Heart Disease: What are the predictors?

2023-09-17

Contents

INTRODUCTION

Heart disease is a major cause of death all around the world. Since the heart is the most vital organ of the body, its malfunctions affect the overall body to a large degree. Heart disease can take shape in a multitude of facets. My data set includes heart disease data and covers lots of attributes that can be considered as a factor or indicator. I obtained the data from Kaggle at the following link: https://www.kaggle.com/datasets/data855/heart-disease The creators of the data include the Hungarian Instititue of Cardiology, University Hospital of Zurich, Switzerland, and V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. I will be working with numerical variables but also some categorical variables that can be represented in dummy coding. We have over 300 observations and about 15 or so predictors. Let's get familiar with the data:

```
library(tidyverse)
library(tidymodels)
library(ggplot2)
library(corrplot)
library(ggthemes)
library(ISLR)
library(ISLR2)
library(dplyr)
library(discrim)
library(poissonreg)
library(corrr)
library(klaR)
library(pROC)
library(yardstick)
library(glmnet)
library(modeldata)
library(janitor)
```

```r
library(naniar)
library(xgboost)
library(ranger)
library(vip)
tidymodels_prefer()
#setwd("/Users/sayanandrews/Downloads/")
heart <- read.csv(file = "heart.csv")
```

```r
head(heart)
```

```
##   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
##   target
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
```

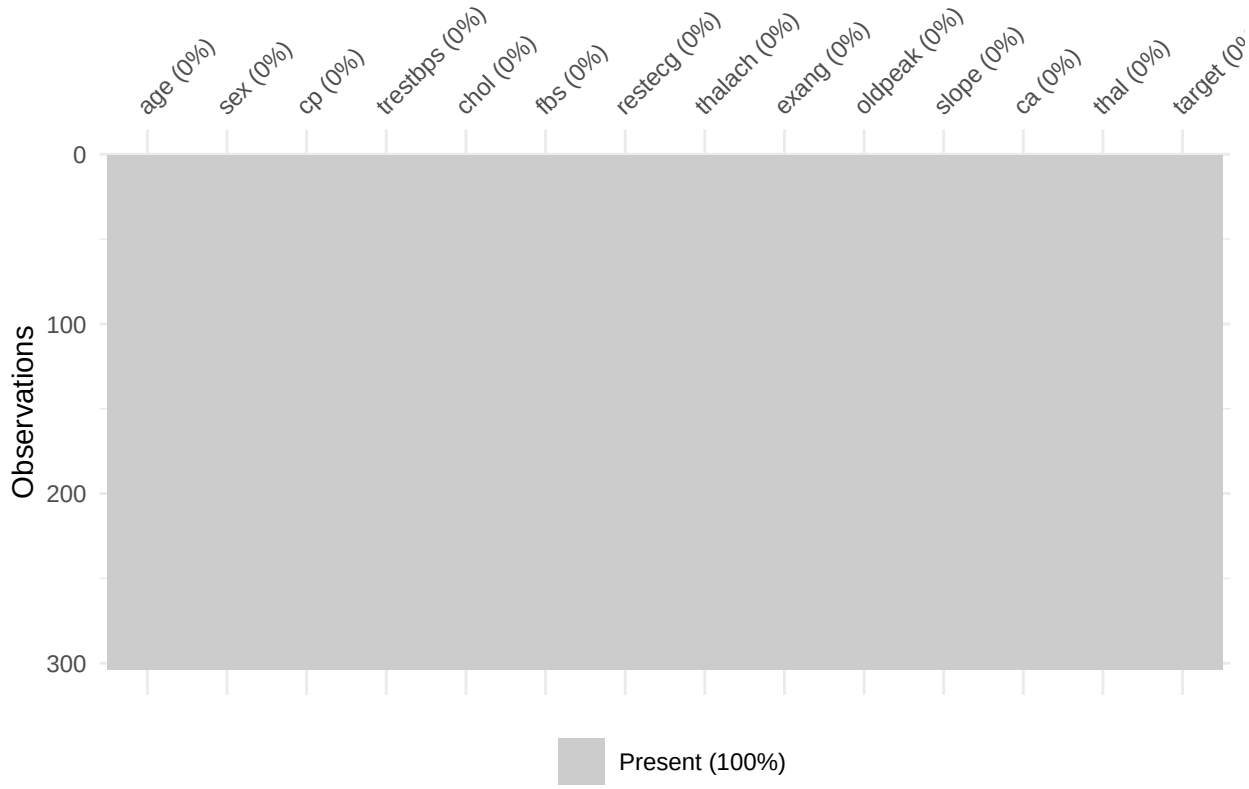
```r
tail(heart, 3)
```

```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 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
## 301      0
## 302      0
## 303      0
```

```r
dim(heart)
```

```
## [1] 303  14
```

```r
library(naniar)
vis_miss(heart)
```

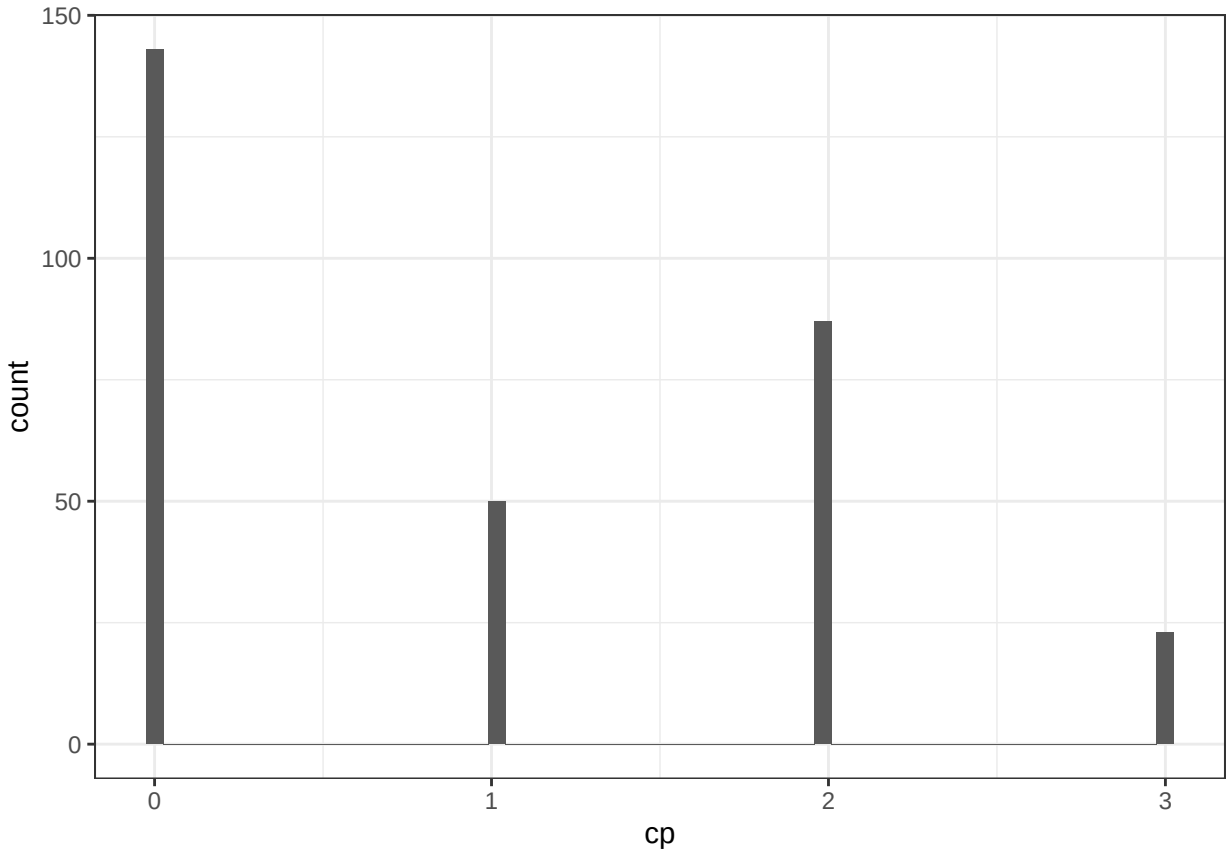


As we can see, there is no missing data. In this project I will be predicting whether or not an observation with this amalgamation of predictors has heart disease or does not have heart disease, and comparing it to the real result. I intend on answering whether or not a person has heart disease based on these indicators, and how these indicators relate to one another as well and impact the featured output. My response output will be the target: no disease or disease. These questions will be best answered with a classification approach, which I will use throughout this project. I believe that the following predictors will prove to be the most useful: chest pain type (cp), number of major vessels (ca), thalassemia (thal), exercise induced angina (exang), max heart rate achieved (thalach), and ST depression (oldpeak). I believe the goal of my model to be inferential and perhaps predictive as well, seeing as I hope to tap into significant features, aim to test theories, possible causal claims, and find the relationship between outcome and predictors. I also hope to find the combination of features that fit best, aiming to predict Y with minimum reducible error. Now, let us begin our EDA (Exploratory Data Analysis):

