HarvardX PH125.9x | Data Science Heart Disease Capstone Project

Anwar A. AbuAlhussein

2025-04-24

Introduction

Overview

Heart disease remains one of the leading causes of mortality worldwide. Early detection and prediction of heart disease can significantly improve patient outcomes through timely intervention and treatment. This project applies machine learning techniques to predict the presence of heart disease based on various patient attributes and clinical measurements. The dataset used in this project is the Heart Disease dataset from the kaggle, which contains medical and demographic information from patients, along with a binary classification indicating the presence or absence of heart disease. This dataset has been widely used in the medical research community and provides a valuable resource for developing predictive models.

```r
knitr::include_graphics("Heart_Disease.jpeg")
```

Project Goals

The primary objectives of this project are:

1. To explore and understand the relationships between various patient attributes and heart disease
2. To develop and evaluate machine learning models that can accurately predict heart disease
3. To compare the performance of different modeling approaches
4. To identify the most important predictors of heart disease

Required Libraries

```r
required_packages <- c(
  # Data Manipulation and Visualization
  "tidyverse",   # Data manipulation and visualization (ggplot2, dplyr, etc.)
  "GGally",      # Pairwise scatter plots and correlation plots
  "patchwork",   #combining multiple plots into a single layout.
  "RColorBrewer",         #Provides color schemes for maps
   "reshape2",           #aggregate data using just two functions: melt and 'dcast'(or 'acast').
    "vcd",               #Visualizing Categorical Data
  # Machine Learning and Evaluation
  "caret",       # Machine learning training and evaluation
  "pROC",        # ROC curve and AUC for model evaluation
  "Metrics",     # Metrics like RMSE, MAE, etc.
  "e1071",    # for skewness()
  "recipes",    # prepares data for modeling
  "tidymodels",   #for modeling and statistical analysis
  "ranger",      #A Fast Implementation of Random Forests
  "kknn",         # For knn
  "kernlab",     # For SVM
  "discrim",        # For Naive Bayes
  "xgboost",
   "vip",
  # Exploratory Data Analysis
  "DataExplorer", # EDA tools
  "corrplot",     # Correlation matrix visualization

  # Reporting and Presentation
  "broom",        # Tidy model results
  "bannerCommenter",  #Make Banner Comments with a Consistent Format
  "kableExtra"    # Making tables more presentable
)

# Loop through each package and install/load it
for (pkg in required_packages) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    tryCatch({
      install.packages(pkg, dependencies = TRUE)
    }, error = function(e) {
      message(paste("Failed to install", pkg, ":", e$message))
    })
  }
  suppressWarnings(suppressPackageStartupMessages(library(pkg, character.only = TRUE)))
}
```

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

Load Dataset

```
urlfile<-'https://raw.githubusercontent.com/ABUALHUSSEIN/Heart-disease-HarvardX-Data-Science-Capstone-ma
data <-read.csv(urlfile)
```

Data exploration and visualization

```
View(data)
colnames(data)
```

```
##  [1] "age"      "sex"      "cp"       "trestbps" "chol"     "fbs"
##  [7] "restecg"  "thalach"  "exang"    "oldpeak"  "slope"    "ca"
## [13] "thal"     "target"
```

```
glimpse(data)
```

```
## Rows: 303
## Columns: 14
## $ age      <int> 63, 37, 41, 56, 57, 57, 56, 44, 52, 57, 54, 48, 49, 64, 58, 5~
## $ sex      <int> 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1~
## $ cp       <int> 3, 2, 1, 1, 0, 0, 1, 1, 2, 2, 0, 2, 1, 3, 3, 2, 2, 3, 0, 3, 0~
## $ trestbps <int> 145, 130, 130, 120, 120, 140, 140, 120, 172, 150, 140, 130, 1~
## $ chol     <int> 233, 250, 204, 236, 354, 192, 294, 263, 199, 168, 239, 275, 2~
## $ fbs      <int> 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ restecg  <int> 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1~
## $ thalach  <int> 150, 187, 172, 178, 163, 148, 153, 173, 162, 174, 160, 139, 1~
## $ exang    <int> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
## $ oldpeak  <dbl> 2.3, 3.5, 1.4, 0.8, 0.6, 0.4, 1.3, 0.0, 0.5, 1.6, 1.2, 0.2, 0~
## $ slope    <int> 0, 0, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1~
## $ ca       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0~
## $ thal     <int> 1, 2, 2, 2, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3~
## $ target   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

```
head(data, 10)
```

```
##    age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 1   63   1  3      145  233   1       0     150     0     2.3     0  0    1
## 2   37   1  2      130  250   0        1      187      0      3.5      0  0    2
## 3   41   0  1      130  204   0        0      172      0      1.4      2  0    2
## 4   56   1  1      120  236   0        1      178      0      0.8      2  0    2
## 5   57   0  0      120  354   0        1      163      1      0.6      2  0    2
## 6   57   1  0      140  192   0        1      148      0      0.4      1  0    1
## 7   56   0  1      140  294   0        0      153      0      1.3      1  0    2
## 8   44   1  1      120  263   0        1      173      0      0.0      2  0    3
## 9   52   1  2      172  199   1        1      162      0      0.5      2  0    3
## 10  57   1  2      150  168   0        1      174      0      1.6      2  0    2
##    target
## 1       1
## 2       1
## 3       1
## 4       1
## 5       1
## 6       1
## 7       1
## 8       1
## 9       1
## 10      1
```

```
tail(data,10)
```

```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 294  67   1  2      152  212   0       0     150     0     0.8     1  0    3
## 295  44   1  0      120  169   0       1     144     1     2.8     0  0    1
## 296  63   1  0      140  187   0       0     144     1     4.0     2  2    3
## 297  63   0  0      124  197   0       1     136     1     0.0     1  0    2
## 298  59   1  0      164  176   1       0      90     0     1.0     1  2    1
## 299  57   0  0      140  241   0       1     123     1     0.2     1  0    3
## 300  45   1  3      110  264   0       1     132     0     1.2     1  0    3
## 301  68   1  0      144  193   1       1     141     0     3.4     1  2    3
## 302  57   1  0      130  131   0       1     115     1     1.2     1  1    3
## 303  57   0  1      130  236   0       0     174     0     0.0     1  1    2
##     target
## 294      0
## 295      0
## 296      0
## 297      0
## 298      0
## 299      0
## 300      0
## 301      0
## 302      0
## 303      0
```

The dataset contains 303 observations (rows) and 14 variables (columns)

Dataset Description:

age: age of the patient (years)

sex:describe the gender of person (1 = male; 0 = female)

cp: chest pain type (Value 0: typical angina) (Value 1: atypical angina) (Value 2: non-anginal pain) (Value 3: asymptomatic)

trestbps: resting blood pressure (in mm Hg on admission to the hospital)

chol: serum cholestoral in mg/dl

fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

restecg: resting electrocardiographic results (Value 0: normal) (Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) (Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria)

thalach: maximum heart rate achieved

exang: exercise induced angina (1 = yes; 0 = no)

oldpeak = ST depression induced by exercise relative to rest

slope: the slope of the peak exercise ST segment (Value 0: upsloping) (Value 1: flat) (Value 2: downsloping)

ca: number of major vessels (0-3) colored by flourosopy

thal: A blood disorder called thalassemia 0 = error (in the original dataset 0 maps to NaN's) 1 = fixed defect 2 = normal 3 = reversable defect

target: variable indicating presence of heart disease (1 = Heart Disease, 0 = No Heart Disease)

Data pre-processing

```
total_missing <- sum(is.na(data))
colSums(is.na(data))
```

```
##      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
```

```
cat("Total missing values in the dataset:", total_missing, "\n")
```

```
## Total missing values in the dataset: 0
```

As we can see from the output above:

The dataset has a total of 303 rows and there are no missing values.

There are a total of 13 features and 1 target variable.

Notes from the discussion forum of the dataset:

data 93, 159, 164, 165 and 252 have ca=4 which is incorrect. In the original Cleveland dataset they are NaNs.

data 49 and 282 have thal = 0, also incorrect.

They are also NaNs in the original dataset.

Action: Drop the faulty data! (7 data entry will be dropped)

```
#Drop by condition (more reliable if row indices might not match)
heart_data_cleaned <- data[!(data$ca == 4 | data$thal == 0), ]

# Print the new number of rows
cat("Cleaned dataset size:", nrow(heart_data_cleaned), "rows\n")
```

```
## Cleaned dataset size: 296 rows
```

```
cat("Removed", nrow(data) - nrow(heart_data_cleaned), "rows with incorrect values\n")
```

```
## Removed 7 rows with incorrect values
```

```
view(heart_data_cleaned)
cat("The length of the data now is:", nrow(heart_data_cleaned) , "instead of:", nrow(data))
```

```
## The length of the data now is: 296 instead of: 303
```

The medical and technical terms are already complex, their shortened versions only add to the challenge. To enhance readability, we will rename the columns using information from the UCL data repository.

Additionally, we will replace numerical category codes (e.g., 0, 1, 2…) with their corresponding medical terms, such as 'atypical angina' and 'typical angina,' to make the data more intuitive.

Note: I borrowed Rob Harrand's idea of re-naming the columns.

```
data <- heart_data_cleaned %>%
  rename(
    chest_pain_type = cp,
    resting_blood_pressure = trestbps,
    cholesterol = chol,
    fasting_blood_sugar = fbs,
    resting_electrocardiogram = restecg,
    max_heart_rate_achieved = thalach,
    exercise_induced_angina = exang,
    st_depression = oldpeak,
    st_slope = slope,
    num_major_vessels = ca,
    thalassemia = thal
  )
```

```
# Check the new column names
colnames(data)
```

```
##  [1] "age"                       "sex"
##  [3] "chest_pain_type"           "resting_blood_pressure"
##  [5] "cholesterol"               "fasting_blood_sugar"
##  [7] "resting_electrocardiogram" "max_heart_rate_achieved"
##  [9] "exercise_induced_angina"   "st_depression"
## [11] "st_slope"                  "num_major_vessels"
## [13] "thalassemia"               "target"
```

Outlier Treatment

check for outliers using the IQR method for the continuous features:

```r
# List of continuous features
num_feats <- c('age', 'cholesterol', 'resting_blood_pressure',
               'max_heart_rate_achieved', 'st_depression', 'num_major_vessels')

# Subset the dataframe to include only continuous features
df_cont <- data[, num_feats]

# Compute Q1, Q3, and IQR for each continuous feature
Q1 <- apply(df_cont, 2, quantile, 0.25, na.rm = TRUE)
Q3 <- apply(df_cont, 2, quantile, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1

# Identify outliers for each feature
outliers <- sapply(names(df_cont), function(col) {
  lower_bound <- Q1[col] - 1.5 * IQR[col]
  upper_bound <- Q3[col] + 1.5 * IQR[col]
  df_cont[[col]] < lower_bound | df_cont[[col]] > upper_bound
})

# Convert outliers matrix to data frame for better readability
outliers_df <- as.data.frame(outliers)

# Count of outliers per feature
outliers_count <- colSums(outliers_df, na.rm = TRUE)

print(outliers_count)
```

```
##                     age             cholesterol  resting_blood_pressure
##                       0                       5                       9
## max_heart_rate_achieved           st_depression       num_major_vessels
##                       1                       5                      20
```

Upon identifying outliers for the specified continuous features, we found the following:

- age: No outliers
- cholesterol: 5 outliers
- resting_blood_pressure: 9 outliers
- max_heart_rate_achieved: 1 outliers
- st_depression: 5 outliers
- num_major_vessels:20 outliers

Data Transformation

To enhance readability and facilitate better interpretation later on, we will replace the coded numerical categories with their corresponding medical meanings.

