Credit Risk Modelling

Behdad Ehsani

The project comprises two main phases: the first phase involves exploratory data analysis, while the second phase involves building a predictive model.

The first step in the machine learning process is to read in the data.

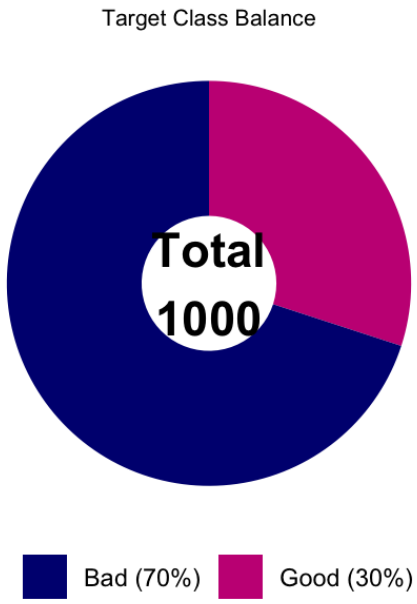
```
data <- read.csv('german_credit_data.csv')
```

Explanatory Data Analysis (EDA)

Target class balance

The donut chart illustrates the distribution of credit risk data in the dataset, categorizing it into good and bad credit. Among the 1000 individuals included in the study, 30% have been classified as having bad credit risk while the remaining individuals have been categorized as good credit risk.

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Target class balance
label_names <- c("Good", "Bad")
color_list <- c("navy", "mediumvioletred")
total <- nrow(data)
title <- "Target Class Balance"

# Create donut plot function
donut_plot <- function(df, col, label_names, colors, title, text) {
  data <- df %>%
    group_by(.data[[col]]) %>%
    summarise(count = n()) %>%
    mutate(percentage = count / sum(count) * 100,
           label = paste0(label_names, " (", round(percentage, 1), "%)"))

  ggplot(data, aes(x = "", y = count, fill = label)) +
    geom_bar(stat = "identity", width = 1) +
    coord_polar("y", start = 0) +
    scale_fill_manual(values = colors) +
    theme_void() +
    labs(title = title) +
    theme(plot.title = element_text(hjust = 0.5, size = 8),
          legend.title = element_blank(),
          legend.position = "bottom") +
    annotate("text", x = 0, y = 0, label = paste0("Total\n", text),
             size = 6, fontface = "bold", color = "black")
}

# Visualizing it through a donut chart
donut_plot(df = data, col = 'Risk', label_names = label_names, colors = color_list,
title = title, text = total)
```

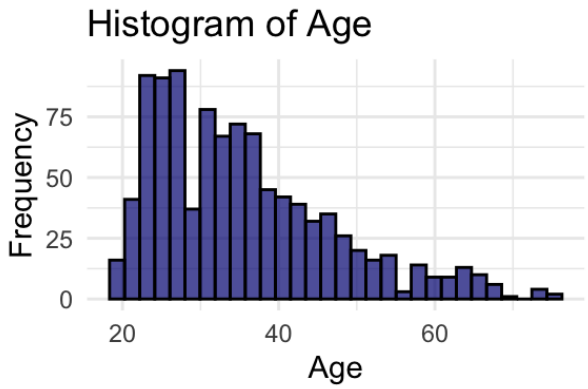


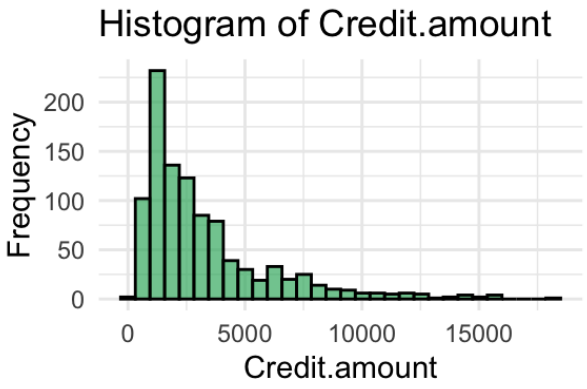
Feature Analysis

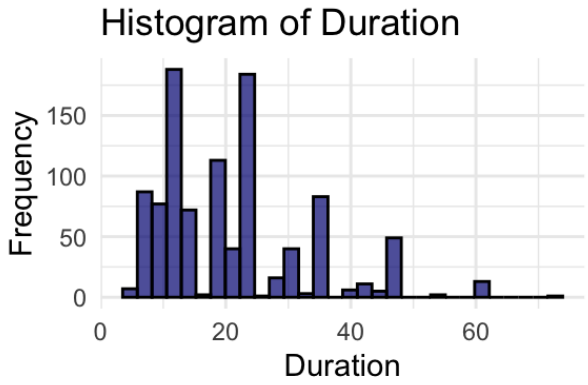
Numerical Feature Analysis

The distribution of the numerical variables in our study has been examined, and the figures indicate that the *Age* and *Credit_amount* variables follow a Gamma distribution that is skewed to the right. However, the distribution of the *Duration* variable is not clearly discernible.

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Define column names
num_cols <- c("Age", "Credit.amount", "Duration")

# Define color sequence
color_sequence <- c("navy", "mediumseagreen", "navy")

# Create a list to store the plots
numplot_list <- list()

# Loop through the columns and create histograms for each
for (i in 1:length(num_cols)) {
  p <- ggplot(data = data, aes_string(x = num_cols[i])) +
    geom_histogram(fill = color_sequence[i], color = "black", alpha = 0.7, bins = 3
0) +
    labs(title = paste("Histogram of", num_cols[i]), x = num_cols[i], y = "Frequenc
y") +
    theme_minimal()
  numplot_list[[i]] <- p
}

# Display the plots
for (i in 1:length(numplot_list)) {
  print(numplot_list[[i]])
}
```

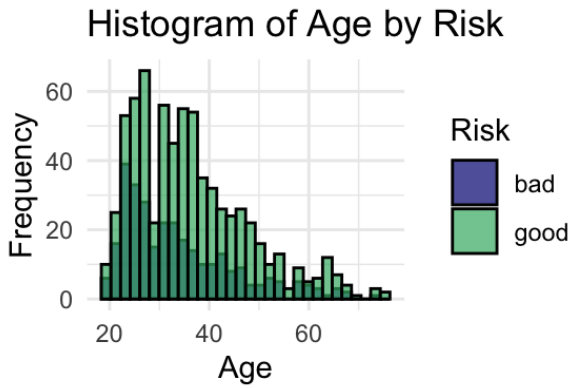


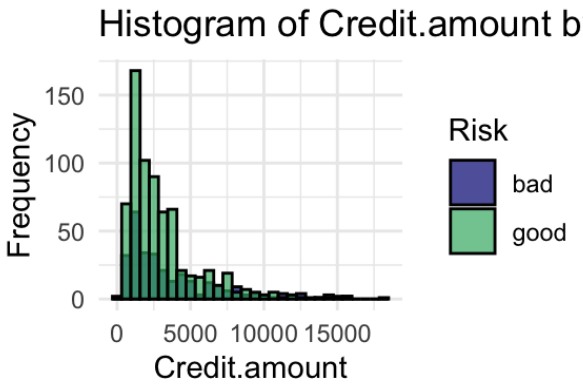


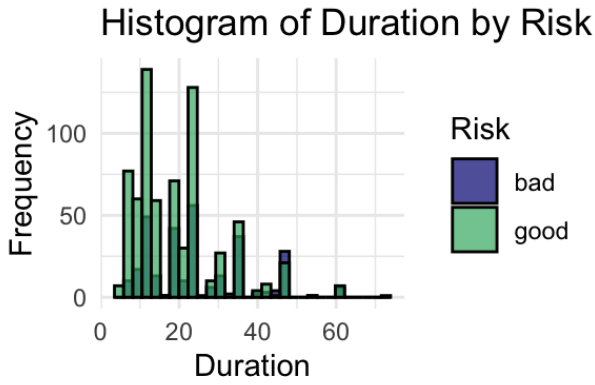


Upon further analysis, we have observed an improvement in the distribution of the numerical variables, particularly in the classification based on the Risk variable. For both the Age and Credit_amount variables, it has been determined that the distribution for each class is Gamma, which is skewed to the right. This new finding applies to both good and bad credit risk classes.

