

Data 622 - Machine Learning and Big Data - HW #2 - Decision Trees Algorithms and Essay

Enid Roman

2024-03-29

```
#install.packages("RCurl", repos = "http://cran.us.r-project.org")
#install.packages("randomForest")
#install.packages("caTools")
library(devtools)
library(RCurl)
library(tidyverse)
library(dplyr)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caTools)
```

Context

This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. The initial dataset came from Kaggle. The referenced dataset is from University of California Irvine (UCI) Machine Learning Repository. It contains 76 attributes, including the predicted attribute, but all published experiments refer to using a subset of 14 of them. The “target” field refers to the presence of heart disease in the patient. It is integer valued 0 = no disease and 1 = disease.

Content

Attribute Information

There are 14 attributes in this dataset and here are their descriptions:

Age: Patient's Age in years. Sex: Patient's Gender. (1 = Male, 0 = Female) ChestPainType(CP): Chest Pain type. (4 values: 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain, 4 = asymptomatic) RestingBP(trestbps): Resting Blood Pressure. (in mm Hg) Cholesterol(chol): Serum Cholesterol. (in mg/dl) FastingBS(fbs): Fasting Blood Sugar > 120 mg/dl. (1 = True, 0 = False) RestingECG(restecg): Resting Electroencephalographic result. (values:0 = Normal, 1 = ST(abnormality stress test), 2 = LVH(probale or definite left ventricular hypertrophy)) Maximum Heart Rate(thalach): Maximum Heart Rate achieved in stress test. Exercise Angina(exang): Exercise Indused Angina (1 = Yes, 0 = No) Stress Test Depression(oldpeak): ST Depression induced by Exercise relative to rest. Slope for Peak Exercise(slope): Slope of the peak exercise ST segment. (1 = Up, 2 = Flat, 3 = Down) Number of major vessels(ca): (0-3) colored by flourosopy Thallium Heart Rate(thal): (0: NULL, Value 1: fixed defect (no blood flow in some part of the heart), Value 2: normal blood flow, Value 3: reversible defect (a blood flow is observed but it is not normal)) HeartDisease(target): Heart Disease occurred. (0 = No, 1 = Yes)

Prior to commencing our data analysis, we'll inspect our dataset for any anomalies that need addressing. It's imperative to emphasize that the reliability of insights hinges on the quality of the underlying data, underscoring the necessity for clean and readily usable data.

Table Preview

```
heart <- read.csv("https://raw.githubusercontent.com/enidroman/Data-622-Machine-Learning-and-Big-Data/main/heart.csv")
head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 1  52  1  0   125   212   0         1    168     0     1.0     2  2    3
## 2  53  1  0   140   203   1         0    155     1     3.1     0  0    3
## 3  70  1  0   145   174   0         1    125     1     2.6     0  0    3
## 4  61  1  0   148   203   0         1    161     0     0.0     2  1    3
## 5  62  0  0   138   294   1         1    106     0     1.9     1  3    2
## 6  58  0  0   100   248   0         0    122     0     1.0     1  0    2
##   target
## 1       0
## 2       0
## 3       0
## 4       0
## 5       0
## 6       1
```

Data Wrangling

Data type

```
glimpse(heart)
```

```
## Rows: 1,025
## Columns: 14
## $ age      <int> 52, 53, 70, 61, 62, 58, 58, 55, 46, 54, 71, 43, 34, 51, 52, 3~
## $ sex      <int> 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1~
## $ cp       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 2, 0, 1, 2, 2~
## $ trestbps <int> 125, 140, 145, 148, 138, 100, 114, 160, 120, 122, 112, 132, 1~
## $ chol     <int> 212, 203, 174, 203, 294, 248, 318, 289, 249, 286, 149, 341, 2~
## $ fbs      <int> 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0~
## $ restecg  <int> 1, 0, 1, 1, 1, 0, 2, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0~
## $ thalach  <int> 168, 155, 125, 161, 106, 122, 140, 145, 144, 116, 125, 136, 1~
## $ exang    <int> 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0~
## $ oldpeak  <dbl> 1.0, 3.1, 2.6, 0.0, 1.9, 1.0, 4.4, 0.8, 0.8, 3.2, 1.6, 3.0, 0~
## $ slope    <int> 2, 0, 0, 2, 1, 1, 0, 1, 2, 1, 1, 1, 2, 1, 1, 2, 2, 1, 2, 2, 1~
## $ ca       <int> 2, 0, 0, 1, 3, 0, 3, 1, 0, 2, 0, 0, 0, 3, 0, 0, 1, 1, 0, 0, 0~
## $ thal     <int> 3, 3, 3, 3, 2, 2, 1, 3, 3, 2, 2, 3, 2, 3, 0, 2, 2, 3, 2, 2, 2~
## $ target   <int> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0~
```

Data type seems to be fine.

Summary

```
summary(heart)
```

```
##      age      sex      cp      trestbps
## Min.   :29.00  Min.   :0.0000  Min.   :0.0000  Min.    : 94.0
## 1st Qu.:48.00  1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:120.0
## Median :56.00  Median :1.0000  Median :1.0000  Median :130.0
## Mean   :54.43  Mean   :0.6956  Mean   :0.9424  Mean   :131.6
## 3rd Qu.:61.00  3rd Qu.:1.0000  3rd Qu.:2.0000  3rd Qu.:140.0
## Max.   :77.00  Max.   :1.0000  Max.   :3.0000  Max.   :200.0
##      chol      fbs      restecg      thalach
## Min.   :126  Min.   :0.0000  Min.   :0.0000  Min.    : 71.0
## 1st Qu.:211  1st Qu.:0.0000  1st Qu.:0.0000  1st Qu.:132.0
## Median :240  Median :0.0000  Median :1.0000  Median :152.0
## Mean   :246  Mean   :0.1493  Mean   :0.5298  Mean   :149.1
## 3rd Qu.:275  3rd Qu.:0.0000  3rd Qu.:1.0000  3rd Qu.:166.0
## Max.   :564  Max.   :1.0000  Max.   :2.0000  Max.   :202.0
##      exang      oldpeak      slope      ca
## Min.   :0.0000  Min.   :0.000  Min.   :0.000  Min.   :0.0000
## 1st Qu.:0.0000  1st Qu.:0.000  1st Qu.:1.000  1st Qu.:0.0000
## Median :0.0000  Median :0.800  Median :1.000  Median :0.0000
## Mean   :0.3366  Mean   :1.072  Mean   :1.385  Mean   :0.7541
## 3rd Qu.:1.0000  3rd Qu.:1.800  3rd Qu.:2.000  3rd Qu.:1.0000
## Max.   :1.0000  Max.   :6.200  Max.   :2.000  Max.   :4.0000
##      thal      target
## Min.   :0.000  Min.   :0.0000
## 1st Qu.:2.000  1st Qu.:0.0000
## Median :2.000  Median :1.0000
## Mean   :2.324  Mean   :0.5132
## 3rd Qu.:3.000  3rd Qu.:1.0000
## Max.   :3.000  Max.   :1.0000
```

Here we can see the minimum age is 29, maximum age is 77, and average age is 56. Resting BPS is minimum 94, maximum 200, and average is 131. Cholesterol minimum is 126, maximum is 564, and average is 240.

Check for Missing Values

```
colSums(is.na(heart))
```

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

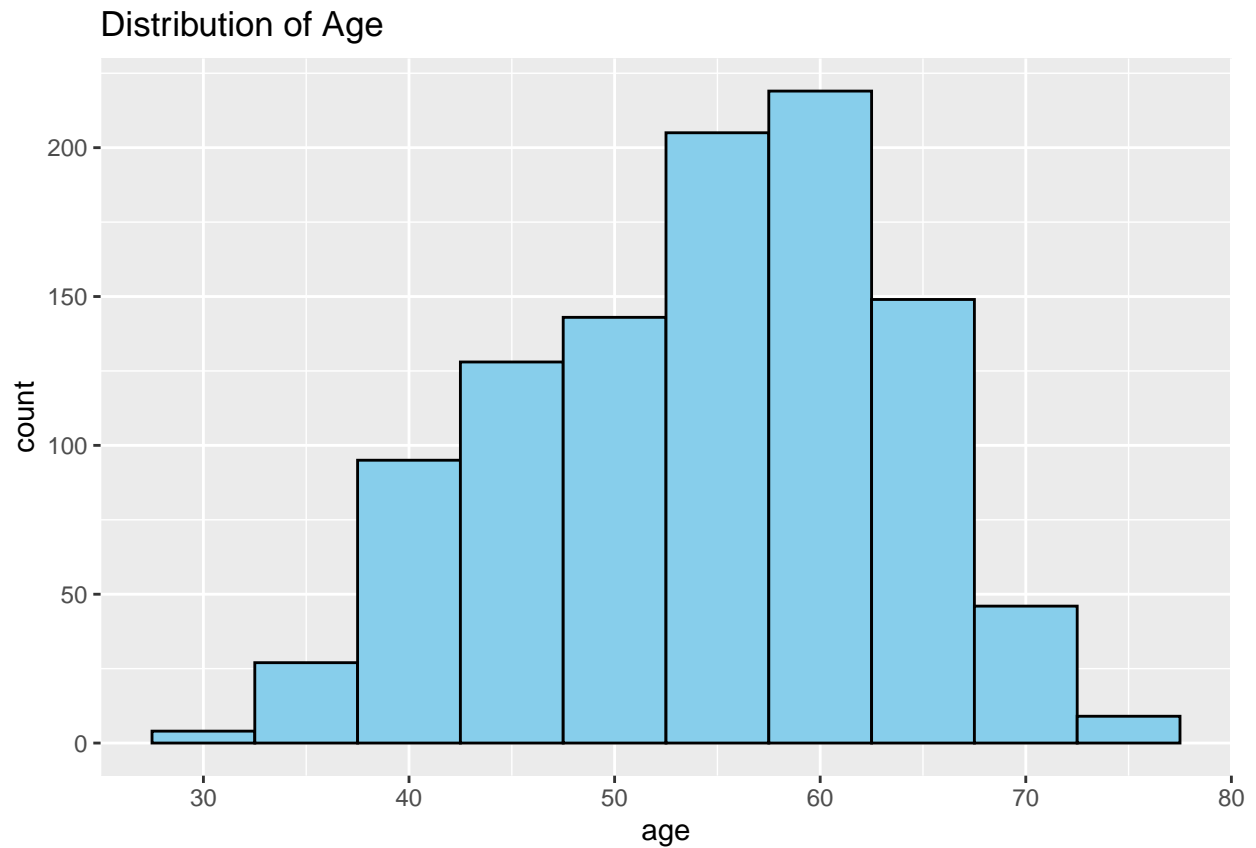
No missing values.

Exploratory Data Analysis

Exploratory data analysis (EDA) involves using graphics and visualizations to explore and analyze a data set. The goal is to explore, investigate and learn the data variables within our dataset.

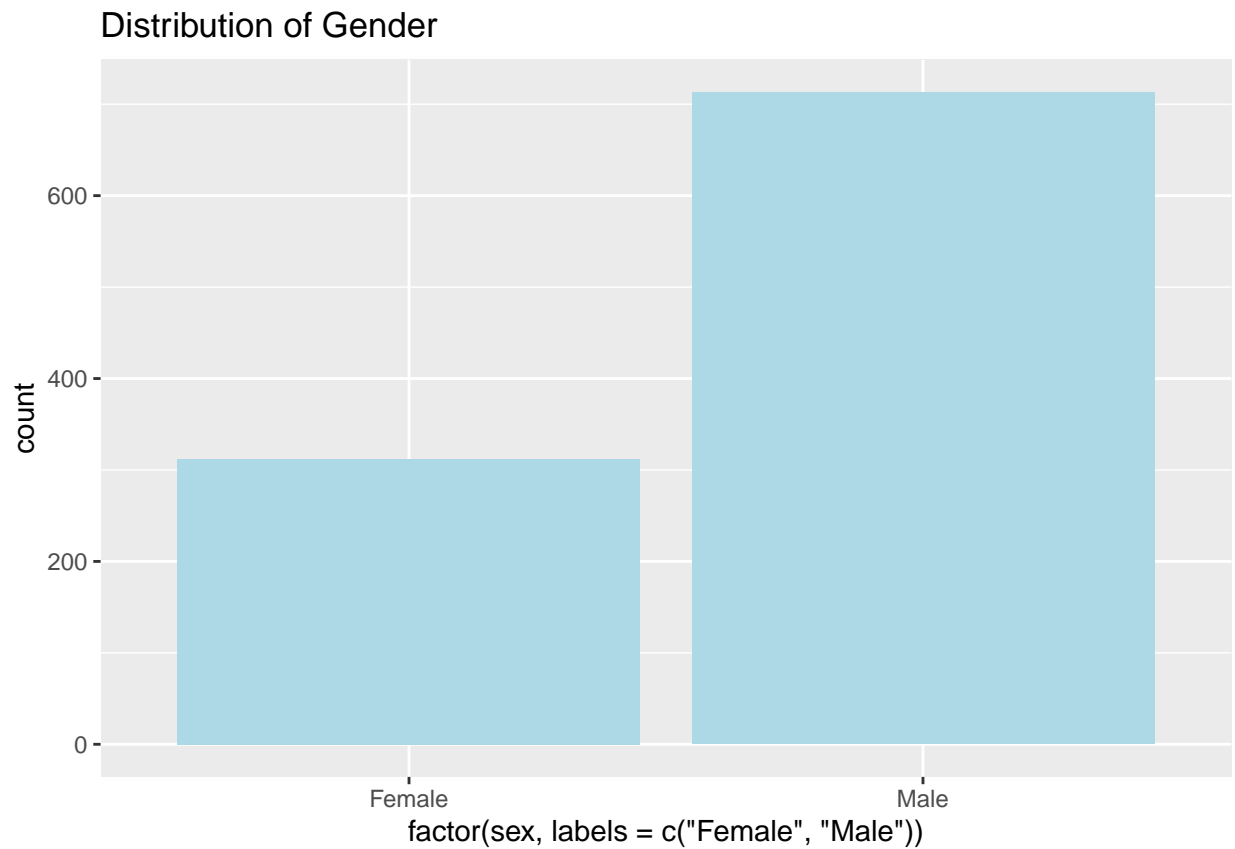
All Attributes

```
ggplot(heart, aes(x = age)) +  
  geom_histogram(binwidth = 5, fill = "skyblue", color = "black") +  
  ggtitle("Distribution of Age")
```



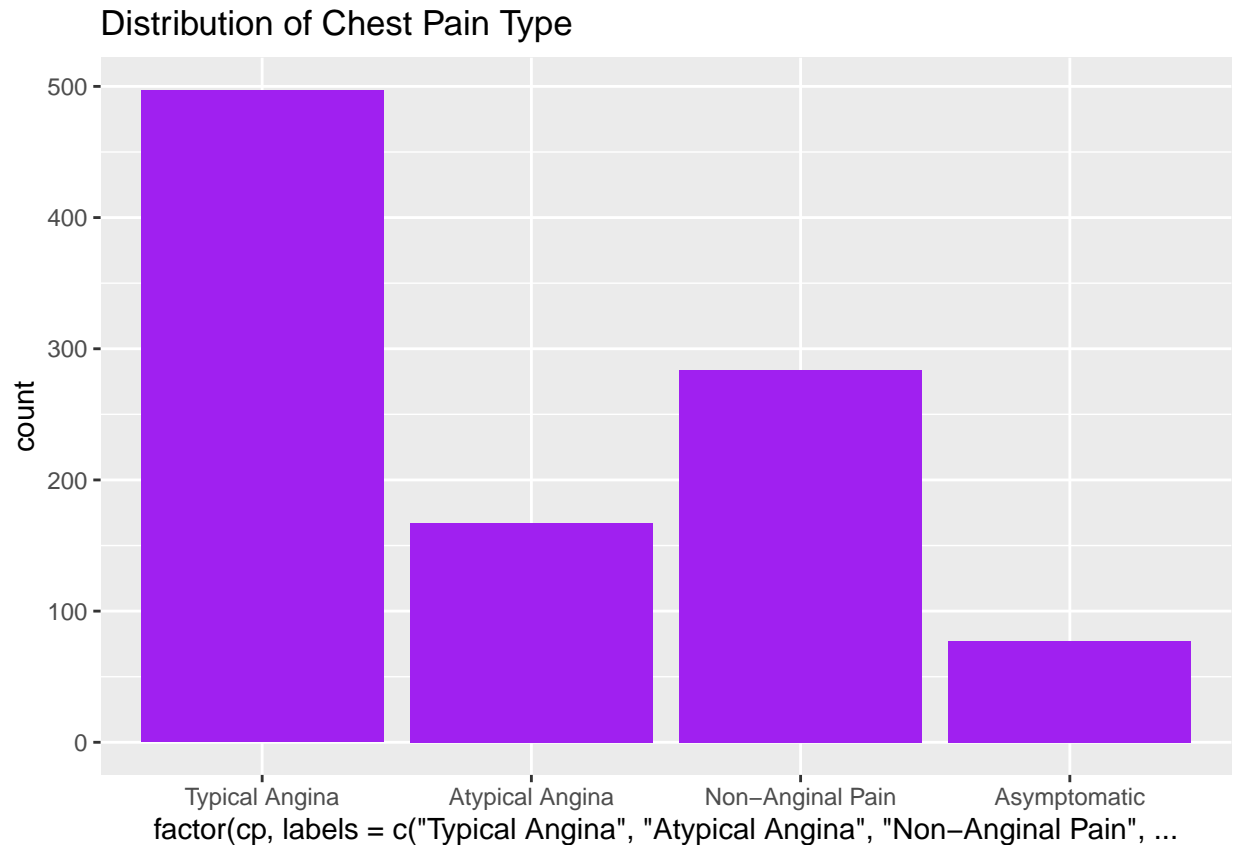
Maximum age of patient is between 50 and 60.

```
ggplot(heart, aes(x = factor(sex, labels = c("Female", "Male")))) +  
  geom_bar(fill = "lightblue") +  
  ggtitle("Distribution of Gender")
```



There are more male patient then female pateint.