```r
data <- data %>%
  mutate(
    target = case_when(
      target== 0 ~ "No Heart Disease",
      target == 1 ~ "Heart Disease"
    )%>% as.factor(),
    sex = case_when(
      sex == 0 ~ "female",
      sex == 1 ~ "male"
    )%>% as.factor(),
    chest_pain_type = case_when(
      chest_pain_type == 0 ~ "typical angina",
      chest_pain_type == 1 ~ "atypical angina",
      chest_pain_type == 2 ~ "non-anginal pain",
      chest_pain_type == 3 ~ "asymptomatic"
    )%>% as.factor(),
    fasting_blood_sugar = case_when(
      fasting_blood_sugar == 0 ~ "lower than 120mg/ml",
      fasting_blood_sugar == 1 ~ "greater than 120mg/ml"
    )%>% as.factor(),
    resting_electrocardiogram = case_when(
      resting_electrocardiogram == 0 ~ "normal",
      resting_electrocardiogram == 1 ~ "ST-T wave abnormality",
      resting_electrocardiogram == 2 ~ "left ventricular hypertrophy"
    )%>% as.factor(),
    exercise_induced_angina = case_when(
      exercise_induced_angina == 0 ~ "no",
      exercise_induced_angina == 1 ~ "yes"
    ) %>% as.factor(),
    st_slope = case_when(
      st_slope == 0 ~ "upsloping",
      st_slope == 1 ~ "flat",
      st_slope == 2 ~ "downsloping"
    ) %>% as.factor(),
    thalassemia = case_when(
      thalassemia == 1 ~ "fixed defect",
      thalassemia == 2 ~ "normal",
      thalassemia == 3 ~ "reversable defect"
    ) %>% as.factor()
  )
```

```r
str(data)
```

```
## 'data.frame':    296 obs. of  14 variables:
##  $ age                      : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex                      : Factor w/ 2 levels "female","male": 2 2 1 2 1 2 1 2 2 2 ...
##  $ chest_pain_type          : Factor w/ 4 levels "asymptomatic",..: 1 3 2 2 4 4 2 2 3 3 ...
##  $ resting_blood_pressure   : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ cholesterol              : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fasting_blood_sugar      : Factor w/ 2 levels "greater than 120mg/ml",..: 1 2 2 2 2 2 2 2 1 2 ...
##  $ resting_electrocardiogram: Factor w/ 3 levels "left ventricular hypertrophy",..: 2 3 2 3 3 3 2 3
##  $ max_heart_rate_achieved  : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exercise_induced_angina  : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 1 1 ...
##  $ st_depression           : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ st_slope                : Factor w/ 3 levels "downsloping",..: 3 3 1 1 1 2 2 1 1 1 ...
##  $ num_major_vessels       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thalassemia             : Factor w/ 3 levels "fixed defect",..: 1 2 2 2 2 1 2 3 3 2 ...
##  $ target                  : Factor w/ 2 levels "Heart Disease",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
head(data)
```

```
##   age    sex  chest_pain_type resting_blood_pressure cholesterol
## 1  63   male     asymptomatic                    145         233
## 2  37   male non-anginal pain                    130         250
## 3  41 female  atypical angina                    130         204
## 4  56   male  atypical angina                    120         236
## 5  57 female   typical angina                    120         354
## 6  57   male   typical angina                    140         192
##     fasting_blood_sugar resting_electrocardiogram max_heart_rate_achieved
## 1 greater than 120mg/ml                    normal                     150
## 2   lower than 120mg/ml     ST-T wave abnormality                     187
## 3   lower than 120mg/ml                    normal                     172
## 4   lower than 120mg/ml     ST-T wave abnormality                     178
## 5   lower than 120mg/ml     ST-T wave abnormality                     163
## 6   lower than 120mg/ml     ST-T wave abnormality                     148
##   exercise_induced_angina st_depression    st_slope num_major_vessels
## 1                      no           2.3   upsloping                 0
## 2                      no           3.5   upsloping                 0
## 3                      no           1.4 downsloping                 0
## 4                      no           0.8 downsloping                 0
## 5                     yes           0.6 downsloping                 0
## 6                      no           0.4        flat                 0
##    thalassemia        target
## 1 fixed defect Heart Disease
## 2       normal Heart Disease
## 3       normal Heart Disease
## 4       normal Heart Disease
## 5       normal Heart Disease
## 6 fixed defect Heart Disease
```

```r
View(data)
```

As we have seen above there are three datatypes i.e int,chr and num. Let's group them according to type.

```r
# numerical fearures 6
num_feats <- c('age', 'cholesterol', 'resting_blood_pressure',
               'max_heart_rate_achieved', 'st_depression', 'num_major_vessels')

# Binary categorical features
bin_feats <- c('sex', 'fasting_blood_sugar', 'exercise_induced_angina', 'target')

# Nominal (multi-category) categorical features
nom_feats <- c('chest_pain_type', 'resting_electrocardiogram',
               'st_slope', 'thalassemia')

# Combine binary and nominal categorical features
cat_feats <- c(nom_feats, bin_feats)
```

```r
# Generate summary statistics for numerical features
summary(data[, num_feats])
```

```
##       age         cholesterol    resting_blood_pressure max_heart_rate_achieved
##  Min.   :29.00   Min.   :126.0   Min.   : 94.0          Min.   : 71.0
##  1st Qu.:48.00   1st Qu.:211.0   1st Qu.:120.0          1st Qu.:133.0
##  Median :56.00   Median :242.5   Median :130.0          Median :152.5
##  Mean   :54.52   Mean   :247.2   Mean   :131.6          Mean   :149.6
##  3rd Qu.:61.00   3rd Qu.:275.2   3rd Qu.:140.0          3rd Qu.:166.0
##  Max.   :77.00   Max.   :564.0   Max.   :200.0          Max.   :202.0
##  st_depression   num_major_vessels
##  Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.800   Median :0.0000
##  Mean   :1.059   Mean   :0.6791
##  3rd Qu.:1.650   3rd Qu.:1.0000
##  Max.   :6.200   Max.   :3.0000
```

Statistical summary of the numerical features:

Age:

The average age in the dataset is 54.5 years.

The youngest individual is 29 years old, while the oldest is 77 years.

Cholesterol:

The mean cholesterol level is 247.2 mg/dL.

The highest recorded level is 564 mg/dL, and the lowest is 126 mg/dL.

Resting Blood Pressure:

The average resting blood pressure is 131.6 mmHg.

The highest recorded value is 200 mmHg, while the lowest is 94 mmHg.

Maximum Heart Rate Achieved:

The mean maximum heart rate is 149.6 bpm.

The highest recorded value is 202 bpm, and the lowest is 71 bpm.

ST Depression:

The average ST depression value is 1.059.

The maximum recorded value is 6.2, while the minimum is 0.

Number of Major Blood Vessels:

The dataset records a minimum of 0 and a maximum of 3 major blood vessels.

The average number of major blood vessels is 0.6791.

Exploratory Data Analysis

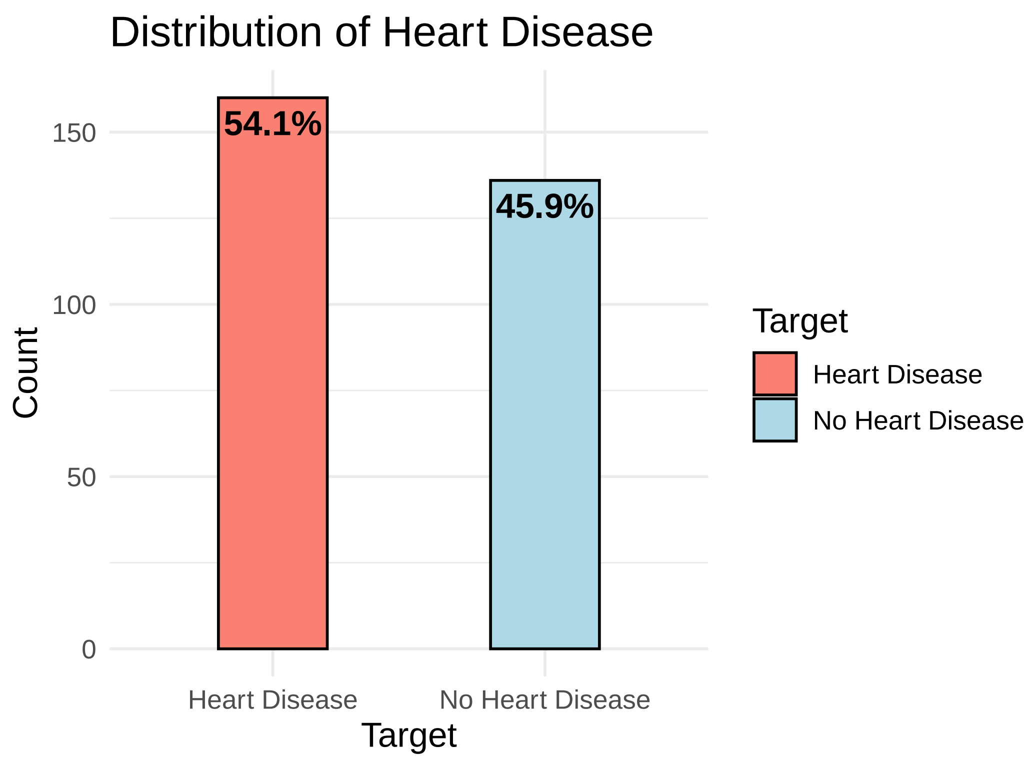
Before building predictive models, we conducted an exploratory data analysis to understand the distribution of variables and their relationships with heart disease.

This included:

1. Examining the distribution of the target variable (heart disease)
2. Creating visualizations to better understand the data

Target Variable Distribution

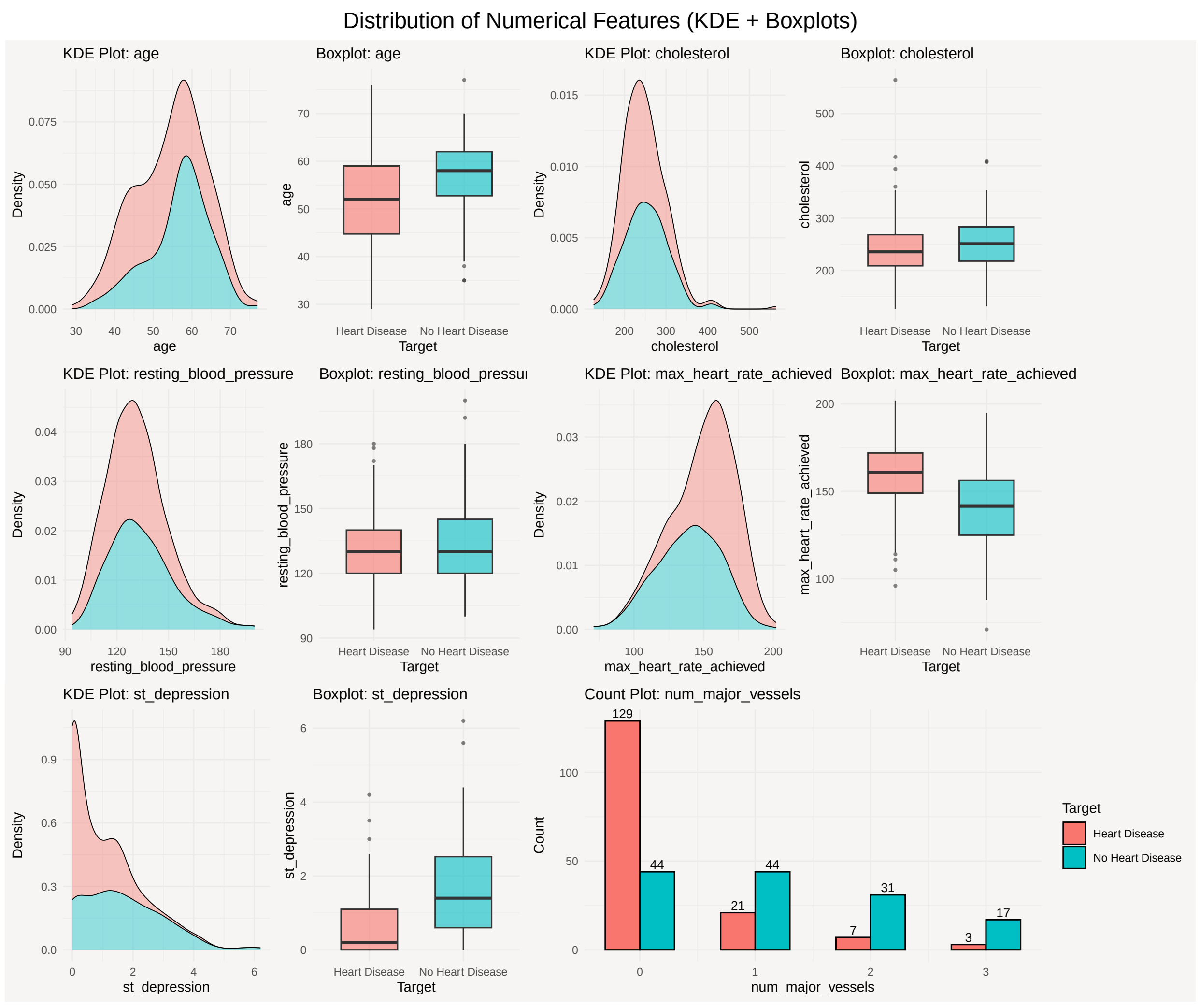
```r
# Compute percentages
data_perc <- data %>%
  count(target) %>%
  mutate(percentage = (n / sum(n)) * 100)

# Bar plot with percentage labels inside the bars
ggplot(data_perc, aes(x = factor(target), y = n, fill = factor(target))) +
  geom_bar(stat = "identity", width = 0.4, color = "black") +
  geom_text(aes(label = paste0(round(percentage, 1), "%")),
            vjust = 1.5,                # Move label inside the bar
            color = "black",            # Contrast color
            size = 4.5,
            fontface = "bold") +
  labs(title = "Distribution of Heart Disease", x = "Target", y = "Count") +
  scale_fill_manual(values = c("salmon", "lightblue"),
                    name = "Target",
                    labels = c("Heart Disease", "No Heart Disease")) +
  scale_x_discrete(labels = c("Heart Disease", "No Heart Disease")) +
  theme_minimal() +
  theme(text = element_text(size = 13))
```



The dataset is relatively balanced with respect to the target variable (heart disease), with 54.1% of patients having heart disease and 45.9% not having heart disease.