EXPLORATORY DATA ANALYSIS

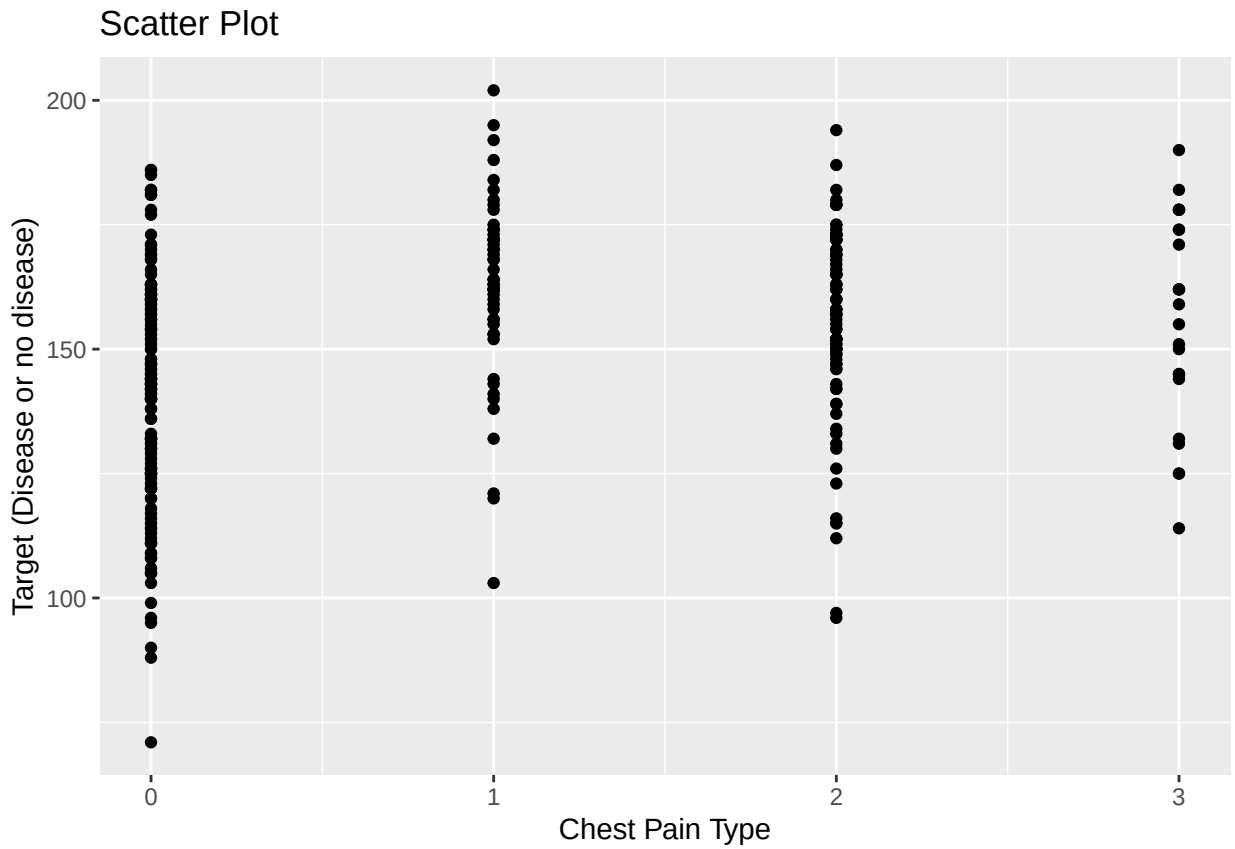
As one might infer, chest pain is a common indicator that someone with heart issues would experience. Let's take a look at the commonality of the different types of chest pain:

```
heart %>%
  ggplot(aes(x = cp)) +
  geom_histogram(bins = 60) +
  theme_bw()
```



As we can see, type 0, or typical angina, is the most common by far. However, as someone with a heart patient mother, I know very well that chest pain can come with an elevated heart rate. Having no prior knowledge of the details of the differences between these types of chest pains, I can find the relationships that they have with other indicators. Let's start with max heart rate achieved:

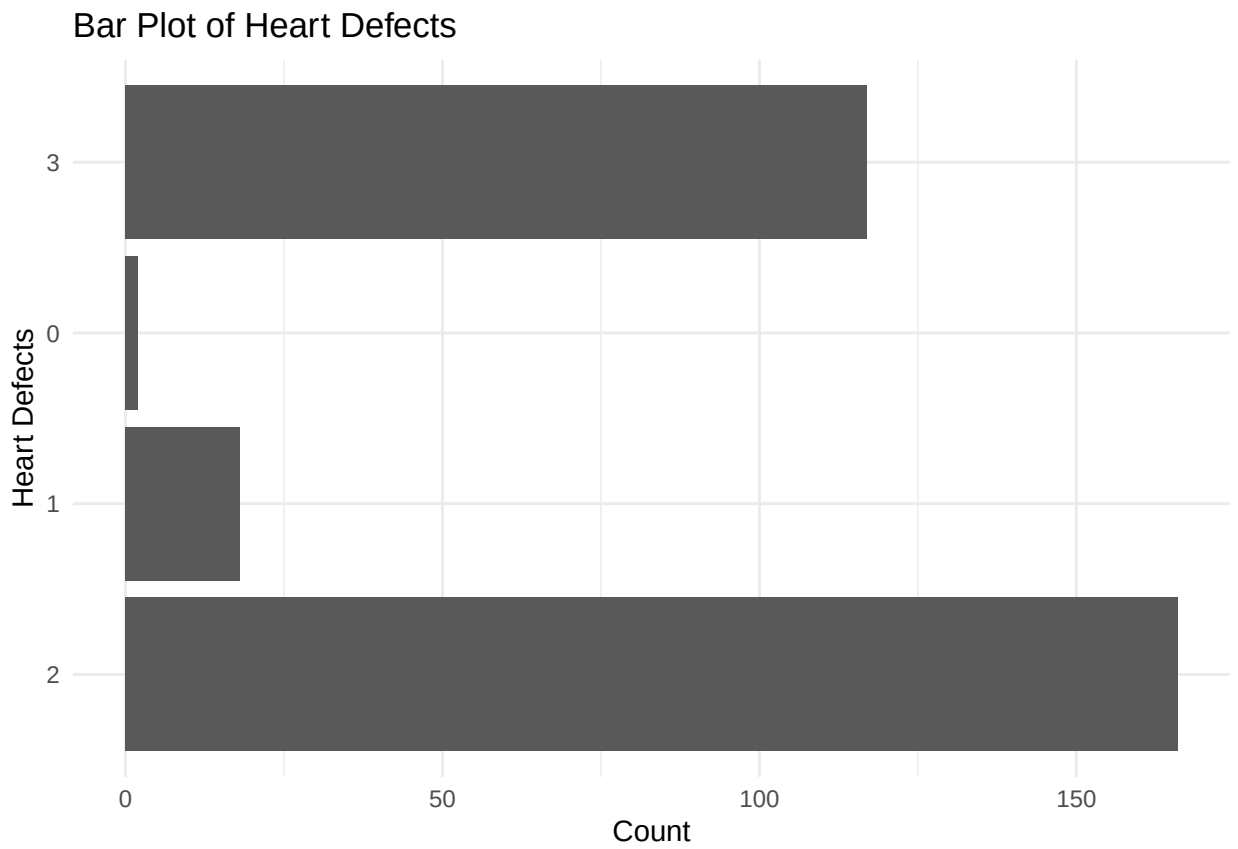
```
heart %>%
  ggplot(aes(x = cp, y = thalach)) +
    geom_point() +
    labs(x = "Chest Pain Type", y = "Target (Disease or no disease)") +
    ggtitle("Scatter Plot")
```



While type 0 is the most common as mentioned before, we see that types 1, 2, and 3, or atypical angina, non-anginal pain, and asymptomatic tend to yield slightly higher distributions of max heart rates.