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)

# Add 'risk' to the column names
num_cols <- c("Age", "Credit.amount", "Duration")
num_cols <- c(num_cols, "Risk")

# Define color list
color_list <- c("navy", "mediumseagreen", "navy")

# Create a list to store the plots
numplot_list <- list()

# Loop through the columns and create histograms for each, except for the 'risk' co
lumn
for (i in 1:(length(num_cols) - 1)) {
  p <- ggplot(data = data, aes_string(x = num_cols[i], fill = "Risk")) +
    geom_histogram(position = "identity", color = "black", alpha = 0.7, bins = 30)
+
    scale_fill_manual(values = color_list) +
    labs(title = paste("Histogram of", num_cols[i], "by Risk"), x = num_cols[i], y
= "Frequency") +
    theme_minimal()
  numplot_list[[i]] <- p
}

# Display the plots
for (i in 1:length(numplot_list)) {
  print(numplot_list[[i]])
}
```

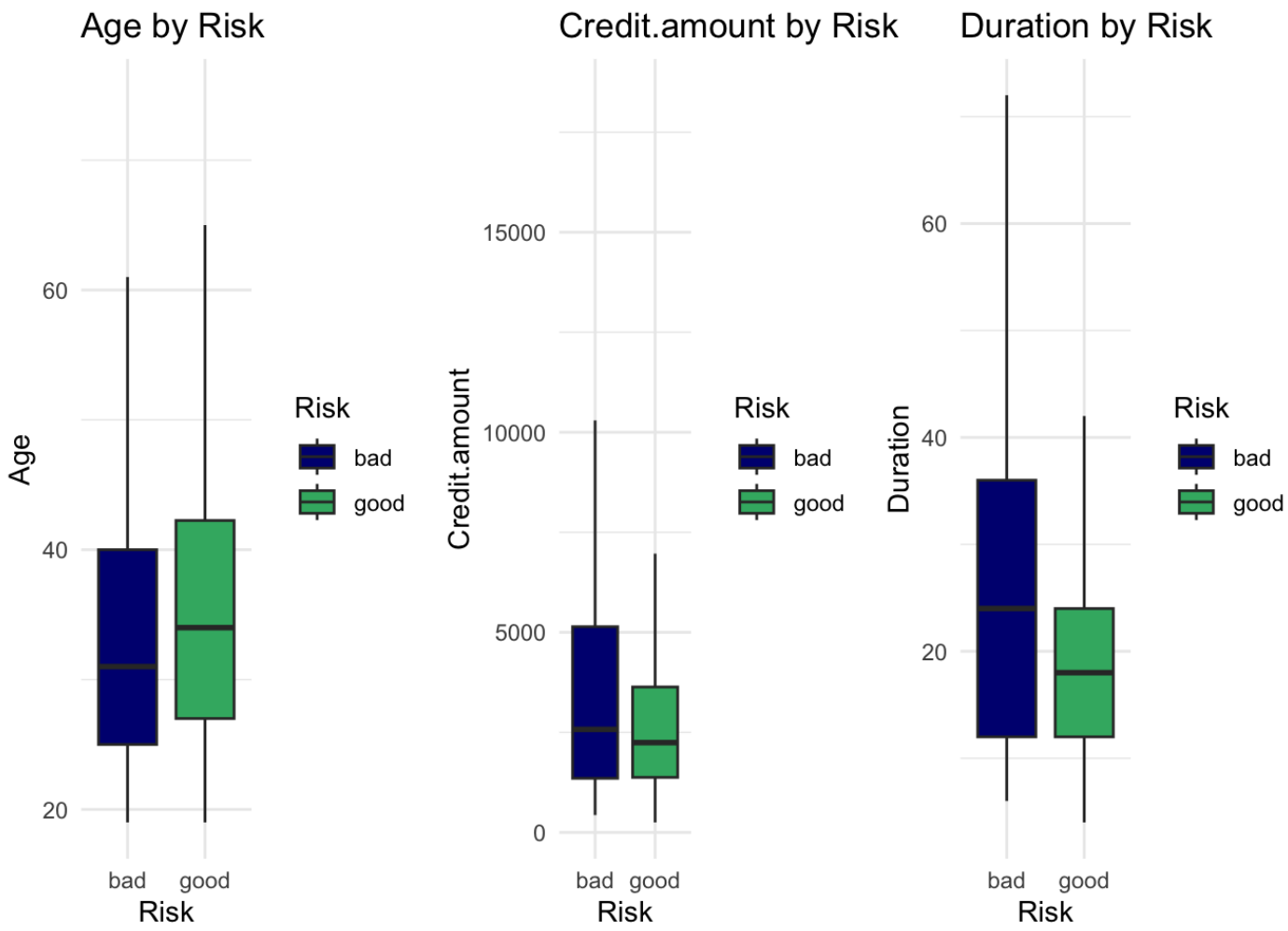






Here is another analysis with Box Plot:

```r
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(gridExtra)

# Define the features and hue
features <- c("Age", "Credit.amount", "Duration")
hue <- "Risk"

# Define the color list
color_list <- c("navy", "mediumseagreen", "navy")

# Create a list to store the plots
boxenplot_list <- list()

# Loop through the features and create boxen plots for each
for (i in 1:length(features)) {
  p <- ggplot(data = data, aes_string(x = hue, y = features[i], fill = hue)) +
    geom_boxplot(outlier.shape = NA) +
    scale_fill_manual(values = color_list) +
    labs(title = paste(features[i], "by Risk"), x = "Risk", y = features[i]) +
    theme_minimal()
  boxenplot_list[[i]] <- p
}

