Data 622 - Machine Learning and Big Data - HW #3 - Decision Trees and Support Vector Machine Algorithms and Essay

Enid Roman

2024-03-29

Please note I continued from HW #2. You can scroll down to line # 316 where I added analysis of the Comparing the attributes to the target plots. Then scroll to line # 546 Random Forest. Since Random Forest 500 trees scored an 100% accuracy I did an evaluation to make sure it was not due to overfitting. I hope I did the analysis correct for the evaluation and SVM. If it is not correct please advise. Thank you.

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

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

Atribute 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(abnornality 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 Thalium 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 occured. (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/ma
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
```

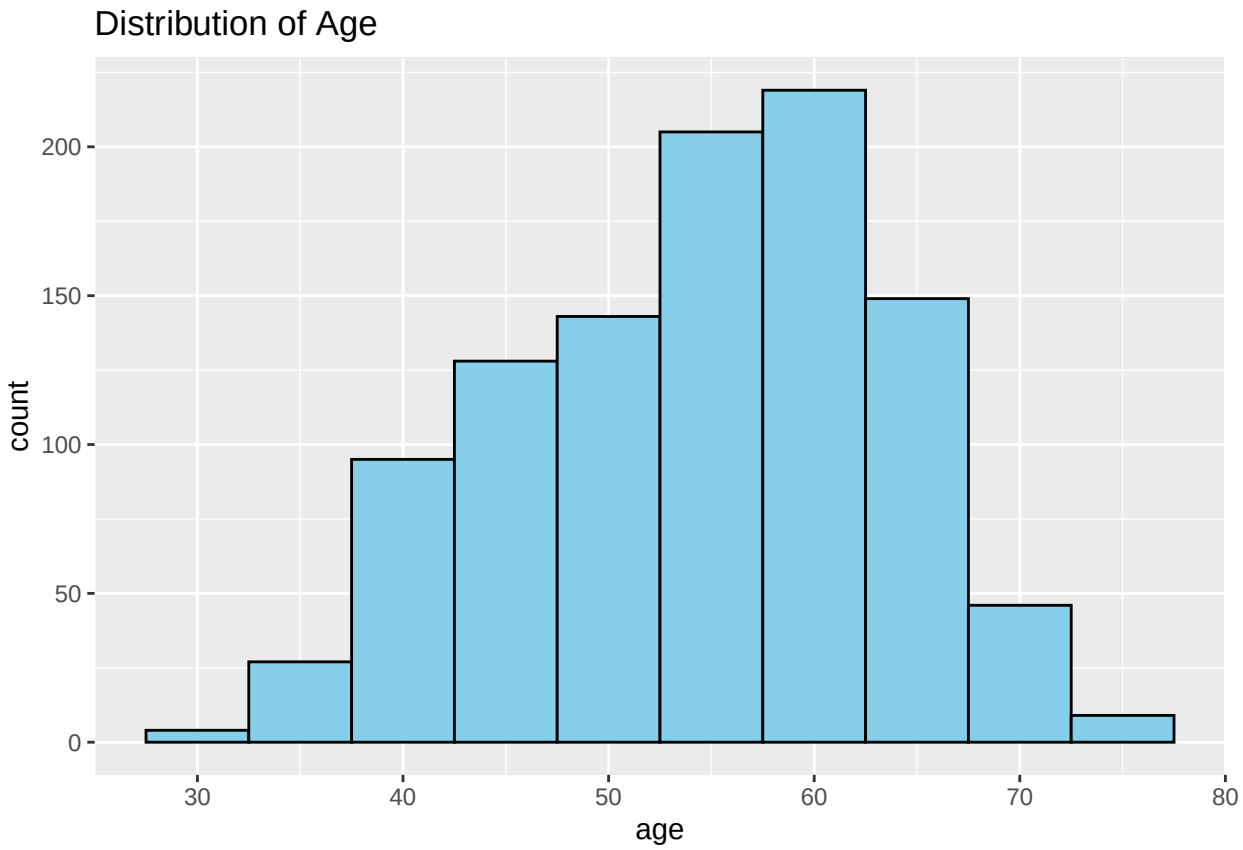
No missing values.

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.

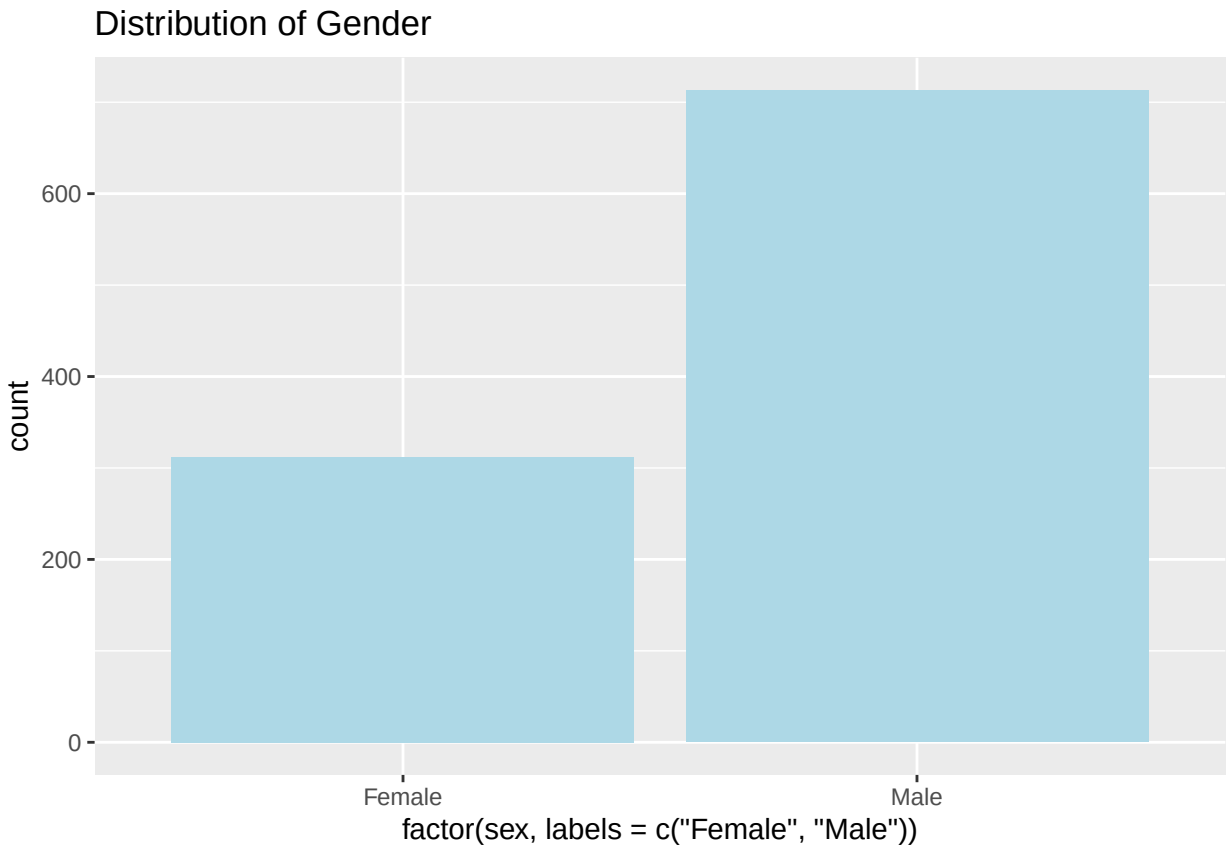
All Attributes

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



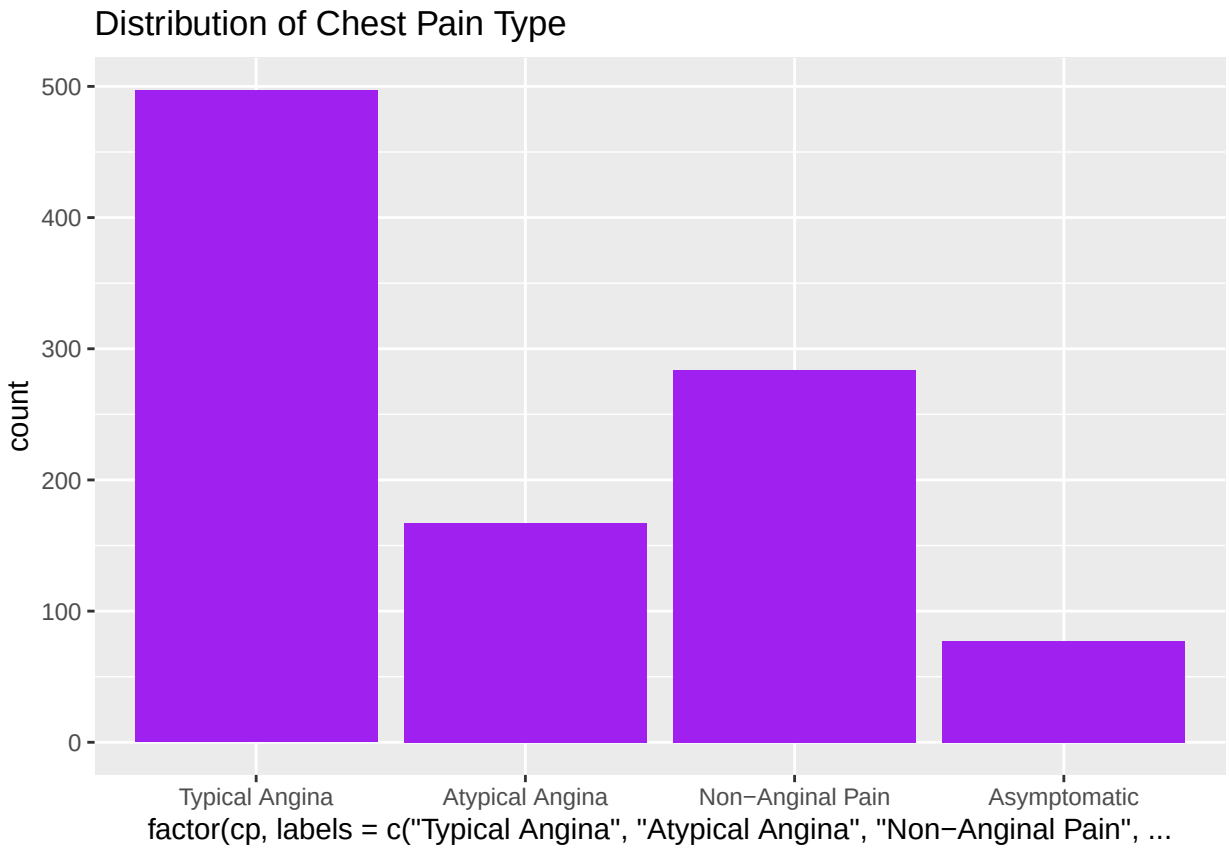
Maximum age of patient is between 50 adn 60.

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



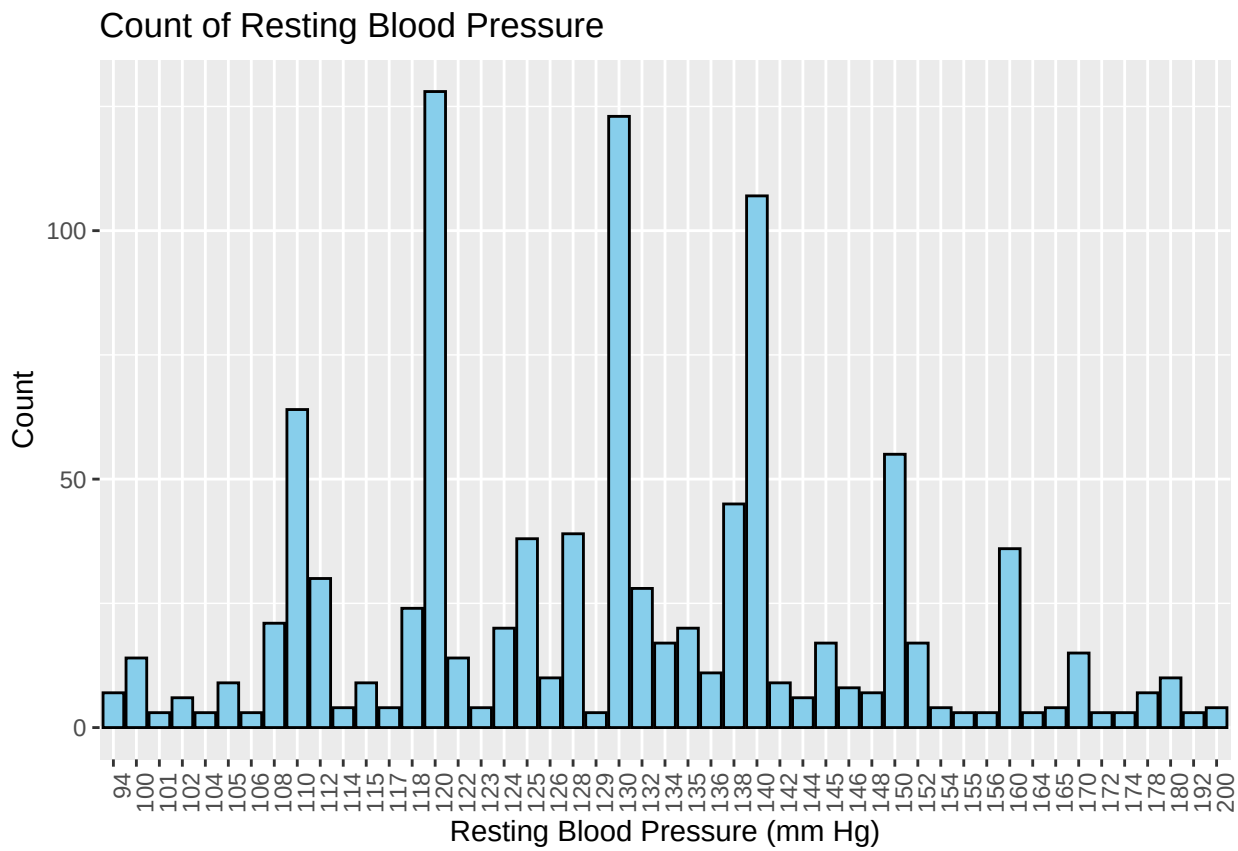
There are more male patient then female patient.

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



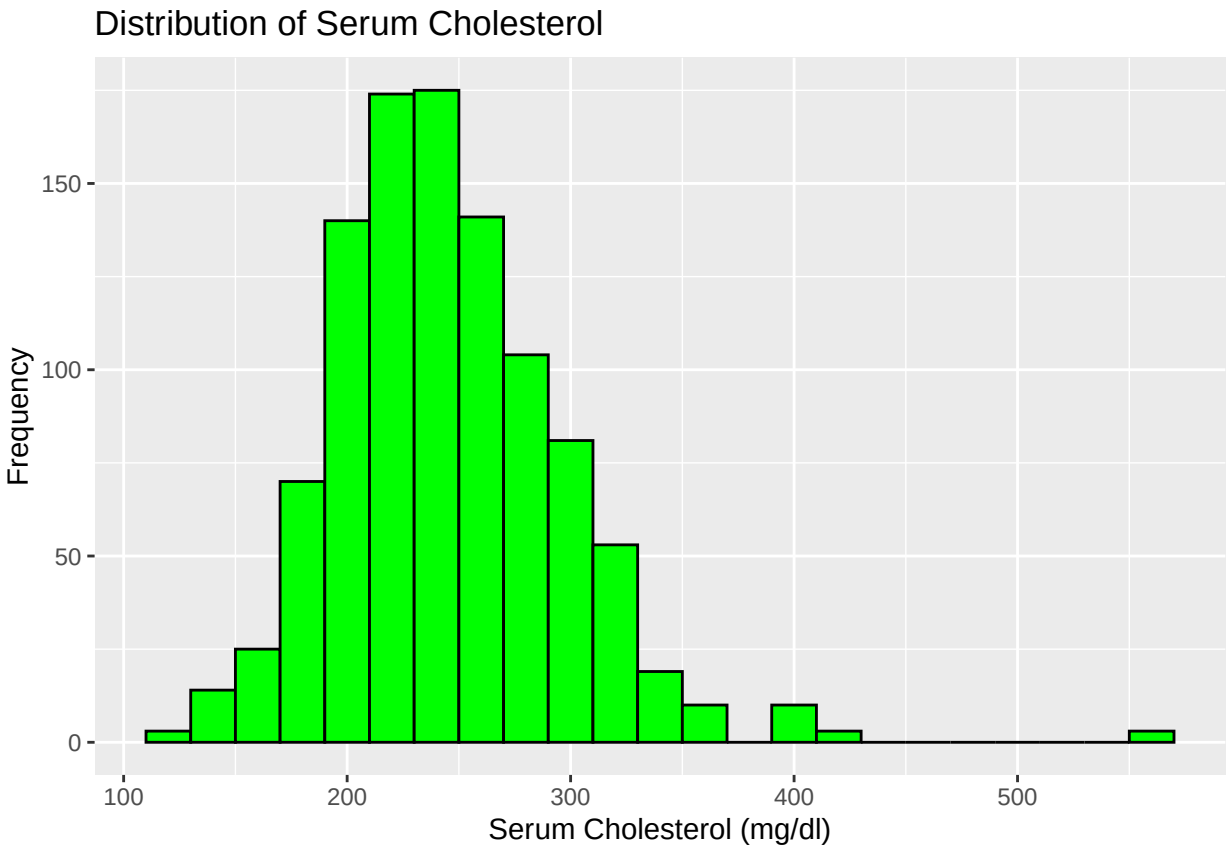
Most patient complaing of typical angina, a symptom of myocardial ischemia. Typical angina is characterized by chest discomfort or anginal equivalent that is provoked with exertion and alleviated at rest or with nitroglycerin.