```r
# Create a list to store plots
plots <- list()

for (col in num_feats) {

  if (col == "num_major_vessels") {
    # Count plot only for this discrete variable
    p <- ggplot(data, aes_string(x = col, fill = "factor(target)")) +
      geom_bar(position = "dodge", width = 0.6, color = "black") +
      geom_text(stat = "count", aes(label = ..count..),
                position = position_dodge(width = 0.6),
                vjust = -0.3, size = 3) +
      labs(title = paste("Count Plot:", col), x = col, y = "Count", fill = "Target") +
      theme_minimal(base_size = 10) +
      theme(
        plot.background = element_rect(fill = "#F6F5F4", color = NA),
        panel.background = element_rect(fill = "#F6F5F4", color = NA),
        legend.position = "right",
        plot.title = element_text(size = 11)
      )
    plots[[col]] <- p

  } else {
    # KDE Plot
    p_density <- ggplot(data, aes_string(x = col, fill = "factor(target)")) +
      geom_density(alpha = 0.4, position = "stack", size = 0.3) +
      labs(title = paste("KDE Plot:", col), x = col, y = "Density", fill = "Target") +
      theme_minimal(base_size = 10) +
      theme(
        plot.background = element_rect(fill = "#F6F5F4", color = NA),
        panel.background = element_rect(fill = "#F6F5F4", color = NA),
        legend.position = "none",
        plot.title = element_text(size = 11)
      )

    # Boxplot
    p_box <- ggplot(data, aes_string(x = "factor(target)", y = col, fill = "factor(target)")) +
      geom_boxplot(outlier.shape = 16, outlier.size = 1, width = 0.6, alpha = 0.6) +
      labs(title = paste("Boxplot:", col), x = "Target", y = col, fill = "Target") +
      theme_minimal(base_size = 10) +
      theme(
        plot.background = element_rect(fill = "#F6F5F4", color = NA),
        panel.background = element_rect(fill = "#F6F5F4", color = NA),
        legend.position = "none",
        plot.title = element_text(size = 11)
      )

    combined <- p_density + p_box
    plots[[col]] <- combined
  }
}
```

```r
# Combine all plots
final_plot <- wrap_plots(plots, ncol = 2) +
  plot_annotation(
    title = "Distribution of Numerical Features (KDE + Boxplots)",
    theme = theme(plot.title = element_text(size = 16, hjust = 0.5))
  )


final_plot
```



Based on the plot:

Most discriminative features for heart disease seem to be:

1. **max_heart_rate_achieved**

   Target = 1 (heart disease) group tends to have lower maximum heart rate.

2. **st_depression**

   Clear right-skewed distribution in KDE. Heart disease patients (target == 1) show higher ST depression, which typically reflects abnormal ECG response to stress.

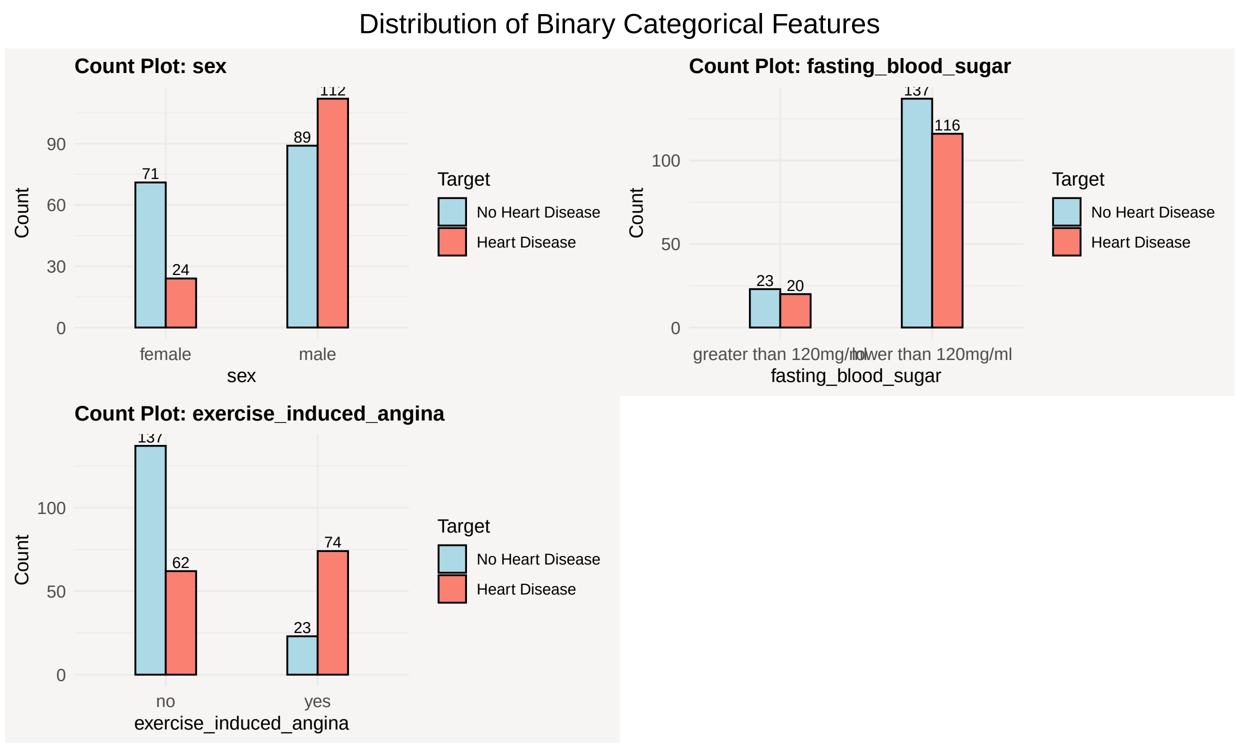
3. **num_major_vessels**
   The count plot shows a strong difference in the number of major vessels colored by fluoroscopy. Higher counts (2 or 3 vessels) are much more common in patients without heart disease (target == 0), indicating that fewer colored vessels may signal higher risk.

Moderately Important / Ambiguous Features:

1. Age Both groups have a peak around 55-60 years, but patients without heart disease (target = 0) show slightly higher densities in older age brackets.
2. Resting Blood Pressure Distributions are similar across both classes, with a central tendency near 130 mm Hg. Minimal separation implies this feature may have low predictive power on its own for classification.
3. Cholesterol Heart disease patients (target = 1) are more concentrated in the lower cholesterol range (200-300). Surprisingly, patients without heart disease have a broader and higher cholesterol distribution.

Categorical Features

```r
bin_feats <- c('sex', 'fasting_blood_sugar', 'exercise_induced_angina')
plots <- list()

for (col in bin_feats) {
  p <- ggplot(data, aes_string(x = col, fill = "factor(target)")) +
    geom_bar(position = "dodge", width = 0.4, color = "black") +
    geom_text(stat = "count", aes(label = ..count..),
              position = position_dodge(width = 0.4),
              vjust = -0.3, size = 3) +
    labs(
      title = paste("Count Plot:", col),
      x = col,
      y = "Count",
      fill = "Target"
    ) +
    scale_fill_manual(values = c("lightblue", "salmon"),
                      labels = c("No Heart Disease", "Heart Disease")) +
    theme_minimal(base_size = 11) +
    theme(
      plot.background = element_rect(fill = "#F6F5F4", color = NA),
      panel.background = element_rect(fill = "#F6F5F4", color = NA),
      legend.position = "right",
      axis.title = element_text(size = 11),
      axis.text = element_text(size = 10),
      plot.title = element_text(size = 12, face = "bold")
    )

  plots[[col]] <- p
}

final_plot <- wrap_plots(plots, ncol = 2) +
  plot_annotation(
    title = "Distribution of Binary Categorical Features",
    theme = theme(plot.title = element_text(size = 16, hjust = 0.5))
  )

print(final_plot)
```

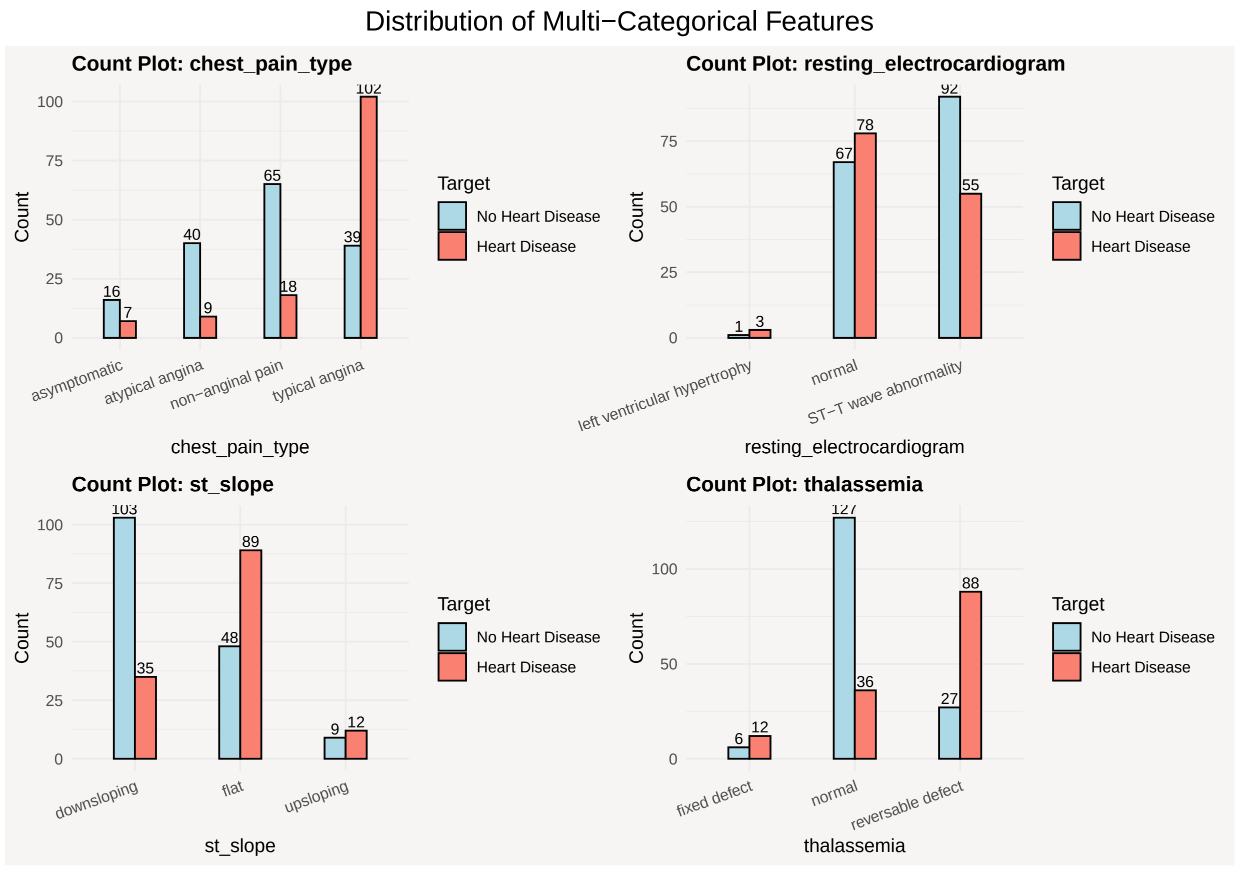


The plot shows the following observations:

- Males have a higher prevalence of heart disease compared to females.
- Fasting blood sugar levels (less than 120mg/ml) are strongly associated with heart disease.
- Patients without exercise-induced angina are more likely to have heart disease.

```
# Nominal (multi-category) categorical features
nom_feats <- c('chest_pain_type', 'resting_electrocardiogram',
               'st_slope', 'thalassemia')

plots <- list()

for (col in nom_feats) {
  p <- ggplot(data, aes_string(x = col, fill = "factor(target)")) +
    geom_bar(position = "dodge", width = 0.4, color = "black") +
    geom_text(stat = "count", aes(label = ..count..),
              position = position_dodge(width = 0.4),
              vjust = -0.3, size = 3) +
    labs(
      title = paste("Count Plot:", col),
      x = col,
```

```r
      y = "Count",
      fill = "Target"
    ) +
    scale_fill_manual(values = c("lightblue", "salmon"),
                      labels = c("No Heart Disease", "Heart Disease")) +
    theme_minimal(base_size = 11) +
    theme(
      plot.background = element_rect(fill = "#F6F5F4", color = NA),
      panel.background = element_rect(fill = "#F6F5F4", color = NA),
      legend.position = "right",
      axis.text.x = element_text(angle = 20, hjust = 1),  # Tilt x-axis labels for clarity
      plot.title = element_text(size = 12, face = "bold")
    )

  plots[[col]] <- p
}

final_plot <- wrap_plots(plots, ncol = 2) +
  plot_annotation(
    title = "Distribution of Multi-Categorical Features",
    theme = theme(plot.title = element_text(size = 16, hjust = 0.5))
  )

print(final_plot)
```



The plot shows the following observations:

1. Atypical angina and non-anginal pain are strongly associated with heart disease, while typical angina is more common among patients without heart disease.
2. ST-T wave abnormalities are highly indicative of heart disease, while normal ECGs show a balanced distribution.
3. Downsloping ST segments are strongly associated with heart disease, while flat ST segments are more common among patients without heart disease.
4. Normal thalassemia results are more common among patients with heart disease, while reversible defects are more common among patients without heart disease.

Transforming Skewed Features:correct transformations based on dataset

let's analyze the numeric variables in dataset, check for skewness, and then recommend the best transformation (log, Box-Cox, Yeo-Johnson, or none) for each.