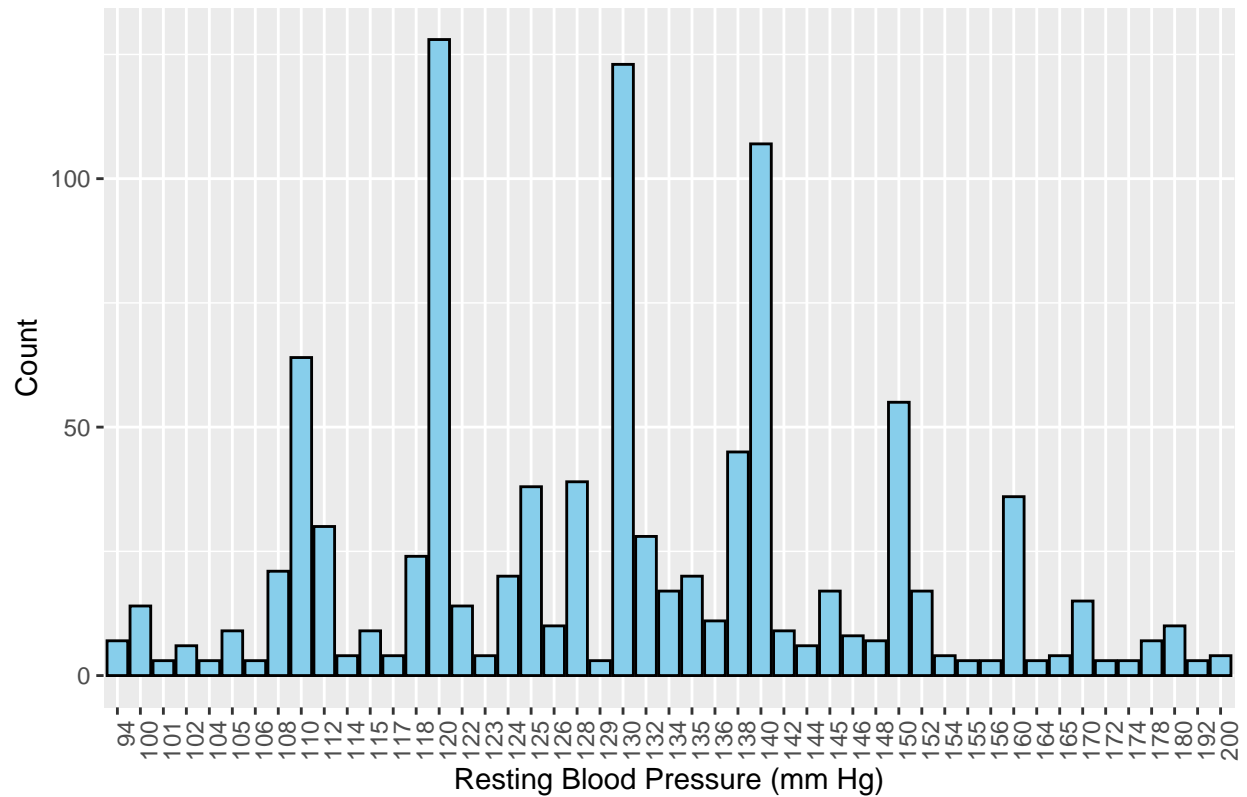
```
ggplot(heart, aes(x = factor(cp, labels = c("Typical Angina", "Atypical Angina", "Non-Anginal Pain", "A  
geom_bar(fill = "purple") +  
ggtitle("Distribution of Chest Pain Type")
```



Most patient complaining of typical angina, a symptom of myocardial ischemia. Typical angina is characterized by chest discomfort or anginal equivalent that is provoked with exertion and alleviated at rest or with nitroglycerin.

```
ggplot(heart, aes(x = factor(trestbps))) +
  geom_bar(fill = "skyblue", color = "black", stat = "count") +
  ggtitle("Count of Resting Blood Pressure") +
  xlab("Resting Blood Pressure (mm Hg)") +
  ylab("Count") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

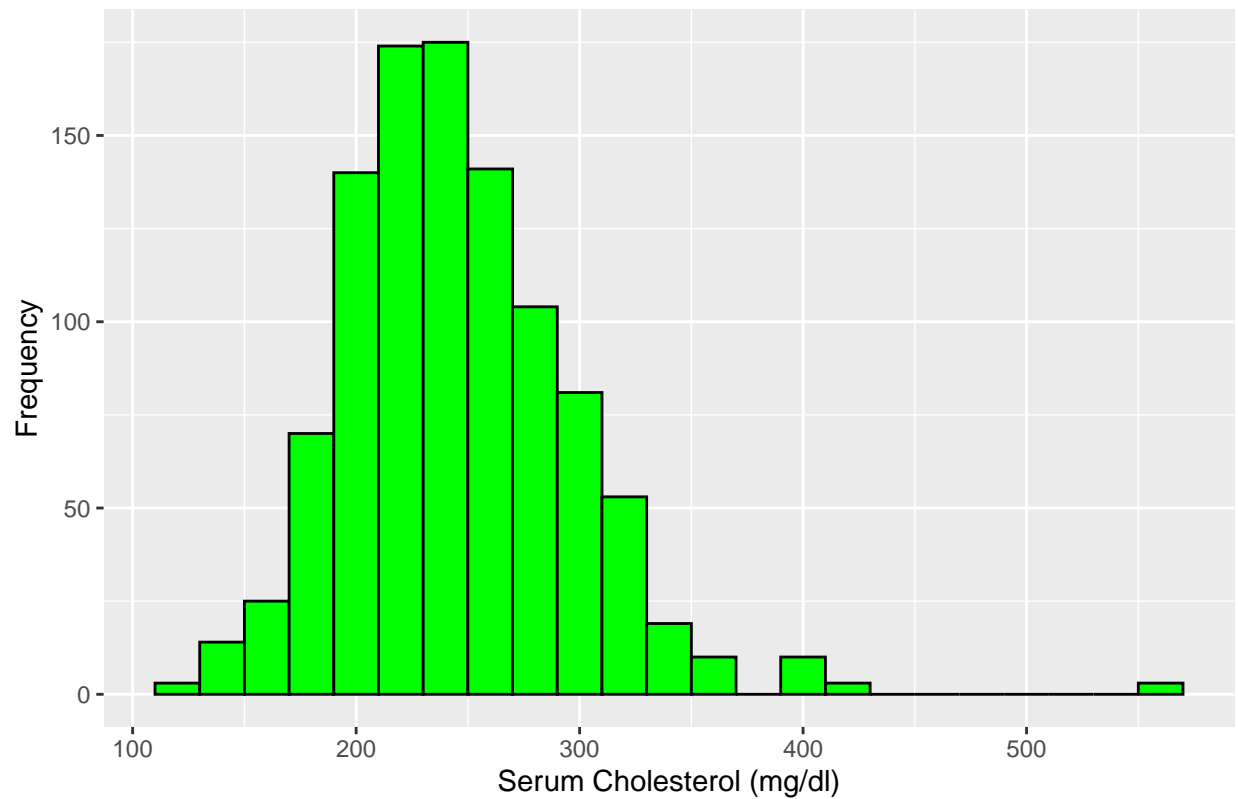
Count of Resting Blood Pressure



Most patient blood pressure was 120, 130, and 140. 120 to 139 are at risk of high bp. 140 is high bp.

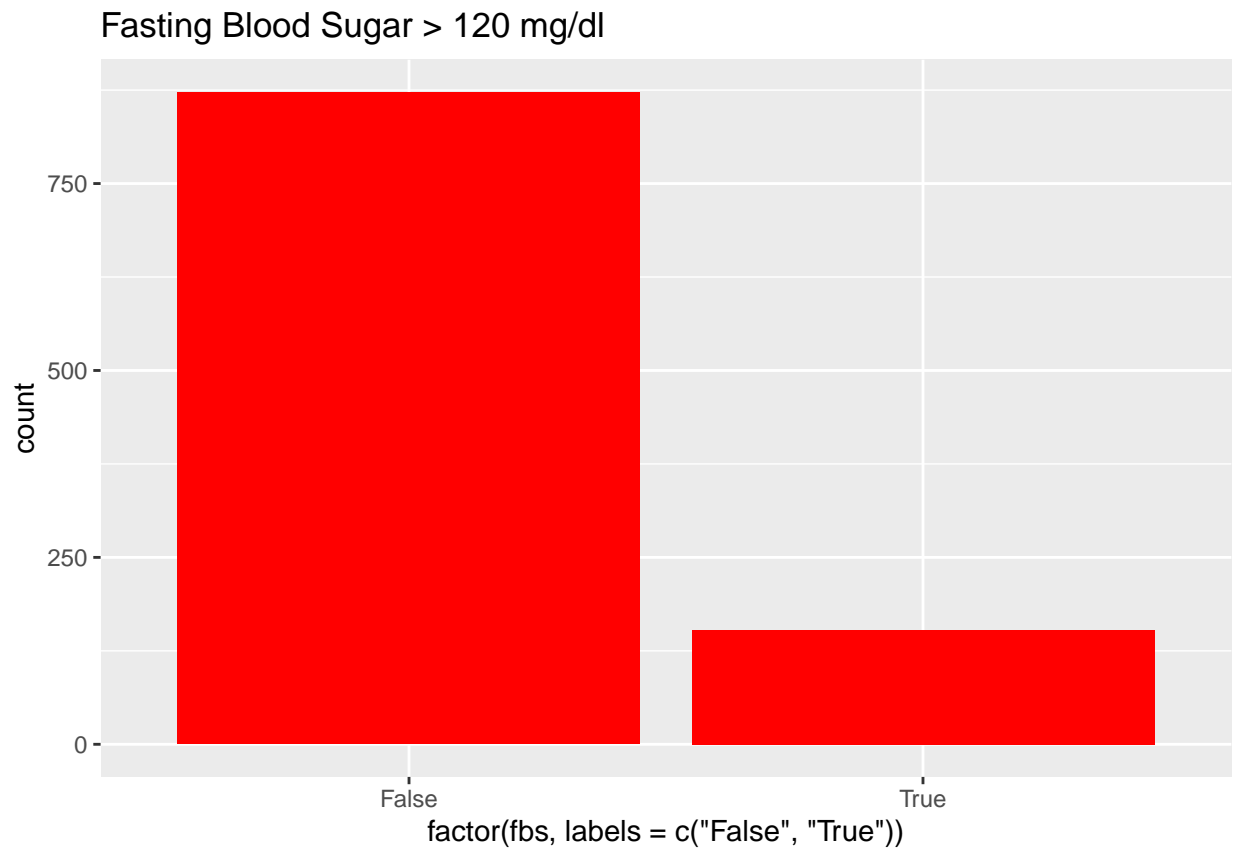
```
ggplot(heart, aes(x = chol)) +
  geom_histogram(binwidth = 20, fill = "green", color = "black") +
  ggtitle("Distribution of Serum Cholesterol") +
  xlab("Serum Cholesterol (mg/dl)") +
  ylab("Frequency")
```

Distribution of Serum Cholesterol



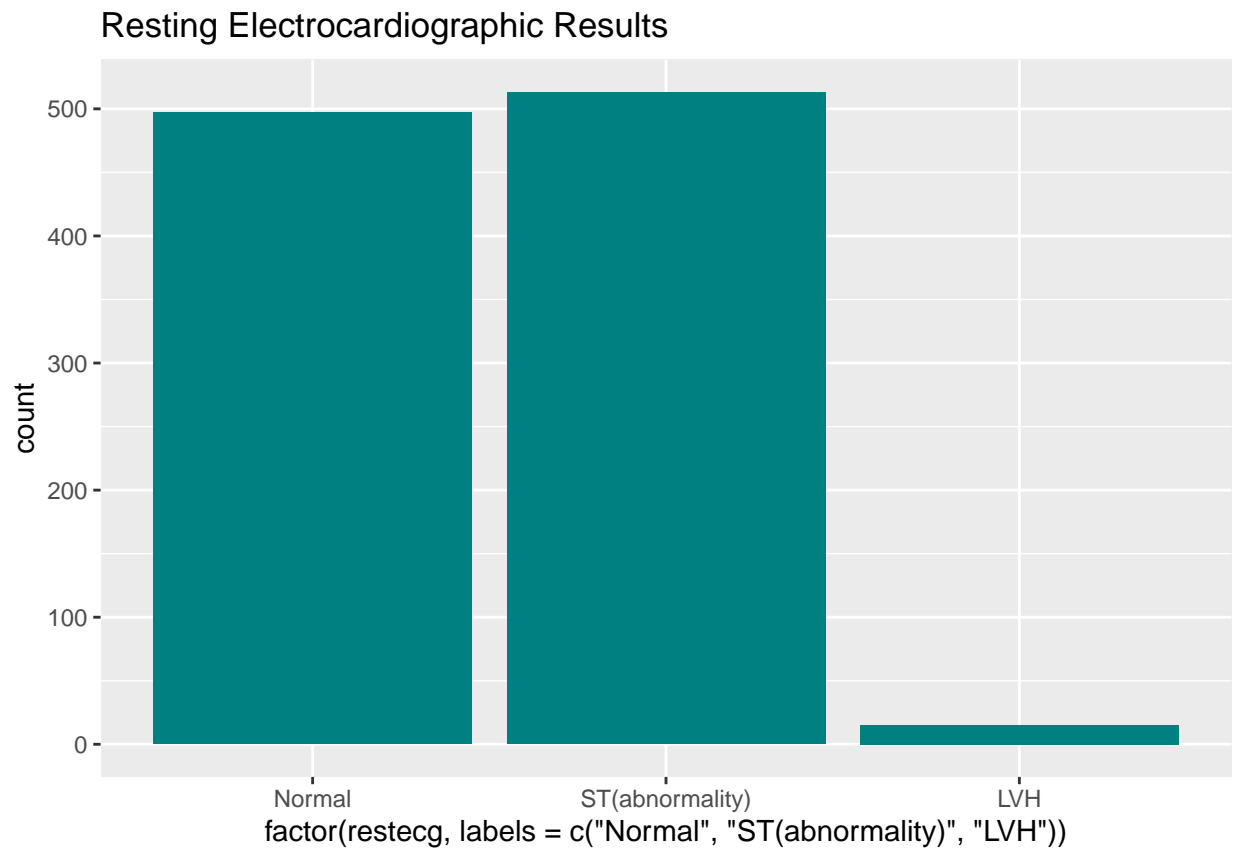
Most patient had cholesterol approximately 200 to 280. 200 to 239 is borderline. 240 and over are high cholesterol.

