Using a Machine Learning Ensemble to Make Heart Disease Prediction in R

Jarred Priester

1/14/2022

Table of Content

1. Overview

Before we jump into any data let's first briefly look at what is Heart Disease and how can data science be used as a tool to help with this disease.

Below is from John Hopkins University:

Heart disease accounts for 1 in every 4 deaths in the U.S., according to the Centers for Disease Control and Prevention (CDC). It is the leading cause of death for both American women and men.

Cardiovascular disease includes stroke, coronary heart disease, high blood pressure and rheumatic heart disease. The most common type of heart condition is coronary heart disease, according to the CDC.

Heart conditions develop when plaque builds up in the arteries, causing a narrowing of the interior of the arteries. As the arteries narrow, the flow of blood is decreased or blocked. In some cases, plaques can rupture and cause blood clots to form.

Understanding the anatomy of the heart, cardiac diagnostic tests, common symptoms of heart conditions and strategies of prevention will help safeguard cardiovascular health.

https://www.hopkinsmedicine.org/health/wellness-and-prevention/heart-conditions#:~:text=Heart%20disease%20accounts%

As you can see heart disease is a serious issue being the leading cause of death for both American women and men. As was mentioned above, understanding the common symptoms of heart disease can help, that is where machine learning comes in.

Machine learning can help find patterns in data and in this case help predict if someone has heart disease.

In this project we will use data to train models using machine learning in order to predict if someone has heart disease or not.

1.1 Description of dataset

We will be analyzing the heart disease data set from University of California Irvine machine learning repository. This data set consist fo 14 different features and 303 observations. The description of the features from the website is the following:

- ***age***: age in years
- ***sex***: sex (1 = male; 0 = female)
- ***cp***: chest pain type
  - Value 1: typical angina
  - Value 2: atypical angina
  - Value 3: non-anginal pain

- ***trestbps***: resting blood pressure (in mm Hg on admission to the hospital)
- ***chol***: serum cholestoral in mg/dl
- ***fbs***: (fasting blood sugar $>$ 120 mg/dl) (1 = true; 0 = false)
- ***restecg***: resting electrocardiographic results
  - Value 0: normal
  - Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of $>$ 0.05 mV)
  - Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
- ***thalach***: maximum heart rate achieved
- ***exang***: exercise induced angina (1 = yes; 0 = no)
- ***oldpeak*** = ST depression induced by exercise relative to rest
- ***slope***: the slope of the peak exercise ST segment
  - Value 1: upsloping
  - Value 2: flat
  - Value 3: downsloping
- ***ca***: number of major vessels (0-3) colored by flourosopy
- ***thal***: 3 = normal; 6 = fixed defect; 7 = reversable defect
- ***num***: diagnosis of heart disease (angiographic disease status)
  - Value 0: $<$ 50% diameter narrowing
  - Value 1: $>$ 50% diameter narrowing

The num feature is the feature we will be trying to predict for this project.

Link to the UCI heart disease data: https://archive.ics.uci.edu/ml/datasets/heart+disease

1.2 Goal of project

The goal for this project is to create a model that can predict the patient's heart disease status with a F1 score of 85% or higher. The other goals will be to explore the data we have been given and find key insights into heart disease that could be helpful for the medical community going forward.

1.3 Step to acheive the goal

To achieve this goal we will be creating 10 different models and an ensemble and comparing their results. Because the nature of the problem is to determine if a patient is negative or positive, i.e 0 or 1, this is a binary classification problem and we will pick 10 algorithms that work well with binary classification. The algorithms we will be using are the following:

- Logistic Regression
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Loess Model
- K-Nearest Neighbors
- Random Forest
- Tree Models from Genetic Algorithms
- Least Squares Support Vector Machine
- Bayesian Generalized Linear Model
- Neural Network

We will split the data into training and test data. We will be using the K-fold cross variation technique in order to test and validate our models so that we make sure to not over train the models. We will then train the models and apply them to the test set giving us our F1 score.

2. Data Cleaning

2.1 Downloading the data

```r
#importing libraries
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(caret)) install.packages("dplyr")
if(!require(dplyr)) install.packages("dplyr")
if(!require(matrixStats)) install.packages("matrixStats")
if(!require(gam)) install.packages("gam")
if(!require(evtree)) install.packages("evtree")
if(!require(knitr)) install.packages("knitr")

library(tidyverse)
library(caret)
library(dplyr)
library(matrixStats)
library(gam)
library(evtree)
library(knitr)

#importing the University of California, Irvine Heart Disease Data set
heart <- read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cl
names(heart) <- c( "age", "sex", "cp", "trestbps", "chol","fbs", "restecg",
                        "thalach","exang", "oldpeak","slope", "ca", "thal", "num")
```

2.2 Data Cleaning

First we are going to take a look at the data set and get an idea of what we need to clean and how the data set is structured

```r
head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3  0    6   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2  3    3   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2  2    7   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3  0    3   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1  0    3   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1  0    3   0
```

```r
dim(heart)
```

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

```r
str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : num  6 3 7 3 3 3 3 3 7 7 ...
##  $ num     : int  0 2 1 0 0 0 3 0 2 1 ...
```

```
summary(heart)
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :54.44   Mean   :0.6799   Mean   :3.158   Mean   :131.7
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
##
##       chol            fbs            restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
##  Median :241.0   Median :0.0000   Median :1.0000   Median :153.0
##  Mean   :246.7   Mean   :0.1485   Mean   :0.9901   Mean   :149.6
##  3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##
##      exang           oldpeak         slope             ca
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.80   Median :2.000   Median :0.0000
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601   Mean   :0.6722
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000   Max.   :3.0000
##                                                  NA's   :4
##       thal            num
##  Min.   :3.000   Min.   :0.0000
##  1st Qu.:3.000   1st Qu.:0.0000
##  Median :3.000   Median :0.0000
##  Mean   :4.734   Mean   :0.9373
##  3rd Qu.:7.000   3rd Qu.:2.0000
##  Max.   :7.000   Max.   :4.0000
##  NA's   :2
```

The feature num is the angiographic disease status. 0 represents no heart disease while 1-4 represents the extent of heart disease in the patient. So for num, we are going to convert anything greater than 0 to equal 1, leaving us with 0 (no heart disease) and 1 (heart disease)

```
heart$num[heart$num > 0] <- 1
```

check to make sure 1-4 were converted to 1

```
summary(heart$num)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4587  1.0000  1.0000
```

checking to see if the changes were made correctly

```
heart$sex
```

```
##   [1] 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1
##  [38] 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1
##  [75] 1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0
## [112] 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0
## [186] 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 0 1 0 0
## [223] 0 1 0 0 1 0 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1
## [260] 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 0 1
## [297] 1 0 1 1 1 0 1
```

check for any NAs

```
sum(is.na(heart) == TRUE)
```

```
## [1] 6
```

we have 6 NAs which is not too many so we will delete those rows

```
heart <- na.omit(heart)
dim(heart)
```

```
## [1] 297  14
```

