# Advanced Analytics for Predicting Heart Disease Using Machine Learning

DA5030

Sowmya Muthyala

FALL 2023

## Introduction

In my DA5030 project, I delved into heart disease prediction using various machine learning models. I first explored and preprocessed the dataset, detecting outliers and handling missing values. Investigating correlations and distribution patterns shed light on the data's nuances. Leveraging SVM, Naive Bayes, Logistic Regression, Decision Trees, and Random Forest models, I achieved impressive predictive accuracy (up to 86.44%). Each model demonstrated strengths in precision, recall, and AUC, indicating their ability to distinguish between disease-positive and disease-negative cases. The ensemble model further improved predictions, showcasing an accuracy of 88.14% and robust performance across key metrics, solidifying its potential in heart disease prognosis.

—— Data Acquisition ——-

```
# Loading necessary libraries
library(readr) # Used for reading and writing data
library(caret) # Contains functions for training and plotting machine learning models
library(ggplot2) # A system for creating graphics
library(e1071) # Contains functions for statistical learning
library(randomForest) # For fitting random forest models
library(lightgbm) # For LightGBM models, a gradient boosting framework
library(corrplot) # For visualizing correlation matrices
library(pROC)
library(rpart)
library(kernlab)
```

```
# Reading the dataset from the URL from where the data will be fetched
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data

df <- read.csv(url, header = FALSE, na.strings = "?")

# Setting column names for the dataframe
colnames(df) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach", "exang", "oldpeak
```

—— Data Exploration ——

```
# The 'head' function displays the first six rows of the dataframe for a quick overview of the data.
head(df)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 1  63   1  1      145  233   1       2     150     0     2.3     3  0    6
## 2  67   1  4      160  286   0       2     108     1     1.5     2  3    3
## 3  67   1  4      120  229   0       2     129     1     2.6     2  2    7
## 4  37   1  3      130  250   0       0     187     0     3.5     3  0    3
## 5  41   0  2      130  204   0       2     172     0     1.4     1  0    3
## 6  56   1  2      120  236   0       0     178     0     0.8     1  0    3
##   condition
## 1         0
## 2         2
## 3         1
## 4         0
## 5         0
## 6         0
```

```r
# The 'summary' function provides a statistical summary of the dataframe.
summary(df)
```

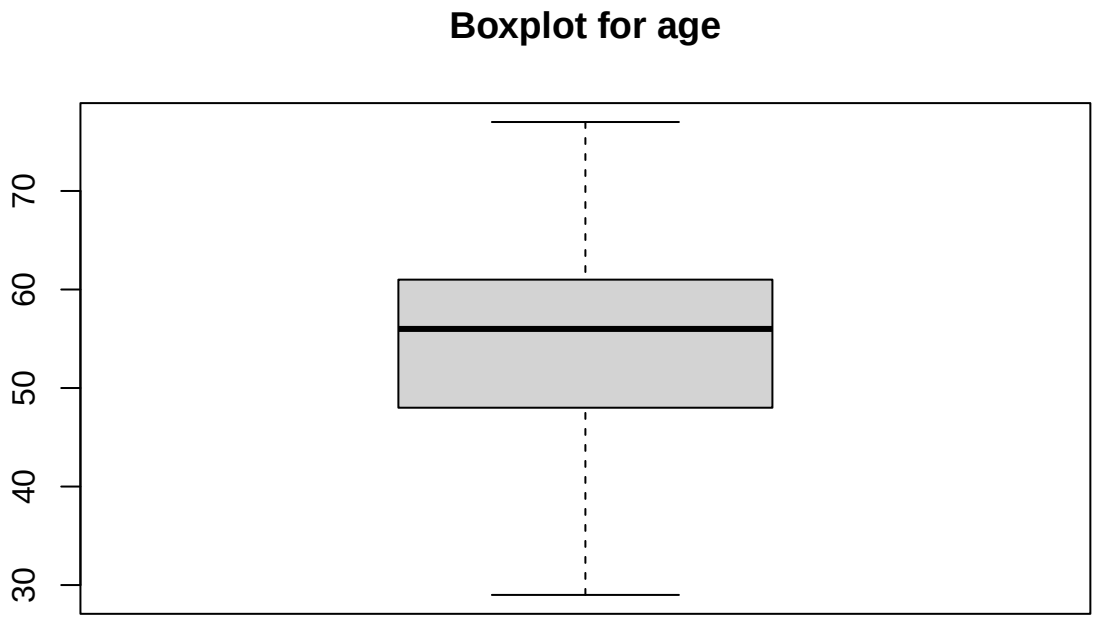
```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :54.44   Mean   :0.6799   Mean   :3.158   Mean   :131.7
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
##
##       chol            fbs            restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :241.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.7   Mean   :0.1485   Mean   :0.9901   Mean   :149.6
##  3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##
##      exang           oldpeak         slope             ca
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.80   Median :2.000   Median :0.0000
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601   Mean   :0.6722
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000   Max.   :3.0000
##                                                  NA's   :4
##       thal         condition
##  Min.   :3.000   Min.   :0.0000
##  1st Qu.:3.000   1st Qu.:0.0000
##  Median :3.000   Median :0.0000
##  Mean   :4.734   Mean   :0.9373
##  3rd Qu.:7.000   3rd Qu.:2.0000
##  Max.   :7.000   Max.   :4.0000
##  NA's   :2
```

```r
# The 'str' function displays the structure of the dataframe.
str(df)
```

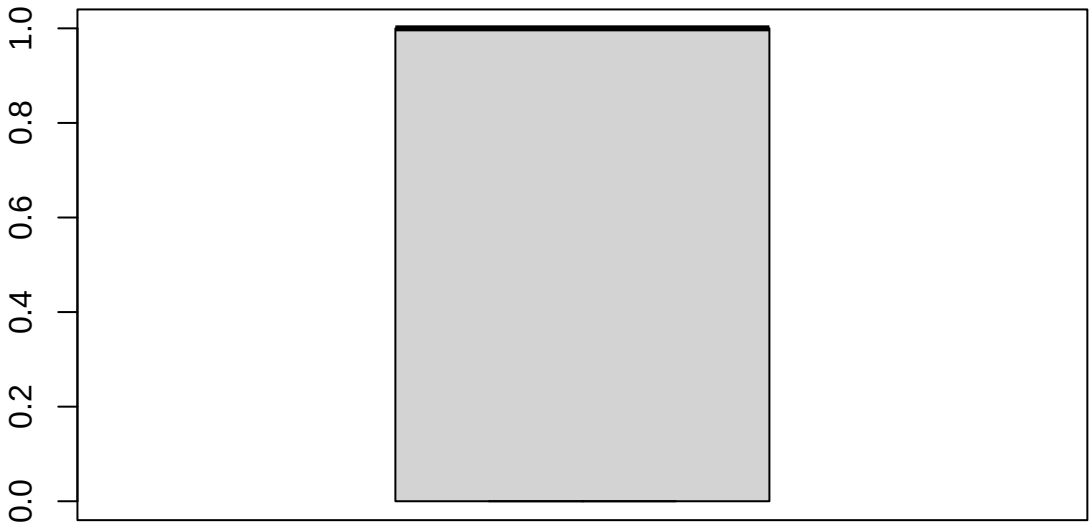
```
## 'data.frame':    303 obs. of  14 variables:
##  $ age      : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex      : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp       : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps : num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol     : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs      : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg  : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach  : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang    : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak  : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope    : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca       : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal     : num  6 3 7 3 3 3 3 3 7 7 ...
##  $ condition: int  0 2 1 0 0 0 3 0 2 1 ...
```

**Exploratory Data plots**

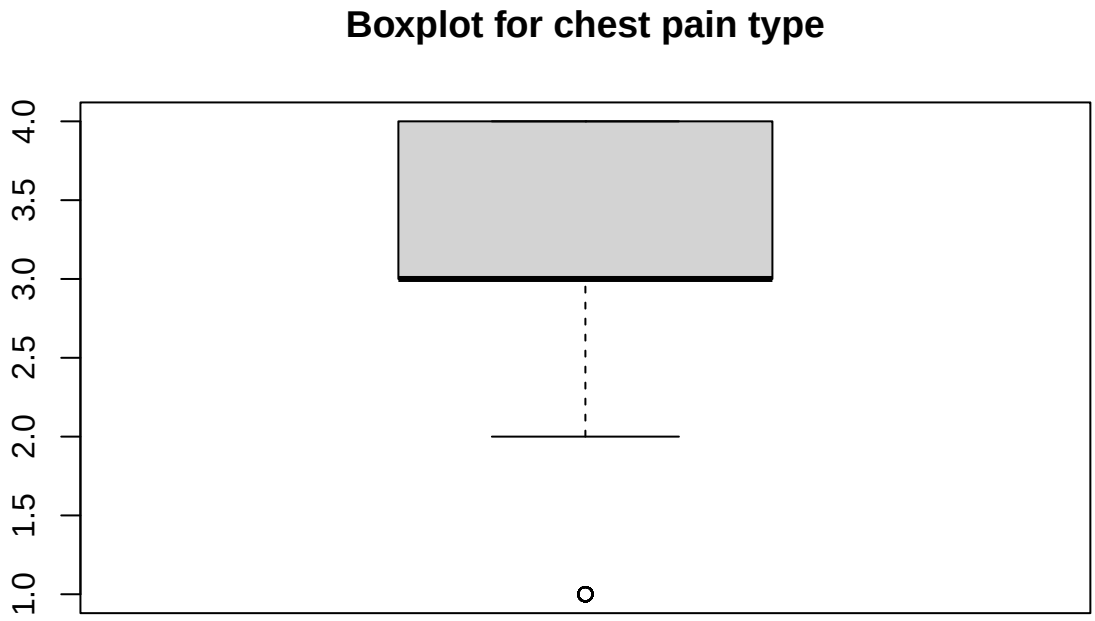
```
# Identifying outliers through boxplots for each variable
boxplot(df$age, main = "Boxplot for age")
```



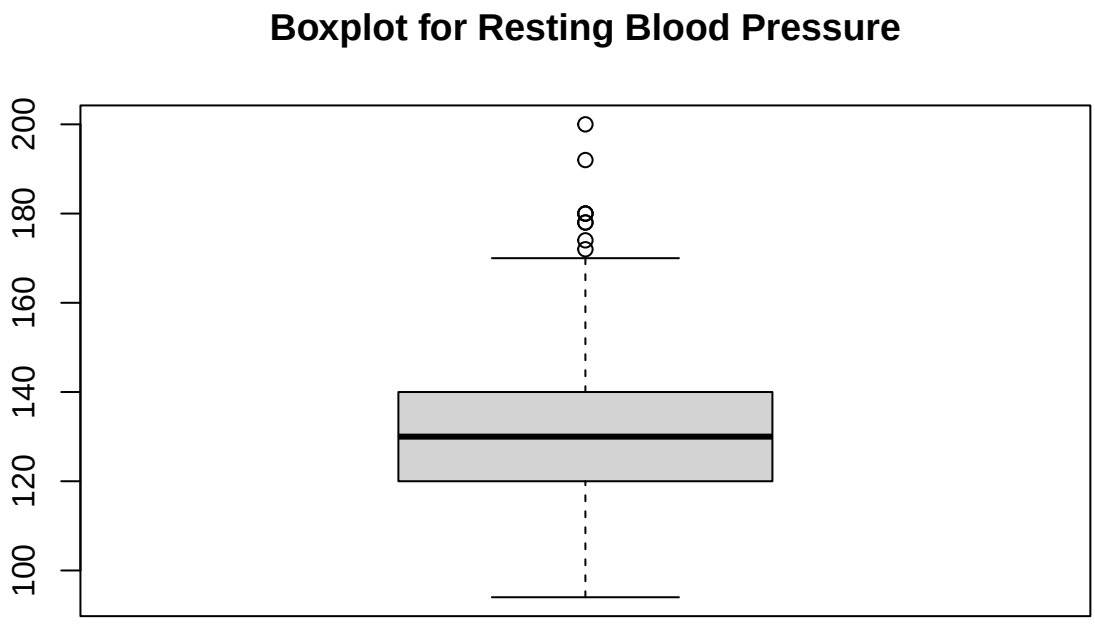

```
boxplot(df$sex, main = "Boxplot for sex")
```