```
ggplot(heart, aes(x = factor(fbs, labels = c("False", "True")))) +  
  geom_bar(fill = "red") +  
  ggtitle("Fasting Blood Sugar > 120 mg/dl")
```

Most patient did not have high sugar level.

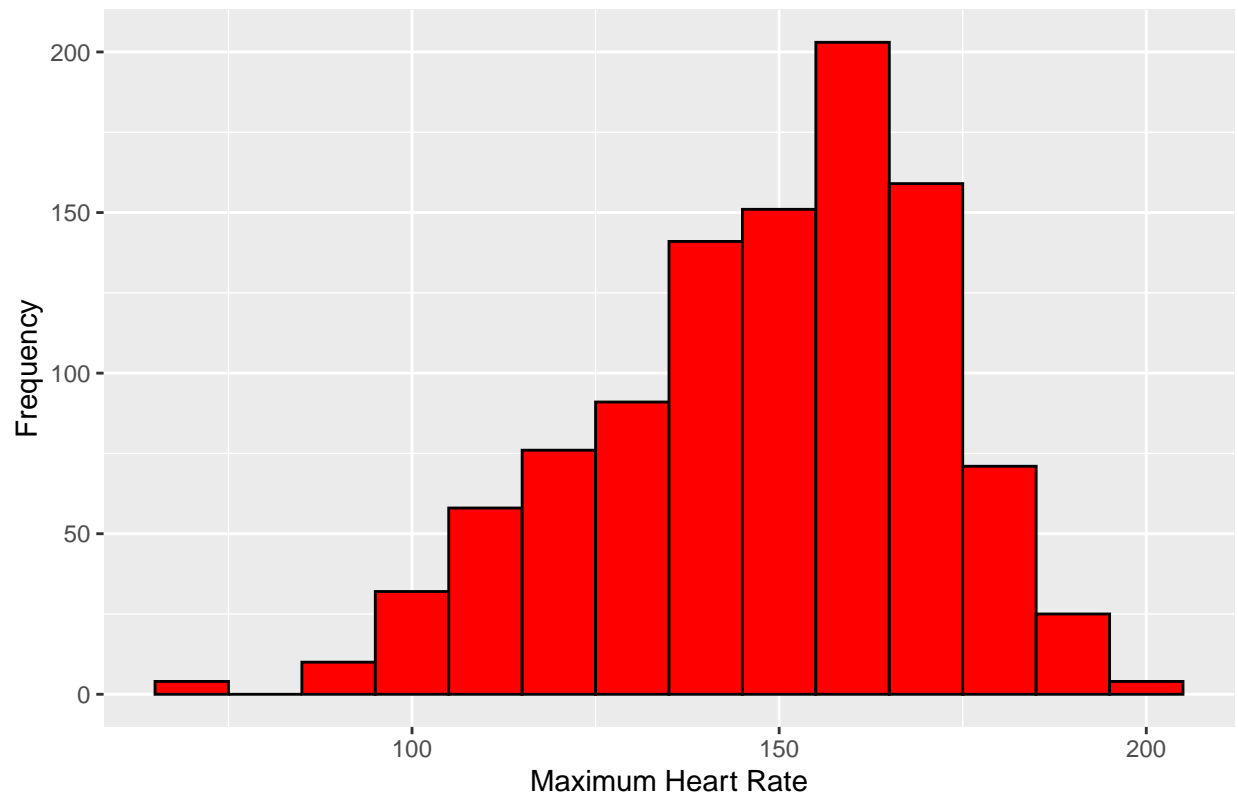
```
ggplot(heart, aes(x = factor(restecg, labels = c("Normal", "ST(abnormality)", "LVH")))) +  
  geom_bar(fill = "#008080") +  
  ggtitle("Resting Electrocardiographic Results")
```



Most patient had abnormal electrocardiogram. But also there was almost the same amount of patient that had normal electrocardiogram.

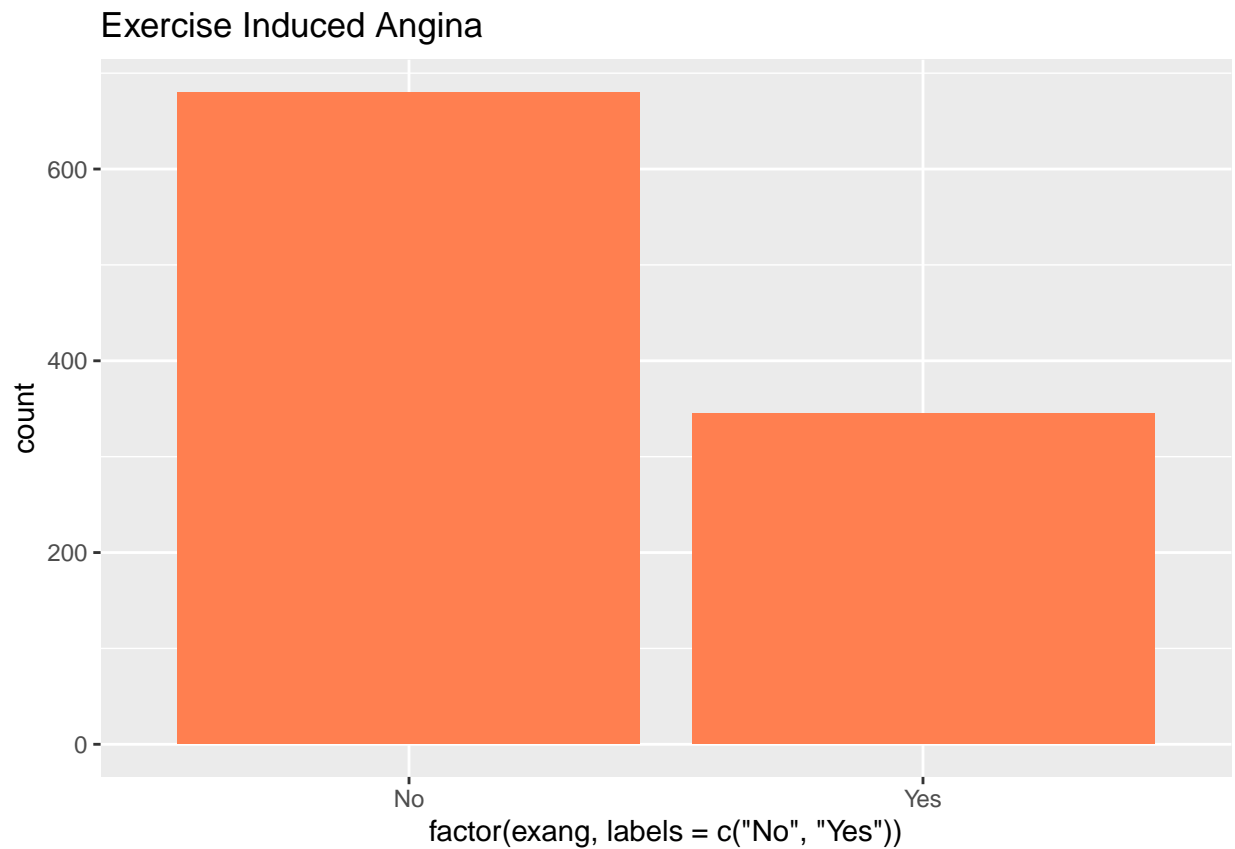
```
ggplot(heart, aes(x = thalach)) +  
  geom_histogram(binwidth = 10, fill = "red", color = "black") +  
  ggtitle("Distribution of Maximum Heart Rate Achieved in Stress Test") +  
  xlab("Maximum Heart Rate") +  
  ylab("Frequency")
```

Distribution of Maximum Heart Rate Achieved in Stress Test



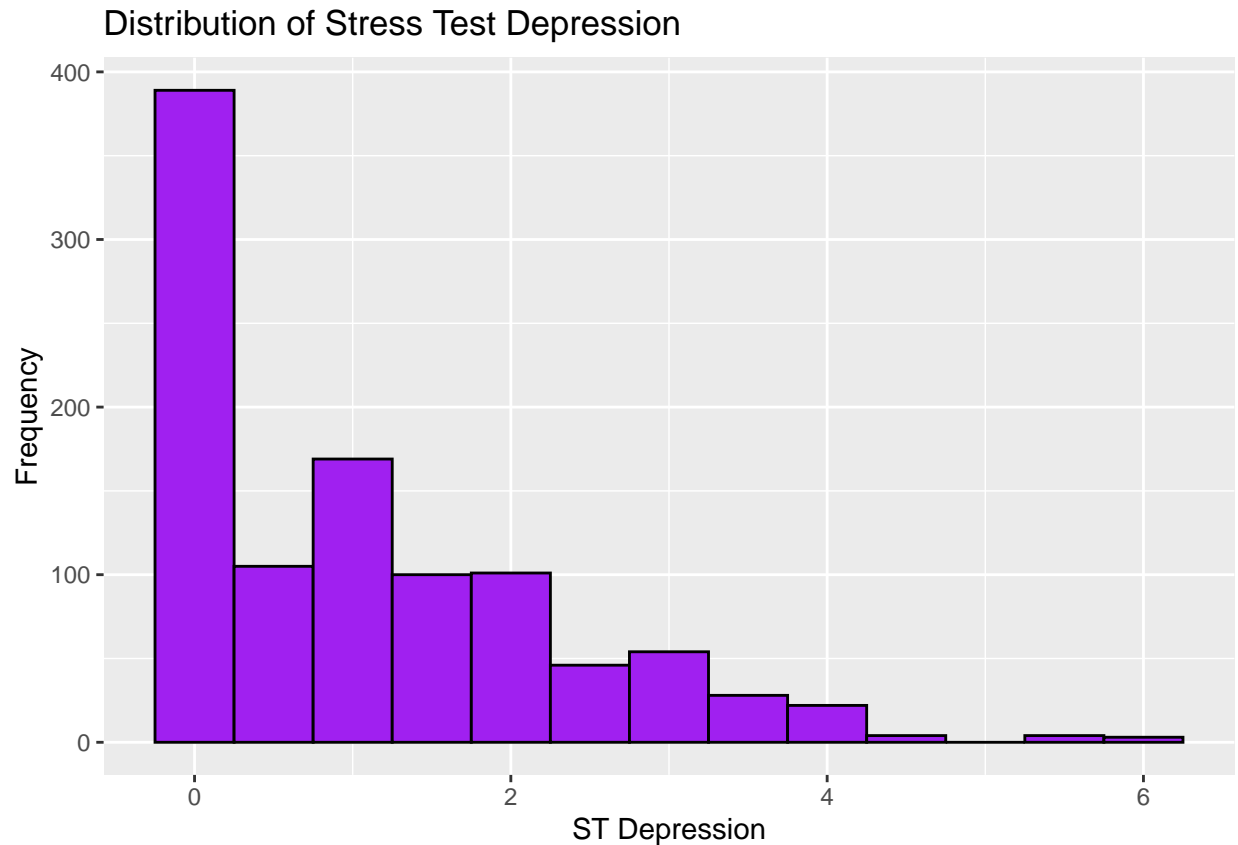
Maximum heart rate patients reached in a stress test was approximately 153. The target heart rate during a stress test depends on your age. Is the maximum predicted heart rate minus the age.

```
ggplot(heart, aes(x = factor(exang, labels = c("No", "Yes")))) +  
  geom_bar(fill = "coral") +  
  ggtitle("Exercise Induced Angina")
```



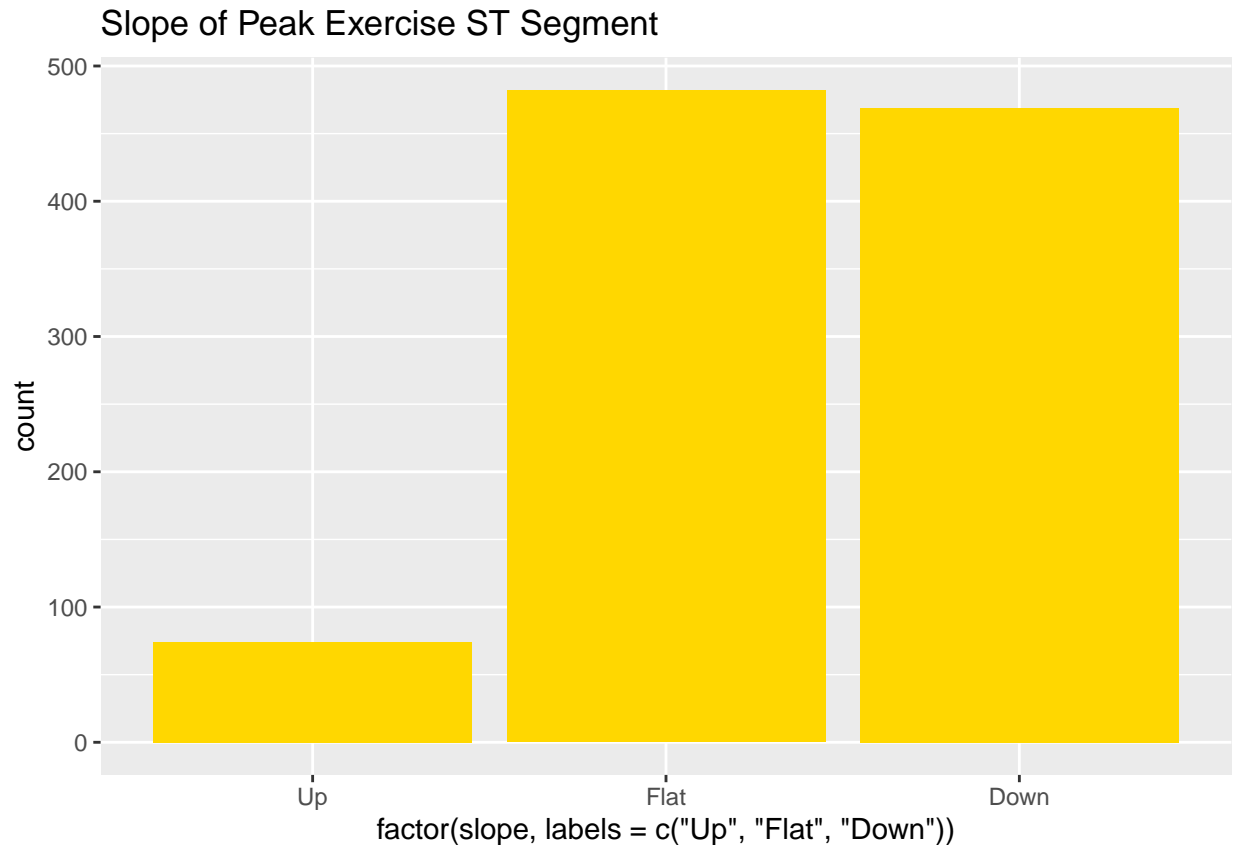
The majority patient did not have angina induced when exercised.

```
ggplot(heart, aes(x = oldpeak)) +  
  geom_histogram(binwidth = 0.5, fill = "purple", color = "black") +  
  ggtitle("Distribution of Stress Test Depression") +  
  xlab("ST Depression") +  
  ylab("Frequency")
```



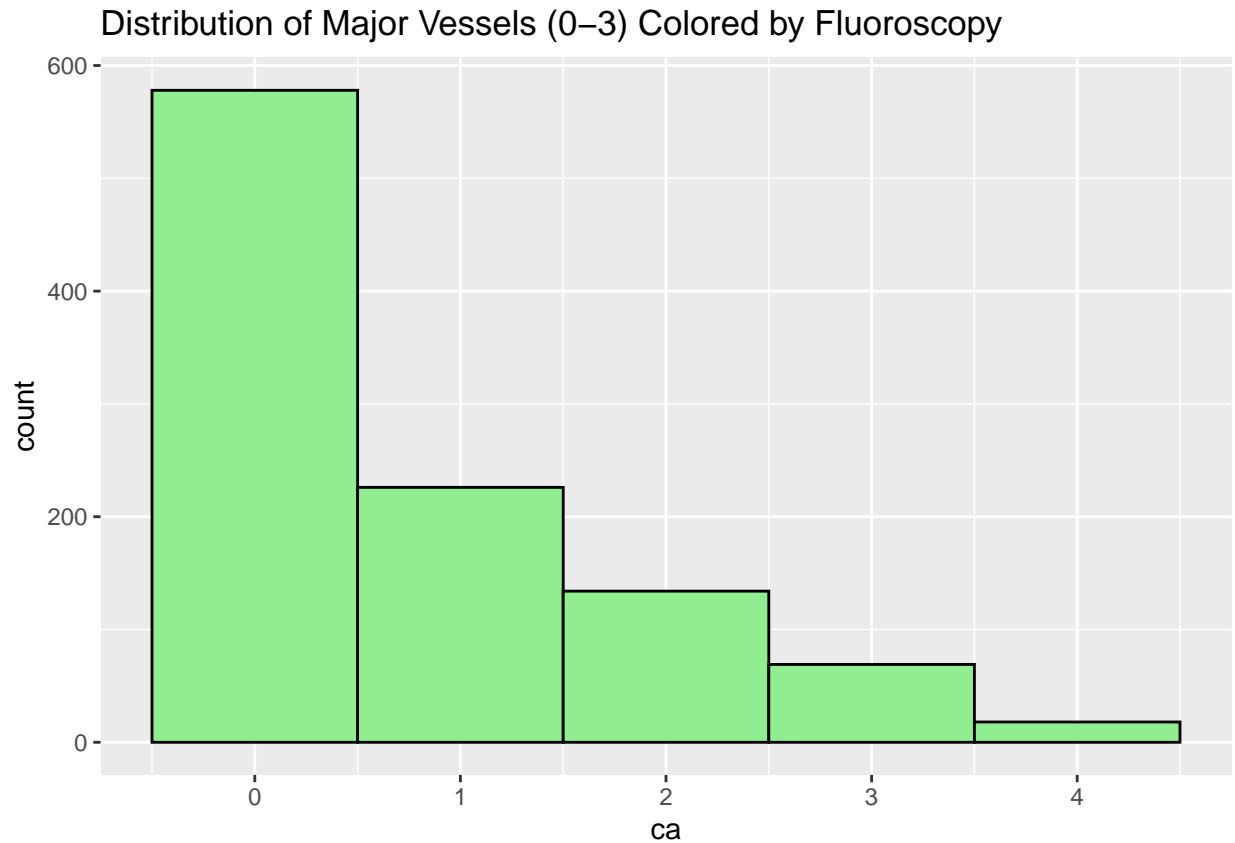
Most patient did not have cardiac abnormalities. The values of “oldpeak” range from 0 to 6, indicating the magnitude of the depression observed during the stress test. A higher value typically suggests more significant cardiac abnormalities.

