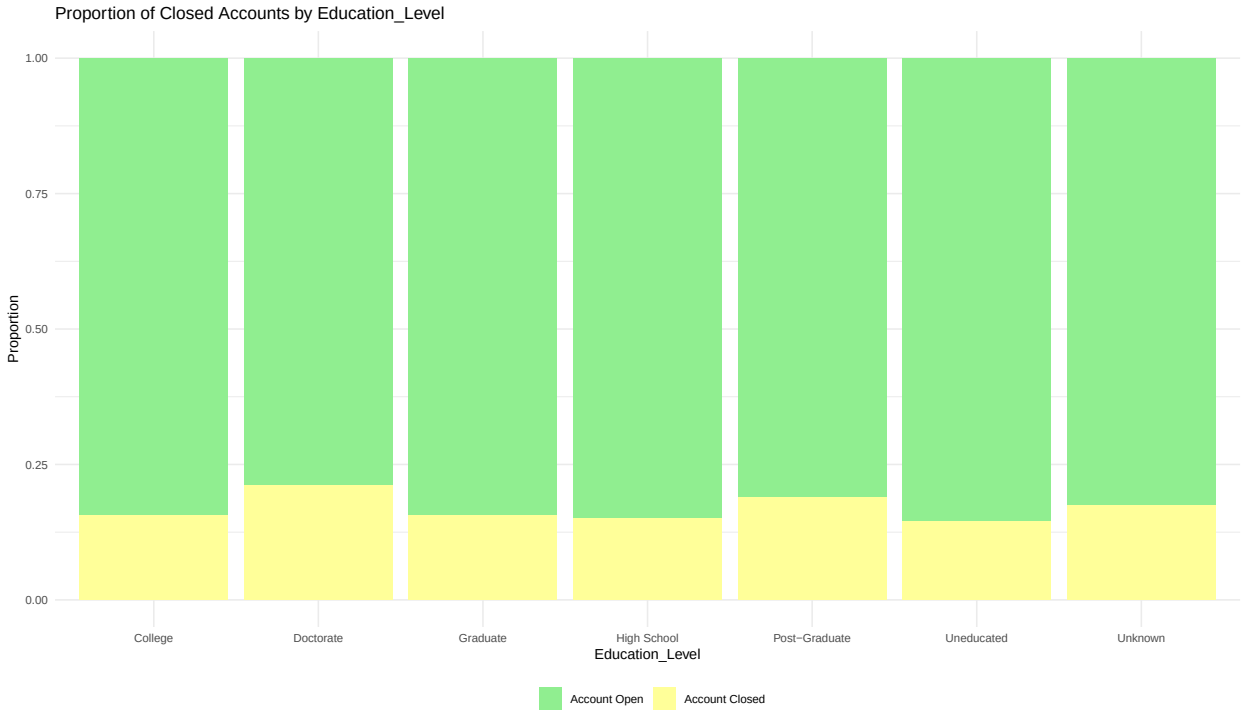
Marital Status



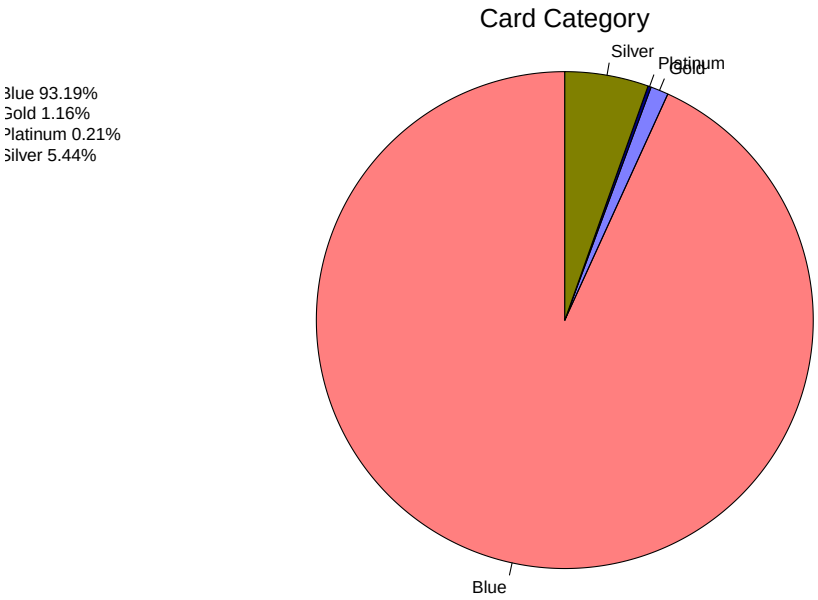


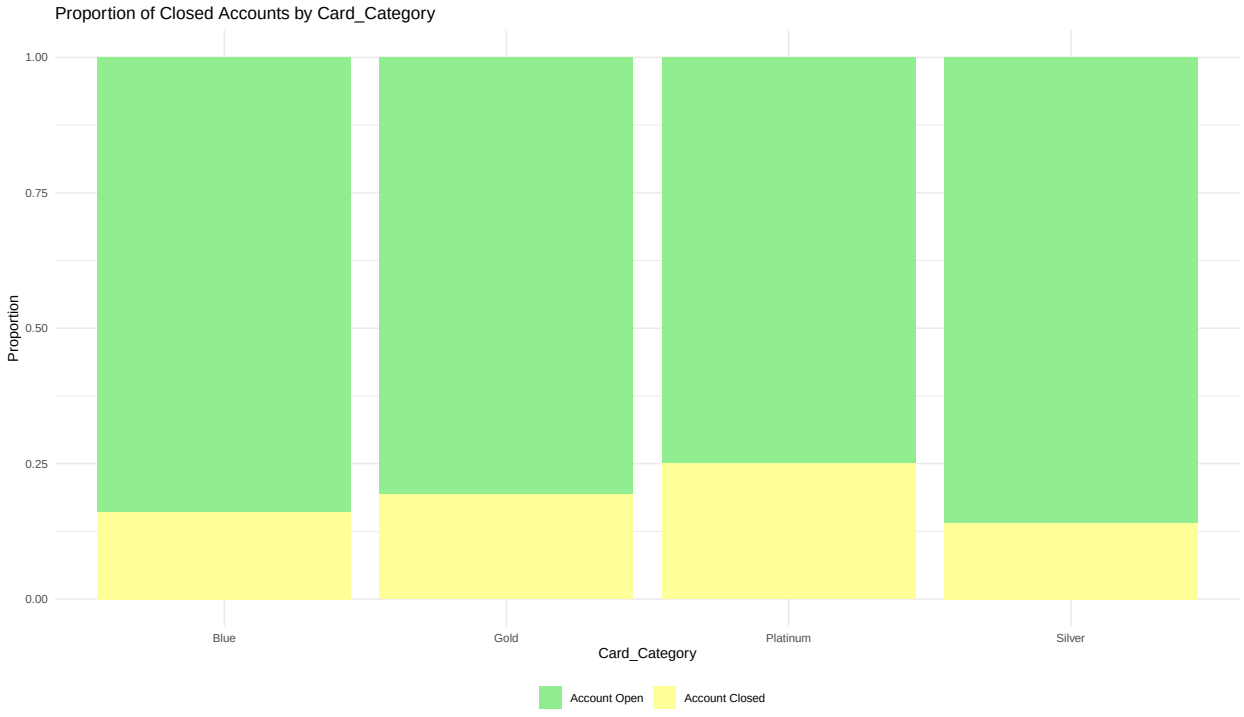
Education Level





Card Category



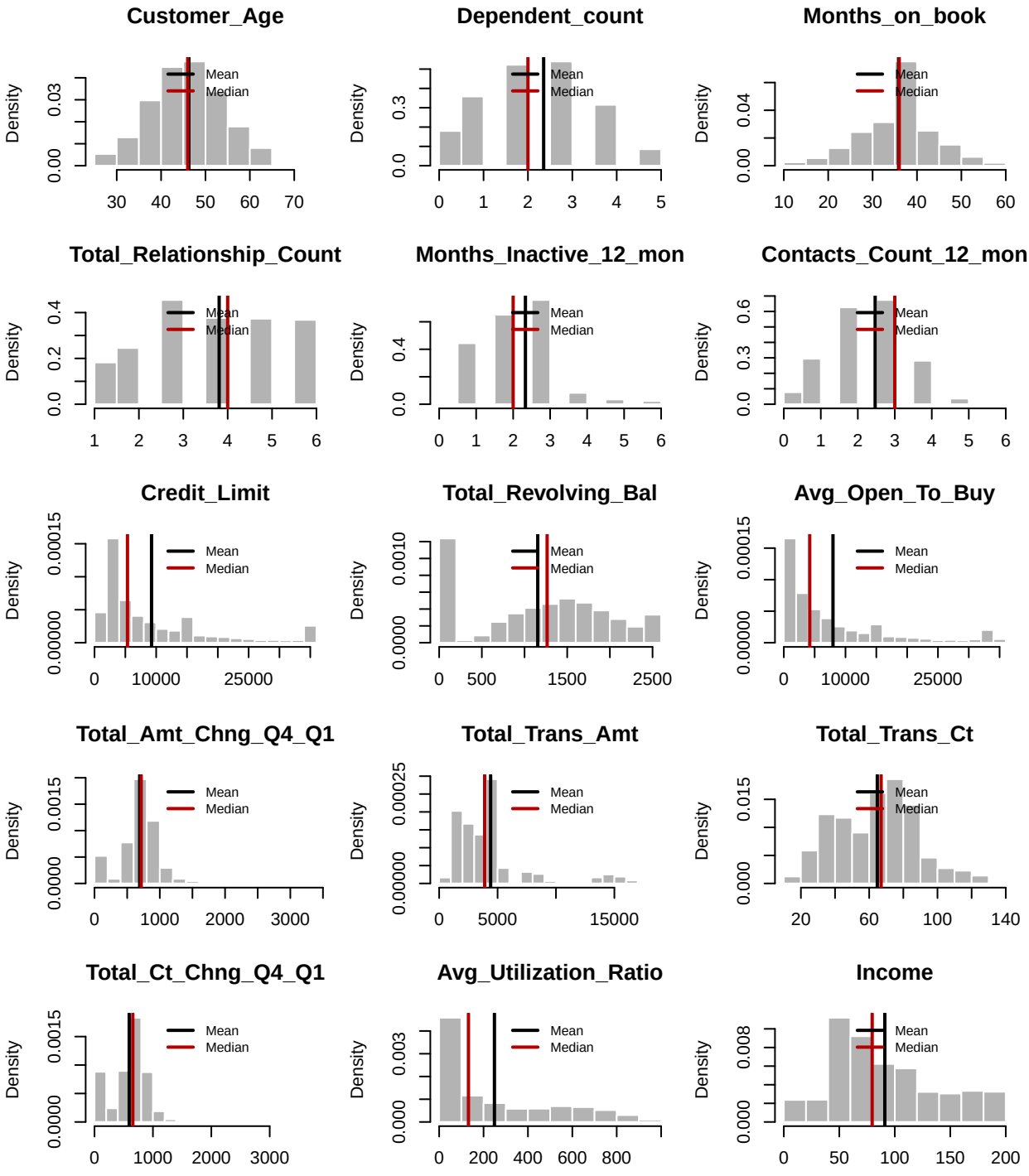


From the analysis of the categorical variables in our dataset, it's evident that there's a significant imbalance in the target variable — the number of people closing accounts is consistently below 20% across all categories. This skew could present challenges when building predictive models, as there's a smaller pool of closure events to learn from. Moreover, none of the categories across our variables show a disproportionate tendency towards account closure, indicating no single categorical predictor strongly influences churn. A point of concern is

the presence of 'Unknown' values within 'Educational_Level' and 'Marital_Status'. These entries, which represent a lack of data, might complicate our analysis. They introduce a level of uncertainty that could distort the models' predictions, making them less reliable. It's crucial to address these anomalies to mitigate their potential impact on the predictive performance of our models.