3. Exploritory Data Exploration

3.1 Data exploration

Let's look at a summary of the data with just the observations without heart disease

```
heart %>% filter(num == 0) %>% summary()
```

```
##       age             sex               cp           trestbps
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
##  1st Qu.:44.75   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:120.0
##  Median :52.00   Median :1.0000   Median :3.000   Median :130.0
##  Mean   :52.64   Mean   :0.5563   Mean   :2.794   Mean   :129.2
##  3rd Qu.:59.00   3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.:140.0
##  Max.   :76.00   Max.   :1.0000   Max.   :4.000   Max.   :180.0
##       chol            fbs            restecg          thalach
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 96.0
##  1st Qu.:208.8   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:149.0
##  Median :235.5   Median :0.0000   Median :0.0000   Median :161.0
##  Mean   :243.5   Mean   :0.1437   Mean   :0.8438   Mean   :158.6
##  3rd Qu.:268.2   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:172.0
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0
##      exang           oldpeak           slope             ca
##  Min.   :0.0000   Min.   :0.0000   Min.   :1.000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.000
##  Median :0.0000   Median :0.2000   Median :1.000   Median :0.000
##  Mean   :0.1437   Mean   :0.5988   Mean   :1.413   Mean   :0.275
##  3rd Qu.:0.0000   3rd Qu.:1.1000   3rd Qu.:2.000   3rd Qu.:0.000
##  Max.   :1.0000   Max.   :4.2000   Max.   :3.000   Max.   :3.000
##       thal            num
##  Min.   :3.000   Min.   :0
##  1st Qu.:3.000   1st Qu.:0
##  Median :3.000   Median :0
##  Mean   :3.788   Mean   :0
##  3rd Qu.:3.000   3rd Qu.:0
##  Max.   :7.000   Max.   :0
```

Now let's take a look at a summary of the data with just the observations with heart disease

```
heart %>% filter(num == 1) %>% summary()
```

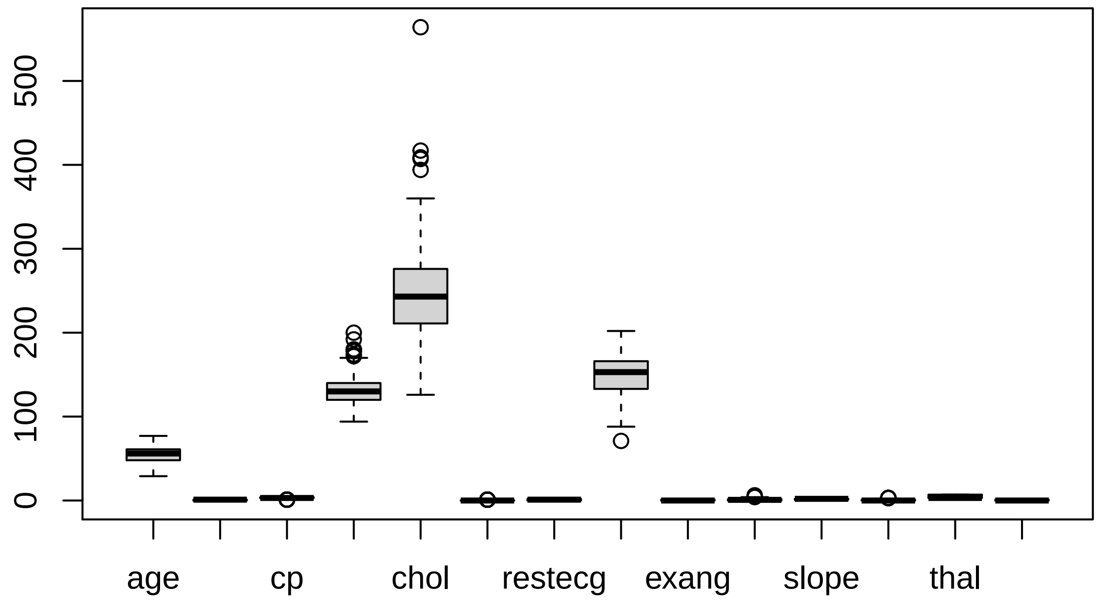
```
##       age             sex              cp           trestbps
##  Min.   :35.00   Min.   :0.0000   Min.   :1.000   Min.   :100.0
##  1st Qu.:53.00   1st Qu.:1.0000   1st Qu.:4.000   1st Qu.:120.0
##  Median :58.00   Median :1.0000   Median :4.000   Median :130.0
##  Mean   :56.76   Mean   :0.8175   Mean   :3.584   Mean   :134.6
##  3rd Qu.:62.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:145.0
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
##       chol            fbs           restecg         thalach
##  Min.   :131.0   Min.   :0.000   Min.   :0.000   Min.   : 71.0
##  1st Qu.:218.0   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:125.0
##  Median :253.0   Median :0.000   Median :2.000   Median :142.0
##  Mean   :251.9   Mean   :0.146   Mean   :1.175   Mean   :139.1
##  3rd Qu.:284.0   3rd Qu.:0.000   3rd Qu.:2.000   3rd Qu.:157.0
##  Max.   :409.0   Max.   :1.000   Max.   :2.000   Max.   :195.0
##      exang           oldpeak          slope             ca
##  Min.   :0.0000   Min.   :0.000   Min.   :1.000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.600   1st Qu.:1.000   1st Qu.:0.000
##  Median :1.0000   Median :1.400   Median :2.000   Median :1.000
##  Mean   :0.5401   Mean   :1.589   Mean   :1.825   Mean   :1.146
##  3rd Qu.:1.0000   3rd Qu.:2.500   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :1.0000   Max.   :6.200   Max.   :3.000   Max.   :3.000
##       thal            num
##  Min.   :3.000   Min.   :1
##  1st Qu.:3.000   1st Qu.:1
##  Median :7.000   Median :1
##  Mean   :5.832   Mean   :1
##  3rd Qu.:7.000   3rd Qu.:1
##  Max.   :7.000   Max.   :1
```

There looks to be some differences when you compare the observations for positive heart disease and negative heart disease. For examples, the sex for the positive heart disease leaned more towards male. The age on average was younger for negative heart disease observations. The average maximum heart rate achieved (thalach) was on average higher for the negative heart disease observations. Next let's continue analyzing the data with visualization.