```
ggplot(heart, aes(x = factor(slope, labels = c("Up", "Flat", "Down")))) +  
  geom_bar(fill = "gold") +  
  ggtitle("Slope of Peak Exercise ST Segment")
```



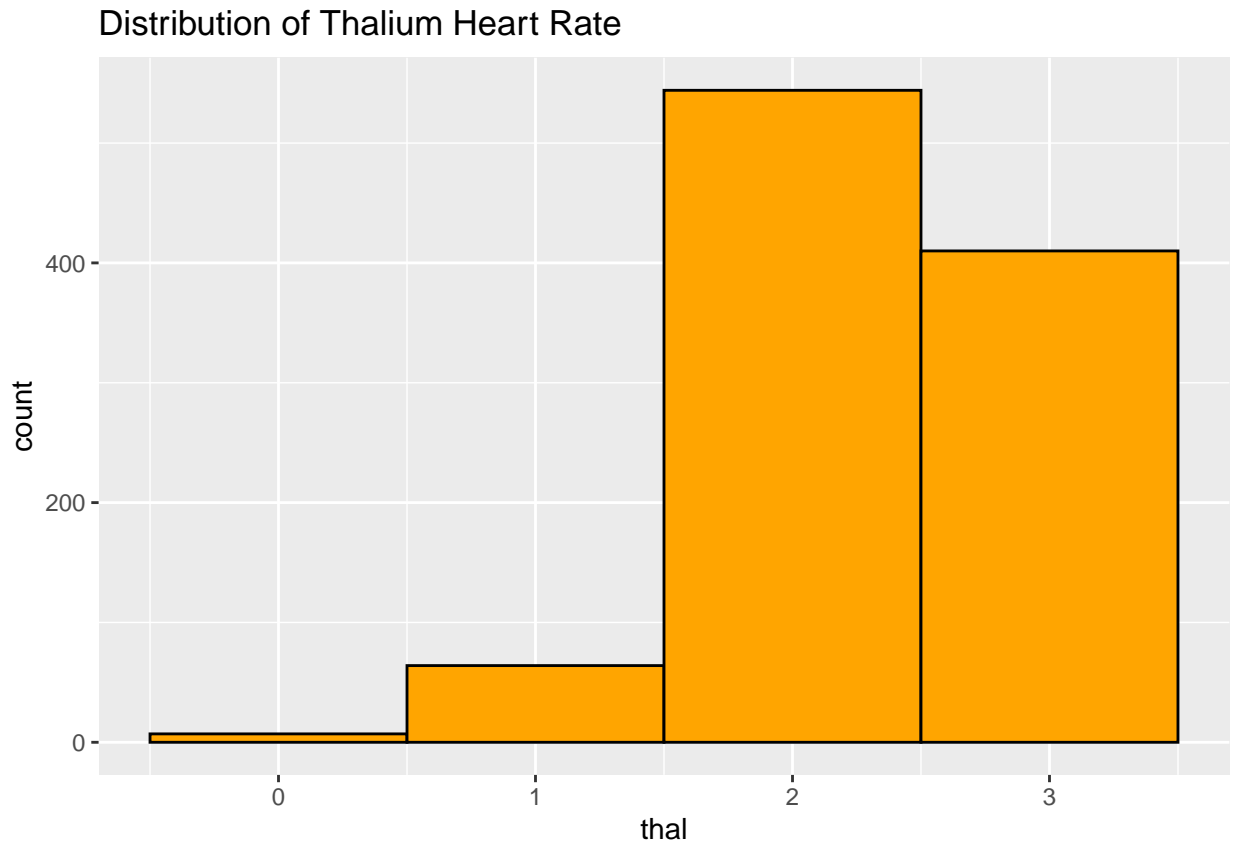
An upward slope may indicate a transient elevation of the ST segment, which can be a sign of various cardiac conditions, such as myocardial ischemia or infarction. A flat or horizontal slope it suggests a lack of significant deviation from the resting state. This is often interpreted as a normal response, which most patient had. A downward slope it suggests a deviation from the resting state, potentially indicating myocardial ischemia or other cardiac abnormalities. This downward slope may indicate insufficient blood flow to the heart muscle during exertion, which alot of patients had.

```
ggplot(heart, aes(x = ca)) +
  geom_histogram(binwidth = 1, fill = "lightgreen", color = "black") +
  ggtitle("Distribution of Major Vessels (0-3) Colored by Fluoroscopy")
```



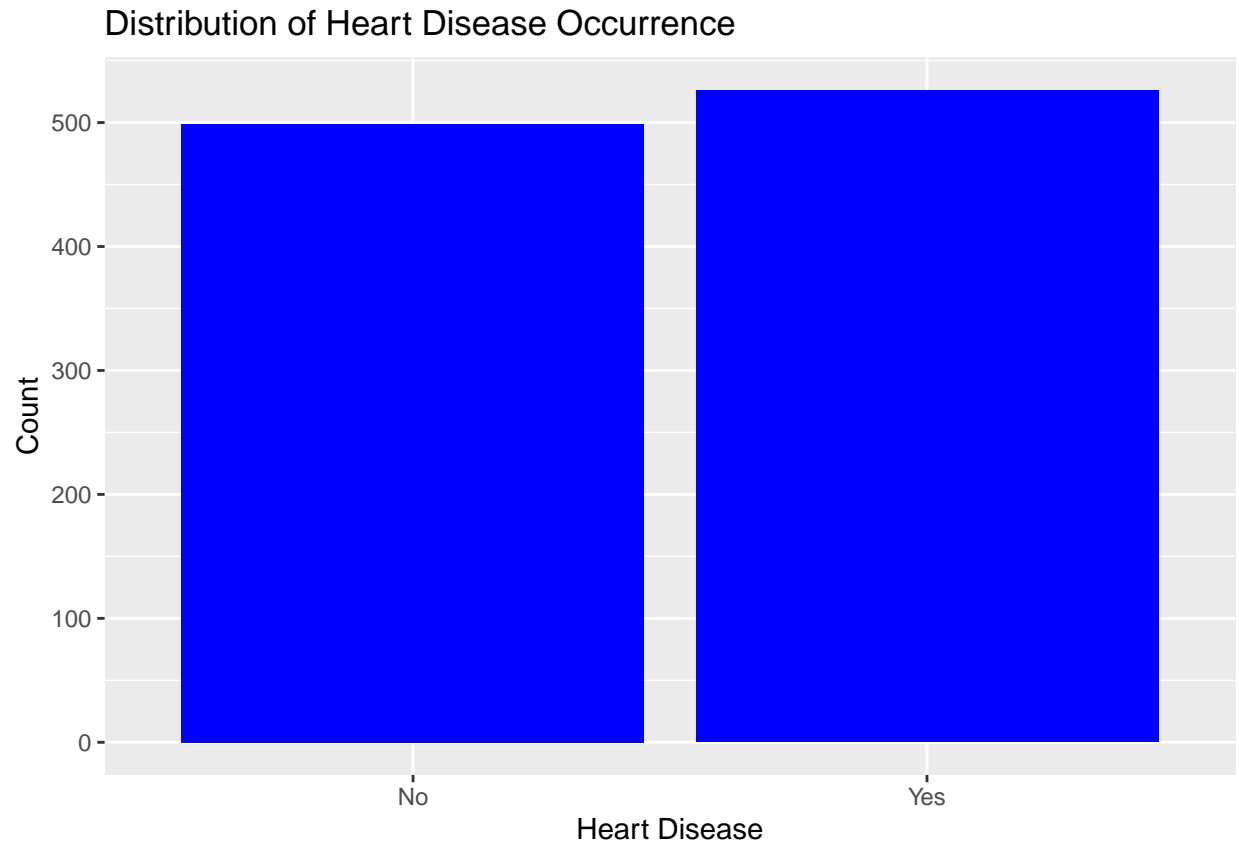
The values of “ca” range from 0 to 3, indicating the number of major vessels that were observed and colored during the fluoroscopy procedure. Which most patients did not have their major vessels observed.

```
ggplot(heart, aes(x = thal)) +  
  geom_histogram(binwidth = 1, fill = "orange", color = "black") +  
  ggtitle("Distribution of Thallium Heart Rate")
```



thal: A blood disorder called thalassemia. Most patient have normal blood flow. But is followed by patients that have reversible defect(a blood flow is observed but it is not normal)).

```
ggplot(heart, aes(x = factor(target, labels = c("No", "Yes")))) +  
  geom_bar(fill = "blue") +  
  ggtitle("Distribution of Heart Disease Occurrence") +  
  xlab("Heart Disease") +  
  ylab("Count")
```

Most of the patients have heart disease.

Comparing all attributes with target heart disease.

```
# Create a function to replace numeric values with labels
replace_labels <- function(data, attribute) {
  if (attribute == "sex") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("Female", "Male"))
  } else if (attribute == "cp") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("Typical Angina", "Atypical Angina", "Non-anginal"))
  } else if (attribute == "fbs") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("False", "True"))
  } else if (attribute == "restecg") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("Normal", "ST Abnormality", "LVH"))
  } else if (attribute == "exang") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("No", "Yes"))
  } else if (attribute == "slope") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("Up", "Flat", "Down"))
  } else if (attribute == "target") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("No Heart Disease", "Heart Disease"))
  } else if (nlevels(factor(data[[attribute]])) > 2) {
    data[[attribute]] <- factor(data[[attribute]])
  }
  return(data)
}
```

```

# Replace numeric values with labels for specified attributes
heart <- replace_labels(heart, "sex")
heart <- replace_labels(heart, "cp")
heart <- replace_labels(heart, "fbs")
heart <- replace_labels(heart, "restecg")
heart <- replace_labels(heart, "exang")
heart <- replace_labels(heart, "slope")
heart <- replace_labels(heart, "target") # Add this line to replace labels for the 'target' variable

# Create a function to generate bar plots with labels
plot_comparison_bar <- function(data, attribute) {
  # Replace numeric values with labels
  data <- replace_labels(data, attribute)

  # Plot bar chart
  p <- ggplot(data, aes(x = !!sym(attribute), fill = factor(target))) +
    geom_bar(position = "dodge", alpha = 0.8) +
    labs(title = paste("Heart Disease vs", attribute),
         x = attribute,
         y = "Count",
         fill = "Heart Disease") +
    theme_minimal()

  # Rotate x-axis labels by 90 degrees for specified attributes
  if (attribute %in% c("age", "trestbps", "chol", "thalach", "oldpeak")) {
    p <- p + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
  }

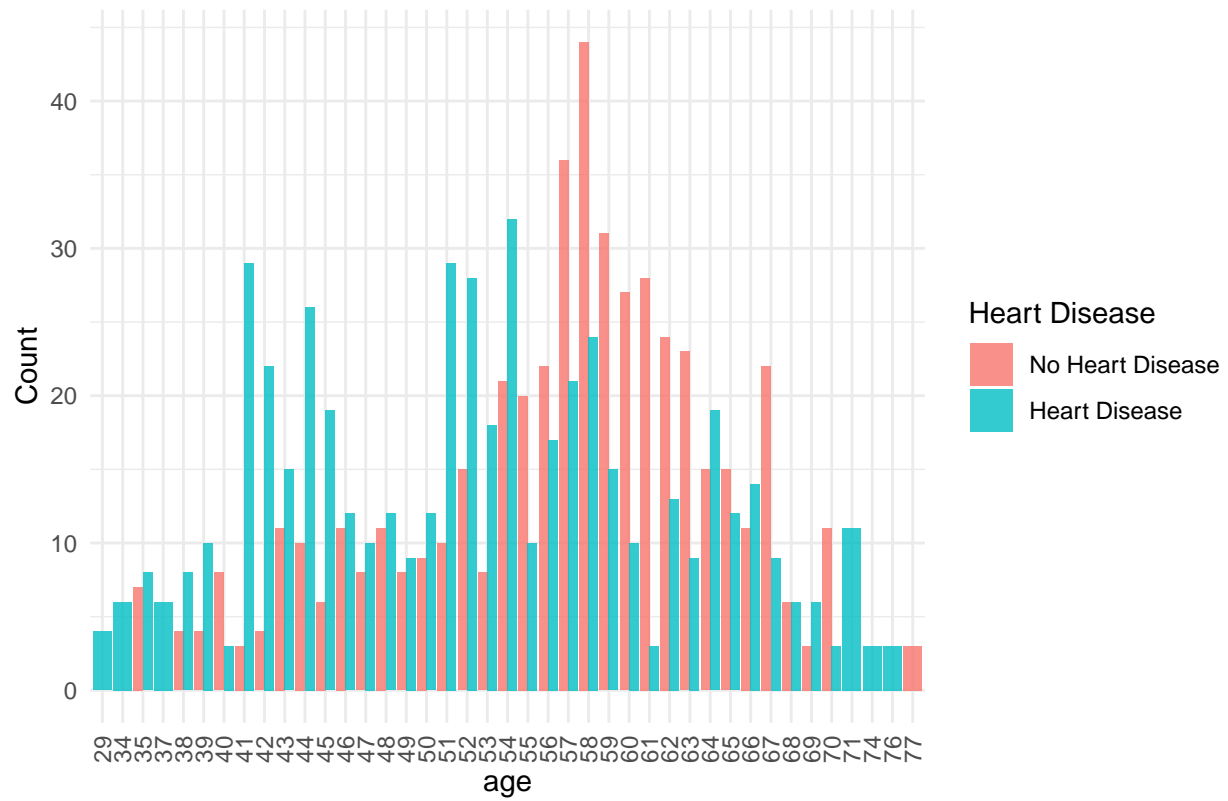
  # Adjust spacing for "chol" and "thalach" labels
  if (attribute %in% c("chol", "thalach")) {
    p <- p + theme(axis.text.x = element_text(angle = 45, vjust = 0.5, hjust=1))
  }

