Cardiovascular Disease Prediction with Machine Learning

Andrew Choi

2022-10-16

Contents

Introduction

This project will attempt to predict whether a patient has cardiovascular disease using different types of classification models.

What is Cardiovascular Disease?

Cardiovascular disease refers to a group of health conditions affecting the heart and blood vessels. Some examples of cardiovascular disease include coronary artery disease, pericardial disease, and cardiomyopathy. Heart and blood vessel disease is one of the leading causes of death in the United States, killing approximately 696,962 people per year. In addition, the necessary health care, medication, and premature deaths caused by coronary and blood vessel conditions cause an economic burden of $219 billion in the U.S. alone each year. Both of these statistics are directly from the CDC (Centers for Disease Control and Prevention). Cardiovascular disease is often so lethal because many people don't show symptoms of their condition until the damage is irreversible. Although the effects of cardiovascular disease are devastating to everyone, early or on-time detection by healthcare professionals have the potential to save patients who might otherwise pass away or face life-altering health conditions.

How Might This Model be Used?

As mentioned, the earlier cardiovascular disease is detected, the better the chances of survival and recovery. This model will only require arguments that are quick and easy for healthcare personnel to measure, and will try to predict with high accuracy whether or not a patient has cardiovascular disease. Although other forms of heart and blood vessel disease testing exist, they can be inaccessible for patients without health insurance or time. The aim of this model is to ensure that people who have symptoms of cardiovascular disease, patients who have close family members with a history of heart conditions, or ordinary folk simply worried about their cardiac health have the means of receiving affordable and accurate cardiovascular disease screening.

Overview of Patient Data

Data on 68,783 patients has been obtained from the following data set: https://www.kaggle.com/datasets/aiaiaidavid/cardio-data-dv13032020 There are a total of 12 variables that have been given the following descriptions:

- `AGE`: integer (years of age)
- `HEIGHT`: integer (cm)
- `WEIGHT`: integer (kg)
- `GENDER`: categorical (1: female, 2: male)
- `AP_HIGH`: systolic blood pressure, integer
- `AP_LOW`: diastolic blood pressure, integer
- `CHOLESTEROL`: categorical (1: normal, 2: above normal, 3: well above normal)
- `GLUCOSE`: categorical (1: normal, 2: above normal, 3: well above normal)
- `SMOKE`: categorical (0: no, 1: yes)
- `ALCOHOL`: categorical (0: no, 1: yes)
- `PHYSICAL_ACTIVITY`: categorical (0: no, 1: yes)
- `CARDIO_DISEASE`: categorical (0: no, 1: yes)

Loading and Cleaning Raw Data

Although the data set has already been cleaned, we can take several steps to ensure that the data and column names are as tidy as possible.

```
# Loading in the data set
cardio <- read.csv('cardiovascular_diseases_dv3 2.csv', sep = ';')
```

- Cleaning variable names

```
cardio <- cardio %>%
  clean_names()
```

- Converting height to feet, weight to pounds, and pounds to integer values
- Converting categorical variables to factors
- One-hot-encoding our factor variables
- Removing one dummy variable from each category to avoid the dummy variable trap

```
# Preparing cardio data set for use in correlation heat map
cardio_cor <- cardio

# Preparing cardio data set for use in histogram and bar plots
cardio <- cardio %>%
  mutate(
    # Converting height from centimeters to feet
    height = height * 0.0328084,
    # Converting weight from kilograms to pounds
    weight = round(weight * 2.20462))

# Converting categorical variables to factors
cardio$gender <- as.factor(cardio$gender)
cardio$cholesterol <- as.factor(cardio$cholesterol)
```

```r
cardio$glucose <- as.factor(cardio$glucose)
cardio$smoke <- as.factor(cardio$smoke)
cardio$alcohol <- as.factor(cardio$alcohol)
cardio$physical_activity <- as.factor(cardio$physical_activity)
cardio$cardio_disease <- as.factor(cardio$cardio_disease)

# One-hot-encoding all factors
cardio_one_hot <- one_hot(as.data.table(cardio))

cardio_one_hot <- cardio_one_hot %>%
  mutate(
    # Converting height from centimeters to feet
    height = height * 0.0328084,
    # Converting weight from kilograms to pounds
    weight = round(weight * 2.20462),
    # Converting new one-hot-encoded variables to factors
    cardio_disease_1 = factor(cardio_disease_1),
    gender_2 = factor(gender_2),
    cholesterol_2 = factor(cholesterol_2),
    cholesterol_3 = factor(cholesterol_3),
    glucose_2 = factor(glucose_2),
    glucose_3 = factor(glucose_3)
  )

# Removing one dummy variable from each category
cardio_one_hot[ , c('gender_1', 'cholesterol_1', 'glucose_1', 'smoke_0', 'alcohol_0', 'physical_activity
```

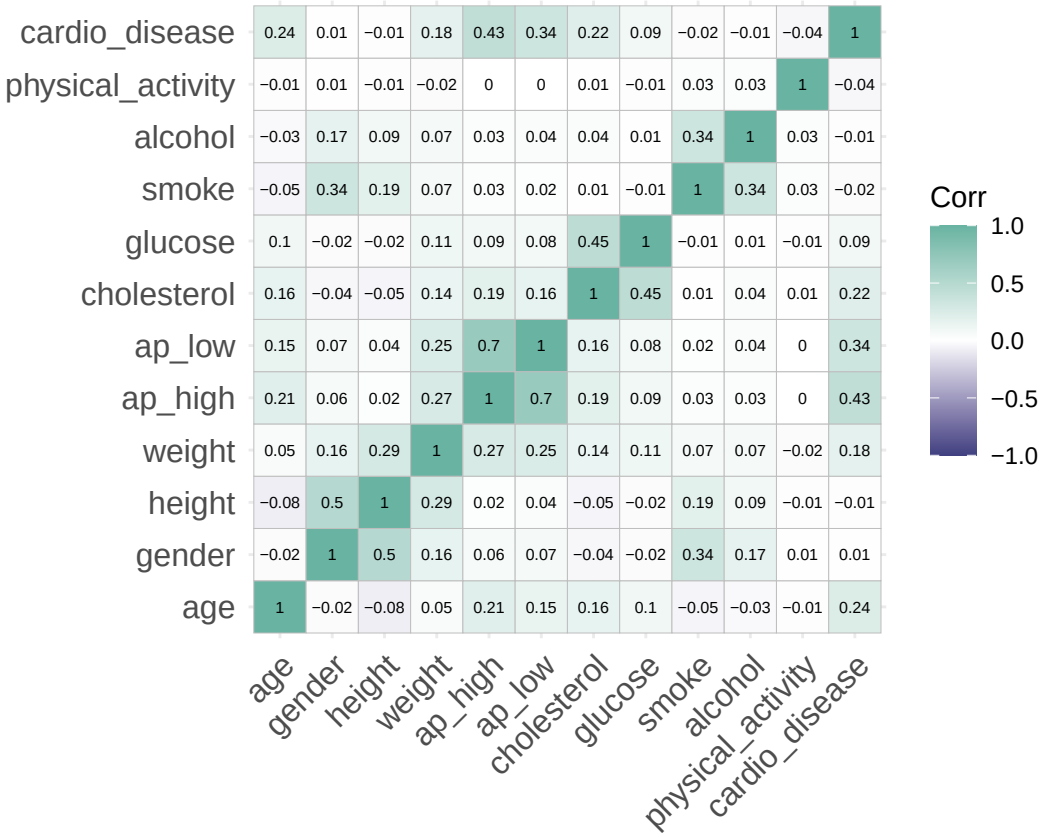
Exploratory Data Analysis

Although it's not likely that any single variable in our data set directly causes or prevents cardiovascular disease, there may be a link between certain variables and coronary disease. We will visualize these links through a correlation plot, as well as individual barplots of our predictor variables against `cardio_disease`.

Correlation Plot

Creating the equivalent of a correlation plot between our variables can help determine which predictors have the largest impact on `cardio_disease`.

```r
cor(cardio_cor, use = 'all.obs') %>%
  ggcorrplot(colors = c('#404080', 'white',  '#69b3a2'), lab = TRUE, lab_size = 2)
```

