FUNDAMENTALS OF DATA SCIENCE

HOMEWORK - 5

Mrunali Vikas Patil

Load necessary libraries

```
library(readr)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

library(ggplot2)
library(caret)

## Loading required package: lattice

library(cluster)
library(pROC)

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

##
## Attaching package: 'pROC'

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

library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa

library(ROCR)
```

A) DATA GATHERING AND INTEGRATION

```
data = read.csv("Churn_Modelling.csv", header = T)

dim(data)

## [1] 10000    14

head(data)

##   RowNumber CustomerId  Surname CreditScore Geography Gender Age
Tenure
## 1         1   15634602 Hargrave         619    France Female  42
2
## 2         2   15647311     Hill         608     Spain Female  41
1
## 3         3   15619304     Onio         502    France Female  42
8
## 4         4   15701354     Boni         699    France Female  39
1
## 5         5   15737888 Mitchell         850     Spain Female  43
2
## 6         6   15574012      Chu         645     Spain   Male  44
8
##     Balance NumOfProducts HasCrCard IsActiveMember EstimatedSalary
Exited
## 1      0.00             1         1              1       101348.88
1
## 2  83807.86             1         0              1       112542.58
0
## 3 159660.80             3         1              0       113931.57
1
## 4      0.00             2         0              0        93826.63
0
## 5 125510.82             1         1              1        79084.10
0
## 6 113755.78             2         1              0       149756.71
1
```

```r
str(data)
```

```
## 'data.frame':    10000 obs. of  14 variables:
##  $ RowNumber      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ CustomerId     : int  15634602 15647311 15619304 15701354
15737888 15574012 15592531 15656148 15792365 15592389 ...
##  $ Surname        : chr  "Hargrave" "Hill" "Onio" "Boni" ...
##  $ CreditScore    : int  619 608 502 699 850 645 822 376 501
684 ...
##  $ Geography      : chr  "France" "Spain" "France" "France" ...
##  $ Gender         : chr  "Female" "Female" "Female" "Female" ...
##  $ Age            : int  42 41 42 39 43 44 50 29 44 27 ...
##  $ Tenure         : int  2 1 8 1 2 8 7 4 4 2 ...
##  $ Balance        : num  0 83808 159661 0 125511 ...
##  $ NumOfProducts  : int  1 1 3 2 1 2 2 4 2 1 ...
##  $ HasCrCard      : int  1 0 1 0 1 1 1 1 0 1 ...
##  $ IsActiveMember : int  1 1 0 0 1 0 1 0 1 1 ...
##  $ EstimatedSalary: num  101349 112543 113932 93827 79084 ...
##  $ Exited         : int  1 0 1 0 0 1 0 1 0 0 ...
```

```r
summary(data)
```

```
##    RowNumber       CustomerId         Surname           CreditScore
##  Min.   :    1   Min.   :15565701   Length:10000       Min.
:350.0
##  1st Qu.: 2501   1st Qu.:15628528   Class :character   1st
Qu.:584.0
##  Median : 5000   Median :15690738   Mode  :character   Median
:652.0
##  Mean   : 5000   Mean   :15690941                      Mean
:650.5
##  3rd Qu.: 7500   3rd Qu.:15753234                      3rd
Qu.:718.0
##  Max.   :10000   Max.   :15815690                      Max.
:850.0
##   Geography            Gender               Age            Tenure
##  Length:10000       Length:10000       Min.   :18.00   Min.   :
0.000
##  Class :character   Class :character   1st Qu.:32.00   1st Qu.:
3.000
##  Mode  :character   Mode  :character   Median :37.00   Median :
5.000
```

```
##                                          Mean   :38.92   Mean   :
5.013
##                                          3rd Qu.:44.00   3rd Qu.:
7.000
##                                          Max.   :92.00   Max.
:10.000
##     Balance       NumOfProducts    HasCrCard      IsActiveMember
##  Min.   :     0   Min.   :1.00   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:     0   1st Qu.:1.00   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 97199   Median :1.00   Median :1.0000   Median :1.0000
##  Mean   : 76486   Mean   :1.53   Mean   :0.7055   Mean   :0.5151
##  3rd Qu.:127644   3rd Qu.:2.00   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :250898   Max.   :4.00   Max.   :1.0000   Max.   :1.0000
##  EstimatedSalary        Exited
##  Min.   :    11.58   Min.   :0.0000
##  1st Qu.: 51002.11   1st Qu.:0.0000
##  Median :100193.91   Median :0.0000
##  Mean   :100090.24   Mean   :0.2037
##  3rd Qu.:149388.25   3rd Qu.:0.0000
##  Max.   :199992.48   Max.   :1.0000
```

B)DATA CLEANING AND PREPROCESSING

Checking if data has unique value columns

```
unique_counts <- sapply(data, function(x) length(unique(x)))
```

Display the count of unique values for each column

```
print(unique_counts)
```

```
##       RowNumber      CustomerId         Surname     CreditScore
Geography
##           10000           10000            2932             460
3
##          Gender             Age          Tenure         Balance
NumOfProducts
##               2              70              11            6382
4
##       HasCrCard  IsActiveMember EstimatedSalary          Exited
##               2               2            9999               2
```

Removing Unique value columns

```
data <- data %>% select(-RowNumber, -CustomerId, -Surname)
head(data)

##   CreditScore Geography Gender Age Tenure   Balance NumOfProducts
HasCrCard
## 1         619    France Female  42      2      0.00             1
1
## 2         608     Spain Female  41      1  83807.86             1
0
## 3         502    France Female  42      8 159660.80             3
1
## 4         699    France Female  39      1      0.00             2
0
## 5         850     Spain Female  43      2 125510.82             1
1
## 6         645     Spain   Male  44      8 113755.78             2
1
##   IsActiveMember EstimatedSalary Exited
## 1              1       101348.88      1
## 2              1       112542.58      0
## 3              0       113931.57      1
## 4              0        93826.63      0
## 5              1        79084.10      0
## 6              0       149756.71      1
```

Checking the NA values

```
na_check <- sapply(data, function(x) any(is.na(x)))
print(na_check)

##     CreditScore       Geography          Gender             Age
Tenure
##           FALSE           FALSE           FALSE           FALSE
FALSE
##         Balance   NumOfProducts       HasCrCard  IsActiveMember
EstimatedSalary
##           FALSE           FALSE           FALSE           FALSE
FALSE
##          Exited
##           FALSE

colSums(is.na(data))
```

```
##     CreditScore       Geography          Gender              Age
Tenure
##               0               0               0                0
0
##         Balance   NumOfProducts       HasCrCard   IsActiveMember
EstimatedSalary
##               0               0               0                0
0
##          Exited
##               0
```

```
colMeans(is.na(data)) * 100
```

```
##     CreditScore       Geography          Gender              Age
Tenure
##               0               0               0                0
0
##         Balance   NumOfProducts       HasCrCard   IsActiveMember
EstimatedSalary
##               0               0               0                0
0
##          Exited
##               0
```

No missing(NA) values in the data

```
data$Exited <- as.factor(data$Exited)
```

Function to detect outliers based on IQR

```
detect_outliers <- function(x) {
  Q1 <- quantile(x, 0.25)
  Q3 <- quantile(x, 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  return(x < lower_bound | x > upper_bound)
}
```

Apply the function to each numerical column

```
outliers <- sapply(data, function(x) if(is.numeric(x))
sum(detect_outliers(x)) else NA)
```

Print the number of outliers in each numerical column

```
print(outliers)

##     CreditScore       Geography          Gender             Age
Tenure
##              15              NA              NA             359
0
##         Balance   NumOfProducts       HasCrCard  IsActiveMember
EstimatedSalary
##               0              60               0               0
0
##          Exited
##              NA
```

Function to remove outliers

```
remove_outliers <- function(df, col) {
  Q1 <- quantile(df[[col]], 0.25)
  Q3 <- quantile(df[[col]], 0.75)
  IQR <- Q3 - Q1
  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR
  df <- df[df[[col]] >= lower_bound & df[[col]] <= upper_bound, ]
  return(df)
}
```

Remove outliers from ‘CreditScore’, ‘Age’, and ‘NumOfProducts’

```
data <- remove_outliers(data, "CreditScore")
data <- remove_outliers(data, "Age")
data <- remove_outliers(data, "NumOfProducts")
```

Apply the function to each numerical column

```
outliers_1 <- sapply(data, function(x) if(is.numeric(x))
sum(detect_outliers(x)) else NA)
```

Print the number of outliers in each numerical column

```
print(outliers_1)
```

```
##     CreditScore       Geography          Gender             Age
Tenure
##               1              NA              NA             163
0
##         Balance   NumOfProducts       HasCrCard  IsActiveMember
EstimatedSalary
##               0               0               0               0
0
##          Exited
##              NA
```