# Arrange the plots in a grid
grid.arrange(grobs = boxenplot_list, ncol = 3)
```



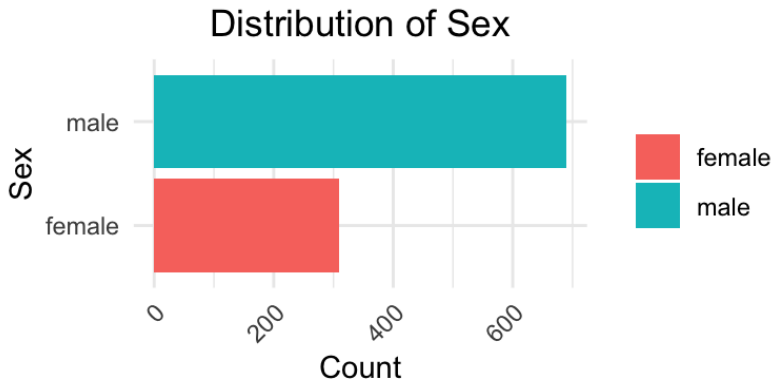
Based on the previous analysis on Numerical data, some findings are presented in the following:

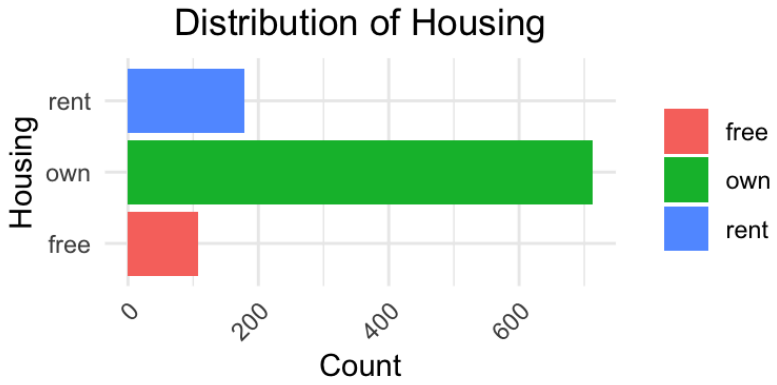
1. Younger individuals are often considered riskier than their older counterparts.
2. This idea is logical, as older people generally have greater financial stability than younger ones;
3. Larger credit sums pose a greater risk than smaller ones.
4. This notion is fairly straightforward and entirely reasonable;
5. Longer loan duration are associated with increased risk.

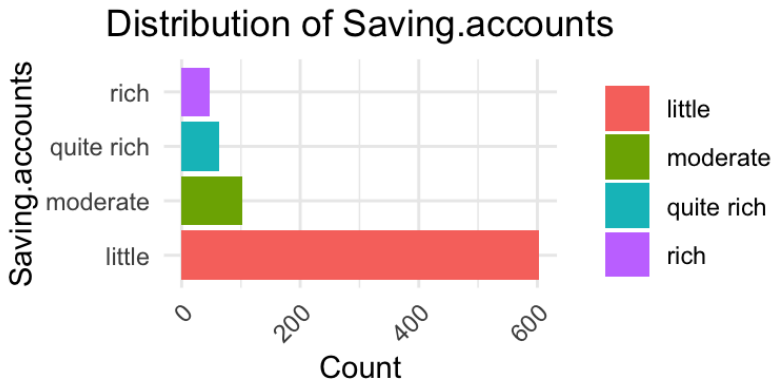
Categorical Feature Analysis

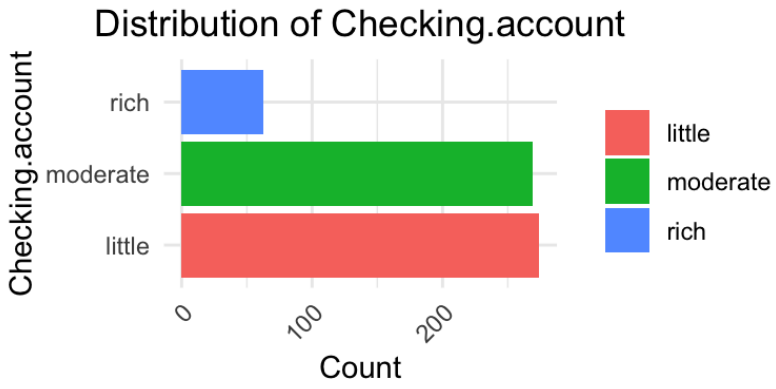
The following displays the quantity for each category.

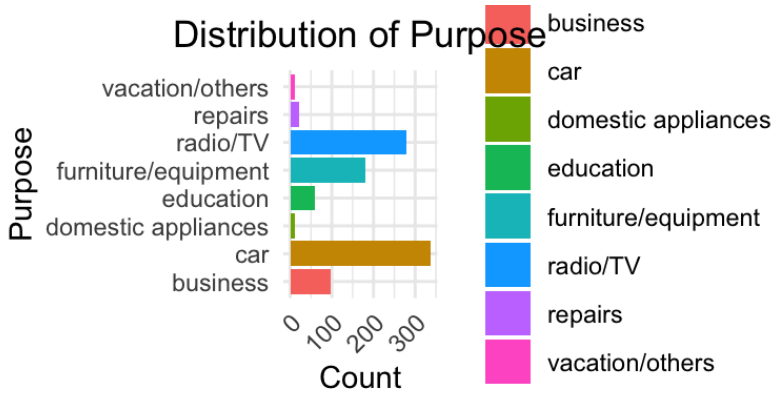
```
data$Sex <- factor(data$Sex)
data$Housing <- factor(data$Housing)
data$Risk <- factor(data$Risk)
data$Saving.accounts <- factor(data$Saving.accounts)
data$Checking.account <- factor(data$Checking.account)
data$Purpose <- factor(data$Purpose)

# Get categorical features
cat_features <- names(data)[sapply(data, is.factor)]

# Function to create bar plots for categorical variables
create_bar_plots <- function(data, cat_features) {
  for (feature in cat_features) {
    plot_data <- data %>%
      filter(!is.na(.data[[feature]])) %>%
      group_by(.data[[feature]]) %>%
      summarize(count = n())

    plot <- ggplot(plot_data, aes(x = .data[[feature]], y = count, fill = .data[[fe
ature]])) +
      geom_bar(stat = "identity", position = "dodge") +
      labs(title = paste0("Distribution of ", feature),
           x = feature,
           y = "Count") +
      theme_minimal() +
      theme(plot.title = element_text(hjust = 0.5),
            axis.text.x = element_text(angle = 45, hjust = 1),
            legend.title = element_blank()) +
      coord_flip()