```r
## Step 1: Identify numeric variables
num_vars <- data %>% select(where(is.numeric))

## Step 2: Check for zeros or negatives
check_neg_zero <- num_vars %>%
  summarise_all(list(
    min = ~min(., na.rm = TRUE),
    zero = ~sum(. == 0, na.rm = TRUE)
  )) %>%
  pivot_longer(everything(), names_to = c("variable", ".value"), names_sep = "_")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 8 rows [2, 4, 5, 6, 8, 10,
## 11, 12].
```

```r
## Step 3: Calculate skewness
skews <- num_vars %>%
  summarise(across(everything(), ~ skewness(., na.rm = TRUE))) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "skewness")

## Step 4: Merge & classify
outlier_summary <- left_join(skews, check_neg_zero, by = "variable") %>%
  mutate(
    transformation = case_when(
      min <= 0 ~ "Yeo-Johnson",  # if any value ??? 0
      abs(skewness) < 0.5 ~ "None",  # roughly symmetric
      abs(skewness) < 1.0 ~ "Box-Cox",
      TRUE ~ "Log"
    )
  )

print(outlier_summary)
```

```
## # A tibble: 6 x 9
##   variable      skewness   min blood heart depression major  zero transformation
##   <chr>            <dbl> <int> <int> <int>      <dbl> <int> <int> <chr>
## 1 age             -0.212    29    NA    NA         NA    NA     0 None
## 2 resting_bloo~    0.704    NA    NA    NA         NA    NA    NA Box-Cox
## 3 cholesterol       1.12     126     NA     NA          NA     NA      0 Log
## 4 max_heart_ra~    -0.526     NA     NA     NA          NA     NA     NA Box-Cox
## 5 st_depression     1.23      NA     NA     NA          NA     NA     NA Log
## 6 num_major_ve~     1.16      NA     NA     NA          NA     NA     NA Log
```

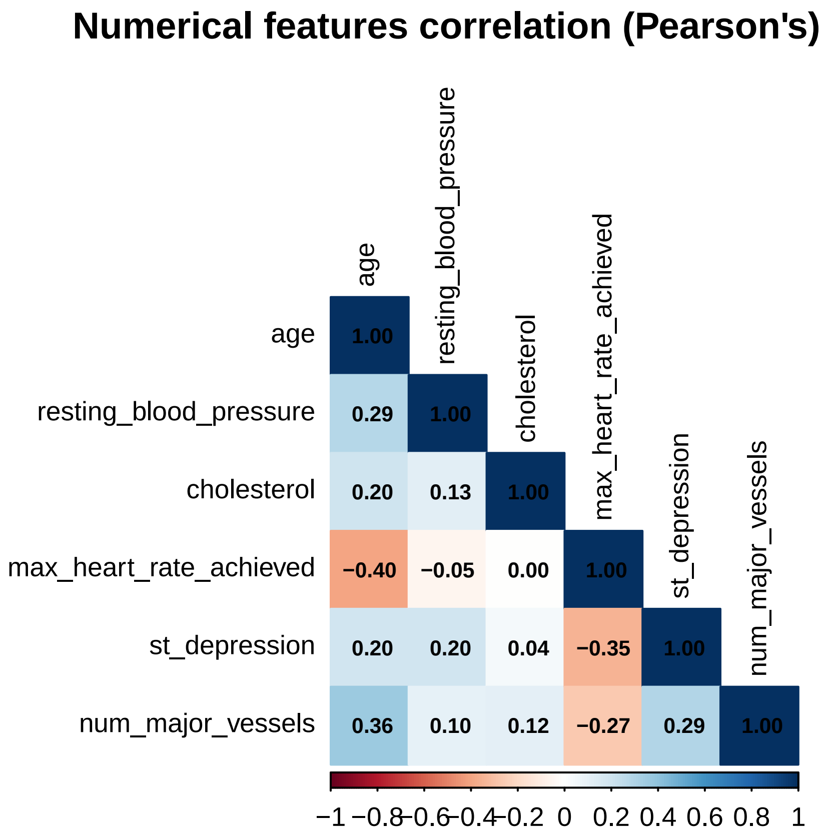
we'll focus on applying transformations like Box-Cox and log in the subsequent steps to reduce the impact of outliers and make the data more suitable for modeling.

Study the correlation

Pearson's correlation:

We use Pearson's correlation to measure that quantifies the strength and direction of a linear relationship between two continuous variables.

```r
numeric_vars <- data %>%
  select_if(is.numeric)

# Compute the correlation matrix
cor_matrix <- cor(numeric_vars, use = "complete.obs")

# Plot the correlation matrix
corrplot(cor_matrix, method = "color", type = "lower",
         tl.col = "black", tl.cex = 0.8, number.cex = 0.7, addCoef.col = "black")
title("Numerical features correlation (Pearson's)", line = 2.5)
```



The heatmap reveals moderate correlations between age, maximum heart rate achieved, and ST depression, while many other relationships are weak or negligible.

Point biserial correlation:

Useful for binary classification problems to see the correlation between numeric features and the binary target.

Calculate point-biserial correlation Even though it's called "point-biserial," it's mathematically equivalent to Pearson correlation when one variable is binary (0/1).

```
data1 <- data %>%
  mutate(target_numeric = recode(target, "No Heart Disease" = 0, "Heart Disease" = 1))

str(data1$target_numeric)
```

```
##  num [1:296] 1 1 1 1 1 1 1 1 1 1 ...
```

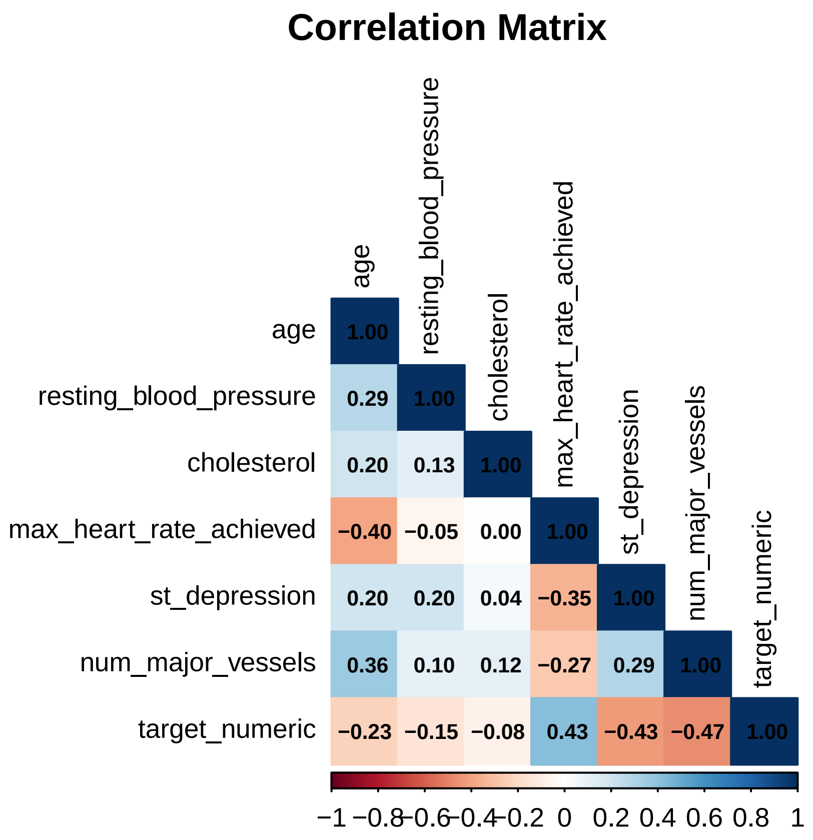
```
# Check the levels first
print(levels(data1$target))
```

```
## [1] "Heart Disease"    "No Heart Disease"
```

```
data1 %>% count(target)
```

```
##             target   n
## 1    Heart Disease 160
## 2 No Heart Disease 136
```

```
# --- Step 3: Select numeric features (excluding target_numeric) ---
numeric_features <- data1 %>%
  select_if(is.numeric) %>%
  names() %>%
  setdiff("target_numeric")

# --- Step 4: Compute point-biserial correlations ---
correlation_results <- sapply(numeric_features, function(feature) {
  test_result <- cor.test(data1[[feature]], data1$target_numeric, method = "pearson")
  test_result$estimate  # This is the correlation coefficient
})



# --- Step 5: Create a correlation matrix including target_numeric ---
# Select only the numeric columns you care about
num_data <- data1 %>%
  select(all_of(c(numeric_features, "target_numeric")))

cor_matrix <- cor(num_data, use = "complete.obs")

# --- Step 6: Plot the correlation matrix ---

corrplot(cor_matrix, method = "color", type = "lower",
         tl.col = "black", tl.cex = 0.8, number.cex = 0.7, addCoef.col = "black")
title("Correlation Matrix", line = 2.5)
```



The correlation matrix reveals varying degrees and directions of linear relationships between the variables. Notably, max_heart_rate_achieved, st_depression, and num_major_vessels show the strongest correlations with the target_numeric variable, with max_heart_rate_achieved being positively correlated and the other two being negatively correlated.

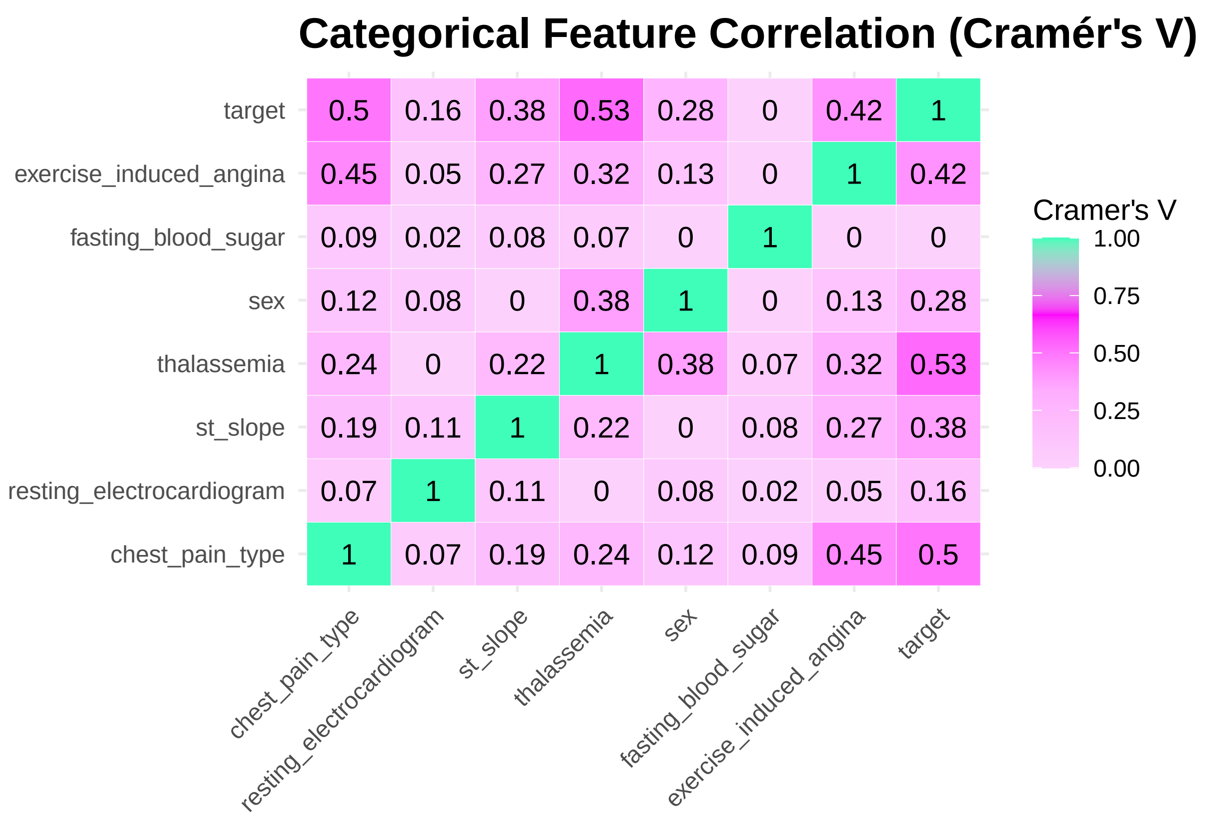
The plot shows the following observations:

1. max_heart_rate_achieved (0.43) Moderate positive correlation: Higher max heart rate is associated with higher chance of heart disease in this dataset.
2. st_depression (-0.43) Moderate negative correlation: Higher ST depression is linked with lower probability of disease.
3. num_major_vessels (-0.47) Strongest negative correlation: More blocked vessels are associated with lower chance of heart disease in this dataset.
4. age (-0.23) Weak negative correlation: Older age slightly linked to absence of heart disease (possibly due to survivorship or treatment bias).
5. resting_blood_pressure (-0.15) Very weak negative correlation.
6. cholesterol -0.08 Essentially no correlation. Cholesterol alone may not predict heart disease effectively.

Categorical Features Correlation (Cramer's V):

Cramér's V is a statistical measure used to assess the strength of association between two categorical variables.

