Heart Failure Prediction

Using Machine Learning Models to Predict Heart Failure

Julie Ku

Introduction

Cardiovascular disease (CVD) refer to a range of diseases that affect the heart and blood vessels, such as heart failure, heart attack, and other heart diseases. CVDs are the leading cause of death globally, representing 32% of all global deaths.

The aim of this project is to build a machine learning model that can be of great help to early detection and management by predicting a possible heart disease. We will be using data from kaggle (https://www.kaggle.com/fedesoriano/heart-failure-prediction.) and implementing various machine learning techniques to yield the most accurate model for this classification problem.

Exploratory Data Analysis

Loading and Exploring the Data

This dataset was created by combining 5 separate heart datasets over 11 common features, making this the largest heart disease dataset available thus far for research purposes with a total of 918 observations.

The variables in this dataset are as follows:

1. `Age` : age of the patient [years]
2. `Sex` : sex of the patient [M: Male, F: Female]
3. `ChestPainType` : chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4. `RestingBP` : resting blood pressure [mm Hg]
5. `Cholesterol` : serum cholesterol [mm/dl]
6. `FastingBS` : fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
7. `RestingECG` : resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
8. `MaxHR` : maximum heart rate achieved [Numeric value between 60 and 202]
9. `ExerciseAngina` : exercise-induced angina [Y: Yes, N: No]
10. `Oldpeak` : oldpeak = ST [Numeric value measured in depression]
11. `ST_Slope` : the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
12. `HeartDisease` : output class [1: heart disease, 0: Normal]

Loading the data:

```
heart_data <- read.csv("heart.csv") # loading the data

summary(heart_data)
```

```
##       Age            Sex            ChestPainType        RestingBP
##  Min.   :28.00   Length:918         Length:918         Min.   :  0.0
##  1st Qu.:47.00   Class :character   Class :character   1st Qu.:120.0
##  Median :54.00   Mode  :character   Mode  :character   Median :130.0
##  Mean   :53.51                                         Mean   :132.4
##  3rd Qu.:60.00                                         3rd Qu.:140.0
##  Max.   :77.00                                         Max.   :200.0
##   Cholesterol      FastingBS       RestingECG           MaxHR
##  Min.   :  0.0   Min.   :0.0000   Length:918         Min.   : 60.0
##  1st Qu.:173.2   1st Qu.:0.0000   Class :character   1st Qu.:120.0
##  Median :223.0   Median :0.0000   Mode  :character   Median :138.0
##  Mean   :198.8   Mean   :0.2331                      Mean   :136.8
##  3rd Qu.:267.0   3rd Qu.:0.0000                      3rd Qu.:156.0
##  Max.   :603.0   Max.   :1.0000                      Max.   :202.0
##  ExerciseAngina        Oldpeak          ST_Slope          HeartDisease
##  Length:918         Min.   :-2.6000   Length:918         Min.   :0.0000
##  Class :character   1st Qu.: 0.0000   Class :character   1st Qu.:0.0000
##  Mode  :character   Median : 0.6000   Mode  :character   Median :1.0000
##                     Mean   : 0.8874                      Mean   :0.5534
##                     3rd Qu.: 1.5000                      3rd Qu.:1.0000
##                     Max.   : 6.2000                      Max.   :1.0000
```

```
head(heart_data)
```

```
##   Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR
## 1  40   M           ATA       140         289         0     Normal   172
## 2  49   F           NAP       160         180         0     Normal   156
## 3  37   M           ATA       130         283         0         ST    98
## 4  48   F           ASY       138         214         0     Normal   108
## 5  54   M           NAP       150         195         0     Normal   122
## 6  39   M           NAP       120         339         0     Normal   170
##   ExerciseAngina Oldpeak ST_Slope HeartDisease
## 1              N     0.0       Up            0
## 2              N     1.0     Flat            1
## 3              N     0.0       Up            0
## 4              Y     1.5     Flat            1
## 5              N     0.0       Up            0
## 6              N     0.0       Up            0
```

```
dim(heart_data) # getting the dimensions of our data
```

```
## [1] 918  12
```

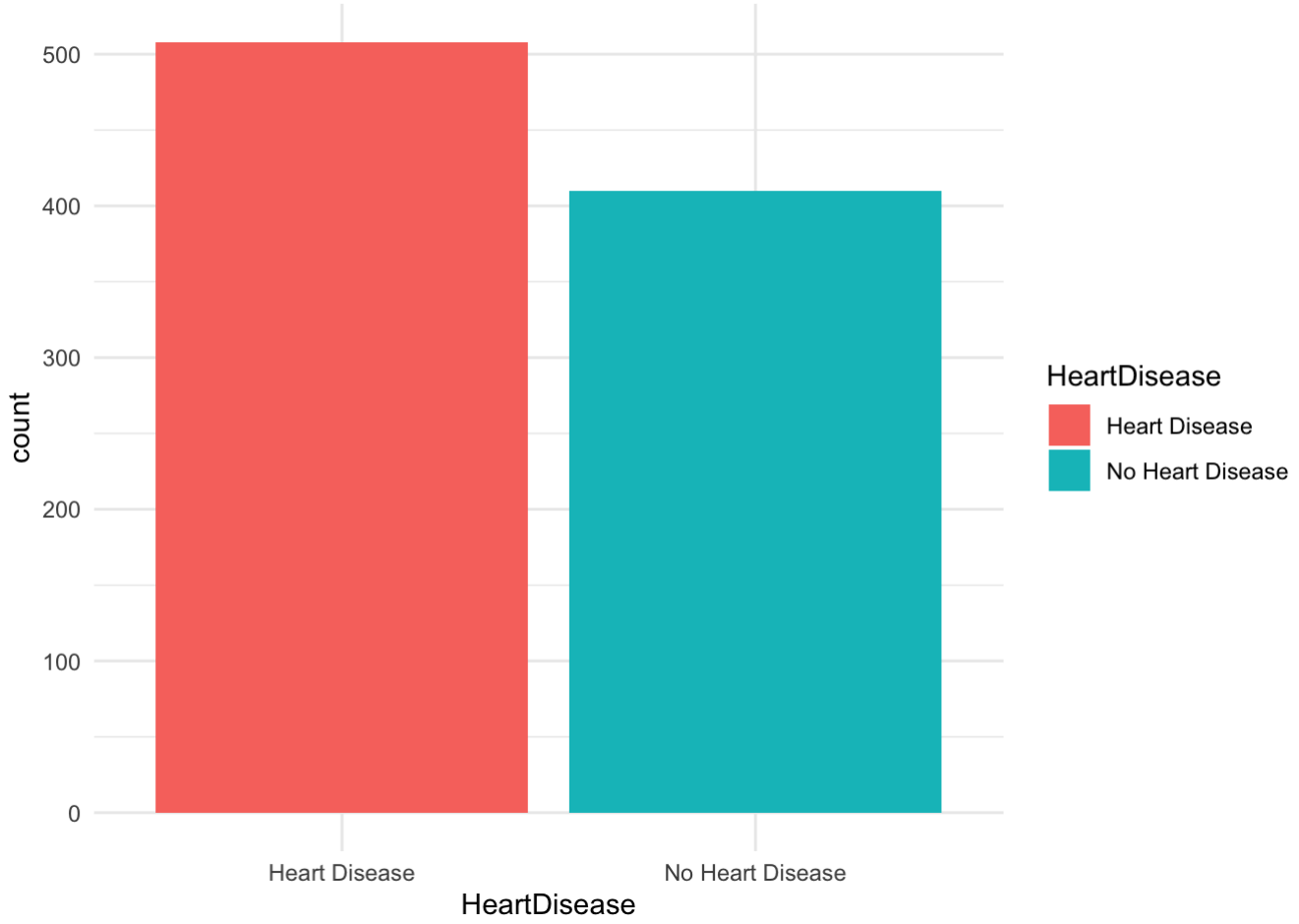
```
sum(is.na(heart_data)) # total number of missing values
```

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

Again, we can see that our dataset consists of 918 total observations with 12 variables. There are 0 missing values in the data.