3.2 Data Visualization

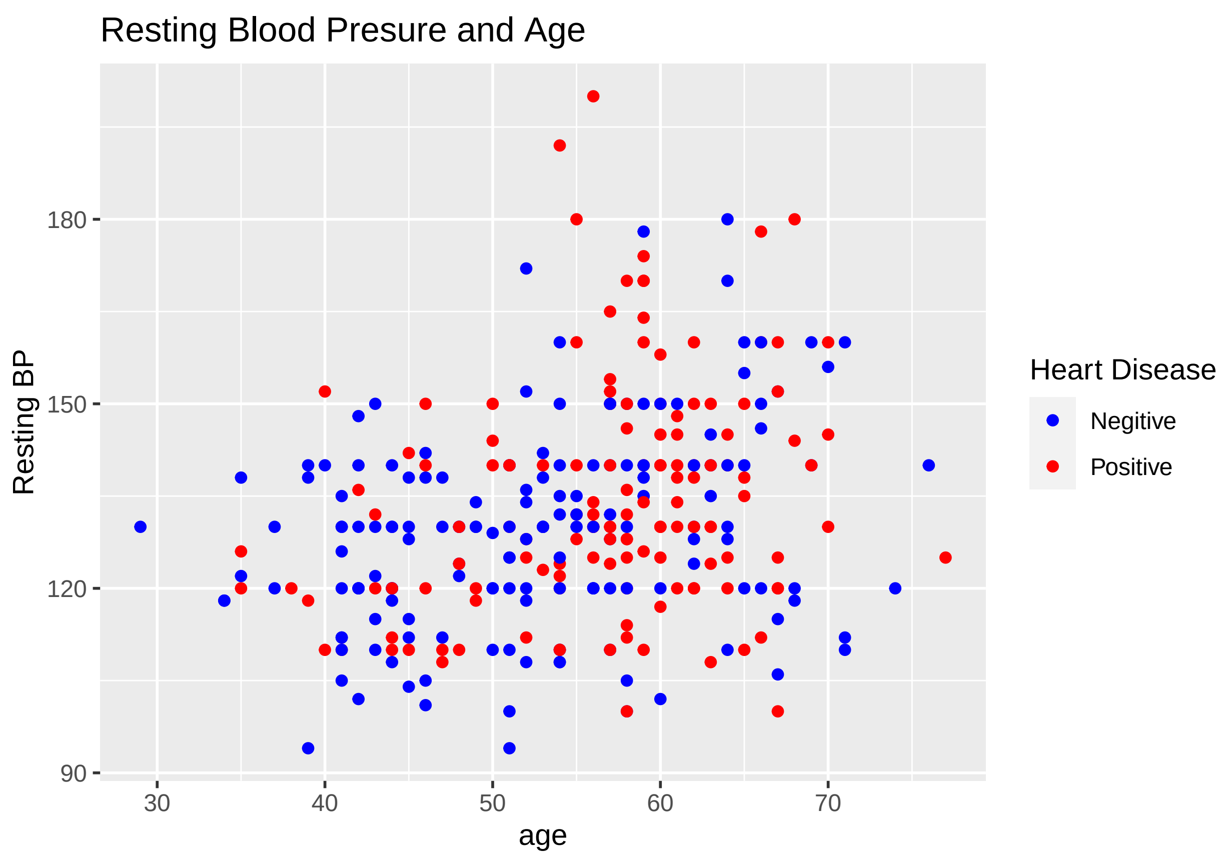
box plot of the data set

```
boxplot(heart)
```



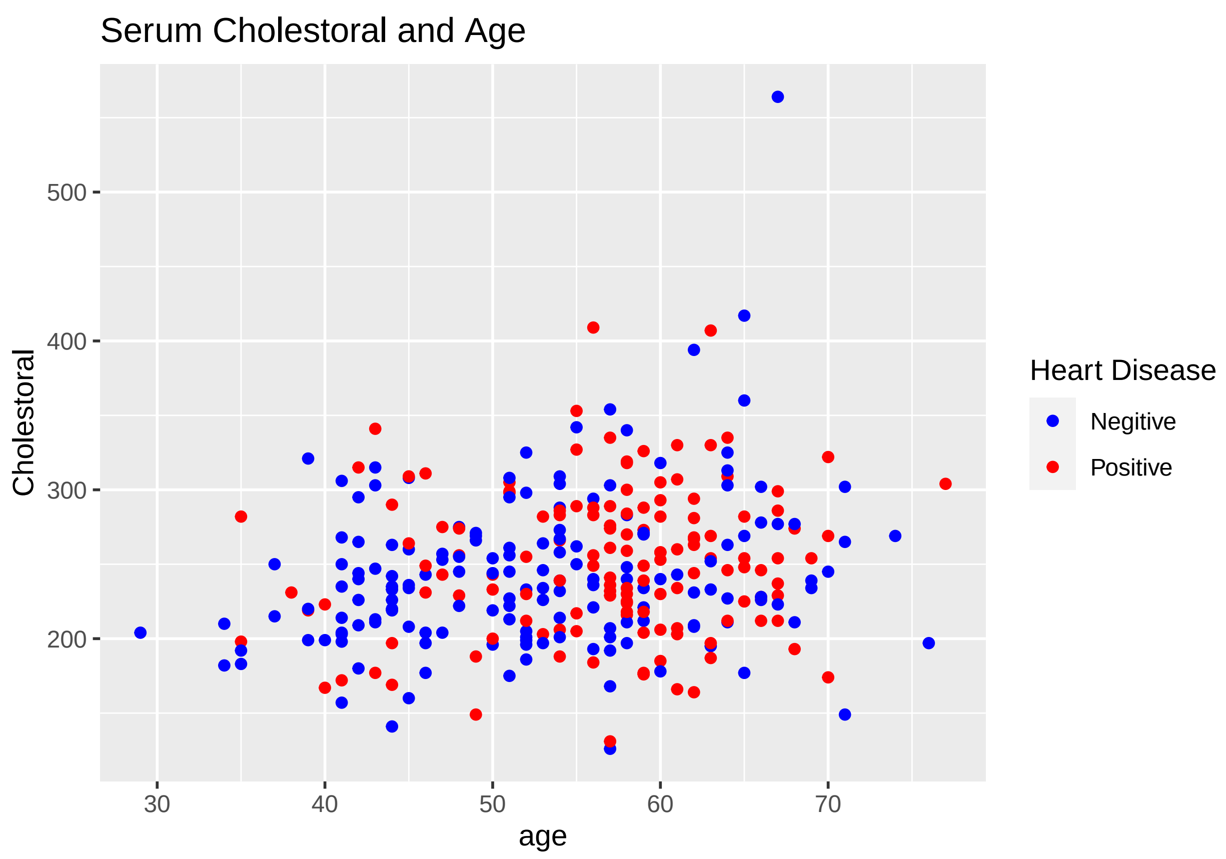
from the box plot, 3 features stand out to me that we will look at in the next few graphs: tresbps, chol, thalach

```
#Scatter plot of resting blood pressure and age
heart %>% ggplot(aes(age,trestbps)) +
  geom_point(aes(color = factor(num)) +
  ggtitle("Resting Blood Presure and Age") +
  ylab("Resting BP") +
  scale_color_manual(name = "Heart Disease",
                      labels = c("Negitive","Positive"),
                     values = c("blue","red"))
```