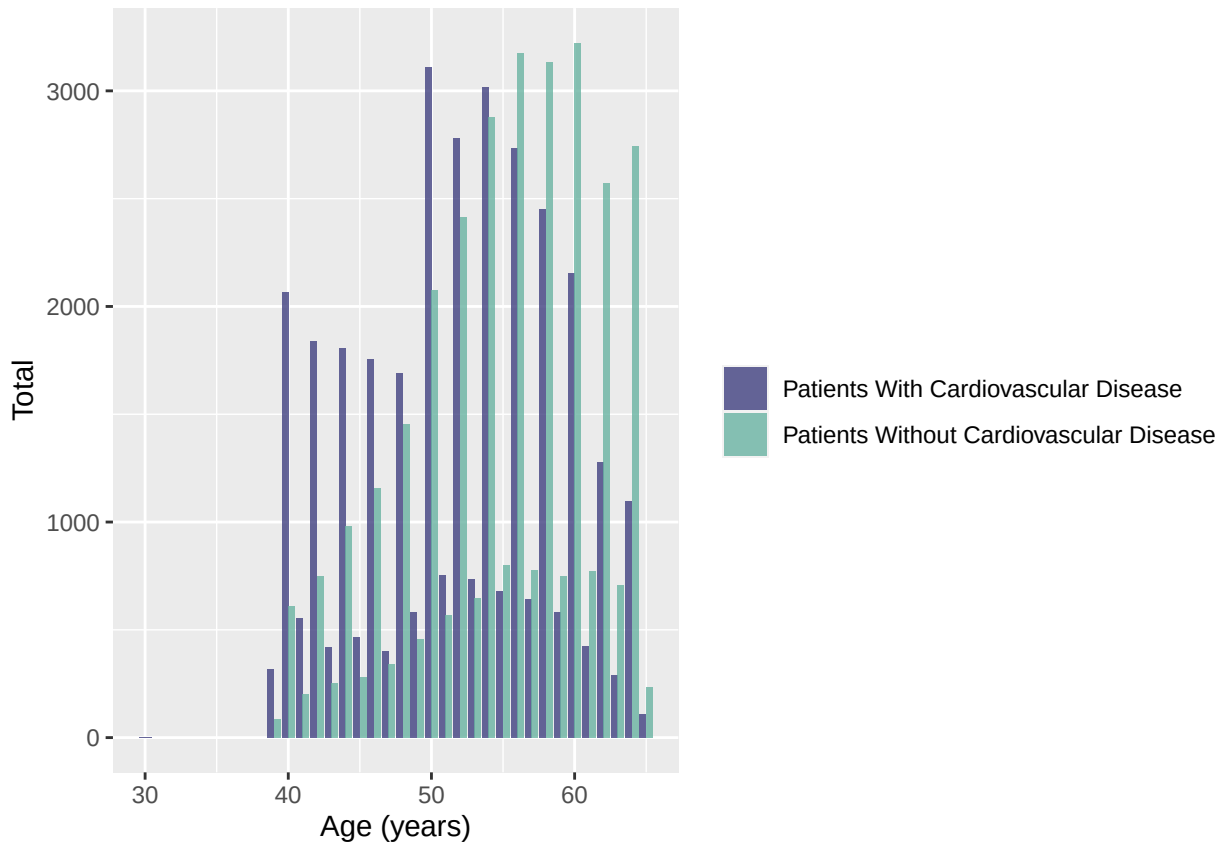


Many of the results of this heat map show connections that were expected. For example, the fact that age, weight, diastolic and systolic blood pressure, and high cholesterol levels are all positively correlated with cardiovascular disease is unsurprising. However, there are also a few results that are a bit more interesting. Smoking, drinking alcohol, and exercise all have close to no correlation with cardiovascular disease. This can be attributed to a variety of factors, one of which is the fact that `smoke`, `alcohol`, and `physical_activity` are all binary variables. This implies that someone who smokes a pack of Marlboro reds a day, drinks enough alcohol to score an integer value BAC percentage, and lives an extremely sedentary lifestyle is given the same score as someone who smokes as rarely as they drink, and chooses not to exercise. Another possible factor is that every person in this data set drinks, smokes, and exercises so moderately that their cardiovascular health is not affected. We can continue to explore the relationships between our predictors and cardiovascular disease by generating histograms and barplots between these variables.

Age In general, it appears as though from ages 40 to 55, patients have higher rates of cardiovascular disease. However, from ages 55 to 70, patients have significantly lower rates of cardiovascular disease. These results go against intuition; It's common sense that heart and artery disease occur most often in the elderly. Yet according to the CDC,'...high rates of obesity and high blood pressure among younger people are putting them at risk for heart disease earlier in life.' This partially explains why our histogram has shows high rates of cardiovascular disease in younger patients. In addition, the elderly are often much more conscientious of their health because they know they're at risk of a variety of complications if they're not careful, which explains the lower rates of cardiovascular disease in our elderly patient population.

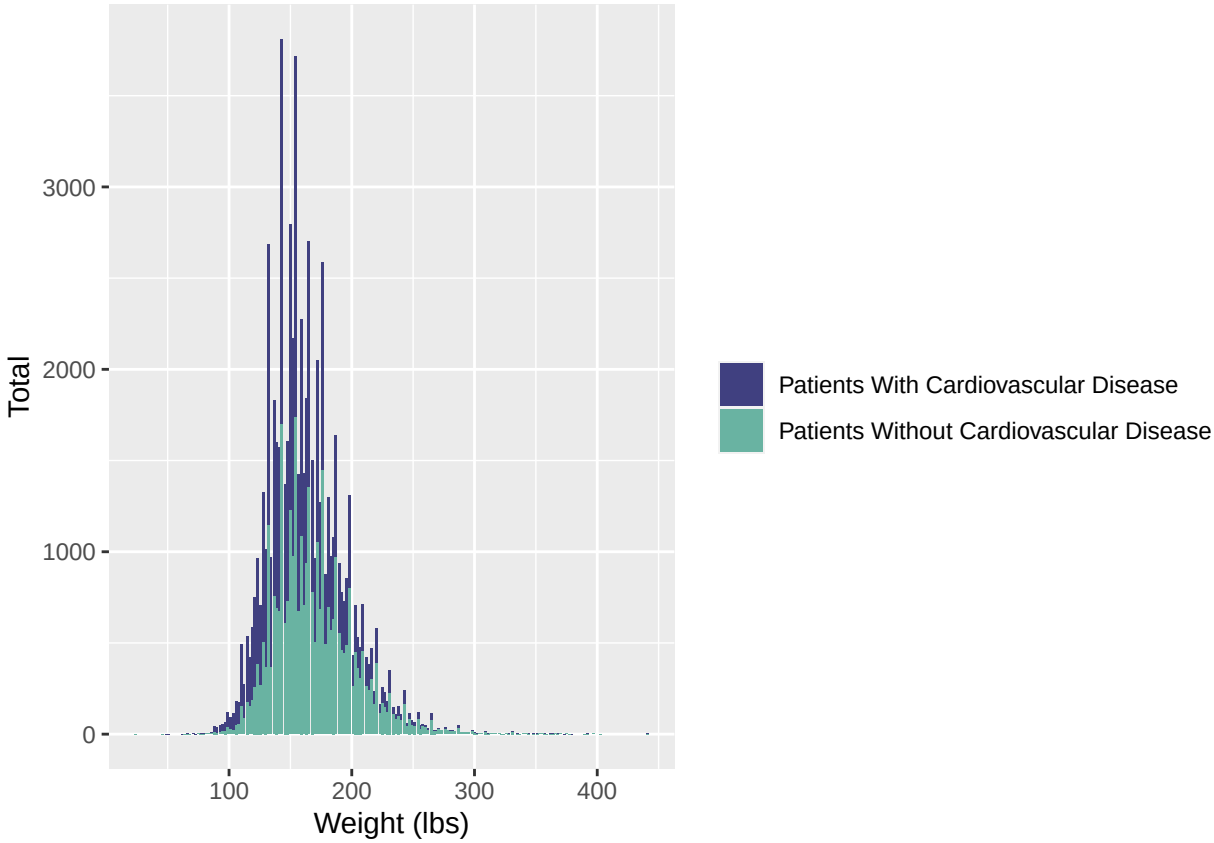
```
# Creating data frame that contains 'age' and 'cardio_disease' if 'cardio_disease' is 0
cardio_disease_age_0 <- select(
  cardio[cardio$cardio_disease == 0, ],
  c('age', 'cardio_disease')
  )
```

```r
cardio_disease_age_0$disease <- 'Patients With Cardiovascular Disease'


# Creating data frame that contains 'age' and 'cardio_disease' if 'cardio_disease' is 1
cardio_disease_age_1 <- select(
  cardio[cardio$cardio_disease == 1, ],
  c('age', 'cardio_disease')
  )

cardio_disease_age_1$disease <- 'Patients Without Cardiovascular Disease'


# Combining the new data frames
cardio_age <- rbind(
  cardio_disease_age_0,
  cardio_disease_age_1
  )


# Plotting the data side by side
ggplot(cardio_age, aes(x = age)) +
  geom_bar(stat = 'count', aes(fill = disease), position = 'dodge', alpha = .8) +
  scale_fill_manual(values = c('#404080', '#69b3a2'), name = '') +
  labs(x = 'Age (years)', y = 'Total')
```



Weight Extreme values of weight don't appear to have higher or lower rates of cardiovascular disease when compared to the average values of weight. In fact, it seems as though patients with average weights have cardiovascular conditions much more commonly than patients with weights that are much higher or lower than the average.