The response variable will be the HeartDisease and the rest of 11 variables are all predictors.

```
heart_data$HeartDisease<-ifelse(heart_data$HeartDisease==1,"Heart Disease","No Hear
t Disease")
ggplot(data=heart_data,aes(HeartDisease,fill = HeartDisease)) +
  geom_histogram(stat = "count") +
  theme_minimal()
```

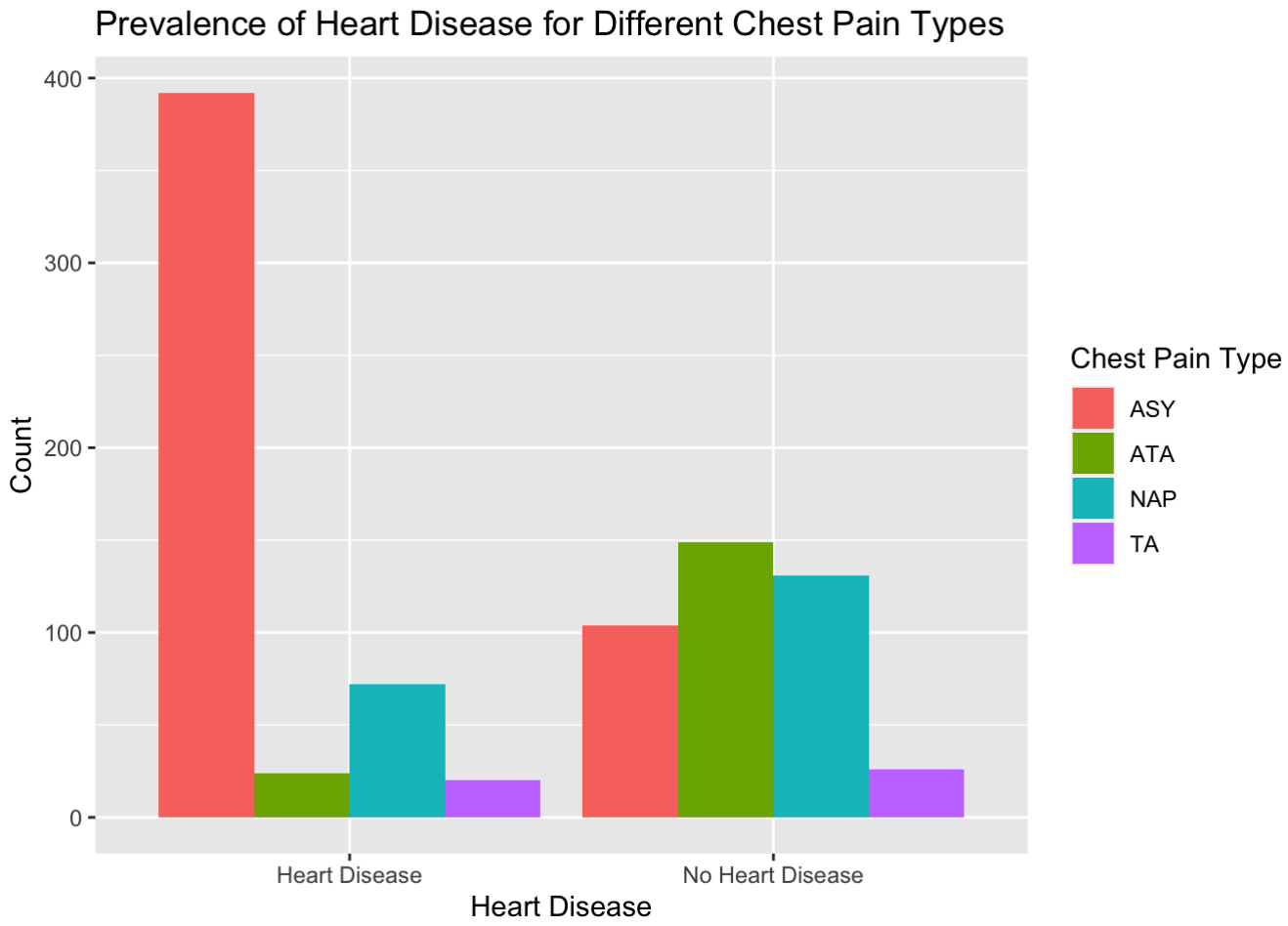



```
ggplot(heart_data, mapping = aes(x = Age, fill = HeartDisease)) +
  geom_histogram() +
  facet_wrap(vars(HeartDisease)) +
  labs(title = "Prevalence of Heart Disease Across Age", x = "Age (years)", y = "Co
unt", fill = "Heart Disease")
```

There are 508 participants with heart disease versus 410 without heart disease.

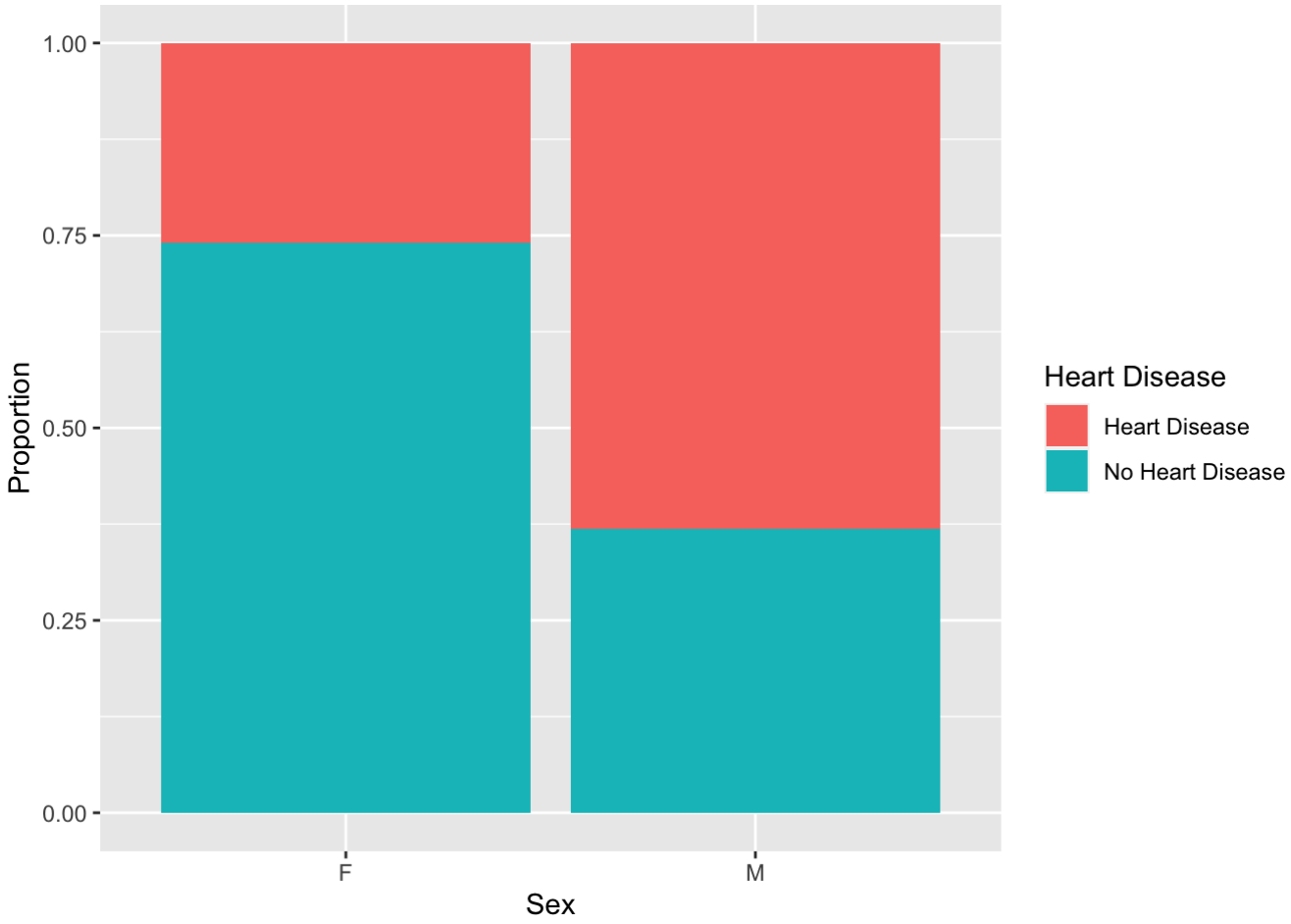
We can also see from the histograms above that `age` has a relationship with our response `heart_disease`, thus there is a prevalence of heart disease with age. There is a left skewed distribution for the presence of heart disease, suggesting that there are more older people with heart disease than younger people with heart disease. The distribution of the absence of heart disease against age appear to be more normally distributed.

```
ggplot(heart_data, mapping = aes(x=HeartDisease, fill = ChestPainType)) +
  geom_bar(position = "dodge") +
  labs(title = "Prevalence of Heart Disease for Different Chest Pain Types", x = "H
eart Disease", y = "Count", fill = "Chest Pain Type")
```