```
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))
```



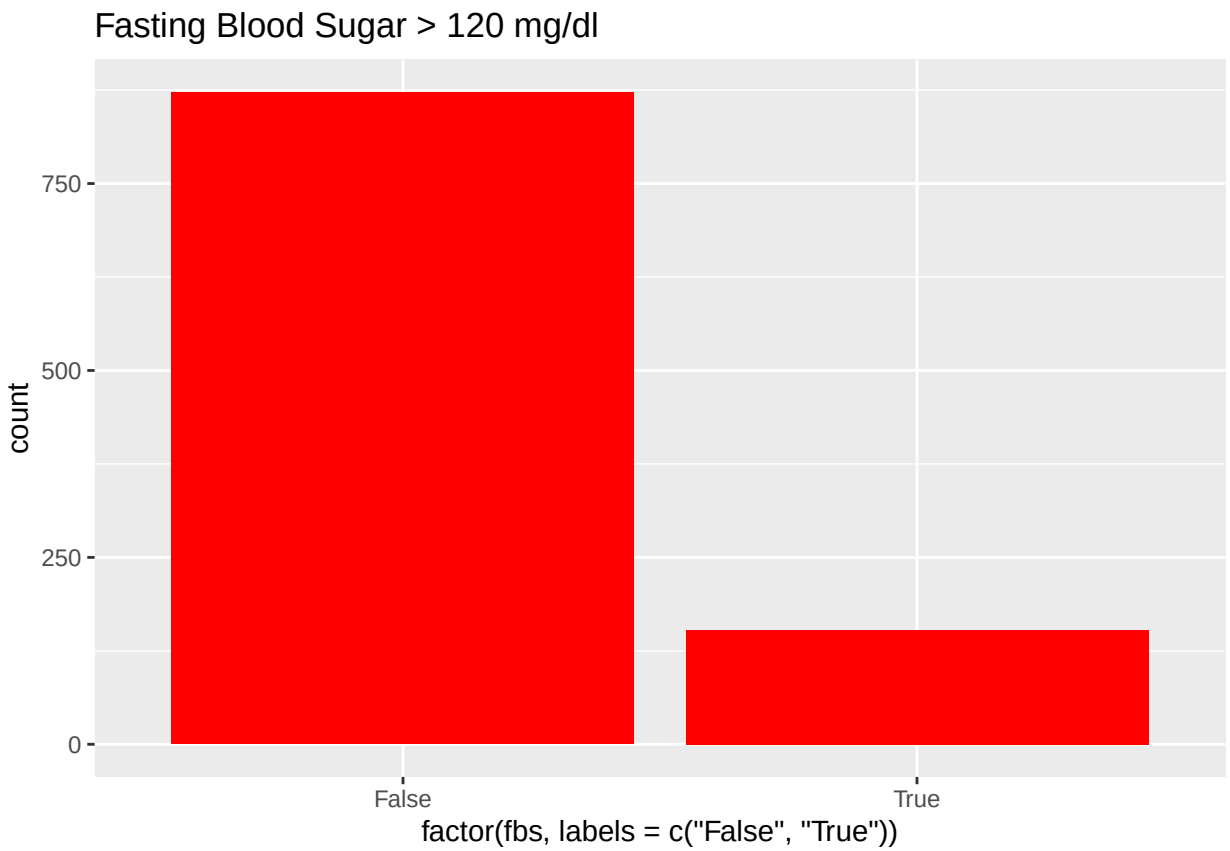
Most patient blood pressure was 120, 130, and 140. 120 to 139 are at risk of high bp. 140 is high bp.

```
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")
```



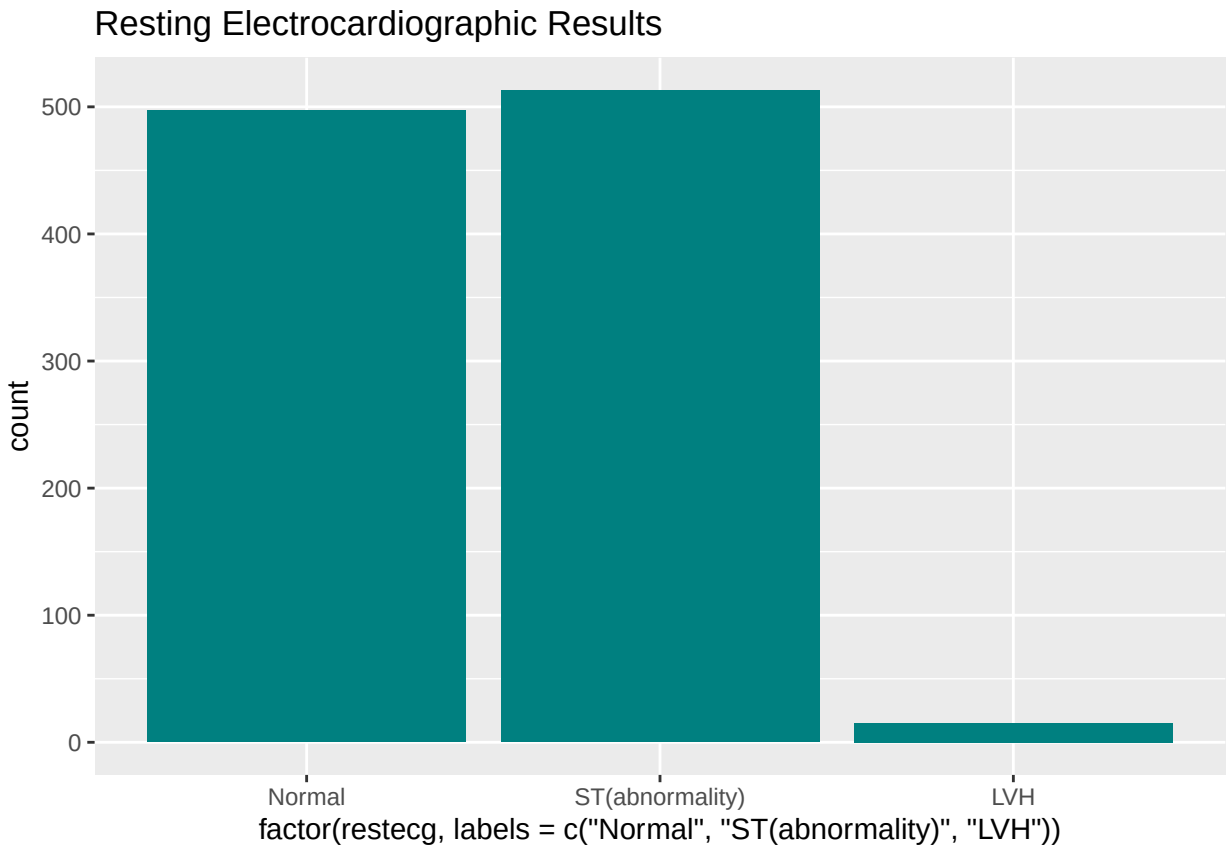
Most patient had cholesterol approximately 200 to 280. 200 to 239 is borderline. 240 and over are high cholesterol.

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



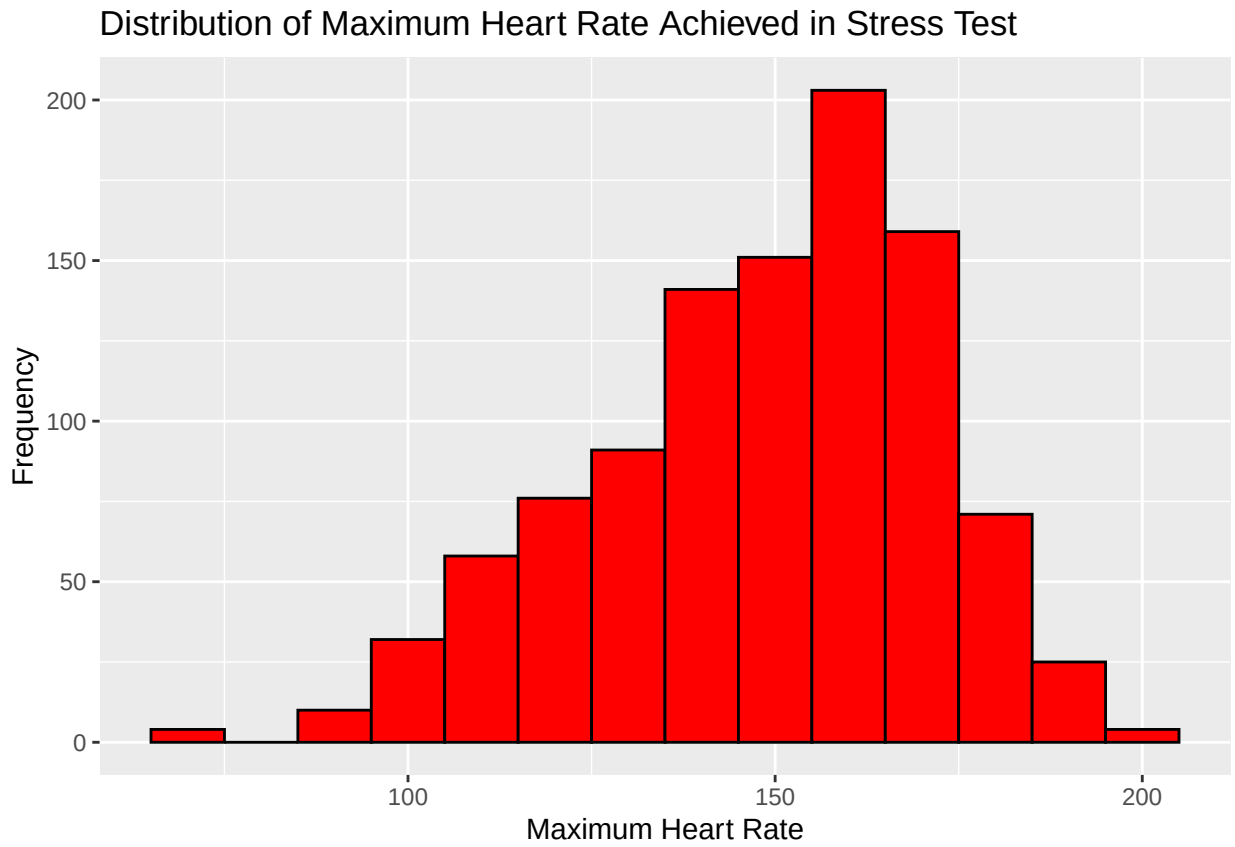
Most patient did not have high sugar level.

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



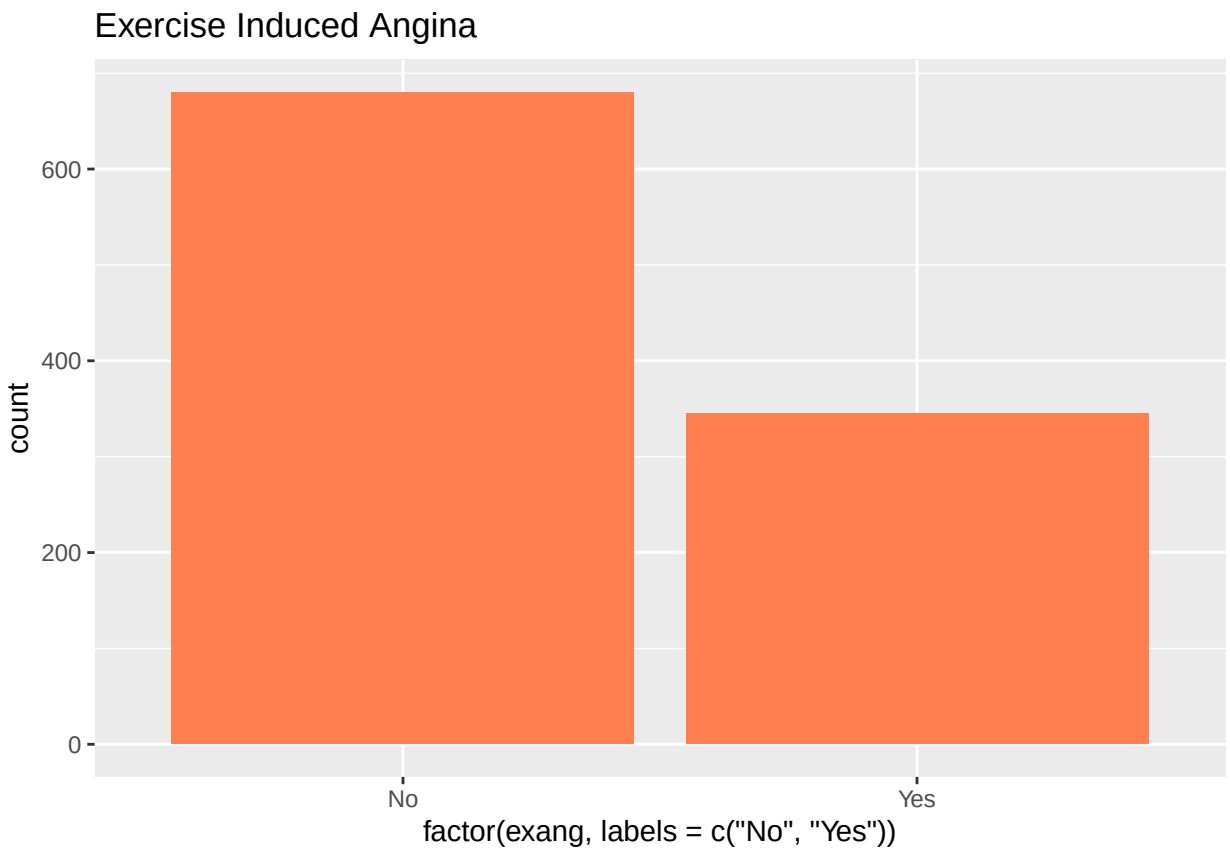
Most patient had abnormal electrocardiogram. But also there was almost the same amount of patient that had normal electrocardiogram.