Another indicator that is interesting to explore is heart defects: predetermined outcomes that could affect the likelihood of heart disease.

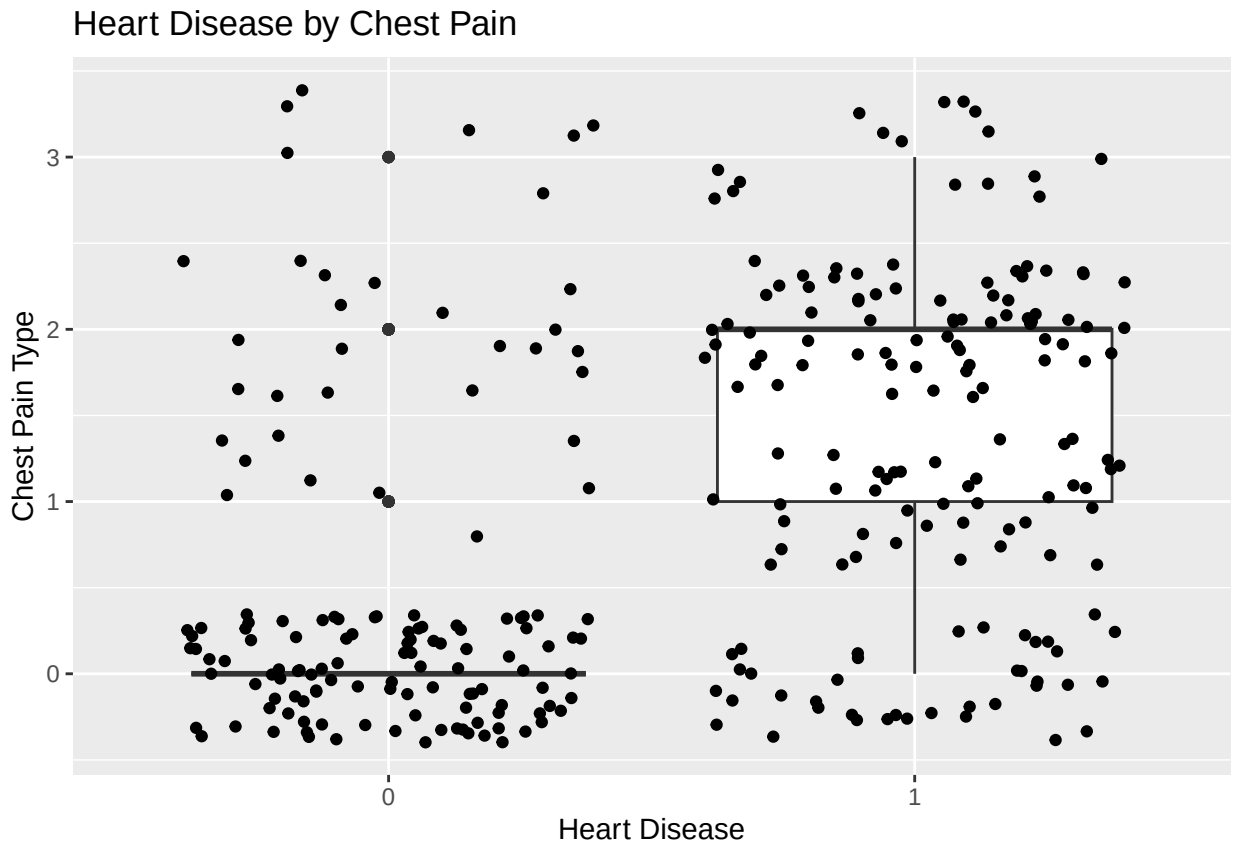
```
thal_counts <- table(heart$thal)
thal_counts <- thal_counts[order(thal_counts, decreasing = TRUE)]
ggplot(data = heart, aes(x = reorder(thal, -thal_counts[thal]))) +
  geom_bar() +
  coord_flip() +
  labs(title = "Bar Plot of Heart Defects",
       x = "Heart Defects",
       y = "Count") +
  theme_minimal() +
  theme(axis.text.y = element_text(hjust = 0.5))
```



We can see that most people are normal, but there are much more reversible defects than fixed defects, which is quite interesting.

Going back to chest pain, let's see how well it indicates the target variable in the patients of this dataset.

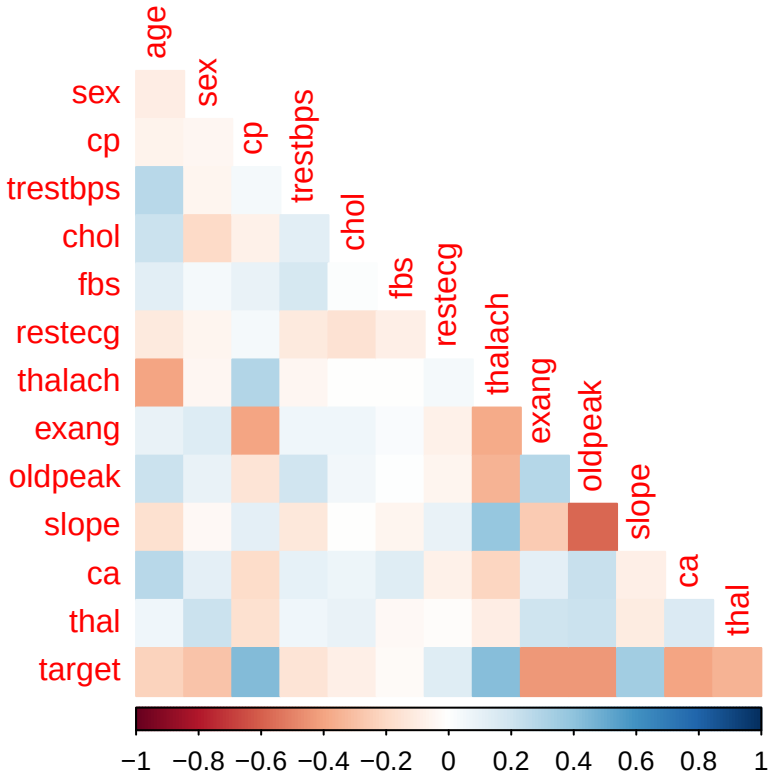
```
heart %>%
  ggplot(aes(x = factor(target), y = cp)) +
    geom_boxplot() +
    geom_jitter(alpha = 2) +
    labs(title = "Heart Disease by Chest Pain",
      x = "Heart Disease",
      y = "Chest Pain Type")
```



This result is in line with what we discovered before. Although type 0 chest pain is more common, most with this form of chest pain experience lower heart rates than those with the other types. And now, we can see that those without heart disease tend to have type 0 chest pain, while those with heart disease are much more evenly distributed throughout the other types.

Now, let us see how all of the numerical variables in this dataset relate to one another, and make some inferences.

```
heart %>%
  select(is.numeric) %>%
  cor() %>%
  corrplot::corrplot(type = 'lower', diag = FALSE,
           method = 'color')
```



The strongest positive correlations exist mainly within the target variable. We can see that having heart disease is strongly correlated to chest pain, maximum heart rate achieved, and slope of peak exercise ST depression segment. That slope is also strongly correlated with max heart rate achieved, which is strongly correlated with chest pain as well. Age also seems to hold positive correlations with resting blood pressure, ST depression induced by exercise relative to rest, and number of major vessels (0-3) colored by flourosopy. ST depression induced by exercise is strongly correlated to exercise induced angina, which makes sense. Most of these variables are positively and linearly related with age.