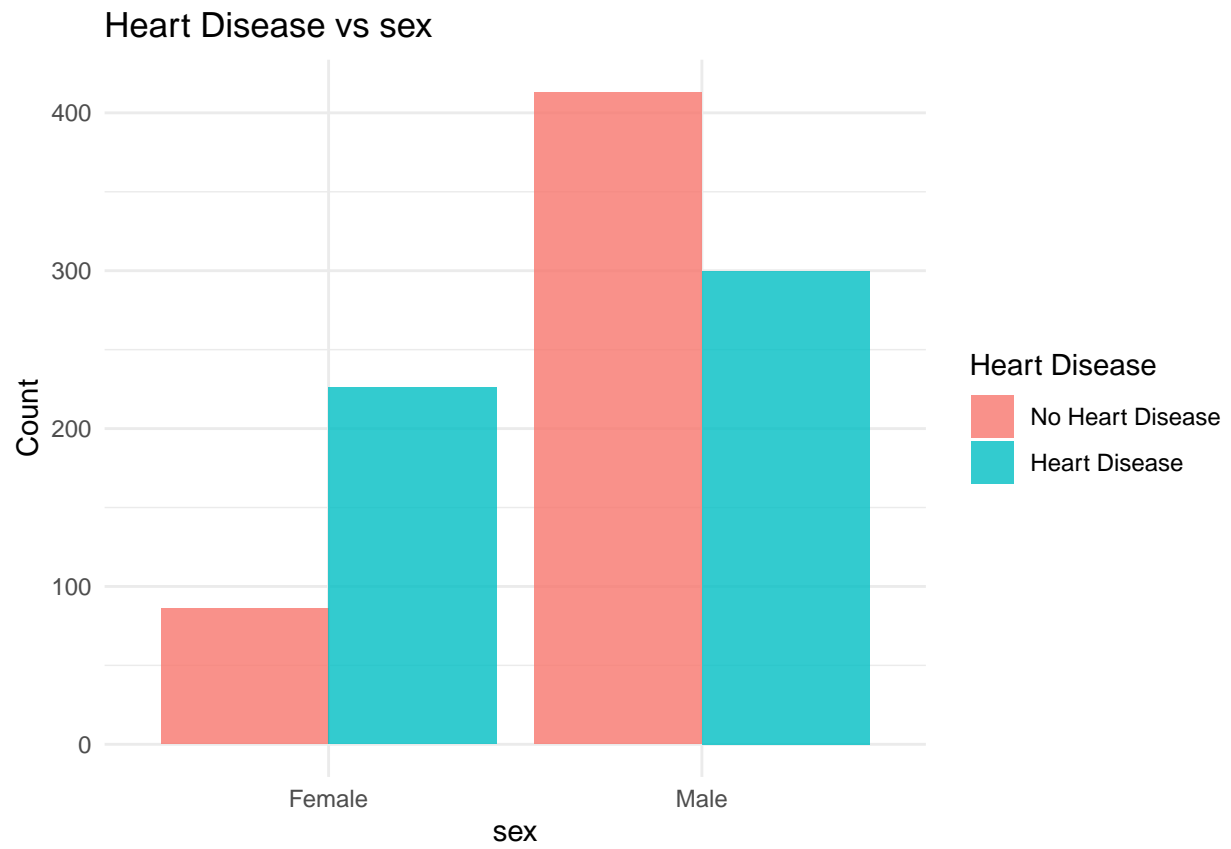
  return(p)
}

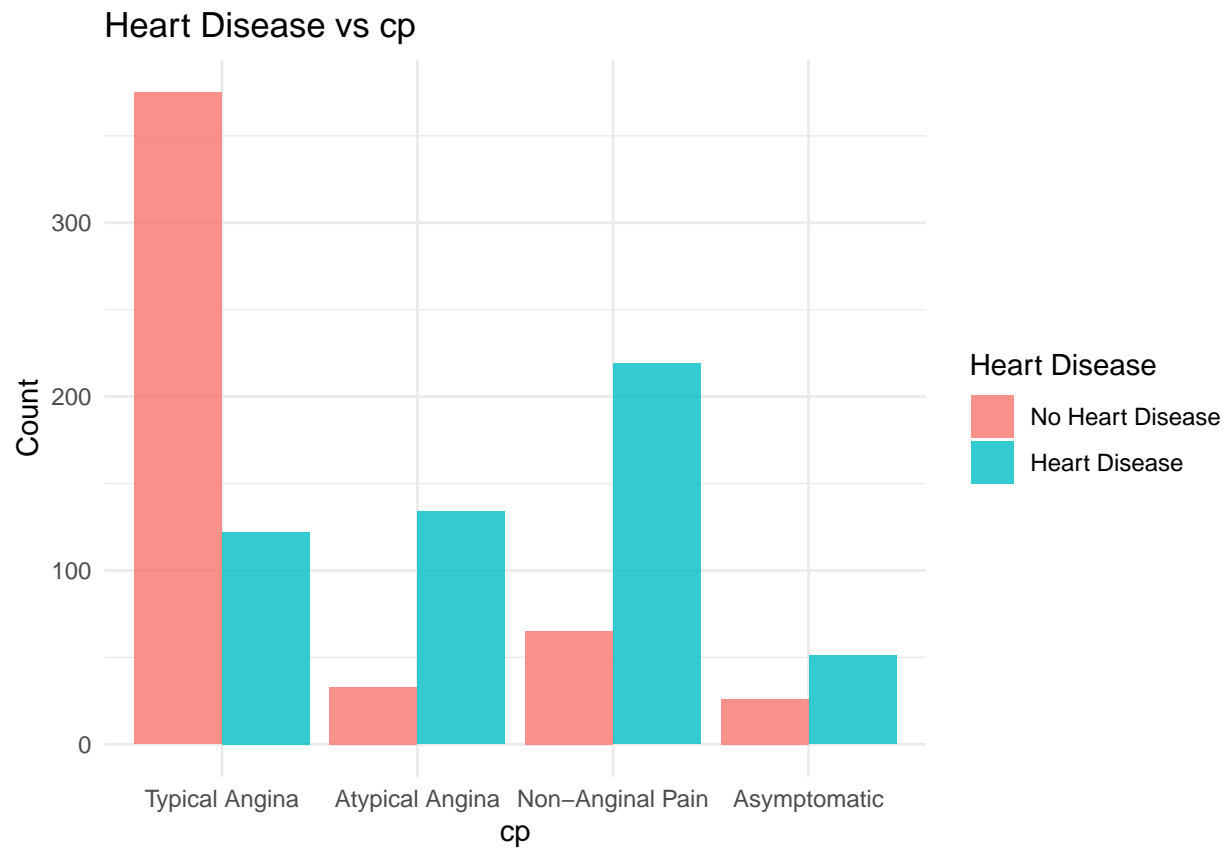
# Create bar plots for each attribute and save them individually
for (attr in names(heart)[names(heart) != "target"]) {
  plot <- plot_comparison_bar(heart, attr)
  ggsave(paste(attr, ".png", sep = ""), plot, width = 6, height = 6)
  print(plot)
}