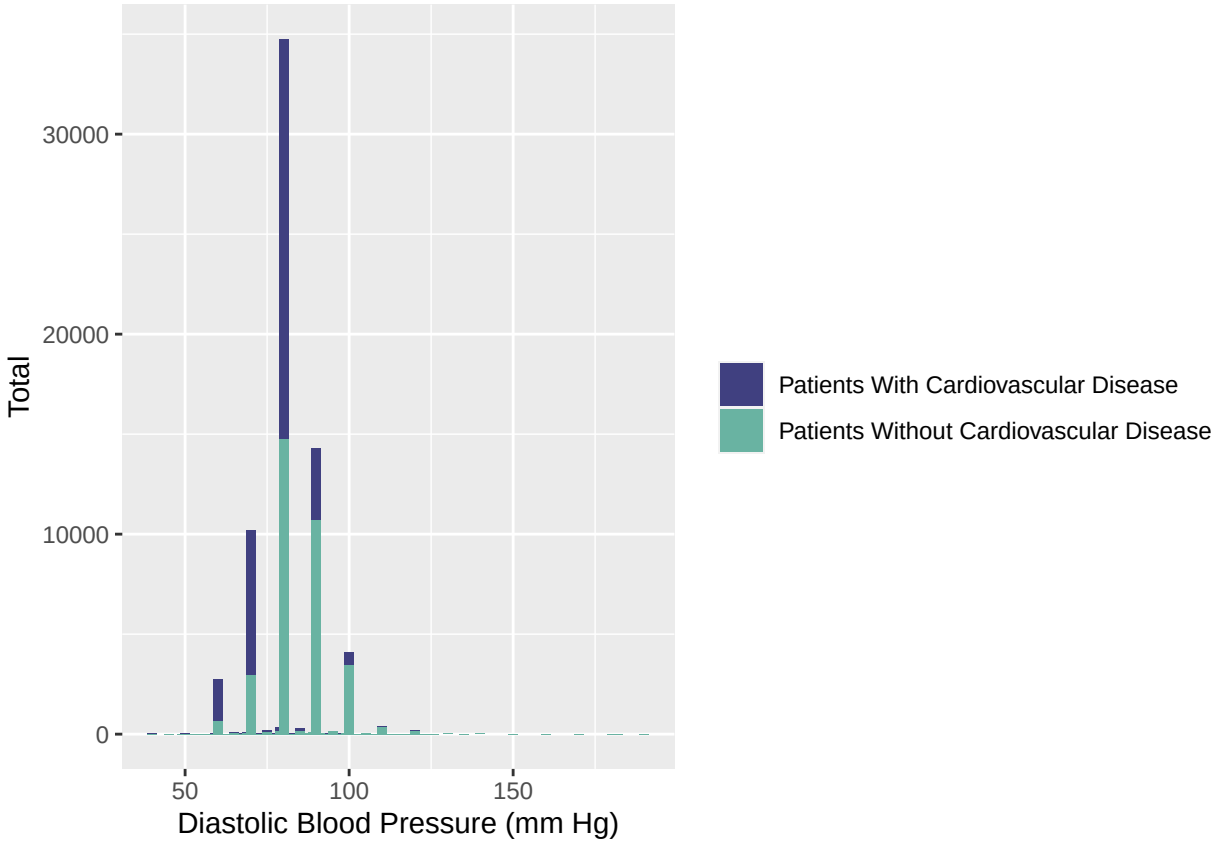
```
# Creating data frame that contains 'weight' and 'cardio_disease' if 'cardio_disease' is 0
cardio_disease_weight_0 <- select(
  cardio[cardio$cardio_disease == 0,],
  c('weight' ,'cardio_disease')
)

cardio_disease_weight_0$disease <- 'Patients With Cardiovascular Disease'


# Creating data frame that contains 'weight' and 'cardio_disease' if 'cardio_disease' is 1
cardio_disease_weight_1 <- select(
  cardio[cardio$cardio_disease == 1,],
  c('weight', 'cardio_disease')
)

cardio_disease_weight_1$disease <- 'Patients Without Cardiovascular Disease'


# Combining the new data frames
cardio_weight <- rbind(
  cardio_disease_weight_0,
  cardio_disease_weight_1
)


# Plotting the data
ggplot(cardio_weight, aes(x = weight, fill = disease), alpha = .8) +
  geom_bar() +
  labs(x = 'Weight (lbs)', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69b3a2'), name = '')
```



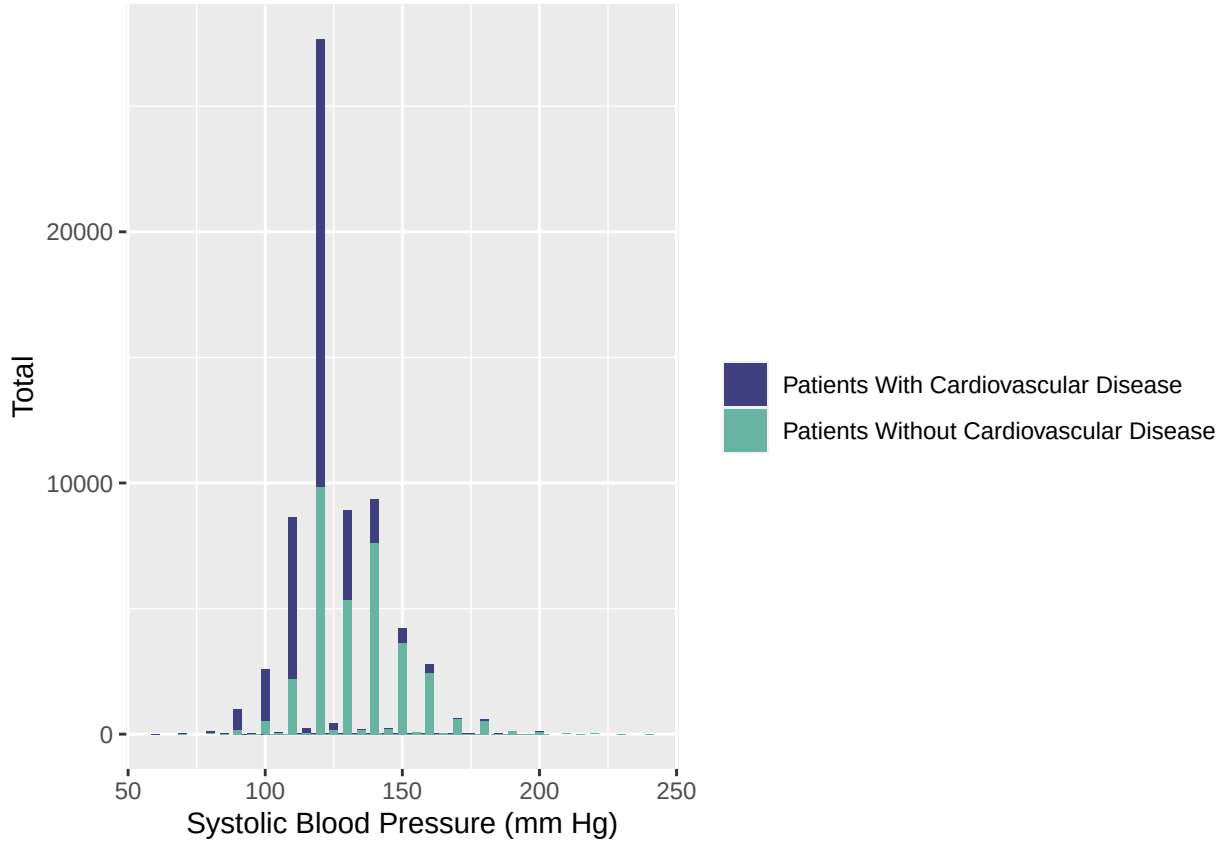
AP High (Systolic Blood Pressure) and AP Low (Diastolic Blood Pressure) Similar to the `weight` variable, patients with average values of systolic and diastolic blood pressure appear to have higher rates of cardiovascular disease than patients with more extreme levels of blood pressure.

```
## Prepping data set to plot Systolic Blood Pressure against Cardiovascular Disease
# Creating data frame that contains 'ap_high' and 'cardio_disease' if 'cardio_disease' is 0
cardio_disease_ap_high_0 <- select(
  cardio[cardio$cardio_disease == 0,],
  c('ap_high' ,'cardio_disease')
)

cardio_disease_ap_high_0$disease <- 'Patients With Cardiovascular Disease'


# Creating data frame that contains 'ap_high' and 'cardio_disease' if 'cardio_disease' is 1
cardio_disease_ap_high_1 <- select(
  cardio[cardio$cardio_disease == 1,],
  c('ap_high', 'cardio_disease')
)

cardio_disease_ap_high_1$disease <- 'Patients Without Cardiovascular Disease'


# Combining the new data frames
cardio_ap_high <- rbind(
```

```
  cardio_disease_ap_high_0,
  cardio_disease_ap_high_1
)



## Prepping data set to plot Diastolic Blood Pressure against Cardiovascular Disease
# Creating data frame that contains 'ap_low' and 'cardio_disease' if 'cardio_disease' is 0
cardio_disease_ap_low_0 <- select(
  cardio[cardio$cardio_disease == 0,],
  c('ap_low' ,'cardio_disease')
)

cardio_disease_ap_low_0$disease <- 'Patients With Cardiovascular Disease'


# Creating data frame that contains 'ap_low' and 'cardio_disease' if 'cardio_disease' is 1
cardio_disease_ap_low_1 <- select(
  cardio[cardio$cardio_disease == 1,],
  c('ap_low', 'cardio_disease')
)

cardio_disease_ap_low_1$disease <- 'Patients Without Cardiovascular Disease'


# Combining the new data frames
cardio_ap_low <- rbind(
  cardio_disease_ap_low_0,
  cardio_disease_ap_low_1
)


# Plotting Diastolic Blood Pressure and Systolic Blood Pressure side by side
par(mfrow = c(2, 1))

# Plotting Diastolic Blood Pressure
ggplot(cardio_ap_low, aes(x = ap_low, fill = disease), alpha = .7) +
  geom_bar(width = 3) +
  labs(x = 'Diastolic Blood Pressure (mm Hg)', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69b3a2'), name = '')
```