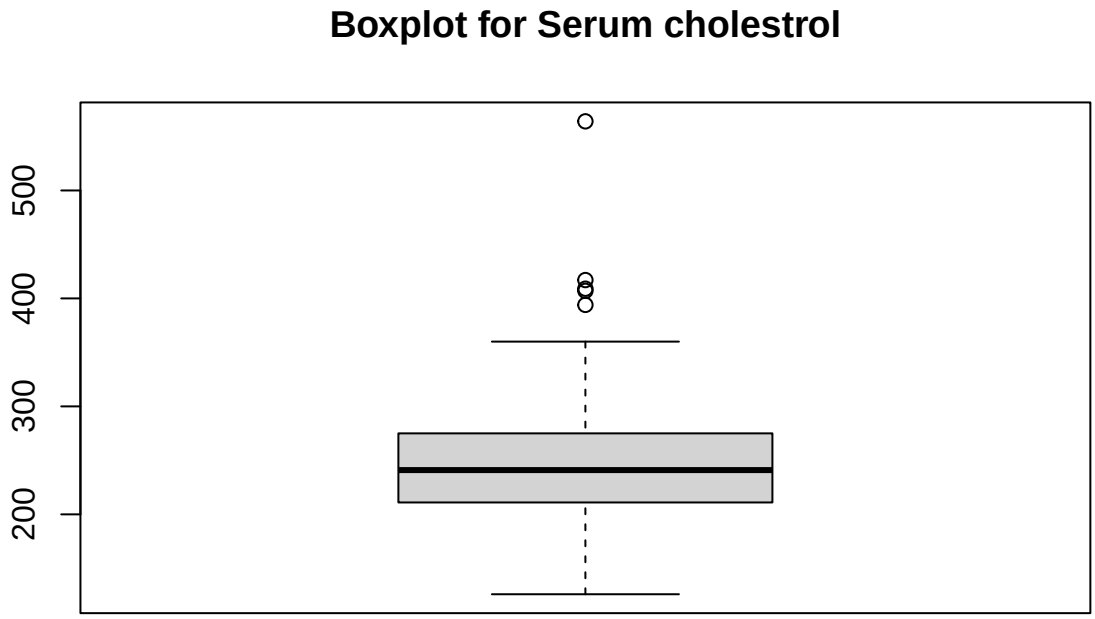

```r
boxplot(df$cp, main = "Boxplot for chest pain type")
```



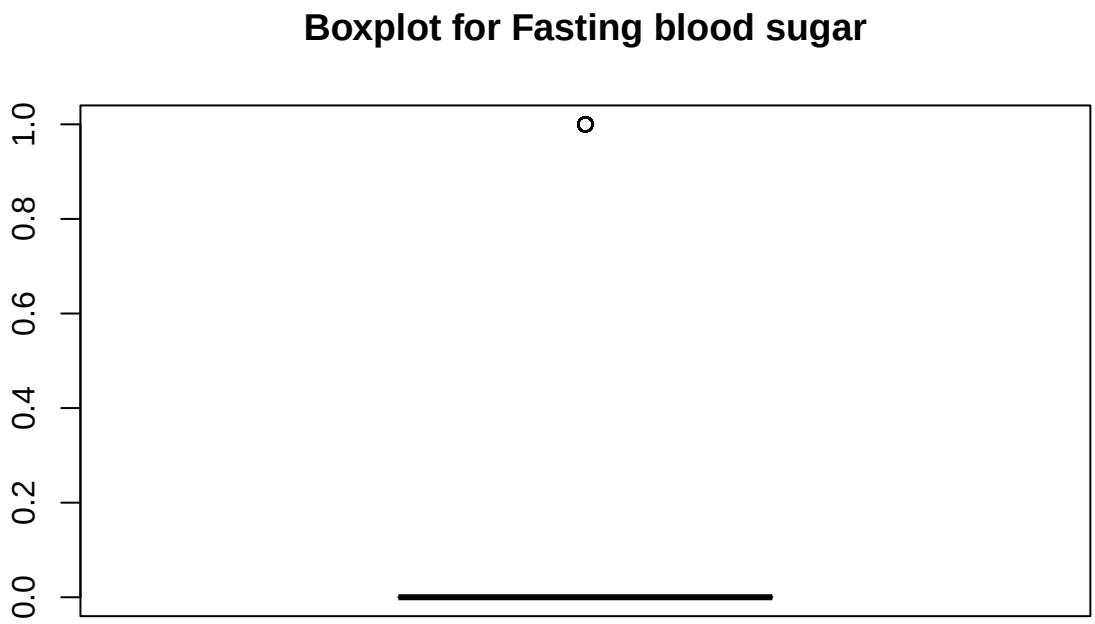

```r
boxplot(df$trestbps, main = "Boxplot for Resting Blood Pressure")
```



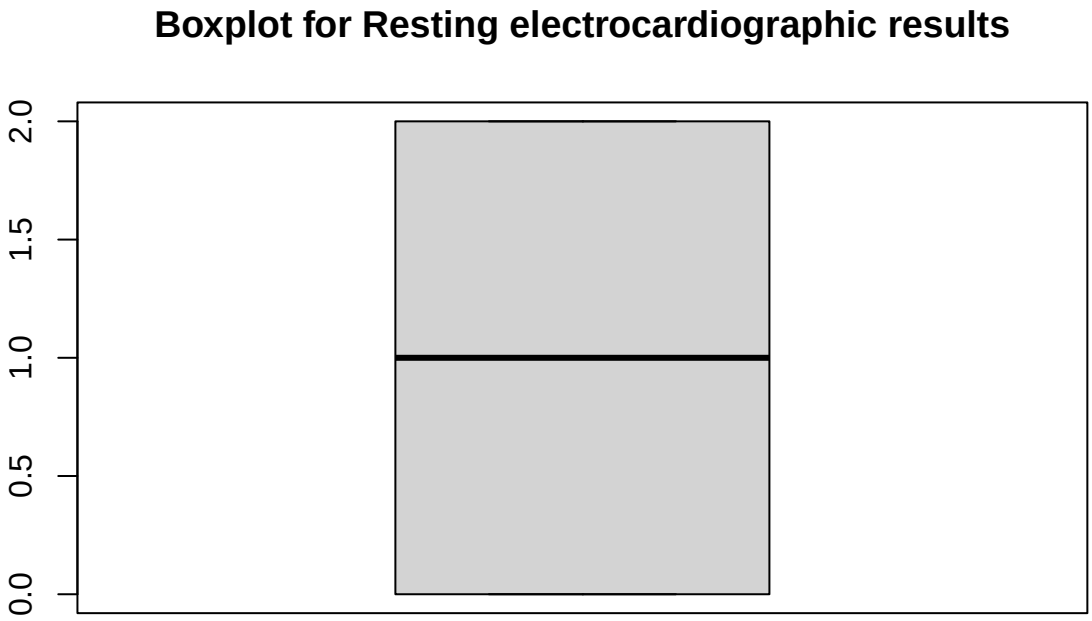

```r
boxplot(df$chol, main = "Boxplot for Serum cholestrol")
```



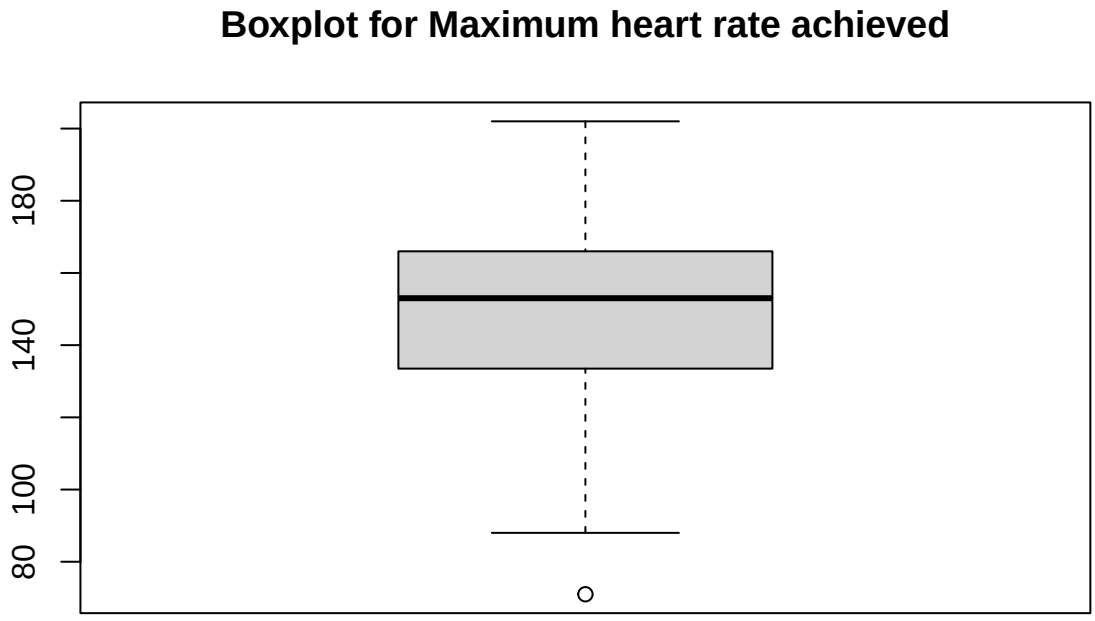

```r
boxplot(df$fbs, main = "Boxplot for Fasting blood sugar")
```



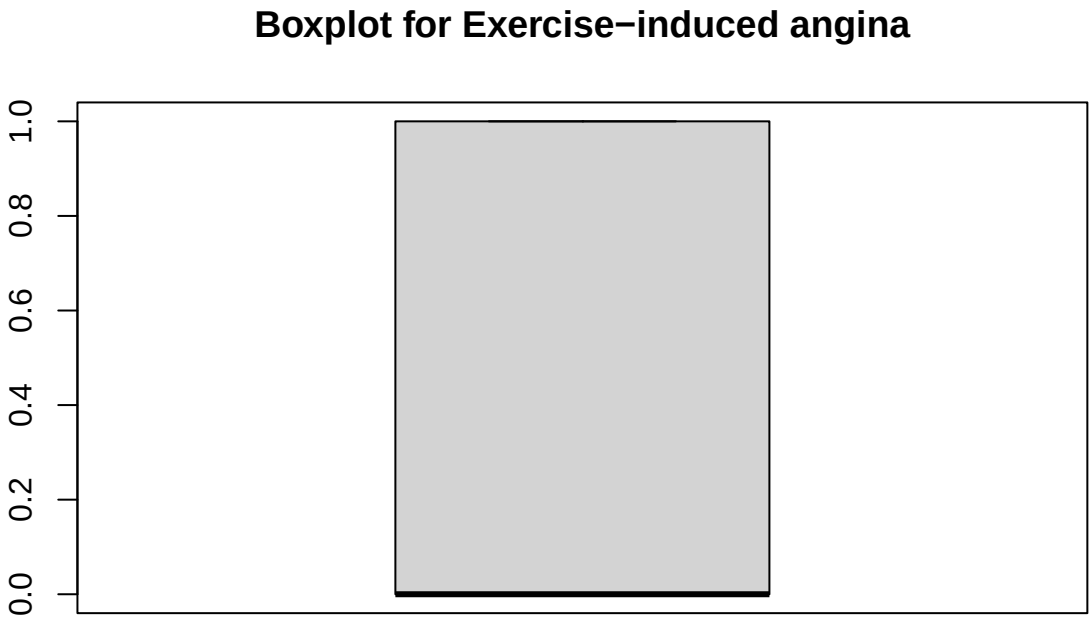

```
boxplot(df$restecg, main = "Boxplot for Resting electrocardiographic results")
```



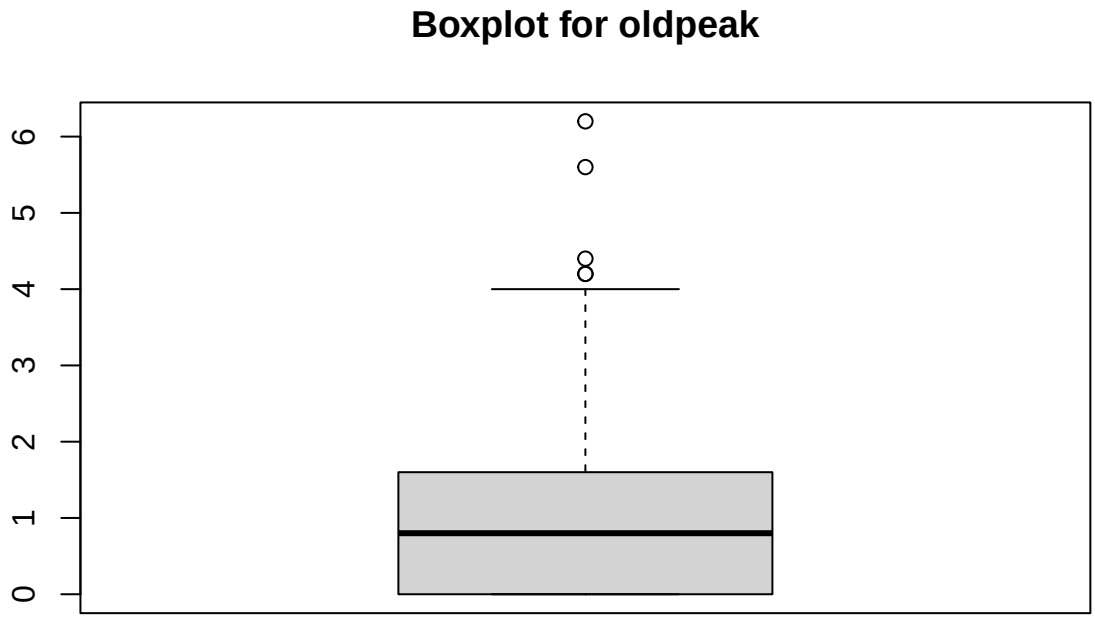

```r
boxplot(df$thalach, main = "Boxplot for Maximum heart rate achieved")
```