```
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")
```



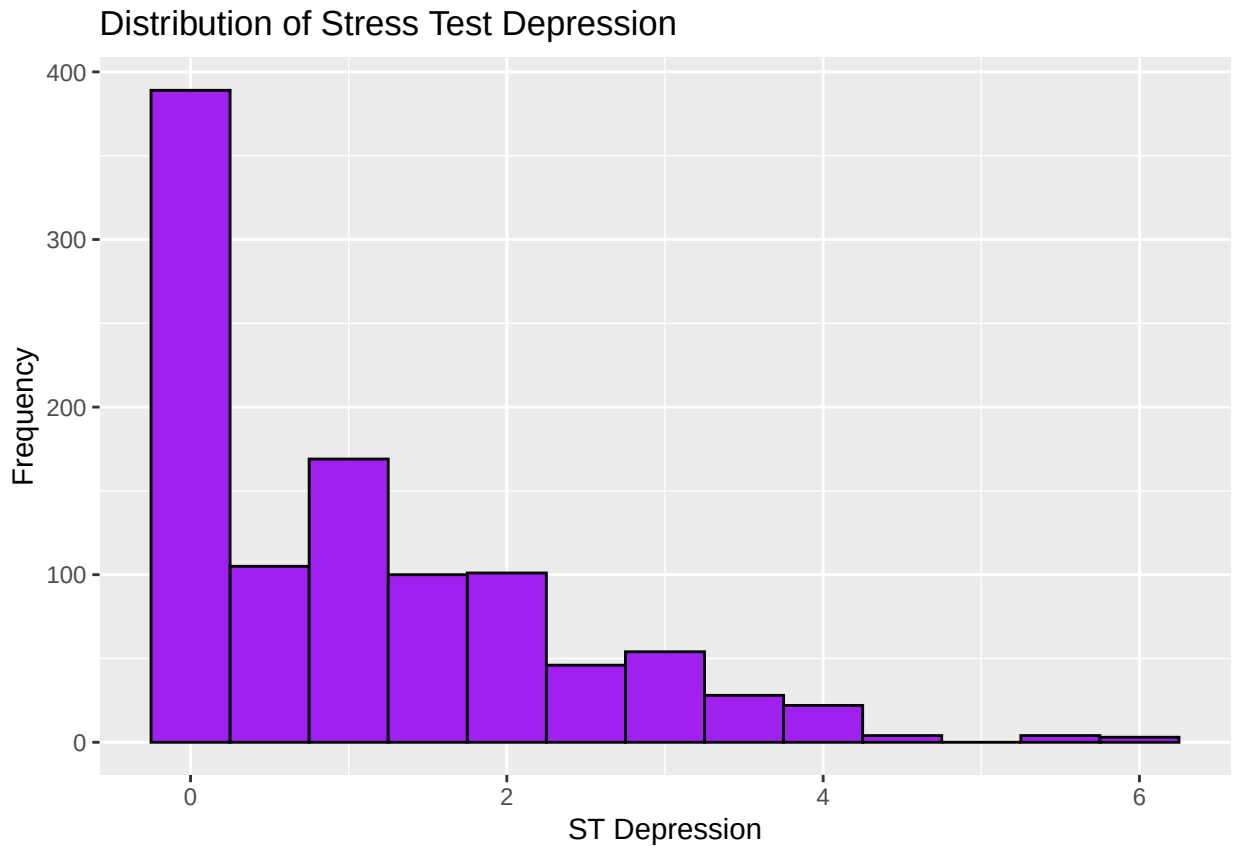
Maximum heart rate patients reached in a stress test was approximately 153. The target heart rate during a stress test depends on your age. Is the maximum predicted heart rate minus the age.

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



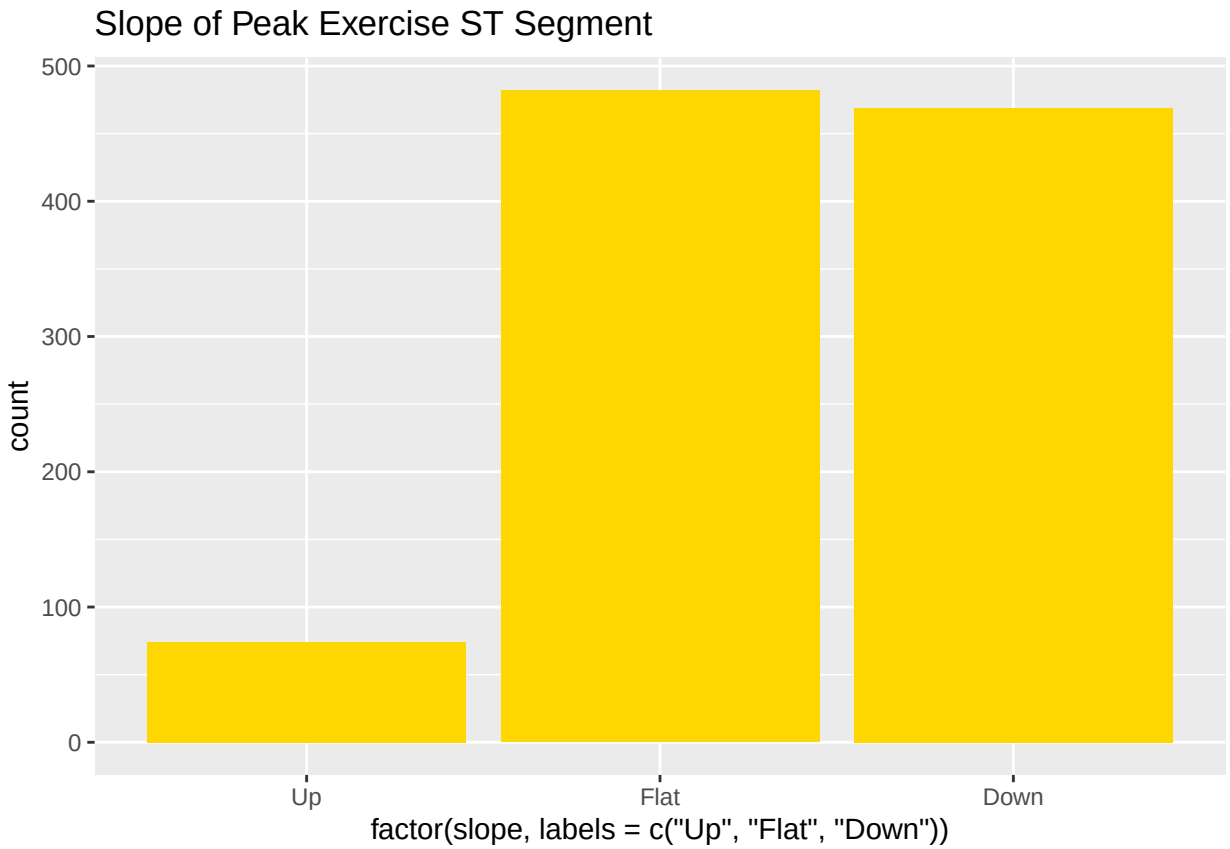
The majority patient did not have angina induced when exercised.

```
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")
```



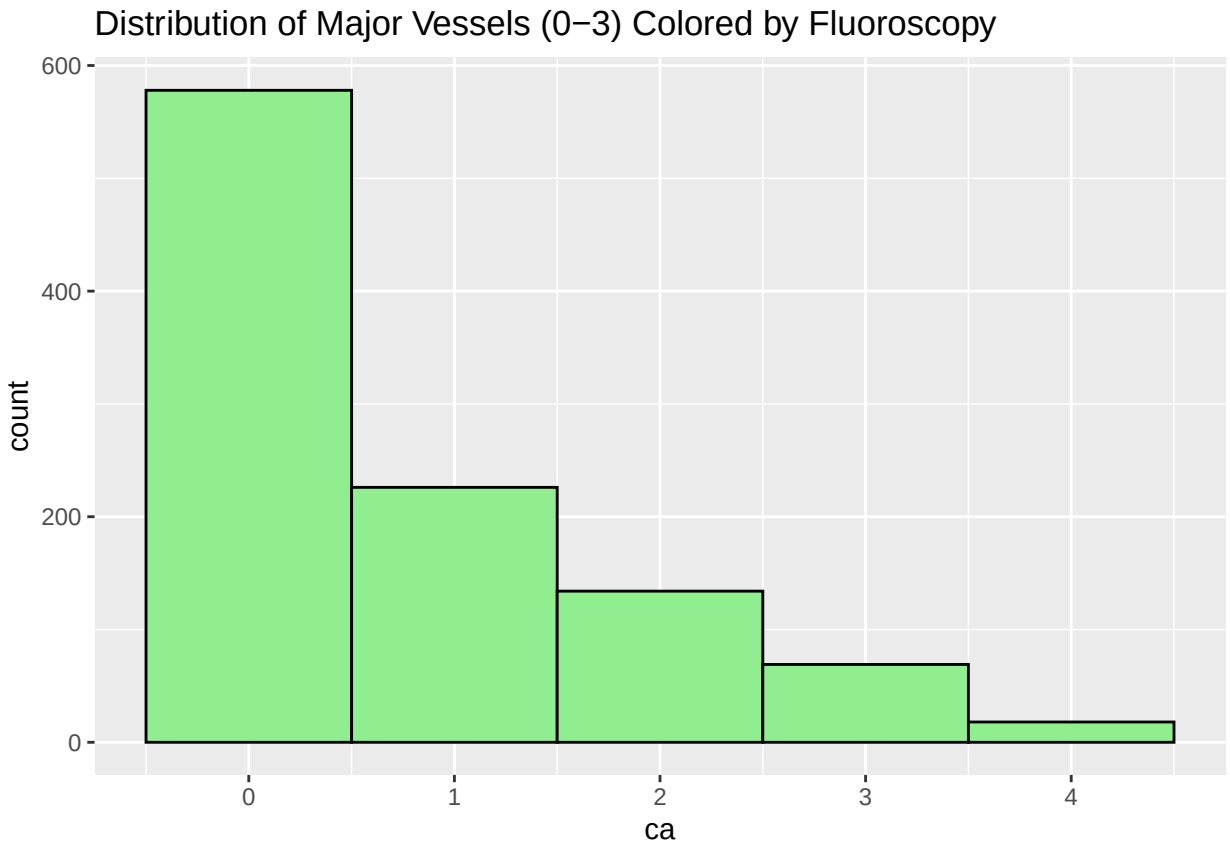
Most patient did not have cardiac abnormalities. The values of "oldpeak" range from 0 to 6, indicating the magnitude of the depression observed during the stress test. A higher value typically suggests more significant cardiac abnormalities.

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



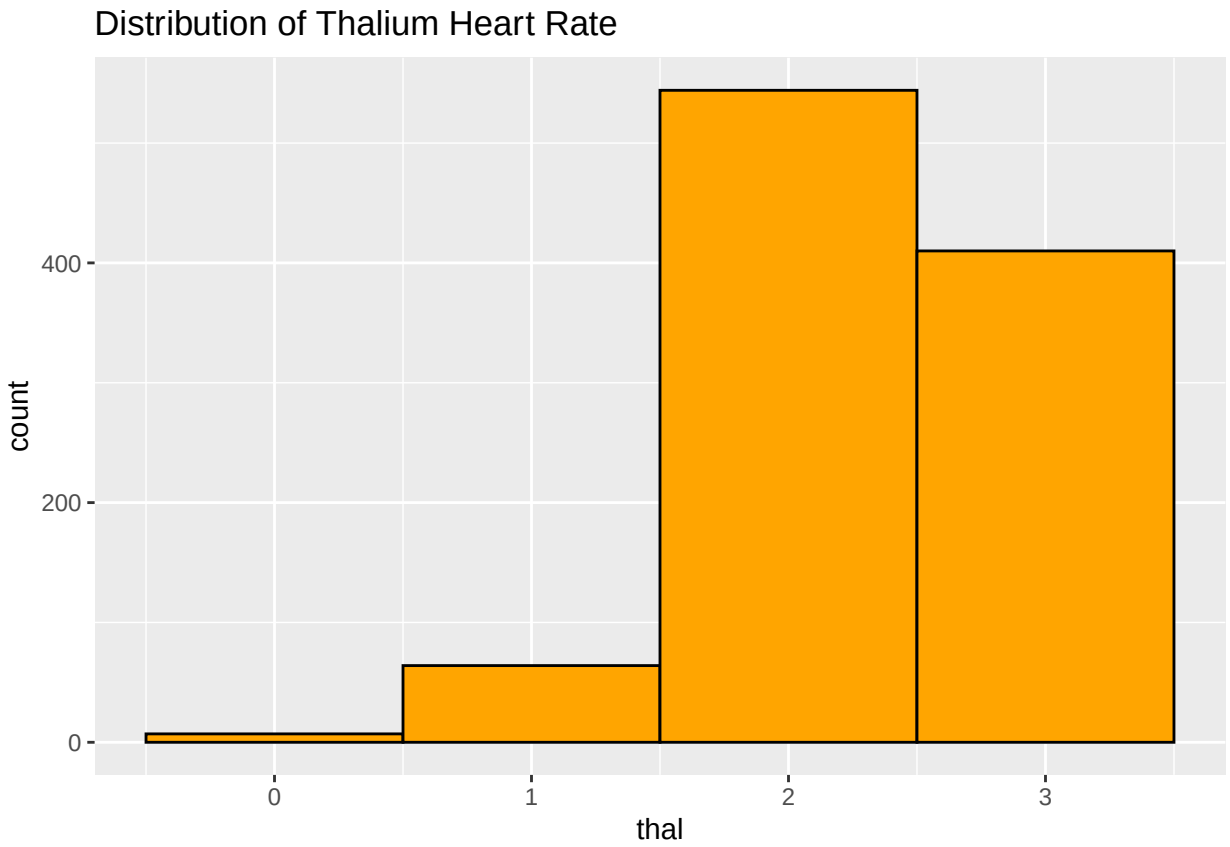
An upward slope may indicate a transient elevation of the ST segment, which can be a sign of various cardiac conditions, such as myocardial ischemia or infarction. A flat or horizontal slope it suggests a lack of significant deviation from the resting state. This is often interpreted as a normal response, 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")
```



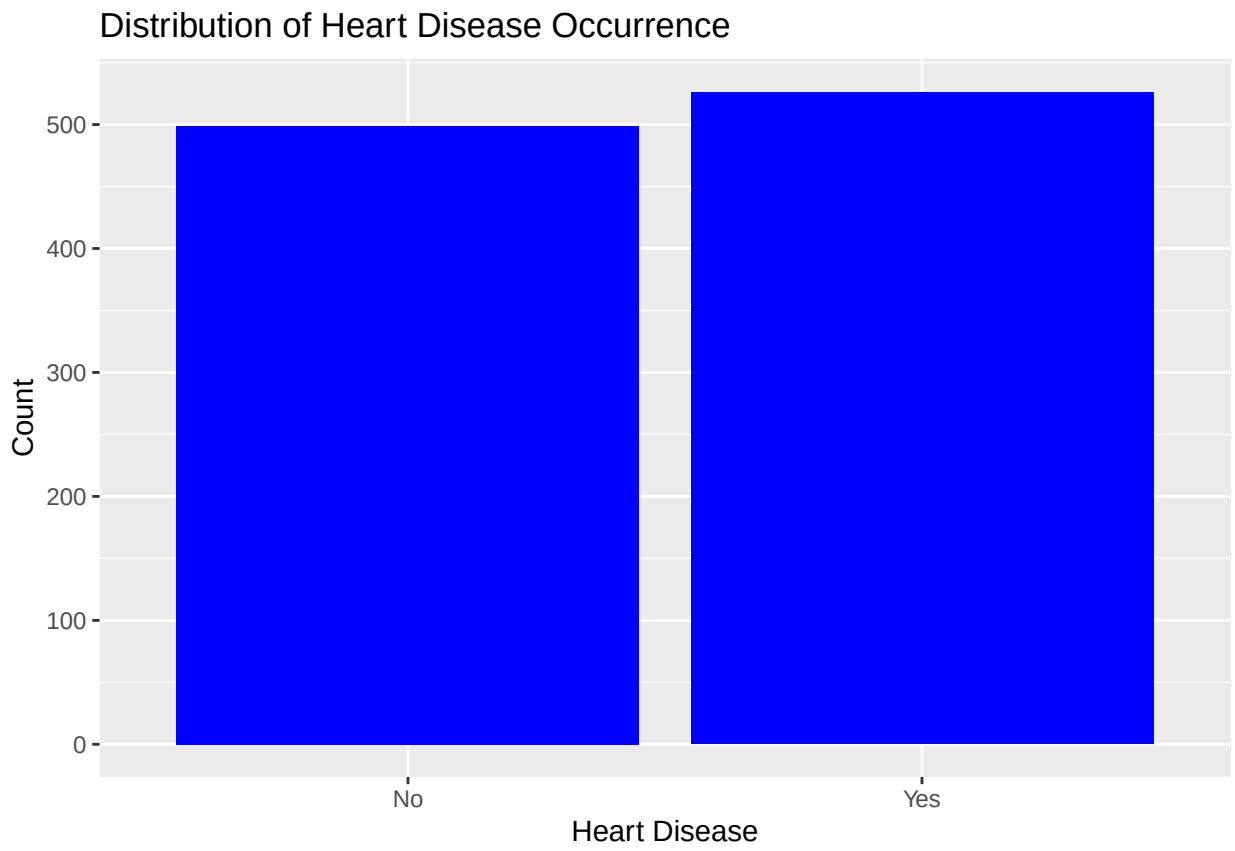
The values of "ca" range from 0 to 3, indicating the number of major vessels that were observed and colored during the fluoroscopy procedure. Which most patients did not have their major vessels observed.