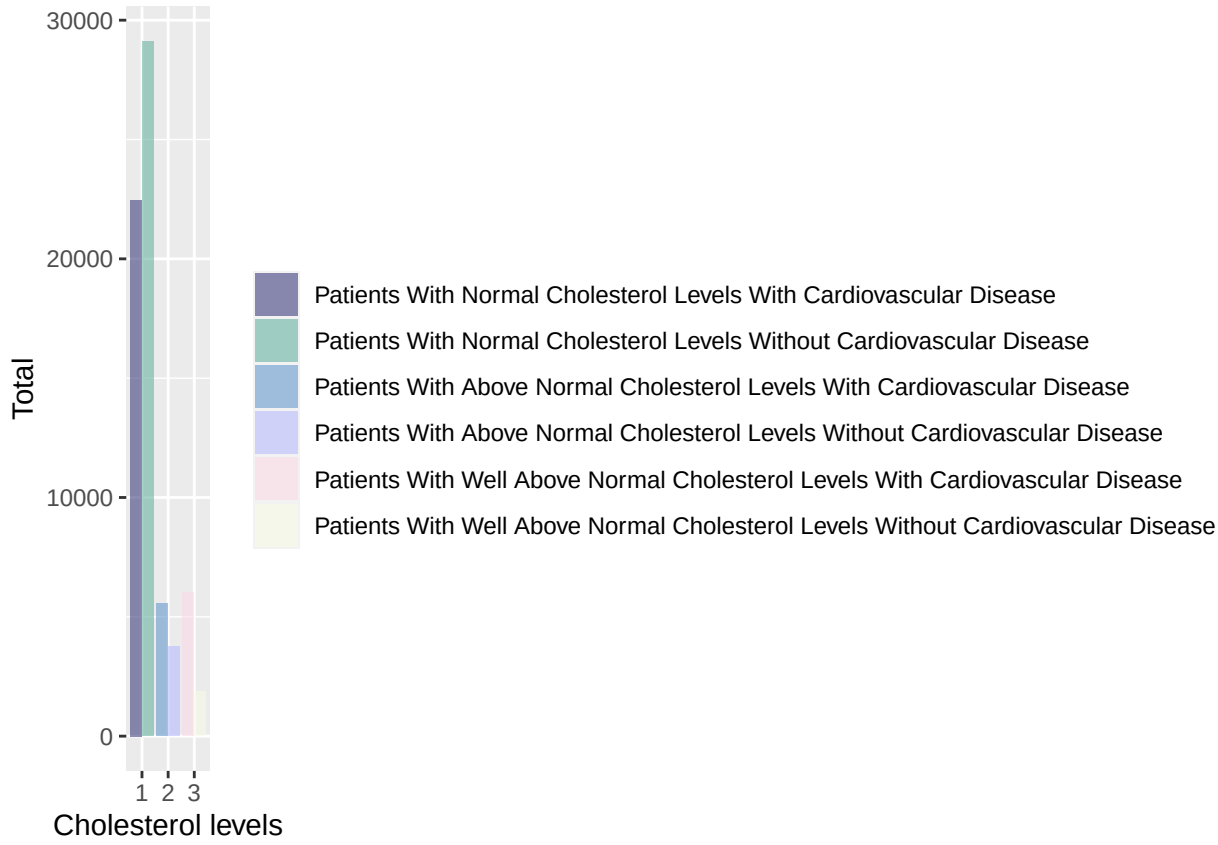

```
# Plotting Systolic Blood Pressure
ggplot(cardio_ap_high, aes(x = ap_high, fill = disease), alpha = .7) +
  geom_bar(width = 3) +
  labs(x = 'Systolic Blood Pressure (mm Hg)', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69b3a2'), name = '')
```



Cholesterol We begin seeing expected results upon plotting `cholesterol` against `cardio_disease`. Taking a look at the histogram, it's clear that patients with above normal and well above normal levels of cholesterol have much higher rates of cardiovascular disease than patients with normal levels of cholesterol. In fact, we can see that out of all the patients with well above normal cholesterol levels, there are over three times as many patients with cardiovascular disease than there are patients without.

```
# Creating data frame that contains 'cholesterol' and 'cardio_disease' if 'cardio_disease' is 0 and 'ch
cardio_disease_cholesterol_0_1 <- select(
  cardio[cardio$cardio_disease == 0 & cardio$cholesterol == 1, ],
  c('cholesterol', 'cardio_disease')
)

cardio_disease_cholesterol_0_1$disease <- 'Patients With Normal Cholesterol Levels Without Cardiovascul


# Creating data frame that contains 'cholesterol' and 'cardio_disease' if 'cardio_disease' is 0 and 'ch
cardio_disease_cholesterol_0_2 <- select(
  cardio[cardio$cardio_disease == 0 & cardio$cholesterol == 2, ],
  c('cholesterol', 'cardio_disease')
)

cardio_disease_cholesterol_0_2$disease <- 'Patients With Above Normal Cholesterol Levels Without Cardiov


# Creating data frame that contains 'cholesterol' and 'cardio_disease' if 'cardio_disease' is 0 and 'ch
```

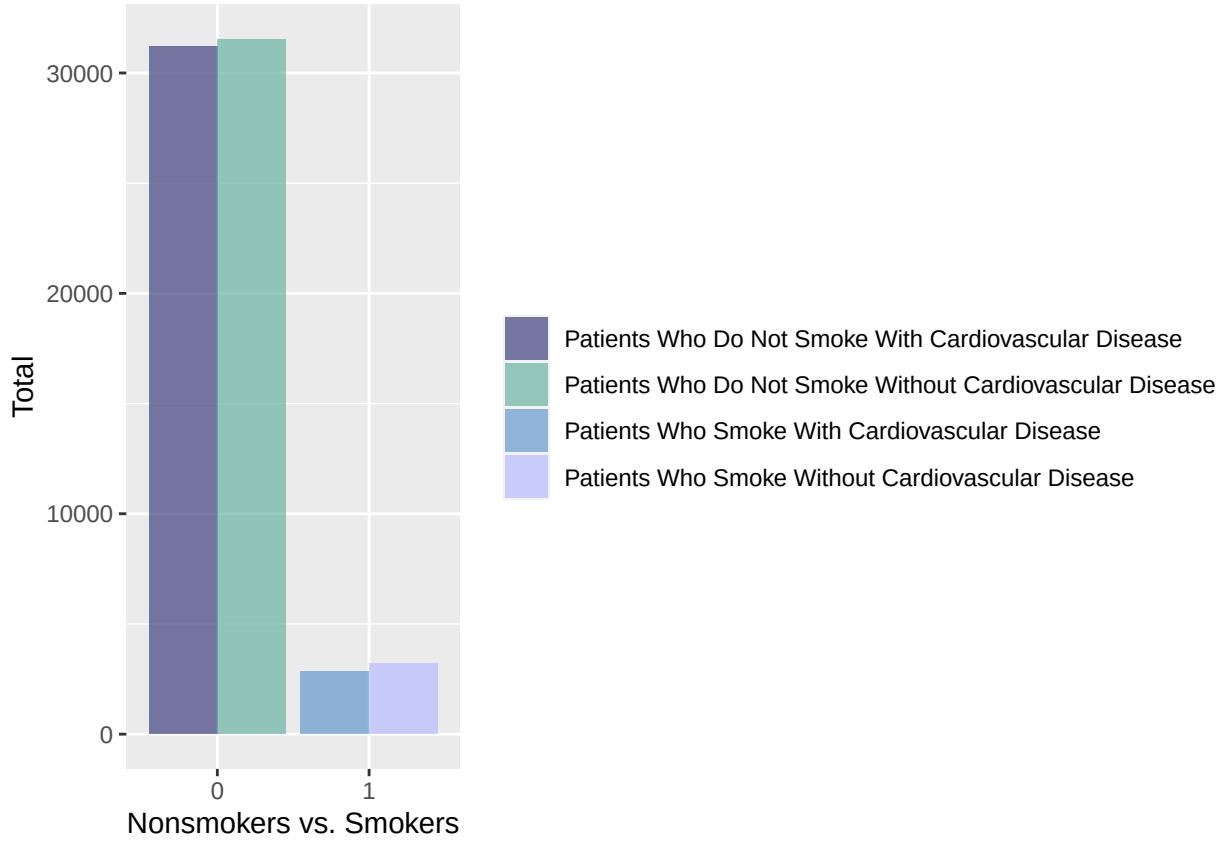
```r
cardio_disease_cholesterol_0_3 <- select(
  cardio[cardio$cardio_disease == 0 & cardio$cholesterol == 3, ],
  c('cholesterol', 'cardio_disease')
)

cardio_disease_cholesterol_0_3$disease <- 'Patients With Well Above Normal Cholesterol Levels Without Ca


# Creating data frame that contains 'cholesterol' and 'cardio_disease' if 'cardio_disease' is 1 and 'ch
cardio_disease_cholesterol_1_1 <- select(
  cardio[cardio$cardio_disease == 1 & cardio$cholesterol == 1, ],
  c('cholesterol', 'cardio_disease')
)

cardio_disease_cholesterol_1_1$disease <- 'Patients With Normal Cholesterol Levels With Cardiovascular D


# Creating data frame that contains 'cholesterol' and 'cardio_disease' if 'cardio_disease' is 1 and 'ch
cardio_disease_cholesterol_1_2 <- select(
  cardio[cardio$cardio_disease == 1 & cardio$cholesterol == 2, ],
  c('cholesterol', 'cardio_disease')
)

cardio_disease_cholesterol_1_2$disease <- 'Patients With Above Normal Cholesterol Levels With Cardiovasc


# Creating data frame that contains 'cholesterol' and 'cardio_disease' if 'cardio_disease' is 1 and 'ch
cardio_disease_cholesterol_1_3 <- select(
  cardio[cardio$cardio_disease == 1 & cardio$cholesterol == 3, ],
  c('cholesterol', 'cardio_disease')
)

cardio_disease_cholesterol_1_3$disease <- 'Patients With Well Above Normal Cholesterol Levels With Cardi


# Combining the new data frames
cardio_cholesterol <- rbind(
  cardio_disease_cholesterol_0_1,
  cardio_disease_cholesterol_0_2,
  cardio_disease_cholesterol_0_3,
  cardio_disease_cholesterol_1_1,
  cardio_disease_cholesterol_1_2,
  cardio_disease_cholesterol_1_3)

cardio_cholesterol$disease <- factor(cardio_cholesterol$disease, levels = c('Patients With Normal Choles


# Plotting the data
ggplot(cardio_cholesterol, aes(x = cholesterol)) +
  geom_bar(stat = 'count', aes(fill = disease), position = 'dodge', alpha = .6) +
  labs(x = 'Cholesterol levels', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69B3A2', '#6699CC', '#B8BCFF', '#FBD9E6', '#F6FBE4'), name =
```