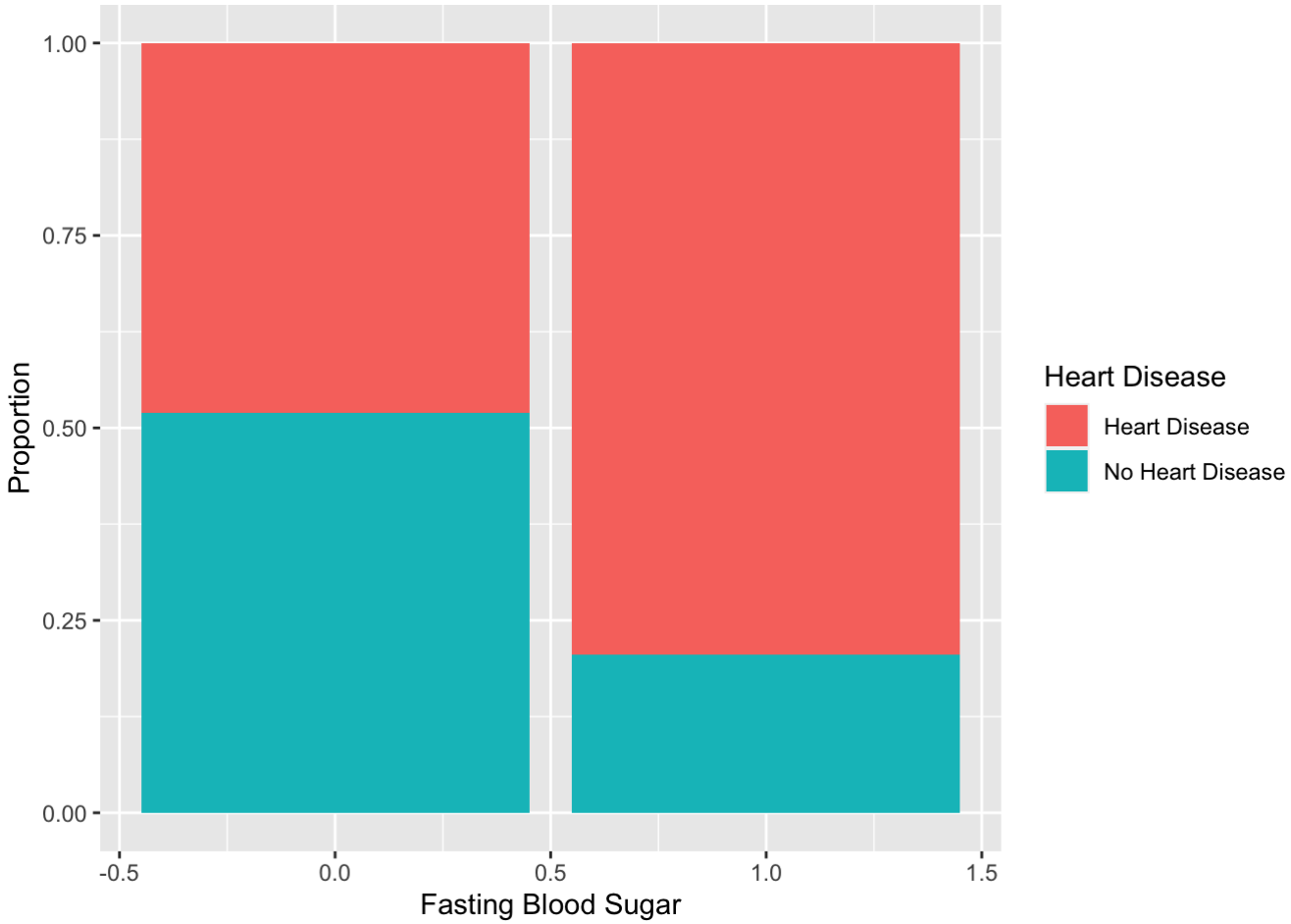


Once again, there does appear to be a relationship between type of `chest_pain_type` and `heart_disease`. It is interesting to note that asymptomatic chest pain type has the highest affiliation with the presence of heart disease, while typical angina pain has the lowest count despite being one of the most common symptoms of heart attack. While our results seem to require further investigation, we will continue on and assume it is correct for the current analysis.