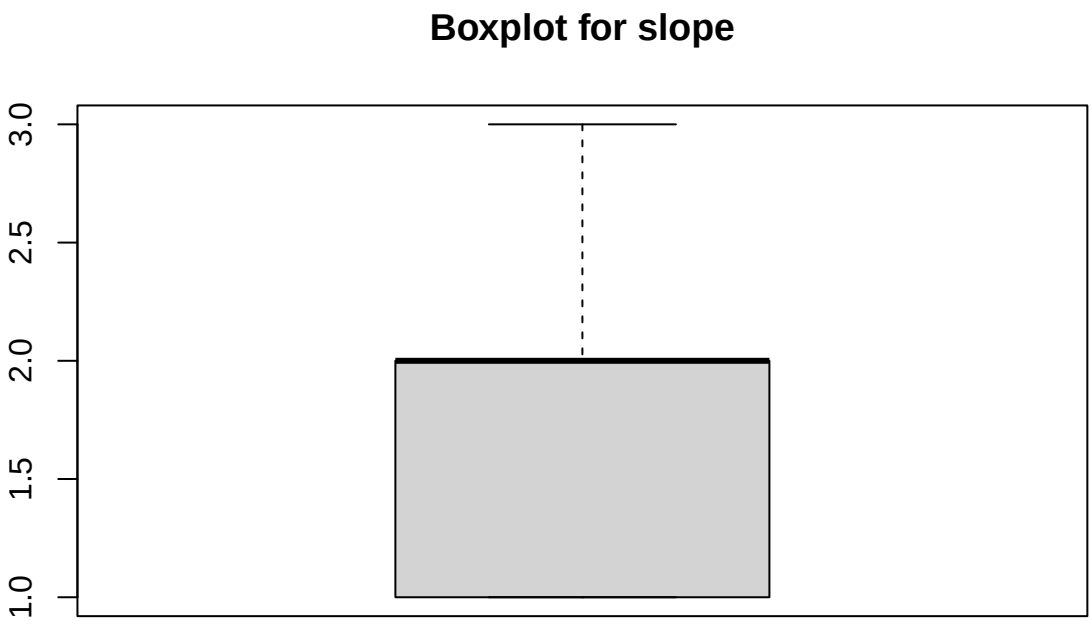

```r
boxplot(df$exang, main = "Boxplot for Exercise-induced angina")
```



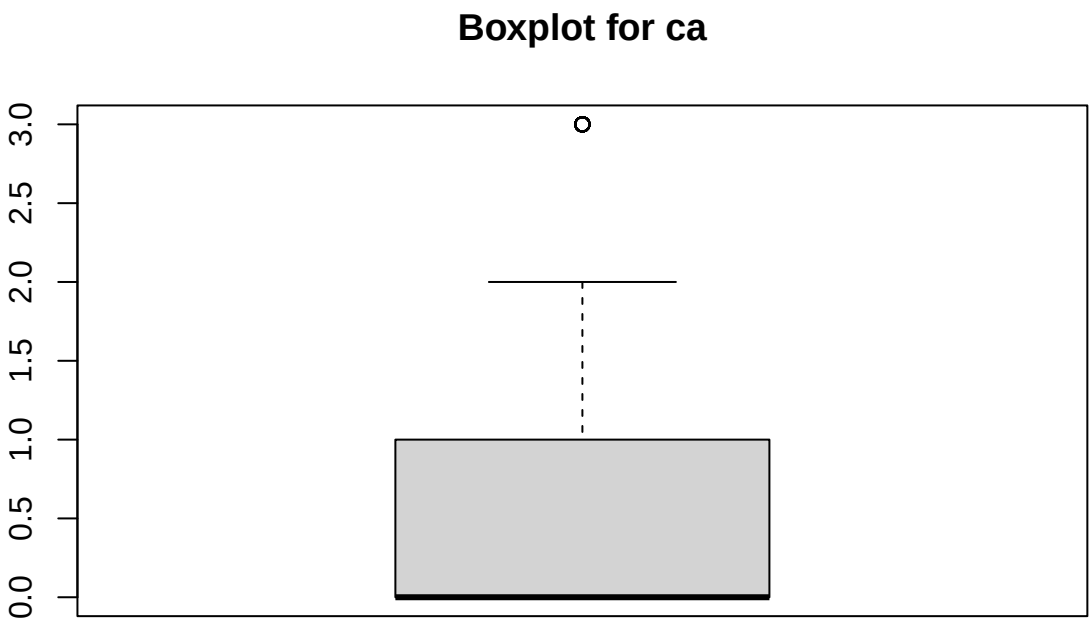

```
boxplot(df$oldpeak, main = "Boxplot for oldpeak")
```



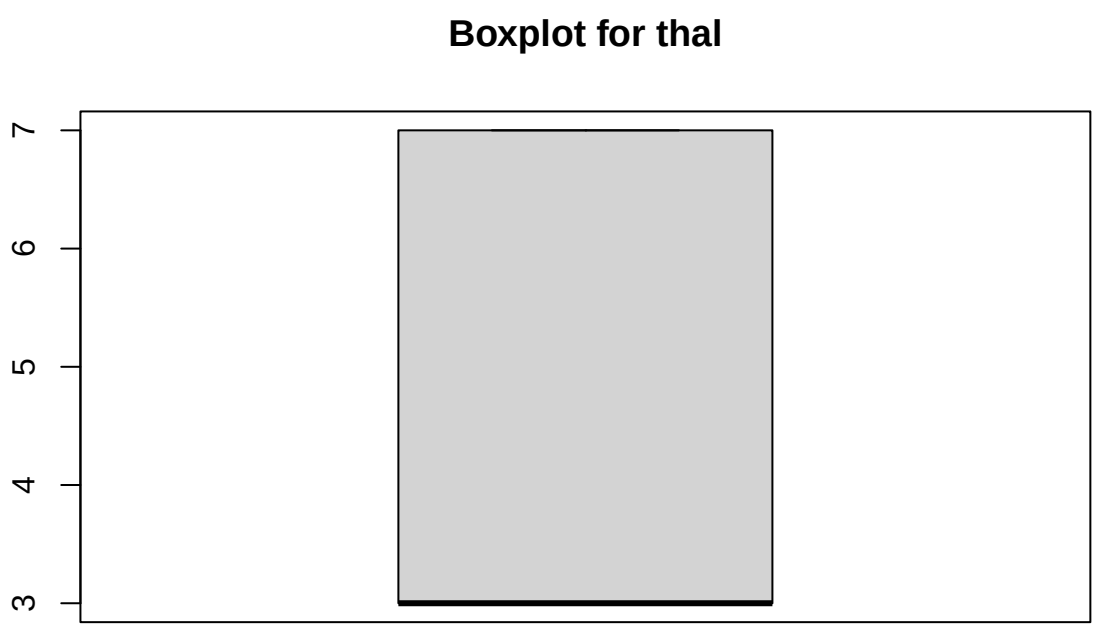

```r
boxplot(df$slope, main = "Boxplot for slope")
```



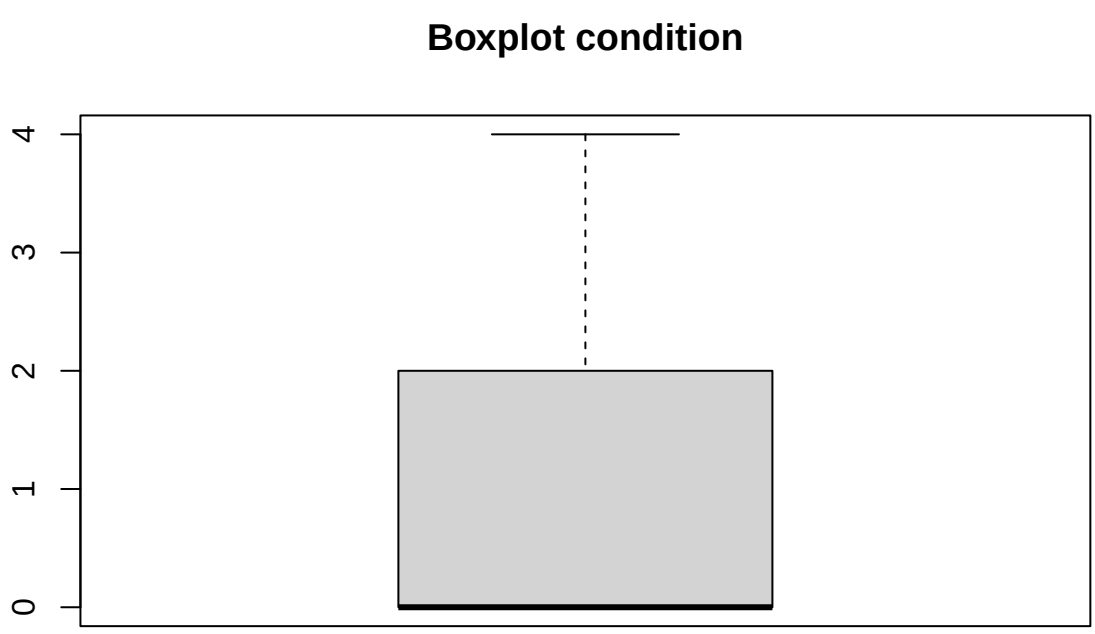

```r
boxplot(df$ca, main = "Boxplot for ca")
```




```
boxplot(df$thal, main = "Boxplot for thal")
```



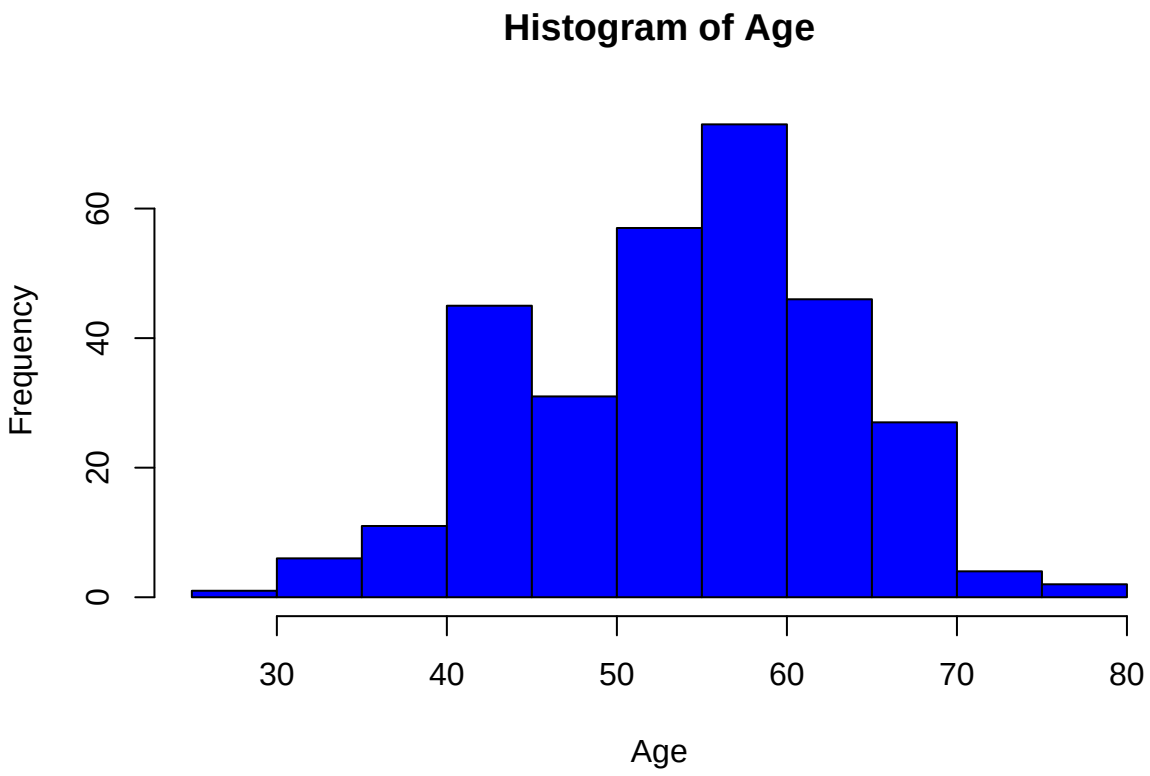

```
boxplot(df$condition, main = "Boxplot condition")
```