Smoke Whether or not a patient smokes appears to have little effect on their rates of cardiovascular disease. Modern medical knowledge strongly asserts that there is a strong link between smoking and positive rates of cardiovascular disease, so the results of the histogram can be attributed to the fact that the `smoke` variable is binary.

```
# Creating data frame that contains 'smoke' and 'cardio_disease' if 'smoke' is 0 and 'cardio_disease' i
cardio_disease_smoke_0_0 <- select(
  cardio[cardio$smoke == 0 & cardio$cardio_disease == 0,],
  c('smoke' ,'cardio_disease')
)

cardio_disease_smoke_0_0$disease <- 'Patients Who Do Not Smoke Without Cardiovascular Disease'


# Creating data frame that contains 'smoke' and 'cardio_disease' if 'smoke' is 0 'cardio_disease' is 1
cardio_disease_smoke_0_1 <- select(
  cardio[cardio$smoke == 0 & cardio$cardio_disease == 1,],
  c('smoke' ,'cardio_disease')
)

cardio_disease_smoke_0_1$disease <- 'Patients Who Do Not Smoke With Cardiovascular Disease'


# Creating data frame that contains 'smoke' and 'cardio_disease' if 'smoke' is 1 'cardio_disease' is 0
cardio_disease_smoke_1_0 <- select(
```

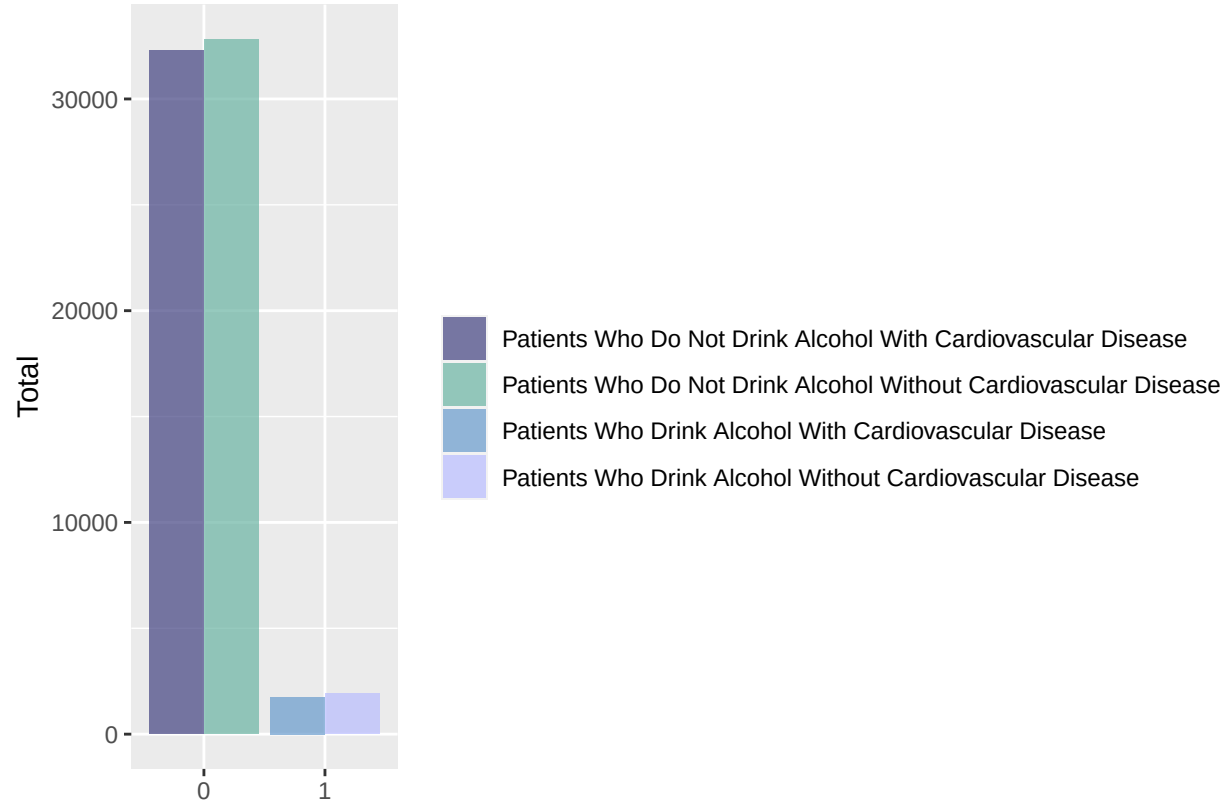
```
  cardio[cardio$smoke == 1 & cardio$cardio_disease == 0,],
  c('smoke' ,'cardio_disease')
)

cardio_disease_smoke_1_0$disease <- 'Patients Who Smoke Without Cardiovascular Disease'


# Creating data frame that contains 'smoke' and 'cardio_disease' if 'smoke' is 1 'cardio_disease' is 1
cardio_disease_smoke_1_1 <- select(
  cardio[cardio$smoke == 1 & cardio$cardio_disease == 1,],
  c('smoke' ,'cardio_disease')
)

cardio_disease_smoke_1_1$disease <- 'Patients Who Smoke With Cardiovascular Disease'


# Combining the new data frames
cardio_smoke <- rbind(
  cardio_disease_smoke_0_0,
  cardio_disease_smoke_0_1,
  cardio_disease_smoke_1_0,
  cardio_disease_smoke_1_1
)


# Plotting the data
ggplot(cardio_smoke, aes(x = smoke)) +
  geom_bar(stat = 'count', aes(fill = disease), position = 'dodge', alpha = .7) +
  labs(x = 'Nonsmokers vs. Smokers', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69B3A2', '#6699CC', '#B8BCFF'), name = '')
```



Alcohol Similar to the `smoke` variable, whether or not a patient drinks alcohol does not appear to have any effect on their cardiovascular health.

```
# Creating data frame that contains 'alcohol' and 'cardio_disease' if 'alcohol' is 0 and 'cardio_diseas
cardio_disease_alcohol_0_0 <- select(
  cardio[cardio$alcohol == 0 & cardio$cardio_disease == 0,],
  c('alcohol' ,'cardio_disease')
)

cardio_disease_alcohol_0_0$disease <- 'Patients Who Do Not Drink Alcohol Without Cardiovascular Disease


# Creating data frame that contains 'alcohol' and 'cardio_disease' if 'alcohol' is 0 'cardio_disease' i
cardio_disease_alcohol_0_1 <- select(
  cardio[cardio$alcohol == 0 & cardio$cardio_disease == 1,],
  c('alcohol' ,'cardio_disease')
)

cardio_disease_alcohol_0_1$disease <- 'Patients Who Do Not Drink Alcohol With Cardiovascular Disease'


# Creating data frame that contains 'alcohol' and 'cardio_disease' if 'alcohol' is 1 'cardio_disease' i
cardio_disease_alcohol_1_0 <- select(
  cardio[cardio$alcohol == 1 & cardio$cardio_disease == 0,],
  c('alcohol' ,'cardio_disease')
```

```
)

cardio_disease_alcohol_1_0$disease <- 'Patients Who Drink Alcohol Without Cardiovascular Disease'


# Creating data frame that contains 'alcohol' and 'cardio_disease' if 'alcohol' is 1 'cardio_disease' i
cardio_disease_alcohol_1_1 <- select(
  cardio[cardio$alcohol == 1 & cardio$cardio_disease == 1,],
  c('alcohol' ,'cardio_disease')
)

cardio_disease_alcohol_1_1$disease <- 'Patients Who Drink Alcohol With Cardiovascular Disease'


# Combining the new data frames
cardio_alcohol <- rbind(
  cardio_disease_alcohol_0_0,
  cardio_disease_alcohol_0_1,
  cardio_disease_alcohol_1_0,
  cardio_disease_alcohol_1_1
)


# Plotting the data
ggplot(cardio_alcohol, aes(x = alcohol)) +
  geom_bar(stat = 'count', aes(fill = disease), position = 'dodge', alpha = .7) +
  labs(x = 'Patients Who Drink vs. Patients Who Abstain', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69B3A2', '#6699CC', '#B8BCFF'), name = '')
```