We can now proceed with a graphical exploration (including also the computation of basic stats) for our Numerical Variables.

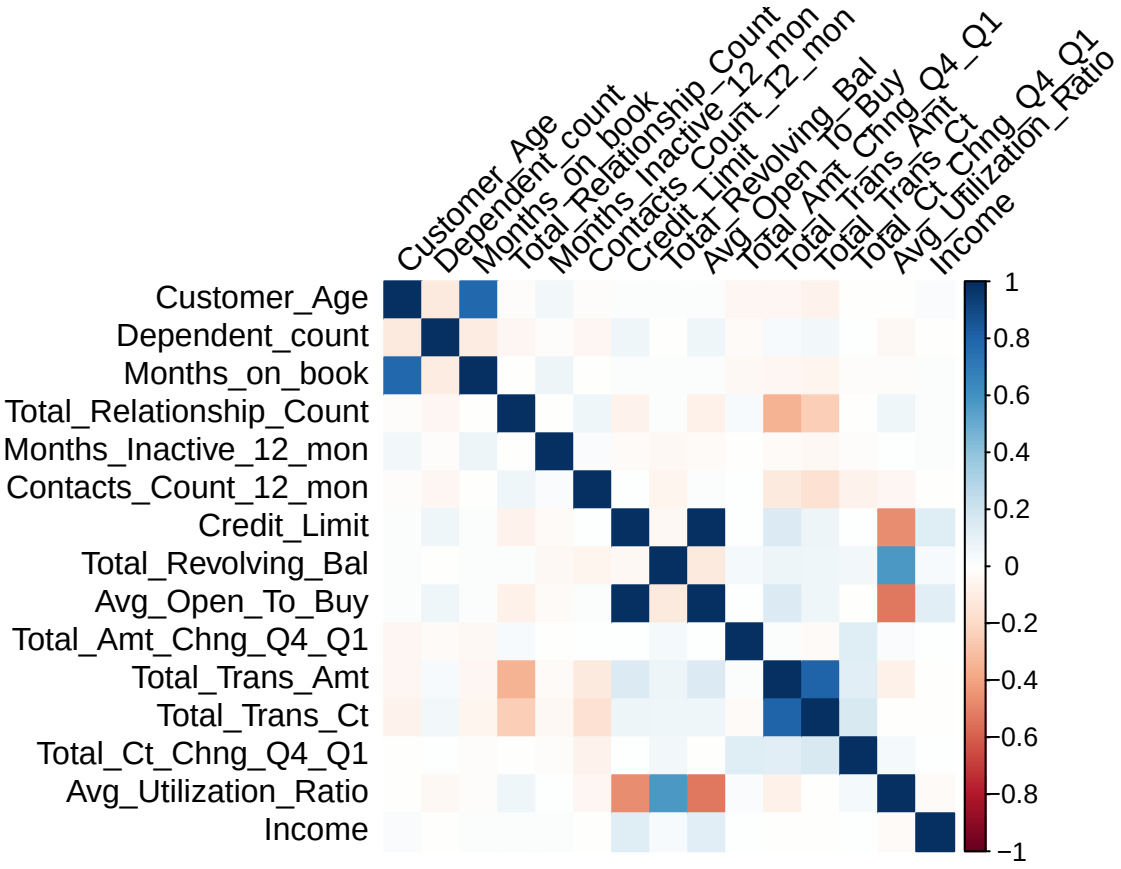
Graphical Exploration of Numerical Variables



The density plots suggest that most numerical variables in our dataset have symmetric distributions, as indicated by the mean and median being closely aligned. This symmetry is visually confirmed by the density plots. Only a few variables, like Credit_Limit, Avg_Open_To_Buy, and Avg_Utilization_Ratio, exhibit significant skewness, with tails stretching to the left or right. However, even for these skewed distributions, the mean and median remain relatively close to one another, implying that the skewness is not excessively pronounced. This nearness of mean and median values across most variables indicates a data set with generally well-behaved, symmetric distributions.

Correlation Matrix

In the following chunk, we analyzed the correlation between numerical variables plotting a correlation matrix.