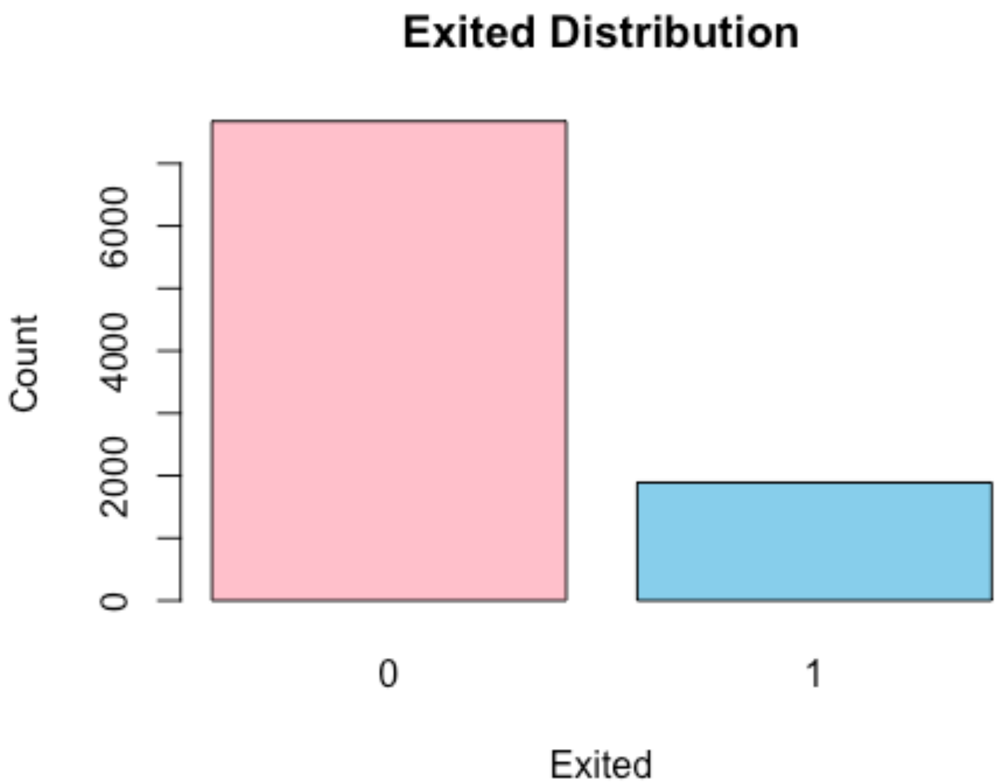
Now we have less number of outliers

Convert ‘Geography’ and ‘Gender’ to factor variables

```
data$Geography <- as.factor(data$Geography)
data$Gender <- as.factor(data$Gender)
```

C) TARGET VARIABLE ANALYSIS

```
Exited <- table(data$Exited)

barplot(Exited, main="Exited Distribution", xlab="Exited",
ylab="Count", col=c("pink", "skyblue"))
```



The bar chart represents the distribution of a binary 'Exited' variable, which likely indicates whether customers have left or stayed with a service.

The number '0' (pink bar) represents customers who have not exited, which is significantly higher than the number '1' (blue bar), representing customers who have exited.

This shows that within this dataset, a larger number of customers have stayed rather than exited.

Calculate the percentage of 'Yes' and 'No' values

```
Exited_percentage <- prop.table(table(data$Exited)) * 100

cat("Percentage of 'Yes' in Churn:", Exited_percentage["1"], "%\n")

## Percentage of 'Yes' in Churn: 19.7638 %
```

```
cat("Percentage of 'No' in Churn:", Exited_percentage["0"], "%\n")

## Percentage of 'No' in Churn: 80.2362 %
```

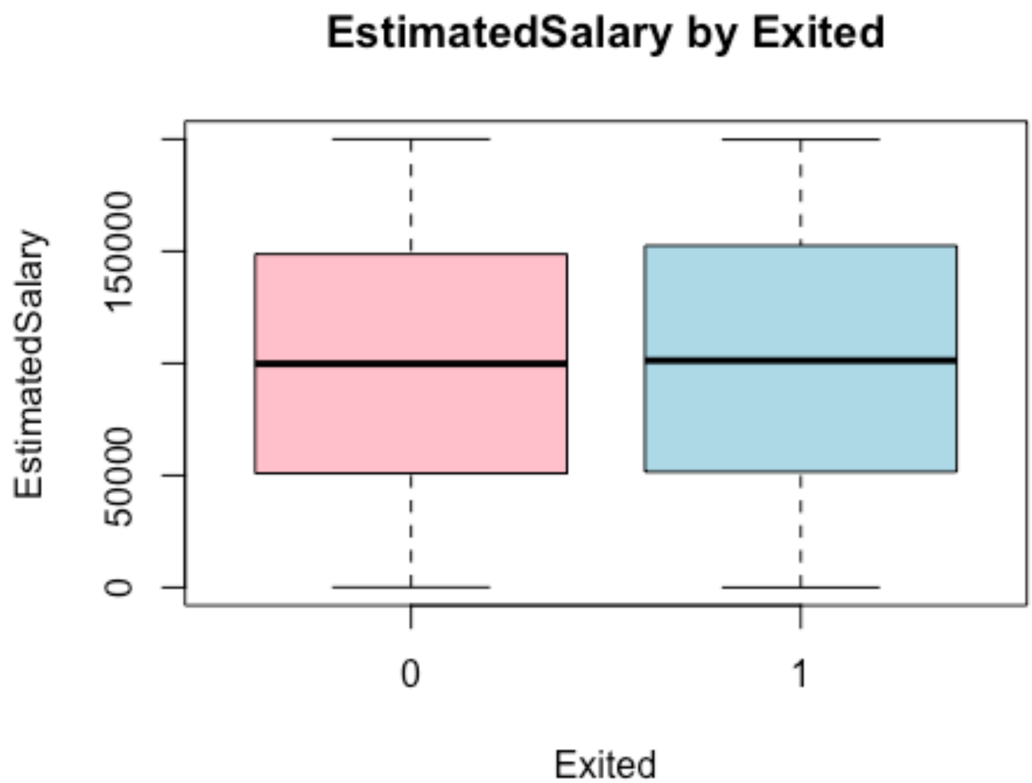
D) DATA EXPLORATION - EXPLORATORY DATA ANALYSIS

Separate categorical and numerical data

```
data_categorical <- data %>% select_if(is.factor)
data_numerical <- data %>% select_if(is.numeric)

sapply(data, class)

##     CreditScore       Geography          Gender             Age
Tenure
##       "integer"        "factor"        "factor"       "integer"
"integer"
##         Balance   NumOfProducts       HasCrCard  IsActiveMember
EstimatedSalary
##       "numeric"       "integer"       "integer"       "integer"
"numeric"
##          Exited
##        "factor"
```

Boxplot of EstimatedSalary by Exited

```
par(mfrow=c(1,1))
boxplot(EstimatedSalary ~ Exited, data = data, main = "EstimatedSalary
by Exited",
        xlab = "Exited", ylab = "EstimatedSalary", col = c("pink",
"lightblue"))
```



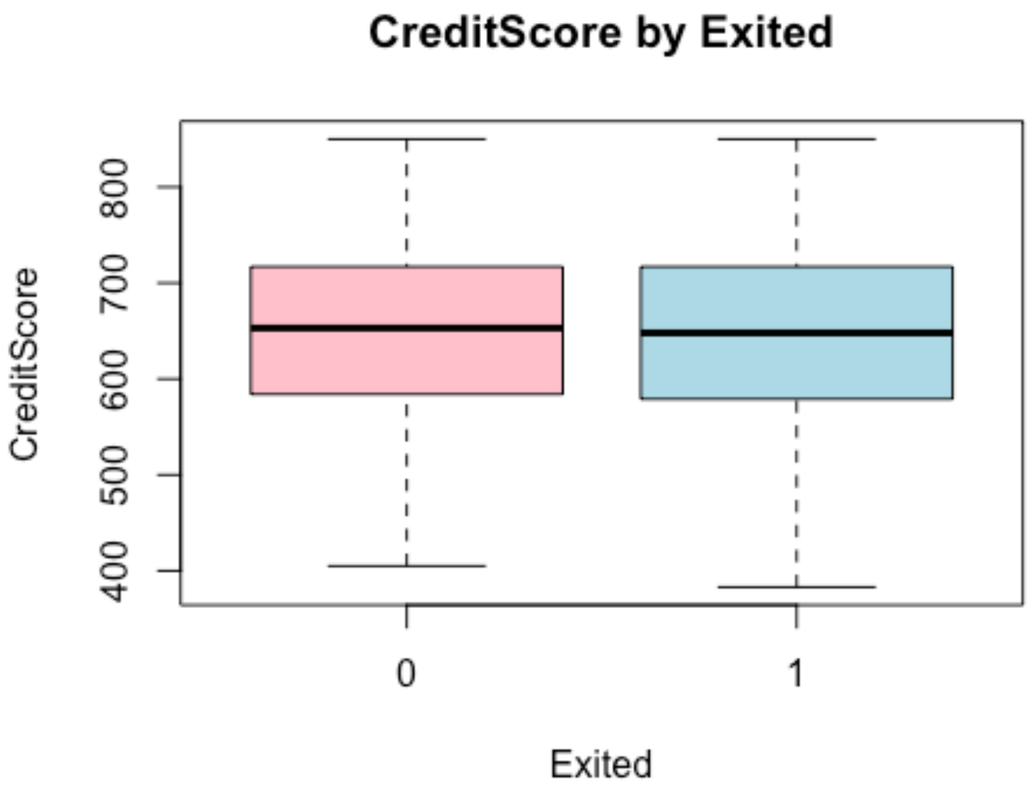
The box plot compares the distribution of estimated salaries between customers who have not exited (0) and those who have exited (1).

Both categories display a similar median salary around 100,000, with the interquartile range (IQR) indicating the middle 50% of salaries is similar for both groups.

The similar spread and central tendency suggest that estimated salary may not be a distinguishing factor between customers who stay and those who leave.