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

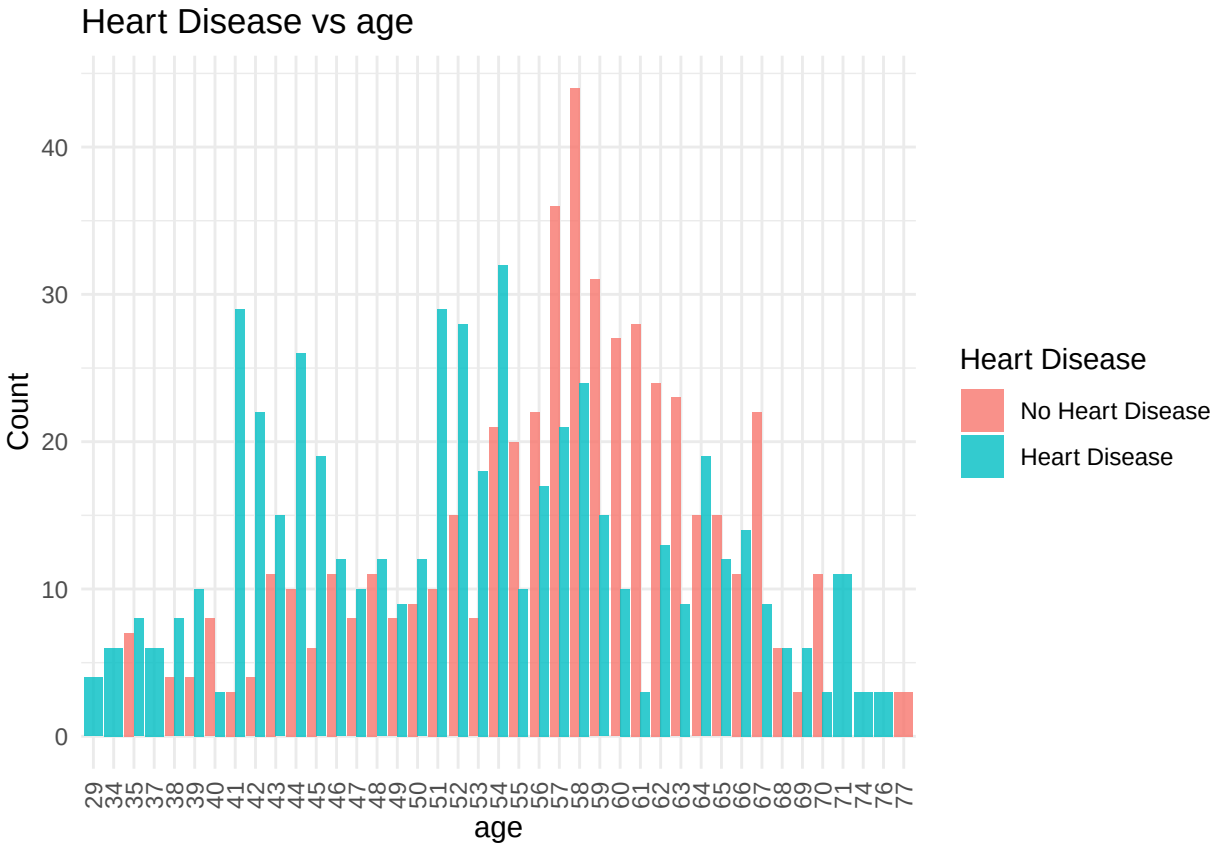


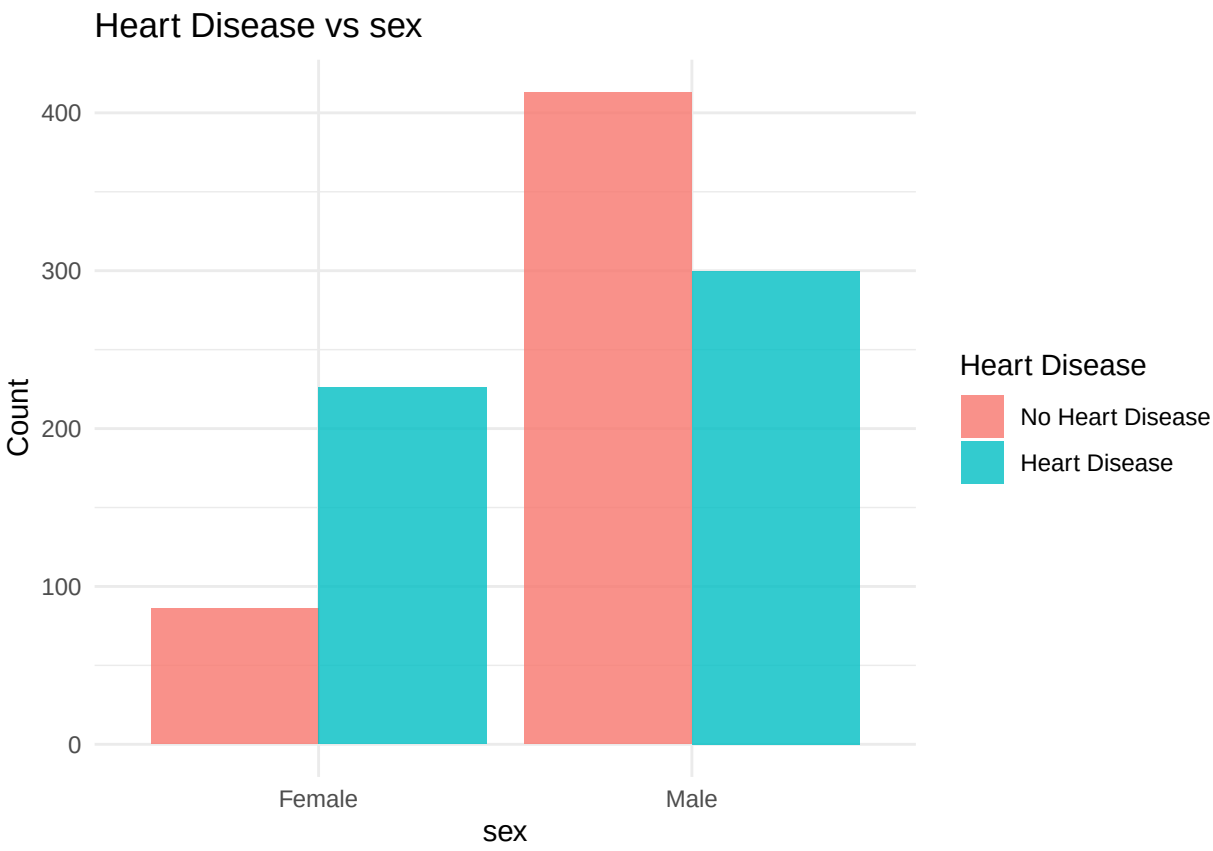
Most of the patients have heart disease.

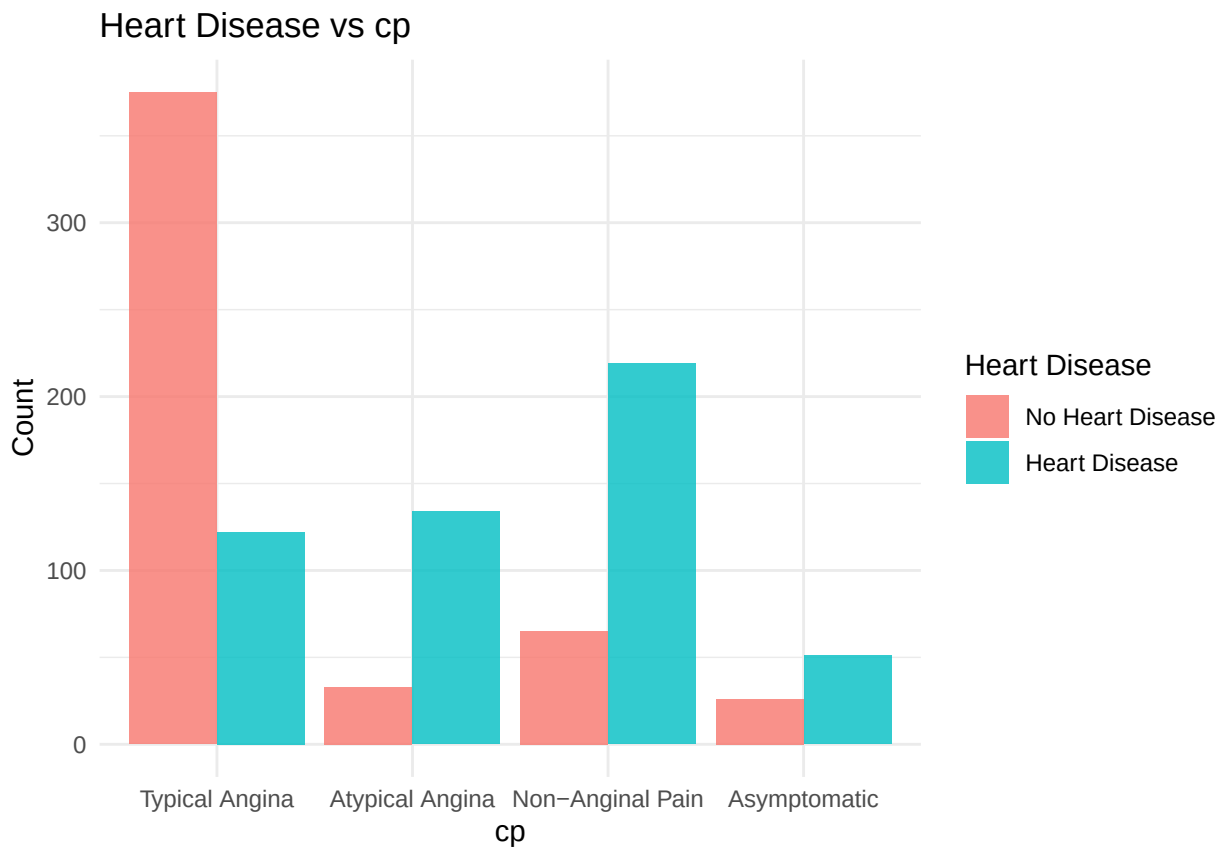
Comparing all attributes with target heart disease.

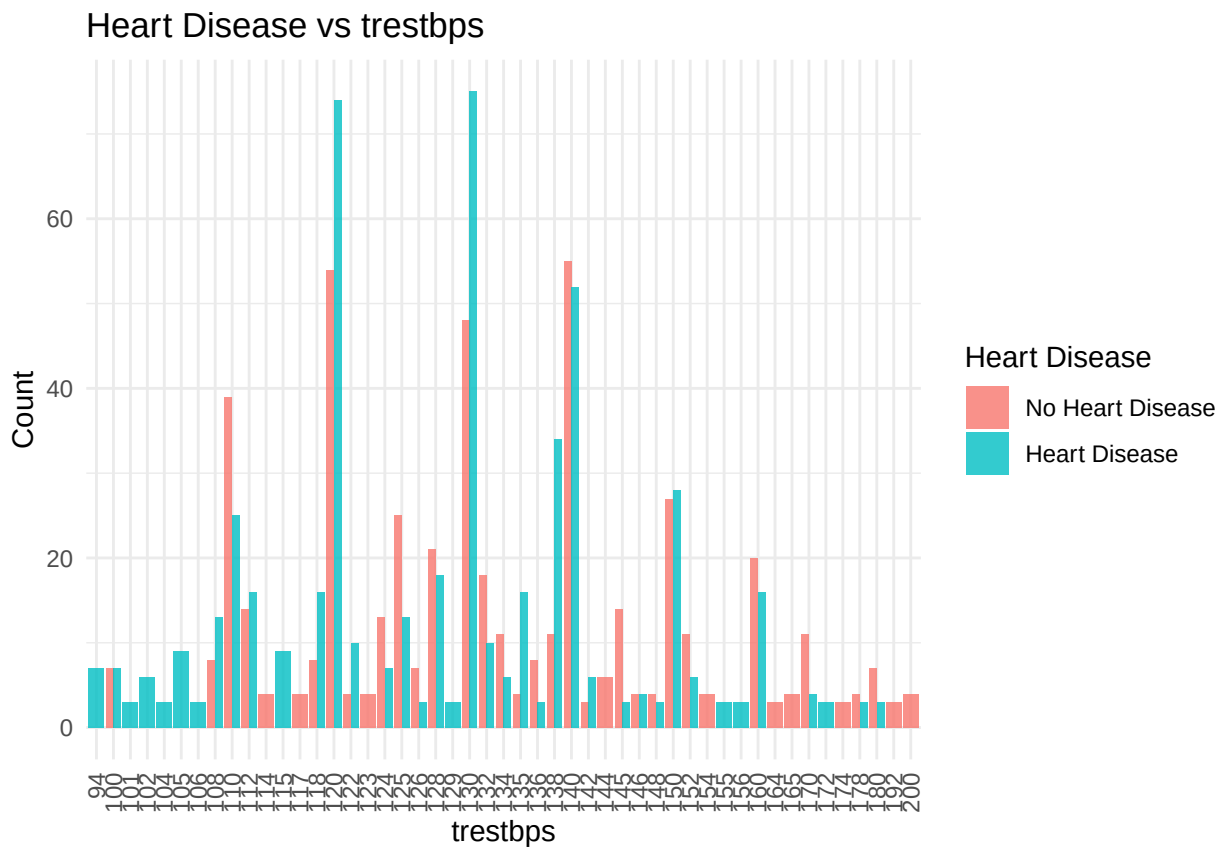
```
# Create a function to replace numeric values with labels
replace_labels <- function(data, attribute) {
  if (attribute == "sex") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("Female", "Male"))
  } else if (attribute == "cp") {
    data[[attribute]] <- factor(data[[attribute]], labels = c("Typical Angina", "Atypical Angina", "Non
  } 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)
}
```

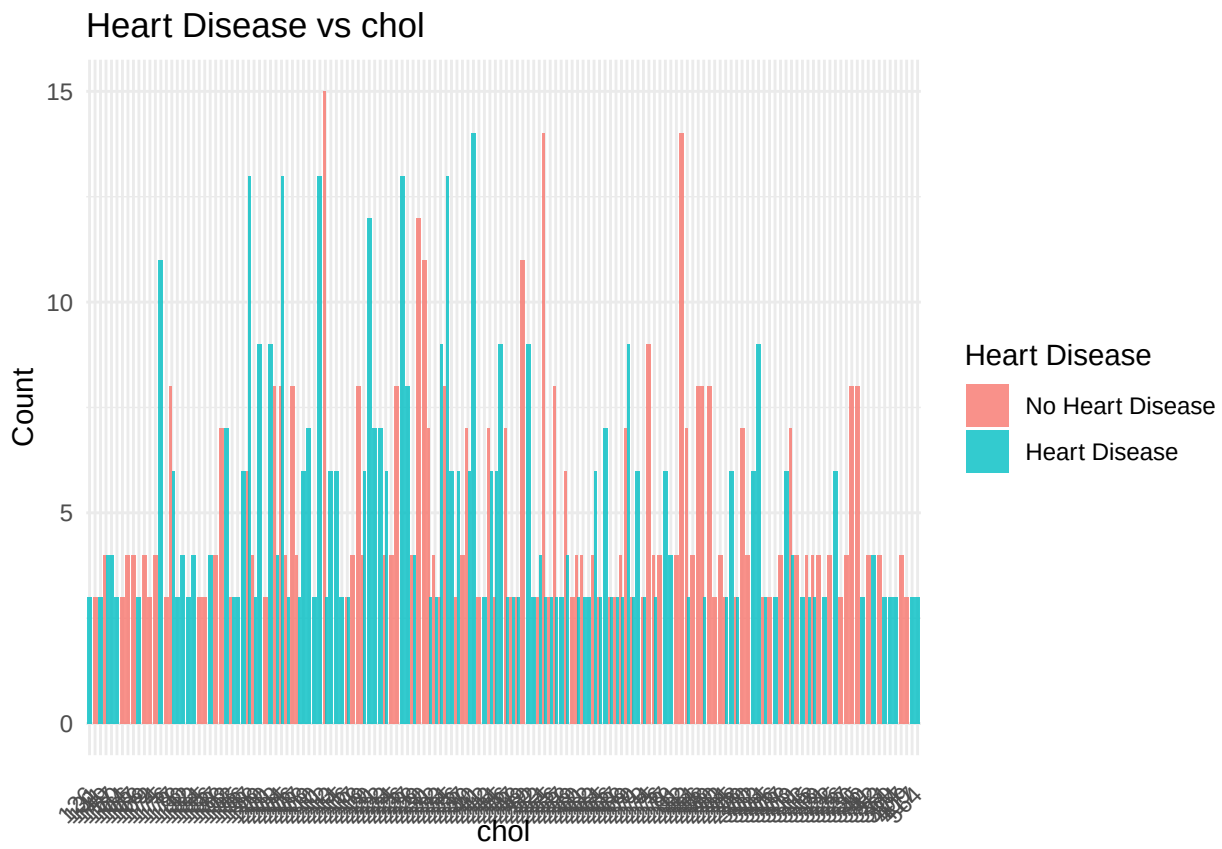
```r
# Replace numeric values with labels for specified attributes
heart <- replace_labels(heart, "sex")
heart <- replace_labels(heart, "cp")
heart <- replace_labels(heart, "fbs")
heart <- replace_labels(heart, "restecg")
heart <- replace_labels(heart, "exang")
heart <- replace_labels(heart, "slope")
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)
}
```