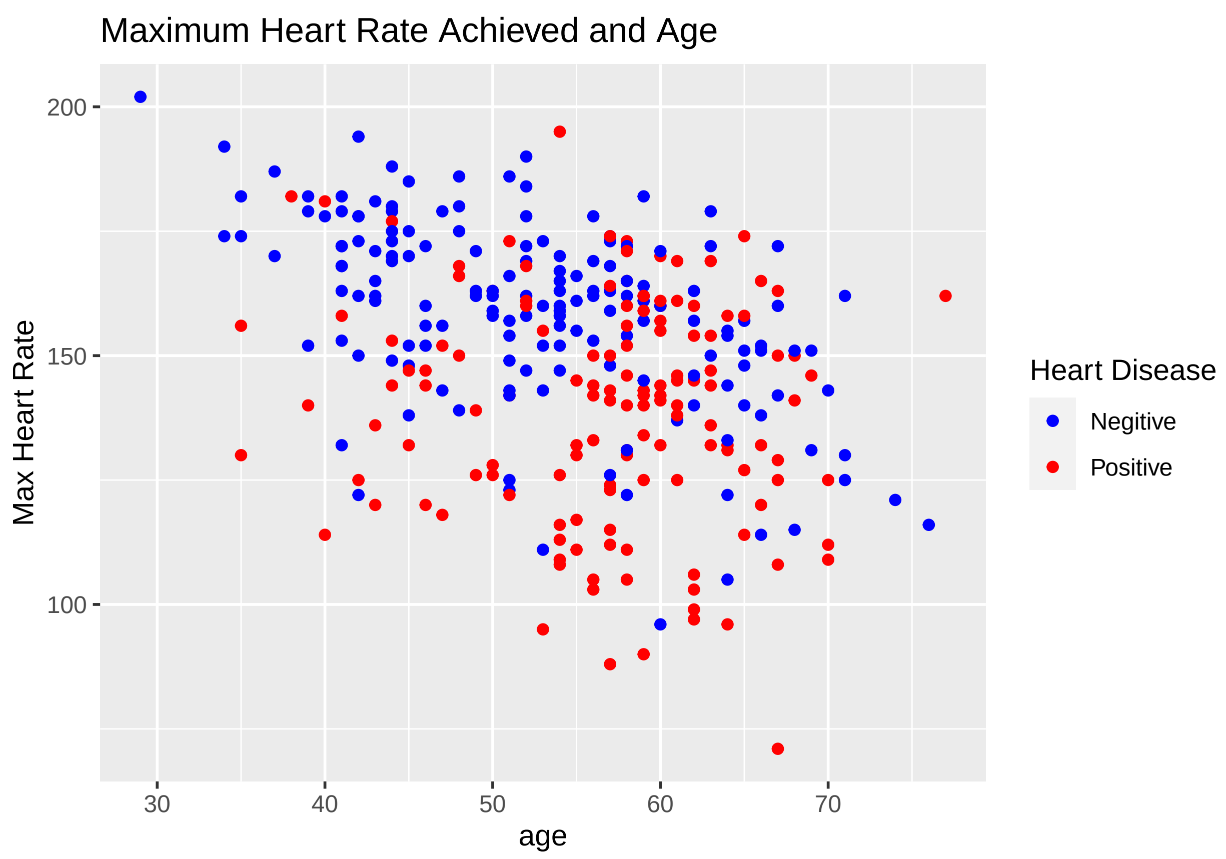
Scatter plot of serum cholestoral and age

```
heart %>% ggplot(aes(age,chol)) +
  geom_point(aes(color = factor(num))) +
  ggtitle("Serum Cholestoral and Age") +
  ylab("Cholestoral") +
  scale_color_manual(name = "Heart Disease",
                    labels = c("Negitive","Positive"),
                    values = c("blue","red"))
```



Scatter plot of maximum heart rate achieved and age

```r
heart %>% ggplot(aes(age,thalach)) +
  geom_point(aes(color = factor(num)))+
  ggtitle("Maximum Heart Rate Achieved and Age") +
  ylab("Max Heart Rate") +
  scale_color_manual(name = "Heart Disease",
                     labels = c("Negitive","Positive"),
                     values = c("blue","red"))
```



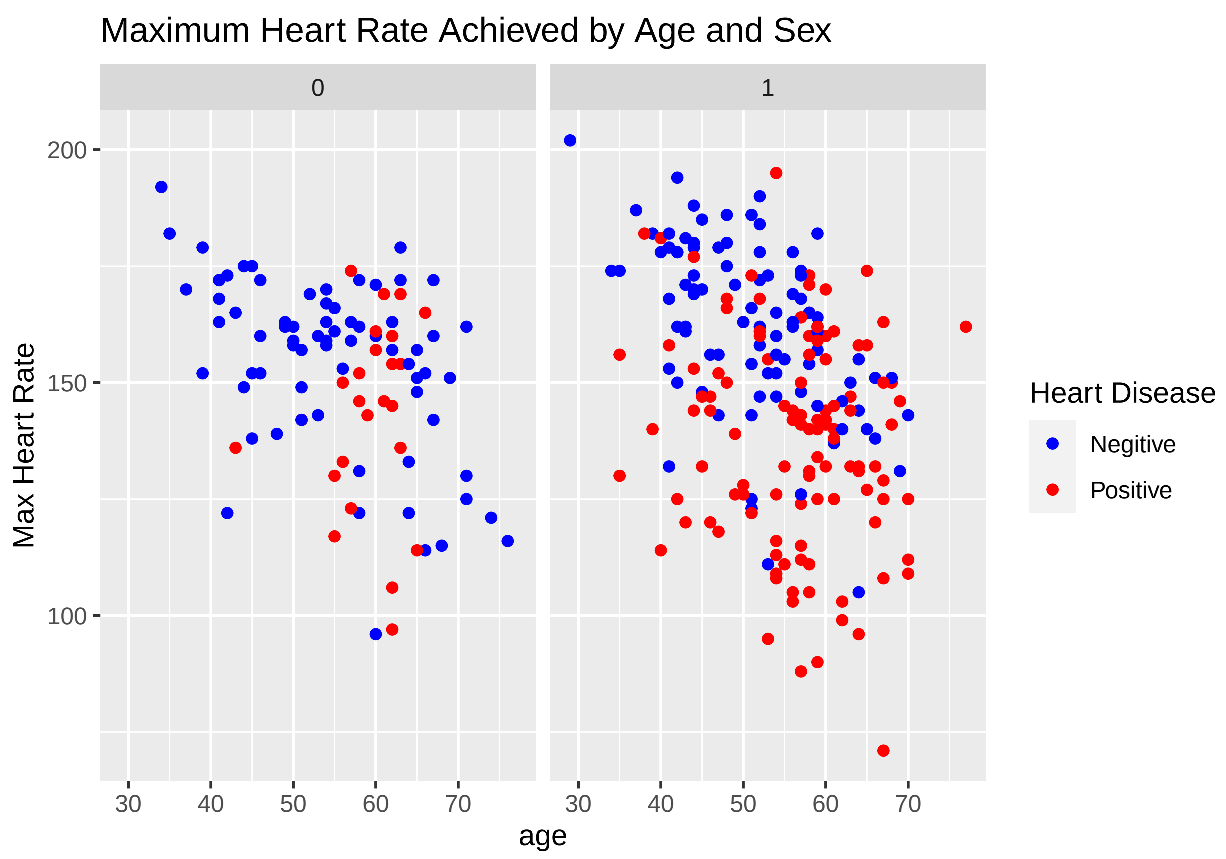
Key insights from the last 3 graphs:

- Max Heart Rate had the largest separation between negative and positive
- Most positive results fall between age 55 - 70
- Very few observations of positive results after the age of 70. This could be due to having a very small sample size. This could also be because people with heart disease do not live past 70 years old very often.

I would like to look more into the max heart rate. This time we will create a scatter plot split by sex.

Scatter plot of maximum heart rate achieved, age and sex 0 = female and 1 = male

```
heart %>% ggplot(aes(age,thalach)) +
  geom_point(aes(color = factor(num))) +
  facet_grid(.~sex) +
  ggtitle("Maximum Heart Rate Achieved by Age and Sex") +
  ylab("Max Heart Rate") +
  scale_color_manual(name = "Heart Disease",
                     labels = c("Negitive","Positive"),
                     values = c("blue","red"))
```

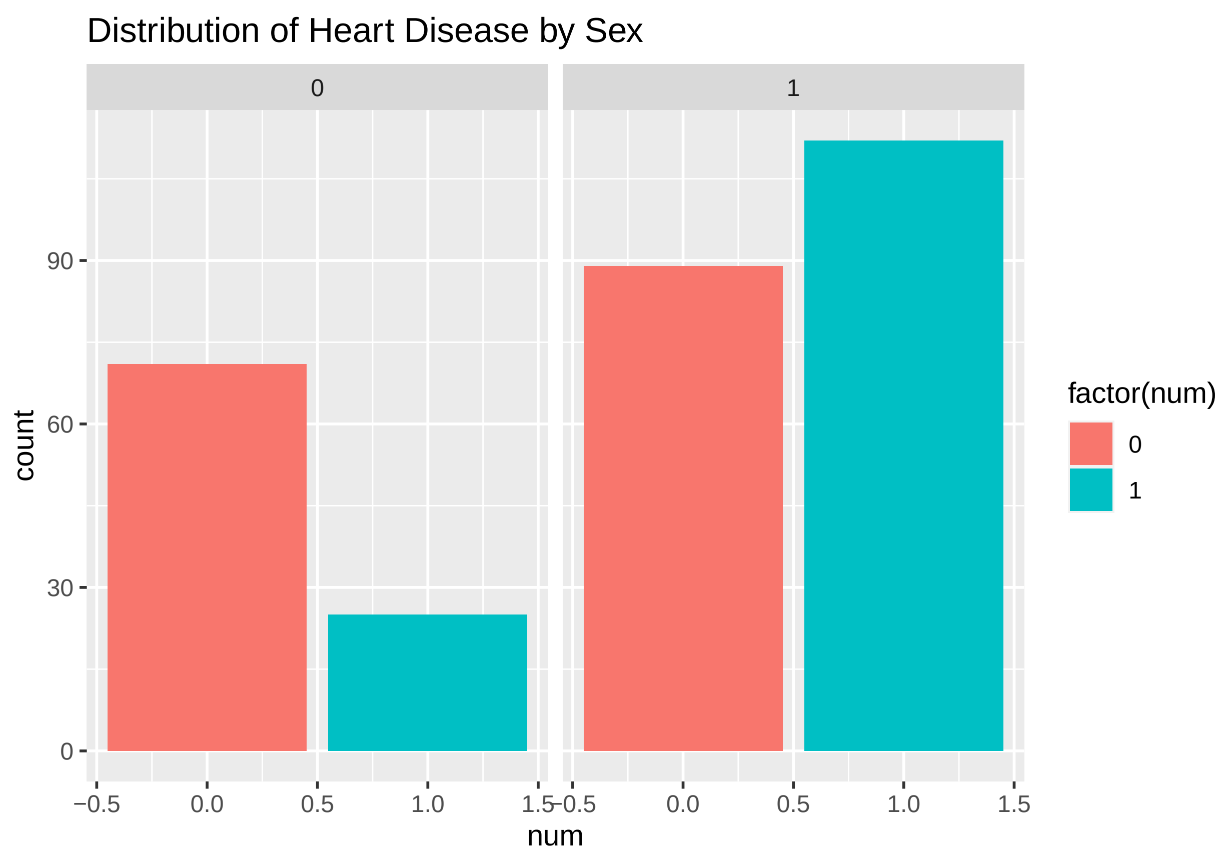


Key insights from this graph:

- Only one female under the age of 50 observed positive
- The majority of male negative observations had max hr over 150
- For the males, the lower the max hr the more positive observations
- For the females, the lower the max hr does not result in more positive observations

bar chart of distribution of heart disease split by sex

```
heart %>% ggplot(aes(num)) +
  geom_bar(aes(fill=factor(num))) +
  facet_grid(.~sex) +
  ggtitle("Distribution of Heart Disease by Sex") +
  scale_color_manual(name = "Heart Disease",
                     labels = c("Negitive","Positive"),
                     values = c("blue","red"))
```