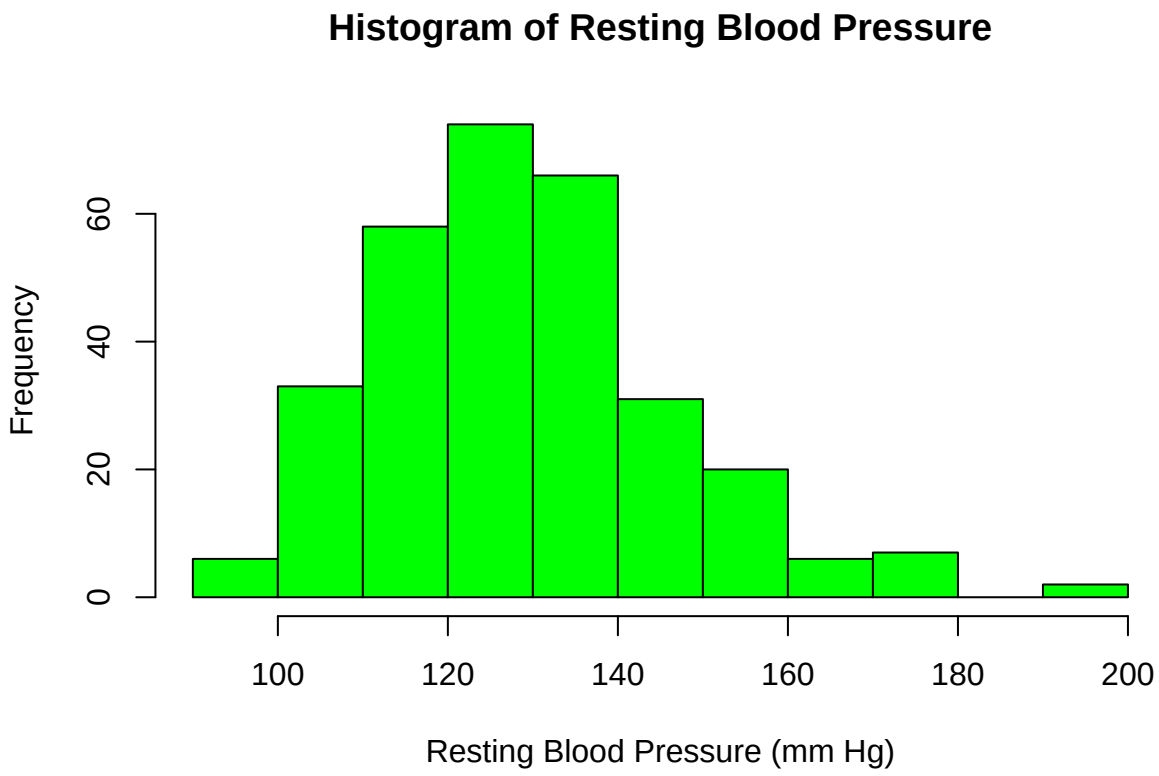

Upon conducting boxplot analysis, I noticed outliers in several key variables within our dataset. Specifically, I found outliers in Chest Pain Type (CPT), Resting Blood Pressure (Trestbps), Serum Cholesterol (Chol), Fasting Blood Sugar (FBS), Maximum Heart Rate Achieved (Thalach), ST Depression Induced by Exercise Relative to Rest (Oldpeak), and the count of Major Vessels Colored by Fluoroscopy (CA). These outliers indicate significant deviations of certain data points from the typical distribution. Such deviations might stem from various reasons, like measurement or data entry errors, or they could represent genuine variations within the patient population.

further in the analysis I will normalize the data to better fit the model and deal with outliers this way.

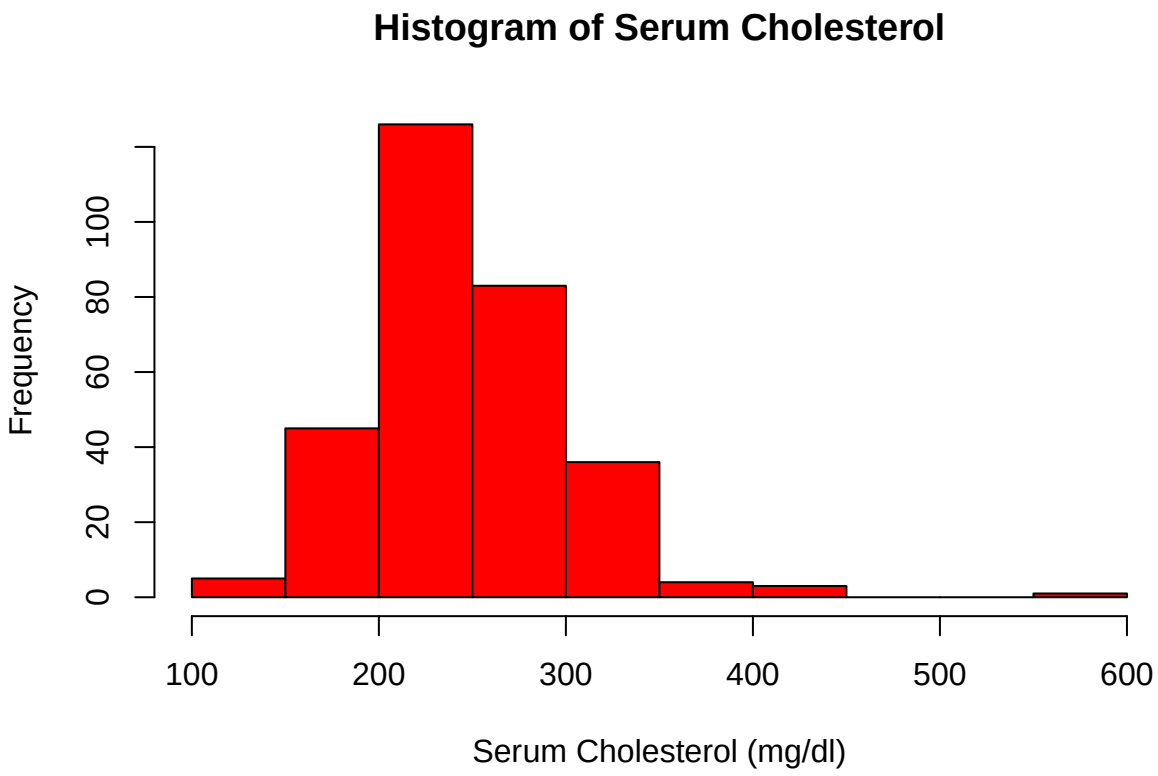
```
# Histograms for continuous variables
# Histogram for 'age'
hist(df$age,
     main = "Histogram of Age",
     xlab = "Age",
     ylab = "Frequency",
     col = "blue",
     border = "black")
```



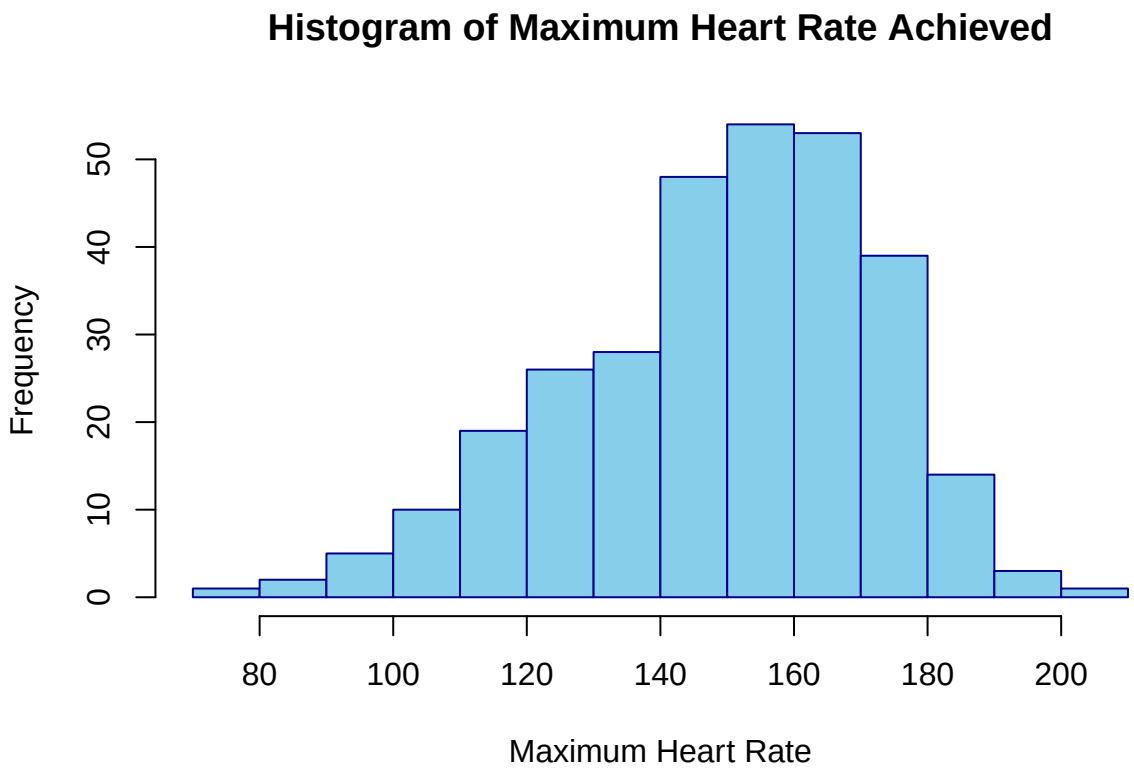

```r
# Histogram for 'trestbps' (Resting Blood Pressure)
hist(df$trestbps,
     main = "Histogram of Resting Blood Pressure",
     xlab = "Resting Blood Pressure (mm Hg)",
     ylab = "Frequency",
     col = "green",
     border = "black")
```



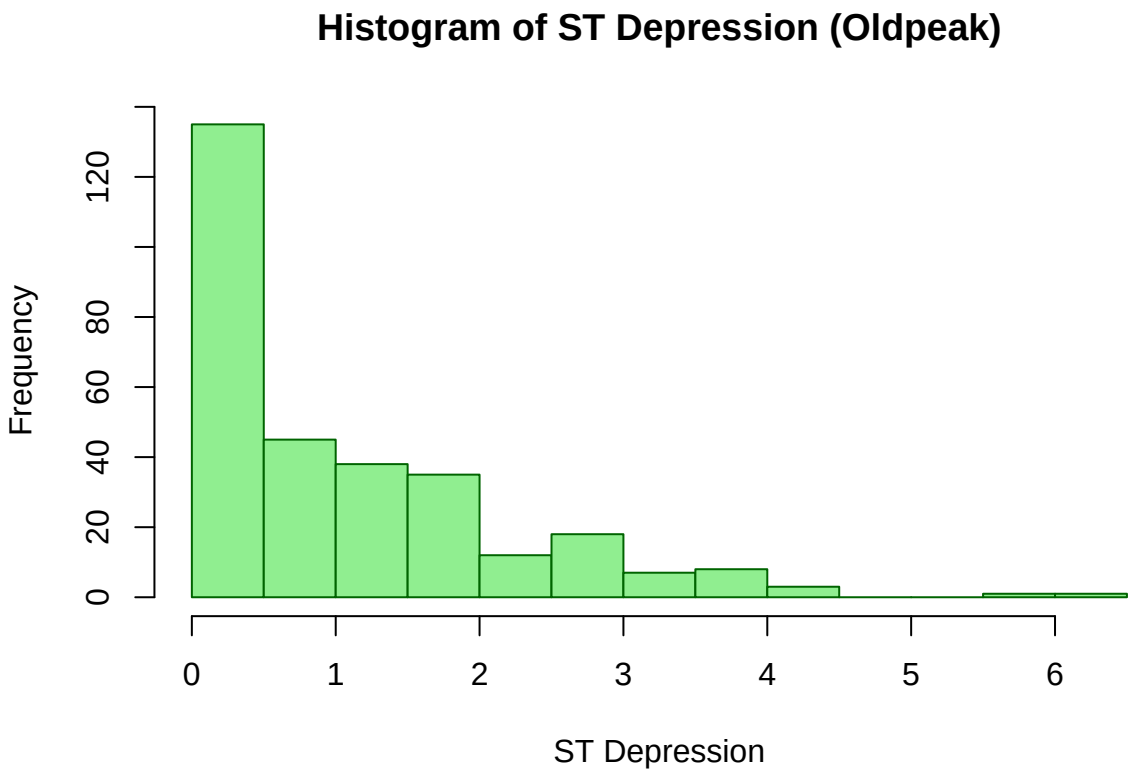

```r
# Histogram for 'chol' (Serum Cholesterol)
hist(df$chol,
     main = "Histogram of Serum Cholesterol",
     xlab = "Serum Cholesterol (mg/dl)",
     ylab = "Frequency",
     col = "red",
     border = "black")
```




```r
# Histogram for 'thalach' (Maximum Heart Rate Achieved)
hist(df$thalach,
     main = "Histogram of Maximum Heart Rate Achieved",
     xlab = "Maximum Heart Rate",
     ylab = "Frequency",
     col = "skyblue",
     border = "darkblue")
```