In terms of the strongest negative correlations, we see that exercise angina is such with the other types of angina chest pain, which also makes perfect sense, as well as maximum heart rate achieved. Old peak falls in line with this, being negatively correlated to max heart rate as well. Max heart rate achieved goes down with age as mentioned before. The target variable seems to be negatively correlated with exercise induced angina, ST depression induced by exercise, number of major vessels (0-3) colored by flourosopy, and defects. This indicates that exercise induced heart problems generally do not lead to heart disease, which one might infer.

Some other takeaways: Younger patients with higher max hearts rates achieved are more likely to have a heart condition.And lower st depression regardless of age is also likely an indication of a heart disease. Features that have higher predictive power could be chest pain type, number of major vessels, thalassemia, exercise induced angina, max heart rate achieved and st depression. We will see which features will appear as important by the classification models.

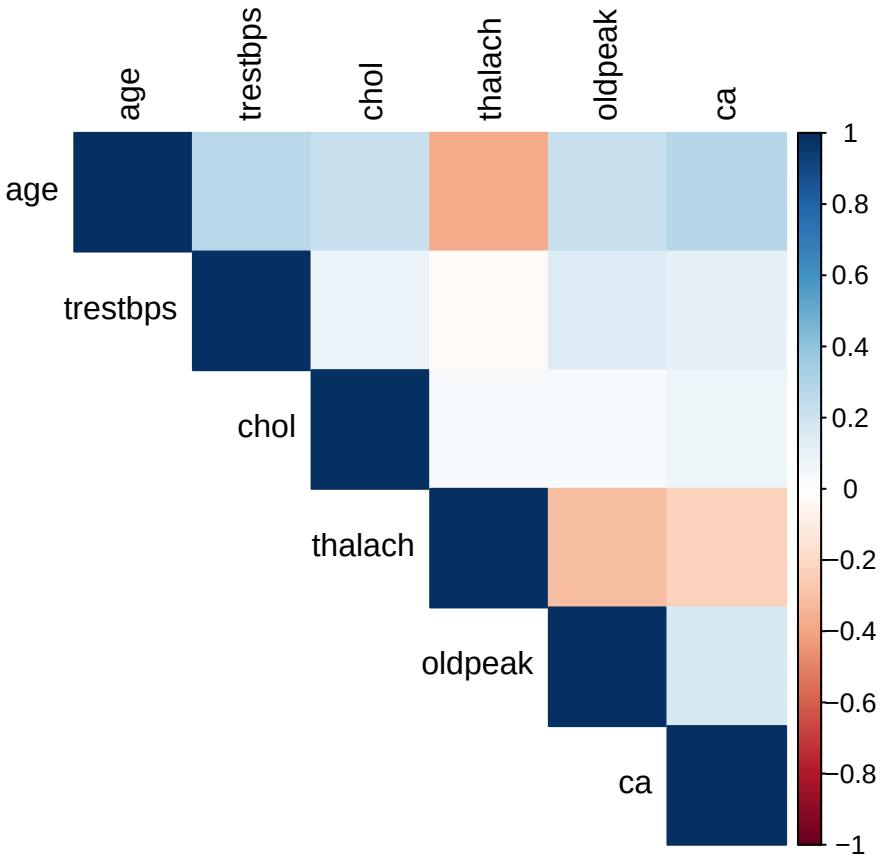
DATA SPLITTING

First, we can split our data into training and testing, and then get into creating a recipe for the data:

```r
set.seed(3436)

heart <- heart %>%
  mutate(target = factor(target)) %>%
  mutate(sex = factor(sex)) %>%
  mutate(cp = factor(cp)) %>%
  mutate(fbs = factor(fbs)) %>%
  mutate(restecg = factor(restecg)) %>%
  mutate(exang = factor(exang)) %>%
  mutate(slope = factor(slope)) %>%
  mutate(thal = factor(thal))

heart_split <- initial_split(heart, prop = 0.70,
                             strata = target)
heart_train <- training(heart_split)
heart_test <- testing(heart_split)

heart_folds <- vfold_cv(heart_train, v = 5, strata = "target")
```

Here's a look at the correlations between the continuous variables in the training data:

```r
continuous_vars <- select_if(heart_train, is.numeric)
correlation_matrix <- cor(continuous_vars)
corrplot(correlation_matrix, method = "color", type = "upper", tl.col = "black")
```



```
print(correlation_matrix)
```

```
##                 age    trestbps       chol     thalach     oldpeak          ca
## age       1.0000000  0.27723504 0.22381094 -0.37873772  0.22269518  0.28646707
## trestbps  0.2772350  1.00000000 0.08850220 -0.02240392  0.14819359  0.10499342
## chol      0.2238109  0.08850220 1.00000000  0.03030674  0.03157391  0.07938541
## thalach  -0.3787377 -0.02240392 0.03030674  1.00000000 -0.30797147 -0.22270073
## oldpeak   0.2226952  0.14819359 0.03157391 -0.30797147  1.00000000  0.17390508
## ca        0.2864671  0.10499342 0.07938541 -0.22270073  0.17390508  1.00000000
```

Now, let's use our training data to create a recipe prediction for our outcome variable.

```
heart_recipe <- recipe(target ~ cp + sex + age + trestbps + chol + fbs + restecg + thalach + exang + old
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with('sex'):thal) %>%
  step_interact(~age:oldpeak)
```

```
heart_recipe <- recipe(target ~ age + trestbps + chol + thalach + oldpeak + ca, data = heart_train) %>%
  step_impute_linear(age) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(~ starts_with('age'):ca) %>%
  step_interact(~age:oldpeak)
```

We have our recipe and stratified sampling, so we can go ahead create some models for classification using various engines. To start, we can use a logistic regression model:

```
logistic_model <- logistic_reg() %>%
  set_engine("glm") %>%
  set_mode("classification")

heart_workflow <- workflow() %>%
  add_model(logistic_model) %>%
  add_recipe(heart_recipe)

heart_fit <- fit(heart_workflow, heart_train)

heart_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
```

```
## -- Model ---------------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##   (Intercept)            age       trestbps           chol        thalach
##     -3.271594       0.016502      -0.007253      -0.001425       0.034121
##       oldpeak             ca       age_x_ca  age_x_oldpeak
##     -0.912218      -0.410883      -0.011171       0.007182
##
## Degrees of Freedom: 210 Total (i.e. Null);  202 Residual
## Null Deviance:       290.8
## Residual Deviance: 207.9     AIC: 225.9
```

Here is a linear discriminant analysis model for classification:

```
lda_model <- discrim_linear() %>%
  set_engine("MASS") %>%
  set_mode("classification")

heart_workflow_lda <- workflow() %>%
  add_recipe(heart_recipe) %>%
  add_model(lda_model)

heart_fit_lda <- fit(heart_workflow_lda, data = heart_train)

heart_fit_lda
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: discrim_linear()
##
## -- Preprocessor ---------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model ----------------------------------------------------------------------
## Call:
## lda(..y ~ ., data = data)
##
## Prior probabilities of groups:
##         0         1
## 0.4549763 0.5450237
##
## Group means:
##        age trestbps     chol  thalach   oldpeak        ca age_x_ca
## 0 56.86458 133.4167 248.6979 139.1562 1.3958333 1.2187500 71.10417
## 1 52.74783 130.5739 244.4696 157.7565 0.6452174 0.3217391 18.51304
##   age_x_oldpeak
```

```
## 0      80.80208
## 1      35.43826
##
## Coefficients of linear discriminants:
##                           LD1
## age             0.0050365052
## trestbps       -0.0020739726
## chol           -0.0007596553
## thalach         0.0254239633
## oldpeak        -0.8450357216
## ca             -0.5150448520
## age_x_ca       -0.0046688354
## age_x_oldpeak   0.0081658894
```

Continuing on, a quadratic discriminant analysis model for classification:

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

heart_workflow_qda <- workflow() %>%
  add_recipe(heart_recipe) %>%
  add_model(qda_model)

heart_fit_qda <- fit(heart_workflow_qda, data = heart_train)

heart_fit_qda
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: discrim_quad()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
## Call:
## qda(..y ~ ., data = data)
##
## Prior probabilities of groups:
##         0         1
## 0.4549763 0.5450237
##
## Group means:
##        age trestbps     chol  thalach   oldpeak        ca age_x_ca
## 0 56.86458 133.4167 248.6979 139.1562 1.3958333 1.2187500 71.10417
## 1 52.74783 130.5739 244.4696 157.7565 0.6452174 0.3217391 18.51304
##   age_x_oldpeak
```