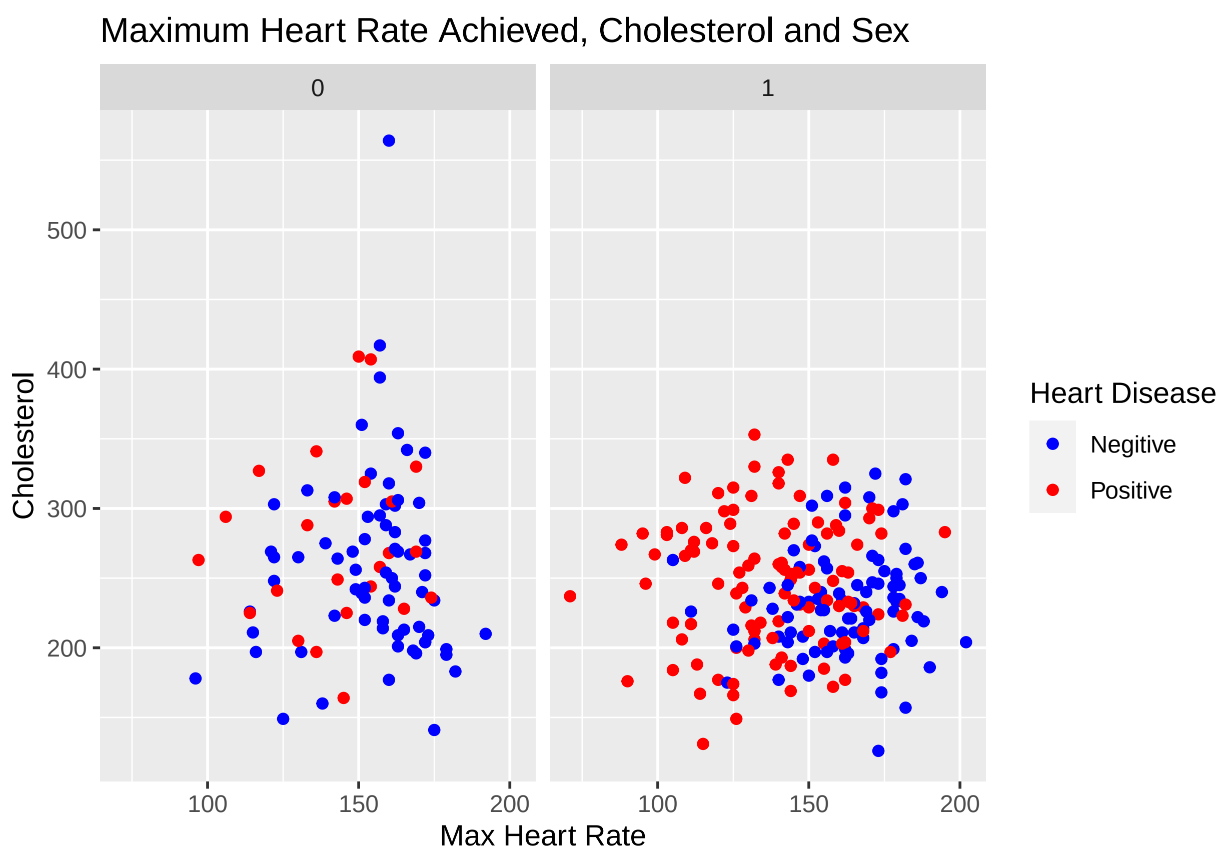


Key insight from the graph:

- The large majority of positive observations in this data set are male

scatter plot of maximum heart rate achieved, cholesterol, sex

```
heart %>% ggplot(aes(thalach,chol)) +
  geom_point(aes(col=factor(num))) +
  facet_grid(.~sex) +
  labs(title = "Maximum Heart Rate Achieved, Cholesterol and Sex",
       x = "Max Heart Rate", y = "Cholesterol") +
  scale_color_manual(name = "Heart Disease",
                     labels = c("Negitive","Positive"),
                     values = c("blue","red"))
```

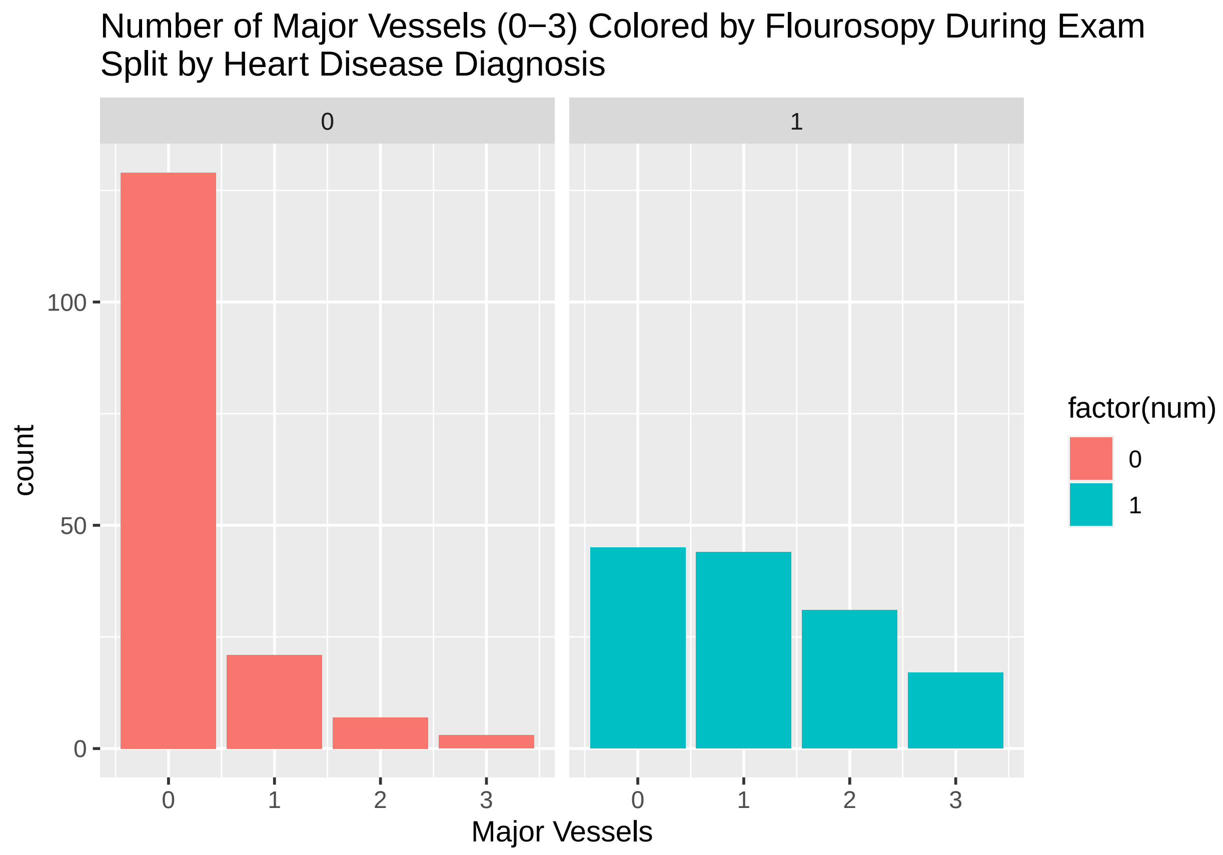


Key insights from this graph:

- For both males and females, high cholesterol and low max hr is more likely to have a positive observation
- For both males and females, low cholesterol and high max hr is more likely to have a negative observation

bar chart of number of major vessels (0-3) colored by flourosopy during exam split by Heart Disease diagnosis

```r
heart %>% ggplot(aes(ca)) +
  geom_bar(aes(fill=factor(num))) +
  facet_grid(.~num) +
  ggtitle("Number of Major Vessels (0-3) Colored by Flourosopy During Exam
Split by Heart Disease Diagnosis") +
  xlab("Major Vessels") +
  scale_color_manual(name = "Heart Disease",
                     labels = c("Negitive","Positive"),
                     values = c("blue","red"))
```



Key insight from this graph:

- Majority of negative observations have all three major vessels functioning properly

4. Models

4.1 Setting up the models

creating variable x which will consist of the data set expect the feature we are trying to predict

```
x <- heart[,-14]
```

creating variable y which will consist the feature we are trying to predict

```
y <- heart$num
```

We will split these two up into training and test

```
set.seed(10,sample.kind = "Rounding")
test_index <- createDataPartition(y,times = 1, p=.2,list = FALSE)
test_x <- x[test_index,]
test_y <- y[test_index]
train_x <- x[-test_index,]
train_y <- y[-test_index]
```

checking to see if the proportions are the same for the train and test set

```
mean(test_y == 0)
```

```
## [1] 0.5666667
```

```
mean(train_y == 0)
```

```
## [1] 0.5316456
```

Will be using k-fold cross validation on all the algorithms creating the k-fold parameters, k is 10

```
control <- trainControl(method = "cv", number = 10, p = .9)
```

4.2 Logistic regression model

```
#training the model using train set
train_glm <- train(train_x, as.factor(train_y), method = "glm",
                   family = "binomial",
                   trControl = control)

#creating the predictions
glm_preds <- predict(train_glm, test_x)

#creating a confusion matrix
logistic_regression <- confusionMatrix(glm_preds,as.factor(test_y),positive = "1")

#viewing accuracy results
logistic_regression
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28  3
##          1  6 23
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 2.679e-06
##
##                   Kappa : 0.6987
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.8846
##             Specificity : 0.8235
##          Pos Pred Value : 0.7931
##          Neg Pred Value : 0.9032
##               Prevalence : 0.4333
##           Detection Rate : 0.3833
##     Detection Prevalence : 0.4833
##        Balanced Accuracy : 0.8541
##
##         'Positive' Class : 1
##
```

4.3 Linear discriminant analysis

```r
#training the model using the train set
train_lda <- train(train_x, as.factor(train_y), method = "lda",
                   trControl = control)

#creating the predictions
lda_preds <- predict(train_lda, test_x)

#creating a confustion matrix
LDA <- confusionMatrix(lda_preds,
                       as.factor(test_y),
                       positive = "1")

#viewing accuracy results
LDA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  3
##          1  5 23
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 5.858e-07
##
##                   Kappa : 0.7309
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.8846
##             Specificity : 0.8529
##          Pos Pred Value : 0.8214
##          Neg Pred Value : 0.9062
##              Prevalence : 0.4333
##          Detection Rate : 0.3833
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.8688
##
##        'Positive' Class : 1
##
```

4.4 Quadratic discriminant analysis

```
#training the model using the train set
train_qda <- train(train_x, as.factor(train_y), method = "qda",
                   trControl = control)

#creating the predictions
qda_preds <- predict(train_qda, test_x)

#creating a confustion matrix
QDA <- confusionMatrix(qda_preds,
                       as.factor(test_y),
                       positive = "1")

#viewing accuracy results
QDA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 25  3
##          1  9 23
##
##                Accuracy : 0.8
##                  95% CI : (0.6767, 0.8922)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 0.0001278
##
##                   Kappa : 0.6035
##
##  Mcnemar's Test P-Value : 0.1489147
##
##             Sensitivity : 0.8846
##             Specificity : 0.7353
##          Pos Pred Value : 0.7188
##          Neg Pred Value : 0.8929
##              Prevalence : 0.4333
##          Detection Rate : 0.3833
##    Detection Prevalence : 0.5333
##       Balanced Accuracy : 0.8100
##
##        'Positive' Class : 1
##
```

4.5 Loess model

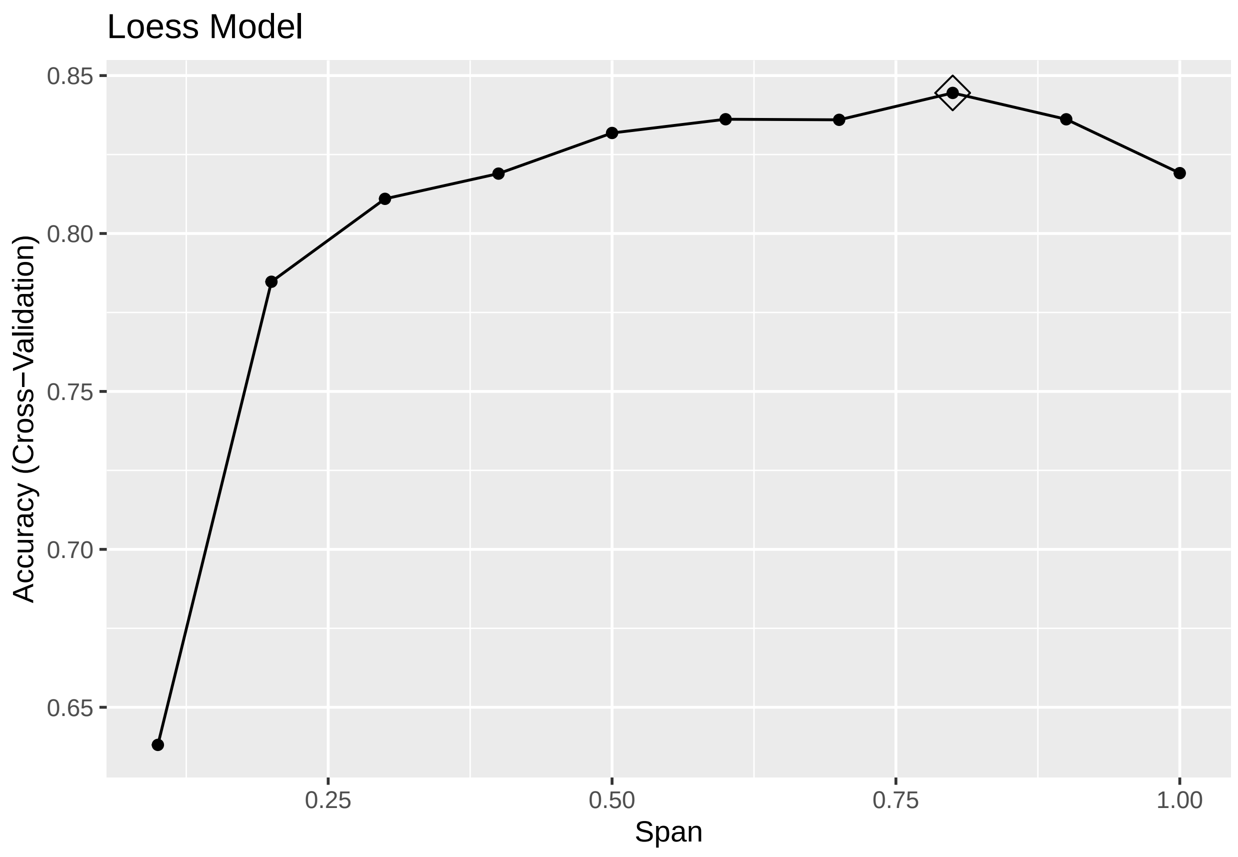
```
#creating grid for the two parameter: span, degree
grid <- expand.grid(span = seq(.1,1, len = 10), degree = 1)

#setting the seed
```

```r
set.seed(5, sample.kind = "Rounding")

#training the model using the training set
train_loess <- train(train_x, as.factor(train_y), method = "gamLoess",
                     trControl = control,
                     tuneGrid = grid)

#creating graph of the tuning result
ggplot(train_loess, highlight = TRUE) +
  ggtitle("Loess Model")
```



```r
#creating the predictions
loess_preds <- predict(train_loess, test_x)

#creating confustion matrix
Loess <- confusionMatrix(loess_preds,
                         as.factor(test_y),
                         positive = "1")

#viewing accuracy result
Loess
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##          0 28  3
##          1  6 23
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 2.679e-06
##
##                   Kappa : 0.6987
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.8846
##             Specificity : 0.8235
##          Pos Pred Value : 0.7931
##          Neg Pred Value : 0.9032
##              Prevalence : 0.4333
##          Detection Rate : 0.3833
##    Detection Prevalence : 0.4833
##       Balanced Accuracy : 0.8541
##
##        'Positive' Class : 1
##
```