    print(plot)
  }
}

# Call create_bar_plots function
create_bar_plots(data = data, cat_features = cat_features)
```

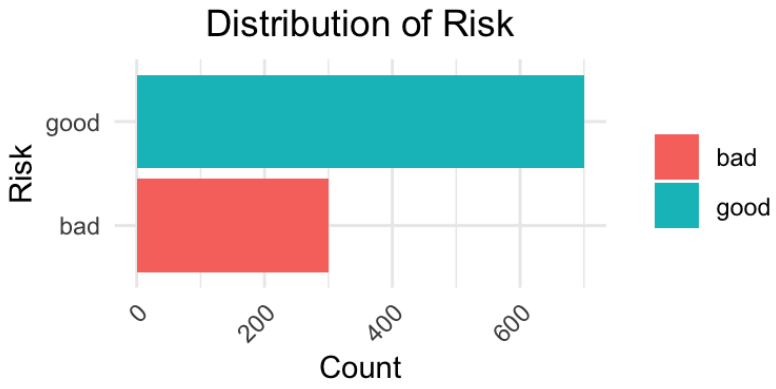

Distribution of Sex
Sex
male
female
0
200
400
600
Count
female
male


Distribution of Housing
Housing
rent
own
free
0
200
400
600
Count
free
own
rent


Distribution of Saving.accounts
Saving.accounts
rich
quite rich
moderate
little
0
200
400
600
Count
little
moderate
quite rich
rich

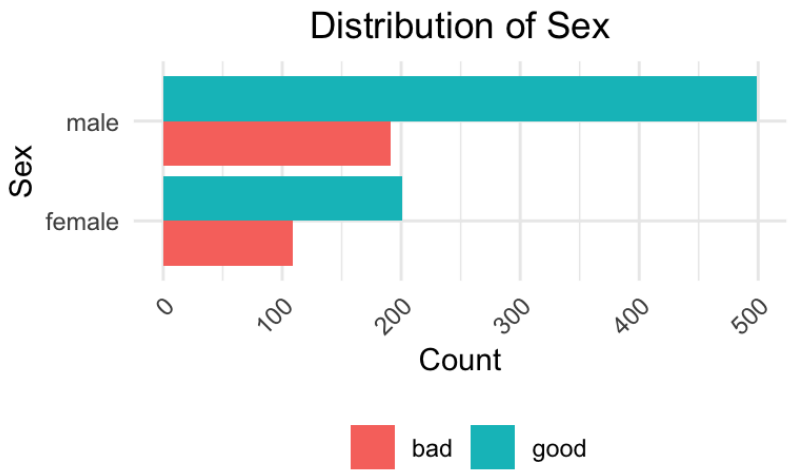


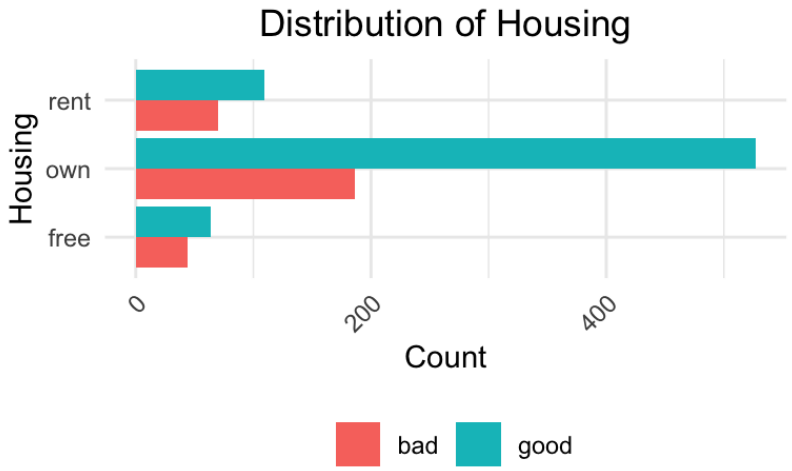


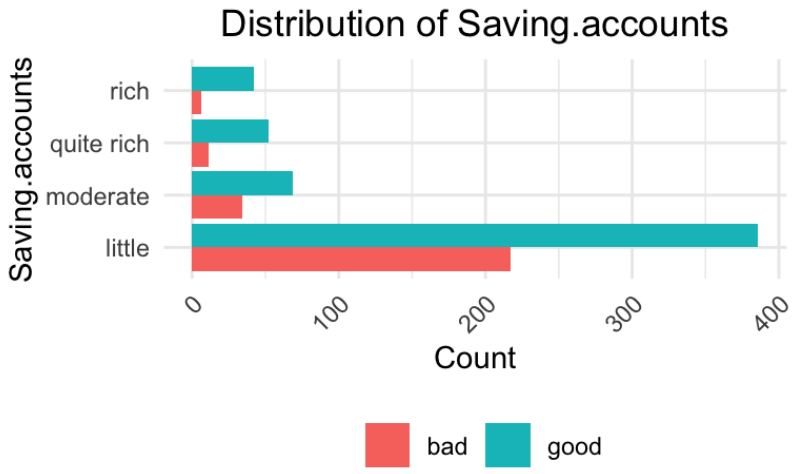


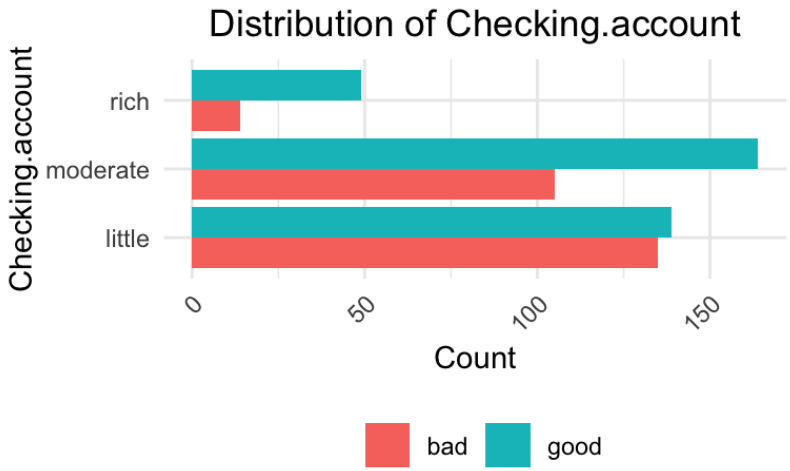
The analysis of these categories gives us a summary of the quantity of data in our dataset. For every categorical column, we can now identify the predominant entries. To enhance the analysis further, we can classify each categorical entry based on its risk approach.

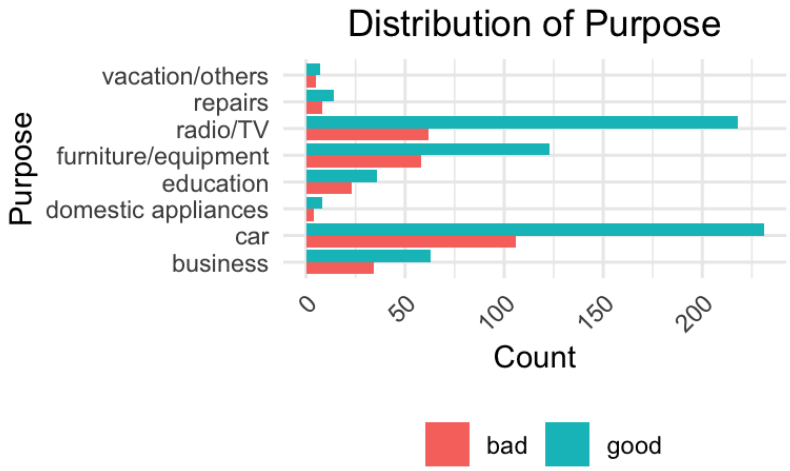
```r
# Get categorical features
cat_features <- names(data)[sapply(data, is.factor)]

# Function to create bar plots for categorical variables
create_bar_plots <- function(data, cat_features) {
  for (feature in cat_features) {
    if (feature == "Risk") {
      next # skip Risk feature
    }
    plot_data <- data %>%
      filter(!is.na(.data[[feature]])) %>%
      group_by(.data[[feature]], Risk) %>%
      summarize(count = n(), .groups = "drop")

    plot <- ggplot(plot_data, aes(x = .data[[feature]], y = count, fill = Risk)) +
      geom_bar(stat = "identity", position = "dodge") +
      labs(title = paste0("Distribution of ", feature),
           x = feature,
           y = "Count",
           fill = "Risk") +
      theme_minimal() +
      theme(plot.title = element_text(hjust = 0.5),
            axis.text.x = element_text(angle = 45, hjust = 1),
            legend.title = element_blank(),
            legend.position = "bottom") +
      coord_flip()

    print(plot)
  }
}

# Call create_bar_plots function
create_bar_plots(data = data, cat_features = cat_features)
```


Distribution of Sex
Sex
male
female
0
100
200
300
400
500
Count
bad
good


Distribution of Housing
Housing
rent
own
free
0
200
400
Count
bad
good


Distribution of Saving.accounts
Saving.accounts
rich
quite rich
moderate
little
0
100
200
300
400
Count
bad
good





Some insights taken from the categorical feature analysis:

1. Individuals who have low savings and checking account balances pose a greater risk and vice versa.
2. If customers are taking out credit specifically for vacations or other non-essential expenses, it could be an indication of higher risk.
3. Individuals who own their own homes tend to have lower risk.
4. Men tend to have lower risk rather women.

Insights in Data

What was the amount of credit extended to customers with a high level of risk?

A donut chart has been provided to address the question, which illustrates that individuals with a high risk were assigned 36.1 of the total credit, amounting to $3,271,258\$$.