```
## 0      80.80208
## 1      35.43826
```

And now, the fourth and final model will be a KNN model (or k-nearest neighbors model):

```
knn_model <- nearest_neighbor(neighbors = 7) %>%
  set_mode("classification") %>%
  set_engine("kknn")

heart_workflow_knn <- workflow() %>%
  add_recipe(heart_recipe) %>%
  add_model(knn_model)

heart_fit_knn <- fit(heart_workflow_knn, data = heart_train)

heart_fit_knn
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------------
##
## Call:
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(7,     data, 5))
##
## Type of response variable: nominal
## Minimal misclassification: 0.2227488
## Best kernel: optimal
## Best k: 7
```

We can use the metric of area under the ROC Curve to measure each model's performance with the training data:

```
logistic_preds <- predict(heart_fit, new_data = heart_train, type = "prob") %>%
  select(.pred_0)
lda_preds <- predict(heart_fit_lda, new_data = heart_train, type = "prob") %>%
  select(.pred_0)
qda_preds <- predict(heart_fit_qda, new_data = heart_train, type = "prob") %>%
  select(.pred_0)
knn_preds <- predict(heart_fit_knn, new_data = heart_train, type = "prob") %>%
  select(.pred_0)
four_models <- bind_cols(logistic_preds, lda_preds, qda_preds, knn_preds)
four_models
```

```
## # A tibble: 211 x 4
##    .pred_0...1 .pred_0...2 .pred_0...3 .pred_0...4
##          <dbl>       <dbl>       <dbl>       <dbl>
##  1       0.917       0.920       1.00        0.925
##  2       0.569       0.553       0.461       0.886
##  3       0.515       0.494       0.582       0.756
##  4       0.538       0.518       0.369       0.912
##  5       0.173       0.154       0.102       0.405
##  6       0.273       0.253       0.262       0.490
##  7       0.801       0.804       0.998       1
##  8       0.911       0.917       0.991       0.987
##  9       0.733       0.749       0.950       0.769
## 10       0.697       0.704       0.917       1
## # i 201 more rows
```

```
roc_logistic <- roc(heart_train$target, logistic_preds$.pred_0)
roc_lda <- roc(heart_train$target, lda_preds$.pred_0)
roc_qda <- roc(heart_train$target, qda_preds$.pred_0)
roc_knn <- roc(heart_train$target, knn_preds$.pred_0)

cat("Logistic Regression AUC (Training):", auc(roc_logistic), "\n")
```

```
## Logistic Regression AUC (Training): 0.8414855
```

```
cat("Linear Discriminant Analysis AUC (Training):", auc(roc_lda), "\n")
```

```
## Linear Discriminant Analysis AUC (Training): 0.842029
```

```
cat("Quadratic Discriminant Analysis AUC (Training):", auc(roc_qda), "\n")
```

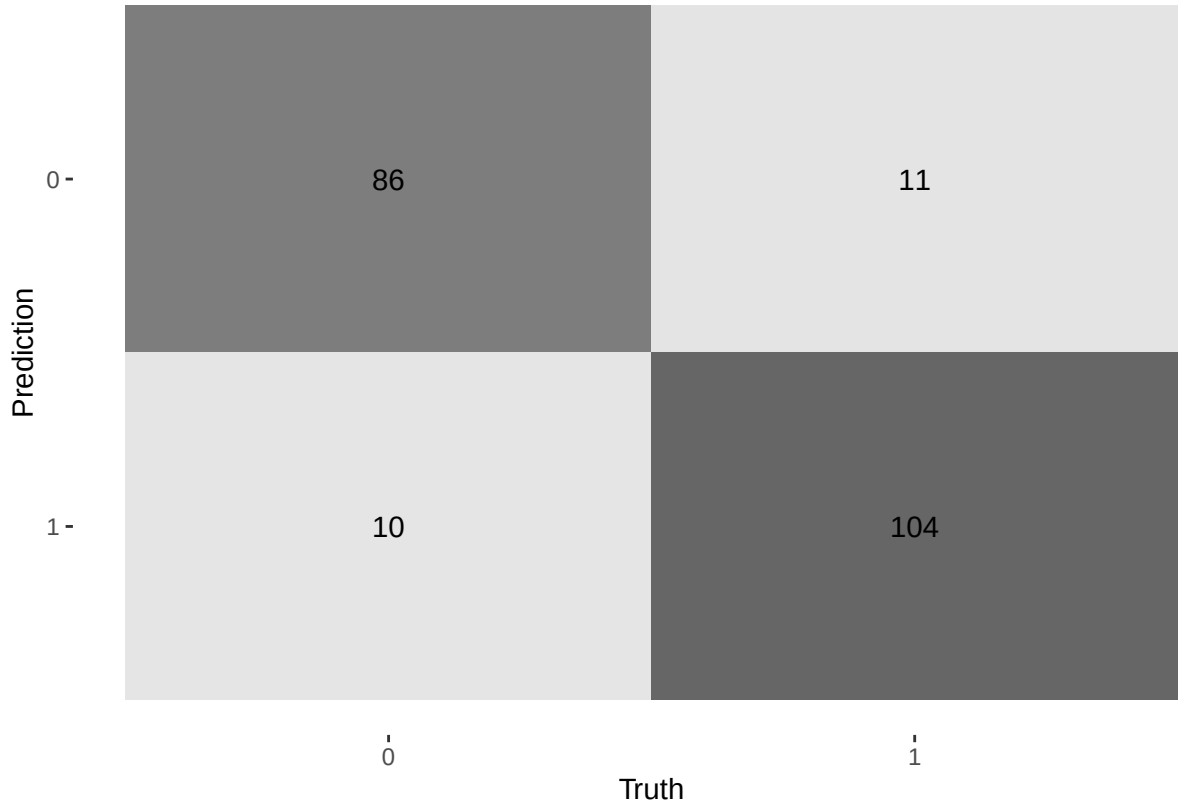
```
## Quadratic Discriminant Analysis AUC (Training): 0.8459239
```

```
cat("K-Nearest Neighbors AUC (Training):", auc(roc_knn), "\n")
```

```
## K-Nearest Neighbors AUC (Training): 0.9814312
```

As we can see, the KNN model was the top performer on the training data. Let's see a confusion matrix of this model:

```
augment(heart_fit_knn, new_data = heart_train) %>%
  conf_mat(truth = target, estimate = .pred_class) %>%
  autoplot(type="heatmap")
```



Next, we will look at this metric with regards to the testing data:

```
logistic_test_preds <- predict(heart_fit, new_data = heart_test, type = "prob") %>%
  select(.pred_0)
lda_test_preds <- predict(heart_fit_lda, new_data = heart_test, type = "prob") %>%
  select(.pred_0)
qda_test_preds <- predict(heart_fit_qda, new_data = heart_test, type = "prob") %>%
  select(.pred_0)
knn_test_preds <- predict(heart_fit_knn, new_data = heart_test, type = "prob") %>%
  select(.pred_0)
roc_logistic_test <- roc(heart_test$target, logistic_test_preds$.pred_0)
roc_lda_test <- roc(heart_test$target, lda_test_preds$.pred_0)
roc_qda_test <- roc(heart_test$target, qda_test_preds$.pred_0)
roc_knn_test <- roc(heart_test$target, knn_test_preds$.pred_0)

cat("Logistic Regression AUC (Testing):", auc(roc_logistic_test), "\n")
```

```
## Logistic Regression AUC (Testing): 0.85
```

```
cat("Linear Discriminant Analysis AUC (Testing):", auc(roc_lda_test), "\n")
```

```
## Linear Discriminant Analysis AUC (Testing): 0.8442857
```

```
cat("Quadratic Discriminant Analysis AUC (Testing):", auc(roc_qda_test), "\n")
```