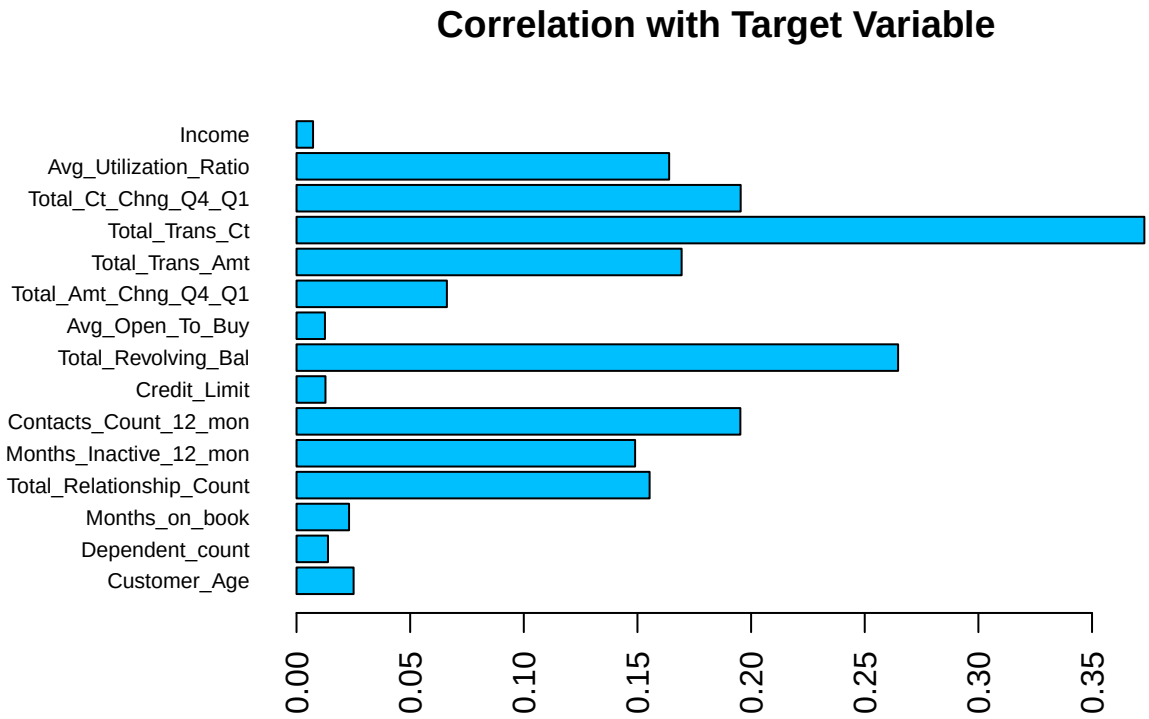


From the correlation matrix we can clearly see some relevant correlations that make us think of a possible deletion. In particular we are searching for variables that are highly correlated so that we can discard one of them in order to avoid multicollinearity. It is worth to considering the relation between "Months_on_Book" and "Customer_Age","Total_Trans_Ct" and "Total_Trans_Amt", "Credit_Limit" and "Avg_Open_to_Buy". The most relevant one is the relation between "Credit_Limit" and "Avg_Open_to_Buy".However, we will just take into account these findings without any further modification to the original dataset.

Correlation with target variable

The provided plot illustrates the correlations between numerical variables and the target variable, offering a preliminary insight that can be crucial before building predictive models. Such an analysis helps in pinpointing variables that might have limited predictive power regarding the target variable. By identifying

these less influential variables early on, we can streamline the modeling process by focusing on more significant predictors.



The plot reveals that 'Income' has a notably low correlation with the target variable. This suggests that 'Income' may not be a valuable predictor for determining the category of the target variable, indicating that it might have limited influence in the subsequent model. This is something we will take into considerations for the future.

Dealing with missing values

Firstly, in order to have a better understanding of the number of data instances that contains Unknown values, we convert them in NA values and we directly count them using basic R commands.

```
#Rename missing values with NA notation
Data[Data == 'Unknown'] <- NA
```

```
## Rows with NA before preprocessing: 1613
```

The dataset contains a considerable number of rows with missing values, predominantly within two variables. Discarding these rows would significantly reduce our dataset and potentially introduce bias by removing valuable information. Therefore, a preferable approach is to replace each 'Unknown' value with the mode of its respective variable. This strategy maintains the dataset's integrity and avoids the information loss that would occur with row deletion. The mode is the most frequent value in a variable, ensuring that we impute a statistically common category, which preserves the underlying distribution and relationships between variables. Moreover, using the mode for imputation is particularly suitable for categorical data, where mean or median replacements are not applicable

```r
#Basic function to compute mode
getSimpleMode <- function(x) {
  tbl <- table(x)
  mode_value <- names(tbl[tbl == max(tbl)])[1]
  return(mode_value)
}


for (var in categorical_variables) {
  if (any(is.na(Data[[var]]))) {
    Data[[var]][is.na(Data[[var]])] <- getSimpleMode(Data[[var]])
  }
}
```

```
## Rows with NA after preprocessing: 0
```

This strategy maintains the dataset's integrity and avoids the information loss that would occur with row deletion. The mode is the most frequent value in a variable, ensuring that we impute a statistically common category, which preserves the underlying distribution and relationships between variables. Moreover, using the mode for imputation is particularly suitable for categorical data, where mean or median replacements are not applicable.

3 Logistic Regression Model to estimate effects of Income and Gender on Closed_Account

We start by fitting the logistic regression model to our data, carefully examining the model summary. In this instance, we do not discard the possibility that 'Income' and 'Gender' might interact. Therefore, our model includes both these predictors and their interaction term to capture any potential combined effect on the target variable. This approach ensures we consider both individual and interactive influences in our analysis.

```r
# Fit logistic regression model
set.seed(1)
gender_income_model <- glm(Closed_Account ~ Income * Gender,
                           family = binomial(link = "logit"),
                           data = Data)

# View model summary
summary(gender_income_model)
```

```
##
## Call:
## glm(formula = Closed_Account ~ Income * Gender, family = binomial(link = "logit"),
##     data = Data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6211  -0.6185  -0.5764  -0.5443   2.0382
##
## Coefficients:
```

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

Intercept: The estimated probability of an account being closed when Income is 0 and Gender is Female is -1.5474767. This is statistically significant with a very low p-value (< 2e-16), indicating strong evidence against the null hypothesis of no effect.This is something that we could have expected considering the highly unbalanced distribution of the target variable, as stated in the EDA process.

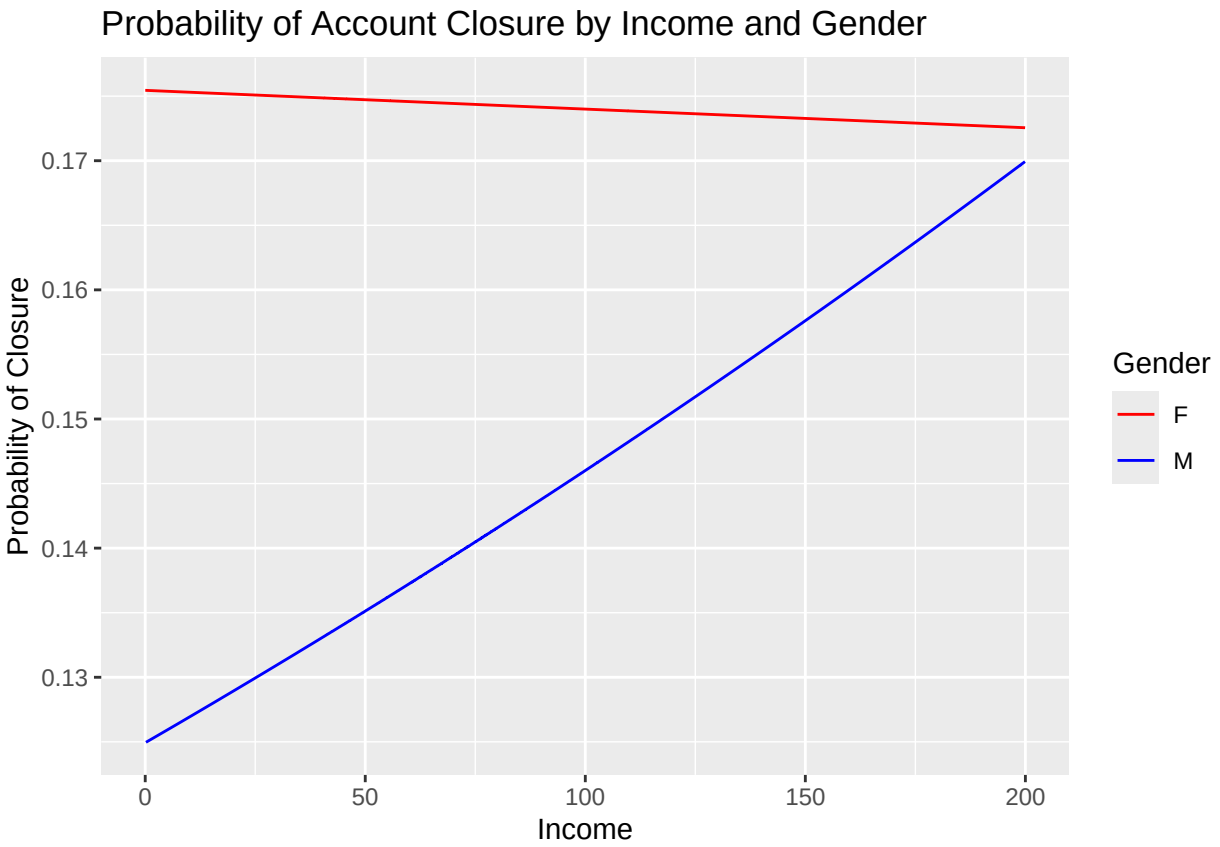
Income: The coefficient for Income is -0.0001008 and is not statistically significant (p-value 0.89207), suggesting that this variables does not have a strong effect on the probability of account closure. It is important to state that this hypothesis was already been made during the EDA, when we saw that the correlation between the target variable and "Income" was one of the lowest.