```
library(plotly)

# Calculate the sum and percentage of credit given to "bad" and "good" risks
risk_summary <- data %>%
  group_by(Risk) %>%
  summarize(credit_sum = sum(Credit.amount, na.rm = TRUE),
            credit_pct = (credit_sum / sum(data$Credit.amount, na.rm = TRUE)) * 10
0)

# Filter the summary data for "bad" and "good" risks
risk_summary <- risk_summary %>%
  filter(Risk %in% c("bad", "good"))

# Create a donut chart using plotly
plot_ly(risk_summary, labels = ~Risk, values = ~credit_sum, type = 'pie', hole = 0.
5,
        textposition = 'inside', textinfo = 'percent+label',
        marker = list(colors = c("#FF6F61", "#5B9BD5"), line = list(color = '#FFFFF
F', width = 2))) %>%
  layout(title = list(text = "Credit Sum and Percentage by Risk"),
         xaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         yaxis = list(showgrid = FALSE, zeroline = FALSE, showticklabels = FALSE),
         showlegend = FALSE)
```

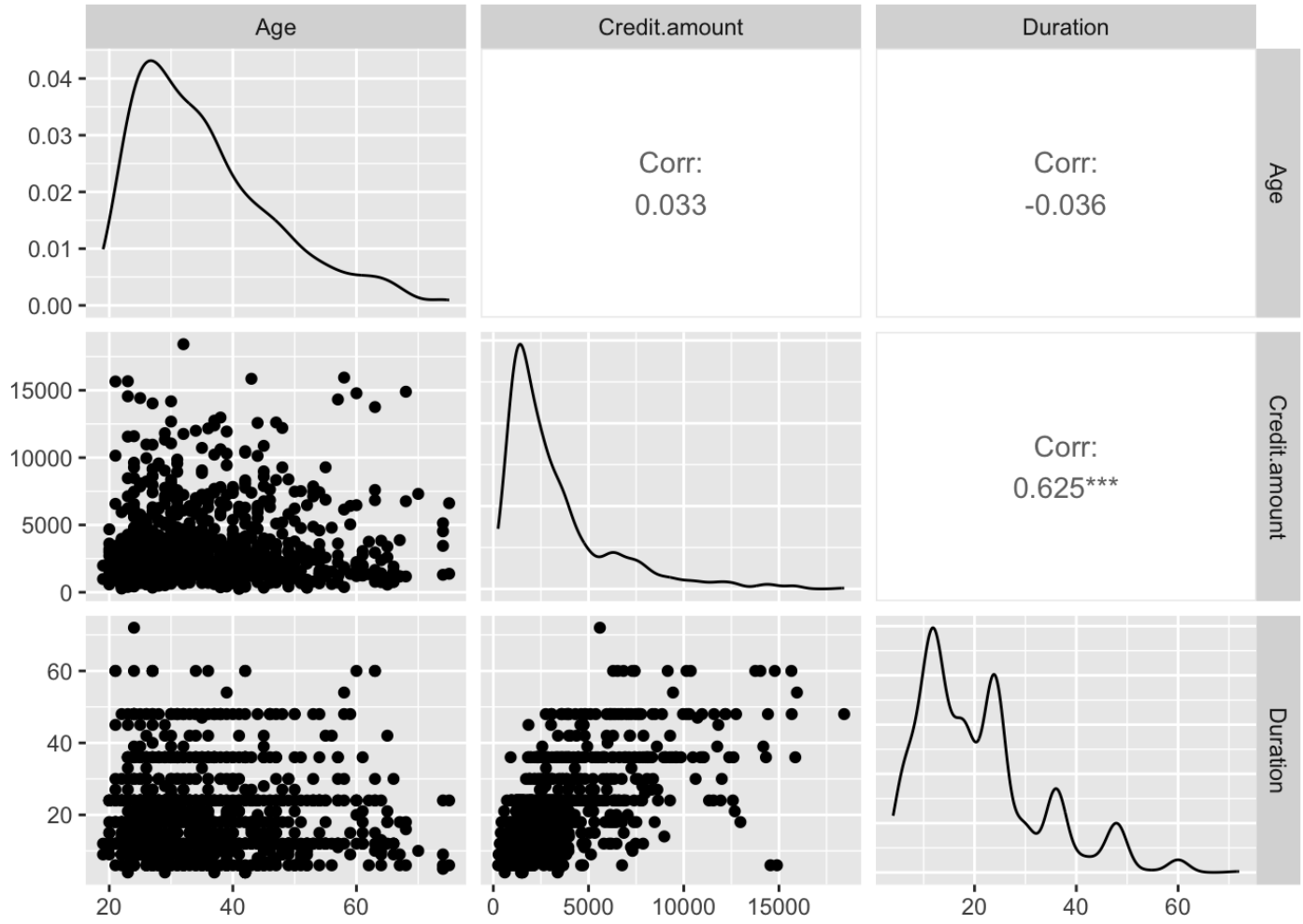
Do any of the features have a correlation with each other?

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
num_cols<-c('Age', 'Credit.amount', 'Duration')
# Assuming df is the data frame and num_cols is a vector of numeric column names
ggpairs(data[num_cols], palette = color_list)
```

```
## Warning in warn_if_args_exist(list(...)): Extra arguments: 'palette' are being
## ignored. If these are meant to be aesthetics, submit them using the 'mapping'
## variable within ggpairs with ggplot2::aes or ggplot2::aes_string.
```



The plots included in the pairplot suggest that there may be a positive correlation between the credit amount and the duration of the credit request. This correlation is logical, as longer-term credits are typically associated with higher amounts.

Prediction Section

Following a thorough exploratory data analysis (EDA), several machine learning models have been employed to address the credit risk problem. The aim of this section is to determine the model that performs best in identifying individuals with either good or bad risk.

Data Preprocessing

Before modeling, it is essential to check for any missing values. If any null values are found, they should be imputed before proceeding with the modeling process.

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

```
##                 X               Age               Sex               Job
##                 0                 0                 0                 0
##           Housing  Saving.accounts Checking.account    Credit.amount
##                 0               183               394                 0
##          Duration           Purpose              Risk
##                 0                 0                 0
```

It has been observed that there are missing values in the variables *Saving.accounts* and *Checking.account*.

To impute the missing values in the *Saving.accounts* and *Checking.account* variables, a **KNN** (k-nearest neighbors) model has been employed.

```
library(DMwR2)

# Perform kNN imputation
data_imputed <- knnImputation(data, k = 5)  # 'k' specifies the number of neighbors
```

Once the missing values have been imputed, it is advisable to convert the categorical variables into binary (0/1) form using one-hot encoding technique.

```
# Create a dummyVars object with the desired formula
library(caret)
```

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

```
data <- data_imputed

formula <- formula(~ Sex + Housing + Saving.accounts + Checking.account + Purpose +
Risk)

dummy_variables <- dummyVars(formula, data = data)

# Generate one-hot encoded data frame
one_hot_encoded_data <- as.data.frame(predict(dummy_variables, newdata = data))

# Combine the one-hot encoded data frame with the original data frame
data_encoded <- cbind(data, one_hot_encoded_data)

# Remove the original categorical column
data_encoded$Sex <- NULL
data_encoded$Housing <- NULL
data_encoded$Saving.accounts <- NULL
data_encoded$Checking.account <- NULL
data_encoded$Purpose <- NULL
data_encoded$Risk <- NULL
data_encoded$X <- NULL

#removing a class from each categorical feature (n-1 features are enough for n cate
gory)
data_encoded$Sex.female <- NULL
data_encoded$Housing.free <- NULL
data_encoded$Saving.accounts.little <- NULL
data_encoded$Checking.account.little <- NULL
data_encoded$Purpose.business <- NULL
data_encoded$Risk.bad <- NULL
```

Modelling

With the data prepared (missing values imputed and categorical variables transformed into binary form using one-hot encoding), it is now ready to be used to train machine learning models. Several models can be evaluated by looking at their confusion matrix and comparative metrics, including accuracy, precision, recall, and F1 score.