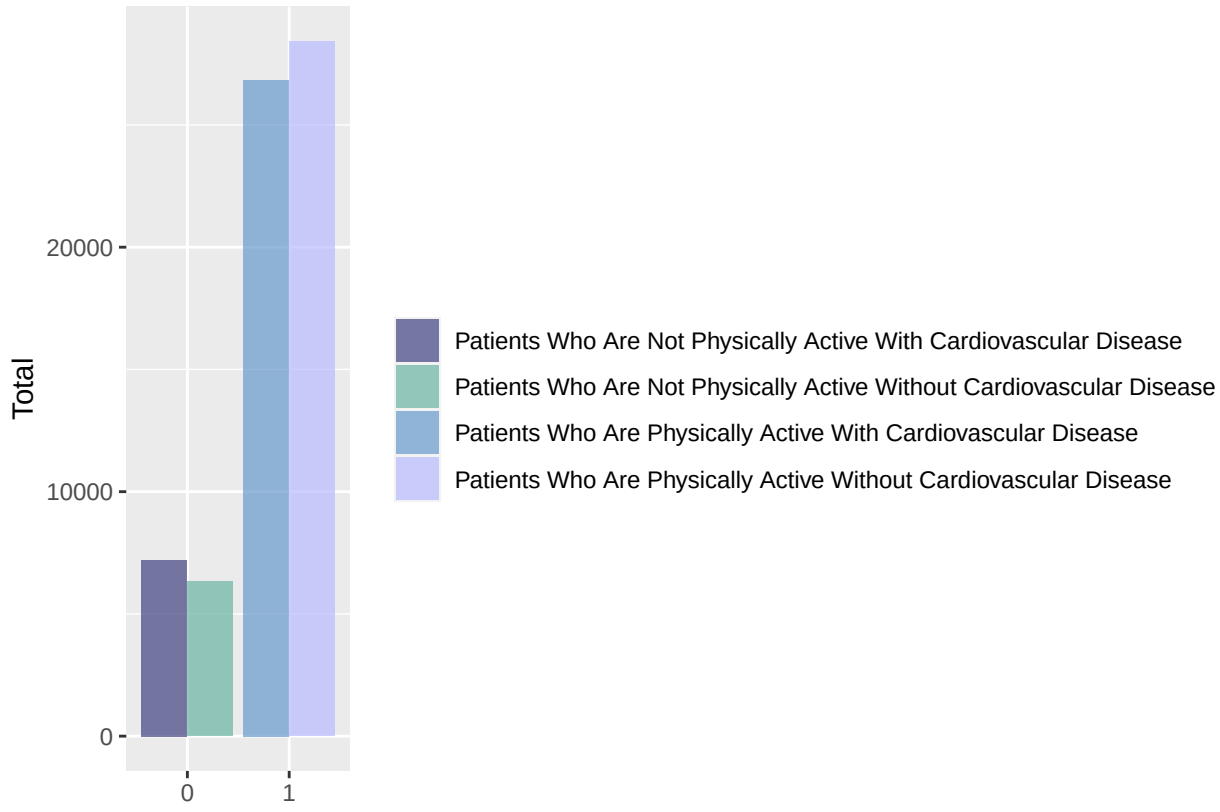


ıtients Who Drink vs. Patients Who Abstain

Physical Activity Again, a lack of physical activity shows no link with positive rates of cardiovascular disease.

```
# Creating data frame that contains 'physical_activity' and 'cardio_disease' if 'physical_activity' is
cardio_disease_p_a_0_0 <- select(
  cardio[cardio$physical_activity == 0 & cardio$cardio_disease == 0,],
  c('physical_activity' ,'cardio_disease')
)

cardio_disease_p_a_0_0$disease <- 'Patients Who Are Not Physically Active Without Cardiovascular Diseas


# Creating data frame that contains 'physical_activity' and 'cardio_disease' if 'physical_activity' is
cardio_disease_p_a_0_1 <- select(
  cardio[cardio$physical_activity == 0 & cardio$cardio_disease == 1,],
  c('physical_activity' ,'cardio_disease')
)

cardio_disease_p_a_0_1$disease <- 'Patients Who Are Not Physically Active With Cardiovascular Disease'


# Creating data frame that contains 'physical_activity' and 'cardio_disease' if 'physical_activity' is
cardio_disease_p_a_1_0 <- select(
  cardio[cardio$physical_activity == 1 & cardio$cardio_disease == 0,],
  c('physical_activity' ,'cardio_disease')
```

```
)

cardio_disease_p_a_1_0$disease <- 'Patients Who Are Physically Active Without Cardiovascular Disease'


# Creating data frame that contains 'physical_activity' and 'cardio_disease' if 'physical_activity' is
cardio_disease_p_a_1_1 <- select(
  cardio[cardio$physical_activity == 1 & cardio$cardio_disease == 1,],
  c('physical_activity' ,'cardio_disease')
)

cardio_disease_p_a_1_1$disease <- 'Patients Who Are Physically Active With Cardiovascular Disease'


# Combining the new data frames
cardio_p_a <- rbind(
  cardio_disease_p_a_0_0,
  cardio_disease_p_a_0_1,
  cardio_disease_p_a_1_0,
  cardio_disease_p_a_1_1
)


# Plotting the data
ggplot(cardio_p_a, aes(x = physical_activity)) +
  geom_bar(stat = 'count', aes(fill = disease), position = 'dodge', alpha = .7) +
  labs(x = '', y = 'Total') +
  scale_fill_manual(values = c('#404080', '#69B3A2', '#6699CC', '#B8BCFF'), name = '')
```



Results of our Exploratory Data Analysis

Based off the correlation plot, histograms, and barplots, there appear to be some variables that demonstrate a clear relationship with `cardio_disease`. However, the `smoke`, `alcohol`, and `physical_activity` variables show almost no connection to positive rates of cardiovascular disease. As explained before, this is likely due to the fact that these variables are all binary. This means that patients who drink, smoke, and are physically sedentary to extreme levels are considered the same as patients who drink, smoke, and exercise rarely. Due to the fact that these variables are ill-defined, we will remove them from our models to ensure they do not interfere with our predictions.

Preparing our Models

Before we begin running our models, there are a few steps we must take. First, we must split our data in training and testing sets. We then utilize a procedure known as K-Fold Cross-Validation to ensure the models perform well on the data they've been trained on as well as data they haven't been exposed to. We will then create our recipe and begin training our models on the training data.

Splitting our Data

We begin by first splitting the cardiovascular disease data set into training and testing sets. Because the data set we're working with has a large number of observations, we can afford to split it into 90% training and 10% testing without worrying that there won't be enough observations in the testing set. We will also stratify the split on our response variable, `cardio_disease`.

```
cardio_split <- cardio_one_hot %>%
  initial_split(strata = cardio_disease_1, prop = .90)

cardio_train <- training(cardio_split) %>% as_tibble()
cardio_train[ ,c('smoke_1', 'alcohol_1', 'physical_activity_1')] <- list(NULL)

cardio_test <- testing(cardio_split) %>% as_tibble()
cardio_test[ ,c('smoke_1', 'alcohol_1', 'physical_activity_1')] <- list(NULL)
```

Creating the Recipe

Next up, we create a recipe that includes all the transformations we'll be making to our data. We turn all our one-hot encoded variables into dummy variables. We also exclude `smoke`, `alcohol`, and `physical_activity` from our recipe. Finally, we center and scale our data.

```
# Creating the recipe
cardio_recipe <- recipe(cardio_disease_1 ~ age + gender_2 + height + weight + ap_high + ap_low + choles
  # Dummy coding our one-hot encoded variables
  step_dummy(gender_2) %>%
  step_dummy(cholesterol_2) %>%
  step_dummy(cholesterol_3) %>%
  step_dummy(glucose_2) %>%
  step_dummy(glucose_3) %>%
  # Centering and scaling the data
  step_normalize(all_predictors())
```

10-Fold Cross Validation

In order to avoid overfitting and create models that are able to generalize, we utilize a concept known as K-Fold Cross Validation. K-Fold Cross Validation is the process of splitting our training data set into K groups (K = 10 in our case), training the model on K - 1 of these groups and testing on the remaining group, then repeating the above steps until each group was used to test the model once. We will also stratify the folds on our response variable, `cardio_disease_1`.

```
# Generating 10 folds and stratifying on response variable
cardio_folds <- vfold_cv(cardio_train, v = 10, strata = cardio_disease_1)
```

Loading in our Models

We will now begin generating and running our models. I chose a total of seven models: LDA, QDA, Niave Bayes, Logistic Regression, Random Forest, Boosted Forest, and Support Vector Machines. We will be using ROC AUC as our metric. ROC AUC tells us how good a model is at distinguishing between classes, so choosing it as our metric comes naturally considering the fact that our outcome variable is binary.

The Linear Discriminant Analysis Model

Despite the relative ease of setting up this model and the quick run time, the LDA model gives us a decent ROC AUC score of .789.

```r
# Setting up the model, mode, and engine
cardio_lda <- discrim_linear() %>%
  set_mode('classification') %>%
  set_engine('MASS')

# Setting up the workflow
cardio_lda_workflow <- workflow() %>%
  add_model(cardio_lda) %>%
  add_recipe(cardio_recipe)

# Fitting the model on the training data
cardio_lda_fit <- fit(cardio_lda_workflow, cardio_train)

# Determining the ROC AUC of our model
results_lda <- augment(cardio_lda_fit, new_data = cardio_train) %>%
  roc_auc(truth = cardio_disease_1, estimate = .pred_0)

results_lda <- cbind('Linear Discriminant Analysis', results_lda)

colnames(results_lda)[1] <- 'model'

results_lda
```

```
##                           model .metric .estimator .estimate
## 1 Linear Discriminant Analysis roc_auc     binary 0.7886048
```

The Quadratic Discriminant Analysis Model

The QDA model unfortunately performs a bit more poorly than the LDA model, coming in at an ROC AUC score of .756. However, this is still a fair score.

```r
# Setting up the model, mode, and engine
cardio_qda <- discrim_quad() %>%
  set_mode('classification') %>%
  set_engine('MASS')

# Setting up the workflow
cardio_qda_workflow <- workflow() %>%
  add_model(cardio_qda) %>%
  add_recipe(cardio_recipe)

# Fitting the model on the training data
cardio_qda_fit <- fit(cardio_qda_workflow, cardio_train)

# Determining the ROC AUC of our model
results_qda <- augment(cardio_qda_fit, new_data = cardio_train) %>%
  roc_auc(truth = cardio_disease_1, estimate = .pred_0)

results_qda <- cbind('Quadratic Discriminant Analysis', results_qda)

colnames(results_qda)[1] <- 'model'

results_qda
```

```
##                              model .metric .estimator .estimate
## 1 Quadratic Discriminant Analysis roc_auc     binary  0.756453
```

The Naive Bayes Model

The Naive Bayes model performs similarly to the LDA model, generating an ROC AUC score of .785 despite running faster than the QDA model.

```
# Setting up the model, mode, and engine
cardio_nb <- naive_Bayes() %>%
  set_mode('classification') %>%
  set_engine('klaR')

# Setting up the workflow
cardio_nb_workflow <- workflow() %>%
  add_model(cardio_nb) %>%
  add_recipe(cardio_recipe)

# Fitting the model on the training data
cardio_nb_fit <- fit(cardio_nb_workflow, cardio_train)

# Determining the ROC AUC of our model
results_nb <- augment(cardio_nb_fit, new_data = cardio_train) %>%
  roc_auc(truth = cardio_disease_1, estimate = .pred_0)

results_nb <- cbind('Naive Bayes', results_nb)

colnames(results_nb)[1] <- 'model'

results_nb
```

```
##         model .metric .estimator .estimate
## 1 Naive Bayes roc_auc     binary 0.7848162
```