```r
# Define Cramer's V function
cramers_v <- function(x, y) {
  tbl <- table(x, y)
  chi2 <- suppressWarnings(chisq.test(tbl, correct = FALSE))
  chi2_stat <- chi2$statistic
  n <- sum(tbl)
  phi2 <- chi2_stat / n
  r <- nrow(tbl)
  k <- ncol(tbl)
  phi2_corr <- max(0, phi2 - ((k - 1) * (r - 1)) / (n - 1))
  r_corr <- r - ((r - 1)^2) / (n - 1)
  k_corr <- k - ((k - 1)^2) / (n - 1)
  return(sqrt(phi2_corr / min((k_corr - 1), (r_corr - 1))))
}

# Choose only categorical features

data_cat <- data1[cat_feats]

# Create correlation matrix
n <- length(cat_feats)
results <- matrix(NA, nrow = n, ncol = n)
colnames(results) <- rownames(results) <- cat_feats

# Calculate pairwise Cramér's V
for (i in 1:n) {
  for (j in 1:n) {
    results[i, j] <- round(cramers_v(data_cat[[i]], data_cat[[j]]), 2)
  }
}

# Convert matrix to long format for ggplot
results_df <- melt(results)
colnames(results_df) <- c("Var1", "Var2", "value")

# Plot heatmap
ggplot(results_df, aes(Var2, Var1, fill = value)) +
  geom_tile(color = "white") +
  geom_text(aes(label = value), color = "black") +
  scale_fill_gradientn(colors = c("#FCD2FC", "#FEAEFE", "#FC05FB", "#3FFEBA")) +
  theme_minimal() +
  labs(title = "Categorical Feature Correlation (Cramér's V)",
       x = "", y = "", fill = "Cramer's V") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(size = 16, face = "bold"))
```



	chest_pain_type	resting_electrocardiogram	st_slope	thalassemia	sex	fasting_blood_sugar	exercise_induced_angina	target
target	0.5	0.16	0.38	0.53	0.28	0	0.42	1
exercise_induced_angina	0.45	0.05	0.27	0.32	0.13	0	1	0.42
fasting_blood_sugar	0.09	0.02	0.08	0.07	0	1	0	0
sex	0.12	0.08	0	0.38	1	0	0.13	0.28
thalassemia	0.24	0	0.22	1	0.38	0.07	0.32	0.53
st_slope	0.19	0.11	1	0.22	0	0.08	0.27	0.38
resting_electrocardiogram	0.07	1	0.11	0	0.08	0.02	0.05	0.16
chest_pain_type	1	0.07	0.19	0.24	0.12	0.09	0.45	0.5

The plot shows the following observations:

To assess the strength of association between categorical variables in the dataset, we computed Cramer's V correlation coefficients and visualized them using a heatmap.

Cramer's V values range from 0 to 1. The results indicate that chest_pain_type (0.50) and thalassemia (0.53) have the strongest correlation with the target variable. Additionally, exercise_induced_angina (0.42) and st_slope (0.38) show moderate correlations with the target, highlighting their potential importance in classification models. On the other hand, variables such as sex, and resting_electrocardiogram show weaker associations. Notably, fasting_blood_sugar appears to have no meaningful correlation with the target or other variables, which may suggest limited predictive value.

Classification Models Overview

Overview

We are:

- Predicting a binary target (target)
- Using models that have different preprocessing needs
- Applying cross-validation, tuning, and evaluating on a test set

We will:

1- Split the data

2- Create tailored recipes per model (some need normalization, dummies, etc.)

3- Build model specifications

4- Tune (if needed)

5- Fit best models on the full training set

6- Evaluate on the test set

7- Compare results

This is a binary classification problem, where the goal is to predict whether a patient has heart disease (target = 1) or not (target = 0). The tidymodels framework in R offers access to a wide range of machine learning algorithms. In this analysis, we begin by building initial models using several commonly used classification algorithms, including Logistic Regression, k-Nearest Neighbors(kNN), Naive Bayes, Support Vector Machines (SVM), Random Forest, and XGBoost. These models are first implemented with their default parameters to establish baseline performance. In later stages, we will apply hyperparameter tuning and cross-validation to optimize each model and improve predictive accuracy.

Data Splitting

```r
set.seed(123)
data_split <- initial_split(data, prop = 0.7, strata = target)
train_data <- training(data_split)
test_data  <- testing(data_split)
```

Recipes for Models

Before training the models, we will apply preprocessing using tailored recipes, with each model utilizing a different recipe as needed.

```r
 #Logistic Regression /SVM / kNN  Need full preprocessing
  rec_full <- recipe(target ~ ., data = train_data) %>%
    step_log(cholesterol, st_depression, num_major_vessels, offset = 1) %>%
    step_BoxCox(resting_blood_pressure, max_heart_rate_achieved) %>%
    step_normalize(all_numeric_predictors()) %>%
    step_dummy(all_nominal_predictors()) %>%
    step_zv(all_predictors())



#Random Forest Light preprocessing (handles factors and outliers internally)


rec_tree <- recipe(target ~ ., data = train_data) %>%
  step_unknown(all_nominal_predictors()) %>%
  step_zv(all_predictors())

#XGBoost Needs dummy variables, normalization
rec_xgb <- recipe(target ~ ., data = train_data) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors())

#Naive Bayes Needs
rec_nb <- recipe(target ~ ., data = train_data) %>%
  step_zv(all_predictors())
```

Model Definitions

```r
# Logistic Regression
log_spec <- logistic_reg() %>% set_engine("glm")
log_wf <- workflow() %>% add_model(log_spec) %>% add_recipe(rec_full)

# Random Forest
rf_spec <- rand_forest(mtry = tune(), min_n = tune(), trees = 500) %>%
  set_engine("ranger") %>% set_mode("classification")
rf_wf <- workflow() %>% add_model(rf_spec) %>% add_recipe(rec_tree)

# kNN
knn_spec <- nearest_neighbor(neighbors = tune()) %>%
  set_engine("kknn") %>% set_mode("classification")
knn_wf <- workflow() %>% add_model(knn_spec) %>% add_recipe(rec_full)

# SVM
svm_spec <- svm_rbf(cost = tune(), rbf_sigma = tune()) %>%
  set_engine("kernlab") %>% set_mode("classification")
svm_wf <- workflow() %>% add_model(svm_spec) %>% add_recipe(rec_full)



# Naive Bayes
nb_spec <- naive_Bayes() %>% set_engine("naivebayes") %>% set_mode("classification")
nb_wf <- workflow() %>% add_model(nb_spec) %>% add_recipe(rec_nb)

# XGBoost
xgb_spec <- boost_tree(trees = 500, tree_depth = tune(), learn_rate = tune()) %>%
  set_engine("xgboost") %>% set_mode("classification")
xgb_wf <- workflow() %>% add_model(xgb_spec) %>% add_recipe(rec_xgb)
```

Cross-Validation and Tuning

```r
set.seed(234)
folds <- vfold_cv(train_data, v = 5, strata = target)
```

```r
#Use tune_grid() for models with tuning parameters (e.g., SVM, RF, XGBoost), or fit_resamples() for tho

# Logistic Regression & Naive Bayes don't need tuning
```

```r
log_res <- fit_resamples(log_wf, resamples = folds, metrics = metric_set(roc_auc, accuracy))
nb_res  <- fit_resamples(nb_wf,  resamples = folds, metrics = metric_set(roc_auc, accuracy))
```

```
## > A | warning: naive_bayes(): Feature resting_electrocardiogram - zero probabilities are present. Co
```

```
## There were issues with some computations   A: x1There were issues with some computations   A: x1
```

```r
# Extract and finalize the parameter set
rf_param <- extract_parameter_set_dials(rf_spec) %>%
  finalize(train_data)

# Create a random grid
rf_grid <- grid_random(rf_param, size = 10)

rf_res <- tune_grid(rf_wf, resamples = folds, grid = rf_grid, metrics = metric_set(roc_auc, accuracy))

# Tune kNN
knn_res <- tune_grid(knn_wf, resamples = folds, grid = tibble(neighbors = seq(3, 25, 2)), metrics = met

# Tune SVM
svm_res <- tune_grid(svm_wf, resamples = folds, grid = 10, metrics = metric_set(roc_auc, accuracy))

# Tune XGBoost
xgb_res <- tune_grid(xgb_wf, resamples = folds, grid = 10, metrics = metric_set(roc_auc, accuracy))
```

```r
# Show best results based on ROC AUC
show_best(rf_res, metric = "roc_auc")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     2    21 roc_auc binary     0.883     5  0.0210 Preprocessor1_Model09
## 2     2    33 roc_auc binary     0.881     5  0.0194 Preprocessor1_Model07
## 3     5     2 roc_auc binary     0.868     5  0.0243 Preprocessor1_Model06
## 4     8    37 roc_auc binary     0.860     5  0.0204 Preprocessor1_Model02
## 5     8     5 roc_auc binary     0.858     5  0.0255 Preprocessor1_Model10
```

```r
show_best(knn_res, metric = "roc_auc")
```

```
## # A tibble: 5 x 7
##   neighbors .metric .estimator  mean     n std_err .config
##       <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1        25 roc_auc binary     0.873     5  0.0209 Preprocessor1_Model12
## 2        23 roc_auc binary     0.871     5  0.0206 Preprocessor1_Model11
## 3        21 roc_auc binary     0.869     5  0.0217 Preprocessor1_Model10
## 4        19 roc_auc binary     0.866     5  0.0220 Preprocessor1_Model09
## 5        17 roc_auc binary     0.863     5  0.0200 Preprocessor1_Model08
```

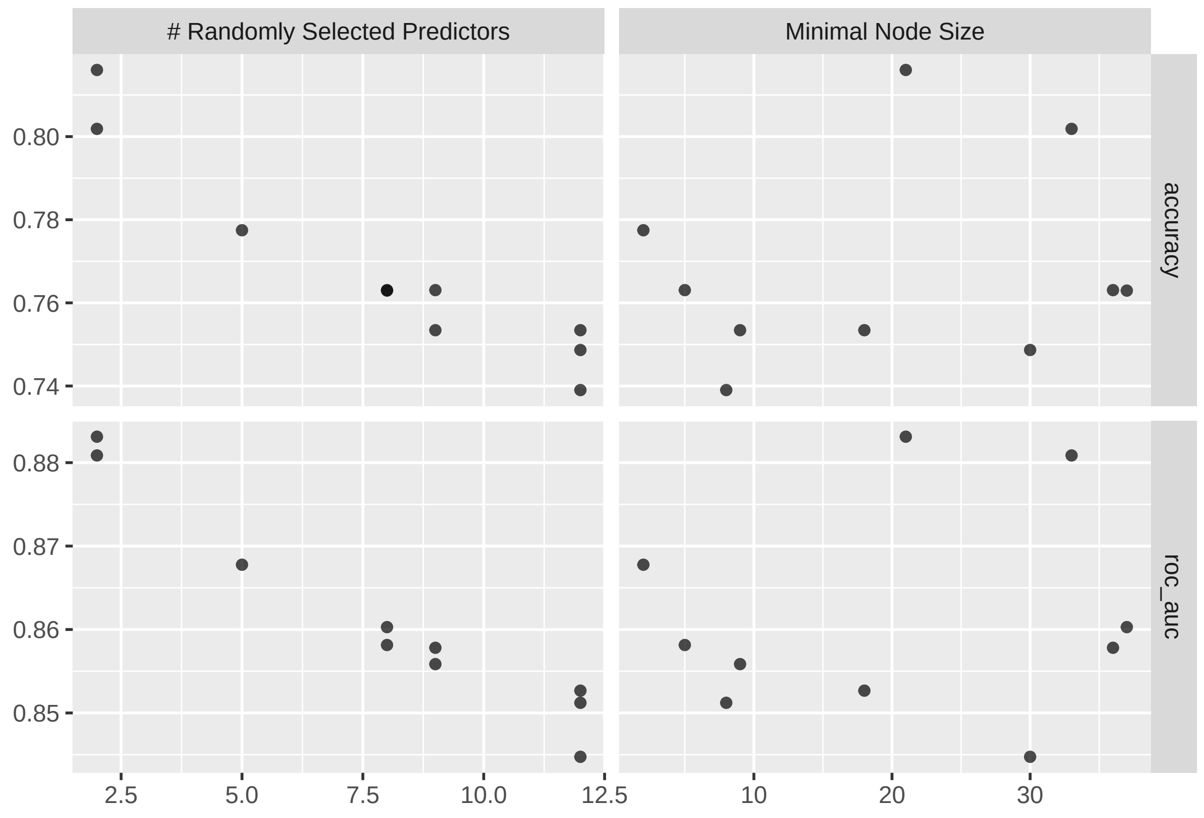
```r
show_best(svm_res, metric = "roc_auc")
```