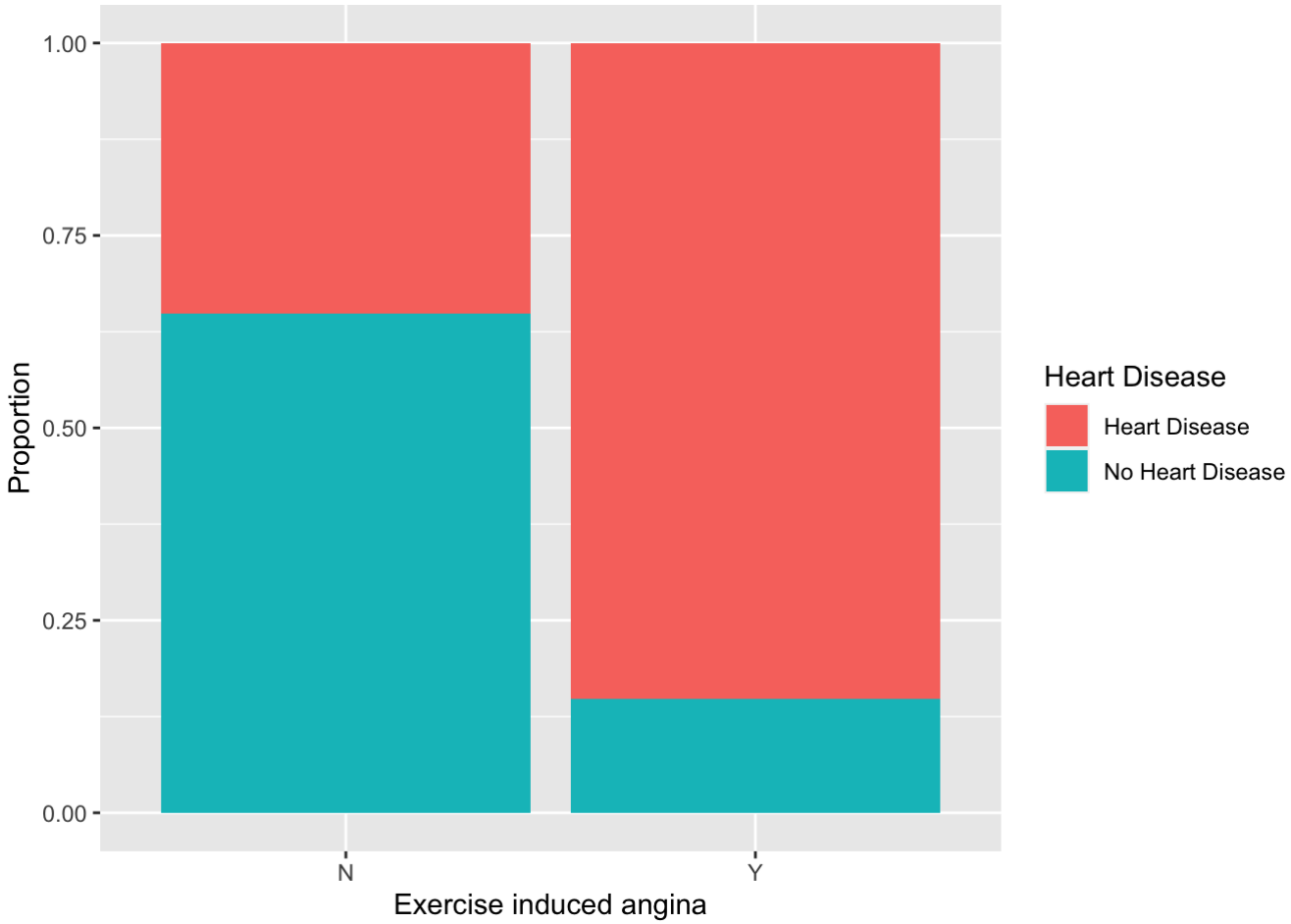
```
ggplot(heart_data, mapping = aes(x = Sex, fill = HeartDisease)) +
  geom_bar(position = "fill") +
  labs(x = "Sex", y = "Proportion", fill = "Heart Disease")
```




```
ggplot(heart_data, mapping = aes(x=FastingBS, fill=HeartDisease)) +
  geom_bar(position = "fill") +
  labs(x = "Fasting Blood Sugar", y = "Proportion", fill = "Heart Disease")
```

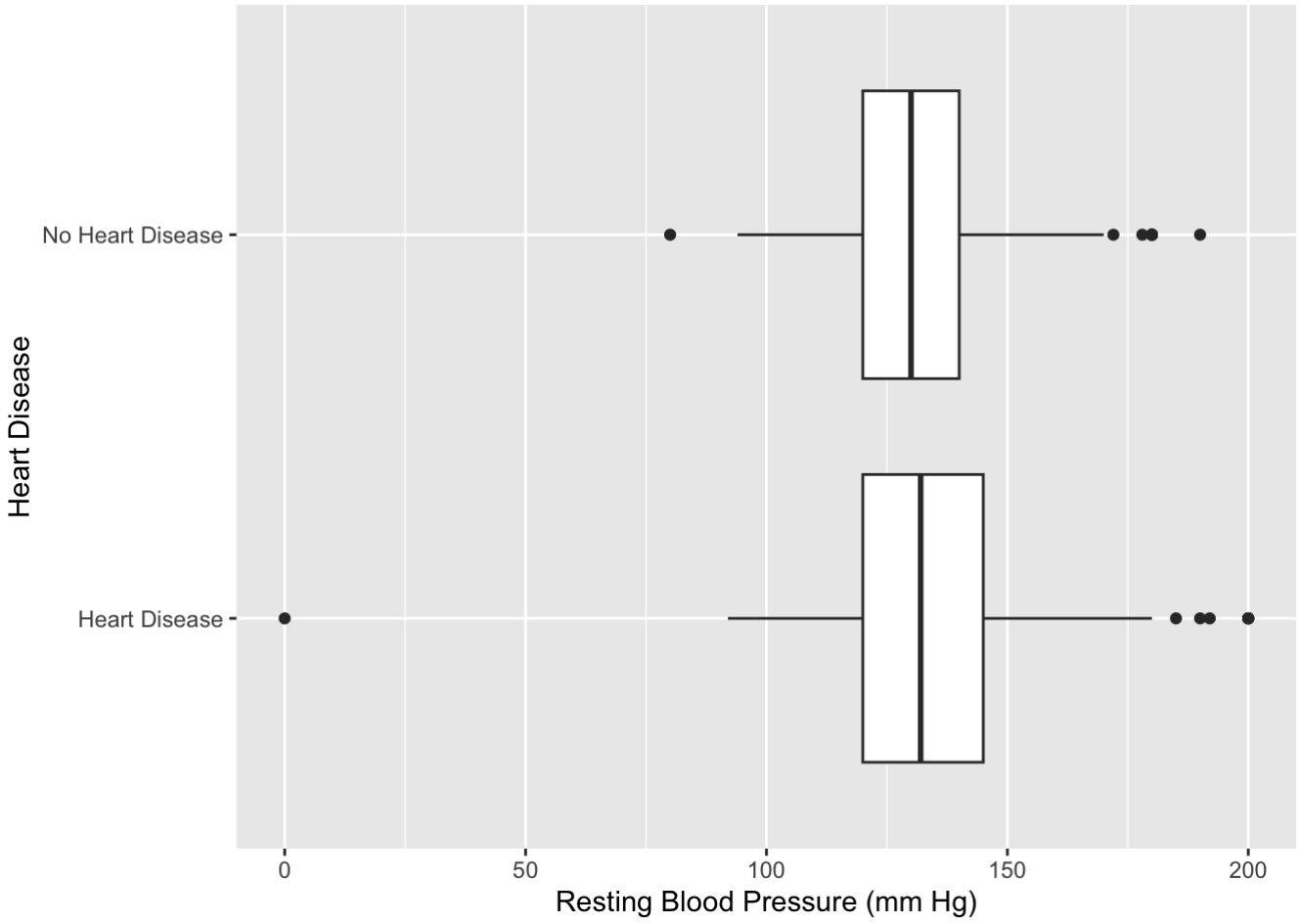



```
ggplot(heart_data, mapping = aes(x = ExerciseAngina, fill = HeartDisease)) +
  geom_bar(position = "fill") +
  labs(x = "Exercise induced angina", y = "Proportion", fill = "Heart Disease")
```