The Logistic Regression Model

The Logistic Regression model performs the best so far, although it only beats out the LDA model by a few decimal points. This model achieves an ROC AUC score of .789.

```
# Setting up the model, mode, and engine
cardio_log <- logistic_reg() %>%
  set_engine('glm') %>%
  set_mode('classification')

# Setting up the workflow
cardio_log_workflow <- workflow() %>%
  add_model(cardio_log) %>%
  add_recipe(cardio_recipe)

# Fitting the model on the training data
cardio_log_fit <- fit(cardio_log_workflow, cardio_train)
```

```r
# Determining the ROC AUC of our model
results_lr <- augment(cardio_log_fit, new_data = cardio_train) %>%
  roc_auc(cardio_disease_1, estimate = .pred_0)

results_lr <- cbind('Logistic Regression', results_lr)

colnames(results_lr)[1] <- 'model'

results_lr
```

```
##                 model .metric .estimator .estimate
## 1 Logistic Regression roc_auc     binary 0.7890865
```

Loading in our Tuned Models

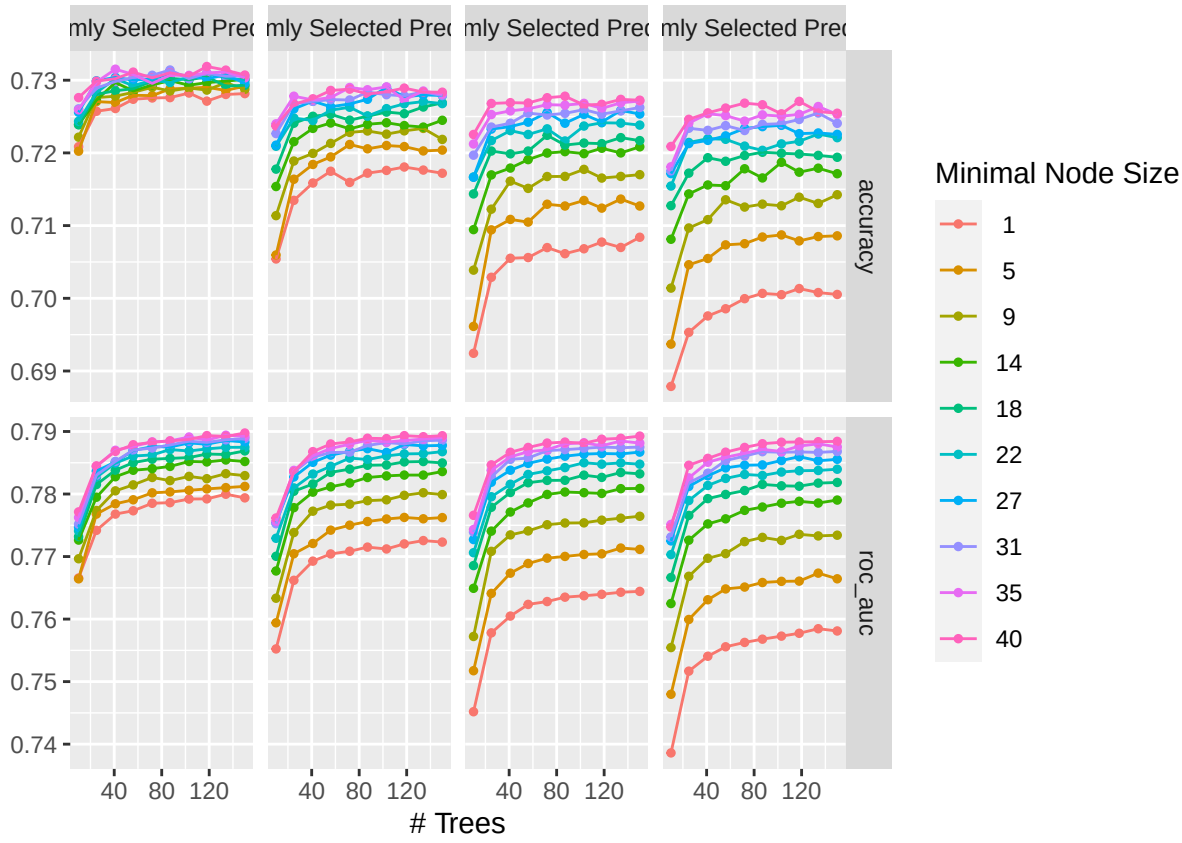
The tuned models we generated took many days to run. As a result, I chose to save these models in the form of an RDA file so we'd be easily able to access them without having to wait for too long.

```r
load('/Users/andrewchoi/Downloads/PSTAT131_FINAL_PROJECT/PSTAT131_FINAL_PROJECT/Random_Forest_Model.rda
load('/Users/andrewchoi/Downloads/PSTAT131_FINAL_PROJECT/PSTAT131_FINAL_PROJECT/Boosted_Model.rda')
load('/Users/andrewchoi/Downloads/PSTAT131_FINAL_PROJECT/PSTAT131_FINAL_PROJECT/Support_Vector_Machine_
```

The Random Forest Model

For the Random Forest Model, I decided to tune the number of variables randomly sampled at each split, the minimum number of data points in a node needed for the node to be split, and the number of trees. Based on the autoplot, it appears that having larger values of trees and minimal node sizes, while having smaller values of randomly selected predictors gave us the best ROC AUC score.

```r
autoplot(cardio_forest_tune_res)
```



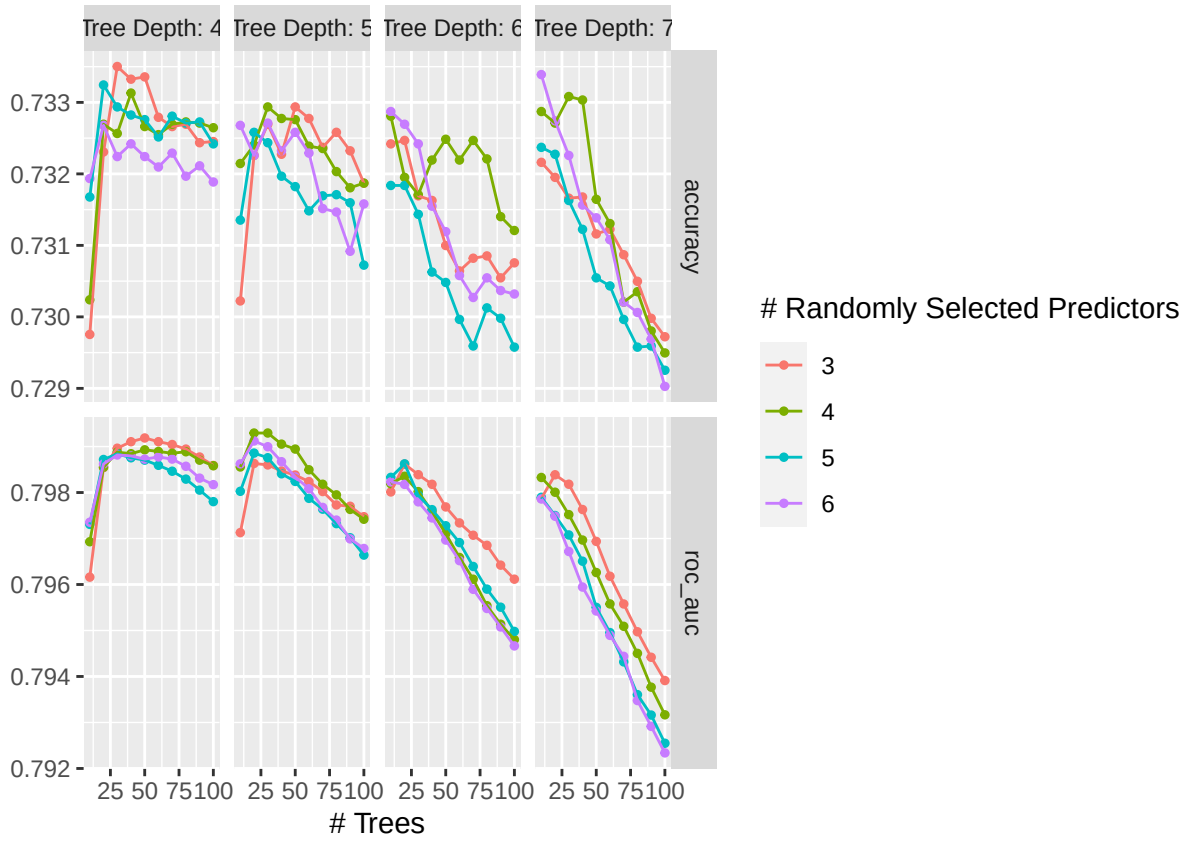

```
results_forest <- augment(cardio_forest_final_fit, new_data = cardio_train) %>%
  roc_auc(cardio_disease_1, estimate = .pred_0)

results_forest <- cbind('Random Forest', results_forest)

colnames(results_forest)[1] <- 'model'

results_forest
```

```
##           model .metric .estimator .estimate
## 1 Random Forest roc_auc     binary 0.8455698
```

The Boosted Trees Model

For the Boosted Trees Model, I tuned the number of variables randomly sampled at each split, the maximum depth of a tree, and the number of trees. Looking at the autoplot, it's clear that having smaller values for all tuned hyperparameters results in a better ROC AUC score.

```
autoplot(cardio_boost_tune_res)
```