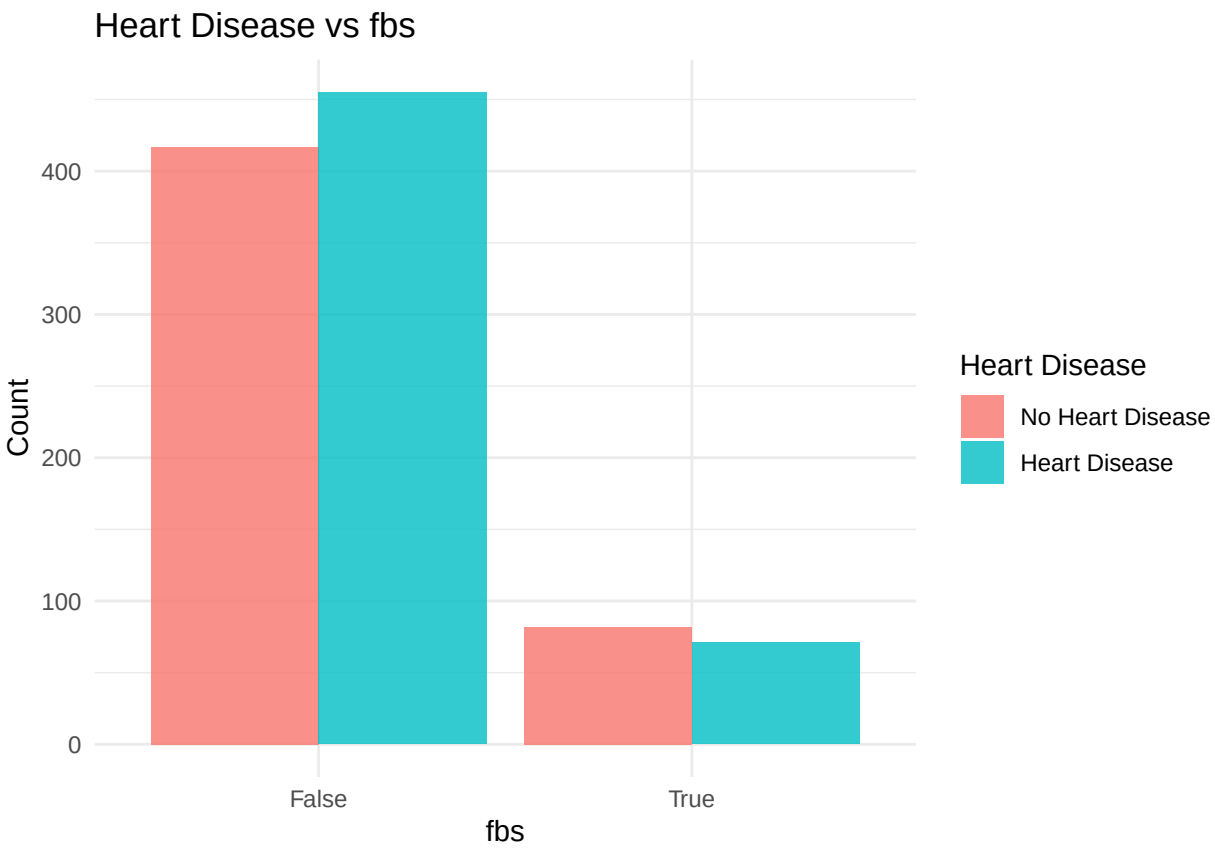

Heart Disease vs age
Count
40
30
20
10
0
29
34
35
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
74
76
77
age
Heart Disease
No Heart Disease
Heart Disease

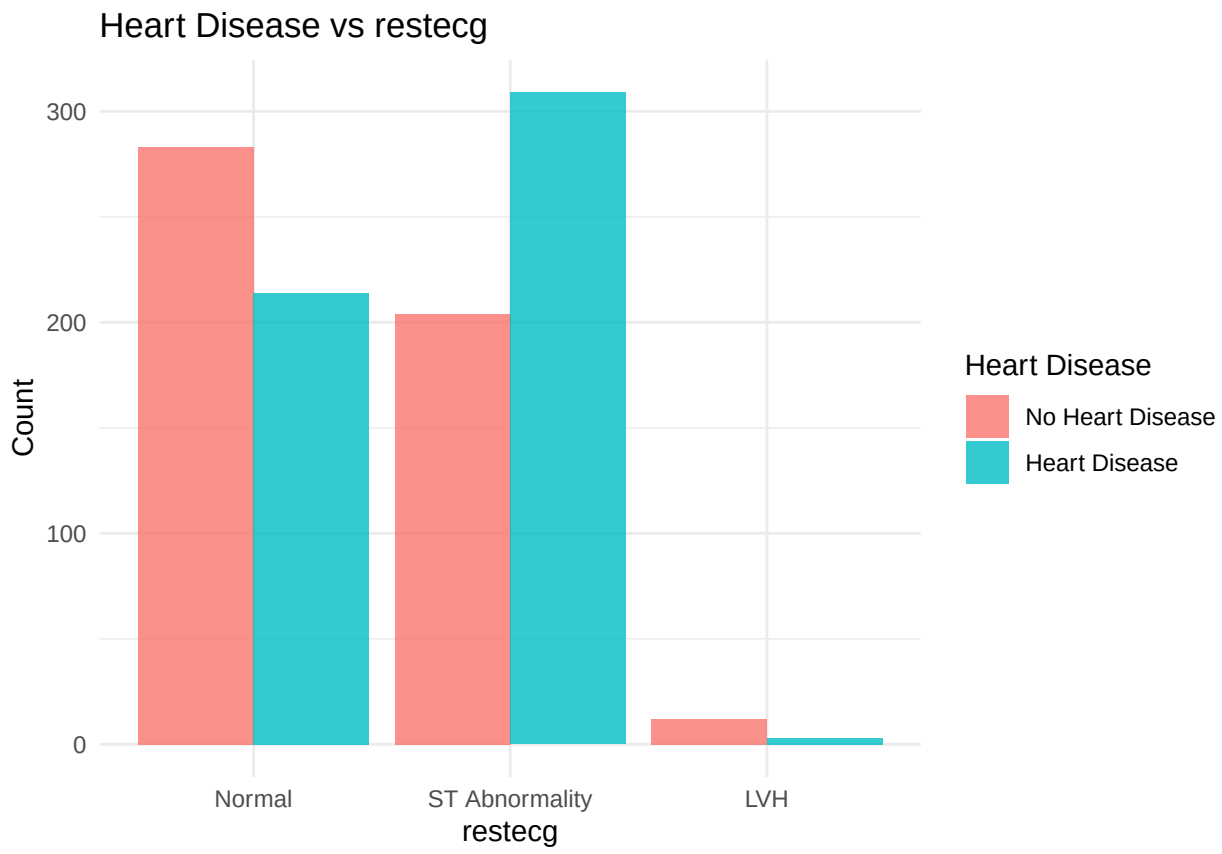

Heart Disease vs sex
400
300
200
100
0
Count
Female
Male
sex
Heart Disease
No Heart Disease
Heart Disease

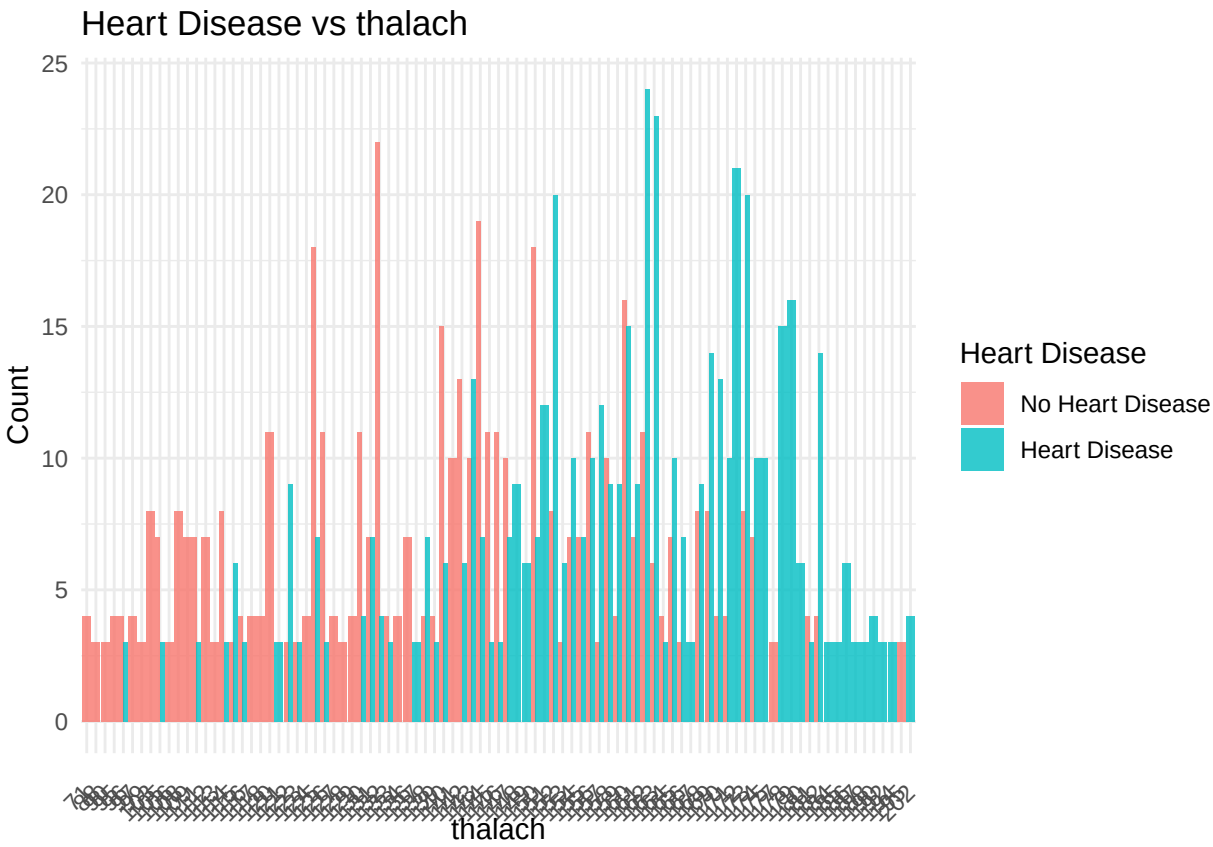

Heart Disease vs cp
Count
300
200
100
0
Typical Angina
Atypical Angina
Non−Anginal Pain
Asymptomatic
cp
Heart Disease
No Heart Disease
Heart Disease

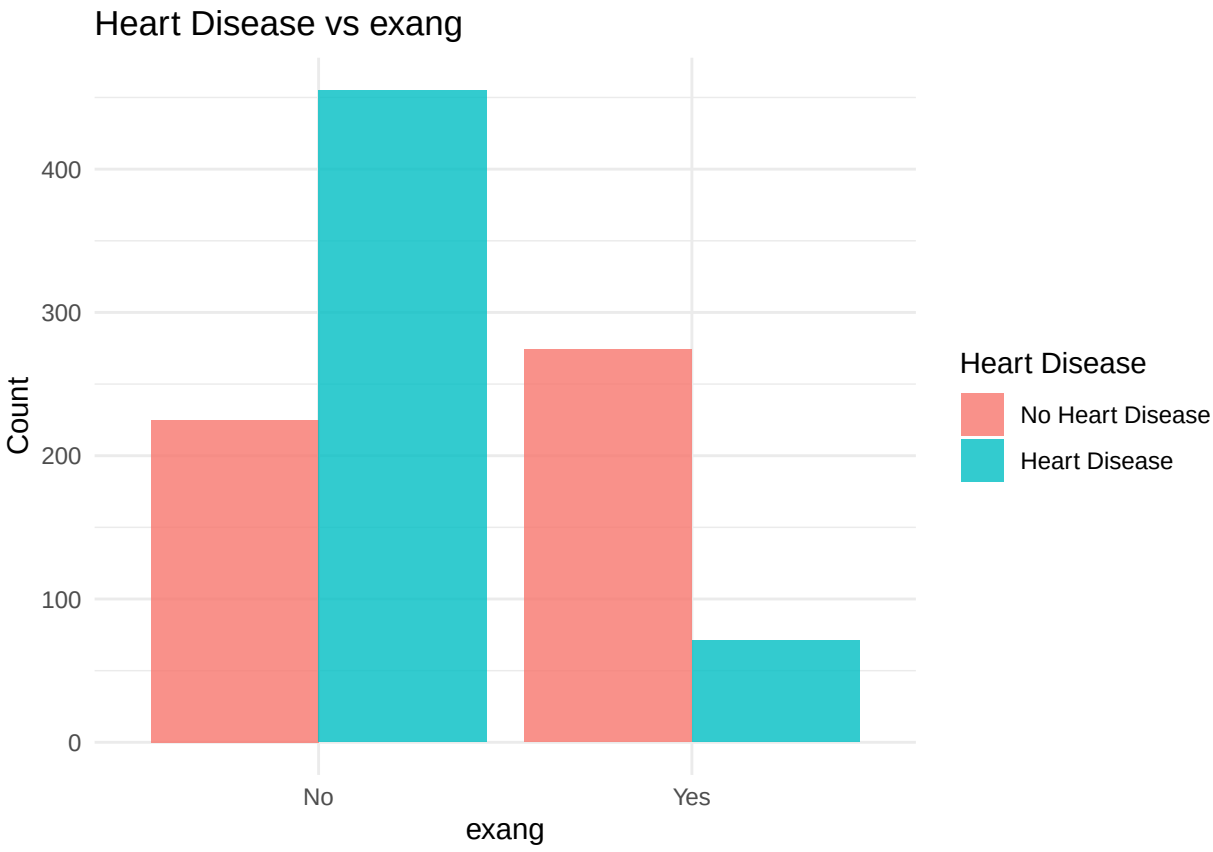

Heart Disease vs trestbps
Count
0
20
40
60
94
100
101
102
104
105
106
108
110
112
114
115
117
118
120
122
123
124
125
126
128
129
130
132
134
135
136
138
140
142
144
145
146
148
150
152
154
155
156
160
164
165
170
172
174
178
180
192
200
trestbps
Heart Disease
No Heart Disease
Heart Disease

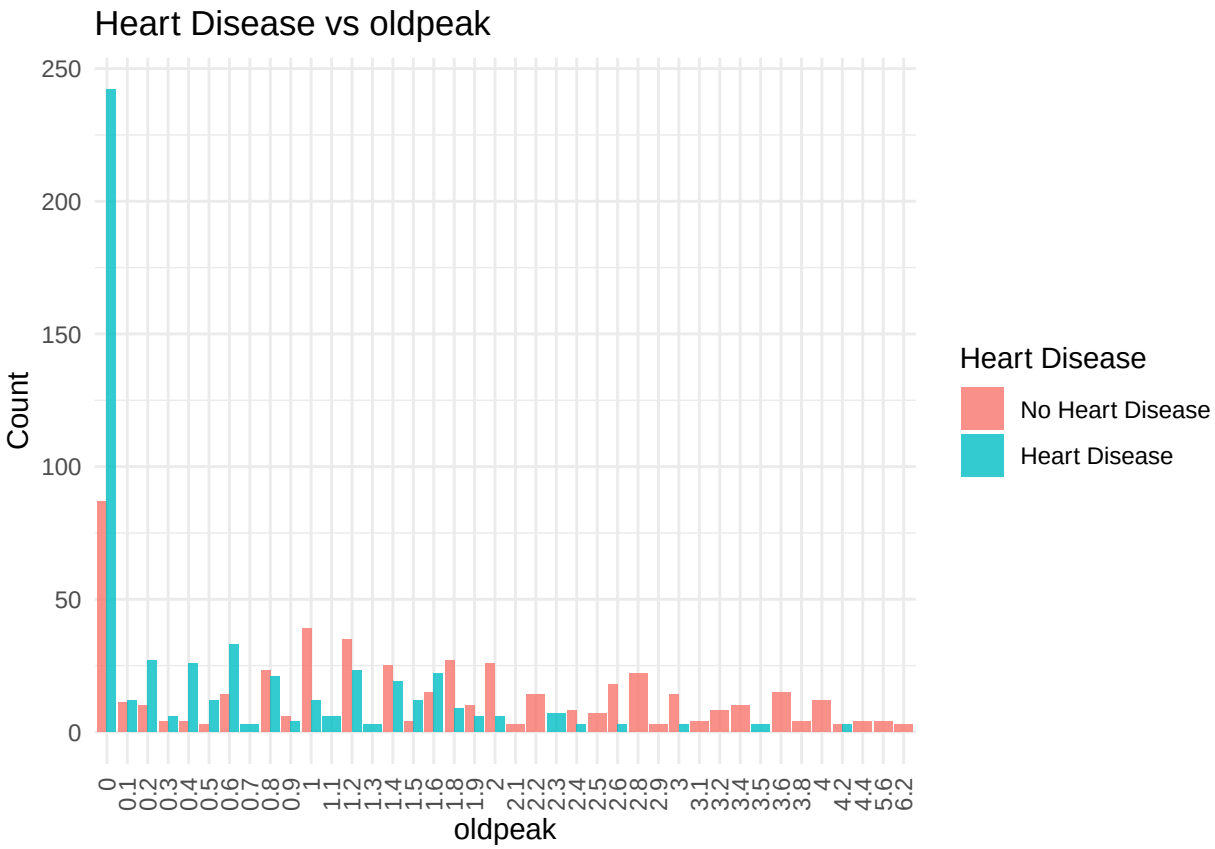

Heart Disease vs chol
Count
15
10
5
0
chol
Heart Disease
No Heart Disease
Heart Disease