Boxplot of CreditScore by Exited

```
boxplot(CreditScore ~ Exited, data = data, main = "CreditScore by
Exited",
        xlab = "Exited", ylab = "CreditScore", col = c("pink",
"lightblue"))
```



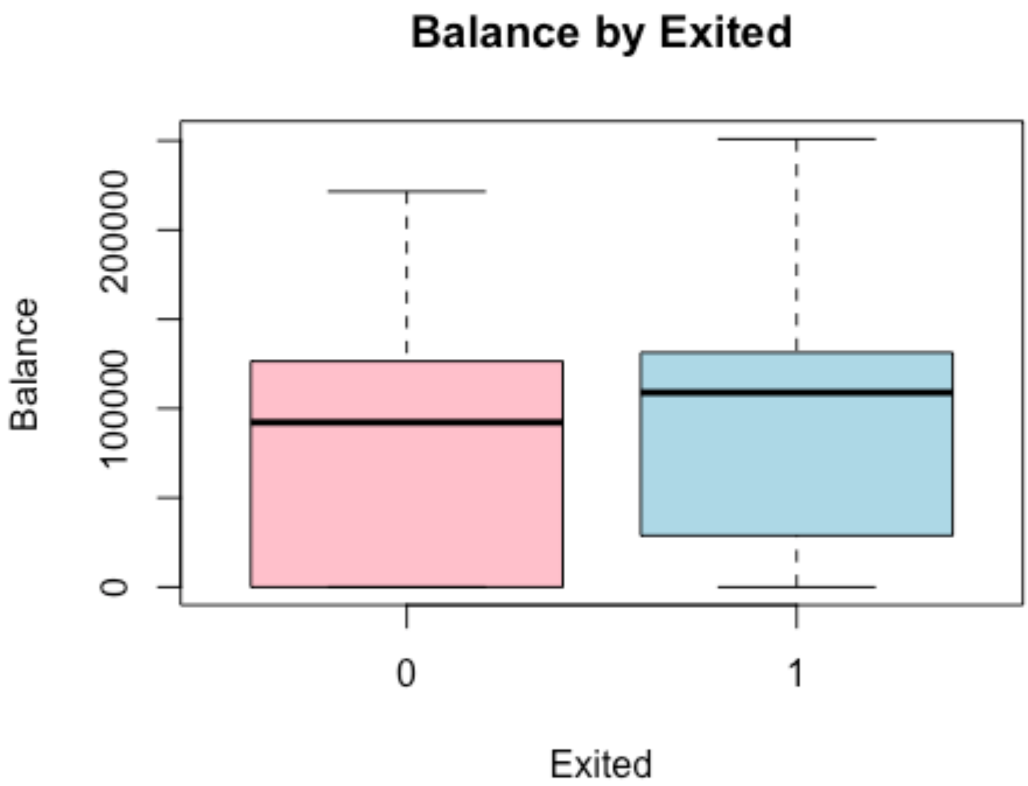
The box plot illustrates the distribution of credit scores among customers who have not exited (0) and those who have exited (1).

Both groups show a similar range of credit scores, with the median credit score for both non-exited and exited customers around the mid-600s.

There are no significant differences between the credit score distributions of the two groups, suggesting that credit score alone may not be a strong predictor of customer exit.

Boxplot of Balance by Exited

```
boxplot(Balance ~ Exited, data = data, main = "Balance by Exited",
        xlab = "Exited", ylab = "Balance", col = c("pink",
"lightblue"))
```



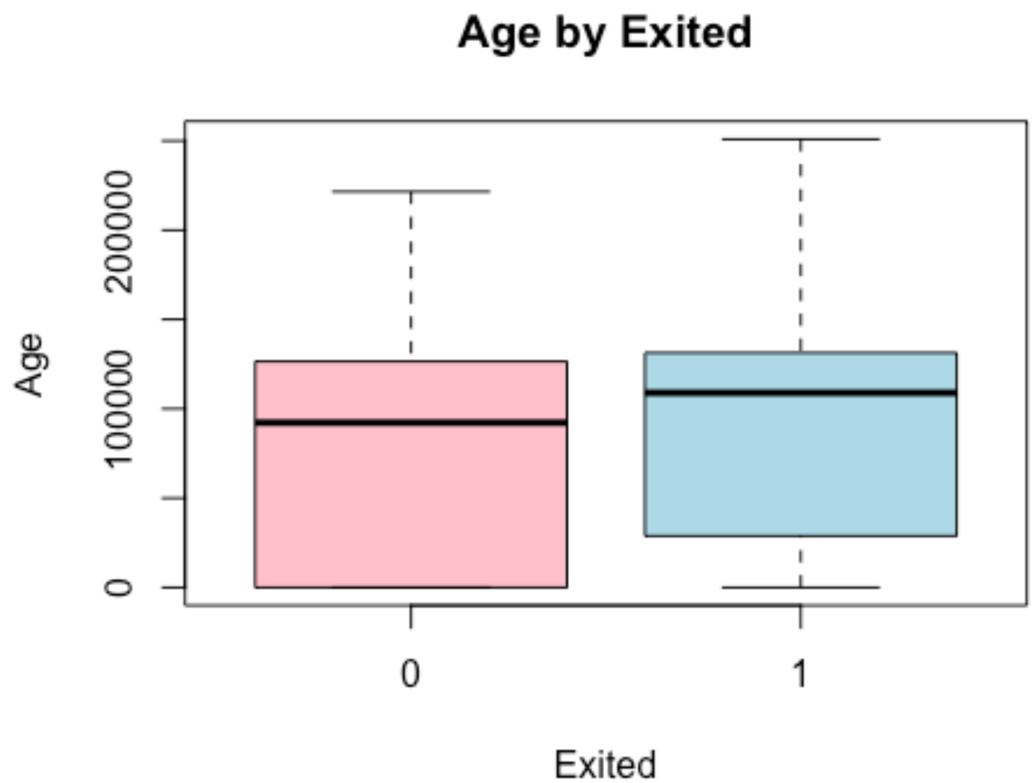
The box plot shows the distribution of account balances for customers who have stayed (0) and those who have exited (1).

Customers who exited tend to have higher median balances compared to those who stayed.

The interquartile range is similar for both, but the median is notably higher for the exited group, suggesting a possible correlation between higher balances and customer churn.

Boxplot of Age by Exited

```
boxplot(Balance ~ Exited, data = data, main = "Age by Exited",
        xlab = "Exited", ylab = "Age", col = c("pink", "lightblue"))
```



The box plot likely contains an error. It is labeled "Age by Exited," but the y-axis values are too high to represent ages and are more in line with a financial metric such as balance or salary.

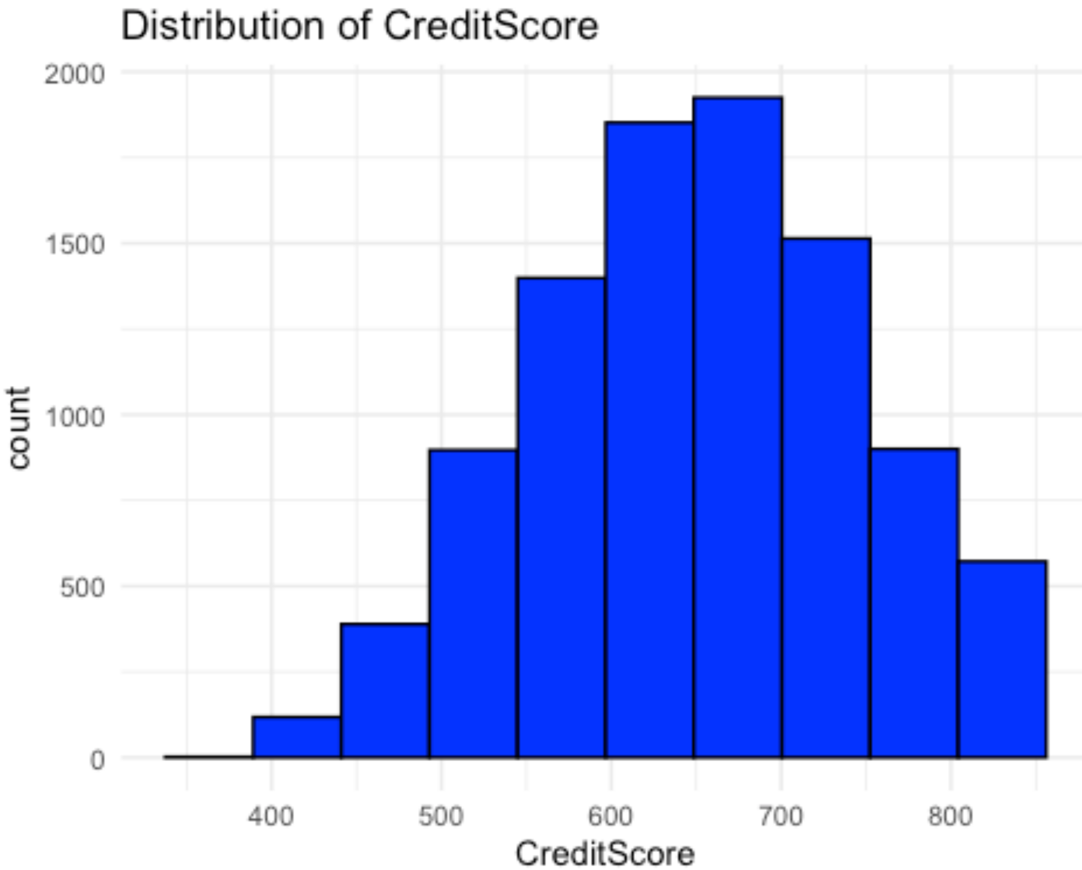
The correct label for the y-axis should be verified, as the data seems to represent a different variable, not age.