Some of the popular machine learning models for credit risk prediction are:

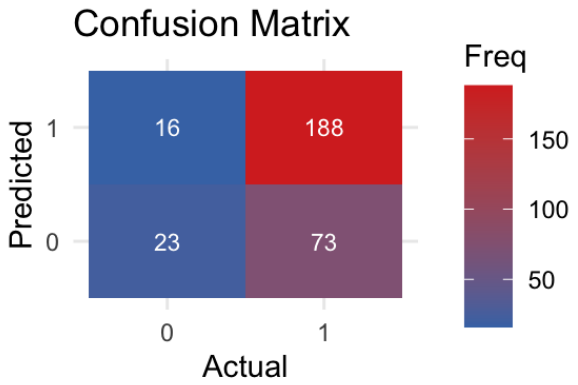
1. Logistic Regression
2. Random Forest
3. XGBoost
4. K-Nearest Neighbors (KNN)
5. Naive Bayes

Once trained, each model's performance can be evaluated by comparing its confusion matrix and metrics. Based on the evaluation, the best-performing model can be chosen for deployment.

Random Forest (RF)

```
# Load the required packages
library(randomForest)
library(caret)

# Split the data into training and testing sets
set.seed(123)
trainIndex <- sample(1:nrow(data_encoded), 0.7*nrow(data_encoded))
trainData <- data_encoded[trainIndex,]
testData <- data_encoded[-trainIndex,]

# Create the random forest model
rfModel <- randomForest(Risk.good ~ ., data = trainData, ntree = 500)

# Make predictions on the testing set
predictions_rf <- predict(rfModel, testData)



threshold <- 0.5
class_pred <- ifelse(predictions_rf > threshold, 1, 0)

actual <- testData$Risk.good

class_pred <- as.numeric(class_pred)

# Evaluate the performance of the model
class_pred_factor <- factor(class_pred, levels = c(0, 1))
actual_factor <- factor(actual, levels = c(0, 1))

# Confusion Matrix (CM)
cm <- confusionMatrix(class_pred_factor, actual_factor)
cm_rf <- cm

# Visualizing CM
visualize_confusion_matrix <- function(cm) {
  cm_table <- as.table(cm$table)

  cm_df <- as.data.frame(cm_table)
  cm_df$Prediction <- as.character(cm_df$Prediction) # Predicted
  cm_df$Reference <- as.character(cm_df$Reference) # Actual

  ggplot(cm_df, aes(x = Prediction, y = Reference, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), color = "white", size = 3) +
    labs(x = "Actual", y = "Predicted", title = "Confusion Matrix") +
    scale_fill_gradient(low = "#4575b4", high = "#d73027") +
    theme_minimal()
}
```

```
# Plot the confusion matrix
plot <- visualize_confusion_matrix(cm)
print(plot)
```

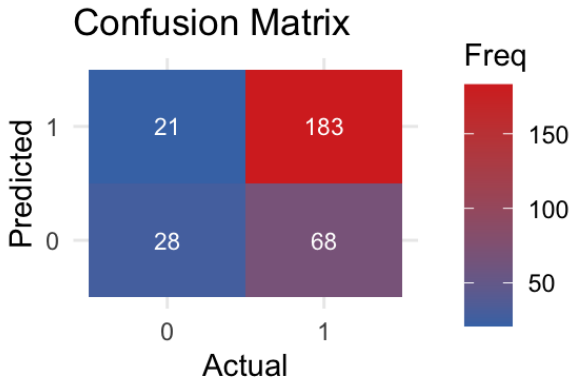



```
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  23  16
##          1  73 188
##
##                Accuracy : 0.7033
##                  95% CI : (0.6481, 0.7545)
##     No Information Rate : 0.68
##     P-Value [Acc > NIR] : 0.2113
##
##                   Kappa : 0.1912
##
##  Mcnemar's Test P-Value : 2.921e-09
##
##             Sensitivity : 0.23958
##             Specificity : 0.92157
##          Pos Pred Value : 0.58974
##          Neg Pred Value : 0.72031
##              Prevalence : 0.32000
##          Detection Rate : 0.07667
##    Detection Prevalence : 0.13000
##       Balanced Accuracy : 0.58058
##
##        'Positive' Class : 0
##
```

XGBoost

```
# Plot the confusion matrix
plot <- visualize_confusion_matrix(cm)
print(plot)
```



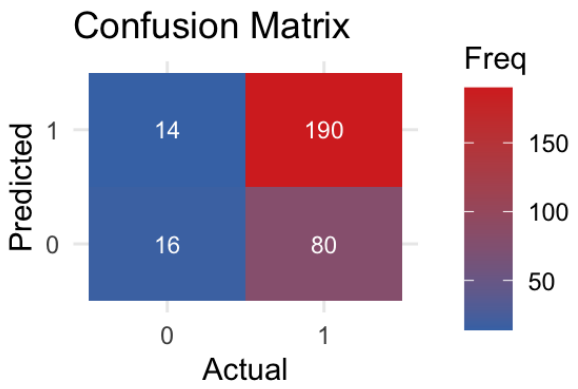

```
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  28  21
##          1  68 183
##
##                Accuracy : 0.7033
##                  95% CI : (0.6481, 0.7545)
##     No Information Rate : 0.68
##     P-Value [Acc > NIR] : 0.2113
##
##                   Kappa : 0.2168
##
##  Mcnemar's Test P-Value : 1.083e-06
##
##             Sensitivity : 0.29167
##             Specificity : 0.89706
##          Pos Pred Value : 0.57143
##          Neg Pred Value : 0.72908
##              Prevalence : 0.32000
##          Detection Rate : 0.09333
##    Detection Prevalence : 0.16333
##       Balanced Accuracy : 0.59436
##
##        'Positive' Class : 0
##
```

Logistic Regression (LR)

```
# Load the required packages
library(glmnet)

# Split the data into training and testing sets
set.seed(123)
trainIndex <- sample(1:nrow(data_encoded), 0.7*nrow(data_encoded))
trainData <- data_encoded[trainIndex,]
testData <- data_encoded[-trainIndex,]

# Create the logistic regression model
logRegModel <- glm(Risk.good ~ ., data = trainData, family = "binomial")

# Make predictions on the testing set
predictions_LR <- predict(logRegModel, testData, type = "response")

threshold <- 0.5
class_pred <- ifelse(predictions_LR > threshold, 1, 0)

actual <- testData$Risk.good

class_pred <- as.numeric(class_pred)

# Evaluate the performance of the model
class_pred_factor <- factor(class_pred, levels = c(0, 1))
actual_factor <- factor(actual, levels = c(0, 1))

# Confusion Matrix (CM)
cm <- confusionMatrix(class_pred_factor, actual_factor)
cm_lr <- cm

# Visualizing CM
visualize_confusion_matrix <- function(cm) {
  cm_table <- as.table(cm$table)

  cm_df <- as.data.frame(cm_table)
  cm_df$Prediction <- as.character(cm_df$Prediction) # Predicted
  cm_df$Reference <- as.character(cm_df$Reference) # Actual

  ggplot(cm_df, aes(x = Prediction, y = Reference, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), color = "white", size = 3) +
    labs(x = "Actual", y = "Predicted", title = "Confusion Matrix") +
    scale_fill_gradient(low = "#4575b4", high = "#d73027") +
    theme_minimal()
}

# Plot the confusion matrix
```

```
plot <- visualize_confusion_matrix(cm)
print(plot)
```