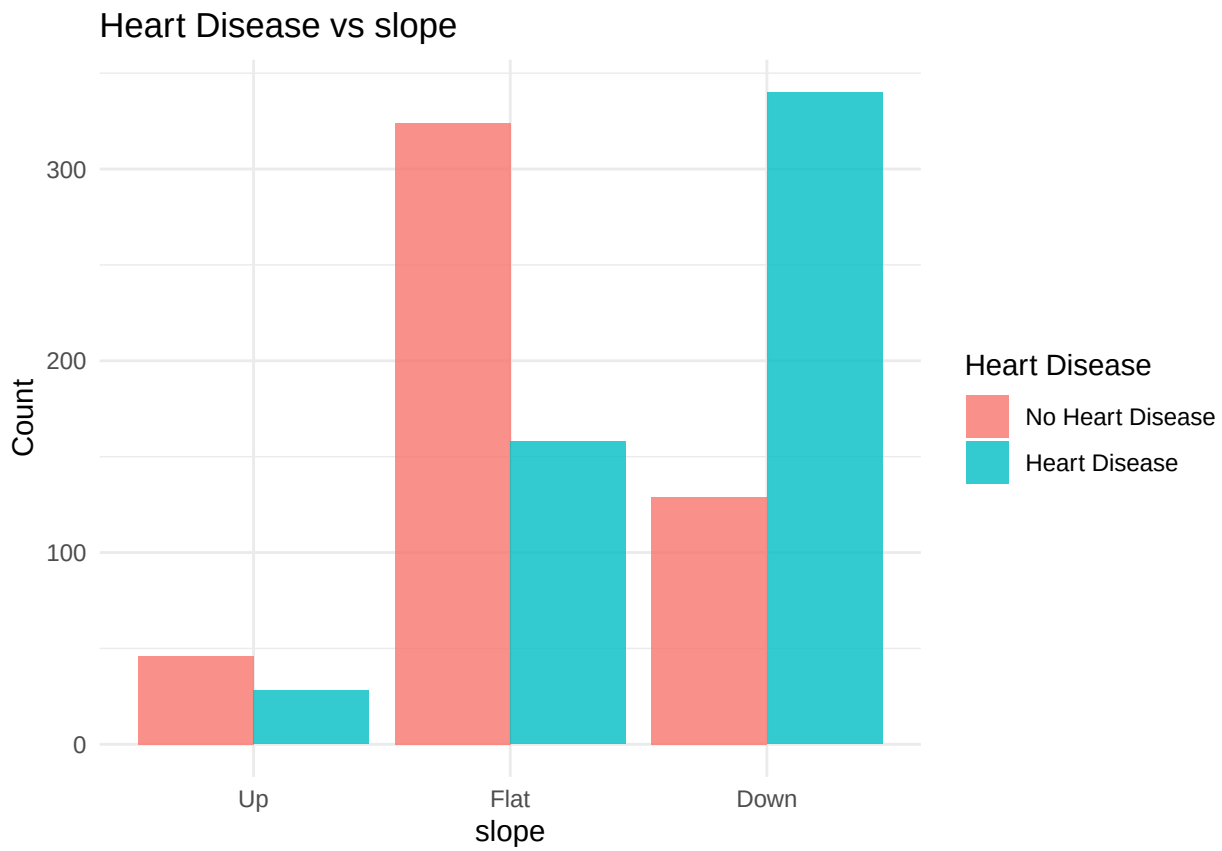

Heart Disease vs fbs
Count
400
300
200
100
0
False
True
fbs
Heart Disease
No Heart Disease
Heart Disease

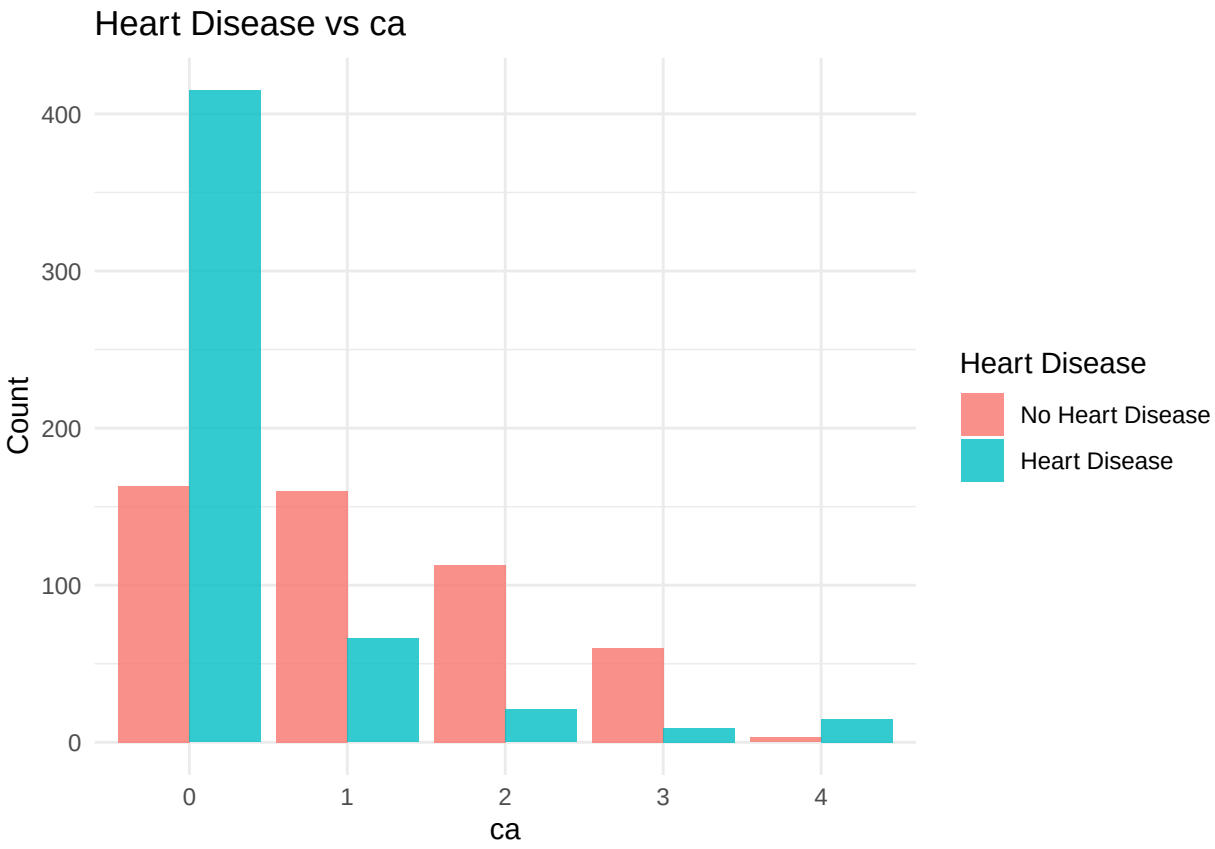

Heart Disease vs restecg
300
200
100
0
Count
Normal
ST Abnormality
LVH
restecg
Heart Disease
No Heart Disease
Heart Disease

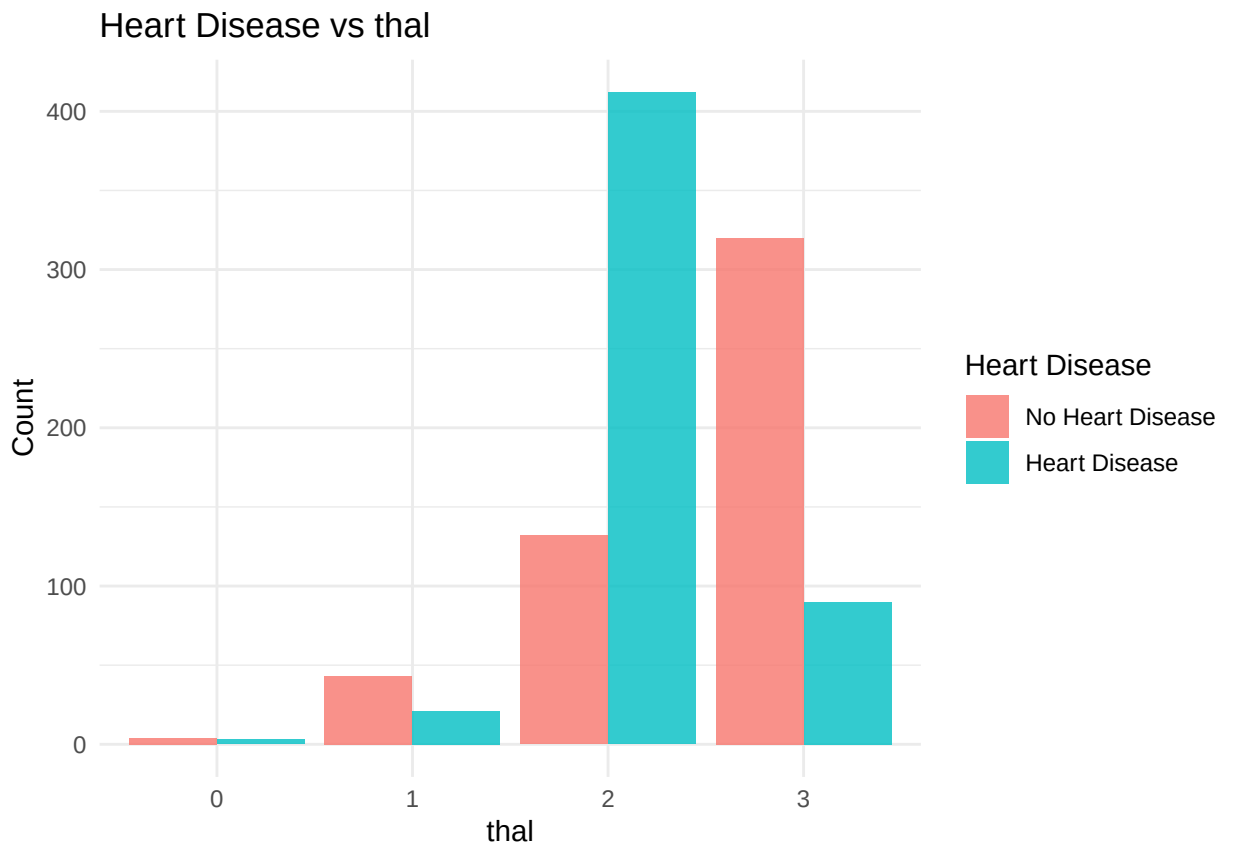

Heart Disease vs thalach
25
20
15
10
5
0
Count
thalach
Heart Disease
No Heart Disease
Heart Disease


Heart Disease vs exang
Count
400
300
200
100
0
No
Yes
exang
Heart Disease
No Heart Disease
Heart Disease


Heart Disease vs oldpeak
Count
250
200
150
100
50
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.8
1.9
2
2.1
2.2
2.3
2.4
2.5
2.6
2.8
2.9
3
3.1
3.2
3.4
3.5
3.6
3.8
4
4.2
4.4
5.6
6.2
oldpeak
Heart Disease
No Heart Disease
Heart Disease


Heart Disease vs slope
300
200
100
0
Count
Up
Flat
Down
slope
Heart Disease
No Heart Disease
Heart Disease


Heart Disease vs ca
Count
400
300
200
100
0
0
1
2
3
4
ca
Heart Disease
No Heart Disease
Heart Disease



The above plots shows Heart Disease(Target) vs the 14 Attributes.

Heart Disease by Age: Most had no heart disease at age 58 and most with heart disease were 54 years of age. Heart Disease by Sex: Most male have more heart disease then female but within femaile category more have heart disease vs male that don't. Heart Disease by Chest Pain: Those that had typical angina had no heart disease. Those with atypical angina had heart disease. Those with Asymptomatic had heart disease. Those with Typical Angina most had no heart disease. Heart Disease by Resting Blood Pressure: Those with 120 and 130 had heart disease. Heart Disease by Fasting Blood Sugar: Those with no fbs had heart disease vs those with fbs that had no heart disease. Heart Disease by Resting EKG: Normal resting EKG had no heart disease, ST Abnormality heart dieseae had heart disease, LVH very minimal had no heart disease. Heart Disease by Exercise Induced Angina: Those with no induced angina had heart disease. Those with induced angina had heart disease. Heart Disease by Stress Test Depression Induced by Exercise: Those with 0 mostly had no heart disease vs the one with 0 that had heart disease. Heart Disease by Slope: Those with Up had no heart disease. Those with flat had no heart disease. Those with down had heart disease. Heart Disease by Number of Major Vessels: Those with 0 and 4 had heart disease. Those with 1, 2, and 3 had no heart disease. Heart Disease by Thalum Heart Rate: Most with 2 had heart disease vs 3 that had no heart disease. Heart Disease by Cholesterol and Heart Disease by Maximum Heart Rate Achieved in Stress Test: Hard to read the numbers.

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
```

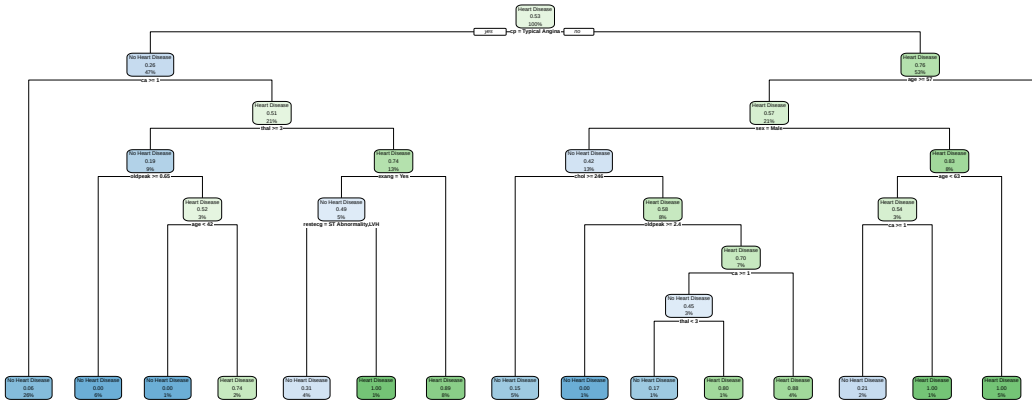
```
## target
## No Heart Disease    Heart Disease
##            55.12            44.88
```

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.)