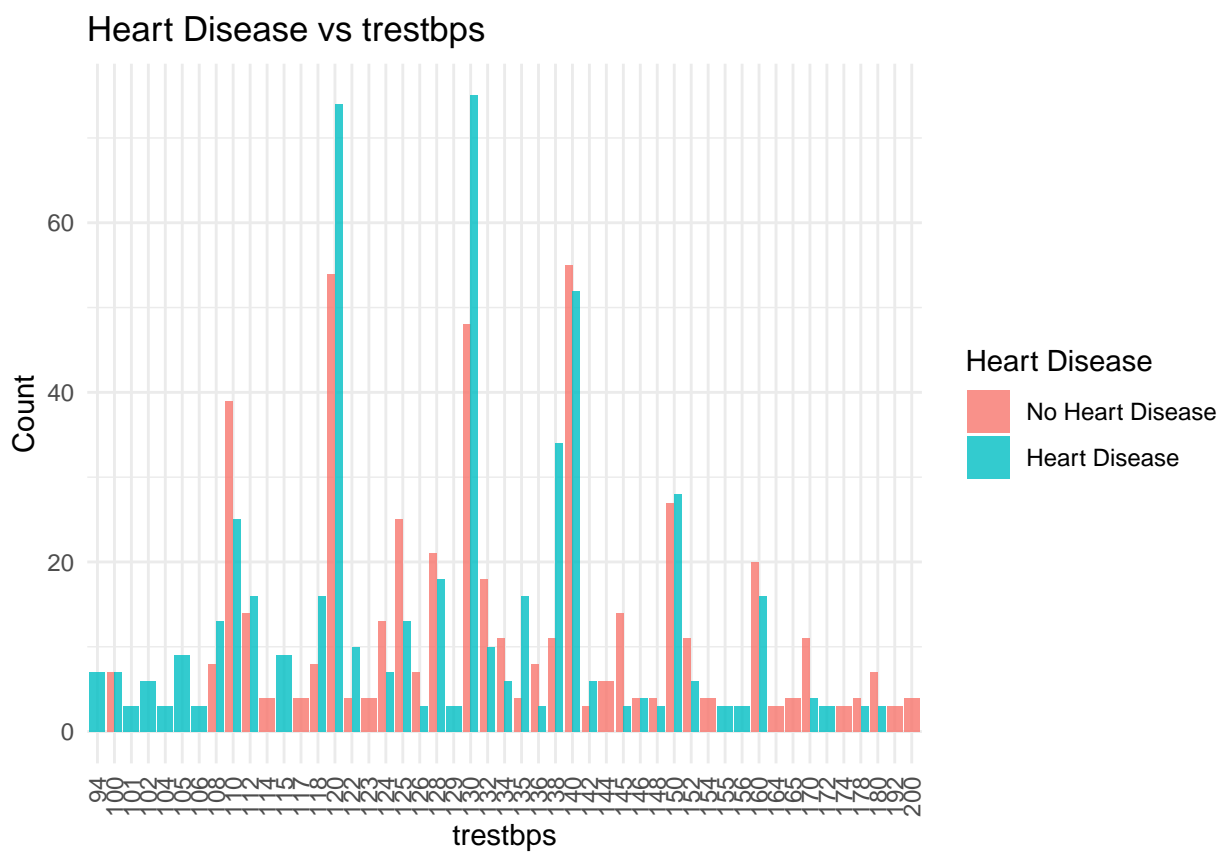
```

Heart Disease vs age

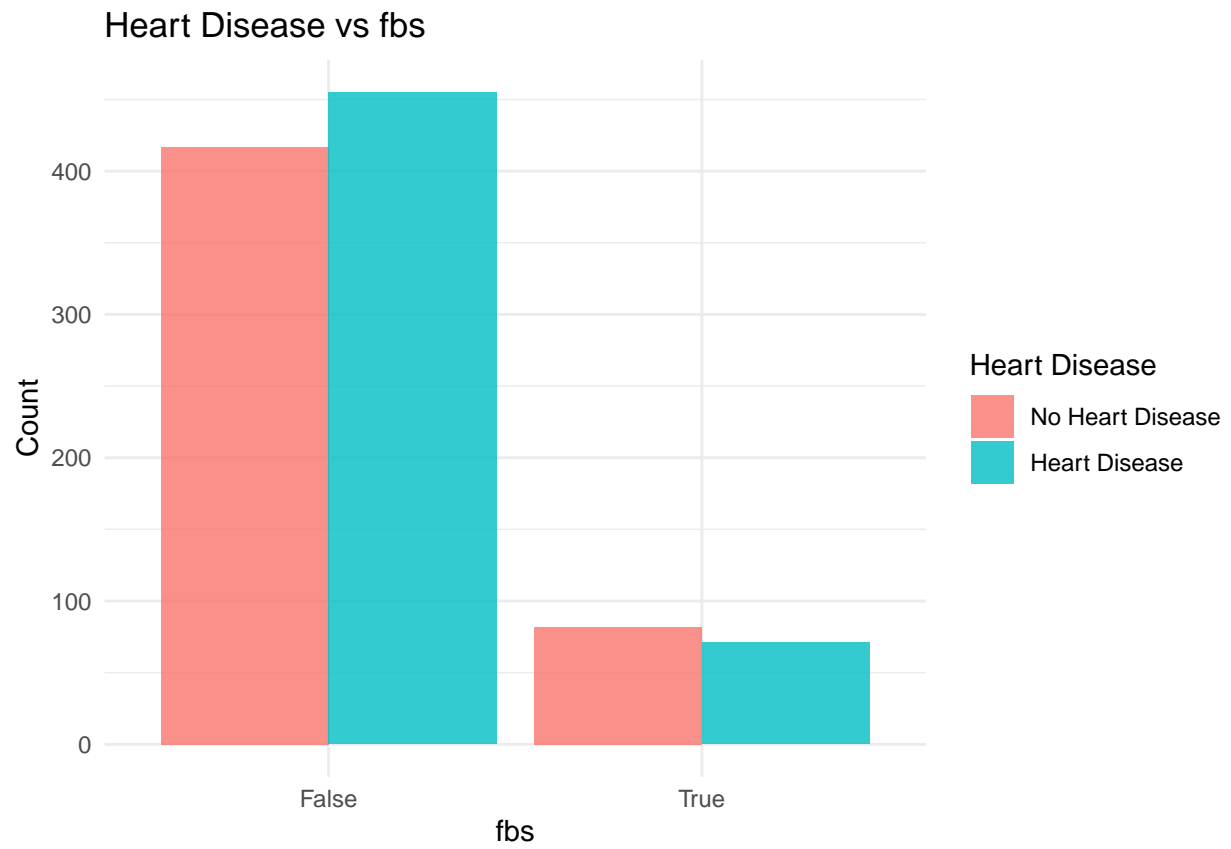


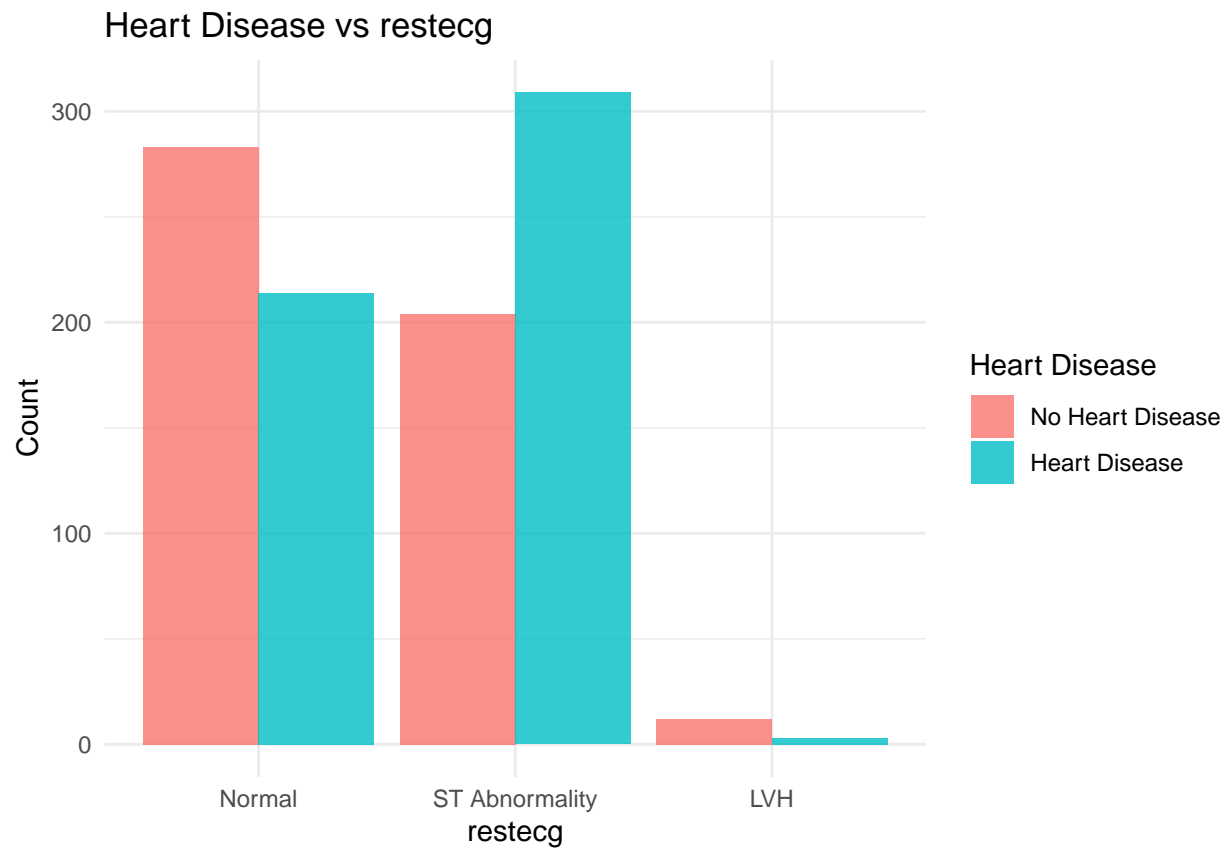


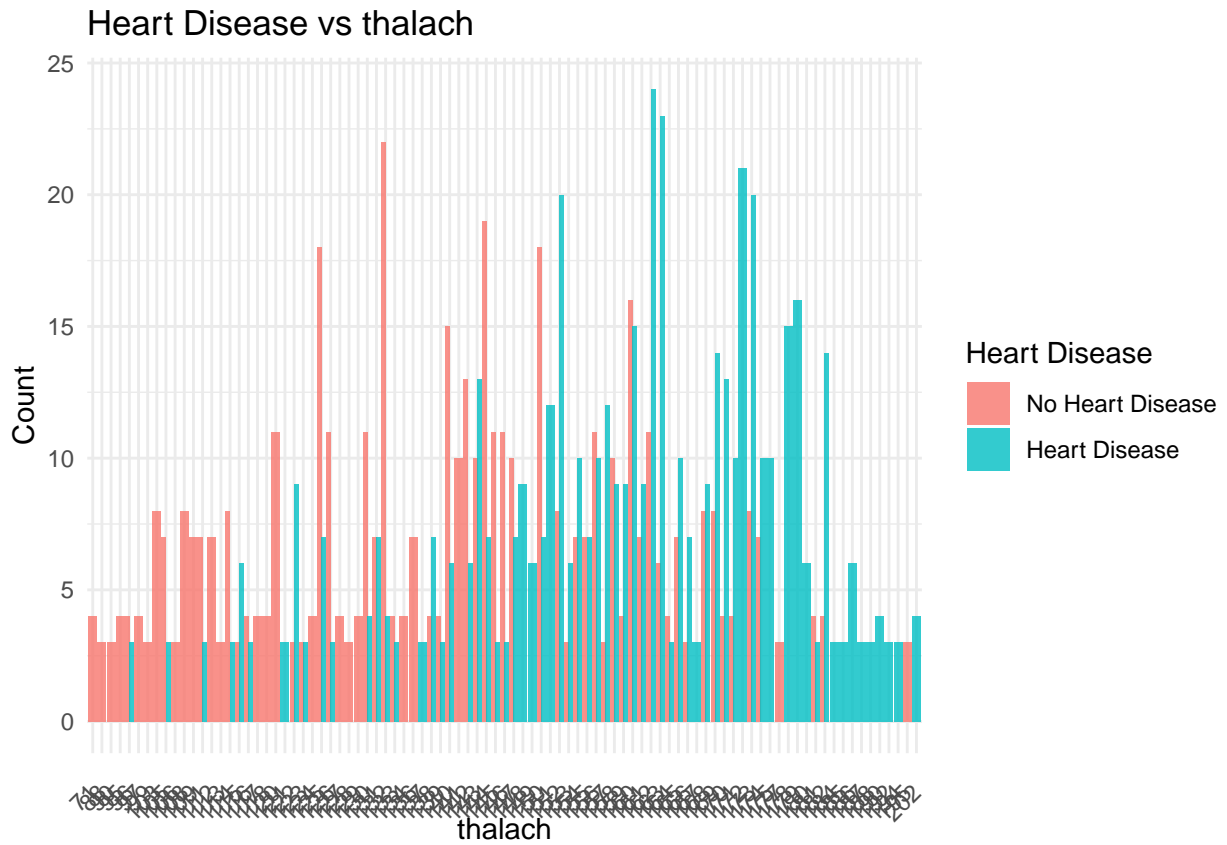


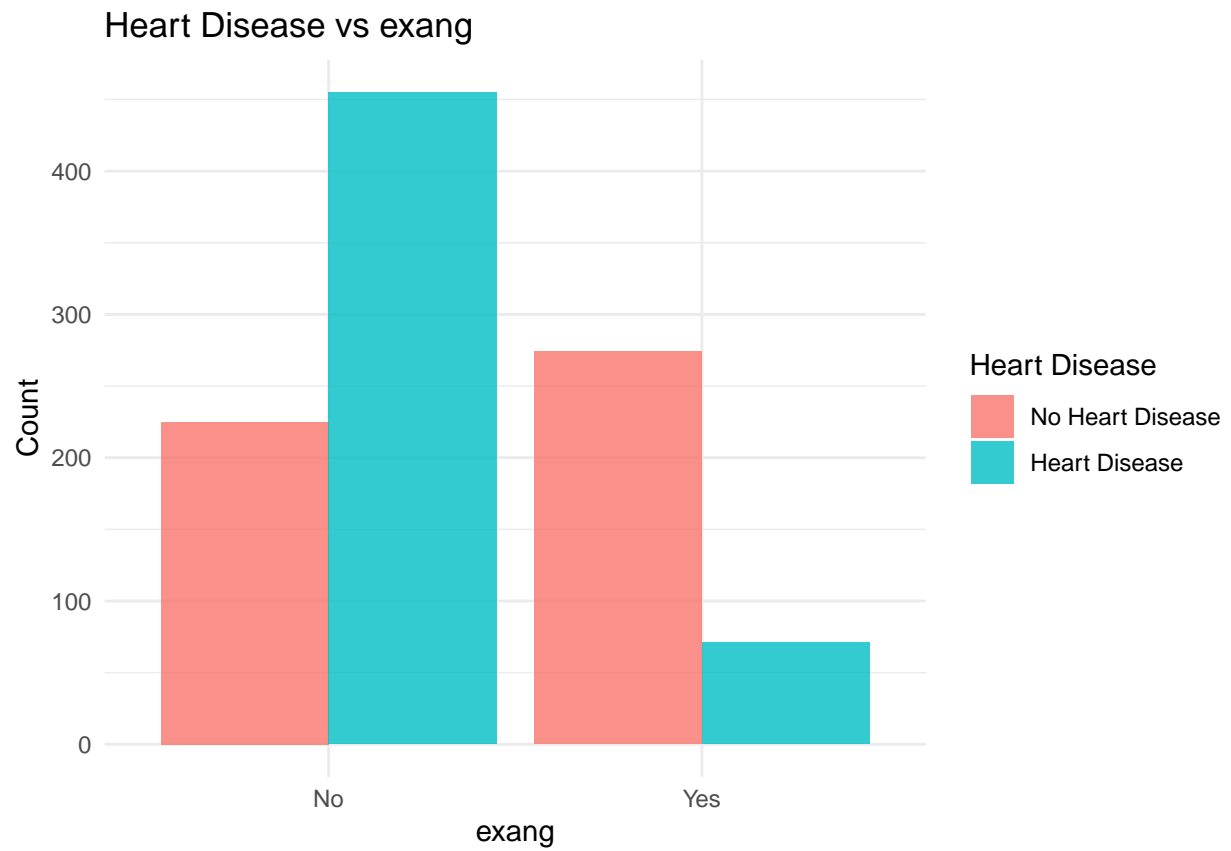


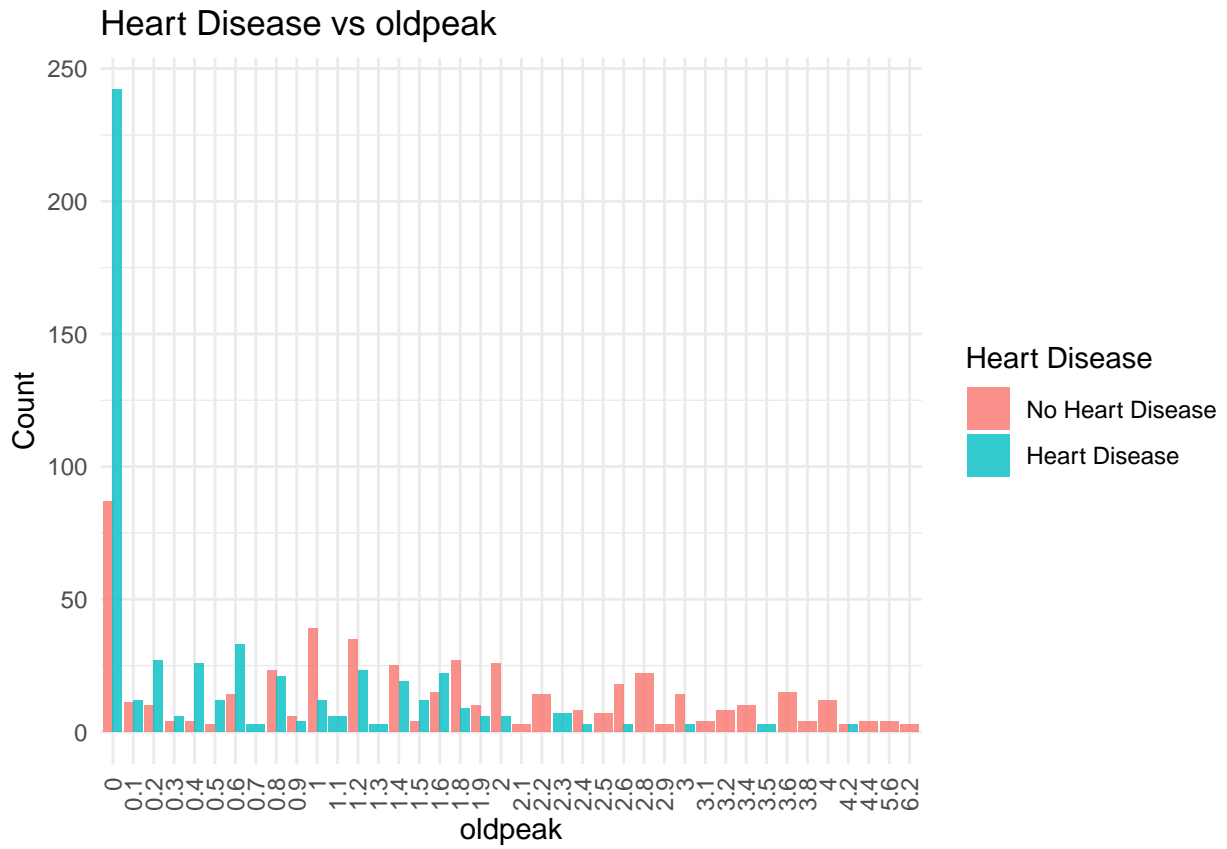


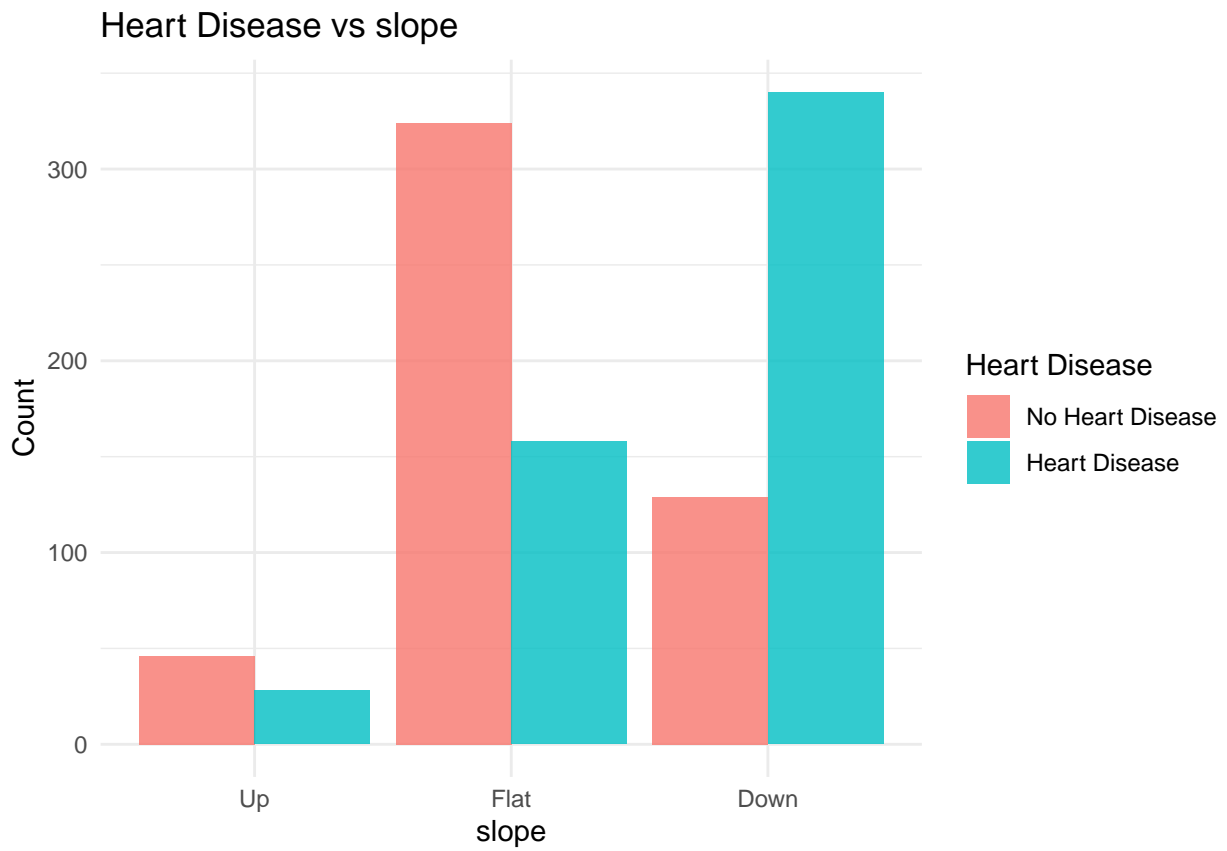


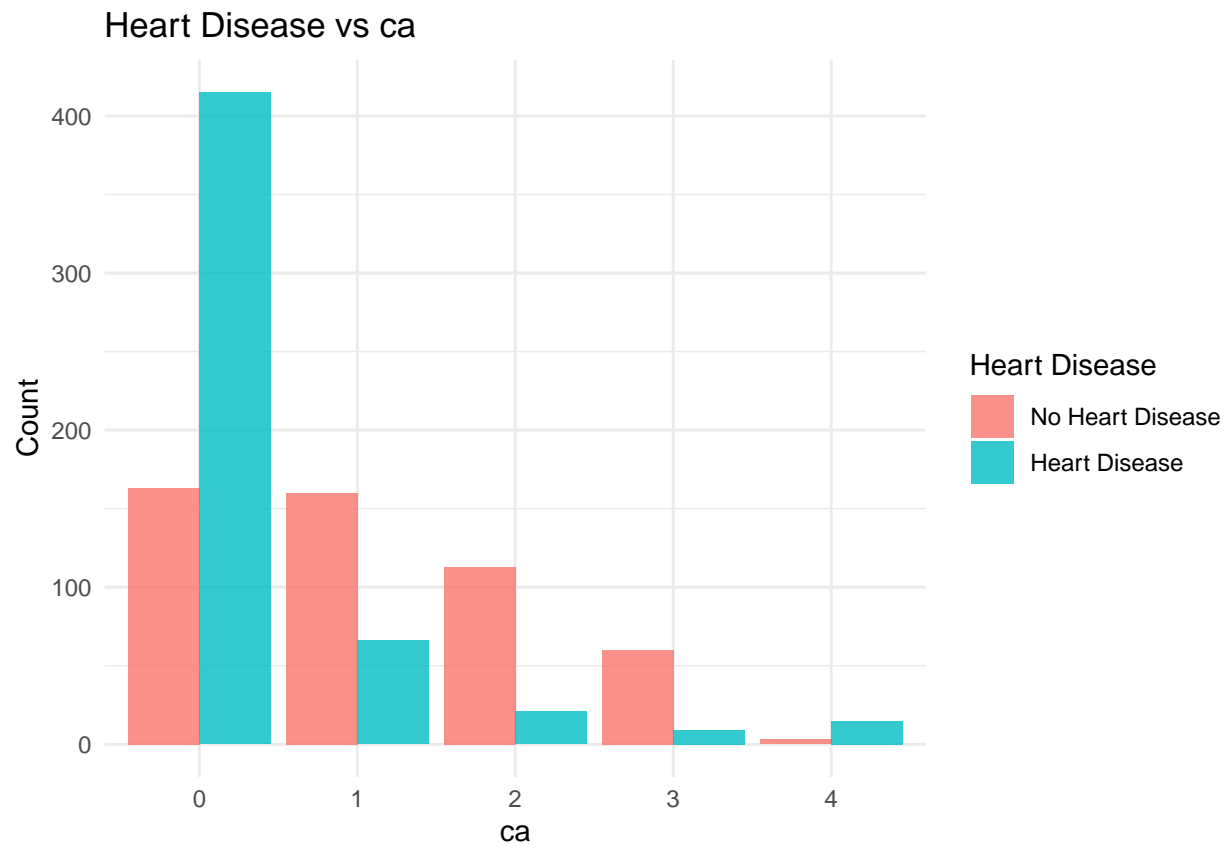


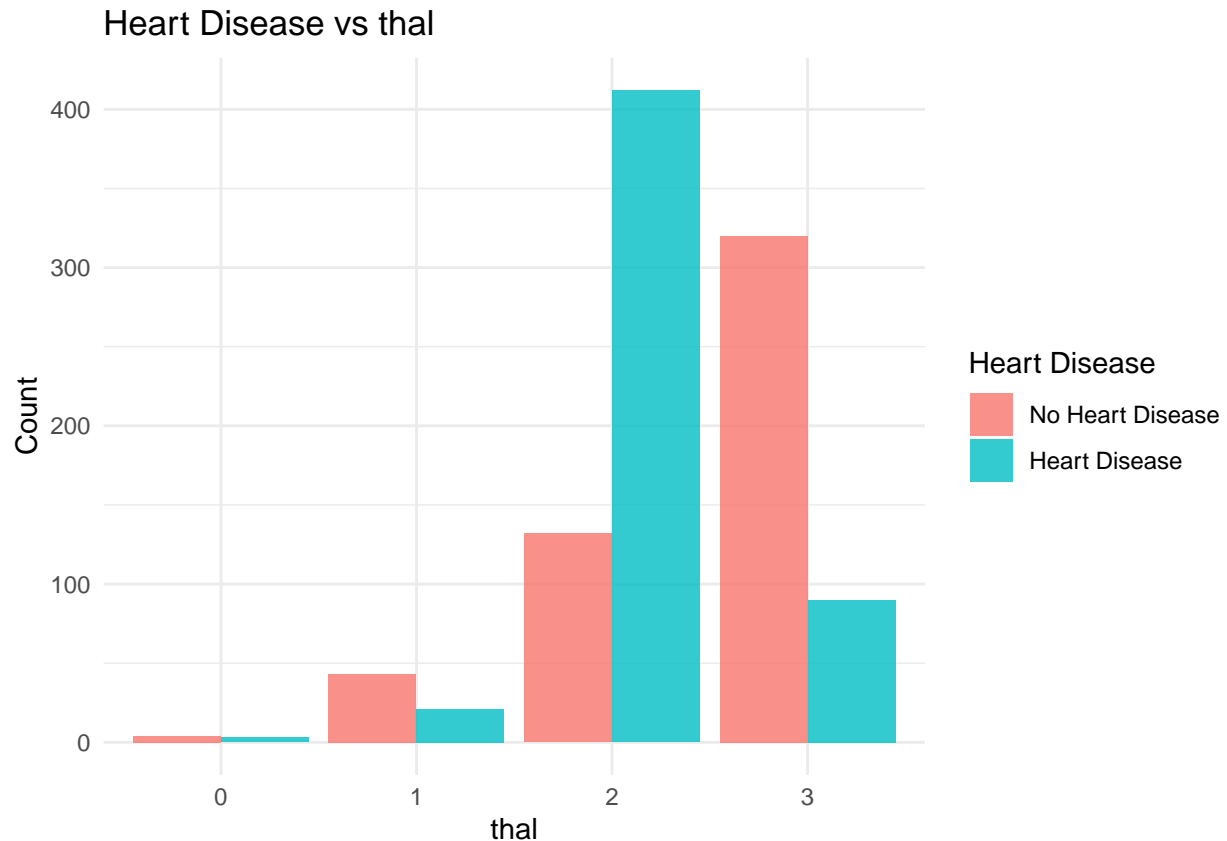












Modelling, Fitting, Prediction

Decision Tree

Decision Tree belongs to the family of Supervised Learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

Decision Tree Model 1

The Target is the output that I am interested in.

Split data into training and test sets Model 1 I will begin with a Cross Validation for our dataset first. The Cross Validation procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on unseen data. In this context, we will take 80% of our dataset to train our model and 20% of our dataset to test our prediction to see the model's performance.