```
## # A tibble: 5 x 8
##       cost     rbf_sigma .metric .estimator  mean     n std_err .config
##      <dbl>         <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.00310  0.00599       roc_auc binary     0.909     5  0.0217 Preprocessor1_M~
## 2 0.00984  0.0000000001  roc_auc binary     0.909     5  0.0196 Preprocessor1_M~
## 3 1        0.0000359     roc_auc binary     0.909     5  0.0206 Preprocessor1_M~
## 4 0.315    0.00000000129 roc_auc binary     0.907     5  0.0187 Preprocessor1_M~
## 5 0.000977 0.000000215   roc_auc binary     0.906     5  0.0219 Preprocessor1_M~
```

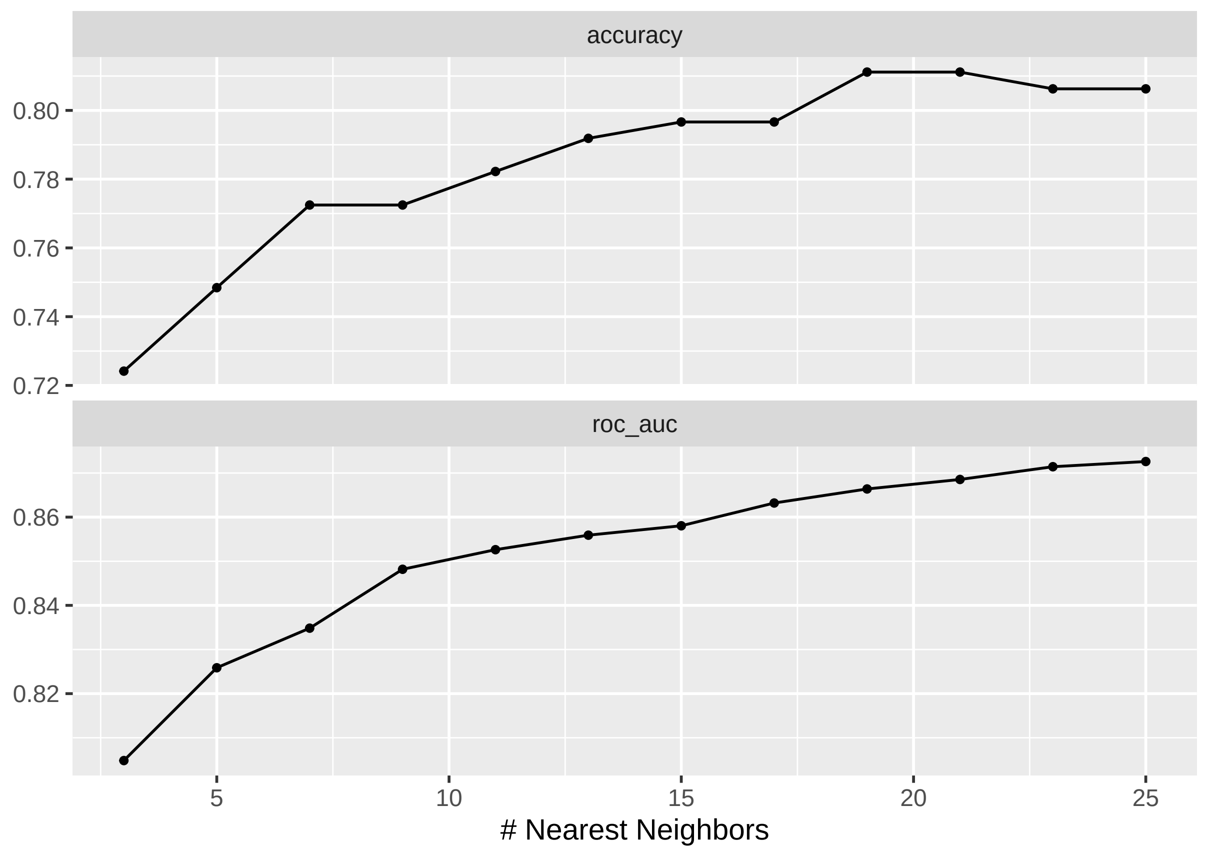
```
show_best(xgb_res, metric = "roc_auc")
```

```
## # A tibble: 5 x 8
##   tree_depth learn_rate .metric .estimator  mean     n std_err .config
##        <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1          1    0.00681 roc_auc binary     0.878     5  0.0275 Preprocessor1_Mo~
## 2          5    0.0129  roc_auc binary     0.855     5  0.0167 Preprocessor1_Mo~
## 3         10    0.0245  roc_auc binary     0.854     5  0.0197 Preprocessor1_Mo~
## 4          2    0.0880  roc_auc binary     0.851     5  0.0219 Preprocessor1_Mo~
## 5         15    0.0464  roc_auc binary     0.850     5  0.0176 Preprocessor1_Mo~
```

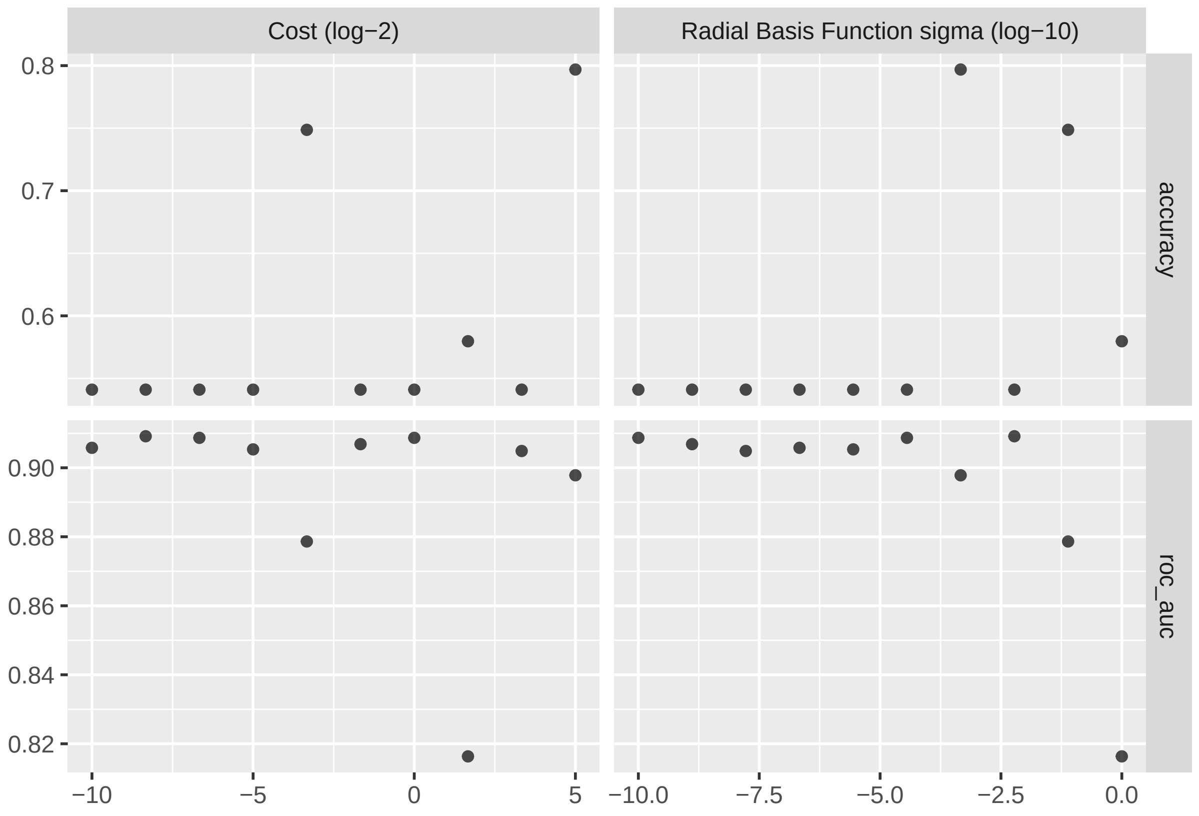
```
#  Plot tuning performance
autoplot(rf_res)
```



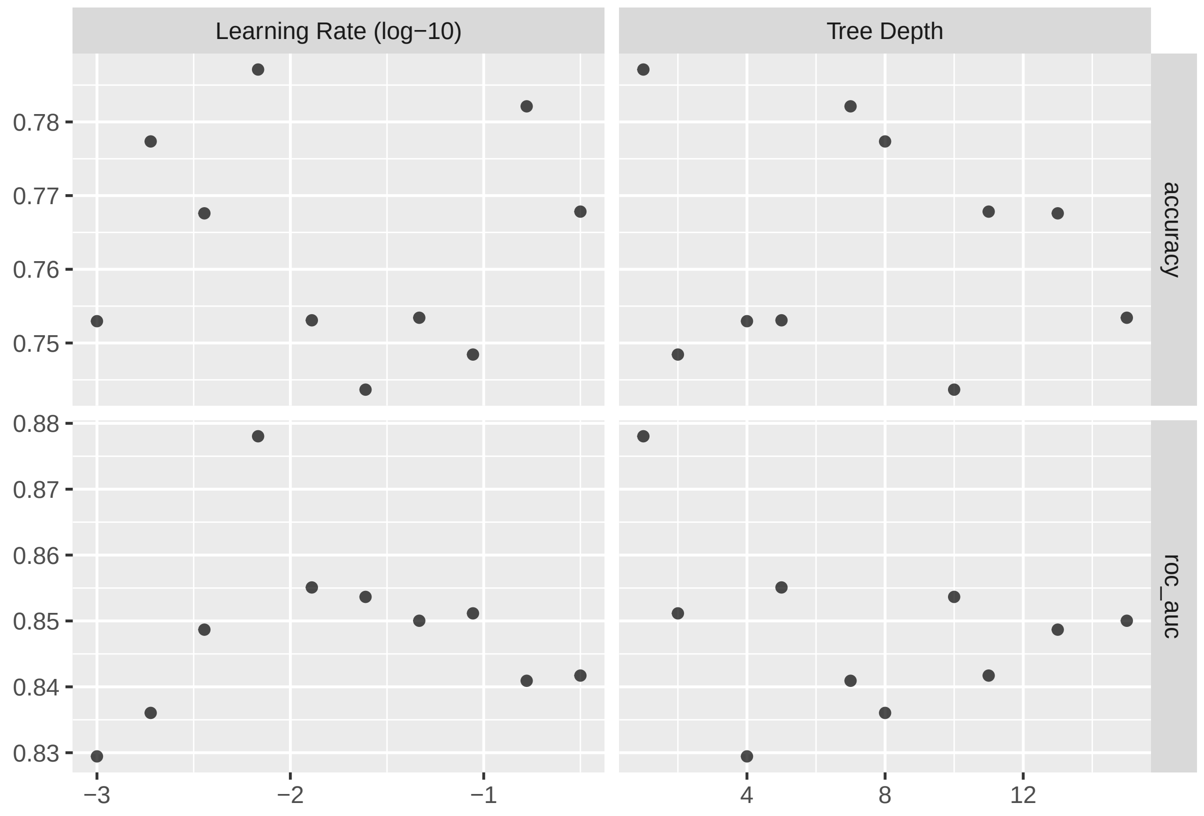
```r
autoplot(knn_res)
```



```r
autoplot(svm_res)
```



```r
autoplot(xgb_res)
```



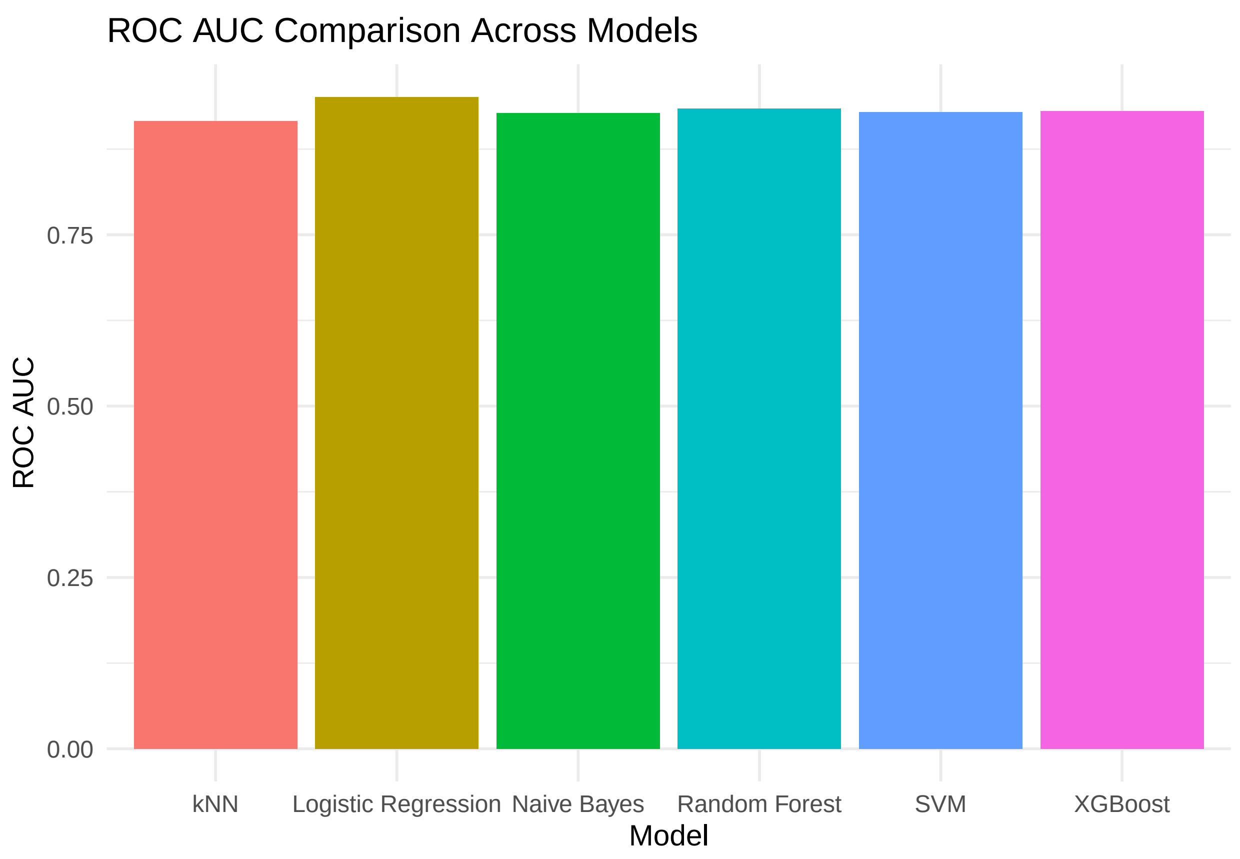
It visually shows how the performance metrics (e.g., roc_auc, accuracy) change as we vary the hyperparameters we're tuning.

Finalize, Fit on Train, Evaluate on Test