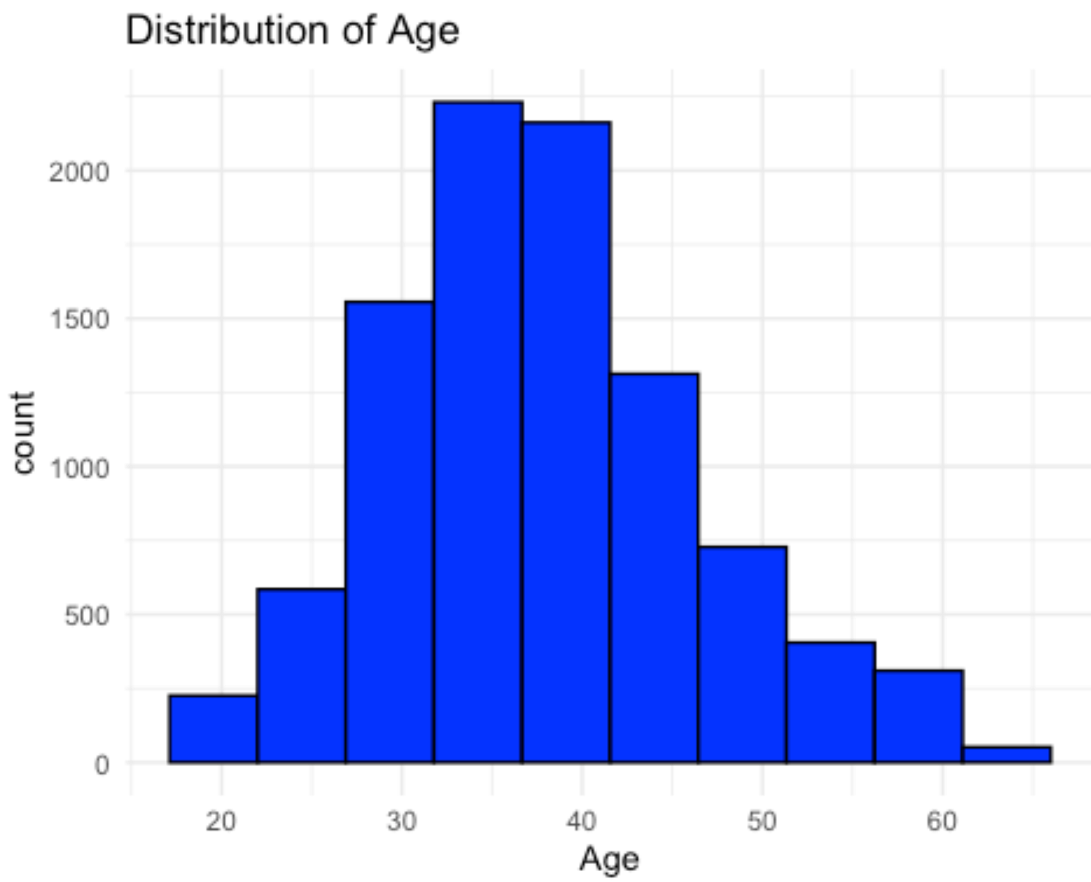
```
library(readr)
library(dplyr)
library(ggplot2)

numerical_features <- c("CreditScore",
"Age","Tenure","Balance","NumOfProducts","EstimatedSalary")
```

Histogram of Estimated Salary

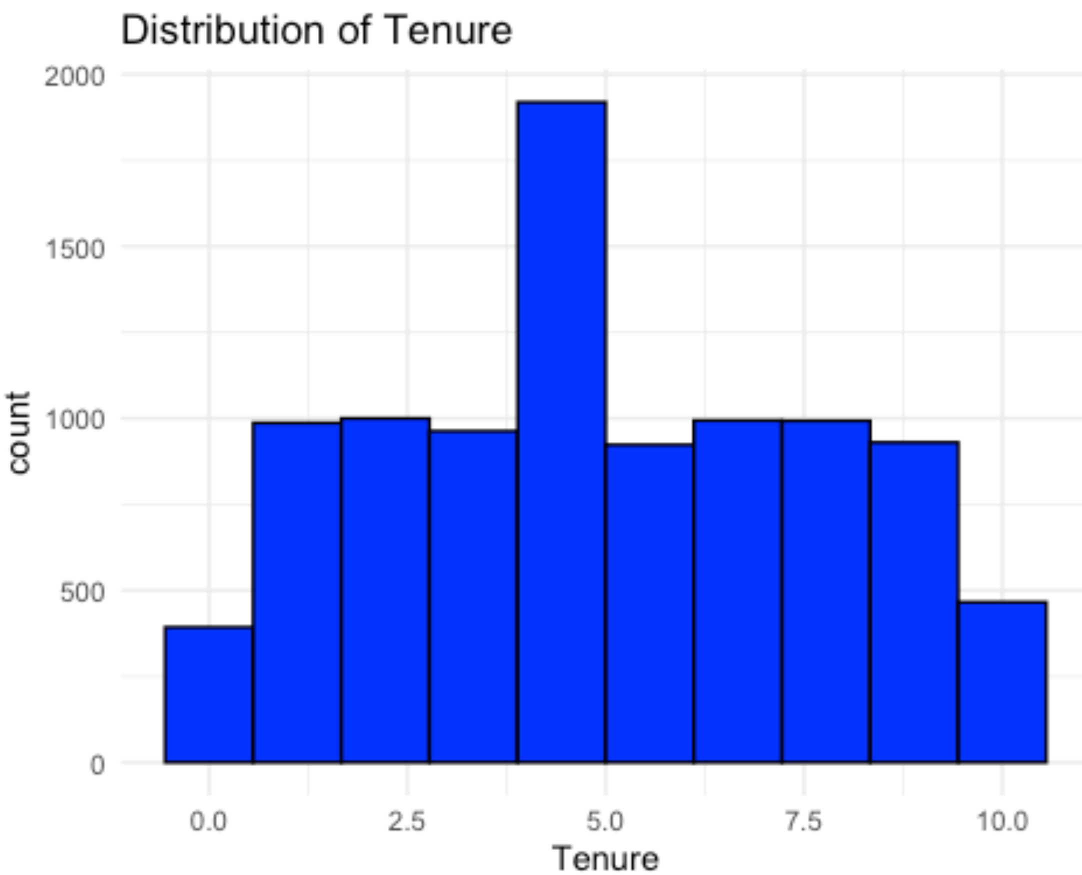
```
for (feat in numerical_features) {
  p <- ggplot(data, aes_string(x = feat)) +
    geom_histogram(bins = 10, fill = "blue", color = "black") +
    theme_minimal() +
    ggtitle(paste("Distribution of", feat))
  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this
warning was
## generated.
```

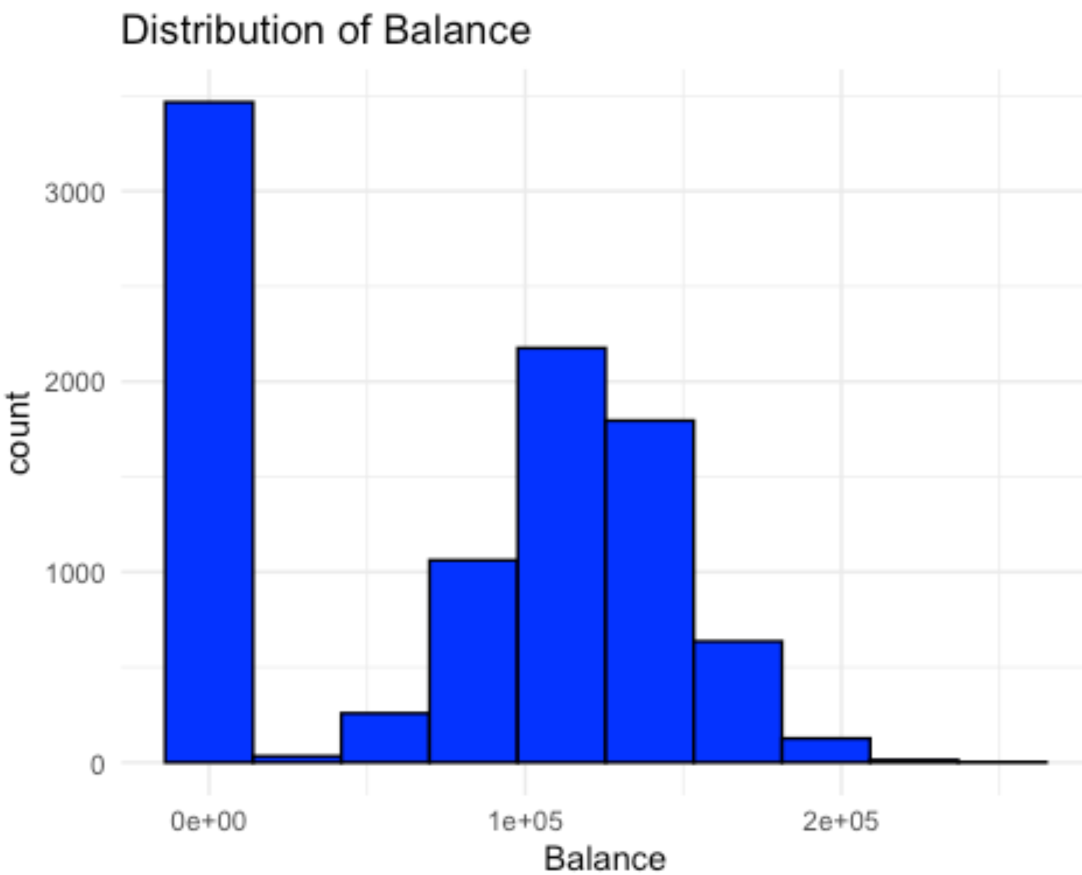

Distribution of CreditScore
2000
1500
1000
500
0
count
400
500
600
700
800
CreditScore