Gender: The coefficient for GenderM is -0.3990676, indicating that being male is associated with a lower likelihood of closing an account compared to being female. In addition, this effect is statistically significant (p-value 0.00577), suggesting it's not due to random chance. This is a surprising outcome since even if there is a slightly higher proportion of churn for females (also highlighted in our EDA), it was not expected to have this impact.

(Income:GenderM): This coefficient measures whether the effect of income on the likelihood of closing an account differs between males and females. Since it is not statistically significant (p= 0.17440), we cannot conclude that the effect of income on the probability of account closure is different for males versus females.

We then proceed by computing predictions and plotting the respective regression lines:

```r
# Compute and store predicted probabilities
predicted_probabilities_logistic <- predict(gender_income_model, newdata = Data, type="response")

Data$predicted_probabilities <- predict(gender_income_model, newdata = Data, type="response")
```



The plot depicts logistic regression lines for the probability of account closure, separated by gender. It visually reinforces the observation that males are generally less likely to close their accounts compared to females. Also, the plot shows different slopes for males and females, hinting at a varying impact of income on the chance of closing an account. However, statistically, the difference isn't significant, suggesting that what we see might not reflect a real-world difference. The way the plot is drawn, without scaling income evenly, might make the slopes look more different than they actually are. So, while the lines look distinct, statistically, income influences account closure similarly for both genders.

4 KNN

Using set.seed() guarantees that anyone rerunning the analysis will get the same results, making our work reproducible and reliable. It's essential for any steps that involve random choices, like dividing data into training and test sets.

```
# Seed setting
set.seed(10)
```

The initial step involves dividing the training data into separate training and validation sets. Once the split is complete, it's essential to verify that the target variable's distribution has stayed consistent across both the new training and the validation sets.

```
# Split Dataset
id_train <- sample(1:nrow(Data), size = 0.75*nrow(Data), replace = F)
train_data <- Data[id_train,]
val_data <- Data[-id_train,]
```

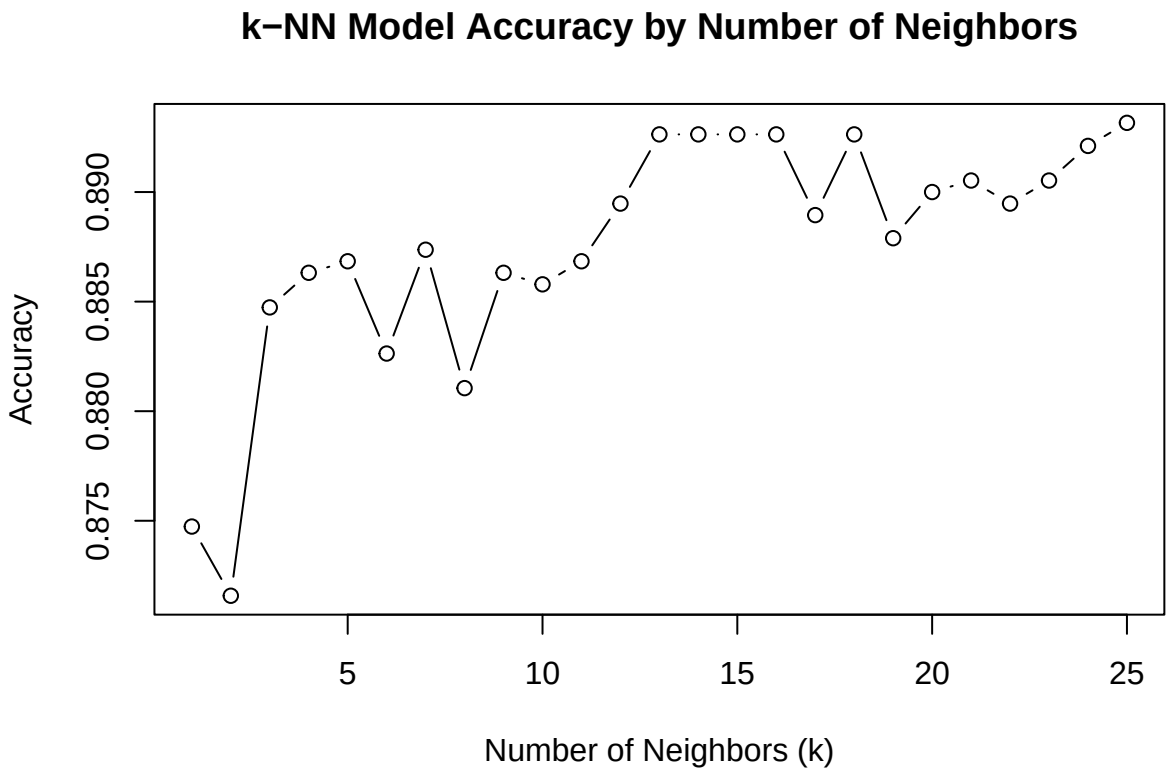
```
## Distribution of the target variable in the original set:

## Counts:

##
##    0    1
## 6380 1217

## Proportions:

##
##         0         1
## 0.8398052 0.1601948

## Distribution of the target variable in the train set:

## Counts:

##
##    0    1
## 4759  938

## Proportions:

##
##         0         1
## 0.8353519 0.1646481

## Distribution of the target variable in the validation set:

## Counts:

##
##    0    1
## 1621  279

## Proportions:

##
##         0         1
## 0.8531579 0.1468421
```

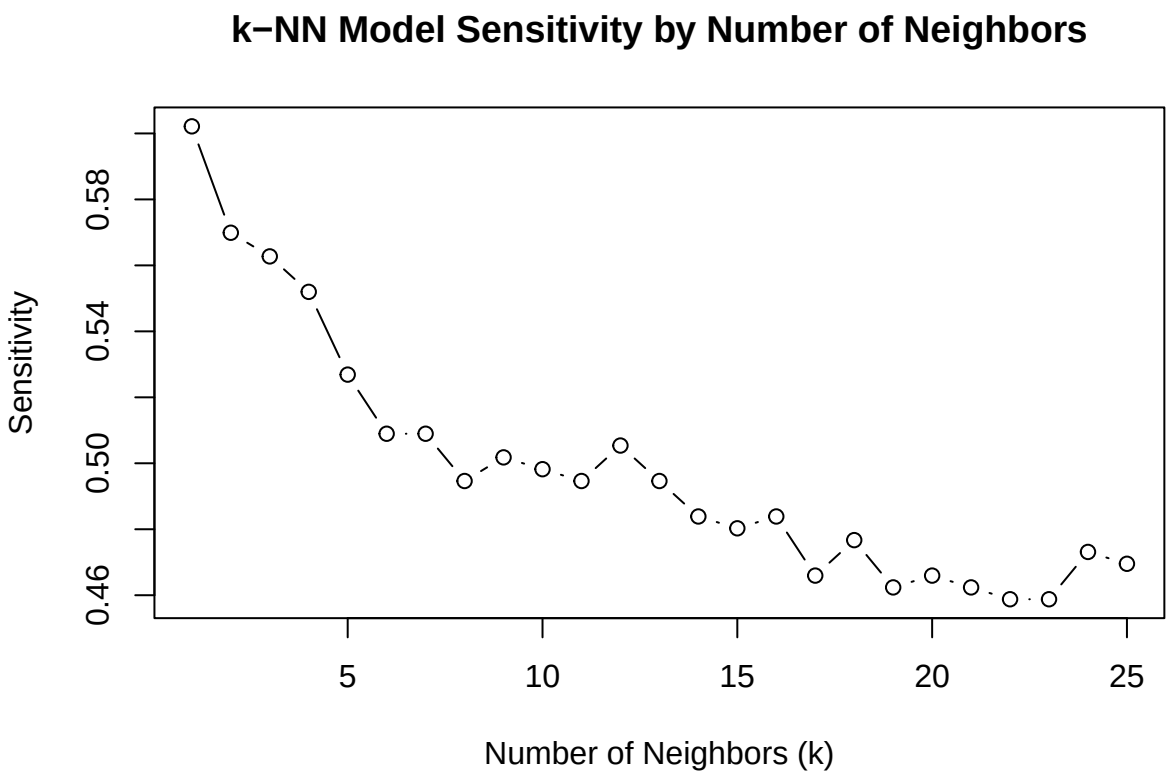
In this scenario, the distribution of the target variable is quite similar across the different sets.