```r
# Select best parameters and finalize
final_rf <- finalize_workflow(rf_wf, select_best(rf_res, metric = "roc_auc"))
final_knn <- finalize_workflow(knn_wf, select_best(knn_res, metric = "roc_auc"))
final_svm <- finalize_workflow(svm_wf, select_best(svm_res, metric = "roc_auc"))
final_xgb <- finalize_workflow(xgb_wf, select_best(xgb_res, metric = "roc_auc"))



# Fit on training set and evaluate on test set

log_final <- last_fit(log_wf, split = data_split)
nb_final  <- last_fit(nb_wf,  split = data_split)
rf_final  <- last_fit(final_rf, split = data_split)
knn_final <- last_fit(final_knn, split = data_split)
svm_final <- last_fit(final_svm, split = data_split)
xgb_final <- last_fit(final_xgb, split = data_split)

# Collect test metrics
model_metrics <- bind_rows(
  collect_metrics(log_final) %>% mutate(model = "Logistic Regression"),
  collect_metrics(nb_final)  %>% mutate(model = "Naive Bayes"),
  collect_metrics(rf_final)  %>% mutate(model = "Random Forest"),
  collect_metrics(knn_final) %>% mutate(model = "kNN"),
  collect_metrics(svm_final) %>% mutate(model = "SVM"),
  collect_metrics(xgb_final) %>% mutate(model = "XGBoost")
)

# Tidy summary of results
model_metrics %>%
  select(model, .metric, .estimate) %>%
  pivot_wider(names_from = .metric, values_from = .estimate) %>%
  arrange(desc(roc_auc))
```

```
## # A tibble: 6 x 4
##   model               accuracy roc_auc brier_class
##   <chr>                  <dbl>   <dbl>       <dbl>
## 1 Logistic Regression    0.888   0.951      0.0798
## 2 Random Forest          0.876   0.934      0.113
## 3 XGBoost                0.876   0.930      0.117
## 4 SVM                    0.539   0.929      0.130
## 5 Naive Bayes            0.876   0.928      0.101
## 6 kNN                    0.865   0.916      0.108
```

```r
model_metrics %>%
  filter(.metric == "roc_auc") %>%
  ggplot(aes(x = model, y = .estimate, fill = model)) +
  geom_col() +
  labs(title = "ROC AUC Comparison Across Models", y = "ROC AUC", x = "Model") +
  theme_minimal() +
  theme(legend.position = "none")
```



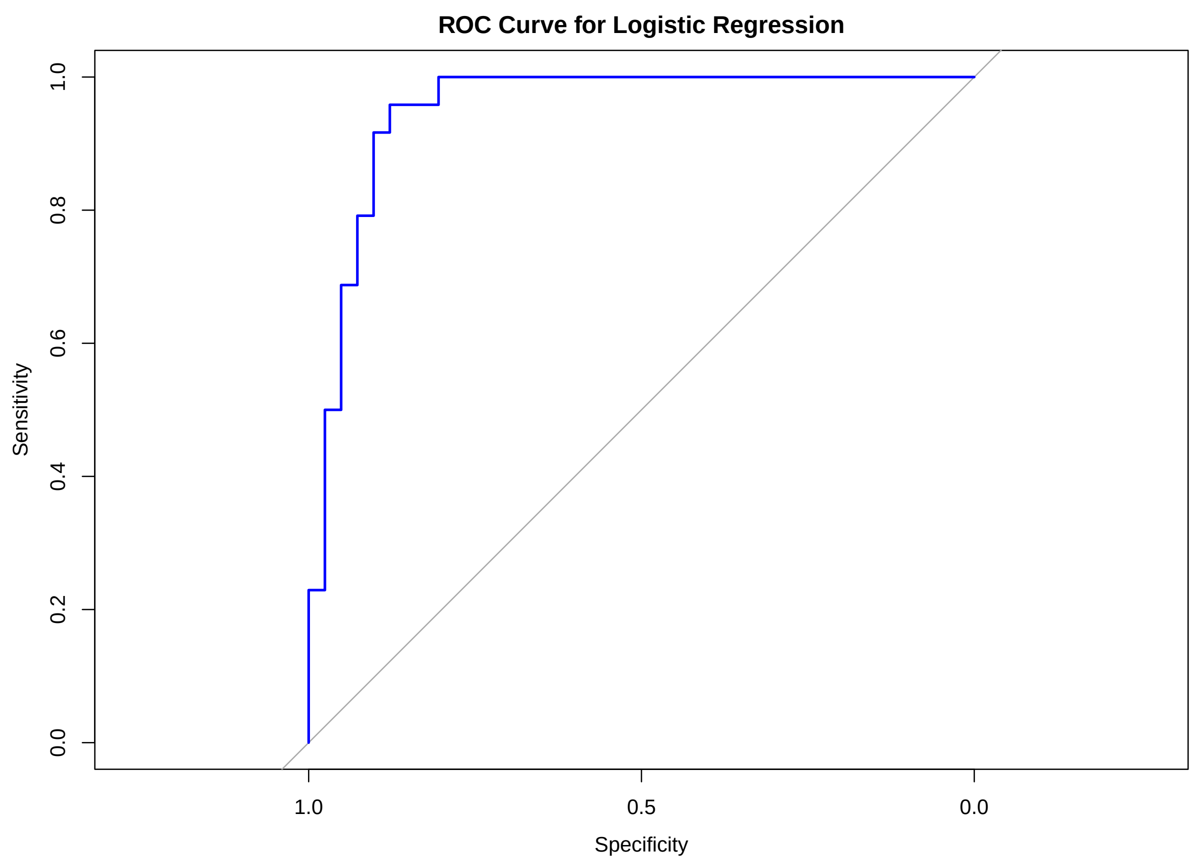
ROC Curve

Now, we can compare the ROC curves for our machine learning models.

```r
# Get predictions
log_preds <- collect_predictions(log_final)

# Find the positive class level (usually the second level of the factor)
positive_class <- levels(log_preds$target)[1]
positive_class
```

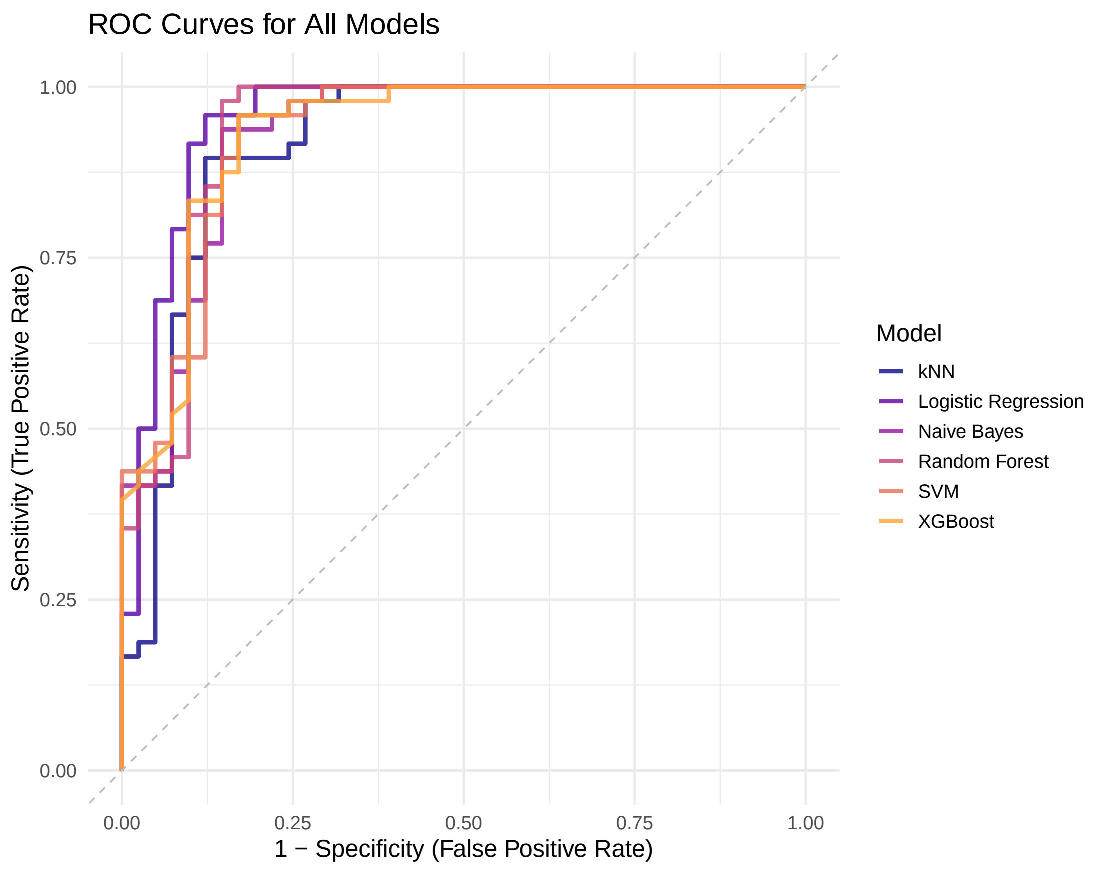
```
## [1] "Heart Disease"
```

```r
# Build ROC curve
log_roc <- roc(
  response = log_preds$target,
  predictor = log_preds[[paste0(".pred_", positive_class)]],
  levels = rev(levels(log_preds$target))
)

# Plot
plot(log_roc, main = "ROC Curve for Logistic Regression", col = "blue", lwd = 2)
```

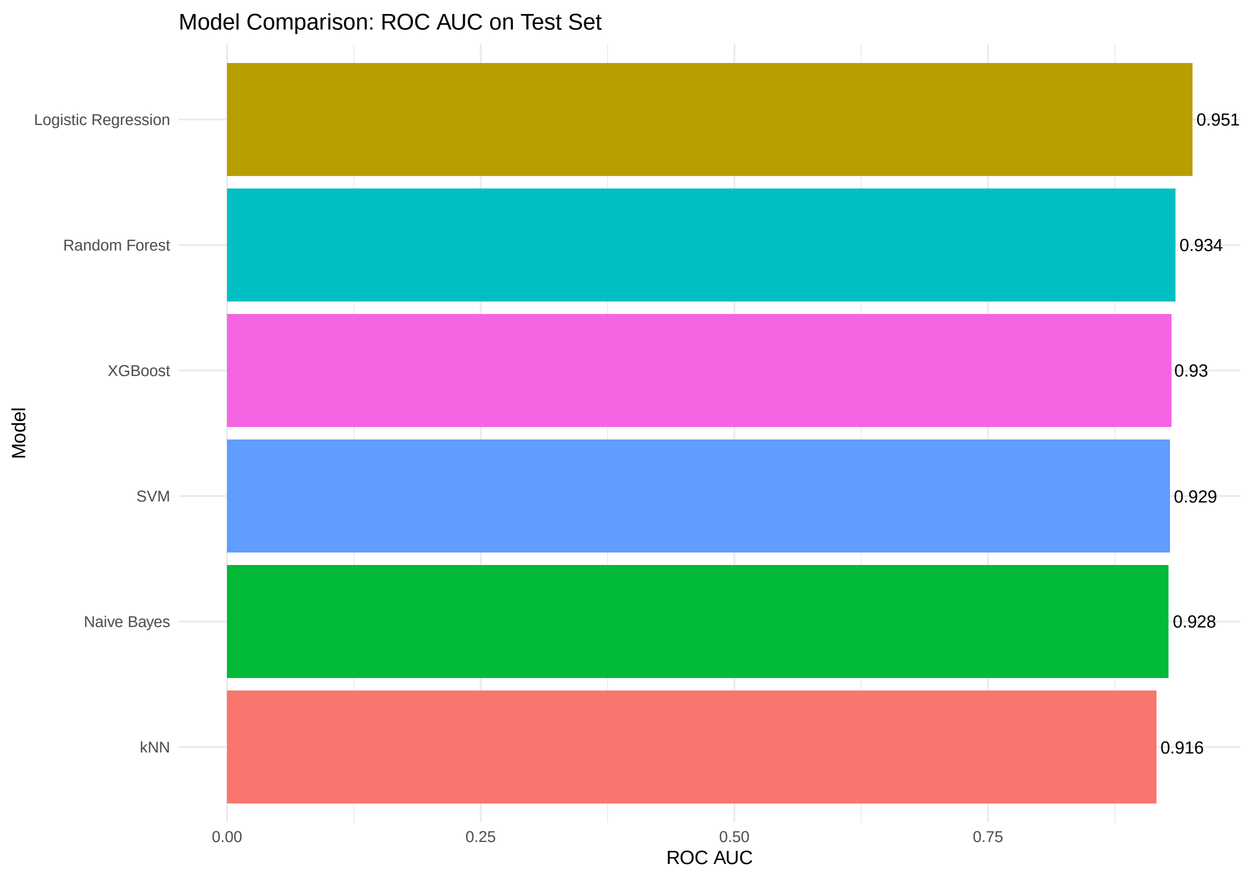


```
#Now, we can compare ROC curves for all models
##__________Step 1: Collect ROC Curve Data for All Models