```
## Quadratic Discriminant Analysis AUC (Testing): 0.7990476
```

```
cat("K-Nearest Neighbors AUC (Testing):", auc(roc_knn_test), "\n")
```

```
## K-Nearest Neighbors AUC (Testing): 0.7595238
```

This time, the logistic model was the top performer, and the KNN model was actually the worst. This means we need to use cross validation to fine tune our data!

CROSS VALIDATION

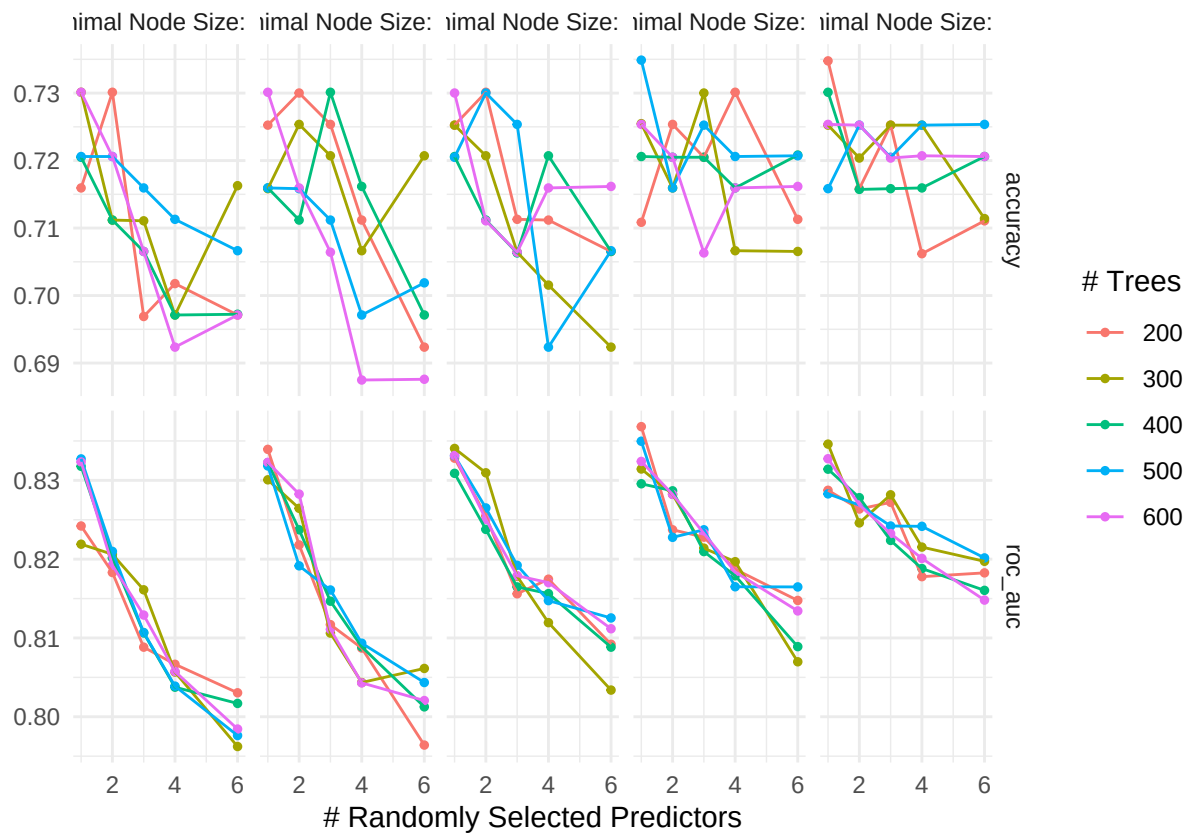
```
rf_class_spec <- rand_forest(mtry = tune(),
                             trees = tune(),
                             min_n = tune()) %>%
  set_engine("ranger") %>%
  set_mode("classification")

rf_class_wf <- workflow() %>%
  add_model(rf_class_spec) %>%
  add_recipe(heart_recipe)
```

```
rf_grid <- grid_regular(mtry(range = c(1, 6)),
                        trees(range = c(200, 600)),
                        min_n(range = c(10, 20)),
                        levels = 5)
rf_grid
```

```
## # A tibble: 125 x 3
##     mtry trees min_n
##    <int> <int> <int>
##  1     1   200    10
##  2     2   200    10
##  3     3   200    10
##  4     4   200    10
##  5     6   200    10
##  6     1   300    10
##  7     2   300    10
##  8     3   300    10
##  9     4   300    10
## 10     6   300    10
## # i 115 more rows
```

```
tune_class <- tune_grid(
  rf_class_wf,
  resamples = heart_folds,
  grid = rf_grid
)
```

```
autoplot(tune_class) + theme_minimal()
```




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

```
## # A tibble: 1 x 9
##    mtry trees min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     1   200    17 roc_auc binary     0.837     5  0.0443 Preprocessor1_Model0~
```

```
best_rf_class <- select_best(tune_class)
```

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

bt_class_wf <- workflow() %>%
  add_model(bt_class_spec) %>%
  add_recipe(heart_recipe)
```

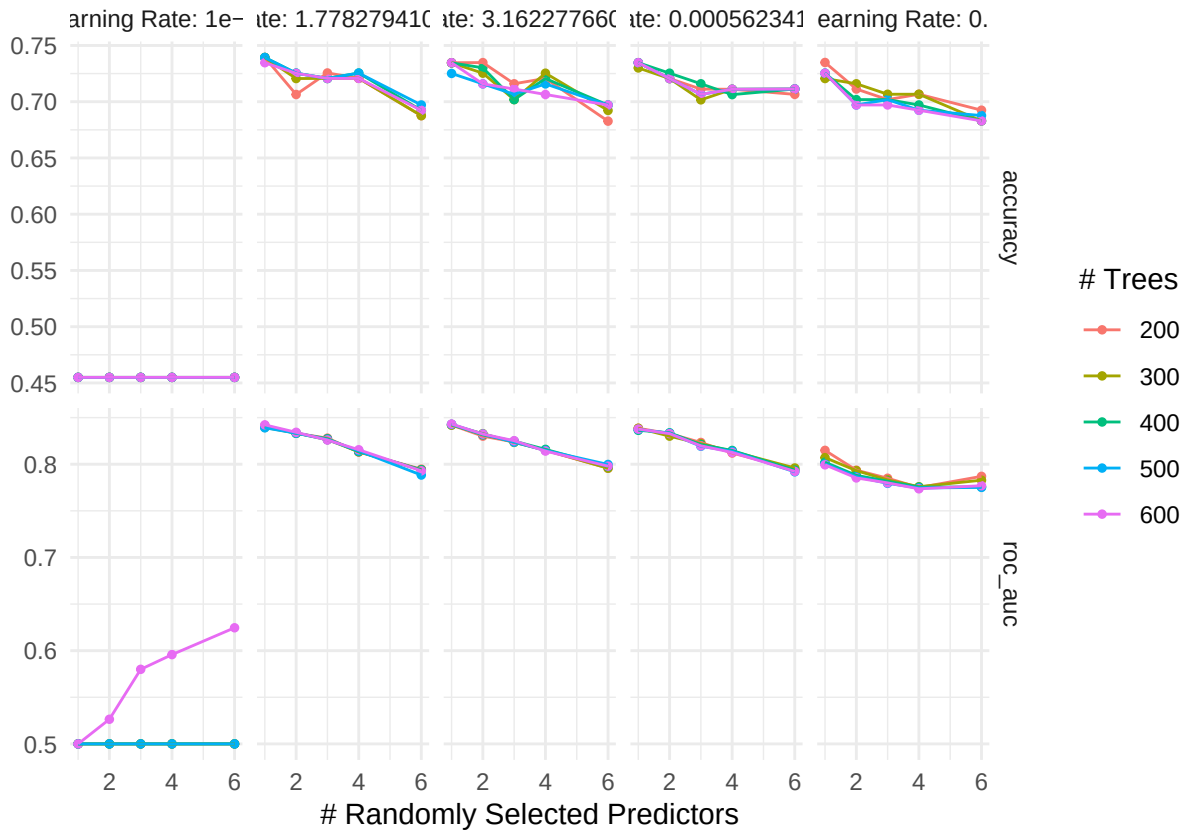
```
bt_grid <- grid_regular(mtry(range = c(1, 6)),
                        trees(range = c(200, 600)),
```

```
                        learn_rate(range = c(-10, -1)),
                        levels = 5)
bt_grid
```

```
## # A tibble: 125 x 3
##     mtry trees   learn_rate
##    <int> <int>        <dbl>
##  1     1   200 0.0000000001
##  2     2   200 0.0000000001
##  3     3   200 0.0000000001
##  4     4   200 0.0000000001
##  5     6   200 0.0000000001
##  6     1   300 0.0000000001
##  7     2   300 0.0000000001
##  8     3   300 0.0000000001
##  9     4   300 0.0000000001
## 10     6   300 0.0000000001
## # i 115 more rows
```

```
tune_bt_class <- tune_grid(
  bt_class_wf,
  resamples = heart_folds,
  grid = bt_grid
)
```

```
autoplot(tune_bt_class) + theme_minimal()
```