KNN is a model that requires both numerical and scaled data. The first prerequisite is satisfied in any case, being Total Trans Amt and Total Trans Ct both numerical variables. Initially, in order to find the best number of neighbors k to use in k-NN, we instantiated a range for k in which it was possible to perform a greedy search. As a second step, the model was trained with all the different hyper parameters in the range and the associated accuracy and sensitivity obtained on the validation set were stored. * In order to compute the sensitivity we used a code chunk that is not visible in the report.

```r
# Set a range for k
k_values <- 1:25
accuracy_scores <- numeric(length(k_values))
sensitivity_scores <- numeric(length(k_values))

# Loop over k values
for (k in k_values) {
  set.seed(100) # for reproducibility
  knn_pred <- knn(train = train_data[, c("Total_Trans_Amt", "Total_Trans_Ct")],
                  test = val_data[, c("Total_Trans_Amt", "Total_Trans_Ct")],
                  cl = train_data$Closed_Account,
                  k = k)

  # Calculate accuracy
  accuracy_scores[k] <- sum(val_data$Closed_Account == knn_pred) / length(knn_pred)
  # Calculate sensitivity
  sensitivity_scores[k] <- calculate_sensitivity(val_data$Closed_Account, knn_pred)
}
```

We can now plot and comment the result






```
## The best number of neighbors for accuracy is: 25 with an accuracy of: 0.8931579
```

```
## The second-best number of neighbors for accuracy is: 13 with an accuracy of: 0.8926316
```

```
## The best number of neighbors for sensitivity is: 1 with a sensitivity of: 0.6021505
```

```
## The second-best number of neighbors for sensitivity is: 2 with a sensitivity of: 0.5698925
```

Given the high imbalance in our dataset, we've prioritized sensitivity in our selection criteria because minimizing false negatives—thus reducing missed positives—is crucial for our analysis. Especially in a banking context, identifying a false positive (predicting a customer will leave who doesn't) is preferable to overlooking a false negative (failing to predict a customer's departure). Therefore, the plot suggests selecting the best k value based on sensitivity, which is our priority. Yet, the optimal choice likely involves balancing sensitivity with accuracy, favoring the former without disregarding the latter.

5 Best Model Selection

In the first place, we consider as baseline the models in which no variable is used and all variables are included. Firstly, we build the model with each variable:

```
logit_fit_baseline <- glm(Closed_Account ~ .,
                  family = "binomial",
                  data = train_data)
```

Then, we consider the model without any variable:

```
logit_fit_0 <- glm(Closed_Account ~ 1,
                   family = "binomial",
                   data = train_data)
```

We then test the hypothesis of equivalence between the two models:

```
logit_fit0 <- glm(Closed_Account ~ 1,
                   family = "binomial",
                   data = train_data)
anova(logit_fit_0, logit_fit_baseline, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Closed_Account ~ 1
## 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 + Avg_Open_To_Buy + 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      5696     5096.5
## 2      5670     2824.7 26   2271.8 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The difference in deviance between the full model and the null model is 2233.5, with 26 degrees of freedom, and this difference is highly significant ($p < 0.00000000000000022$). This significant result suggests that including predictors in the full model offers a much better fit to the data compared to the null model, which includes no predictors.

Variable Selection with AIC and BIC

Despite the previous significant results, the model currently includes many variables, some of which may not significantly contribute to its predictive power. To streamline the model and enhance its performance, we can apply variable selection techniques to retain only the most impactful predictors.

Stepwise variable selection (based on AIC)

Forward selection:

```
# Forward
logit_fit_aic1 <- step(glm(Closed_Account ~ 1,
                           family = "binomial",
                           data = train_data),
                       scope = formula(logit_fit_baseline),
                       direction = "forward")
```

The variable excluded by the forward selection, which obtains the best (least value) AIC=2867.02, are the following: Income, Customer_Age, Months_on_book, Avg_Open_To_Buy, Credit_Limit, Avg_Utilization_Ratio, Card_Category and Gender.

Backward selection:

```
# Backward
logit_fit_aic2 <- step(logit_fit_baseline,
                       direction = "backward")
```

The variable excluded by the backward selection, which obtains the best (least value) AIC=2880.5: are the following: Income, Customer_Age, Months_on_book, Avg_Open_To_Buy, Credit_Limit, Avg_Utilization_Ratio, Card_Category and Gender. In this case, the two selection process eliminates the same variable and obtain the same AIC score.

Both directions:

```
# Both directions
logit_fit_aic3 <- step(logit_fit_baseline,
                       direction = "both")
```

The variable excluded by the selection in both directions, which obtains the best (least value) AIC=2867.02: are the following: Income, Customer_Age, Months_on_book, Avg_Open_To_Buy, Credit_Limit, Avg_Utilization_Ratio, Card_Category and Gender.

As a general principle, in order to select the best model, we identify the one with the lowest number of covariates:

```
print(length(coefficients(logit_fit_aic1)))
```

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

```
print(length(coefficients(logit_fit_aic2)))
```

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

```
print(length(coefficients(logit_fit_aic3)))
```

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

However, in this specific case, every procedure basing on AIC gives the best (lest) AIC score of 2867.02, excluding the same predictors.

Stepwise variable selection (based on BIC)

Forward selection:

```
# Forward
logit_fit_bic1 <- step(glm(Closed_Account ~ 1,
                           family = "binomial",
                           data = train_data),
                       scope = formula(logit_fit_baseline),
                       direction = "forward",
                       k = log(nrow(train_data)))
```

The variables excluded by the forward selection, which obtains the best (least value) AIC=2953.44, are the following: Income, Customer_Age, Months_on_book, Avg_Open_To_Buy, Credit_Limit, Avg_Utilization_Ratio, Card_Category and Educational_Level. The result is really similar to the one obtained in the selection based on aic, except for the fact that Educational Level is excluded instead of Gender.

Backward selection:

```
# Backward
logit_fit_bic2 <- step(logit_fit_baseline,
                       direction = "backward",
                       k = log(nrow(train_data)))
```

The variable excluded by the backward selection, which obtains the best (least value) AIC=2953.44: are the following: Income, Customer_Age, Months_on_book, Avg_Open_To_Buy, Credit_Limit, Avg_Utilization_Ratio, Card_Category and Educational_Level.

Both directions:

```
# Both directions
logit_fit_bic3 <- step(logit_fit_baseline,
                       direction = "both",
                       k = log(nrow(train_data)))
```

The variable excluded by the selection in both directions, which obtains the best (least value) AIC=2953.44: are the following: Income, Customer_Age, Months_on_book, Avg_Open_To_Buy, Credit_Limit, Avg_Utilization_Ratio, Card_Category and Educational_Level.

```
print(length(coefficients(logit_fit_bic1)))
```

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

```
print(length(coefficients(logit_fit_bic2)))
```

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