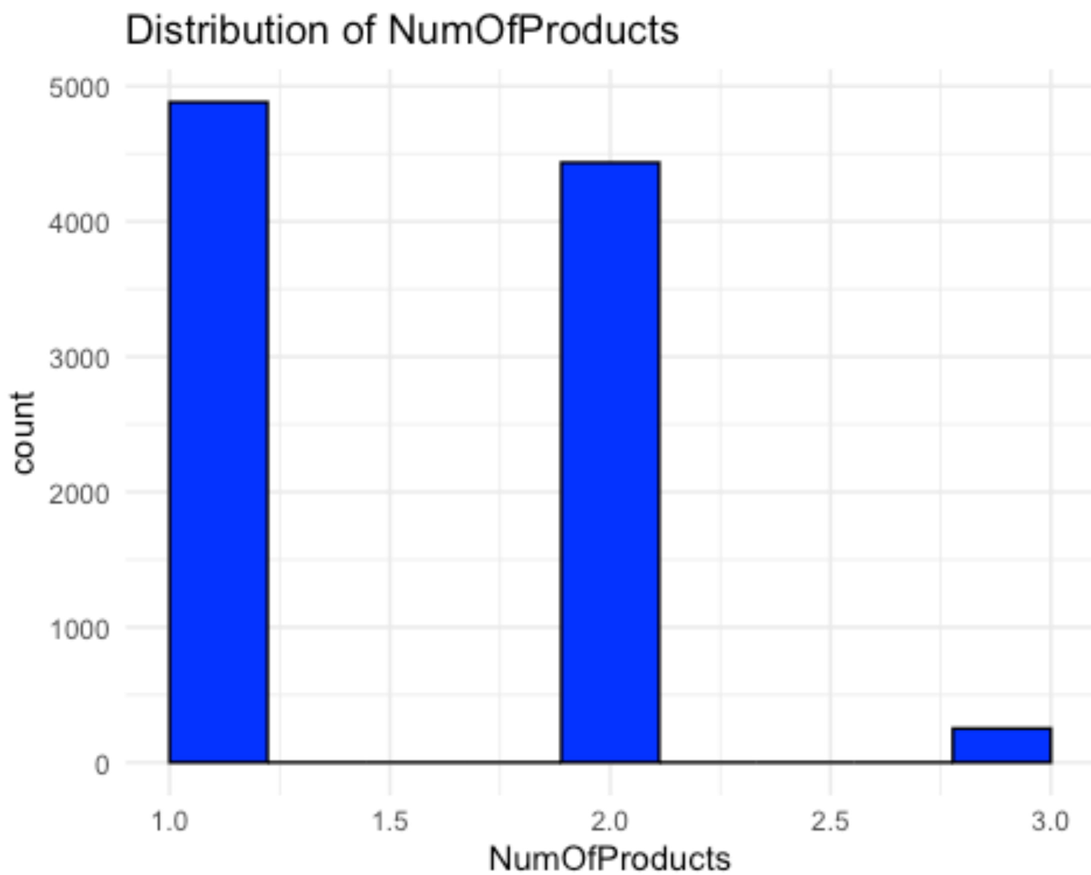


Distribution of Age
count
2000
1500
1000
500
0
20
30
40
50
60
Age

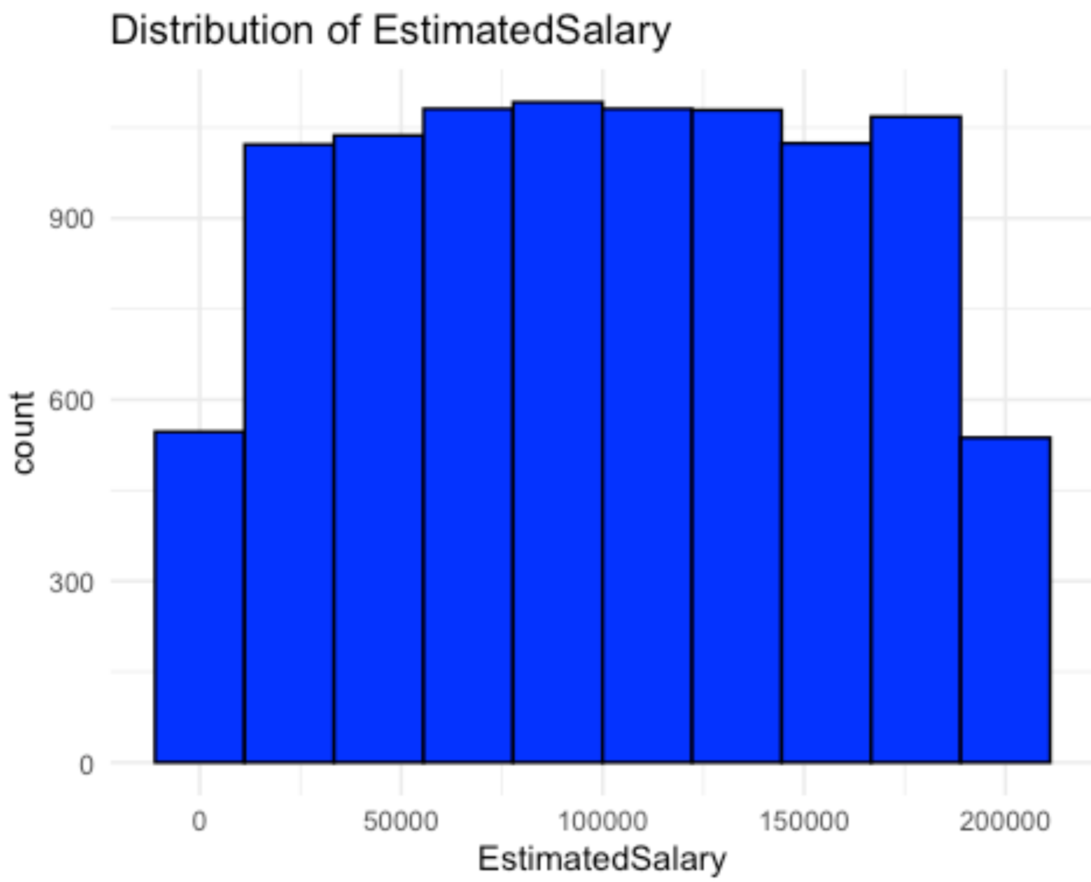


Distribution of Tenure
2000
1500
1000
500
0
count
0.0
2.5
5.0
7.5
10.0
Tenure


Distribution of Balance
count
3000
2000
1000
0
0e+00
1e+05
2e+05
Balance



Distribution of NumOfProducts
5000
4000
3000
2000
1000
0
count
1.0
1.5
2.0
2.5
3.0
NumOfProducts



The histogram shows the distribution of estimated salaries among a group of individuals.

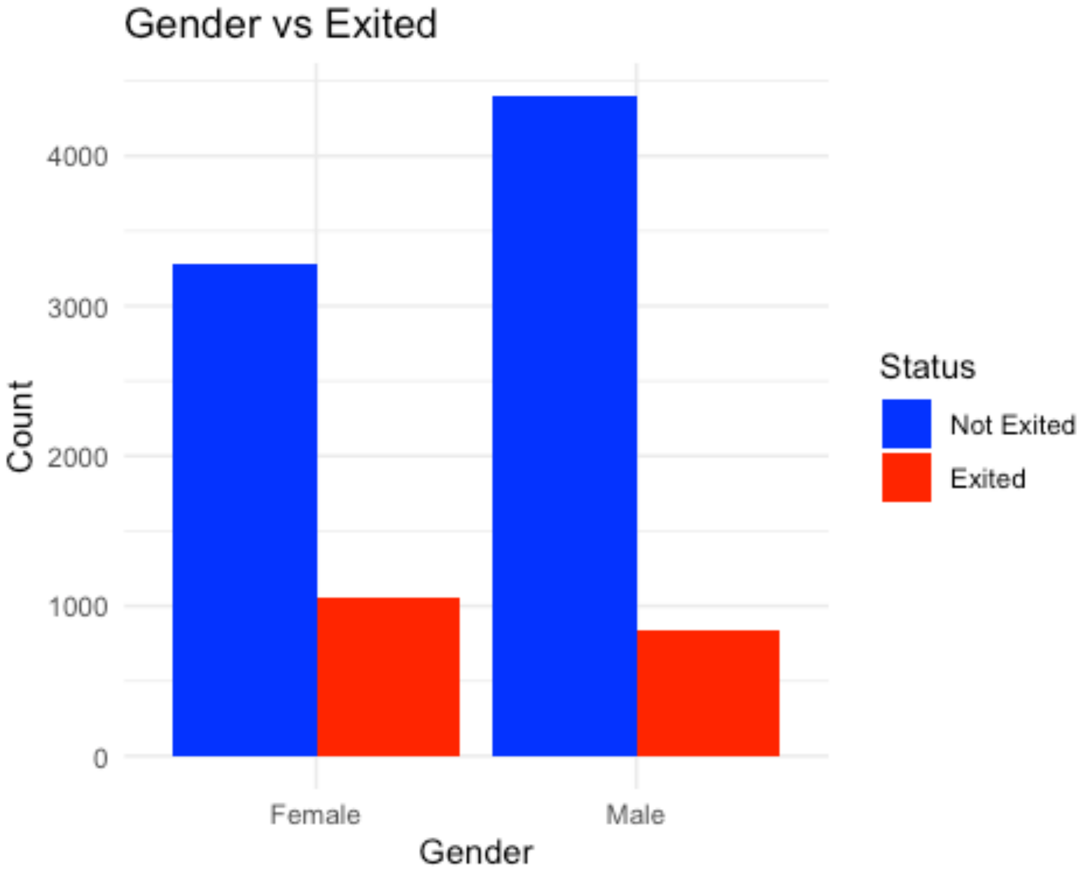
The salaries are spread relatively evenly across different ranges, with the counts of individuals peaking in the central salary ranges.