```
heart1 <- heart %>%
  mutate_if(is.character, as.factor)

set.seed(417)
index <- sample(nrow(heart), size = nrow(heart)*0.80)
```

```
heart_train1 <- heart1[index,] #take 80%
heart_test1 <- heart1[-index,] #take 20%
```

```
round(prop.table(table(select(heart1, target), exclude = NULL)), 4) * 100
```

Check Class Balance Model 1

```
## target
## No Heart Disease    Heart Disease
##           48.68           51.32
```

```
round(prop.table(table(select(heart_train1, target), exclude = NULL)), 4) * 100
```

```
## target
## No Heart Disease    Heart Disease
##           47.07           52.93
```

```
round(prop.table(table(select(heart_test1, target), exclude = NULL)), 4) * 100
```

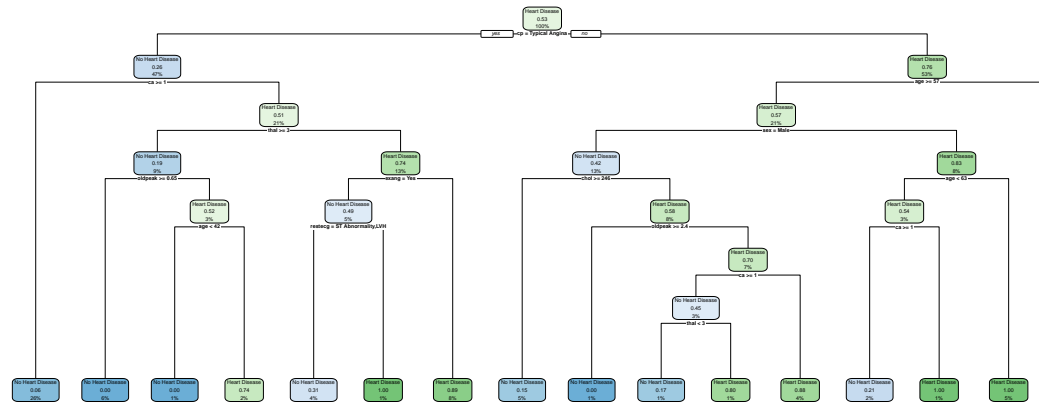
```
## target
## No Heart Disease    Heart Disease
##           55.12           44.88
```

The proportion of the class is balanced.

```
heart_mod1 <-
  rpart(
    target ~ .,
    method = "class",
    data = heart_train1
  )
```

Building the Model 1

```
rpart.plot(heart_mod1)
```

Evaluating the Model 1

This graph is telling me that a patient with a typical angina and age over 57 is 89% chance of having a heart disease. A patient male over 63 has 100% chance of having a heart disease. While a patient with with typical angina and greater than 1 major vessal observed during a fluoroscopy procedure, has 6% chance of not having a heard disease.

(I hope I read this graph right.)

```
heart_pred1 <- predict(heart_mod1, heart_test1, type = "class")
heart_pred_table1 <- table(heart_test1$target, heart_pred1)
heart_pred_table1
```

Prediction Model 1

```
##           heart_pred1
##           No Heart Disease Heart Disease
## No Heart Disease      102          11
## Heart Disease         4           88
```

Confusion matrix values

```
TP <- 88
TN <- 102
FP <- 11
FN <- 4
```

```

# Total instances
total <- TP + TN + FP + FN

# Accuracy
accuracy <- (TP + TN) / total

# Precision
precision <- TP / (TP + FP)

# Recall
recall <- TP / (TP + FN)

# F1 Score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the results
print(paste("Accuracy:", round(accuracy, 4)))

```

```
## [1] "Accuracy: 0.9268"
```

```
print(paste("Precision:", round(precision, 4)))
```

```
## [1] "Precision: 0.8889"
```

```
print(paste("Recall:", round(recall, 4)))
```

```
## [1] "Recall: 0.9565"
```

```
print(paste("F1 Score:", round(f1_score, 4)))
```

```
## [1] "F1 Score: 0.9215"
```

Accuracy: The proportion of correctly classified instances among all instances. In this case, it's 0.9268, indicating that approximately 92.68% of the instances were classified correctly.

Precision: The proportion of true positive predictions (correctly identified instances) out of all positive predictions (true positives + false positives). A high precision indicates a low false positive rate. Here, it's 0.8889, meaning that approximately 88.89% of the instances predicted as positive were actually positive.

Recall (Sensitivity): The proportion of true positive predictions out of all actual positive instances. A high recall indicates a low false negative rate. In this case, it's 0.9565, suggesting that approximately 95.65% of actual positive instances were correctly identified.

F1 Score: The harmonic mean of precision and recall. It balances between precision and recall, providing a single metric to evaluate the model's performance. A higher F1 score indicates better model performance. Here, it's 0.9215, reflecting the overall effectiveness of the model in capturing both precision and recall.

Overall, the model demonstrates strong performance across all metrics, with high accuracy, precision, recall, and F1 score, indicating effective classification of instances.

Decision Tree Model 2 - Switch Variables

I limited my variables to age, sex, and cholesterol because looking at the previous Decision Tree in Model 1 the patients that had the highest percentage of having heart disease was determined by age, sex, and their cholesterol level.

```
heart2 <- heart %>%
  select(
    age,
    sex,
    chol,
    target
  )

set.seed(417)
index <- sample(nrow(heart2), size = nrow(heart2)*0.80)
heart_train2 <- heart2[index,] #take 80%
heart_test2 <- heart2[-index,] #take 20%
```

Split data into training and test sets Model 2

```
round(prop.table(table(select(heart2, target), exclude = NULL)), 4) * 100
```

Check Class Balance Model 2

```
## target
## No Heart Disease    Heart Disease
##           48.68           51.32
```

```
round(prop.table(table(select(heart_train2, target), exclude = NULL)), 4) * 100
```

```
## target
## No Heart Disease    Heart Disease
##           47.07           52.93
```

```
round(prop.table(table(select(heart_test2, target), exclude = NULL)), 4) * 100
```

```
## target
## No Heart Disease    Heart Disease
##           55.12           44.88
```

The proportion of the class is balanced.

```

set.seed(417)
heart_mod2 <-
  rpart(
    target ~ .,
    method = "class",
    data = heart_train2
  )

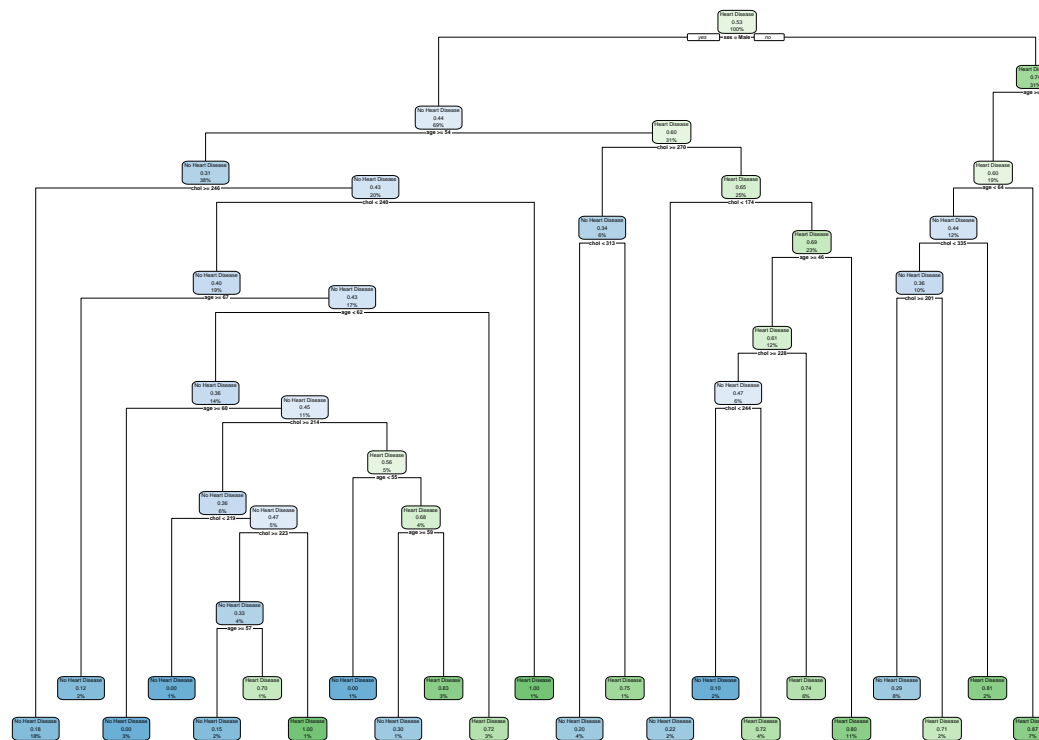
```

Building the Model 2

```

set.seed(417)
rpart.plot(heart_mod2)

```



Evaluating the Model 2

```

# Confusion matrix values
TP <- 73
TN <- 77
FP <- 36

```

```

FN <- 19

# Total instances
total <- TP + TN + FP + FN

# Accuracy
accuracy <- (TP + TN) / total

# Precision
precision <- TP / (TP + FP)

# Recall
recall <- TP / (TP + FN)

# F1 Score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the results
print(paste("Accuracy:", round(accuracy, 4)))

```

Prediction Model 2

```
## [1] "Accuracy: 0.7317"
```

```
print(paste("Precision:", round(precision, 4)))
```

```
## [1] "Precision: 0.6697"
```

```
print(paste("Recall:", round(recall, 4)))
```

```
## [1] "Recall: 0.7935"
```

```
print(paste("F1 Score:", round(f1_score, 4)))
```

```
## [1] "F1 Score: 0.7264"
```

The accuracy is 0.7317, indicating that approximately 73.17% of the predictions made by the model are correct.

The precision is 0.6697, indicating that approximately 66.97% of the instances predicted as positive are actually positive.

The recall is 0.7935, indicating that approximately 79.35% of the actual positive instances are correctly identified by the model.

The F1 score is 0.7264, indicating the overall effectiveness of the model in terms of both precision and recall.

This result demonstrates that while the model has relatively high recall, indicating its ability to correctly identify positive instances, it may have a lower precision, suggesting that it could improve in correctly identifying negative instances. Additionally, the F1 score shows the balance between precision and recall, providing a comprehensive evaluation of the model's performance.

Random Forrest

```
set.seed(1234)
dim(heart)
```

```
## [1] 1025  14
```

```
set.seed(417)
index <- sample(nrow(heart), size = nrow(heart)*0.80)
heart_train <- heart[index,] #take 80%
heart_test <- heart[-index,] #take 20%
```

Split Data into Training and Test sets Random Forrest

```
# Fitting Random Forest to the train dataset
set.seed(120) # Setting seed
rf.heart = randomForest(x = heart_train[-14],
                        y = heart_train$target,
                        ntree = 500)

rf.heart
```

Model Classifier_RF

```
##
## Call:
## randomForest(x = heart_train[-14], y = heart_train$target, ntree = 500)
##              Type of random forest: classification
##              Number of trees: 500
## No. of variables tried at each split: 3
##
##              OOB estimate of  error rate: 0.24%
## Confusion matrix:
##              No Heart Disease Heart Disease class.error
## No Heart Disease      385           1 0.002590674
## Heart Disease         1           433 0.002304147
```

The random forest model, constructed with 500 trees and considering 3 variables at each split, achieved an out-of-bag (OOB) error rate of 0.24%. The OOB estimate of error rate is a reliable measure, calculated using samples not included in the bootstrap sample of each tree. This provides an unbiased estimate of the model's performance on unseen data.

The confusion matrix offers a detailed view of the model's predictions on the training dataset. The "class.error" column indicates the error rate for each class, representing the proportion of misclassified instances. In this instance, the error rate for the "No Heart Disease" class is approximately 0.26%, while for the "Heart Disease" class, it's approximately 0.23%. These low error rates signify the model's excellent performance in accurately classifying instances into their respective categories.

```

y_pred <- predict(rf.heart, newdata = heart_test[, -14])

# Confusion Matrix
confusion_mtx <- table(heart_test[, 14], y_pred)
confusion_mtx

```

Confusion Matrix

```

##                y_pred
##                No Heart Disease Heart Disease
## No Heart Disease          113             0
## Heart Disease              0             92

```

In the provided example:

The model correctly predicted 113 instances of “No Heart Disease” when the actual class was also “No Heart Disease.” It correctly predicted 92 instances of “Heart Disease” when the actual class was also “Heart Disease.” There were no instances where the model incorrectly classified “No Heart Disease” as “Heart Disease” (0 false positives). Similarly, there were no instances where the model incorrectly classified “Heart Disease” as “No Heart Disease” (0 false negatives). Overall, the model appears to have made accurate predictions on the test dataset based on the provided confusion matrix.

```

# Confusion matrix values
TP <- 92
TN <- 113
FP <- 0
FN <- 0

# Total instances
total <- TP + TN + FP + FN

# Accuracy
accuracy <- (TP + TN) / total

# Precision
precision <- TP / (TP + FP)

# Recall
recall <- TP / (TP + FN)

# F1 Score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the results
print(paste("Accuracy:", round(accuracy, 4)))

```

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

```
print(paste("Precision:", round(precision, 4)))
```

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

```
print(paste("Recall:", round(recall, 4)))
```

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

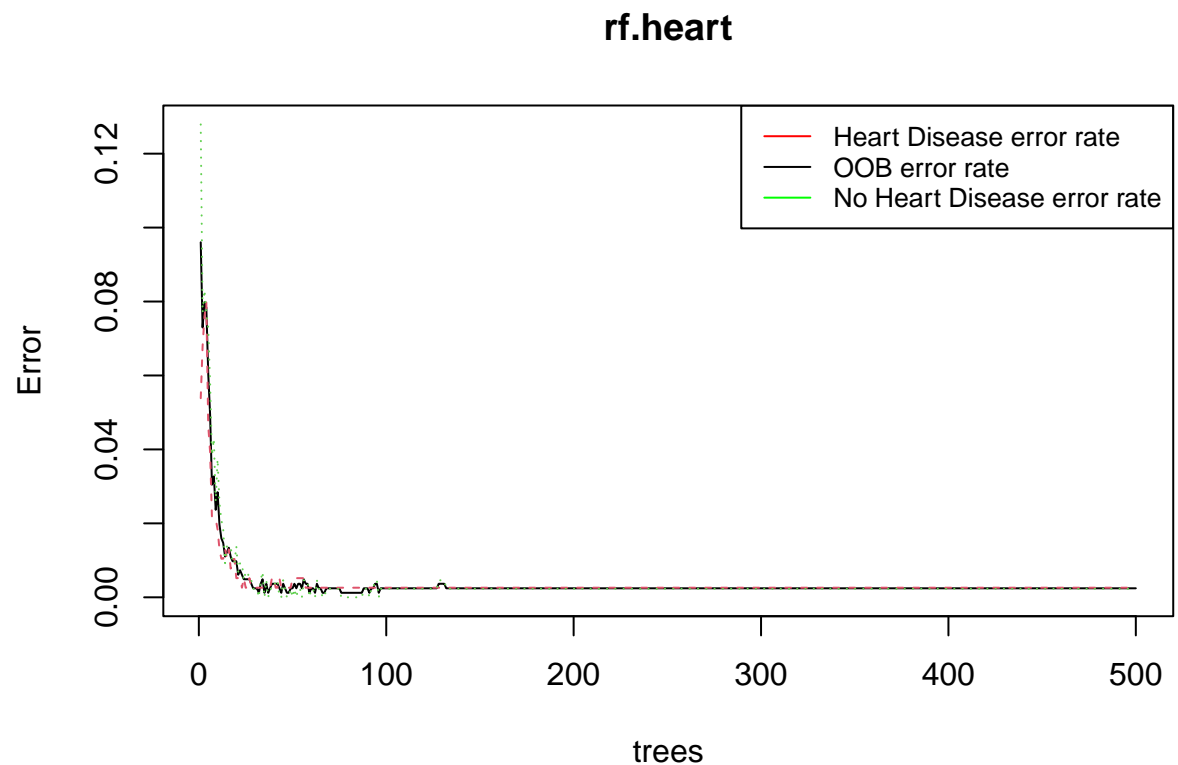
```
print(paste("F1 Score:", round(f1_score, 4)))
```

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

The predictive accuracy is 100%.

The confusion matrix and all performance shows that the model achieved perfect accuracy on the test dataset. All instances were correctly classified, with no false positives or false negatives. This indicates that the model performed very well on this particular test dataset.