```
print(length(coefficients(logit_fit_bic3)))
```

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

Also in this case, every procedure basing on BIC gives the best (lest) AIC score 2953.44 (similar to best AIC output by the procedure basing on AIC itself). Consequently, we can consider each model as equal with each other.

After thorough variable analysis using AIC and BIC we determined that certain variables do not contribute meaningfully to our model's objectives. Specifically, we identified "Customer_Age", "Months_on_book", "Card_Category", "Income", "Credit_Limit", "Avg_Utilization_Ratio", "Avg_Open_To_Buy", and "Education_Level" as extraneous. It is important to highlight that our exploratory data analysis already revealed correlations between some variables that suggested redundancy (multicollinearity). For instance, "Avg_Open_To_Buy" and "Credit_Limit" were closely linked, as were "Months_on_book" and "Customer_Age". Additionally, most of the variable excluded such as "Income","Credit_Limit", "Customer_Age" and "Avg_Open_To_Buy" also showed a smaller correlations with our target variable.

PCA

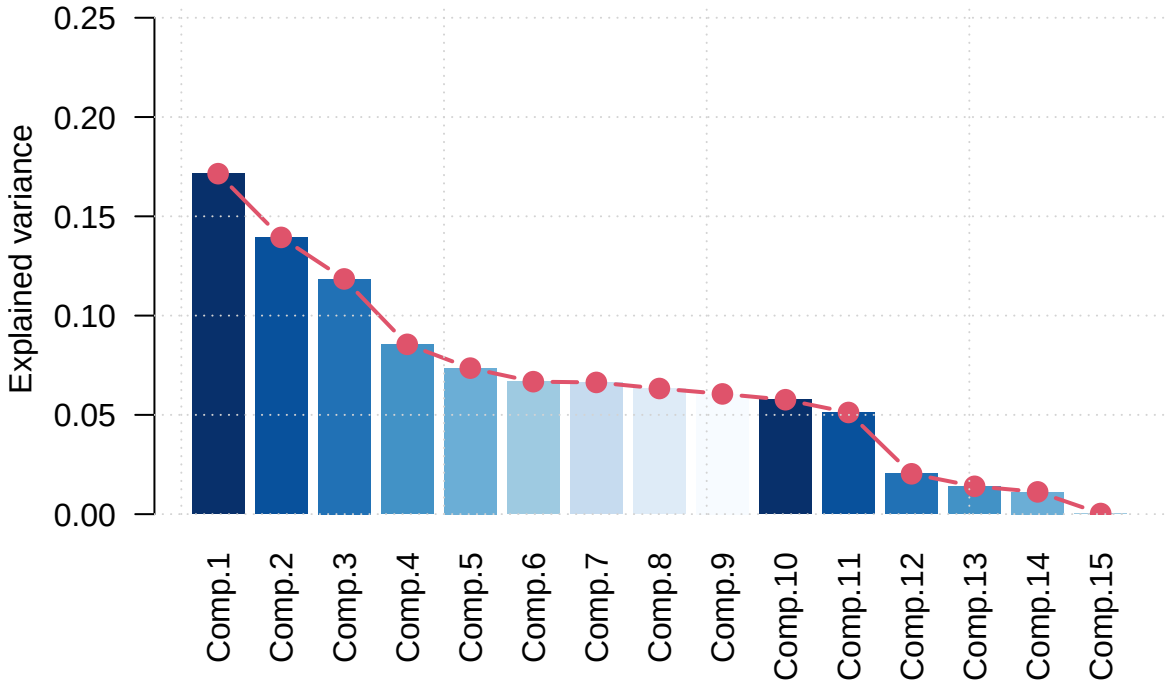
To conduct a Principal Component Analysis (PCA), it's essential to focus solely on numerical variables from the training data. Given the importance of the covariance matrix's scale in PCA, implementing a scaling procedure is crucial to ensure meaningful analysis and results.

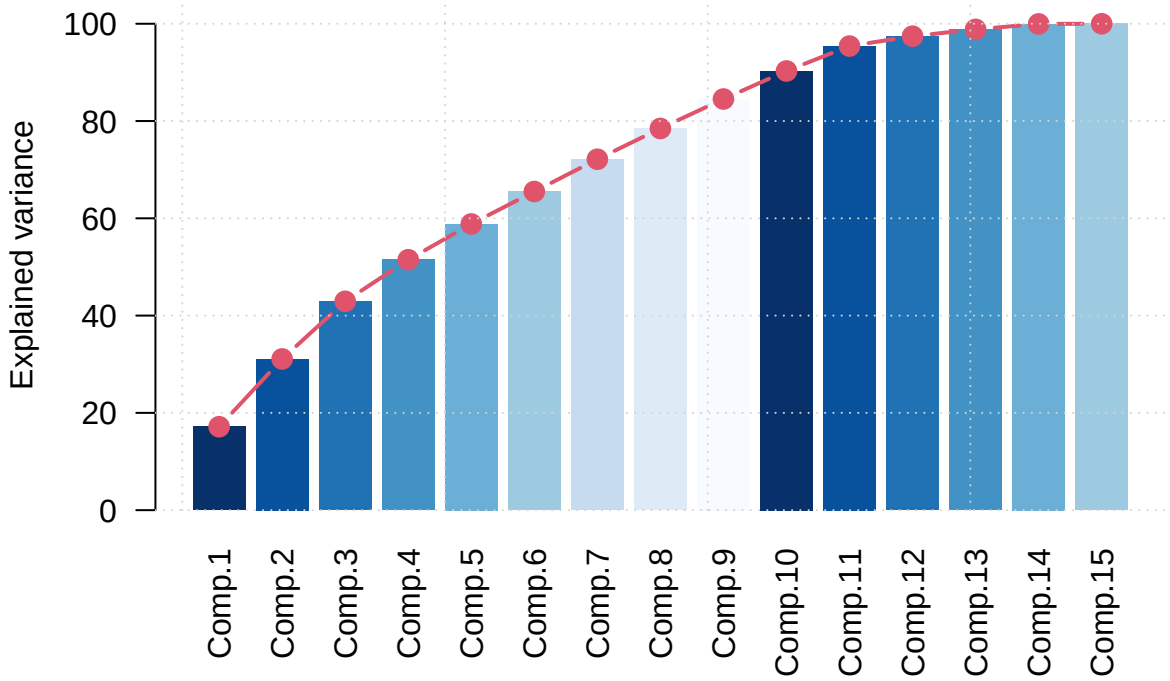
```
# Selecting Numerical DF
train_data_Numerical = train_data[numerical_variables]

cov(train_data_Numerical)

# Scaling Data
train_data_Numerical_Scaled= scale(train_data_Numerical)

cov(train_data_Numerical_Scaled)
```

The PCA is then executed on the pre-scaled numerical training data. After this, we compute the variance explained by each principal component both in absolute terms and as a percentage of the total variance. This helps in understanding the importance of each principal component in explaining the variation in the data.

```
pca <- princomp(train_data_Numerical_Scaled, cor = T, scale = F) #already scaled

pca$loadings

# Calculating the variance (in percentage) explained by each PCA component
pca_var <- pca$sdev^2
pca_var_percent <- pca_var / sum(pca_var)
```

After the computation, we are able to plot the results:


Explained variance
0.25
0.20
0.15
0.10
0.05
0.00
Comp.1
Comp.2
Comp.3
Comp.4
Comp.5
Comp.6
Comp.7
Comp.8
Comp.9
Comp.10
Comp.11
Comp.12
Comp.13
Comp.14
Comp.15



The barplot indicates that the first 11 principal components account for the majority of the total variance in the dataset, following the "elbow rule." This rule suggests selecting components up to the point where the plot of explained variance against the number of components levels off, forming an "elbow." The first component, contributing 17% of the explained variance, is the most significant, with subsequent components adding progressively less. In this way, most of the data's variability is explained while the dimensionality is minimized.

Hence, we proceed in building our model:

```
# Validation data is scaled as the train data
val_data_Numerical = val_data[numerical_variables]
val_data_Numerical_Scaled= scale(val_data_Numerical)

# Predicting scores for the validation and test set using the PCA model
pca_data_train <- predict(pca, train_data_Numerical_Scaled)
pca_data_val <- predict(pca, val_data_Numerical_Scaled)

# Selecting the first 11 principal components
pca_data_train <- pca_data_train[, 1:11]
pca_data_val <- pca_data_val[, 1:11]