The distribution appears to be fairly uniform, without a strong skew toward lower or higher salaries, suggesting that people in this group have a wide range of estimated salaries with no single salary range dominating.

Bar plot for Gender vs Exited

```
ggplot(data, aes(x = Gender, fill = factor(Exited))) +
  geom_bar(position = "dodge") +
  scale_fill_manual(values = c("blue", "red"),
                    labels = c("Not Exited", "Exited")) +
  labs(title = "Gender vs Exited", x = "Gender", y = "Count", fill =
```

```
"Status") +
  theme_minimal()
```



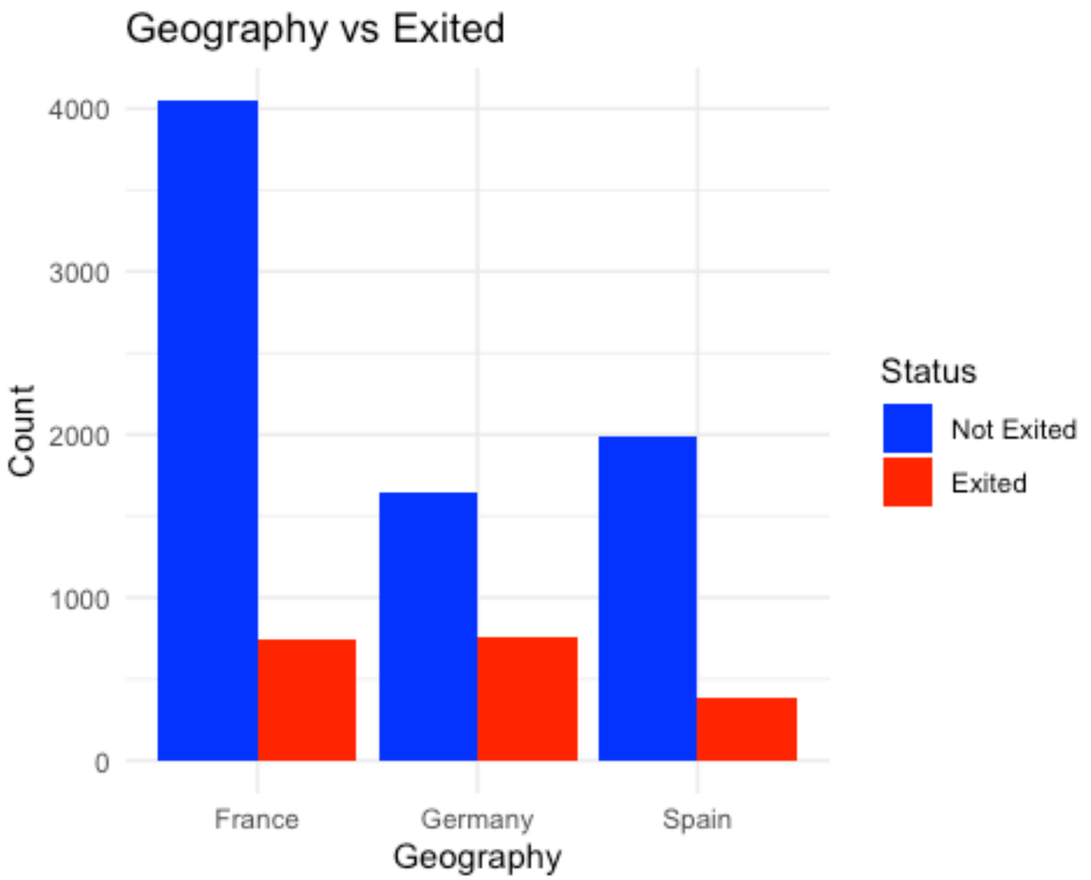
The bar plot compares the number of male and female customers who have either stayed with (blue) or exited (red) a service.

Both genders have more customers staying than leaving, with males slightly higher in both categories.

Bar plot for Geography vs Exited

```
ggplot(data, aes(x = Geography, fill = factor(Exited))) +
  geom_bar(position = "dodge") +
  scale_fill_manual(values = c("blue", "red"),
                    labels = c("Not Exited", "Exited")) +
  labs(title = "Geography vs Exited", x = "Geography", y = "Count",
```

```
fill = "Status") +
  theme_minimal()
```



The bar plot depicts the count of customers who have stayed (blue) and those who have left (red) across three countries: France, Germany, and Spain.

France shows the highest number of customers, with most staying. Germany has a notable proportion of customers leaving

While Spain has the fewest customers but a similar staying-leaving ratio to France.

Distribution of categorical features

```
gender_distribution <- table(data$Gender)
print(gender_distribution)
```

```
##
## Female   Male
##   4332   5236
```

```r
geography_distribution <- table(data$Geography)
print(geography_distribution)
```

```
##
##  France Germany   Spain
##    4798    2398    2372
```

Correlation matrix

```r
correlation_matrix <- cor(data_numerical)
print(correlation_matrix)
```

```
##                   CreditScore          Age        Tenure
Balance
## CreditScore      1.0000000000 -0.013937430 -0.0003320428
0.005912400
## Age             -0.0139374300  1.000000000 -0.0107524014
0.040236827
## Tenure          -0.0003320428 -0.010752401  1.0000000000
-0.013706487
## Balance          0.0059123995  0.040236827 -0.0137064869
1.000000000
## NumOfProducts    0.0100626686 -0.058914849  0.0137038971
-0.331503618
## HasCrCard       -0.0004804850 -0.016477541  0.0194741499
-0.013807100
## IsActiveMember   0.0213679748  0.018142498 -0.0288130950
-0.006999659
## EstimatedSalary  0.0029403377 -0.005950582  0.0095517744
0.010758769
##                 NumOfProducts    HasCrCard IsActiveMember
EstimatedSalary
## CreditScore       0.010062669 -0.000480485    0.021367975
0.002940338
## Age              -0.058914849 -0.016477541    0.018142498
-0.005950582
## Tenure            0.013703897  0.019474150   -0.028813095
0.009551774
```

```
## Balance          -0.331503618 -0.013807100   -0.006999659
0.010758769
## NumOfProducts     1.000000000  0.004840858    0.013095818
0.012906215
## HasCrCard         0.004840858  1.000000000   -0.012002344
-0.009634615
## IsActiveMember    0.013095818 -0.012002344    1.000000000
-0.010240339
## EstimatedSalary   0.012906215 -0.009634615   -0.010240339
1.000000000

numerical_features <- c("CreditScore", "Age", "Tenure", "Balance",
"NumOfProducts", "EstimatedSalary")
```

Converting data to dummies

```
data_dummies <- data %>% model.matrix(~ . - 1, data = .) %>%
as.data.frame()
dim(data_dummies)

## [1] 9568   13
```

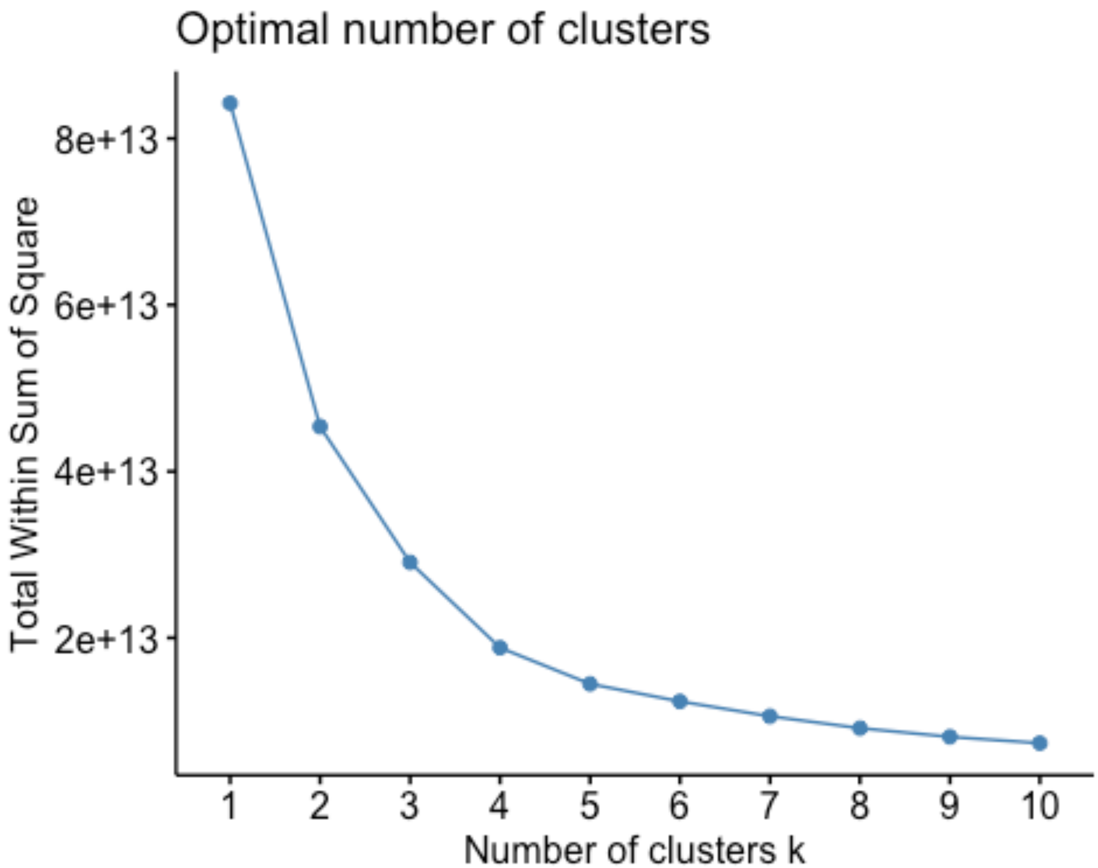
E) CLUSTERING

```
library(cluster)
library(factoextra)
```