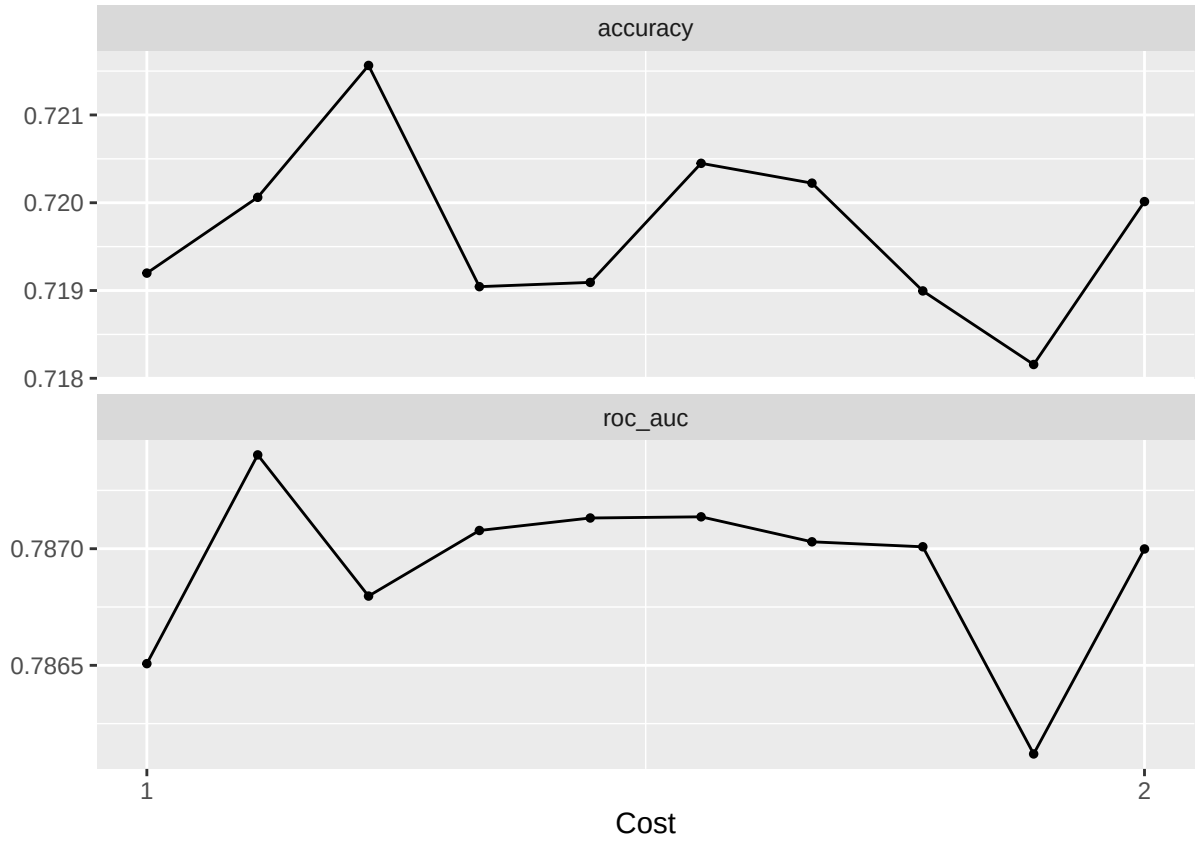

```
results_boost <- augment(cardio_boost_final_fit, new_data = cardio_train) %>%
  roc_auc(cardio_disease_1, estimate = .pred_0)

results_boost <- cbind('Boosted Trees', results_boost)

colnames(results_boost)[1] <- 'model'

results_boost
```

```
##           model .metric .estimator .estimate
## 1 Boosted Trees roc_auc     binary 0.8082098
```

The Linear Support Vector Machine Model

Because the Support Vector Machine model took an extremely long amount of time to run, I decided to only tune the cost hyperparameter. Even then, this model took days at a time to finish loading, so I restricted the cost hyperparameter between 1 and 2. Despite the fact that the difference between 1 and 2 is minor, we can clearly see that this model scores a better ROC AUC when cost is closer to 1.

```
autoplot(cardio_svm_tune_res)
```




```
results_svm <- augment(cardio_svm_final_fit, new_data = cardio_train) %>%
  roc_auc(cardio_disease_1, estimate = .pred_0)

results_svm <- cbind('Support Vector Machine', results_svm)

colnames(results_svm)[1] <- 'model'

results_svm
```

```
##                    model .metric .estimator .estimate
## 1 Support Vector Machine roc_auc     binary 0.6986765
```

Model Performance

Now that we've run all our models and gauged their predictive abilities, we can compare them against each other to see which one works best

```
rbind(results_lda, results_qda, results_nb, results_lr, results_forest, results_boost, results_svm)
```

```
##                             model .metric .estimator .estimate
## 1    Linear Discriminant Analysis roc_auc     binary 0.7886048
## 2 Quadratic Discriminant Analysis roc_auc     binary 0.7564530
## 3                     Naive Bayes roc_auc     binary 0.7848162
## 4             Logistic Regression roc_auc     binary 0.7890865
```

```
## 5                   Random Forest roc_auc     binary 0.8455698
## 6                   Boosted Trees roc_auc     binary 0.8082098
## 7          Support Vector Machine roc_auc     binary 0.6986765
```

The Random Forest Model beats out all of the other models by a pretty far margin in terms of ROC AUC. The second best performing model has an ROC AUC score that's .04 points behind the Random Forest Model. I was extremely surprised by the fact that the Support Vector Machine had an ROC AUC of .699, but this is likely due to the fact that the tuned cost hyperparameter was only allowed to range between 1 and 2. Since the Random Forest Model performed the best, we can fit it to our testing data and see how well it performs on data it hasn't yet seen.

Fitting the Random Forest Model to the Testing Data

We can fit our best performing model to the testing data set and determine the ROC AUC score of the model on data it hasn't been trained on.

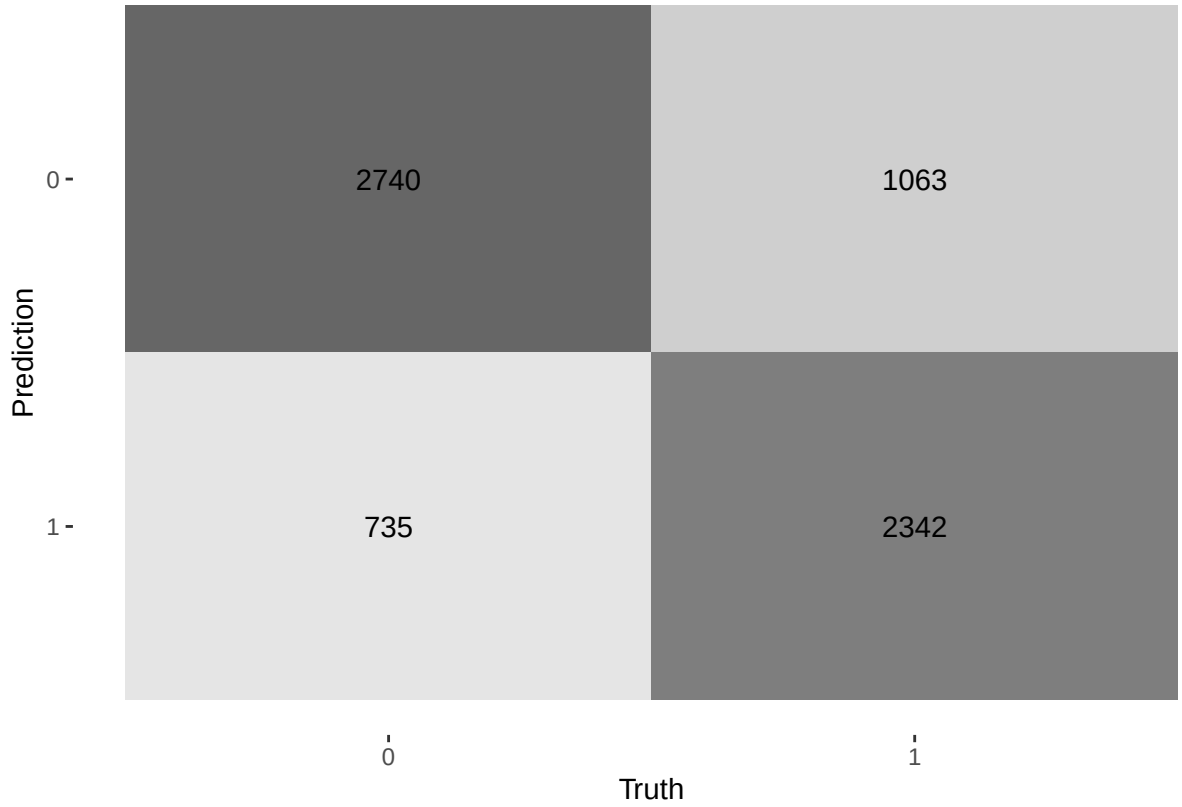
```
augment(cardio_forest_final_fit, new_data = cardio_test) %>%
  roc_auc(cardio_disease_1, estimate = .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.792
```

The fact that this model had an ROC AUC score of .792 on the testing data tells me that the Random Forest Model may have been over fit a tad bit. Of course, I expected the ROC AUC score to drop a bit, considering that this model had never encountered the testing data before, but I didn't expect the score to drop by over .05 points. Still an ROC AUC score between .7 - .8 is considered fair overall, so I'm glad my implementation of the Random Forest model for cardiovascular disease detection is still useful!

Next up, we can test how well our Random Forest model did on our testing data by creating a heat map of the true positive, false positive, true negative, and false negative values.

```
augment(cardio_forest_final_fit, new_data = cardio_test) %>%
  conf_mat(truth = cardio_disease_1, estimate = .pred_class) %>%
  autoplot(type = 'heatmap')
```



Based off these results, it appears as though the Random Forest model is better at correctly predicting that a patient does not have some type of cardiovascular disease, than it is at correctly predicting that a patient does have some type of cardiovascular disease.

Conclusion

It's clear that our models did not have the best ROC AUC scores they could have had. Although they received 'fair' ratings, it's far from the 'excellent' that every model strives for. In order to improve the performance of our models, allowing more of the hyperparameters associated with each model to run with larger ranges could help boost ROC AUC scores. In addition, having predictor variables that are better defined could benefit the models performances greatly. Overall, I'm surprised that the performance of these models did not meet expectations, but I also understand they have room to grow.