logit_pca = glm(Closed_Account ~ .,
                data = data.frame(pca_data_train,
                                  Closed_Account = train_data$Closed_Account),
                family = "binomial")
```

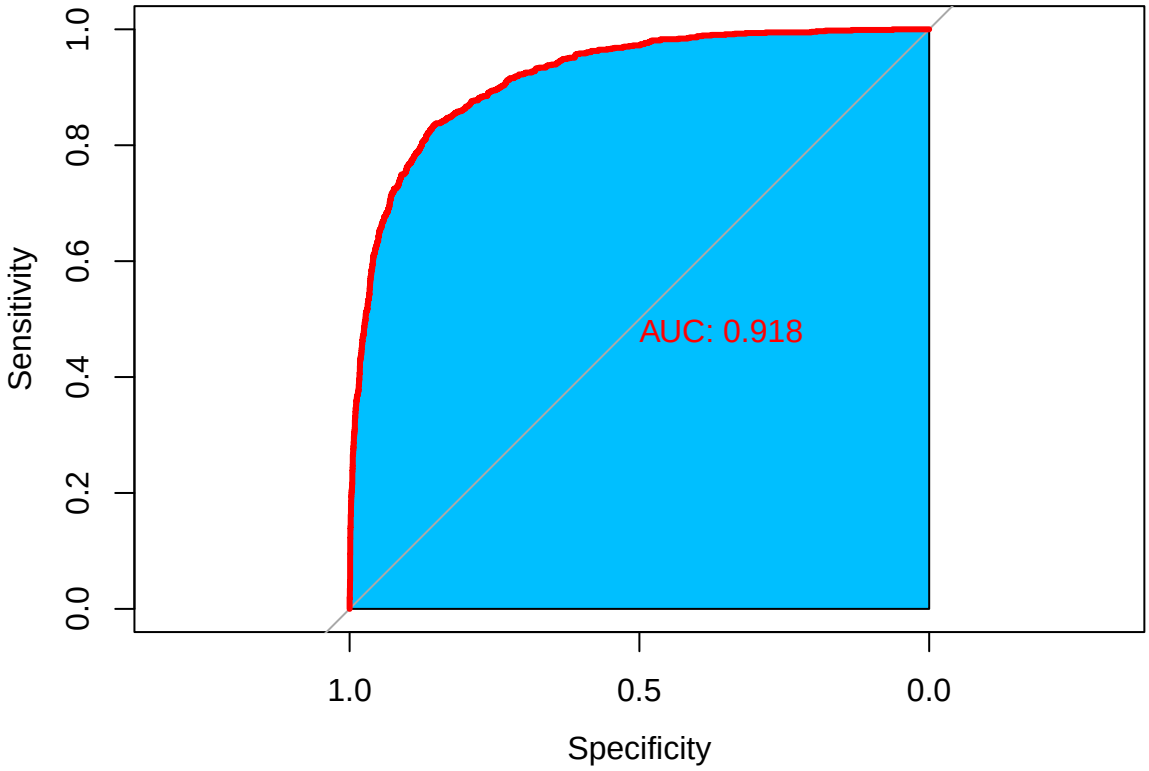
Best model evaluation

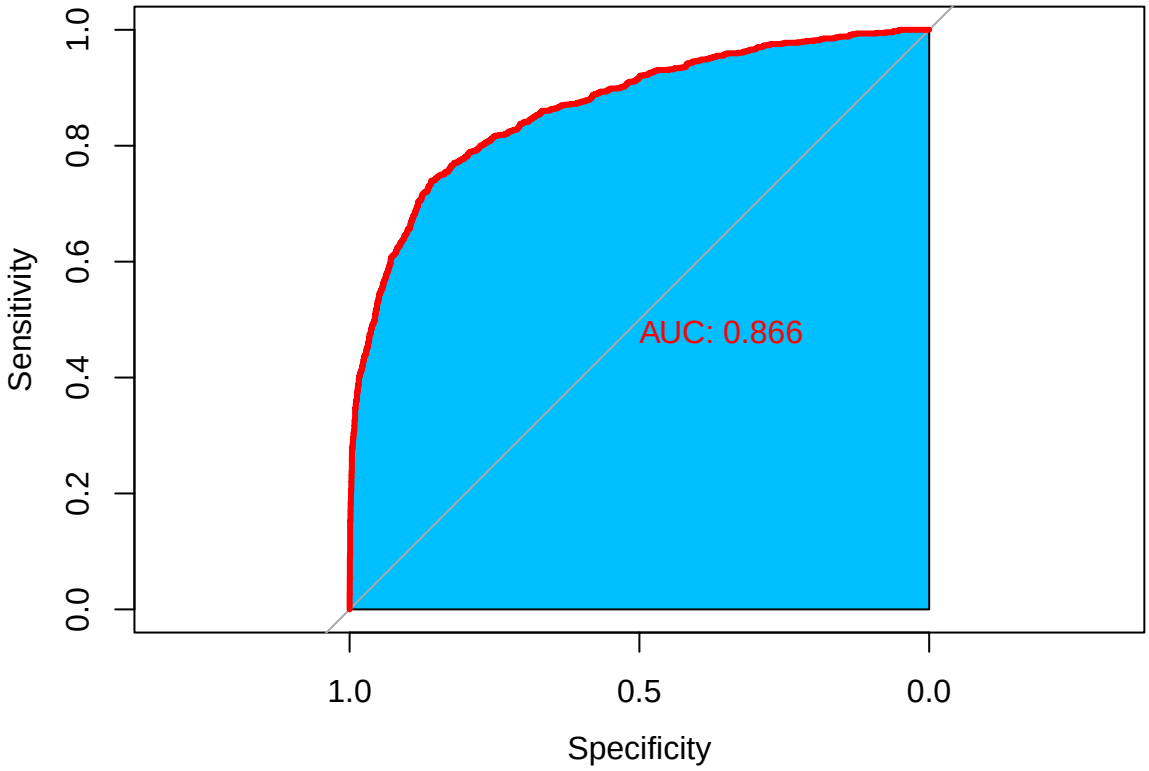
Once we have selected the best model with AIC, BIC and PCA, we are now able to test them on the validation set in order to understand which of them performs in the best way with respect to the AUC metric.

Firstly, we need the ROC curves of the AIC, BIC models we developed and analyzed previously. Since the models obtained with variable selection based on AIC and BIC excluded the same variables, we used only the first one of both.

AUC on the training set

Firstly, we need the ROC curves of the AIC, BIC models we developed and analyzed previously. Since the models obtained with variable selection based on AIC and BIC excluded the same variables, we used only the first one of both. We then proceeded in the same way to analyze the performance





On the train_set, the two models have the same ROC curve with the same AUC=0.918. Even if this is a good result, the models need to be tested on a dataset on which they have not been trained on (validation set).

The PCA model, on the other hand, has AUC=0.866, yielding a result that is not as optimal as the one obtained with the process of variable selection.

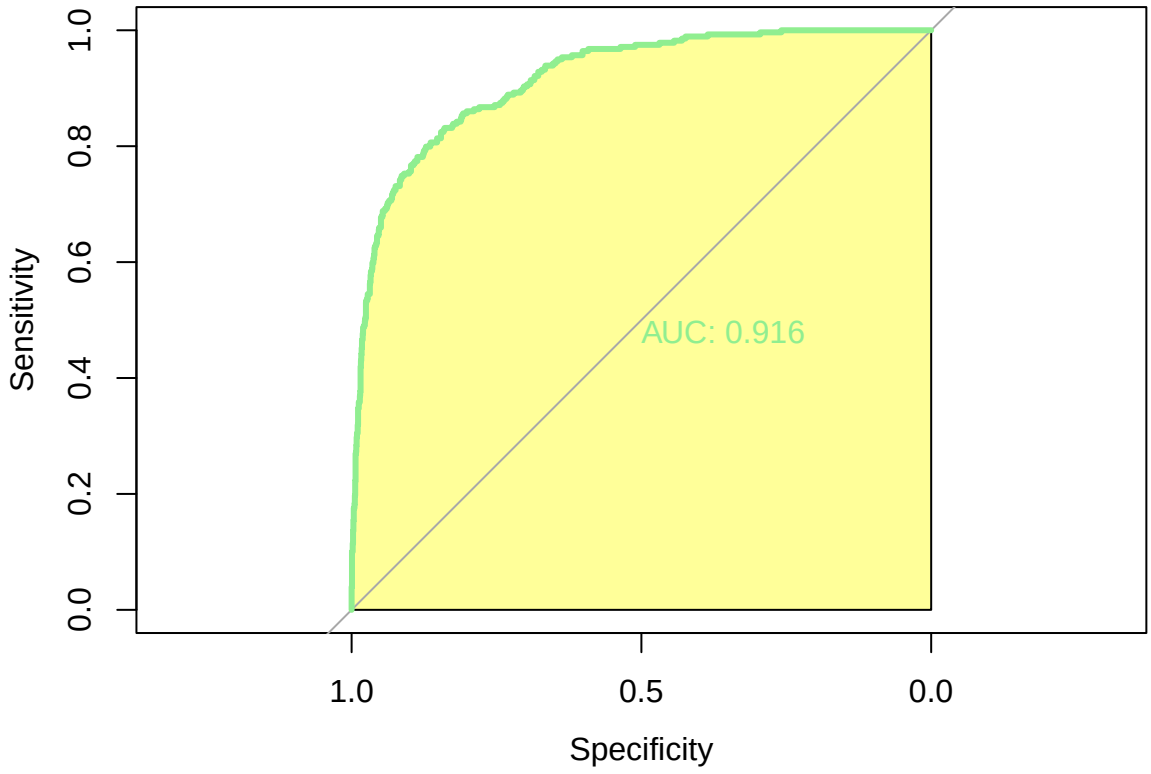
Models testing on the validation set

The following step is to compute the AUC metric on the validation set.

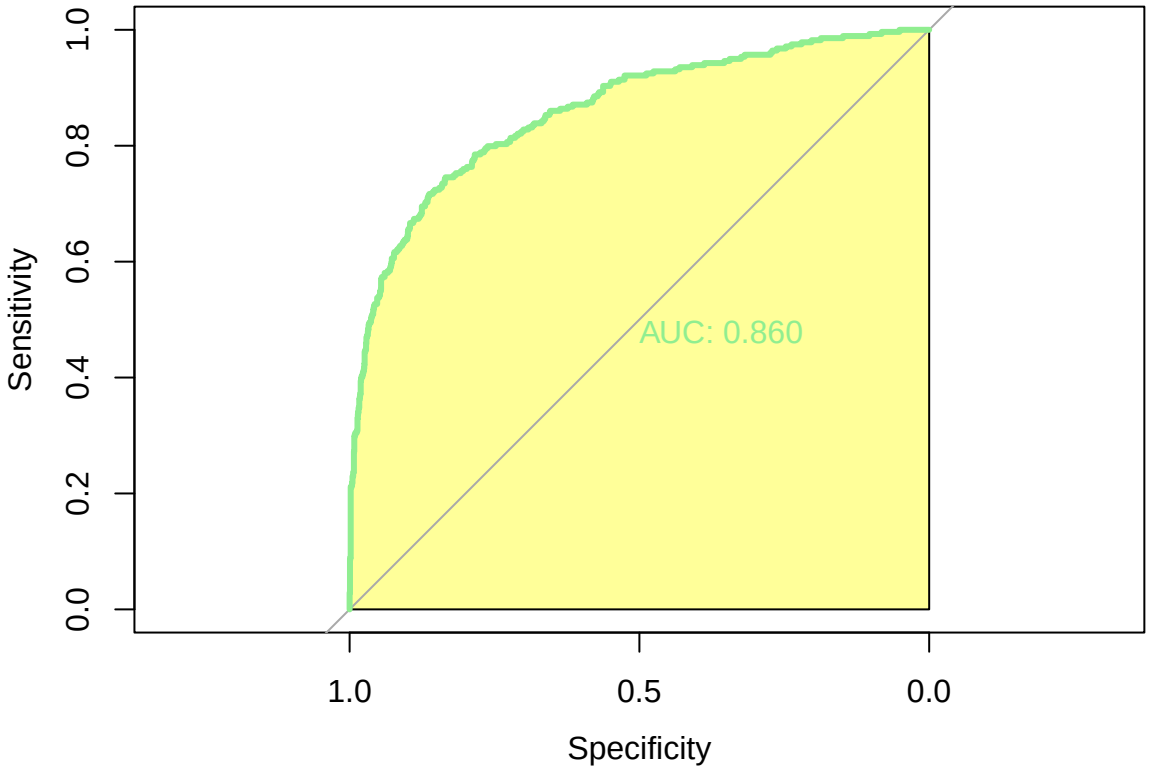
```
## Validation set ##