```r
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
```

```r
# 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"
```

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

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

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

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

```r
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 choloesterol because looking at the previous Decision Tree in Model 1 the patients that had the highest percentange 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
```

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

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

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

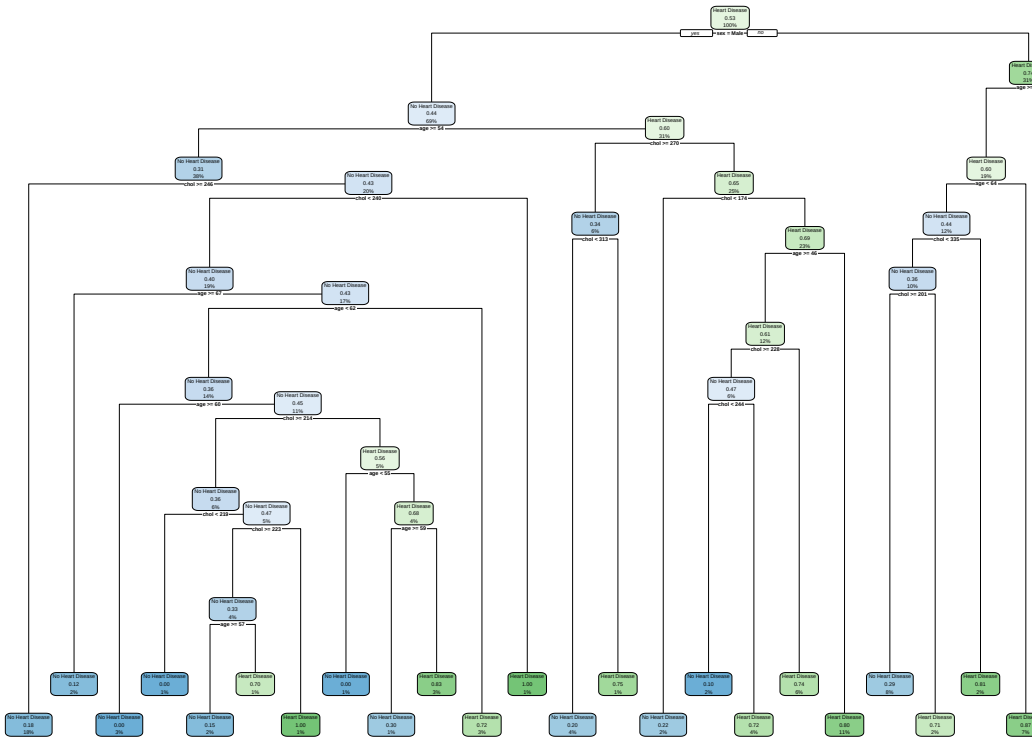
```
## target
## No Heart Disease    Heart Disease
##            55.12            44.88
```

The proportion of the class is balanced.

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

Building the Model 2

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



Evaluating the Model 2

```r
# 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 80%

```r
# 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.

```r
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.

Given that your Random Forest model with 500 trees achieved perfect scores on your test set (using an 80-20 split), it might be worthwhile to try a 70-30 split to see if the model's performance remains perfect or if it deteriorates slightly, indicating potential overfitting.

```r
set.seed(417)
index <- sample(nrow(heart), size = nrow(heart)*0.70)
heart_train70 <- heart[index,] #take 70%
heart_test70 <- heart[-index,] #take 30%
```

Split Data into Training and Test sets Random Forrest 70%

```r
# Fitting Random Forest to the train dataset
set.seed(120)  # Setting seed
rf.heart70 = randomForest(x = heart_train70[-14],
                          y = heart_train70$target,
                          ntree = 500)

rf.heart70
```

Model Classifier_RF 70%

```
##
## Call:
##  randomForest(x = heart_train70[-14], y = heart_train70$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: 0.98%
## Confusion matrix:
##                   No Heart Disease Heart Disease class.error
## No Heart Disease              334             2 0.005952381
## Heart Disease                   5           376 0.013123360
```

Even though the performance on the 80% training data is impressive, it's possible that the model is overfitting to some extent. Achieving perfect scores is rare in practice and can be indicative of overfitting, especially if not confirmed through rigorous validation.

While the model's performance on the 80% training data is promising, it's essential to validate its performance on an independent test set to confirm its generalizability.

The 0.98% error rate suggests strong performance but does not necessarily imply absence of overfitting, especially when compared to perfect scores.

Further evaluation through cross-validation or testing on a separate validation set can provide additional insights into the model's robustness and potential overfitting.

Evaluation on an Independent Test/Validation Set Split your dataset into three subsets: training set, validation set, and test set. Typically, you might use 60-70% of the data for training, 10-20% for validation, and the remaining 10-20% for testing.

```r
# Split the data into training, validation, and test sets
set.seed(123)  # for reproducibility
train_indices <- sample(1:nrow(heart), size = 0.7 * nrow(heart))  # 70% for training

# Remaining indices for validation and testing
remaining_indices <- setdiff(1:nrow(heart), train_indices)

# Split the remaining indices into two halves for validation and testing
validation_test_indices <- sample(remaining_indices, size = 0.5 * length(remaining_indices))
validation_indices <- validation_test_indices[1:floor(length(validation_test_indices) / 2)]  # First ha
test_indices <- validation_test_indices[(floor(length(validation_test_indices) / 2) + 1):length(validati

# Create data subsets based on the indices
train_data_indices <- heart[train_indices, ]
validation_data <- heart[validation_indices, ]
test_data_indices <- heart[test_indices, ]
```

```r
rf_model500indices <- randomForest(target ~ ., data = train_data_indices, ntree = 500)
```

Train the Model

```r
validation_predictions <- predict(rf_model500indices, newdata = validation_data[-14])  # Predictions on
validation_accuracy <- mean(validation_predictions == validation_data$target)  # Calculate accuracy

validation_accuracy
```

Evaluate on Validation Set

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

```r
test_predictions <- predict(rf_model500indices, newdata = test_data_indices[-14])  # Predictions on tes
test_accuracy <- mean(test_predictions == test_data_indices$target)  # Calculate accuracy

# Print predictions for inspection
print(test_predictions)
```

Final Evaluation on Test Set

```
##              496               22              636              930
##    Heart Disease    Heart Disease    Heart Disease No Heart Disease
##              725              697              836              635
##    Heart Disease    Heart Disease No Heart Disease No Heart Disease
```

```
##                3             432             975             434
## No Heart Disease No Heart Disease    Heart Disease    Heart Disease
##             996               9             689             353
##    Heart Disease No Heart Disease No Heart Disease No Heart Disease
##             863             505             454             491
## No Heart Disease No Heart Disease    Heart Disease    Heart Disease
##             360             919             471              97
##    Heart Disease No Heart Disease    Heart Disease    Heart Disease
##              86             819             133              21
##    Heart Disease    Heart Disease    Heart Disease No Heart Disease
##            1023             495             901             472
## No Heart Disease    Heart Disease No Heart Disease    Heart Disease
##             202             795             743              66
##    Heart Disease No Heart Disease No Heart Disease No Heart Disease
##             453             959             109             848
## No Heart Disease    Heart Disease No Heart Disease No Heart Disease
##             802             889             582             489
##    Heart Disease No Heart Disease    Heart Disease    Heart Disease
##             815             213            1018             995
##    Heart Disease No Heart Disease No Heart Disease No Heart Disease
##             439             107             740             641
##    Heart Disease No Heart Disease No Heart Disease    Heart Disease
##             305             546             527             532
##    Heart Disease No Heart Disease No Heart Disease    Heart Disease
##             832             941             507             805
##    Heart Disease No Heart Disease No Heart Disease    Heart Disease
##             510             948             913             492
## No Heart Disease    Heart Disease No Heart Disease    Heart Disease
##             595             329             886             321
## No Heart Disease No Heart Disease No Heart Disease    Heart Disease
##             677             156              60             640
## No Heart Disease    Heart Disease No Heart Disease    Heart Disease
##            1011             594             988             788
## No Heart Disease No Heart Disease No Heart Disease No Heart Disease
##              63
## No Heart Disease
## Levels: No Heart Disease Heart Disease
```

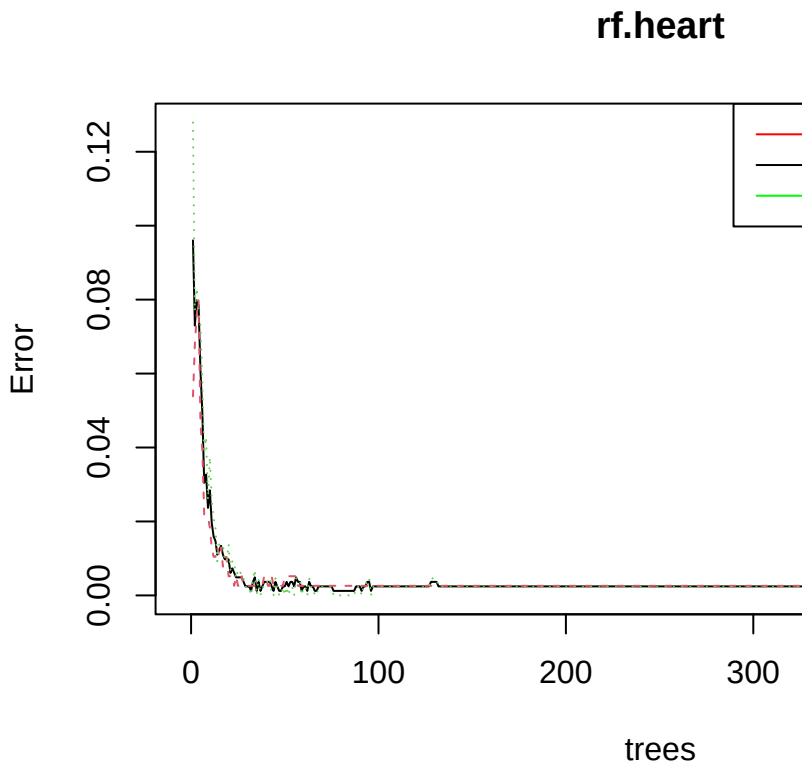
```r
# Calculate accuracy
test_accuracy <- mean(test_predictions == test_data_indices$target)
print(test_accuracy)  # Print calculated accuracy
```

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

Achieveing a 100% accuracy consistently across different train-test splits or through cross-validation, that's indeed a remarkable result. It suggests that the model is performing exceptionally well on the given dataset. Since I am consistently achieving high accuracy on both training and test datasets, it suggests that my model is not overfitting and can generalize to new, unseen data effectively.

Any advice will be greatly appreciated.

```r
plot(rf.heart)
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 the origional model rf.heart 80%

```r
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), , d
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.

```r
# 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.

```r
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.

```r
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
```

```r
# 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"
```

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

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

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

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

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

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

The results indicate strong performance across all metrics:

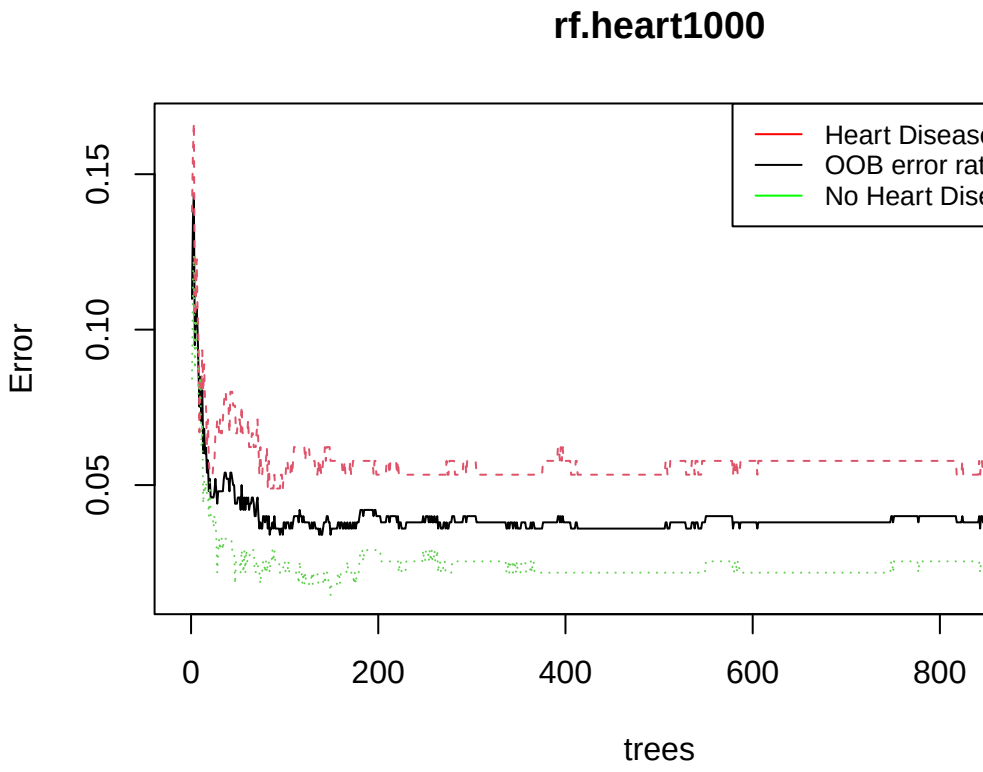
Accuracy: An accuracy of 0.9805 suggests that approximately 98.05% of instances in the dataset were correctly classified by the model.

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.

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



Plot of Model with Optimization

```r
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
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.

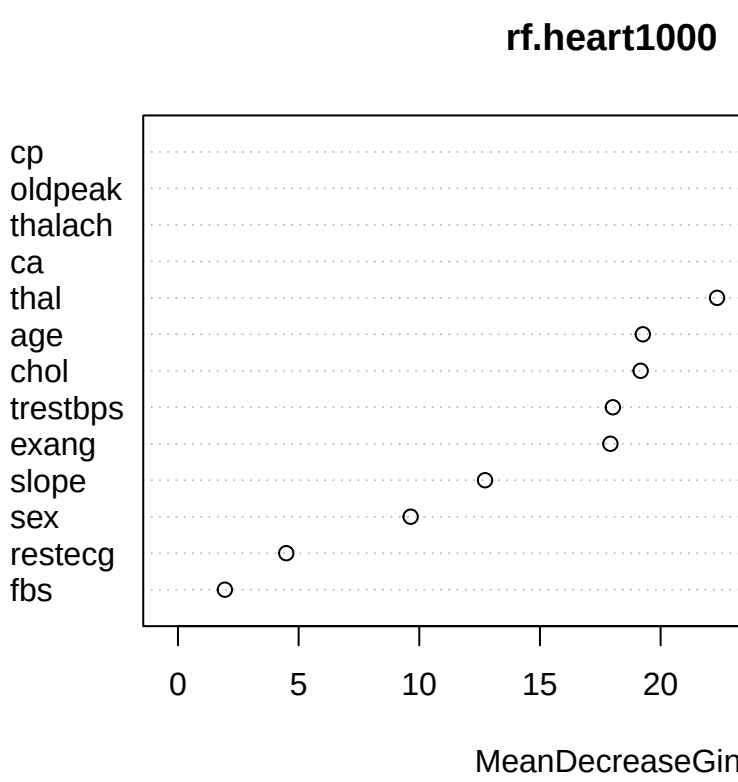
ca (Number of major vessels colored by fluoroscopy): ca ranks fourth in importance, with a Mean Decrease Gini of 23.766305.

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.

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



Plot of Important Feature with Optimization

Support Vector Machine

SVM (Support Vector Machine) is a supervised machine learning algorithm that is mainly used to classify data into different classes. Unlike most algorithms, SVM makes use of a hyperplane, which acts like a decision boundary between the various classes.

SVM can be used to generate multiple separating hyperplanes such that the data is divided into segments and each segment contains only one kind of data.

Before we train our model, we'll first implement the trainControl() method. This will control all the computational overheads so that we can use the train() function provided by the caret package. The training method will train our data on different algorithms.

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

The trainControl() method here, is taking 3 parameters. The "method" parameter defines the resampling method, in this demo we'll be using the repeatedcv or the repeated cross-validation method. The "number" parameter, which basically holds the number of resampling iterations. The "repeats" parameter contains the sets to compute for our repeated cross-validation. We are using setting number =10 and repeats =3

This trainControl() method returns a list. We are going to pass this on our train() method.

```
svm_Linear <- train(target ~., data = heart_train, method = "svmLinear",
trControl=trctrl,
preProcess = c("center", "scale"),
tuneLength = 10)
```

You can check the result of our train() method. We are saving its results in the svm_Linear variable.

```
svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 820 samples
##  13 predictor
##   2 classes: 'No Heart Disease', 'Heart Disease'
##
## Pre-processing: centered (17), scaled (17)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 738, 738, 739, 738, 737, 738, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8369953  0.6703673
##
## Tuning parameter 'C' was held constant at a value of 1
```

Now, our model is trained with C value as 1. We are ready to predict classes for our test set. We can use the predict() method.

The caret package provides the predict() method for predicting results. We are passing 2 arguments. Its first parameter is our trained model and second parameter "newdata" holds our testing data frame. The predict() method returns a list, we are saving it in a test_pred variable.