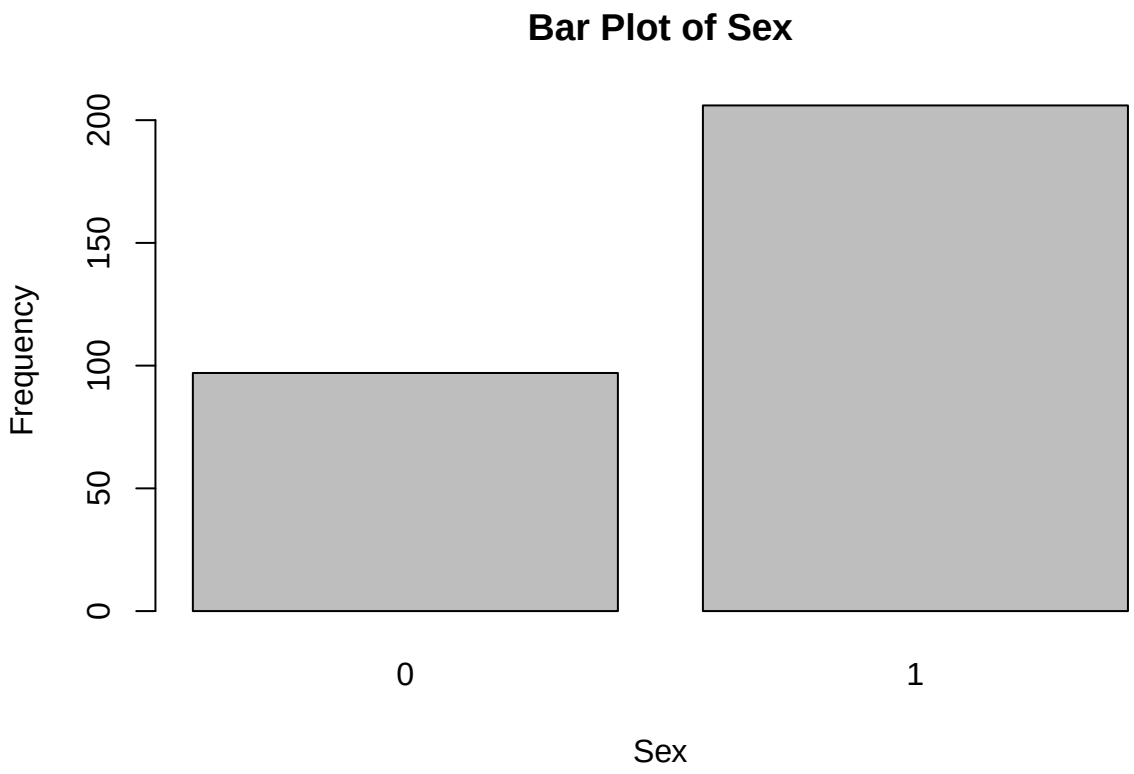



```r
# Histogram for 'oldpeak' (ST Depression)
hist(df$oldpeak,
     main = "Histogram of ST Depression (Oldpeak)",
     xlab = "ST Depression",
     ylab = "Frequency",
     col = "lightgreen",
     border = "darkgreen")
```

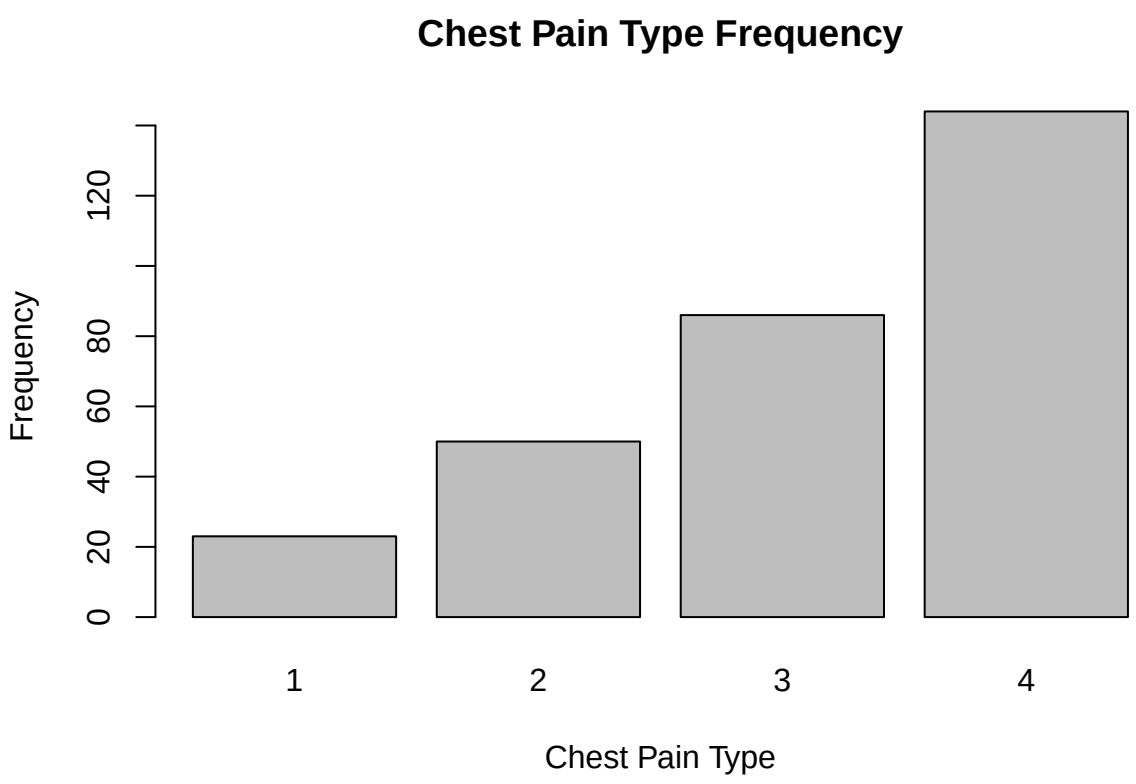



The analysis of the histograms reveals distinctive distributions within the dataset: a concentration of middle-aged subjects, typical resting blood pressure readings, elevated cholesterol levels in some individuals, higher heart rates primarily in stress scenarios, and potentially significant ST depression levels in a few subjects, suggesting varied physiological and potential medical implications within the data.

```
# Bar plots for categorical variables
# Bar plot for the 'sex' variable
barplot(table(df$sex), main = "Bar Plot of Sex", xlab = "Sex", ylab = "Frequency")
```

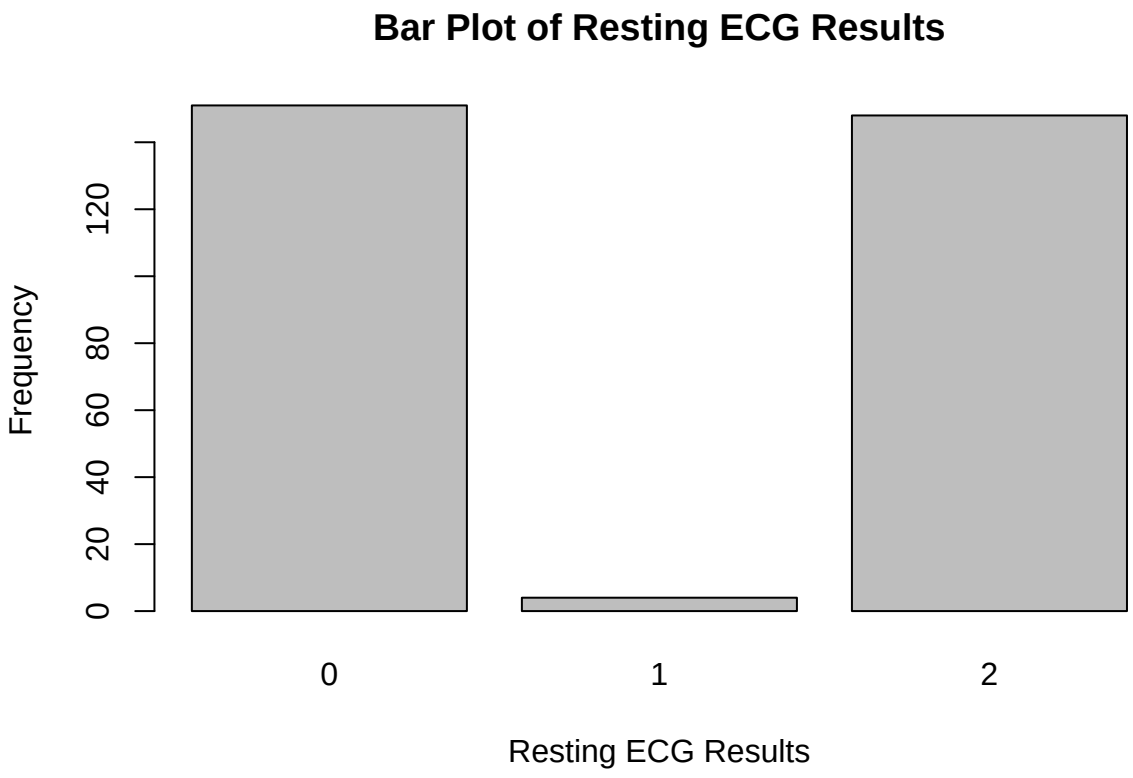



```r
# Bar plot for the 'cp' (Chest Pain Type) variable
barplot(table(df$cp), main = "Chest Pain Type Frequency", xlab = "Chest Pain Type", ylab = "Frequency")
```

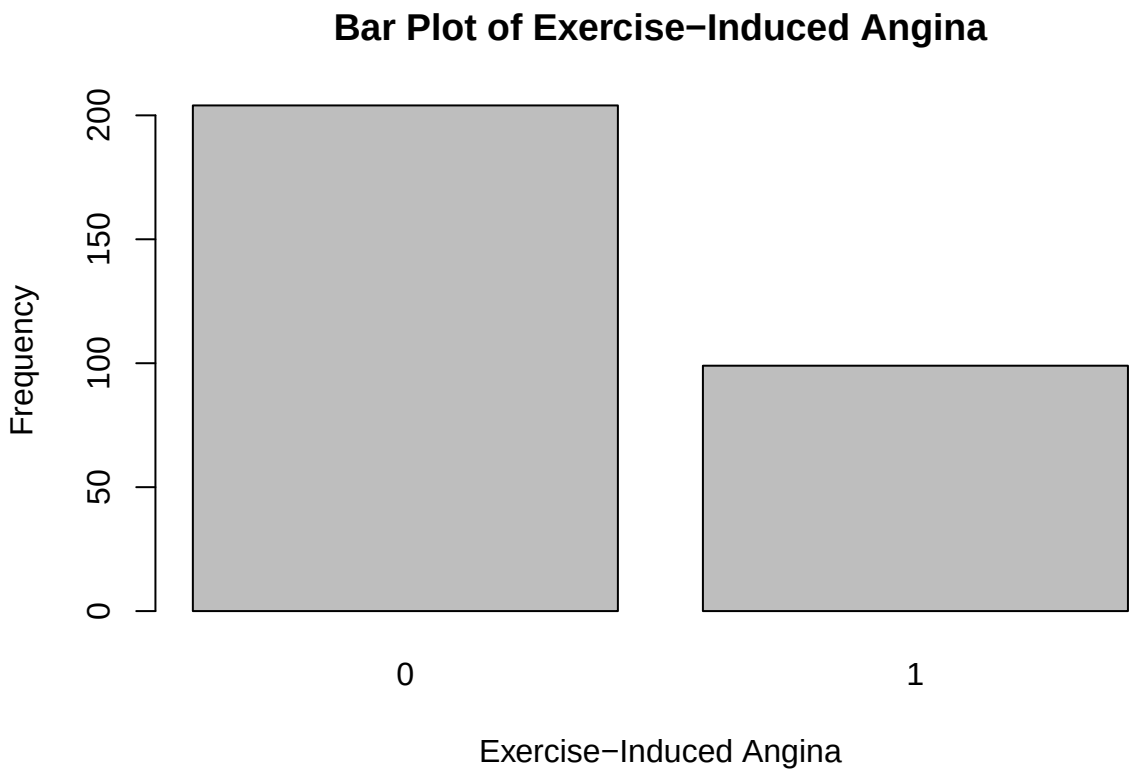



```r
# Bar plot for 'fbs' (Fasting Blood Sugar > 120 mg/dl)
barplot(table(df$fbs), main = "Bar Plot of Fasting Blood Sugar > 120 mg/dl", xlab = "Fasting Blood Sugar
```