# Collect predictions from last_fit objects
log_preds <- collect_predictions(log_final)
rf_preds  <- collect_predictions(rf_final)
knn_preds <- collect_predictions(knn_final)
svm_preds <- collect_predictions(svm_final)
xgb_preds <- collect_predictions(xgb_final)
nb_preds  <- collect_predictions(nb_final)

log_roc_df <- roc_curve(log_preds, truth = target, `.pred_Heart Disease`) %>%
  mutate(model = "Logistic Regression")

rf_roc_df  <- roc_curve(rf_preds,  truth = target, `.pred_Heart Disease`) %>%
  mutate(model = "Random Forest")

knn_roc_df <- roc_curve(knn_preds, truth = target, `.pred_Heart Disease`) %>%
  mutate(model = "kNN")

svm_roc_df <- roc_curve(svm_preds, truth = target, `.pred_Heart Disease`) %>%
  mutate(model = "SVM")

xgb_roc_df <- roc_curve(xgb_preds, truth = target, `.pred_Heart Disease`) %>%
  mutate(model = "XGBoost")
```

```r
nb_roc_df  <- roc_curve(nb_preds,  truth = target, `.pred_Heart Disease`) %>%
  mutate(model = "Naive Bayes")




all_roc_df <- bind_rows(log_roc_df, rf_roc_df, knn_roc_df, svm_roc_df, xgb_roc_df, nb_roc_df)

ggplot(all_roc_df, aes(x = 1 - specificity, y = sensitivity, color = model)) +
  geom_path(size = 1.2, alpha = 0.8) +
  geom_abline(linetype = "dashed", color = "gray") +
  coord_equal() +
  scale_color_viridis_d(option = "plasma", end = 0.8) +
  labs(
    title = "ROC Curves for All Models",
    x = "1 - Specificity (False Positive Rate)",
    y = "Sensitivity (True Positive Rate)",
    color = "Model"
  ) +
  theme_minimal(base_size = 14)
```



```r
# Bar plot comparing ROC AUC across models
model_metrics %>%
  filter(.metric == "roc_auc") %>%
  ggplot(aes(x = reorder(model, .estimate), y = .estimate, fill = model)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  labs(title = "Model Comparison: ROC AUC on Test Set",
       x = "Model", y = "ROC AUC") +
  theme_minimal() +
  geom_text(aes(label = round(.estimate, 3)), hjust = -0.1, size = 3.5)
```



Among all the models, logistic regression delivered the best performance. The ROC curve displayed above evaluates the discriminative ability of the logistic regression model. The curve shows a sharp rise towards the top-left corner, reflecting high sensitivity and specificity. This shape indicates that the model is effective at correctly classifying both positive (heart disease) and negative (no heart disease) cases. The area under the ROC curve (AUC) is approximately 0.951, confirming the model's excellent performance. An AUC near 1.0 suggests the model makes highly reliable predictions across all classification thresholds.

Confusion Matrix for the Best Model

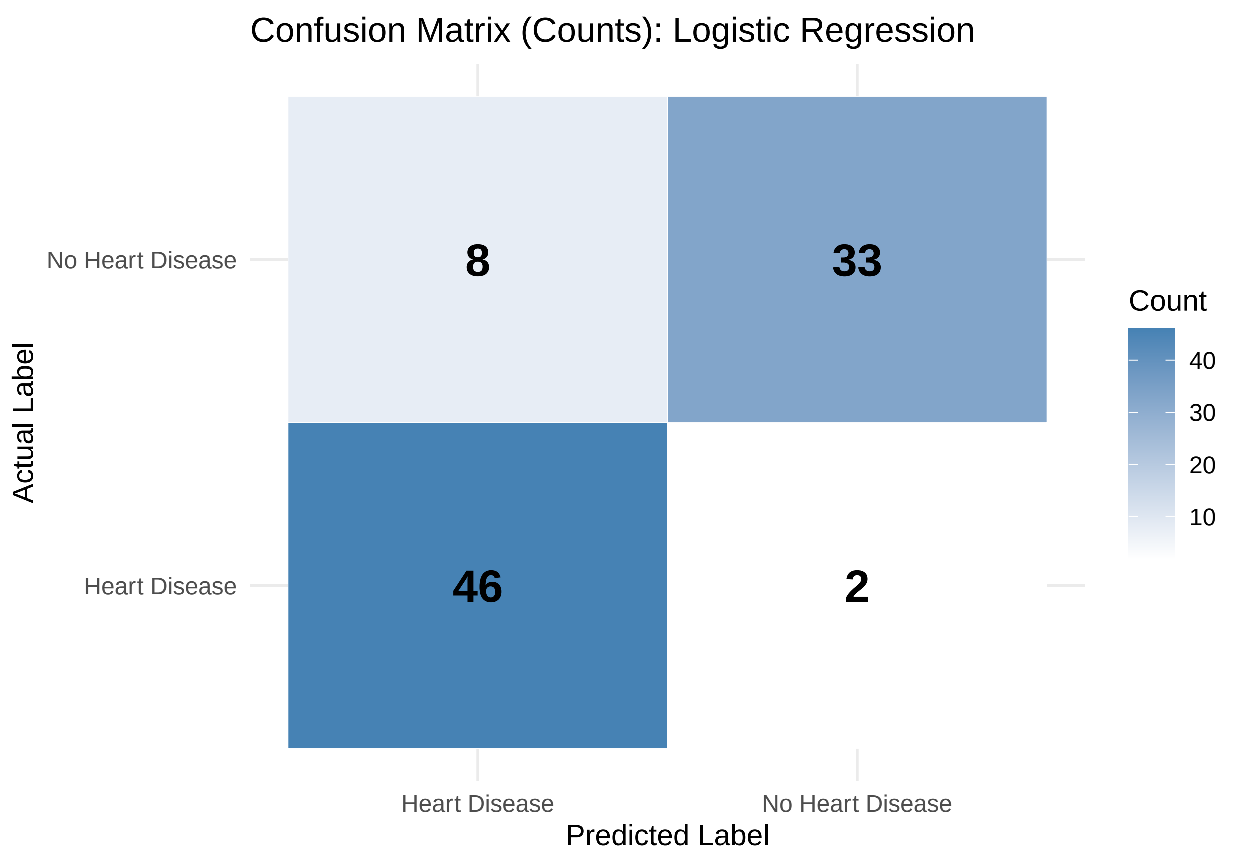
We observe that Logistic Regression performs the best, so we'll proceed to generate its Confusion Matrix and ROC curve.

```
# Confusion Matrix for the best model (Logistic Regression)
log_preds <- collect_predictions(log_final)
log_conf_mat <- conf_mat(log_preds, truth = target, estimate = .pred_class)
```

```r
# Print Confusion Matrix
log_conf_mat
```

```
##                     Truth
## Prediction          Heart Disease No Heart Disease
##   Heart Disease                46                8
##   No Heart Disease              2               33
```

```r
# Confusion Matrix

# Step 1: Collect predictions from last_fit
log_preds <- collect_predictions(log_final)

# Step 2: Compute the confusion matrix
log_conf <- conf_mat(log_preds, truth = target, estimate = .pred_class)

# Step 3: Convert to data frame for ggplot
conf_df <- as.data.frame(log_conf$table)

# Step 4: Plot manually with labels
ggplot(conf_df, aes(x = Prediction, y = Truth, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 6, fontface = "bold") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(
    title = "Confusion Matrix (Counts): Logistic Regression",
    x = "Predicted Label",
    y = "Actual Label",
    fill = "Count"
  ) +
  theme_minimal()
```

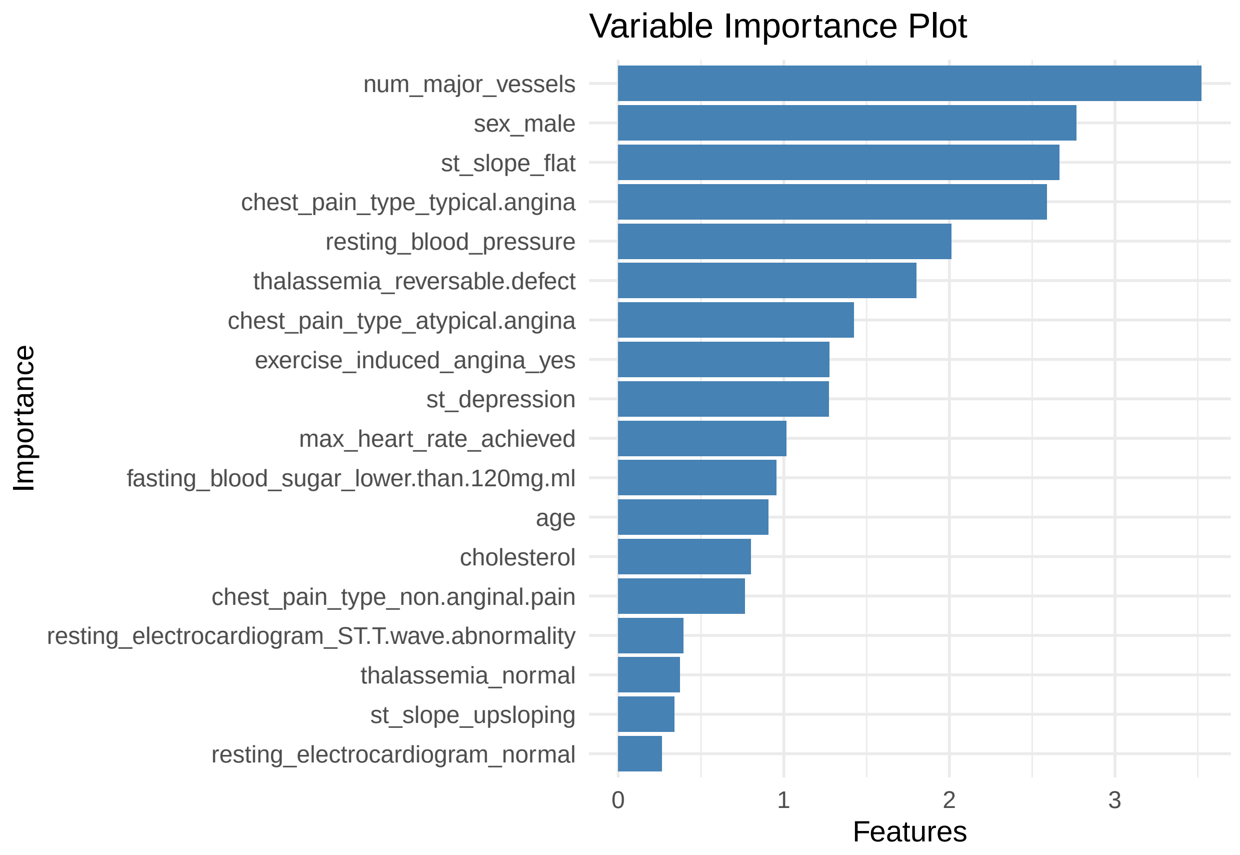


The model performs very well in detecting Heart Disease, achieving a high recall (95.8%) for Heart Disease cases, but there is room to improve precision (85.2%) by reducing false positives. Overall, the model's accuracy (88.7%) and F1-score (90.3%) indicate strong performance, especially in identifying Heart Disease. However, improvements could be made in predicting No Heart Disease more accurately, as shown by the specificity of 80.5%.

Extract model from workflow and generate VIP plot

We will use the vip package to visualize the variable importance scores for the top 20 features: For the sheer sake of adventure, let's make a variable importance plot (VIP) to see which predictor variables have the most impact in our model.

```r
vip_plot <- log_final %>%
  extract_fit_parsnip() %>%
  vip(num_features = 20,
      geom = "col",
      aesthetics = list(fill = "steelblue")) +
  labs(title = "Variable Importance Plot", x = "Importance", y = "Features") +
  theme_minimal()

vip_plot
```



The most important predictors in whether a Heart Disease or not were the number of major vessels, male sex, flat ST slope, typical angina chest pain type, and resting blood pressure.

Conclusion

This project aimed to develop machine learning models to predict heart disease based on patient attributes and clinical measurements. Our key findings include:

1. Demographic factors: Age and sex are associated with heart disease risk, with males showing a higher prevalence in this dataset.
2. Clinical indicators: Several clinical measurements showed strong associations with heart disease, including maximum heart rate achieved, ST depression, number of major vessels, chest pain type, exercise-induced angina, and thalassemia.
3. Model performance: Among all the models evaluated for heart disease prediction, logistic regression achieved the highest accuracy at 0.88
4. Important predictors: The most significant predictors of heart disease identified by the logistic regression model were the number of major vessels, male sex, flat ST slope, typical angina chest pain type, and resting blood pressure.

Limitations

This study has several limitations that should be considered:

1. Dataset size: The dataset contains only 303 patients, which may limit the generalizability of the findings.
2. Data quality: The dataset may not represent the full spectrum of heart disease presentations and risk factors.
3. Feature engineering: We used the original features without extensive feature engineering,which might improve model performance.

Future Work

Future research directions could include:

1. External validation: Validating the models on external datasets to assess their generalizability.
2. Additional models: Implementing other machine learning approaches to potentially improve prediction accuracy.
3. Feature engineering: Exploring feature transformations and interactions to enhance model performance.
4. Interpretability: Developing more interpretable models that can provide actionable insights for clinicians.
5. Integration: Investigating how these models could be integrated into clinical workflows to support decision-making.

References

1. What Causes Heart Disease? Explaining the Model – Kaggle
2. Tidymodels: Case Study
3. Microsoft Learn: Machine Learning with R
4. Julia Silge – GDPR Violations & Modeling
5. TMWR – Preprocessing Table
6. Heart Disease Dataset – Kaggle
7. Heart Disease Dataset – UCI Machine Learning Repository