```
plot(rf.heart)
legend("topright", legend = c("Heart Disease error rate", "OOB error rate", "No Heart Disease error rate",
                              "Heart Disease error rate", "OOB error rate", "No Heart Disease error rate"),
      col = c("red", "black", "green", "red", "black", "green"), lty = 1, cex = 0.8)
```



Plot of Model


```

set.seed(417)
importance_df <- importance(rf.heart)
importance_sorted <- data.frame(
  Variable = rownames(importance_df),
  MeanDecreaseGini = importance_df[, 1]
)
importance_sorted <- importance_sorted[order(importance_sorted$MeanDecreaseGini, decreasing = TRUE), , ]
rownames(importance_sorted) <- NULL # Reset row names

print(importance_sorted)

```

Important Features

##	Variable	MeanDecreaseGini
## 1	cp	51.772016
## 2	ca	50.089382
## 3	thalach	46.092970
## 4	oldpeak	44.038777
## 5	thal	43.284840
## 6	age	37.447212
## 7	chol	33.334145
## 8	trestbps	28.795034
## 9	exang	25.272639
## 10	slope	18.226297
## 11	sex	13.497369
## 12	restecg	7.753257
## 13	fbs	3.807515

This result shows the importance of predictor variables in a random forest model, measured by Mean Decrease in Gini impurity.

cp (Chest Pain Type): This variable has the highest importance, with a Mean Decrease Gini of 51.772016. It suggests that splitting based on this variable leads to the most significant reduction in impurity across all trees in the forest.

ca (Number of major vessels colored by fluoroscopy): Following cp, ca is the next most important predictor, with a Mean Decrease Gini of 50.089382.

thalach (Maximum Heart Rate achieved): thalach is the third most important predictor, with a Mean Decrease Gini of 46.092970.

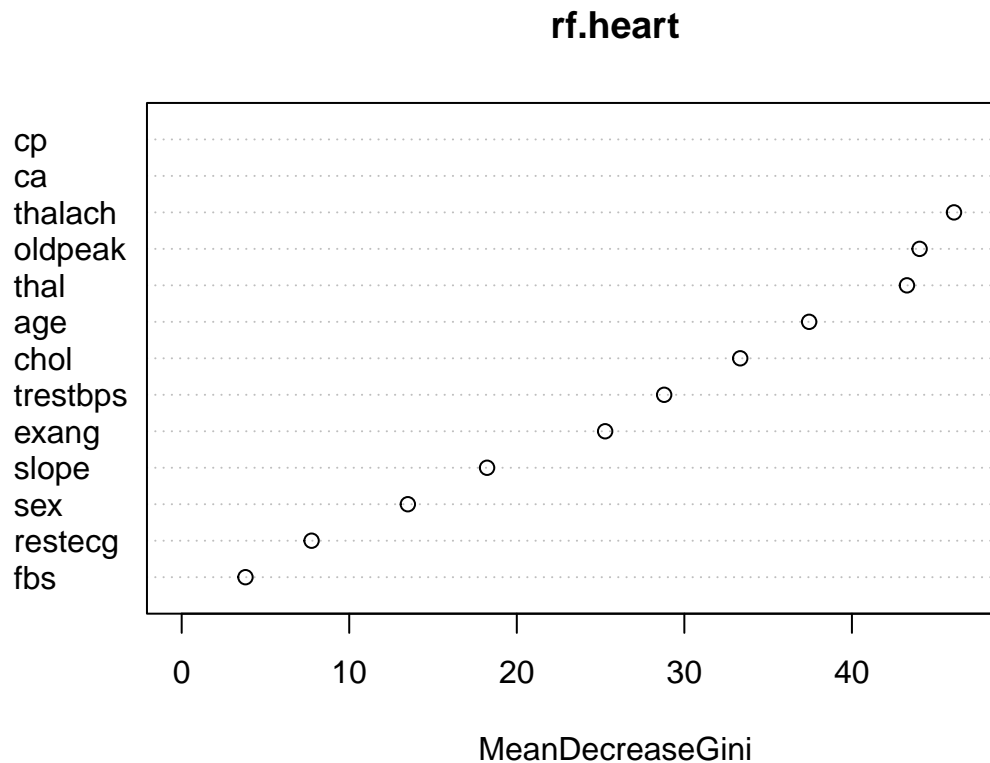
oldpeak (ST Depression induced by exercise relative to rest): oldpeak ranks fourth in importance, with a Mean Decrease Gini of 44.038777.

thal (Thalassemia): Thal comes next in importance, with a Mean Decrease Gini of 43.284840.

The rest of the variables follow in descending order of importance, each contributing to the model's predictive power based on their respective Mean Decrease Gini values.

In summary, feature importance scores provide insights into which features are most influential in the model's decision-making process. Higher importance scores indicate more significant contributions to the model's predictive performance.

```
# Variable importance plot
varImpPlot(rf.heart)
```



Plot of Important Feature

Optimization by Tree Numbers

Increasing the number of trees in a random forest from 500 to 1000 can potentially improve the model's performance in several ways:

Improved Generalization: With more trees, the random forest model has more opportunities to learn from the training data and capture complex patterns in the data. This can lead to better generalization to unseen data and potentially reduce overfitting, especially if the initial model was underfitting.

Better Accuracy: Increasing the number of trees allows the model to make more consensual decisions, as predictions are aggregated over a larger number of trees. This can lead to more accurate predictions, especially for cases where individual trees might make errors.

Increased Stability: Adding more trees can increase the stability of the model's predictions. With a larger ensemble of trees, the random forest model becomes less sensitive to variations in the training data and can provide more consistent predictions.

Improved Robustness: A larger number of trees can enhance the robustness of the model to noise or outliers in the data. By averaging predictions from more trees, the impact of individual noisy observations on the overall predictions is reduced.

However, it's essential to consider the trade-offs of increasing the number of trees. Training and predicting with a larger ensemble can be computationally expensive and may require more memory and processing

power. Additionally, beyond a certain point, increasing the number of trees may lead to diminishing returns in terms of model performance improvement.

In summary, optimizing a random forest by increasing the number of trees can lead to better generalization, accuracy, stability, and robustness of the model, but it's essential to balance these benefits with computational constraints and the law of diminishing returns.

```
set.seed(417)
heart$target = as.factor(heart$target)
train = sample(1:nrow(heart), 500)
rf.heart1000 = randomForest(target ~ ., data = heart, subset = train, ntree = 1000)
rf.heart1000
```

Model classifier_RF with Optimization

```
##
## Call:
## randomForest(formula = target ~ ., data = heart, ntree = 1000,      subset = train)
##               Type of random forest: classification
##               Number of trees: 1000
## No. of variables tried at each split: 3
##
##           OOB estimate of  error rate: 3.8%
## Confusion matrix:
##               No Heart Disease Heart Disease class.error
## No Heart Disease           213           12  0.05333333
## Heart Disease              7           268  0.02545455
```

The out-of-bag (OOB) estimate of the error rate, which is an estimate of the classification error rate of the random forest model calculated using the out-of-bag samples. In this case, the OOB estimate of the error rate is 3.8%.

The class.error column in the confusion matrix provides the error rate for each class. In this case, the error rate for predicting “No Heart Disease” instances is approximately 5.33%, and for predicting “Heart Disease” instances is approximately 2.55%.

Overall, the model appears to have performed relatively well, with low error rates for both classes. However, it's important to further evaluate the model's performance using additional metrics such as accuracy, precision, recall, and F1 score, and to validate it on unseen data to assess its generalization ability.

```
set.seed(417)
y_pred <- predict(rf.heart1000, newdata = heart_test[, -14])

# Confusion Matrix
confusion_mtx1000 <- table(heart_test[, 14], y_pred)
confusion_mtx1000
```

Confusion Matrix with Optimization

```
##               y_pred
##           No Heart Disease Heart Disease
## No Heart Disease         111          2
## Heart Disease           2          90
```

```
# Confusion matrix values
TP <- 90
TN <- 111
FP <- 2
FN <- 2

# Total instances
total <- TP + TN + FP + FN

# Accuracy
accuracy <- (TP + TN) / total

# Precision
precision <- TP / (TP + FP)

# Recall
recall <- TP / (TP + FN)

# F1 Score
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print the results
print(paste("Accuracy:", round(accuracy, 4)))
```

```
## [1] "Accuracy: 0.9805"
```

```
print(paste("Precision:", round(precision, 4)))
```

```
## [1] "Precision: 0.9783"
```

```
print(paste("Recall:", round(recall, 4)))
```

```
## [1] "Recall: 0.9783"
```

```
print(paste("F1 Score:", round(f1_score, 4)))
```

```
## [1] "F1 Score: 0.9783"
```

The results indicate strong performance across all metrics:

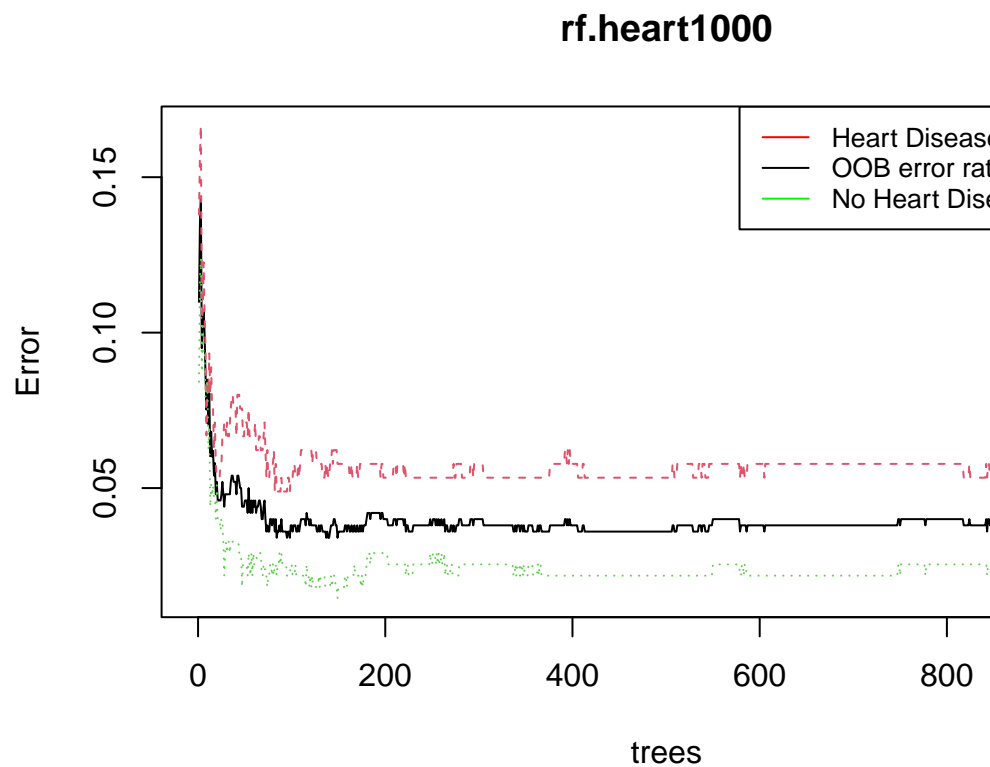
Accuracy: An accuracy of 0.9805 suggests that approximately 98.05% of instances in the dataset were correctly classified by the model.

Precision: With a precision of 0.9783, around 97.83% of instances classified as positive (Heart Disease) were indeed positive.

Recall: The recall score of 0.9783 indicates that approximately 97.83% of actual positive instances (Heart Disease) were correctly identified by the model.

F1 Score: The F1 Score, being the harmonic mean of precision and recall, is also 0.9783. This suggests a high balance between precision and recall, indicating strong overall performance.

```
set.seed(417)
plot(rf.heart1000)
legend("topright", legend = c("Heart Disease error rate", "OOB error rate", "No Heart Disease error rate"),
      col = c("red", "black", "green"), lty = 1, cex = 0.8)
```



Plot of Model with Optimization

```
set.seed(417)
importance_df1000 <- importance(rf.heart1000)
importance_sorted1000 <- data.frame(
  Variable = rownames(importance_df1000),
  MeanDecreaseGini = importance_df1000[, 1]
)
importance_sorted1000 <- importance_sorted1000[order(importance_sorted1000$MeanDecreaseGini, decreasing = TRUE), ]
rownames(importance_sorted1000) <- NULL # Reset row names
print(importance_sorted1000)
```

Important Features with Optimization

```
## Variable MeanDecreaseGini
## 1 cp 35.944511
```

## 2	oldpeak	34.051488
## 3	thalach	24.564869
## 4	ca	23.766305
## 5	thal	22.338299
## 6	age	19.262623
## 7	chol	19.171335
## 8	trestbps	18.021224
## 9	exang	17.917996
## 10	slope	12.724322
## 11	sex	9.642634
## 12	restecg	4.489032
## 13	fbs	1.944607

The result shows the importance of each predictor variable in the random forest model, measured by Mean Decrease in Gini impurity.

cp (Chest Pain Type): This variable has the highest importance, with a Mean Decrease Gini of 35.944511. It indicates that splitting based on this variable provides the most significant reduction in impurity across all trees in the forest.

oldpeak (ST Depression induced by exercise relative to rest): Following cp, oldpeak is the next most important predictor, with a Mean Decrease Gini of 34.051488.

thalach (Maximum Heart Rate achieved): thalach is the third most important predictor, with a Mean Decrease Gini of 24.564869.

ca (Number of major vessels colored by fluoroscopy): ca ranks fourth in importance, with a Mean Decrease Gini of 23.766305.

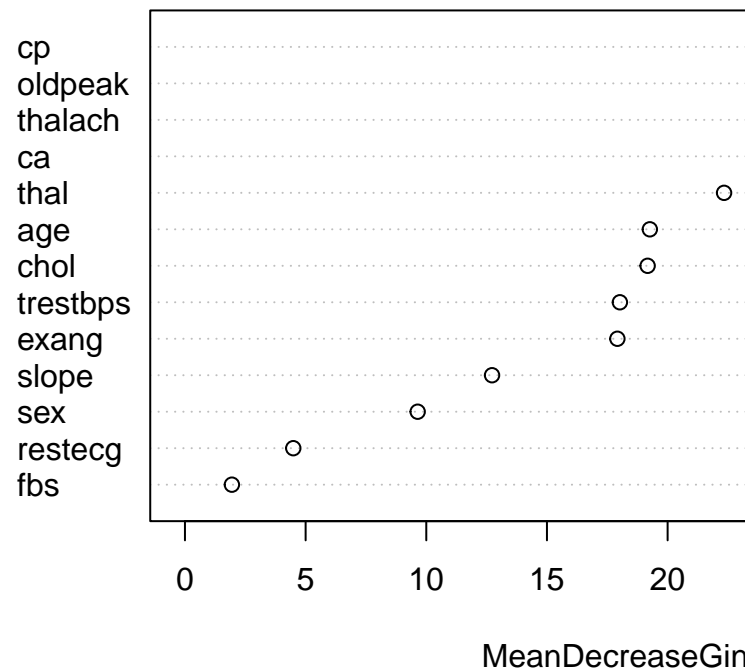
thal (Thalassemia): Thal comes next in importance, with a Mean Decrease Gini of 22.338299.

The rest of the variables follow in descending order of importance, each contributing to the model's predictive power based on their respective Mean Decrease Gini values.

In summary, feature importance scores provide insights into which features are most influential in the model's decision-making process. Higher importance scores indicate more significant contributions to the model's predictive performance.

```
# Variable importance plot
varImpPlot(rf.heart1000)
```

rf.heart1000



Plot of Important Feature with Optimization

Conclusion

Comparison of All 4 Models

```
# Create a data frame for each model's results
model1 <- c("Decision Tree Model 1", 0.9268, 0.8889, 0.9565, 0.9215)
model2 <- c("Decision Tree Model 2", 0.7317, 0.6697, 0.7935, 0.7264)
rf_500 <- c("Random Forest 500 Trees", 1, 1, 1, 1)
rf_1000 <- c("Random Forest 1000 Trees", 0.9805, 0.9783, 0.9783, 0.9783)

# Combine results into a data frame
comparison <- data.frame(Model = c(model1[1], model2[1], rf_500[1], rf_1000[1]),
                          Accuracy = c(model1[2], model2[2], rf_500[2], rf_1000[2]),
                          Precision = c(model1[3], model2[3], rf_500[3], rf_1000[3]),
                          Recall = c(model1[4], model2[4], rf_500[4], rf_1000[4]),
                          F1_Score = c(model1[5], model2[5], rf_500[5], rf_1000[5]))

# Convert Accuracy column to numeric
comparison$Accuracy <- as.numeric(as.character(comparison$Accuracy))

# Sort the comparison table by Accuracy column
sorted_comparison <- comparison[order(-comparison$Accuracy), ]
```

```
# Print the sorted comparison table
print(sorted_comparison)
```

##	Model	Accuracy	Precision	Recall	F1_Score
## 3	Random Forest 500 Trees	1.0000	1	1	1
## 4	Random Forest 1000 Trees	0.9805	0.9783	0.9783	0.9783
## 1	Decision Tree Model 1	0.9268	0.8889	0.9565	0.9215
## 2	Decision Tree Model 2	0.7317	0.6697	0.7935	0.7264

This table provides a comparison of four different models based on their performance metrics: Accuracy, Precision, Recall, and F1 Score.

Random Forest 500 Trees: This model achieved perfect accuracy (1.0000), precision (1), recall (1), and F1 score (1), indicating flawless performance across all metrics.

Random Forest 1000 Trees: This model achieved high accuracy (0.9805) and very high precision (0.9783), recall (0.9783), and F1 score (0.9783), indicating excellent overall performance.

Decision Tree Model 1: This model achieved good accuracy (0.9268) and precision (0.8889) but slightly lower recall (0.9565) and F1 score (0.9215) compared to the random forest models.

Decision Tree Model 2: This model had the lowest performance among the four models, with accuracy (0.7317), precision (0.6697), recall (0.7935), and F1 score (0.7264). Despite lower performance compared to the random forest models, it still demonstrates reasonable predictive ability.

Overall, the comparison of the four models indicates that the Random Forest models with 500 and 1000 trees outperform the Decision Tree models in terms of accuracy, precision, recall, and F1 score. Both Random Forest models achieved high or perfect scores across all metrics, indicating superior performance in classifying instances compared to the Decision Tree models.

Among the Decision Tree models, Model 1 performed better than Model 2 in terms of accuracy, precision, recall, and F1 score. However, both Decision Tree models had lower scores compared to the Random Forest models, suggesting that ensemble methods like Random Forest are more effective in this scenario.

In conclusion, the Random Forest models with 500 and 1000 trees are recommended for this classification task due to their superior performance.

Reference: <https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction> https://libres.uncg.edu/ir/ecs/f/Brandon_Simmons_Thesis-Final.pdf <https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/>