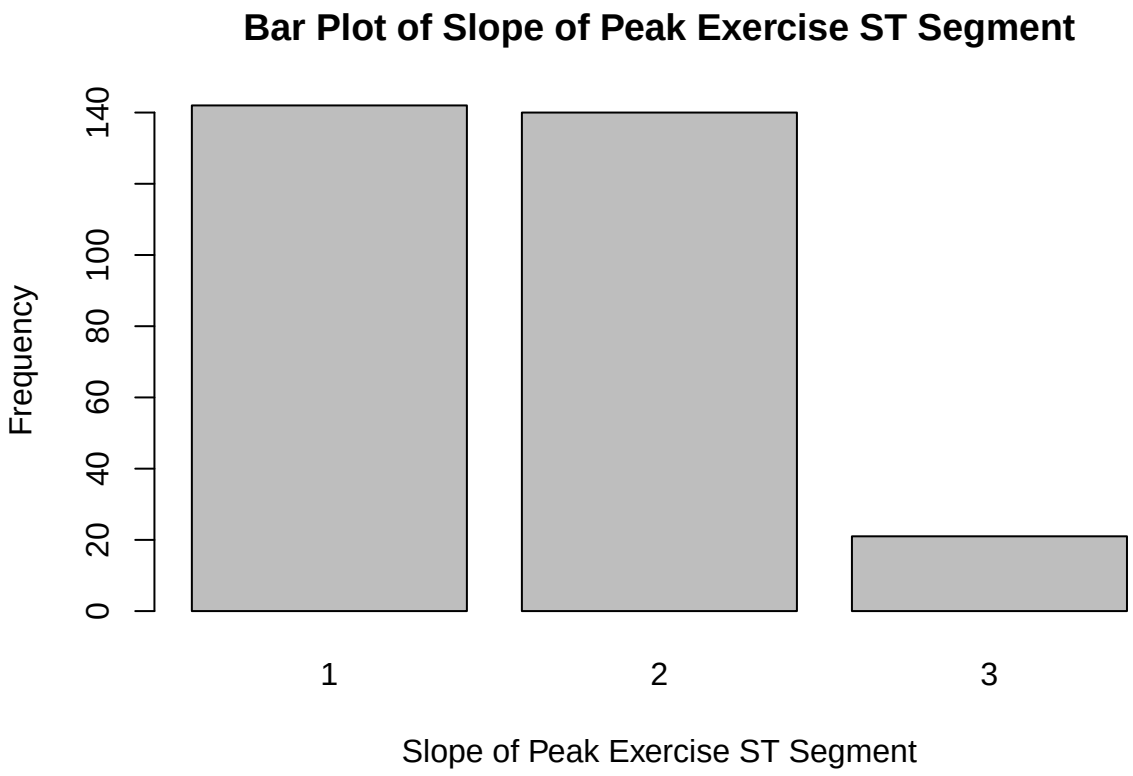

```r
# Bar plot for 'restecg' (Resting ECG Results)
barplot(table(df$restecg), main = "Bar Plot of Resting ECG Results", xlab = "Resting ECG Results", ylab
```



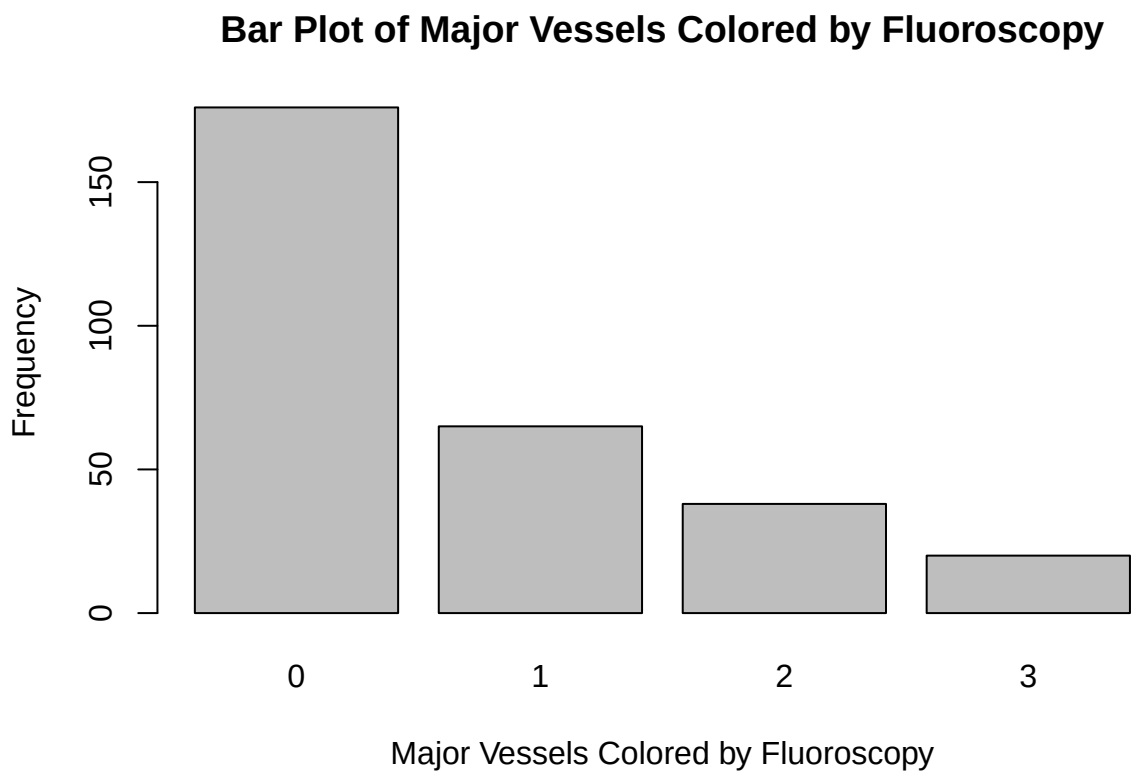

```r
# Bar plot for 'exang' (Exercise-Induced Angina)
barplot(table(df$exang), main = "Bar Plot of Exercise-Induced Angina", xlab = "Exercise-Induced Angina"
```



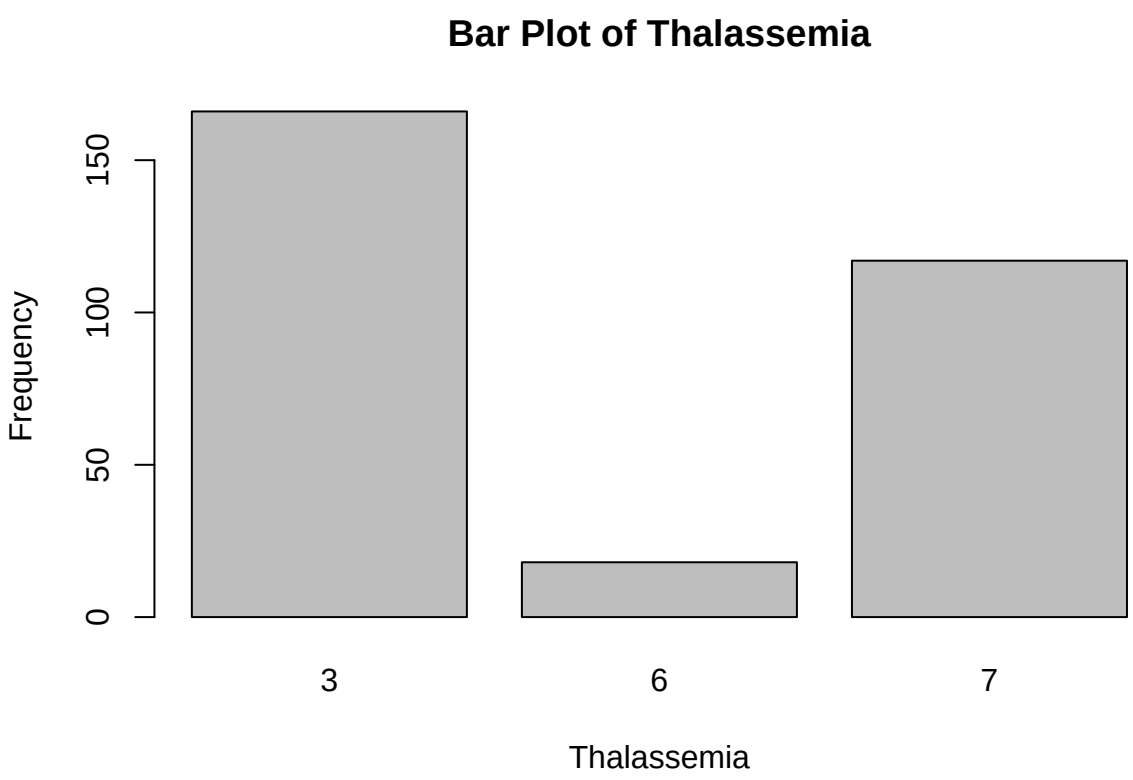

```r
# Bar plot for 'slope' (Slope of Peak Exercise ST Segment)
barplot(table(df$slope), main = "Bar Plot of Slope of Peak Exercise ST Segment", xlab = "Slope of Peak
```




```r
# Bar plot for 'ca' (Major Vessels Colored by Fluoroscopy)
barplot(table(df$ca), main = "Bar Plot of Major Vessels Colored by Fluoroscopy", xlab = "Major Vessels
```




```r
# Bar plot for 'thal' (Thalassemia)
barplot(table(df$thal), main = "Bar Plot of Thalassemia", xlab = "Thalassemia", ylab = "Frequency")
```




**The Welch Two Sample t-test**

is a statistical method used to compare the means of two independent groups when the assumptions of equal variance and normality might not hold.

```
# T-test (comparing cholesterol levels between sexes)
t.test(chol ~ sex, data = df)
```

```
##
##  Welch Two Sample t-test
##
## data:  chol by sex
## t = 3.0643, df = 136.37, p-value = 0.002631
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##   7.855795 36.445477
## sample estimates:
## mean in group 0 mean in group 1
##        261.7526        239.6019
```

```
# T-test for 'age' and condition
t.test(df$age ~ df$condition > 0)
```

```
##
```

```
##  Welch Two Sample t-test
##
## data:  df$age by df$condition > 0
## t = -4.0303, df = 300.93, p-value = 7.061e-05
## alternative hypothesis: true difference in means between group FALSE and group TRUE is not equal to 0
## 95 percent confidence interval:
##  -6.013385 -2.067682
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            52.58537            56.62590
```

```r
# T-test for 'thalach' (max heart rate) and condition
t.test(df$thalach ~ df$condition > 0)
```

```
##
##  Welch Two Sample t-test
##
## data:  df$thalach by df$condition > 0
## t = 7.8579, df = 272.27, p-value = 9.106e-14
## alternative hypothesis: true difference in means between group FALSE and group TRUE is not equal to 0
## 95 percent confidence interval:
##  14.32900 23.90912
## sample estimates:
## mean in group FALSE  mean in group TRUE
##             158.378             139.259
```

In all three cases, the p-values are very low (well below conventional significance levels like 0.05), indicating strong evidence against the null hypothesis of no difference in means between the compared groups. The confidence intervals further support these findings, showing a substantial difference between the means of the groups being compared.

### Identifying and Imputation of Missing values

```r
# Checking for missing values in the dataset
sapply(df, function(x) sum(is.na(x)))
```

```
##       age       sex        cp  trestbps      chol       fbs   restecg   thalach
##         0         0         0         0         0         0         0         0
##     exang   oldpeak     slope        ca      thal condition
##         0         0         0         4         2         0
```

```r
# Imputing missing values in the 'thal' column
df$thal[is.na(df$thal)] <- mean(df$thal, na.rm = TRUE)

# Imputing missing values in the 'ca' column
df$ca[is.na(df$ca)] <- mean(df$ca, na.rm = TRUE)
```

There were a total of six missing values detected, with four missing entries in the 'thal' column and two in the 'ca' (number of major vessels colored by fluoroscopy) column.

### Encoding

process of converting data from one format or structure into another.

```
# encoding sex column
df$sex <- as.integer(factor(df$sex))
# Chest Pain Type (cp): This is a categorical variable with values ranging from 1 to 4. As these catego
df$cp <- as.factor(df$cp)
df$restecg <- as.factor(df$restecg)
df$slope <- as.factor(df$slope)
df$thal <- as.factor(df$thal)


# Fasting Blood Sugar > 120 mg/dl (fbs): This is a binary categorical variable (yes/no or 1/0). It can
df$fbs <- as.integer(df$fbs)
df$exang <- as.integer(df$exang)


# Encoding the target variable 'condition'
df$condition <- ifelse(df$condition== 0, 0, 1)

# Display the structure of the dataframe after encoding
str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age      : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex      : int  2 2 2 2 1 2 1 1 2 2 ...
##  $ cp       : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps : num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol     : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs      : int  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg  : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach  : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang    : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak  : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope    : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca       : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal     : Factor w/ 4 levels "3","4.73421926910299",..: 3 1 4 1 1 1 1 1 4 4 ...
##  $ condition: num  0 1 1 0 0 0 1 0 1 1 ...
```