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

```
## # A tibble: 1 x 9
##    mtry trees learn_rate .metric .estimator  mean     n std_err .config
##   <int> <int>      <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     1   600 0.00000316 roc_auc binary     0.843     5  0.0487 Preprocessor1_M~
```

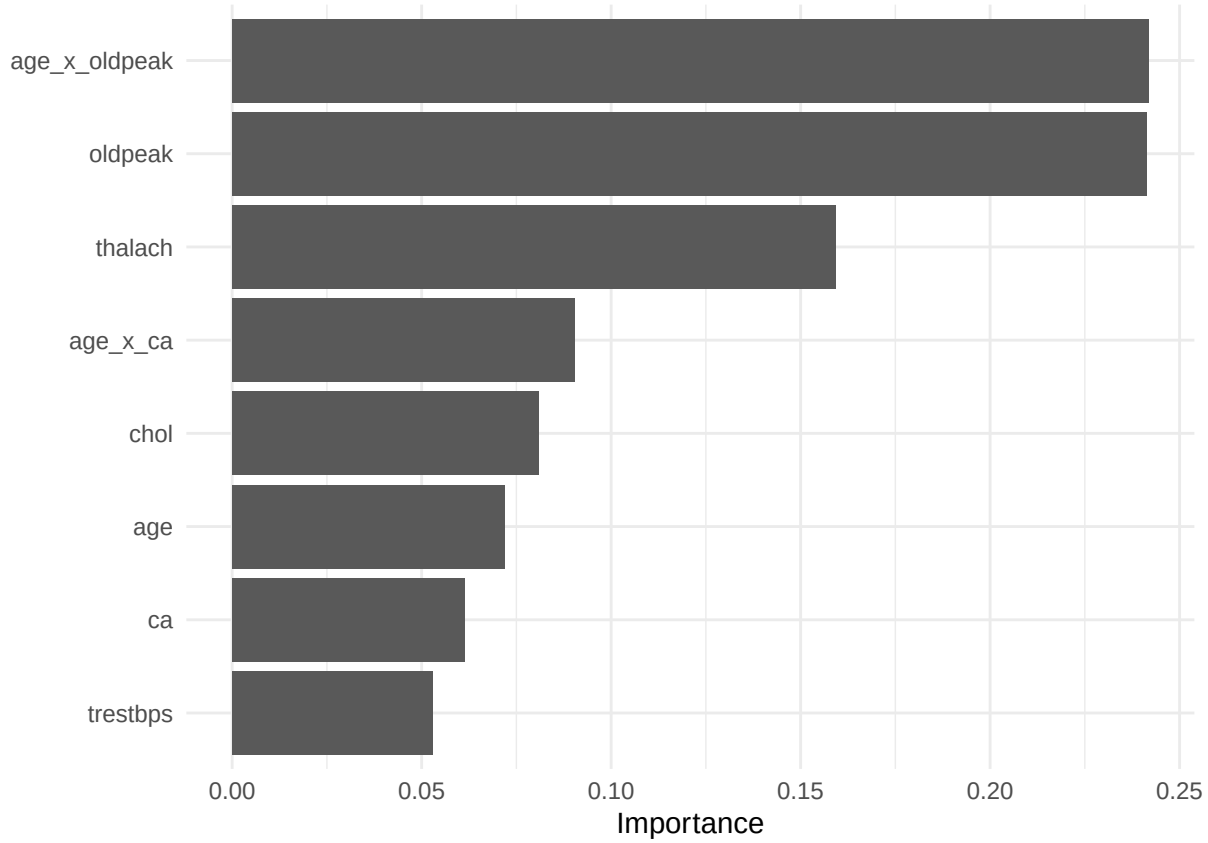
```
best_bt_class <- select_best(tune_bt_class)
```

We have gone through a long and tedious tuning process, so now, let's focus on interpreting the results!

MODEL SELECTION AND PERFORMANCE

```
final_bt_model <- finalize_workflow(bt_class_wf, best_bt_class)
final_bt_model <- fit(final_bt_model, heart_test)
```

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



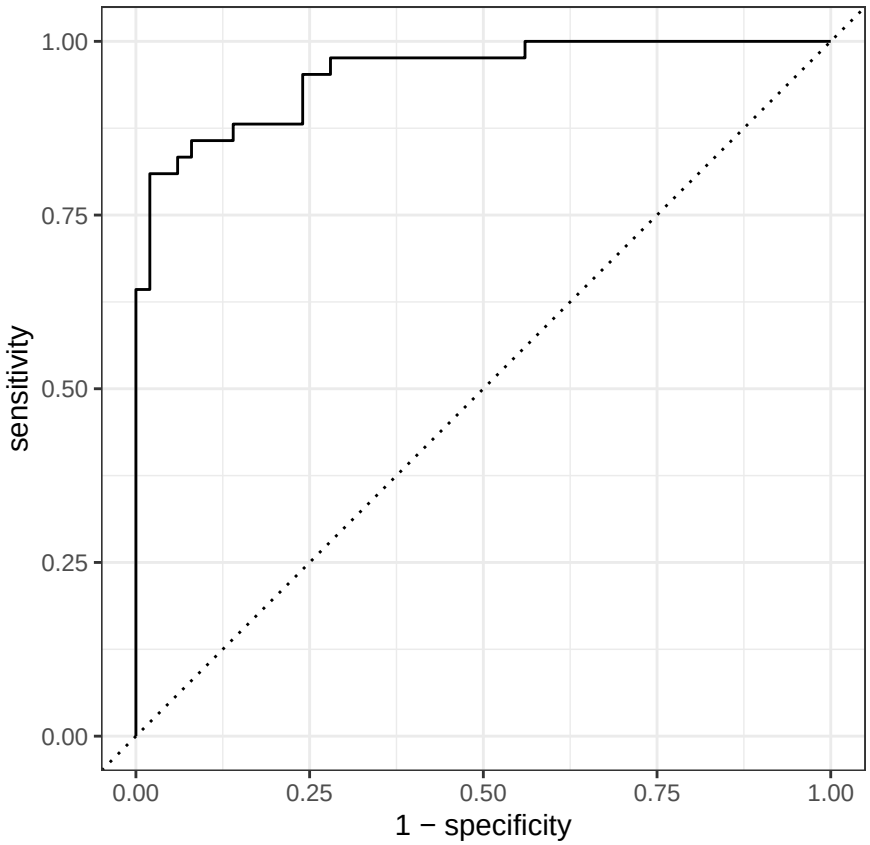
This tells us that the top three most important predictors of heart disease were ST depression induced by exercise relative to rest, age, and maximum heart rate achieved.

```
final_bt_model_test <- augment(final_bt_model,
                               heart_test) %>%
  select(target, starts_with(".pred"))

roc_auc(final_bt_model_test, truth = target, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.953
```