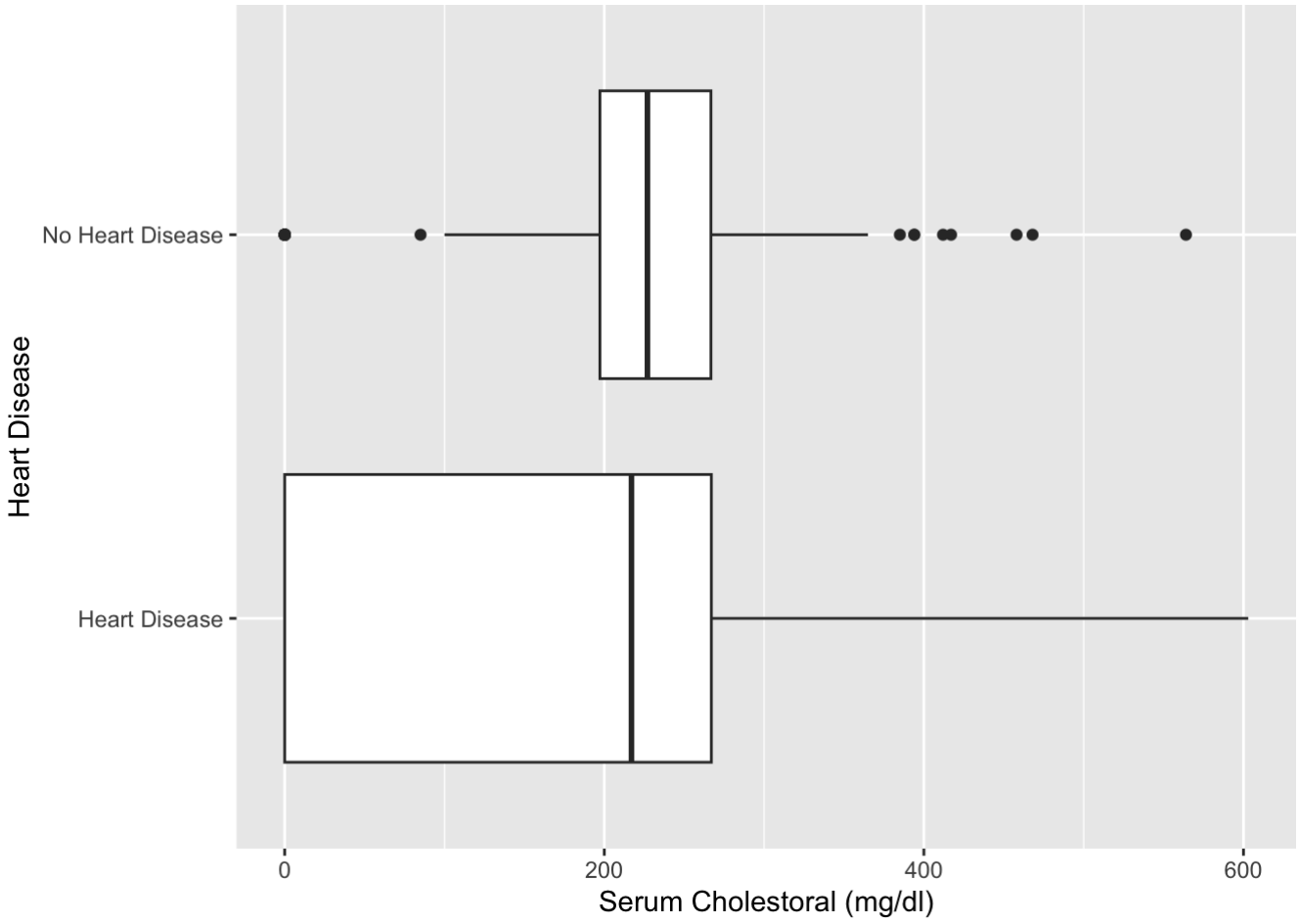


As we can see from the bar plots, there is a higher proportion of males with heart disease than females with heart disease indicating that there is a relationship between the predictor `sex` and the response variable `heart_disease`. Furthermore, it is evident that a high proportion of people with exercise induced angina have heart disease compared to people without exercise induced angina. There also seems to be a relationship between fasting blood sugar (`fasting_bs`) and `heart_disease`.

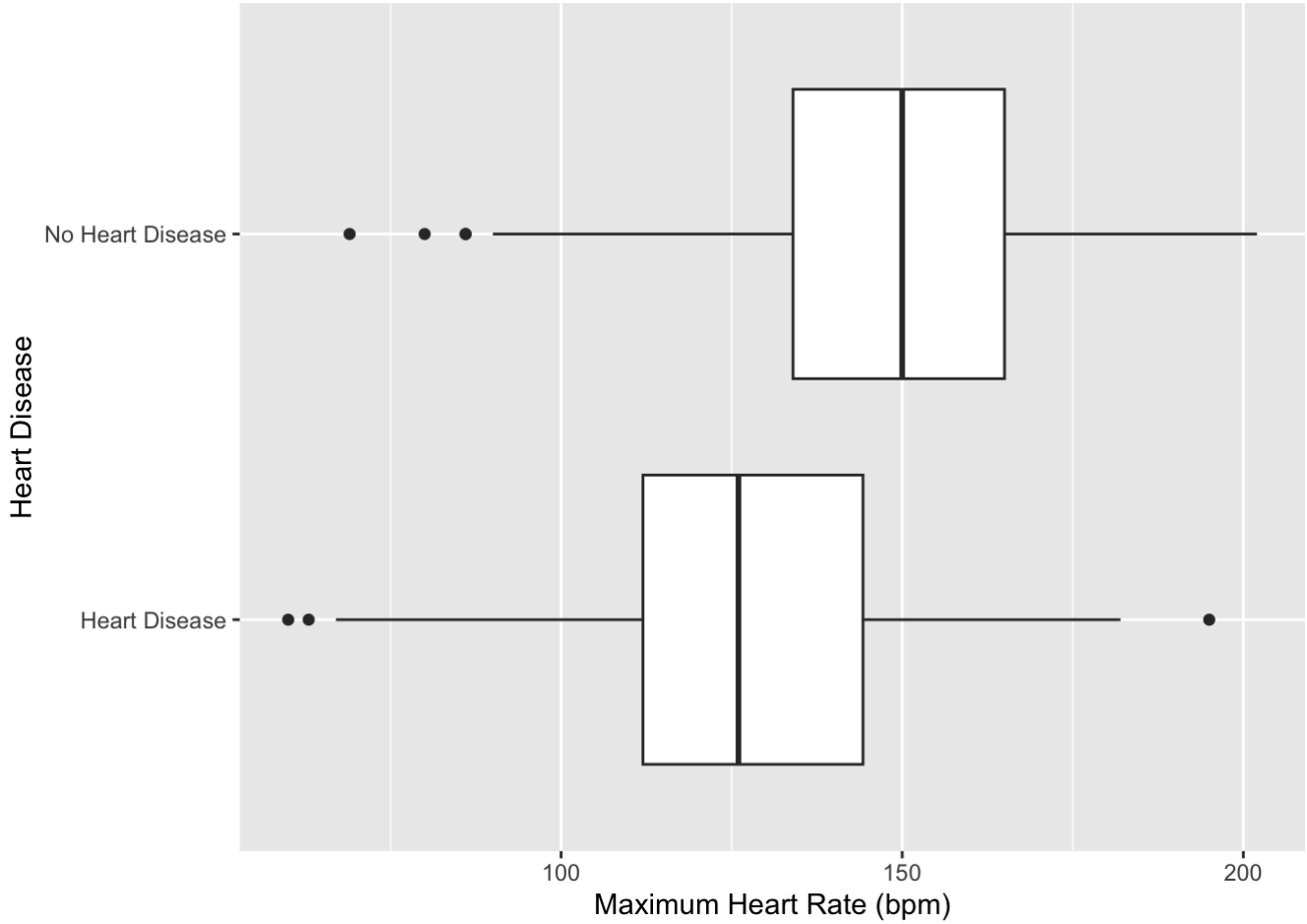
```
ggplot(heart_data, mapping = aes(x=RestingBP, y=HeartDisease)) +
  geom_boxplot() +
  labs(x = "Resting Blood Pressure (mm Hg)", y = "Heart Disease")
```




```
ggplot(heart_data, mapping = aes(x=Cholesterol, y=HeartDisease)) +
  geom_boxplot() +
  labs(x = "Serum Cholestoral (mg/dl)", y = "Heart Disease")
```




```
ggplot(heart_data, mapping = aes(x = MaxHR, y = HeartDisease)) +
  geom_boxplot() +
  labs(x = "Maximum Heart Rate (bpm)", y = "Heart Disease")
```