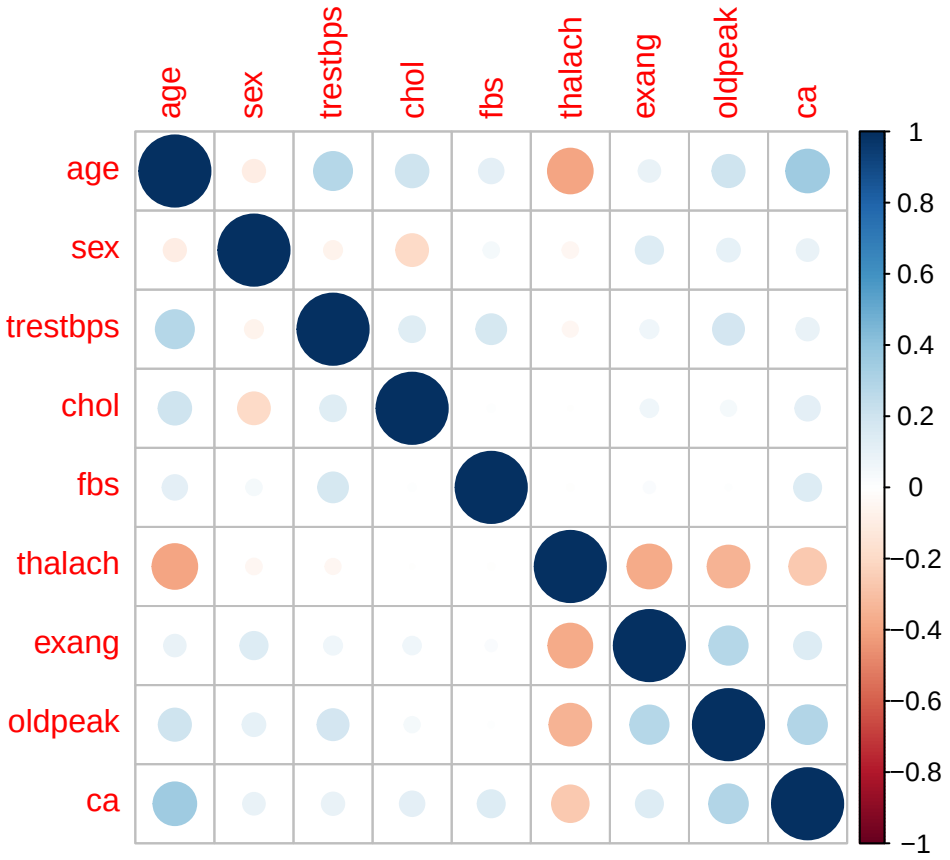
### correlation for numeric variables

```
# Converting 'condition' into a factor
df$condition <- factor(df$condition)

# Display the structure of the dataframe after converting to factor
str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age      : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex      : int  2 2 2 2 1 2 1 1 2 2 ...
```

```
##  $ cp       : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps : num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol     : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs      : int  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg  : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach  : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang    : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak  : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope    : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca       : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal     : Factor w/ 4 levels "3","4.73421926910299",..: 3 1 4 1 1 1 1 1 4 4 ...
##  $ condition: Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
```

```r
# Correlation matrix
correlationMatrix <- cor(df[, sapply(df, is.numeric)]) # Only for numeric variables

# Plotting correlation matrix
library(corrplot)
corrplot(correlationMatrix, method = "circle")
```




Positive correlations can be seen between variables such as age and trestbps (resting blood pressure), suggesting that blood pressure tends to be higher with increasing age. Negative correlations may also be present, indicated by the red circles.

—— Data Cleaning and Shaping ——

```r
sum(is.na(df))
```

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

```r
#function to normalzie numeric columns
normalize <- function(x) {
  if(is.numeric(x)) { # Check if the column is numeric
    return ((x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)))
  } else {
    return (x) # Return the column unchanged if it's not numeric
  }
}
# Applying the normalization function to the entire dataframe
df <- as.data.frame(lapply(df, normalize))


# Function to identify outliers
identify_outliers <- function(data) {
  for(col in names(data)) {
    if(is.numeric(data[[col]])) {
      Q1 <- quantile(data[[col]], 0.25)
      Q3 <- quantile(data[[col]], 0.75)
      IQR <- Q3 - Q1
      outliers <- sum(data[[col]] < (Q1 - 1.5 * IQR) | data[[col]] > (Q3 + 1.5 * IQR))
      if(outliers > 0) {
        cat(col, "has", outliers, "outliers\n")
      }
    }
  }
}


str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age      : num  0.708 0.792 0.792 0.167 0.25 ...
##  $ sex      : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp       : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps : num  0.481 0.623 0.245 0.34 0.34 ...
##  $ chol     : num  0.244 0.365 0.235 0.283 0.178 ...
##  $ fbs      : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg  : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach  : num  0.603 0.282 0.443 0.885 0.771 ...
##  $ exang    : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak  : num  0.371 0.242 0.419 0.565 0.226 ...
##  $ slope    : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca       : num  0 1 0.667 0 0 ...
##  $ thal     : Factor w/ 4 levels "3","4.73421926910299",..: 3 1 4 1 1 1 1 1 4 4 ...
##  $ condition: Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
```

```r
# Apply the function to your dataframe
identify_outliers(df)
```

```
## trestbps has 9 outliers
## chol has 5 outliers
## fbs has 45 outliers
## thalach has 1 outliers
## oldpeak has 5 outliers
## ca has 20 outliers
```

```
# str(df)

# Splitting the data
X <- df[, 1:13]
y <- df[, 14]

# Setting a seed for reproducibility
set.seed(123)
trainIndex <- createDataPartition(df$condition, p = 0.8, list = FALSE)

# Splitting the dataframe into training and testing sets
train <- df[trainIndex, ]
test <- df[-trainIndex, ]

# Display the structure of the training set
str(train)
```

```
## 'data.frame':    244 obs. of  14 variables:
##  $ age      : num  0.708 0.792 0.792 0.562 0.583 ...
##  $ sex      : num  1 1 1 1 0 1 1 1 0 1 ...
##  $ cp       : Factor w/ 4 levels "1","2","3","4": 1 4 4 2 4 4 4 4 2 3 ...
##  $ trestbps : num  0.481 0.623 0.245 0.245 0.245 ...
##  $ chol     : num  0.244 0.365 0.235 0.251 0.521 ...
##  $ fbs      : num  1 0 0 0 0 0 1 0 0 1 ...
##  $ restecg  : Factor w/ 3 levels "0","1","2": 3 3 3 1 1 3 3 1 3 3 ...
##  $ thalach  : num  0.603 0.282 0.443 0.817 0.702 ...
##  $ exang    : num  0 1 1 0 1 0 1 0 0 1 ...
##  $ oldpeak  : num  0.371 0.2419 0.4194 0.129 0.0968 ...
##  $ slope    : Factor w/ 3 levels "1","2","3": 3 2 2 1 1 2 3 2 2 2 ...
##  $ ca       : num  0 1 0.667 0 0 ...
##  $ thal     : Factor w/ 4 levels "3","4.73421926910299",..: 3 1 4 1 1 4 4 3 1 3 ...
##  $ condition: Factor w/ 2 levels "0","1": 1 2 2 1 1 2 2 1 1 2 ...
```

## – MODEL CONSTRUCTION AND EVALUATION –

Support Vector Machine (SVM) is a model used for both classification and regression tasks. It is particularly well-suited for complex but smaller- or medium-sized datasets.

```
# Sum of NA values in test data
sum(is.na(test))
```

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

```r
str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age      : num  0.708 0.792 0.792 0.167 0.25 ...
##  $ sex      : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp       : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps : num  0.481 0.623 0.245 0.34 0.34 ...
##  $ chol     : num  0.244 0.365 0.235 0.283 0.178 ...
##  $ fbs      : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg  : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach  : num  0.603 0.282 0.443 0.885 0.771 ...
##  $ exang    : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak  : num  0.371 0.242 0.419 0.565 0.226 ...
##  $ slope    : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca       : num  0 1 0.667 0 0 ...
##  $ thal     : Factor w/ 4 levels "3","4.73421926910299",..: 3 1 4 1 1 1 1 1 4 4 ...
##  $ condition: Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
```

```r
#### SVM model. ####
train$condition <- as.factor(train$condition)
svm_model <- ksvm(condition ~ ., data = train, prob.model=TRUE, kernel="rbfdot")

# Ensure you get predicted probabilities for ROC analysis
svm_pred_probs <- predict(svm_model, test, type = "probabilities")[,2]  # [,2] for the probability of t

# Summary of the trained SVM model
summary(svm_model)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```r
# Ensure that actual condition is a factor
test$condition <- as.factor(test$condition)

# Creating a confusion matrix using predicted classes
svm_pred_classes <- predict(svm_model, test, type = "response")  # Predicted classes
confusion_matrix <- table(Predicted = svm_pred_classes, Actual = test$condition)

# Calculating the accuracy of the SVM model
accuracy_svm <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

# Detailed confusion matrix analysis using predicted classes
confusion_svm <- confusionMatrix(svm_pred_classes, test$condition)
confusion_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  5
##          1  3 22
##
```

```
##                Accuracy : 0.8644
##                  95% CI : (0.7502, 0.9396)
##     No Information Rate : 0.5424
##     P-Value [Acc > NIR] : 1.458e-07
##
##                   Kappa : 0.7253
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.9062
##             Specificity : 0.8148
##          Pos Pred Value : 0.8529
##          Neg Pred Value : 0.8800
##              Prevalence : 0.5424
##          Detection Rate : 0.4915
##    Detection Prevalence : 0.5763
##       Balanced Accuracy : 0.8605
##
##        'Positive' Class : 0
##
```

```r
# Calculating Precision and Recall using Caret's confusion matrix
precision_svm <- confusion_svm$byClass['Precision']
recall_svm <- confusion_svm$byClass['Recall']

# Calculating F1 Score
F1_score_svm <- 2 * (precision_svm * recall_svm) / (precision_svm + recall_svm)

# Creating the ROC object with the predicted probabilities
roc_result <- roc(response = as.numeric(test$condition), predictor = svm_pred_probs)

# Printing the ROC result
print(roc_result)
```

```
##
## Call:
## roc.default(response = as.numeric(test$condition), predictor = svm_pred_probs)
##
## Data: svm_pred_probs in 32 controls (as.numeric(test$condition) 1) < 27 cases (as.numeric(test$condi
## Area under the curve: 0.9051
```

```r
# AUC is directly available from the roc_result object
AUC_svm <- roc_result$auc

# Print Metrics
print(paste("Accuracy:", accuracy_svm))
```

```
## [1] "Accuracy: 0.864406779661017"
```

```r
print(paste("Precision:", precision_svm))
```

```
## [1] "Precision: 0.852941176470588"
```

```r
print(paste("Recall:", recall_svm))
```

```
## [1] "Recall: 0.90625"
```

```r
print(paste("F1 Score:", F1_score_svm))
```

```
## [1] "F1 Score: 0.878787878787879"
```

```r
print(paste("AUC:", AUC_svm))
```

```
## [1] "AUC: 0.905092592592593"
```

The Support Vector Machine (SVM) model demonstrates strong performance, achieving an accuracy of 86.44% in classifying the data. With a precision of 85.29% and recall of 90.62%, the model showcases balanced predictive power, further confirmed by an F1 Score of 87.88%. Moreover, the Area Under the Curve (AUC) value of 0.905 indicates robust overall model performance in distinguishing between the classes.

### Naive Bayes Model

The Naive Bayes model is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

```r
#### NAIVE BAYES model. ####
# Train Naive Bayes model
nb_model <- naiveBayes(condition ~ ., data = train)

# Make predictions using the model
# For AUC calculation, we need predicted probabilities
nb_pred_probs <- predict(nb_model, newdata = test, type = "raw")

# The predicted class will be the one with the highest probability
nb_pred_classes <- apply(nb_pred_probs, 1, which.max) - 1  # subtract 1 because R indexing starts at 1

# Confusion Matrix using predicted classes
nb_conf <- table(Predicted = nb_pred_classes, Actual = test$condition)

# Calculate the accuracy of the Naive Bayes model using the confusion matrix
accuracy_nb <- sum(diag(nb_conf)) / sum(nb_conf)

# Confusion Matrix Analysis using Caret's confusionMatrix with predicted classes
confusion_nb <- confusionMatrix(as.factor(nb_pred_classes), test$condition)

# Precision and Recall
precision_nb <- confusion_nb$byClass['Precision']
recall_nb <- confusion_nb$byClass['Recall']

# F1 Score
F1_score_nb <- 2 * (precision_nb * recall_nb) / (precision_nb + recall_nb)
```

```r
# For AUC calculation, we use the probabilities of the positive class (usually the second column)
# converted from factors to the appropriate numeric binary representation
numeric_actuals_nb <- as.numeric(levels(test$condition))[test$condition]

# Creating the ROC object with the correct numeric conversion and probabilities
roc_nb <- roc(response = numeric_actuals_nb, predictor = nb_pred_probs[,2])

# Accessing the AUC from the ROC object
AUC_nb <- roc_nb$auc

# Print Metrics
print(paste("Accuracy:", accuracy_nb))
```

```
## [1] "Accuracy: 0.779661016949153"
```

```r
print(paste("Precision:", precision_nb))
```

```
## [1] "Precision: 0.787878787878788"
```

```r
print(paste("Recall:", recall_nb))
```

```
## [1] "Recall: 0.8125"
```

```r
print(paste("F1 Score:", F1_score_nb))
```

```
## [1] "F1 Score: 0.8"
```

```r
print(paste("AUC:", AUC_nb))
```

```
## [1] "AUC: 0.898148148148148"
```

The Naive Bayes model demonstrates respectable performance with an accuracy of 77.97%. Showing balanced precision (78.79%) and recall (81.25%), it maintains a commendable F1 Score of 80%, reflecting its ability to harmonize precision and recall. Additionally, the model exhibits strong discriminatory power with an AUC value of 0.898, indicating its capability to distinguish between the classes effectively.