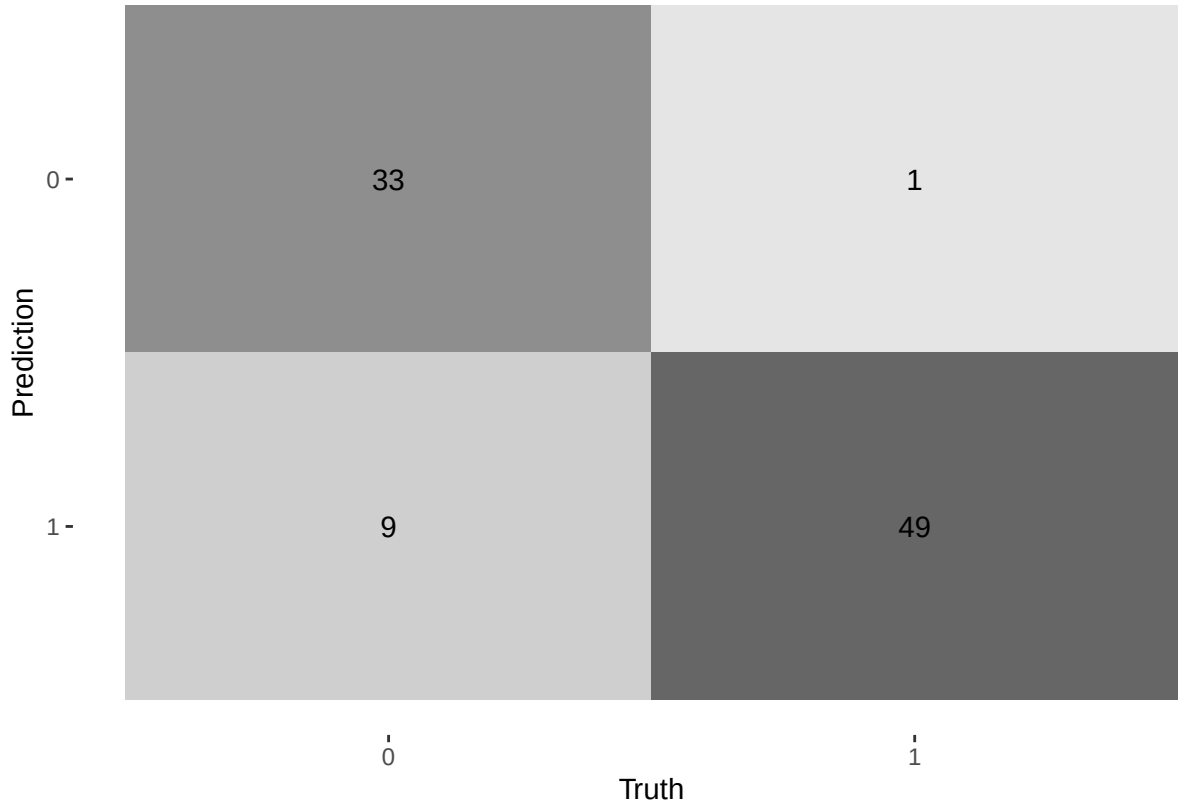
The ROC AUC on the testing data is 0.95, which is somewhat close to 1 but not extremely close; the model has done a decent job classifying the target variable. Let's generate the ROC curve:

```
roc_curve(final_bt_model_test, truth = target, .pred_0) %>%
  autoplot()
```



And the confusion matrix:

```
conf_mat(final_bt_model_test, truth = target,
         .pred_class) %>%
  autoplot(type = "heatmap")
```



The best-performing boosted tree model has done a great job, but not perfect by any means, and thats ok!

MODEL FITTING

To further tune our models and come to the conclusion of the best model between KNN and logistic, we need to fit them to our folded data using grids and workflows, and then collect their ROC AUC metrics:

```
heart_grid <- grid_regular(neighbors(range = c(1, 10)), levels = 10)
knn_mod <- nearest_neighbor(neighbors = tune()) %>%
  set_mode("classification") %>%
  set_engine("kknn")
wflow_kknn <- workflow() %>%
  add_model(knn_mod) %>%
  add_recipe(heart_recipe)
```

```
log_mod <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm")
log_workflow <- workflow() %>%
  add_model(log_mod) %>%
  add_recipe(heart_recipe)
```

```
tune_knn_heart <- tune_grid(
  wflow_kknn,
  resamples = heart_folds,
```

```
  grid = heart_grid
)
tune_log_heart <- tune_grid(
  log_workflow,
  resamples = heart_folds
)
```

```
collect_metrics(tune_knn_heart)
```

```
## # A tibble: 20 x 7
##    neighbors .metric  .estimator  mean     n std_err .config
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1         1 accuracy binary     0.763     5  0.0302 Preprocessor1_Model01
##  2         1 roc_auc  binary     0.760     5  0.0304 Preprocessor1_Model01
##  3         2 accuracy binary     0.763     5  0.0302 Preprocessor1_Model02
##  4         2 roc_auc  binary     0.793     5  0.0348 Preprocessor1_Model02
##  5         3 accuracy binary     0.763     5  0.0302 Preprocessor1_Model03
##  6         3 roc_auc  binary     0.802     5  0.0426 Preprocessor1_Model03
##  7         4 accuracy binary     0.763     5  0.0302 Preprocessor1_Model04
##  8         4 roc_auc  binary     0.816     5  0.0419 Preprocessor1_Model04
##  9         5 accuracy binary     0.749     5  0.0315 Preprocessor1_Model05
## 10         5 roc_auc  binary     0.820     5  0.0405 Preprocessor1_Model05
## 11         6 accuracy binary     0.749     5  0.0315 Preprocessor1_Model06
## 12         6 roc_auc  binary     0.819     5  0.0402 Preprocessor1_Model06
## 13         7 accuracy binary     0.749     5  0.0315 Preprocessor1_Model07
## 14         7 roc_auc  binary     0.821     5  0.0407 Preprocessor1_Model07
## 15         8 accuracy binary     0.749     5  0.0300 Preprocessor1_Model08
## 16         8 roc_auc  binary     0.822     5  0.0401 Preprocessor1_Model08
## 17         9 accuracy binary     0.754     5  0.0290 Preprocessor1_Model09
## 18         9 roc_auc  binary     0.831     5  0.0378 Preprocessor1_Model09
## 19        10 accuracy binary     0.754     5  0.0349 Preprocessor1_Model10
## 20        10 roc_auc  binary     0.830     5  0.0384 Preprocessor1_Model10
```

```
collect_metrics(tune_log_heart)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.749     5  0.0271 Preprocessor1_Model1
## 2 roc_auc  binary     0.818     5  0.0395 Preprocessor1_Model1
```

The logistic model performed the best because it has the highest roc_auc on average.

We have chosen our best model, so we can fit it to the entire training set, and assess its performance across the testing set.

```
final_wf_heart <- finalize_workflow(log_workflow, tune_log_heart$.metrics)
final_wf_heart
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
```

```
##
## -- Preprocessor ----------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```
final_fit_heart <- fit(final_wf_heart, heart_train)
final_fit_heart
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------
## 4 Recipe Steps
##
## * step_impute_linear()
## * step_dummy()
## * step_interact()
## * step_interact()
##
## -- Model -----------------------------------------------------------------
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
##   (Intercept)            age       trestbps           chol        thalach
##     -3.271594       0.016502      -0.007253      -0.001425       0.034121
##       oldpeak             ca       age_x_ca  age_x_oldpeak
##     -0.912218      -0.410883      -0.011171       0.007182
##
## Degrees of Freedom: 210 Total (i.e. Null);  202 Residual
## Null Deviance:       290.8
## Residual Deviance: 207.9     AIC: 225.9
```

```
augment(final_fit_heart, new_data = heart_test) %>%
  roc_auc(target, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.850
```

As we can see, the testing ROC AUC is higher than the average across all folds.

CONCLUSION

At the start of this project, we laid out what we wanted to do; to explore the heart disease data set (EDA) and practice classification modeling.. In part one (EDA) we explored the data set, checked for missing data, and did other pre-processing. We also tried to identify correlation between features and also with the target variable. Then we practiced how to set up base models and finally arriving at our best model via tuning. To summarize:

Our best model happens to be the logistic model, which we tuned. KNN model was also a decent fit, but the others were clearly the worst.

According to our importance calculation, the three most important features of the model are ST depression induced by exercise relative to rest, age, and maximum heart rate achieved. These features are also among better correlated features from our EDA. I was not surprised by max heart rate being of high importance, but the others surprised me due to the fact that I expected some of the other factors mentioned in the introduction to be of more relevance.