#threshold
tt=0.5

# Predictions for the observations in the validation set
prob_out_aic1 <- predict(logit_fit_aic1,
                         newdata = val_data[,-1],
                         type = "response")

pred_out_aic1 <- as.factor(ifelse(prob_out_aic1 > tt, "yes", "no"))


prob_out_bic1 <- predict(logit_fit_bic1,
                         newdata = val_data[,-1],
                         type = "response")
```

```r
pred_out_bic1 <- as.factor(ifelse(prob_out_bic1 > tt, "yes", "no"))


pca_data_val_df <- data.frame(pca_data_val) # A dataframe is needed as parameter
prob_out_pca <- predict(logit_pca,
                        newdata = pca_data_val_df,
                        type = "response")


pred_out_pca <- as.factor(ifelse(prob_out_pca > tt, "yes", "no"))
```

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

## Setting direction: controls < cases

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

## Setting direction: controls < cases
```




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



The model shows a slightly reduced performance on the validation compared with the test set, which is a common occurrence in predictive modeling. However, this minor difference is not significant and points out model's strong generalization capabilities. Such generalization is crucial since it means the model doesn't just memorize the training data, but it rather learns the underlying patterns, enabling it to maintain accuracy when applied to new datasets. A significant performance drop on the validation set would signal overfitting, while the close performance metrics between the two sets in this case suggest that the model has achieved a desirable balance, neither overfitting nor underfitting. It is then capable of providing with consistent predictions on both familiar and unfamiliar data.

The same line of reasoning also works for the PCA model. The performance is still smaller than the models obtained trough variable selection. Our best model is hence the one found with aic.

```
best_model = logit_fit_aic1
```

Test Dataset predictions

Import Dataset

```
TestData = read.csv2("./Dataset/bank_accounts_test.csv",
                     header = T,
                     sep = ",",
                     colClasses = "character")
```

Preprocessing on Test Dataset

We need to remove the Unknown values in the same way we have done for the training set.

```
#Encode NaN values
#Rename missing values with NA notation
TestData[TestData == 'Unknown'] <- NA

for (var in categorical_variables) {
  if (any(is.na(TestData[[var]]))) {
    TestData[[var]][is.na(TestData[[var]])] <- getSimpleMode(TestData[[var]])
  }
}

##substitute the type of the variables from "ch" to "num" and "factor"
for (var in numerical_variables) {
  TestData[[var]] = as.numeric(TestData[[var]])
}

for (var in categorical_variables) {
  TestData[[var]] = as.factor(TestData[[var]])
}
```

Predictions for the observations in the Test set

We then compute the probabilities for our best model and we write them into the csv file in order to store them.

```
predicted_prob <- predict(best_model,
                          newdata = TestData,
                          type = "response")

head(predicted_prob)
```

```
##          1          2          3          4          5          6
## 0.01365572 0.09135783 0.03797448 0.87118137 0.06411663 0.17092844
```

```
##Write the .csv file that contains the predicted probabilities.
write.csv(predicted_prob, "my_prob.csv", row.names = F)
```

6 TRESHOLD CHOICE

First of all, a set of thresholds will be used to perform a greedy search for the optimal value. Then, the function calculate_metrics is set up to apply each threshold to the model's predicted probabilities. For each threshold, it calculates the predicted binary outcomes (1 for event occurrence, 0 for non-occurrence) and constructs a confusion matrix to tabulate the true positives, true negatives, false positives, and false negatives. The function performance metrics computes in addittion key performance metrics such as accuracy, precision, recall (sensitivity), and F1 score to evaluate the model's performance at each threshold.

```r
# Thresholds choices
thresholds <- c(0.5, 0.2, 0.8, 0.28)

# Function to calculate metrics and financial outcome
calculate_metrics <- function(tt) {
  prob_out_best <- predict(logit_fit_bic1, newdata = val_data, type = "response")
  pred_out_best_binary <- ifelse(prob_out_best > tt, 1, 0)
  conf_matrix <- table(Predicted = pred_out_best_binary,
                       Actual = val_data$Closed_Account)

# Extracting elements from the confusion matrix
TP <- conf_matrix["1", "1"]
TN <- conf_matrix["0", "0"]
FP <- conf_matrix["1", "0"]
FN <- conf_matrix["0", "1"]

# Calculating financial outcome
gain_TP <- TP * 50
gain_TN <- TN * 20
loss_FP <- FP * (-20)
loss_FN <- FN * (-50)
tot_outcome <- gain_TP + gain_TN + loss_FP + loss_FN

# Calculating metrics
accuracy <- (TP + TN) / sum(conf_matrix)
precision <- TP / (TP + FP)
recall <- TP / (TP + FN)
F1_score <- 2 * (precision * recall) / (precision + recall)

cat("Threshold: ", tt, "\n",
    "Financial Gain: ", tot_outcome, "\n",
    "Accuracy: ", accuracy, "\n",
    "Precision: ", precision, "\n",
    "Recall (Sensitivity): ", recall, "\n",
    "F1 Score: ", F1_score, "\n\n")
}
```

We can then shows the result and comment them:

```r
# Calculate metrics for each threshold
for (tt in thresholds) {
  calculate_metrics(tt)
}
```

```
## Threshold:  0.5
##  Financial Gain:  32130
##  Accuracy:  0.9089474
##  Precision:  0.7548077
##  Recall (Sensitivity):  0.562724
##  F1 Score:  0.6447639
##
## Threshold:  0.2
##  Financial Gain:  32370
```

```
##  Accuracy:  0.8631579
##  Precision:  0.522673
##  Recall (Sensitivity):  0.7849462
##  F1 Score:  0.6275072
##
## Threshold:  0.8
##  Financial Gain:  25030
##  Accuracy:  0.8842105
##  Precision:  0.8641975
##  Recall (Sensitivity):  0.2508961
##  F1 Score:  0.3888889
##
## Threshold:  0.28
##  Financial Gain:  33510
##  Accuracy:  0.89
##  Precision:  0.6035503
##  Recall (Sensitivity):  0.7311828
##  F1 Score:  0.6612642
```

In our case, we focus on sensitivity because it's critical for the bank to capture as many true positives (actual account closures) as possible. High sensitivity ensures the bank can proactively retain these customers, avoiding a potential loss for each customer who leaves. Threshold of 0.5 yields decent accuracy but falls short in maximizing financial gain due to moderate sensitivity. Threshold of 0.2 boosts sensitivity, catching more customers who might leave, but also increases false positives, where offers are made to customers who would stay regardless. Threshold of 0.8 sharply decreases sensitivity, reducing potential gains significantly. Threshold of 0.28 strikes the best balance, leading to the highest financial gain by achieving a good balance between sensitivity and precision and is hence the optimal solution.