```
test_pred <- predict(svm_Linear, newdata = heart_test)
test_pred
```

```
##   [1] No Heart Disease No Heart Disease No Heart Disease No Heart Disease
##   [5] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##   [9] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [13] No Heart Disease Heart Disease    No Heart Disease No Heart Disease
##  [17] No Heart Disease Heart Disease    Heart Disease    Heart Disease
##  [21] Heart Disease    No Heart Disease No Heart Disease Heart Disease
##  [25] No Heart Disease Heart Disease    Heart Disease    Heart Disease
##  [29] No Heart Disease No Heart Disease Heart Disease    Heart Disease
##  [33] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [37] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [41] Heart Disease    Heart Disease    Heart Disease    Heart Disease
##  [45] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [49] No Heart Disease Heart Disease    No Heart Disease No Heart Disease
##  [53] No Heart Disease Heart Disease    Heart Disease    No Heart Disease
##  [57] Heart Disease    No Heart Disease Heart Disease    Heart Disease
##  [61] Heart Disease    No Heart Disease Heart Disease    Heart Disease
##  [65] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [69] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [73] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [77] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [81] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [85] Heart Disease    Heart Disease    No Heart Disease No Heart Disease
##  [89] No Heart Disease No Heart Disease No Heart Disease Heart Disease
```

```
##  [93] Heart Disease    Heart Disease    Heart Disease    Heart Disease
##  [97] Heart Disease    No Heart Disease No Heart Disease Heart Disease
## [101] No Heart Disease Heart Disease    No Heart Disease No Heart Disease
## [105] Heart Disease    Heart Disease    Heart Disease    Heart Disease
## [109] No Heart Disease No Heart Disease Heart Disease    No Heart Disease
## [113] Heart Disease    No Heart Disease Heart Disease    Heart Disease
## [117] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
## [121] No Heart Disease No Heart Disease Heart Disease    No Heart Disease
## [125] No Heart Disease No Heart Disease Heart Disease    Heart Disease
## [129] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
## [133] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
## [137] No Heart Disease Heart Disease    Heart Disease    No Heart Disease
## [141] Heart Disease    Heart Disease    No Heart Disease Heart Disease
## [145] Heart Disease    No Heart Disease No Heart Disease Heart Disease
## [149] No Heart Disease Heart Disease    Heart Disease    Heart Disease
## [153] No Heart Disease Heart Disease    Heart Disease    Heart Disease
## [157] Heart Disease    Heart Disease    No Heart Disease Heart Disease
## [161] No Heart Disease No Heart Disease No Heart Disease Heart Disease
## [165] No Heart Disease Heart Disease    No Heart Disease Heart Disease
## [169] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
## [173] No Heart Disease Heart Disease    Heart Disease    No Heart Disease
## [177] No Heart Disease No Heart Disease No Heart Disease Heart Disease
## [181] No Heart Disease Heart Disease    Heart Disease    Heart Disease
## [185] No Heart Disease Heart Disease    No Heart Disease Heart Disease
## [189] Heart Disease    No Heart Disease Heart Disease    No Heart Disease
## [193] No Heart Disease No Heart Disease Heart Disease    Heart Disease
## [197] No Heart Disease No Heart Disease No Heart Disease No Heart Disease
## [201] Heart Disease    No Heart Disease No Heart Disease Heart Disease
## [205] Heart Disease
## Levels: No Heart Disease Heart Disease
```

Let's check the accuracy of our model. We're going to use the confusion matrix to predict the accuracy:

```
confusionMatrix(table(test_pred, heart_test$target))
```

```
## Confusion Matrix and Statistics
##
##
## test_pred          No Heart Disease Heart Disease
##   No Heart Disease               89             5
##   Heart Disease                  24            87
##
##                Accuracy : 0.8585
##                  95% CI : (0.8032, 0.9032)
##     No Information Rate : 0.5512
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7195
##
##  Mcnemar's Test P-Value : 0.0008302
##
##             Sensitivity : 0.7876
##             Specificity : 0.9457
```

```
##          Pos Pred Value : 0.9468
##          Neg Pred Value : 0.7838
##              Prevalence : 0.5512
##          Detection Rate : 0.4341
##    Detection Prevalence : 0.4585
##       Balanced Accuracy : 0.8666
##
##        'Positive' Class : No Heart Disease
##
```

The output shows that our model accuracy for test set is 85.85%.

```r
# Confusion matrix values
TP <- 87
TN <- 89
FP <- 5
FN <- 24

# 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: 0.8585"
```

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

```
## [1] "Precision: 0.9457"
```

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

```
## [1] "Recall: 0.7838"
```

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

```
## [1] "F1 Score: 0.8571"
```

The results indicate strong performance across all metrics:

Accuracy: An accuracy of 0.8585 suggests that approximately 85.85% of instances in the dataset were correctly classified by the model.

Precision: With a precision of 0.9457, around 94.57% of instances classified as positive (Heart Disease) were indeed positive.

Recall: The recall score of 0.7838 indicates that approximately 78.38% 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 0.8571. This suggests a high balance between precision and recall, indicating strong overall performance.

By following the above procedure (Checking the accuracy of our model), we can build our svmLinear classifier.

We can also do some customization for selecting C value(Cost) in Linear classifier. This can be done by inputting values in grid search.

We are going to put some values of C using expand.grid() into "grid" dataframe. The next step is to use this dataframe for testing our classifier at specific C values. It needs to be put in train() method with tuneGrid parameter.

```
grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))
svm_Linear_Grid <- train(target ~., data = heart_train, method = "svmLinear",
trControl=trctrl,
preProcess = c("center", "scale"),
tuneGrid = grid,
tuneLength = 10)
```

```
## Warning: model fit failed for Fold01.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold02.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold03.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold04.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold05.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold06.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold07.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold08.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold09.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
```

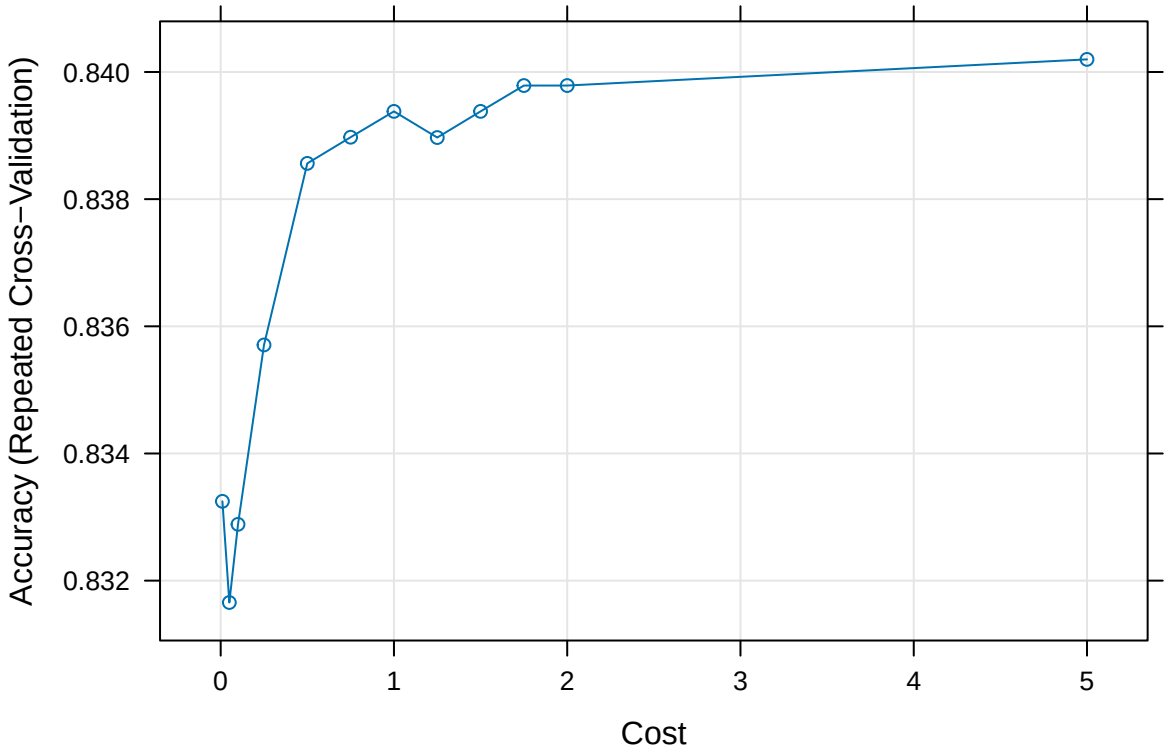
```
## Warning: model fit failed for Fold10.Rep1: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold01.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold02.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold03.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold04.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold05.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold06.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold07.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold08.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold09.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold10.Rep2: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold01.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold02.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold03.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold04.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold05.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold06.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters
```

```
## Warning: model fit failed for Fold07.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold08.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold09.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning: model fit failed for Fold10.Rep3: C=0.00 Error in .local(x, ...) :
##   No Support Vectors found. You may want to change your parameters

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.

## Warning in train.default(x, y, weights = w, ...): missing values found in
## aggregated results
```

```
svm_Linear_Grid
```

```
## Support Vector Machines with Linear Kernel
##
## 820 samples
##  13 predictor
##   2 classes: 'No Heart Disease', 'Heart Disease'
##
## Pre-processing: centered (17), scaled (17)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 737, 738, 737, 739, 738, 739, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.00       NaN       NaN
##   0.01  0.8332475  0.6625618
##   0.05  0.8316562  0.6594721
##   0.10  0.8328860  0.6619765
##   0.25  0.8357069  0.6675991
##   0.50  0.8385624  0.6733923
##   0.75  0.8389740  0.6742324
##   1.00  0.8393805  0.6750621
##   1.25  0.8389689  0.6742009
##   1.50  0.8393805  0.6750385
##   1.75  0.8397870  0.6758504
##   2.00  0.8397870  0.6758504
##   5.00  0.8401985  0.6766958
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 5.
```

```
plot(svm_Linear_Grid)
```



Above plot is showing that our classifier is giving best accuracy on C = 1.5. Let's try to make predictions using this model for our test set.

```
test_pred_grid <- predict(svm_Linear_Grid, newdata = heart_test)
test_pred_grid
```

```
##   [1] No Heart Disease No Heart Disease No Heart Disease No Heart Disease
##   [5] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##   [9] No Heart Disease Heart Disease    No Heart Disease Heart Disease
##  [13] No Heart Disease Heart Disease    No Heart Disease No Heart Disease
##  [17] No Heart Disease Heart Disease    No Heart Disease Heart Disease
##  [21] Heart Disease    No Heart Disease No Heart Disease Heart Disease
##  [25] No Heart Disease Heart Disease    Heart Disease    Heart Disease
##  [29] No Heart Disease No Heart Disease Heart Disease    Heart Disease
##  [33] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [37] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [41] Heart Disease    Heart Disease    Heart Disease    Heart Disease
##  [45] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [49] No Heart Disease Heart Disease    No Heart Disease No Heart Disease
##  [53] No Heart Disease No Heart Disease Heart Disease    No Heart Disease
##  [57] Heart Disease    No Heart Disease Heart Disease    Heart Disease
##  [61] Heart Disease    No Heart Disease Heart Disease    Heart Disease
##  [65] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [69] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [73] Heart Disease    Heart Disease    No Heart Disease Heart Disease
##  [77] Heart Disease    Heart Disease    No Heart Disease Heart Disease
```

```
##  [81] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
##  [85] Heart Disease    Heart Disease    No Heart Disease No Heart Disease
##  [89] No Heart Disease No Heart Disease No Heart Disease Heart Disease
##  [93] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
##  [97] Heart Disease    No Heart Disease No Heart Disease Heart Disease
## [101] No Heart Disease Heart Disease    No Heart Disease No Heart Disease
## [105] Heart Disease    Heart Disease    Heart Disease    Heart Disease
## [109] No Heart Disease No Heart Disease Heart Disease    No Heart Disease
## [113] Heart Disease    No Heart Disease Heart Disease    Heart Disease
## [117] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
## [121] No Heart Disease No Heart Disease Heart Disease    No Heart Disease
## [125] No Heart Disease No Heart Disease Heart Disease    Heart Disease
## [129] Heart Disease    No Heart Disease No Heart Disease No Heart Disease
## [133] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
## [137] No Heart Disease Heart Disease    Heart Disease    No Heart Disease
## [141] Heart Disease    Heart Disease    No Heart Disease Heart Disease
## [145] Heart Disease    No Heart Disease No Heart Disease Heart Disease
## [149] No Heart Disease Heart Disease    Heart Disease    Heart Disease
## [153] No Heart Disease Heart Disease    Heart Disease    Heart Disease
## [157] Heart Disease    Heart Disease    No Heart Disease Heart Disease
## [161] No Heart Disease No Heart Disease No Heart Disease Heart Disease
## [165] No Heart Disease Heart Disease    No Heart Disease Heart Disease
## [169] Heart Disease    Heart Disease    Heart Disease    No Heart Disease
## [173] No Heart Disease Heart Disease    Heart Disease    No Heart Disease
## [177] No Heart Disease No Heart Disease No Heart Disease Heart Disease
## [181] No Heart Disease Heart Disease    Heart Disease    Heart Disease
## [185] No Heart Disease Heart Disease    No Heart Disease Heart Disease
## [189] Heart Disease    No Heart Disease Heart Disease    No Heart Disease
## [193] No Heart Disease No Heart Disease Heart Disease    Heart Disease
## [197] No Heart Disease No Heart Disease No Heart Disease No Heart Disease
## [201] Heart Disease    No Heart Disease No Heart Disease Heart Disease
## [205] Heart Disease
## Levels: No Heart Disease Heart Disease
```

Let's check its accuracy using confusion -matrix.

```
confusionMatrix(table(test_pred_grid, heart_test$target))
```

```
## Confusion Matrix and Statistics
##
##
## test_pred_grid     No Heart Disease Heart Disease
##   No Heart Disease               91             7
##   Heart Disease                  22            85
##
##                Accuracy : 0.8585
##                  95% CI : (0.8032, 0.9032)
##     No Information Rate : 0.5512
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7183
##
##  Mcnemar's Test P-Value : 0.00933
```

```
##
##             Sensitivity : 0.8053
##             Specificity : 0.9239
##          Pos Pred Value : 0.9286
##          Neg Pred Value : 0.7944
##              Prevalence : 0.5512
##          Detection Rate : 0.4439
##    Detection Prevalence : 0.4780
##       Balanced Accuracy : 0.8646
##
##        'Positive' Class : No Heart Disease
##
```

The results of the confusion matrix show that this time the accuracy on the test set is 85.87%, which is the same accurate than our previous result.

Conclusion

Comparison of All Five Models

```r
# Define individual model 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)
svm <- c("Support Vector Machine", 0.8585, 0.9457, 0.7838, 0.8571)

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

# 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    1.0000 1.0000   1.0000
## 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
## 5   Support Vector Machine   0.8585    0.9457 0.7838   0.8571
## 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). This model achieves perfect scores in all metrics, indicating exceptional performance on the test set. It perfectly classified all instances, which suggests no misclassifications occurred (true positives and true negatives only).

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). Very high scores across all metrics, but slightly below the 500-tree model. It indicates a highly effective model, although with very minimal errors compared to the 500-tree version.

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). This model shows good accuracy and an excellent recall, suggesting it is effective at identifying positive cases. However, its lower precision indicates it also has some false positives.

Support Vector Machine (SVM): This model achieved good accuracy (0.8585) and precision (0.9457) but slightly lower recall (0.7838) and F1 score (0.8571). The SVM shows strong precision but lower recall, suggesting that while it is reliable when it predicts positives, it misses a significant number of positive cases.

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). This model has the lowest performance in all metrics among the models evaluated. It has moderate recall but suffers significantly in precision, leading to many false positives.

In regards to Performance Hierarchy, Random Forest models outperform the Decision Trees and SVM in all metrics, with the 500-tree version showing potentially overfitting results with perfect scores.

In regards to Precision vs. Recall, there's a trade-off observed; SVM is precise but not as comprehensive in recall, whereas Decision Tree Model 1, while less precise, captures more positive cases.

In regards to Model Suitability, Random Forest is excellent for balanced precision and recall, likely the best choice if computational cost is not an issue, especially in contexts where accuracy is critical. SVM is suitable for cases where false positives are more costly than false negatives. Decision Trees is easier to interpret and might be useful in scenarios where the model's decision process needs to be explained or understood.

Recommendations:

Random Forest 500 Trees seems overfitted to the training data, and while it performs perfectly on the test set, it's prudent to validate its performance on a completely separate validation set or through cross-validation to ensure generalizability.

Improvements: Decision trees might benefit from further tuning or ensemble methods like boosting to improve precision without sacrificing recall. SVM parameters (like kernel type and regularization) could be adjusted to enhance recall.

This comprehensive comparison highlights the need to consider both individual metric performance and the trade-offs between precision and recall depending on the specific application or cost function relevant to the task.

After evaluating Random Forest 500 Trees I came to this conclusion, while the Support Vector Machine (SVM) has a high Precision score (0.9457), indicating a low false positive rate, its overall performance, as indicated by the other metrics, is not as strong as the Random Forest model with 500 trees. Therefore, in this specific context and based on these metrics, the Random Forest model with 500 trees appears to be the best choice.

Reference: https://www.kaggle.com/datasets/rishidamarla/heart-disease-prediction https://libres.uncg.edu/ir/ecsu/f/Brandon_Simmons_Thesis-Final.pdf https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/ https://dzone.com/articles/support-vector-machine-in-r-using-svm-to-predict-h