Determine optimal number of clusters

```
fviz_nbclust(data, FUN = hcut, method = "wss")

## Warning in stats::dist(x): NAs introduced by coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion
```

```
## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion

## Warning in stats::dist(x, method = method, ...): NAs introduced by
coercion
```



#
————————-Clustering with K-means————————————————————— # K-means clustering with an assumed number of clusters (e.g., 5)

```
set.seed(123)
kmeans_result <- kmeans(data_dummies, centers = 2, nstart = 25)
fviz_cluster(kmeans_result, data = data_dummies)
```


```r
table(kmeans_result$cluster,data_dummies$Exited1)
```

```
##
##         0    1
##    1 4546 1382
##    2 3131  509
```

Values from the confusion matrix

```r
TN <- 4546  # True Negatives (Cluster 1, Class 0)
FN <- 1382  # False Negatives (Cluster 1, Class 1)
FP <- 3131  # False Positives (Cluster 2, Class 0)
TP <- 509   # True Positives (Cluster 2, Class 1)
```

Calculating accuracy, precision, sensitivity, and specificity

```
accuracy_kmeans <- (TP + TN) / (TP + TN + FP + FN)
precision_kmeans <- TP / (TP + FP)
sensitivity_kmeans <- TP / (TP + FN)  # Recall
specificity_kmeans <- TN / (TN + FP)
```

Print the results

```
cat("K-means Accuracy:", accuracy_kmeans, "\n")

## K-means Accuracy: 0.5283236

cat("K-means Precision:", precision_kmeans, "\n")

## K-means Precision: 0.1398352

cat("K-means Sensitivity (Recall):", sensitivity_kmeans, "\n")

## K-means Sensitivity (Recall): 0.2691698

cat("K-means Specificity:", specificity_kmeans, "\n")

## K-means Specificity: 0.5921584

cluster_labels <- ifelse(kmeans_result$cluster == 1, 1, 0)
```

Create a prediction object for ROC analysis

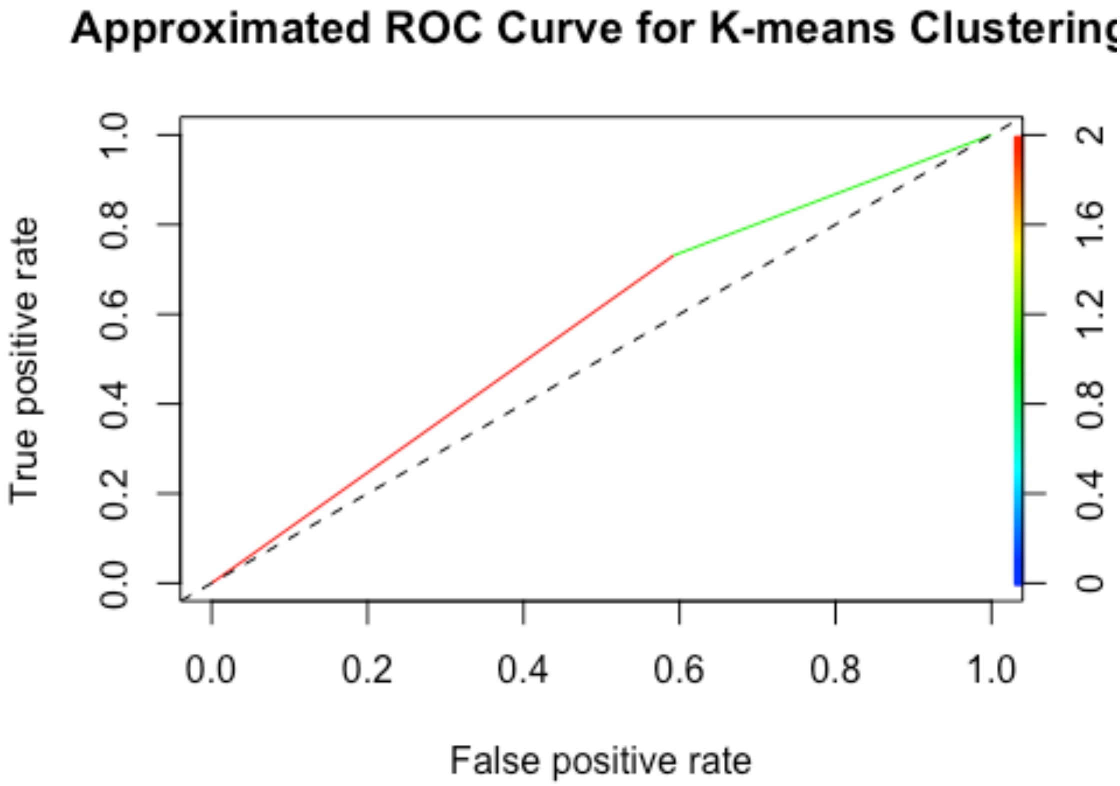
```
pred_kmeans <- prediction(cluster_labels, data_dummies$Exited1)
```

Create performance object for ROC curve

```
perf_kmeans <- performance(pred_kmeans, "tpr", "fpr")
```

Plot the ROC curve

```
plot(perf_kmeans, colorize = TRUE)
abline(a = 0, b = 1, lty = 2)  # Diagonal reference line
title(main = "Approximated ROC Curve for K-means Clustering")
```




```r
xlab("False Positive Rate")

## $x
## [1] "False Positive Rate"
##
## attr(,"class")
## [1] "labels"

ylab("True Positive Rate")

## $y
## [1] "True Positive Rate"
##
## attr(,"class")
## [1] "labels"
```

Calculate AUC

```
auc_kmeans <- performance(pred_kmeans, "auc")
auc_value_kmeans <- auc_kmeans@y.values[[1]]
```

Print the AUC

```
cat("Approximated AUC for K-means Clustering:", auc_value_kmeans,
"\n")

## Approximated AUC for K-means Clustering: 0.5693359
```

F) CLASSIFICATION

- **Decision Tree Model**

```
library(rpart)
library(caret)
```

Summarize the target variable

```
target_summary <- table(data$Exited) / nrow(data)

index <- createDataPartition(data_dummies$Exited1, p = 0.8, list =
FALSE)
train_data <- data_dummies[index, ]
test_data <- data_dummies[-index, ]

dim(test_data)

## [1] 1913   13

dim(train_data)

## [1] 7655   13
```

Train the decision tree model

```
model_tree <- rpart(Exited1 ~ ., data = train_data, method = "class")
```

Predict and evaluate the model

```
predictions_all <- predict(model_tree, data_dummies, type = "class")
table(predictions_all, data_dummies$Exited1)

##
## predictions_all    0    1
##               0 7457 1170
##               1  220  721
```

Values from the confusion matrix

```
TP <- 721  # True Positives
TN <- 7457 # True Negatives
FP <- 220  # False Positives
FN <- 1170 # False Negatives
```

Calculating accuracy, precision, sensitivity (recall), and specificity

```
accuracy <- (TP + TN) / (TP + TN + FP + FN)
precision <- TP / (TP + FP)
sensitivity <- TP / (TP + FN)  # Recall
specificity <- TN / (TN + FP)
```

Printing the results

```
cat("Accuracy:", accuracy, "\n")

## Accuracy: 0.8547241

cat("Precision:", precision, "\n")

## Precision: 0.7662062

cat("Sensitivity (Recall):", sensitivity, "\n")

## Sensitivity (Recall): 0.3812797

cat("Specificity:", specificity, "\n")

## Specificity: 0.971343
```

Predict probabilities for the positive class

```
predictions_tree_probs <- predict(model_tree, data_dummies, type =
"prob")[,2]
```