The box plots for resting blood pressure appear similar for the presence and absence of heart disease as the middle 50% of data overlap with each other and have similar centers. Such similarities between the box plots suggest that there is a weak relationship between the `resting_bs` predictor and our response variable heart disease. We can see from the other box plots that there is a strong relationship between the other predictors: `cholesterol` and `max_hr` and `heart_disease`.

Tidying the Data

Before we get started with building our model, we will clean our dataset using the clean_names() function so that our variable names are easy to read and consistent in one line of code. We will further mutate categorical variables into factors, especially our response variable: `heart_disease`.

```
heart_data$Sex <- as.factor(heart_data$Sex)
heart_data$ChestPainType <- as.factor(heart_data$ChestPainType)
heart_data$FastingBS <- as.factor(heart_data$FastingBS)
heart_data$RestingECG <- as.factor(heart_data$RestingECG)
heart_data$ExerciseAngina <- as.factor(heart_data$ExerciseAngina)
heart_data$ST_Slope <- as.factor(heart_data$ST_Slope)
```

Converting our response variable into a factor:

```
heart_data$HeartDisease <- as.factor(heart_data$HeartDisease)


heart_data <- clean_names(heart_data) # cleaning predictor names
```

Data Split

As we approach model building, we need to first split our data into two separate datasets. One will be our training dataset and used to train the model while the other will be a testing set, that will be used at the end to determine the accuracy of our models by actually testing our models. We will also set a random seed (1234) so that our random split can be reproduced every time we work on our models and train them. Next, we will perform a training / testing split on our data and then stratify on our response variable: `heart_disease`.

```
set.seed(1234)

heart_split <- heart_data %>%
  initial_split(prop = 0.7, strata = 'heart_disease')

heart_train <- training(heart_split) # training split
heart_test <- testing(heart_split) # testing split
```

Dimensions of our training and testing dataset:

```
dim(heart_train) # dimensions of training set
```

```
## [1] 642  12
```

```
dim(heart_test) # dimensions of testing set
```

```
## [1] 276  12
```

There are 642 observations in the training set and 276 observations in the testing set. For splitting the data, I chose a proportion of 0.70 because it allows for more observations in the training data, while retaining a sufficient amount of data for the testing set.

Recipe Building

We will now create a recipe using our predictors and response variable to represent all the models we will be fitting. We will make some categorical variables into dummy variables using the step_dummy() function.

```
heart_recipe <- recipe(heart_disease ~ ., data = heart_train) %>%
step_dummy(all_nominal_predictors()) # setting up a recipe
```

K-Fold Cross Validation

We will stratify our cross validation on the response variable, `heart_disease`, by using 5 folds.

```
heart_folds <- vfold_cv(heart_train, v = 5, strata = 'heart_disease')
```

Model Building

Most models tend to follow a similar process using the same recipe we have created above, except the Logistic Regression, Linear Discriminant Analysis, and Quadratic Discriminant Analysis models that are simpler without the need to tune our hyperparameters. We first set up the model by specifying what type of model it is, then we set its engine and its mode. In our case, our mode was always set to 'classification' due to the nature of our project goal. After we set up the model workflow, we will add the new model as well as our established recipe.

The general workflow of the model building process constituted of the following steps: Keep in mind we will skip steps 3-5 for our Quadratic Discriminant Analysis model.

1. Set up the model by specifying the type of model, setting its engine, and setting its mode to classification.
2. Set up the workflow, add the new model, and add the established `heart_recipe`.
3. Set up the tuning grid with the parameters that we want tuned, and how many different levels of tuning.
4. Tune the model with parameters of your choice.
5. Select the most accurate model from all of the tuning and finalize the workflow.
6. Fit that model with our workflow to the training data set.

Quadratic Discriminant Analysis Model

We will start off by creating a Quadratic Discriminant Analysis model using the recipe we created previously.

```
qda_mod <- discrim_quad() %>%
  set_mode("classification") %>%
  set_engine("MASS")

qda_workflow <- workflow() %>%
  add_model(qda_mod) %>%
  add_recipe(heart_recipe)

qda_fit <- fit(qda_workflow, heart_train)
```

We can view confidence matrices and calculate accuracies on the **training data**:

```
augment(qda_fit, new_data = heart_train) %>%
  conf_mat(truth = heart_disease, estimate = .pred_class)
```

```
##                   Truth
## Prediction         Heart Disease No Heart Disease
##   Heart Disease              316               35
##   No Heart Disease            39              252
```

```
qda_acc <- augment(qda_fit, new_data = heart_train) %>%
  accuracy(truth = heart_disease, estimate = .pred_class)
```

Checking the accuracy of the model on the testing set:

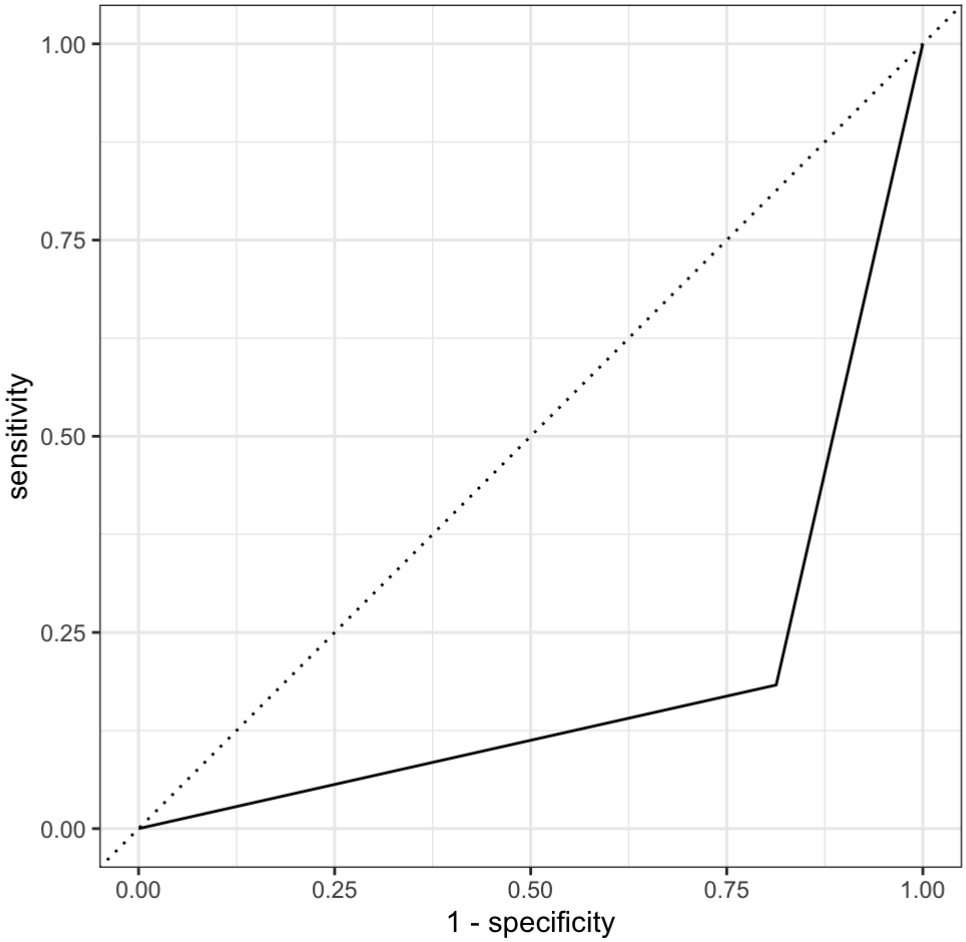
```
augment(qda_fit, new_data = heart_test) %>%
  conf_mat(truth = heart_disease, estimate = .pred_class)
```

```
##                   Truth
## Prediction         Heart Disease No Heart Disease
##   Heart Disease              125               23
##   No Heart Disease            28              100
```

```
multi_metric <- metric_set(accuracy, sensitivity, specificity)

augment(qda_fit, new_data = heart_test) %>%
  mutate(.pred_class = as.factor(.pred_class)) %>%
  multi_metric(truth = heart_disease, estimate = .pred_class)
```

```
## # A tibble: 3 × 3
##   .metric     .estimator .estimate
##   <chr>       <chr>          <dbl>
## 1 accuracy    binary         0.815
## 2 sensitivity binary         0.817
## 3 specificity binary         0.813
```

```
augment(qda_fit, new_data = heart_test) %>%
  mutate(.pred_class = as.numeric(.pred_class)) %>%
  roc_curve(heart_disease, .pred_class) %>%
  autoplot()
```



Random Forest Model

Moving onto our second model: Random Forest Model:

```
#set up the model
rf_mod <- rand_forest(
              min_n = tune(),
              mtry = tune(),
              trees = tune(),
              mode = "classification") %>%
  set_engine("ranger", importance = 'impurity')

#set up the workflow for the model
rf_workflow <- workflow() %>%
  add_recipe(heart_recipe) %>%
  add_model(rf_mod)

#build the grid for tuning
rf_grid <- grid_regular(mtry(range = c(1, 5)),
                        trees(range = c(20, 100)),
                        min_n(range = c(2, 20)),
                        levels = 5)

#Tune the model
tune_res_rf <- tune_grid(
  rf_workflow,
  resamples = heart_folds,
  grid = rf_grid,
  metrics = metric_set(roc_auc)
)

#Show the result
autoplot(tune_res_rf)
```

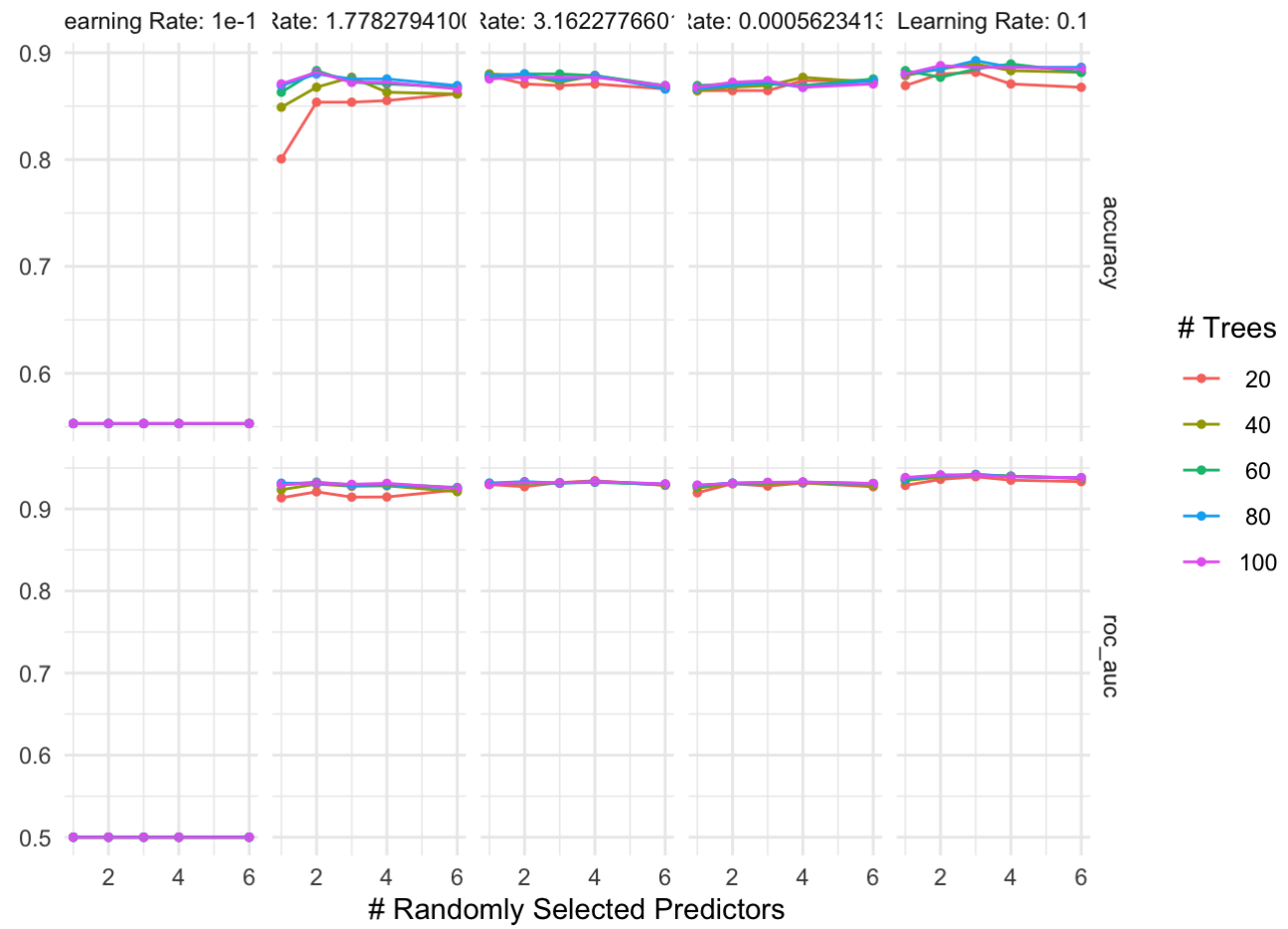
Boosted Tree Model

Our Boosted Tree Model:

```
bt_mod <- boost_tree(mtry = tune(),
                        trees = tune(),
                        learn_rate = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

bt_workflow <- workflow() %>%
  add_model(bt_mod) %>%
  add_recipe(heart_recipe)

bt_grid <- grid_regular(mtry(range = c(1, 6)),
                        trees(range = c(20, 100)),
                        learn_rate(range = c(-10, -1)),
                        levels = 5)
bt_grid
```

```
## # A tibble: 125 × 3
##     mtry trees   learn_rate
##    <int> <int>        <dbl>
##  1     1    20 0.0000000001
##  2     2    20 0.0000000001
##  3     3    20 0.0000000001
##  4     4    20 0.0000000001
##  5     6    20 0.0000000001
##  6     1    40 0.0000000001
##  7     2    40 0.0000000001
##  8     3    40 0.0000000001
##  9     4    40 0.0000000001
## 10     6    40 0.0000000001
## # … with 115 more rows
```

```
tune_bt <- tune_grid(
  bt_workflow,
  resamples = heart_folds,
  grid = bt_grid
)

autoplot(tune_bt) + theme_minimal()
```



```
show_best(tune_bt, n = 1)
```

```
## # A tibble: 1 × 9
##    mtry trees learn_rate .metric .estimator  mean     n std_err .config
##   <int> <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     3    40        0.1 roc_auc binary     0.942     5 0.00806 Preprocessor1_M…
```

```
best_bt <- select_best(tune_bt)
```