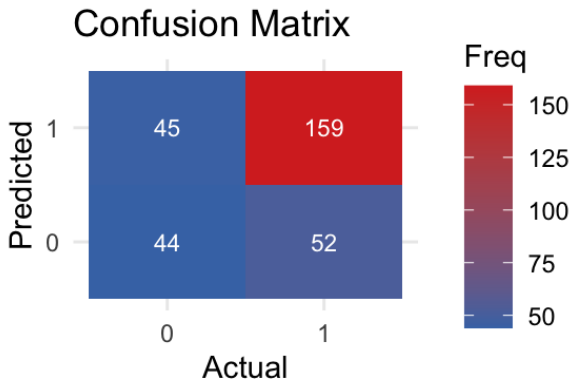



```
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  16  14
##          1  80 190
##
##                Accuracy : 0.6867
##                  95% CI : (0.6309, 0.7387)
##     No Information Rate : 0.68
##     P-Value [Acc > NIR] : 0.4292
##
##                   Kappa : 0.1199
##
##  Mcnemar's Test P-Value : 2.025e-11
##
##             Sensitivity : 0.16667
##             Specificity : 0.93137
##          Pos Pred Value : 0.53333
##          Neg Pred Value : 0.70370
##              Prevalence : 0.32000
##          Detection Rate : 0.05333
##    Detection Prevalence : 0.10000
##       Balanced Accuracy : 0.54902
##
##        'Positive' Class : 0
##
```

Naïve Bayes (NB)

```
# Load the required packages
library(ggplot2)
library(caret)
library(e1071)


# Split the data into training and testing sets
set.seed(123)
trainIndex <- sample(1:nrow(data_encoded), 0.7*nrow(data_encoded))
trainData <- data_encoded[trainIndex,]
testData <- data_encoded[-trainIndex,]

# Create the NB model
nbModel <- naiveBayes(Risk.good ~ ., data = trainData)

predictions_nb_ROC <- predict(nbModel, testData, type = "raw")
predictions_nb <- predict(nbModel, testData)

actual <- testData$Risk.good

# Evaluate the performance of the model
class_pred_factor <- factor(predictions_nb, levels = c(0, 1))
actual_factor <- factor(actual, levels = c(0, 1))

# Confusion Matrix (CM)
cm <- confusionMatrix(class_pred_factor, actual_factor)
cm_nb <- cm

# Visualizing CM
visualize_confusion_matrix <- function(cm) {
  cm_table <- as.table(cm$table)

  cm_df <- as.data.frame(cm_table)
  cm_df$Prediction <- as.character(cm_df$Prediction) # Predicted
  cm_df$Reference <- as.character(cm_df$Reference) # Actual

  ggplot(cm_df, aes(x = Prediction, y = Reference, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), color = "white", size = 3) +
    labs(x = "Actual", y = "Predicted", title = "Confusion Matrix") +
    scale_fill_gradient(low = "#4575b4", high = "#d73027") +
    theme_minimal()
}

# Plot the confusion matrix
plot <- visualize_confusion_matrix(cm)
print(plot)
```



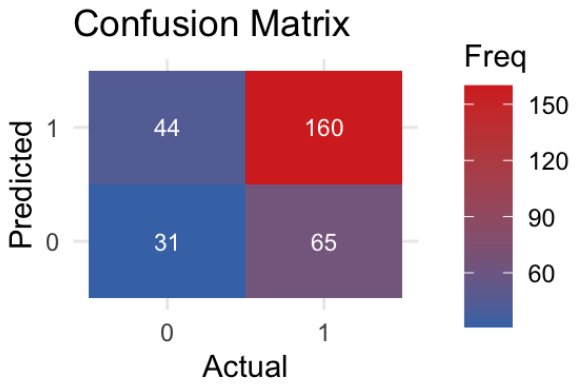

```
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  44  45
##          1  52 159
##
##                Accuracy : 0.6767
##                  95% CI : (0.6205, 0.7293)
##     No Information Rate : 0.68
##     P-Value [Acc > NIR] : 0.5764
##
##                   Kappa : 0.2424
##
##  Mcnemar's Test P-Value : 0.5424
##
##             Sensitivity : 0.4583
##             Specificity : 0.7794
##          Pos Pred Value : 0.4944
##          Neg Pred Value : 0.7536
##              Prevalence : 0.3200
##          Detection Rate : 0.1467
##    Detection Prevalence : 0.2967
##       Balanced Accuracy : 0.6189
##
##        'Positive' Class : 0
##
```

K-Nearest Neighbors (KNN)

```r
# Load the required packages
library(kknn)
library(ggplot2)
library(caret)

# Split the data into training and testing sets
set.seed(123)
trainIndex <- sample(1:nrow(data_encoded), 0.7*nrow(data_encoded))
trainData <- data_encoded[trainIndex,]
testData <- data_encoded[-trainIndex,]

# Create the KNN model
k <- 5 # Choose the number of nearest neighbors
knnModel <- kknn(Risk.good ~ ., trainData, testData, k = k)

# Make predictions on the testing set
predictions_knn <- knnModel$fitted.values

threshold <- 0.5
class_pred <- ifelse(predictions_knn > threshold, 1, 0)

actual <- testData$Risk.good

class_pred <- as.numeric(class_pred)

# Evaluate the performance of the model
class_pred_factor <- factor(class_pred, levels = c(0, 1))
actual_factor <- factor(actual, levels = c(0, 1))

# Confusion Matrix (CM)
cm <- confusionMatrix(class_pred_factor, actual_factor)
cm_knn <- cm

# Visualizing CM
visualize_confusion_matrix <- function(cm) {
  cm_table <- as.table(cm$table)

  cm_df <- as.data.frame(cm_table)
  cm_df$Prediction <- as.character(cm_df$Prediction) # Predicted
  cm_df$Reference <- as.character(cm_df$Reference) # Actual

  ggplot(cm_df, aes(x = Prediction, y = Reference, fill = Freq)) +
    geom_tile() +
    geom_text(aes(label = Freq), color = "white", size = 3) +
    labs(x = "Actual", y = "Predicted", title = "Confusion Matrix") +
    scale_fill_gradient(low = "#4575b4", high = "#d73027") +
    theme_minimal()
```

```
}

# Plot the confusion matrix
plot <- visualize_confusion_matrix(cm)
print(plot)
```




```
print(cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  31  44
##          1  65 160
##
##                Accuracy : 0.6367
##                  95% CI : (0.5794, 0.6912)
##     No Information Rate : 0.68
##     P-Value [Acc > NIR] : 0.95143
##
##                   Kappa : 0.1138
##
##  Mcnemar's Test P-Value : 0.05541
##
##             Sensitivity : 0.3229
##             Specificity : 0.7843
##          Pos Pred Value : 0.4133
##          Neg Pred Value : 0.7111
##              Prevalence : 0.3200
##          Detection Rate : 0.1033
##    Detection Prevalence : 0.2500
##       Balanced Accuracy : 0.5536
##
##        'Positive' Class : 0
##
```