Create a prediction object using ROCR

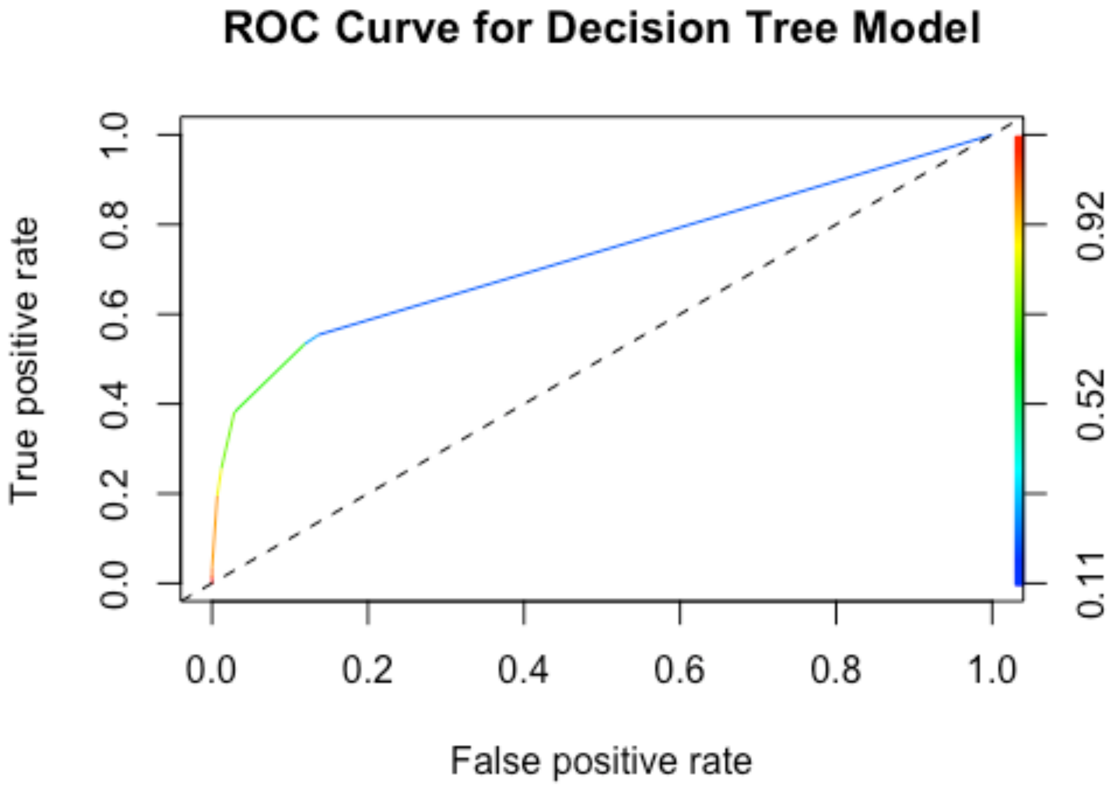
```
pred <- prediction(predictions_tree_probs, data_dummies$Exited1)
```

Create a performance object for TPR and FPR

```
perf <- performance(pred, "tpr", "fpr")
```

Plot the ROC curve

```
plot(perf, colorize = TRUE)
abline(a = 0, b = 1, lty = 2)  # Diagonal reference line
title(main = "ROC Curve for Decision Tree Model")
```




```
xlabel <- "False Positive Rate"
ylabel <- "True Positive Rate"
xlab(xlabel)

## $x
## [1] "False Positive Rate"
##
## attr(,"class")
## [1] "labels"

ylab(ylabel)

## $y
## [1] "True Positive Rate"
##
```

```
## attr(,"class")
## [1] "labels"
```

Calculate the AUC

```
auc <- performance(pred, "auc")
auc_value <- auc@y.values[[1]]
```

Print the AUC

```
cat("AUC for Decision Tree Model:", auc_value, "\n")

## AUC for Decision Tree Model: 0.7290228
```

- **Support Vector Machine(SVM)**

```
library(e1071)
```

Train the SVM model

```
model_svm <- svm(Exited1 ~ ., data = train_data, type = "C-
classification", kernel = "radial")
```

Predict and evaluate the model

```
predictions_svm <- predict(model_svm, test_data)
table(predictions_svm, test_data$Exited1)

##
## predictions_svm    0    1
##               0 1474  249
##               1   49  141
```

Values from the confusion matrix for SVM

```
TP <- 159  # True Positives
TN <- 1459 # True Negatives
FP <- 43   # False Positives
FN <- 252  # False Negatives
```

Calculating accuracy, precision, sensitivity (recall), and specificity

```
accuracy_svm <- (TP + TN) / (TP + TN + FP + FN)
precision_svm <- TP / (TP + FP)
sensitivity_svm <- TP / (TP + FN)  # Recall
specificity_svm <- TN / (TN + FP)
```

Printing the results for SVM

```
cat("SVM Accuracy:", accuracy_svm, "\n")

## SVM Accuracy: 0.8457919

cat("SVM Precision:", precision_svm, "\n")

## SVM Precision: 0.7871287

cat("SVM Sensitivity (Recall):", sensitivity_svm, "\n")

## SVM Sensitivity (Recall): 0.3868613

cat("SVM Specificity:", specificity_svm, "\n")

## SVM Specificity: 0.9713715
```

ROC CURVE

Get predicted probabilities

```
svm_probabilities <- predict(model_svm, test_data, decision.values =
TRUE)
attr(svm_probabilities, "decision.values") -> decision_values
```

Create a prediction object

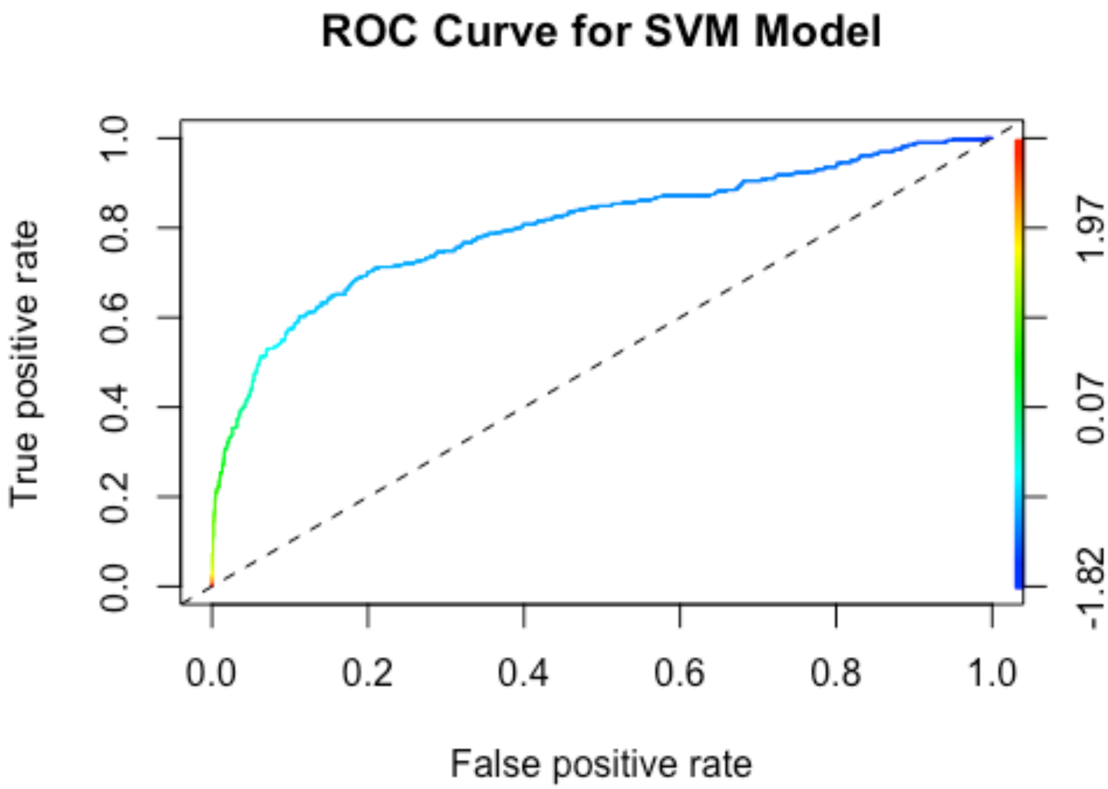
```
pred_svm <- prediction(decision_values, test_data$Exited1)
```

Create a performance object

```
perf_svm <- performance(pred_svm, "tpr", "fpr")
```

Plot the ROC curve

```
plot(perf_svm, colorize = TRUE)
abline(a = 0, b = 1, lty = 2)  # Diagonal reference line
title(main = "ROC Curve for SVM Model")
```




```
xlabel <- "False Positive Rate"
ylabel <- "True Positive Rate"
xlab(xlabel)

## $x
## [1] "False Positive Rate"
##
## attr(,"class")
## [1] "labels"

ylab(ylabel)
```

```
## $y
## [1] "True Positive Rate"
##
## attr(,"class")
## [1] "labels"
```

Calculate the AUC

```
auc_svm <- performance(pred_svm, measure = "auc")
auc_value_svm <- auc_svm@y.values[[1]]
```

Print the AUC

```
cat("AUC for SVM Model:", auc_value_svm, "\n")

## AUC for SVM Model: 0.8012846
```

G) EVALUATION

```
metrics <- data.frame(
    Model = c("K-means Clustering", "Decision Tree", "SVM Model"),
    Accuracy = c(0.8457919, 0.8547241, 0.8457919),
    Precision = c(0.7871287, 0.7662062, 0.7871287),
    Sensitivity = c(0.3868613, 0.3812797, 0.3868613),  # Sensitivity
is Recall
    Specificity = c(0.9713715, 0.971343, 0.9713715),
    AUC = c(0.5693359, 0.7290228, 0.8012846)
  )
```

Print the result table

```
print(metrics)

##                  Model  Accuracy Precision Sensitivity Specificity
AUC
## 1 K-means Clustering 0.8457919 0.7871287   0.3868613   0.9713715
0.5693359
## 2      Decision Tree 0.8547241 0.7662062   0.3812797   0.9713430
0.7290228
```

```
## 3         SVM Model 0.8457919 0.7871287   0.3868613   0.9713715
0.8012846
```

Overall Performance: The Decision Tree has the highest accuracy, suggesting it might be the best overall performer in terms of correctly classifying cases.

Balanced Performance: The SVM Model has the highest AUC, indicating it has the best balance between sensitivity and specificity and is better at distinguishing between the two classes.

Specificity: K-means Clustering, SVM, and Decision Tree have almost the same specificity, which is quite high.

Precision and Sensitivity: K-means Clustering and SVM are tied for both precision and sensitivity.