Naive Bayes

Lastly, our Naive Bayes Model:

```
nb_mod <- naive_Bayes() %>%
  set_mode("classification") %>%
  set_engine("klaR") %>%
  set_args(usekernel = FALSE)

nb_workflow <- workflow() %>%
  add_model(nb_mod) %>%
  add_recipe(heart_recipe)

nb_fit <- fit(nb_workflow, heart_train)

augment(nb_fit, new_data = heart_train) %>%
  conf_mat(truth = heart_disease, estimate = .pred_class)
```

```
##                  Truth
## Prediction         Heart Disease No Heart Disease
##   Heart Disease              317               46
##   No Heart Disease            38              241
```

```
nb_acc <- augment(nb_fit, new_data = heart_train) %>%
  accuracy(truth = heart_disease, estimate = .pred_class)

nb_acc
```

```
## # A tibble: 1 × 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.869
```

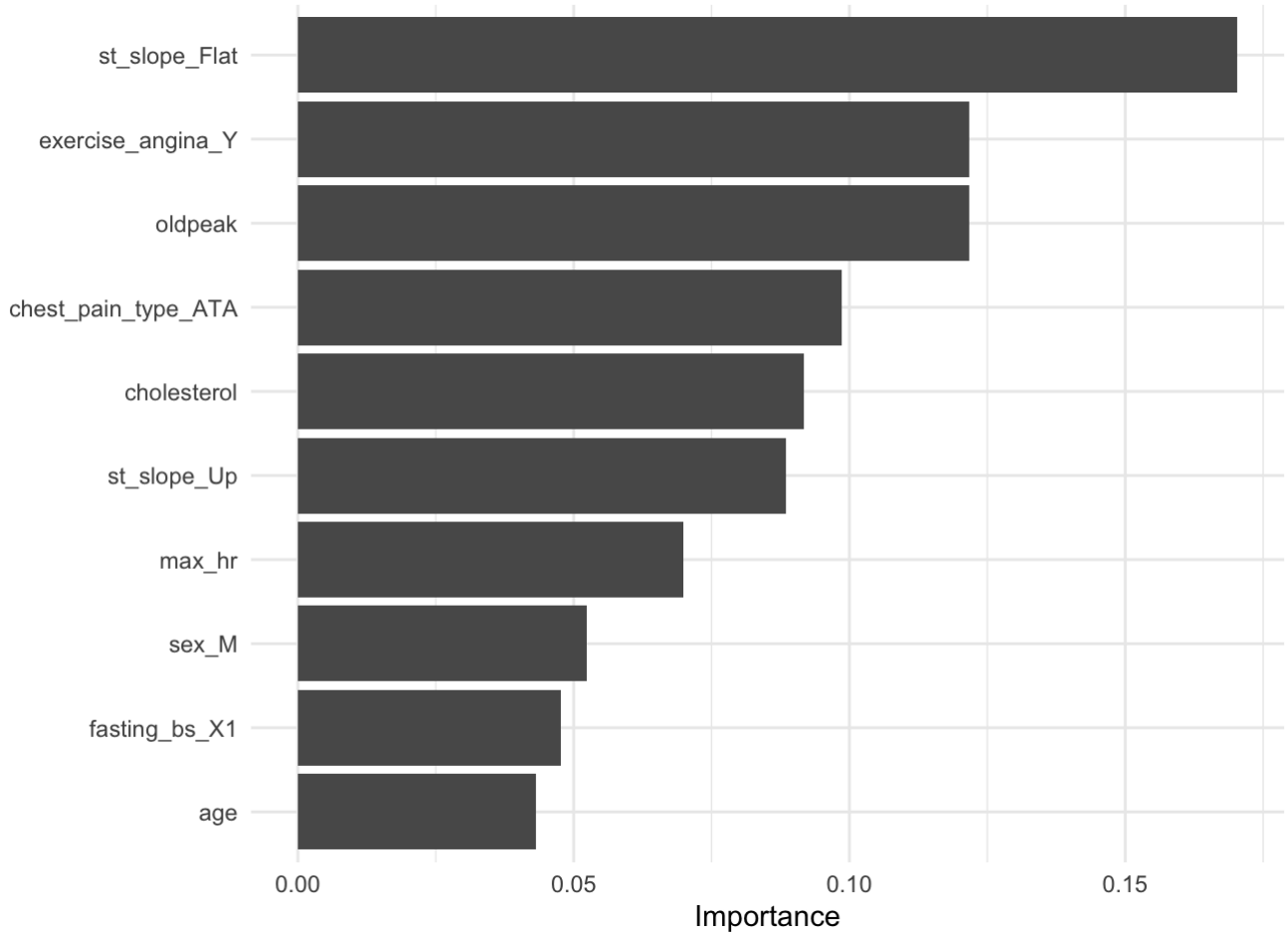
Model Selection

Last but not least, with an accuracy of 94.21%, we can see that our Boosted Tree model is indeed our best performing model. However, the margin of difference is not that influential between the other 3 models, indicating that these are all useful models that could be used to predict heart disease.

```
final_bt_mod <- finalize_workflow(bt_workflow, best_bt)
final_bt_mod <- fit(final_bt_mod, heart_train)
```

Here is the VIP plot:

```
final_bt_mod %>% extract_fit_parsnip() %>%
  vip() +
  theme_minimal()
```



This tells us that the top three most important predictors of `heart_disease` were `st_sope`, `exercise_angina`, and `depth and flipper length` old_peak`.

Conclusion

By now, we have concluded that the Boosted Tree model is the best performing model for our response variable `heart_disease` with the help of all 11 of our predictors. Looking back at the overall process– research, exploratory data analysis, data cleaning, training, testing and analysis, there definitely could have been room for improvement as our model is not perfect.

Overall, my biggest takeaways from this project is learning how to handle binary classification problems with the combination of numerical and categorical features. Working on this project also renewed my appreciation for technological advancements that drastically contribute to society and hopefully help determine how susceptible a person is to developing heart disease. While some of these predictors are not controllable, we were able to show that they have substantial influence on the prediction of heart disease and should be further investigated.

When we put into perspective, our model performed much better than expected. While there was uncertainty surrounding whether I should exclude any predictors or not, I ended up including all 11 predictors as they portrayed some type of relationship with our response variable heart disease. In all, this was a great opportunity to build my experience and skills with machine learning techniques, as well as explore a subject that may be of help in predicting heart disease, which prevails as a deadly condition affecting people globally.