### Logistic Regression Model

Logistic Regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict the probability of a target variable.

```r
####. LOGISTIC REGRESSION MODEL. ####

# Train Logistic Regression model
log_model <- glm(condition ~ ., data = train, family = binomial)
log_pred_probs <- predict(log_model, newdata = test, type = "response")
log_pred <- ifelse(log_pred_probs > 0.5, 1, 0)
```

```r
# Confusion Matrix
confusion_matrix_glm <- table(Predicted = log_pred, Actual = test$condition)

# Calculate the accuracy of Logistic Regression model using the confusion matrix
accuracy_logistic <- sum(diag(confusion_matrix_glm)) / sum(confusion_matrix_glm)

# Confusion Matrix
conf_matrix_glm <- confusionMatrix(as.factor(log_pred), test$condition)

# Precision and Recall
precision_logistic <- conf_matrix_glm$byClass['Precision']
recall_logistic <- conf_matrix_glm$byClass['Recall']

# F1 Score
F1_score_logistic <- 2 * (precision_logistic * recall_logistic) / (precision_logistic + recall_logistic)



numeric_actuals_logistic <- as.numeric(levels(test$condition))[test$condition]

# Creating the ROC object
roc_log <- roc(response = numeric_actuals_logistic, predictor = log_pred_probs)

# AUC is directly available from the ROC object
AUC_logistic <- roc_log$auc


# Print Metrics
print(paste("accuracy (Logistic Regression):", accuracy_logistic))
```

```
## [1] "accuracy (Logistic Regression): 0.864406779661017"
```

```r
print(paste("Precision (Logistic Regression):", precision_logistic))
```

```
## [1] "Precision (Logistic Regression): 0.875"
```

```r
print(paste("Recall (Logistic Regression):", recall_logistic))
```

```
## [1] "Recall (Logistic Regression): 0.875"
```

```r
print(paste("F1 Score (Logistic Regression):", F1_score_logistic))
```

```
## [1] "F1 Score (Logistic Regression): 0.875"
```

```r
print(paste("AUC (Logistic Regression):", AUC_logistic))
```

```
## [1] "AUC (Logistic Regression): 0.881944444444445"
```

The Logistic Regression model showcases robust performance, yielding an accuracy of 86.44%. It maintains balanced precision (87.5%) and recall (87.5%), reflecting its ability to make accurate positive predictions while capturing a significant portion of actual positives. The model's F1 Score of 87.5% emphasizes its harmonized precision and recall, while the AUC value of 0.882 underscores its strong discriminatory capability in distinguishing between the classes.

### Decision Tree

```
####. DECISION TREE MODEL. ####
# Train Decision Tree model
tree_model <- rpart(condition ~ ., data = train, method = "class")

# Predicted probabilities for the positive class (assuming "1" is the positive class)
tree_pred_probs <- predict(tree_model, newdata = test, type = "prob")[,2]

# Binary predictions based on the highest probability
tree_pred_classes <- as.factor(ifelse(tree_pred_probs > 0.5, "1", "0"))

# Confusion Matrix using binary predictions
conf_matrix_tree <- confusionMatrix(tree_pred_classes, test$condition)
print(conf_matrix_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  6
##          1  5 21
##
##                Accuracy : 0.8136
##                  95% CI : (0.6909, 0.9031)
##     No Information Rate : 0.5424
##     P-Value [Acc > NIR] : 1.224e-05
##
##                   Kappa : 0.6233
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8438
##             Specificity : 0.7778
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.8077
##              Prevalence : 0.5424
##          Detection Rate : 0.4576
##    Detection Prevalence : 0.5593
##       Balanced Accuracy : 0.8108
##
##        'Positive' Class : 0
##
```

```
# Calculate the accuracy of Decision Tree model using confusion matrix
accuracy_tree <- sum(diag(conf_matrix_tree$table)) / sum(conf_matrix_tree$table)

# Precision and Recall
precision_tree <- conf_matrix_tree$byClass['Precision']
recall_tree <- conf_matrix_tree$byClass['Recall']

# F1 Score
```

```r
F1_score_tree <- 2 * (precision_tree * recall_tree) / (precision_tree + recall_tree)

# ROC and AUC calculation using predicted probabilities
#  converted from factors to the appropriate numeric binary representation
numeric_actuals_tree <- as.numeric(levels(test$condition))[test$condition]

# Creating the ROC object
roc_tree <- roc(response = numeric_actuals_tree, predictor = tree_pred_probs)

# AUC is directly available from the ROC object
AUC_tree <- roc_tree$auc

# Print Metrics
print(paste("Accuracy (Decision Tree):", accuracy_tree))
```

```
## [1] "Accuracy (Decision Tree): 0.813559322033898"
```

```r
print(paste("Precision (Decision Tree):", precision_tree))
```

```
## [1] "Precision (Decision Tree): 0.818181818181818"
```

```r
print(paste("Recall (Decision Tree):", recall_tree))
```

```
## [1] "Recall (Decision Tree): 0.84375"
```

```r
print(paste("F1 Score (Decision Tree):", F1_score_tree))
```

```
## [1] "F1 Score (Decision Tree): 0.830769230769231"
```

```r
print(paste("AUC (Decision Tree):", AUC_tree))
```

```
## [1] "AUC (Decision Tree): 0.848958333333333"
```

The Decision Tree model demonstrates solid performance, achieving an accuracy of 81.36%. It maintains respectable precision (81.82%) and recall (84.38%), striking a balance between accurate positive predictions and effectively capturing actual positives. The model's F1 Score of 83.08% reflects its harmonized precision and recall, while the AUC value of 0.849 indicates its strong discriminatory ability in distinguishing between the classes.

### Random Forest

```r
### RANDOM FOREST ###
# Train Random Forest model
rf_model <- randomForest(condition ~ ., data = train, ntree = 500)

# Predicted probabilities for the positive class (assuming "1" is the positive class)
rf_pred_probs <- predict(rf_model, newdata = test, type = "prob")[,2]
```

```r
# Binary predictions based on the highest probability
rf_pred_classes <- as.factor(ifelse(rf_pred_probs > 0.5, "1", "0"))

# Confusion Matrix using binary predictions
conf_matrix_rf <- confusionMatrix(rf_pred_classes, test$condition)
print(conf_matrix_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  6
##          1  3 21
##
##                Accuracy : 0.8475
##                  95% CI : (0.7301, 0.9278)
##     No Information Rate : 0.5424
##     P-Value [Acc > NIR] : 7.195e-07
##
##                   Kappa : 0.69
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.9062
##             Specificity : 0.7778
##          Pos Pred Value : 0.8286
##          Neg Pred Value : 0.8750
##              Prevalence : 0.5424
##          Detection Rate : 0.4915
##    Detection Prevalence : 0.5932
##       Balanced Accuracy : 0.8420
##
##        'Positive' Class : 0
##
```

```r
# Calculate the accuracy of Random Forest model using confusion matrix
accuracy_rf <- sum(diag(conf_matrix_rf$table)) / sum(conf_matrix_rf$table)

# Precision and Recall
precision_rf <- conf_matrix_rf$byClass['Precision']
recall_rf <- conf_matrix_rf$byClass['Recall']

# F1 Score
F1_score_rf <- 2 * (precision_rf * recall_rf) / (precision_rf + recall_rf)

# ROC and AUC calculation using predicted probabilities
# converted from factors to the appropriate numeric binary representation
numeric_actuals_rf <- as.numeric(levels(test$condition))[test$condition]

# Creating the ROC object
roc_rf <- roc(response = numeric_actuals_rf, predictor = rf_pred_probs)

# AUC is directly available from the ROC object
```

```r
AUC_rf <- roc_rf$auc

# Print Metrics
print(paste("Accuracy (Random Forest):", accuracy_rf))
```

```
## [1] "Accuracy (Random Forest): 0.847457627118644"
```

```r
print(paste("Precision (Random Forest):", precision_rf))
```

```
## [1] "Precision (Random Forest): 0.828571428571429"
```

```r
print(paste("Recall (Random Forest):", recall_rf))
```

```
## [1] "Recall (Random Forest): 0.90625"
```

```r
print(paste("F1 Score (Random Forest):", F1_score_rf))
```

```
## [1] "F1 Score (Random Forest): 0.865671641791045"
```

```r
print(paste("AUC (Random Forest):", AUC_rf))
```

```
## [1] "AUC (Random Forest): 0.914351851851852"
```

The Random Forest model showcases strong performance, achieving an accuracy of 84.75%. It demonstrates high sensitivity (90.62%) and good specificity (77.78%), indicating its ability to correctly identify the majority of actual positives while maintaining a balanced performance for negatives. With a precision of 82.86% and a recall of 90.62%, the model exhibits a robust F1 Score of 86.57%, signifying its balanced precision and recall. Moreover, the impressive AUC value of 0.914 reflects its strong discriminatory power in distinguishing between the classes.

**Application of ensemble to make a prediction.**

```r
predictOutcomeClass <- function(newdata, nb_model, glm_model, svm_model) {
  # Store ensemble predictions
  ensemble_predictions <- character(nrow(newdata))

  for (i in 1:nrow(newdata)) {
    # Predict using Naive Bayes
    nb_prediction <- predict(nb_model, newdata = newdata[i, , drop = FALSE], type = "class")
    nb_prediction <- as.character(nb_prediction)

    # Predict using Logistic Regression
    glm_prob <- predict(glm_model, newdata = newdata[i, , drop = FALSE], type = "response")
    glm_prediction <- ifelse(glm_prob > 0.5, "1", "0")

    # Predict using SVM
    svm_prediction <- predict(svm_model, newdata = newdata[i, , drop = FALSE])
    svm_prediction <- as.character(svm_prediction)
```

```
    # Combine predictions
    predictions <- c(nb_prediction, glm_prediction, svm_prediction)

    # Use table to count the occurrences of each prediction
    tbl <- table(predictions)

    # Calculate majority vote
    majority_vote <- names(which.max(tbl))

    # Store the majority vote
    ensemble_predictions[i] <- majority_vote
  }

  return(as.factor(ensemble_predictions))
}

# Usage of the Function
ensemble_predictions <- predictOutcomeClass(test, nb_model, log_model ,svm_model)



# Metrics Calculation
conf_matrix_ensemble <- confusionMatrix(ensemble_predictions, test$condition)
accuracy_ensemble <- conf_matrix_ensemble$overall['Accuracy']
precision_ensemble <- conf_matrix_ensemble$byClass['Precision']
recall_ensemble <- conf_matrix_ensemble$byClass['Recall']
F1_score_ensemble <- 2 * (precision_ensemble * recall_ensemble) / (precision_ensemble + recall_ensemble)

# ROC and AUC calculation (ensuring predictions are numeric)
roc_result <- roc(response = as.numeric(test$condition), predictor = as.numeric(ensemble_predictions))

# Extract AUC from roc_result
AUC_ensemble <- roc_result$auc

# Print Metrics
print(paste("Ensemble Model Accuracy:", accuracy_ensemble))
```

```
## [1] "Ensemble Model Accuracy: 0.864406779661017"
```

```
print(paste("Ensemble Model Precision:", precision_ensemble))
```

```
## [1] "Ensemble Model Precision: 0.852941176470588"
```

```
print(paste("Ensemble Model Recall:", recall_ensemble))
```

```
## [1] "Ensemble Model Recall: 0.90625"
```

```
print(paste("Ensemble Model F1 Score:", F1_score_ensemble))
```

```
## [1] "Ensemble Model F1 Score: 0.878787878787879"
```

```r
print(paste("Ensemble Model AUC:", AUC_ensemble))
```

```
## [1] "Ensemble Model AUC: 0.860532407407407"
```

The ensemble model showcases strong performance with an accuracy of 86.44%. It maintains a balanced precision of 85.29% and recall of 90.62%, signifying its ability to make accurate positive predictions while effectively capturing a significant portion of actual positives. The F1 Score of 87.88% underscores its harmonized precision and recall, while the AUC value of 0.861 indicates its robust discriminatory power in distinguishing between the classes. This ensemble method amalgamates the strengths of individual models, resulting in a well-performing predictive model for Heart disease.

**evaluation with other models**

The ensemble model emerged as a standout, showcasing its prowess with an impressive accuracy of 86.44%. This fusion of diverse models not only achieved a commendable accuracy but also displayed balanced precision and recall, affirming its ability to make precise positive predictions while effectively capturing a substantial portion of actual positives. With an F1 Score of 87.88% and an AUC value of 0.861, the ensemble method underscored its harmonized precision-recall balance and robust discriminatory power in distinguishing between various heart disease classes.

Comparatively, the Random Forest model stood strong with an accuracy of 84.75%, demonstrating high sensitivity and good specificity. Its robust F1 Score of 86.57% and an impressive AUC value of 0.914 portrayed significant discriminatory abilities, almost akin to the ensemble approach.

Further exploration into specific models revealed the Logistic Regression's accuracy matching the ensemble at 86.44%, showcasing balanced precision and recall. Additionally, the Naive Bayes model demonstrated respectable performance with strong discriminatory power.

In juxtaposing the Random Forest and SVM models, both exhibited impressive accuracies and robust performances. The Random Forest model slightly outperformed in sensitivity and specificity, yet the SVM closely followed suit, portraying balanced precision-recall rates and competitive performance in pivotal predictive metrics. This comparative analysis accentuates the nuanced strengths and competitive nature of these models in accurately predicting heart disease.