Comparison of models accuarcy

```
print(cm_knn$overall['Accuracy'])
```

```
##  Accuracy
## 0.6366667
```

```
print(cm_lr$overall['Accuracy'])
```

```
##  Accuracy
## 0.6866667
```

```
print(cm_nb$overall['Accuracy'])
```

```
##  Accuracy
## 0.6766667
```

```
print(cm_rf$overall['Accuracy'])
```

```
##  Accuracy
## 0.7033333
```

```
print(cm_xgb$overall['Accuracy'])
```

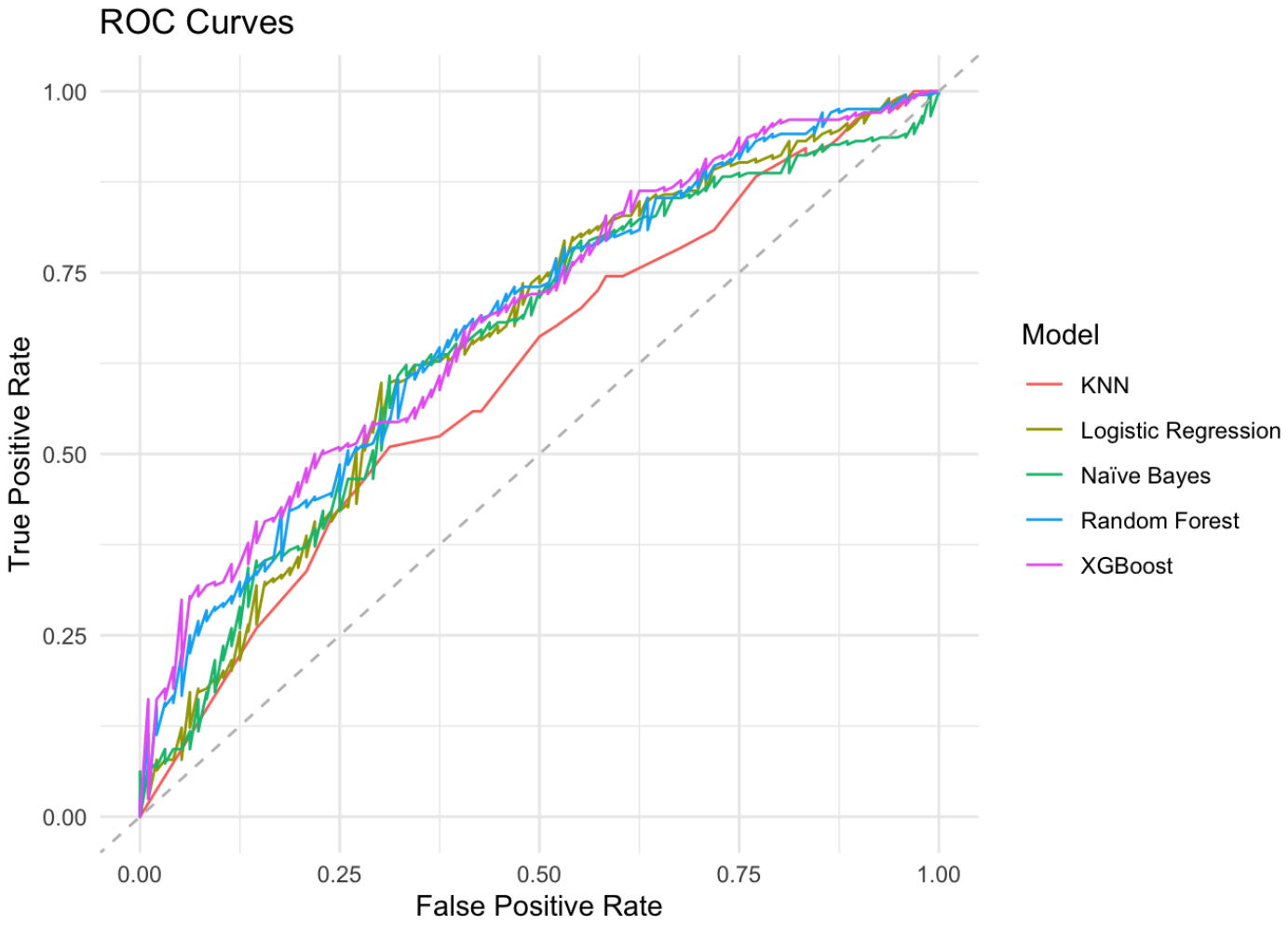
```
##  Accuracy
## 0.7033333
```

When comparing the models, the accuracy of Random Forest and XGBoost stands out as exceptional and unrivaled by the other models. Also, to select the best model, the ROC Curve can be a helpful evaluation metric.

Camparison with ROC/AUC Curve

```
library(pROC)
library(ggplot2)

# Example data
actual <- testData$Risk.good

# Example predicted probabilities from Model 1, 2, 3, 4, 5
model1_probs <- predictions_rf
model2_probs <- predictions_XGB
model3_probs <- predictions_LR
model4_probs <- predictions_knn
model5_probs <- predictions_nb_ROC[, 2]

# Calculate the ROC curve for each model
roc1 <- roc(actual, model1_probs)
roc2 <- roc(actual, model2_probs)
roc3 <- roc(actual, model3_probs)
roc4 <- roc(actual, model4_probs)
roc5 <- roc(actual, model5_probs)

# Extract the coordinates of the ROC curve
roc1_df <- data.frame(
  TPR = roc1$sensitivities,
  FPR = 1 - roc1$specificities,
  Model = "Random Forest"
)
roc2_df <- data.frame(
  TPR = roc2$sensitivities,
  FPR = 1 - roc2$specificities,
  Model = "XGBoost"
)
roc3_df <- data.frame(
  TPR = roc3$sensitivities,
  FPR = 1 - roc3$specificities,
  Model = "Logistic Regression"
)
roc4_df <- data.frame(
  TPR = roc4$sensitivities,
  FPR = 1 - roc4$specificities,
  Model = "KNN"
)
 roc5_df <- data.frame(
  TPR = roc5$sensitivities,
  FPR = 1 - roc5$specificities,
  Model = "Naïve Bayes"
)
# Combine the ROC curve data
```

```
roc_df <- rbind(roc1_df, roc2_df, roc3_df, roc4_df, roc5_df)

# Plot the ROC curves using ggplot2
ggplot(roc_df, aes(x = FPR, y = TPR, color = Model)) +
  geom_line() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "gray") +
  labs(
    x = "False Positive Rate",
    y = "True Positive Rate",
    title = "ROC Curves"
  ) +
  theme_minimal()
```



XGBoost typically outperforms other models based on the ROC curve analysis, while KNN is often the least performing model. After a thorough examination of the ROC curves, it is easier to compare their performance using the AUC metric. The AUC represents the area under the ROC curve, with a higher value indicating better predictive accuracy.

```
# Extract the AUC values for each model
auc1 <- auc(roc1)
auc2 <- auc(roc2)
auc3 <- auc(roc3)
auc4 <- auc(roc4)
auc5 <- auc(roc5)

# Print the AUC values
cat("RF - AUC: ", auc1, "\n")
```

```
## RF - AUC:  0.6796364
```

```
cat("XGB - AUC:", auc2, "\n")
```

```
## XGB - AUC: 0.6893382
```

```
cat("LR - AUC: ", auc3, "\n")
```

```
## LR - AUC:  0.6599265
```

```
cat("KNN - AUC:", auc4, "\n")
```

```
## KNN - AUC: 0.6115196
```

```
cat("NB - AUC: ", auc5, "\n")
```

```
## NB - AUC:  0.652165
```

Based on the evaluation of the different machine learning models, it has been determined that the XGBoost model performs the best in predicting credit risk.