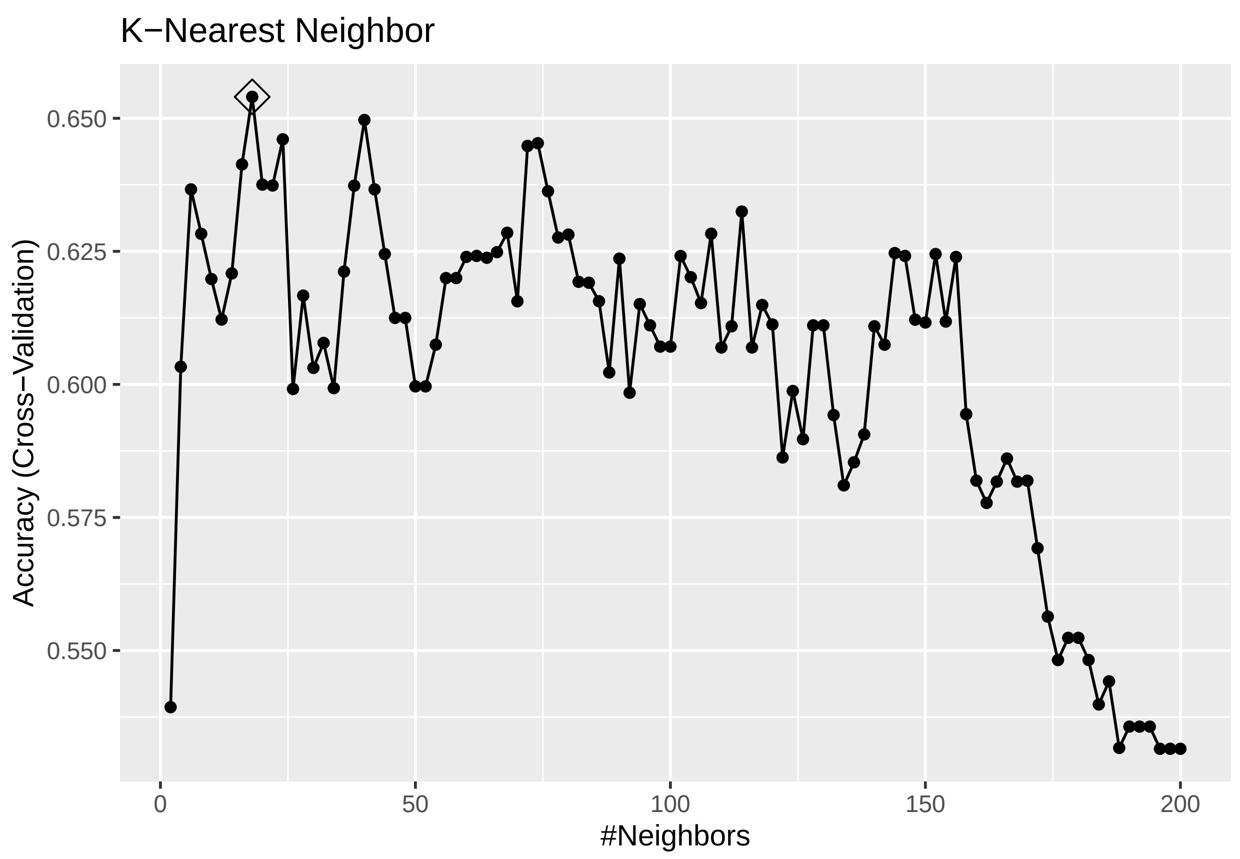
4.6 K-nearest neighbors model

```
#setting the seed
set.seed(7, sample.kind = "Rounding")

#creating tuning parameter for K
tuning <- data.frame(k = seq(2,200,2))

#training the model with the training set
train_knn <- train(train_x, as.factor(train_y),method = "knn",
                   trControl = control,
                   tuneGrid = tuning,)

#creating graph of tuning result
ggplot(train_knn, highlight = TRUE) +
  ggtitle("K-Nearest Neighbor")
```



```
#finding best tuning
train_knn$bestTune
```

```
##    k
## 9 18
```

```
#creating prediction
knn_preds <- predict(train_knn, test_x)

#creating a confustion matrix
Knearest_neighbors <- confusionMatrix(knn_preds,
                                      as.factor(test_y),
                                      positive = "1")

#viewing accuracy result
Knearest_neighbors
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 21  8
##          1 13 18
##
##                Accuracy : 0.65
##                  95% CI : (0.516, 0.7687)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 0.1200
##
##                   Kappa : 0.3031
##
##  Mcnemar's Test P-Value : 0.3827
##
##             Sensitivity : 0.6923
##             Specificity : 0.6176
##          Pos Pred Value : 0.5806
##          Neg Pred Value : 0.7241
##              Prevalence : 0.4333
##          Detection Rate : 0.3000
##    Detection Prevalence : 0.5167
##       Balanced Accuracy : 0.6550
##
##        'Positive' Class : 1
##
```

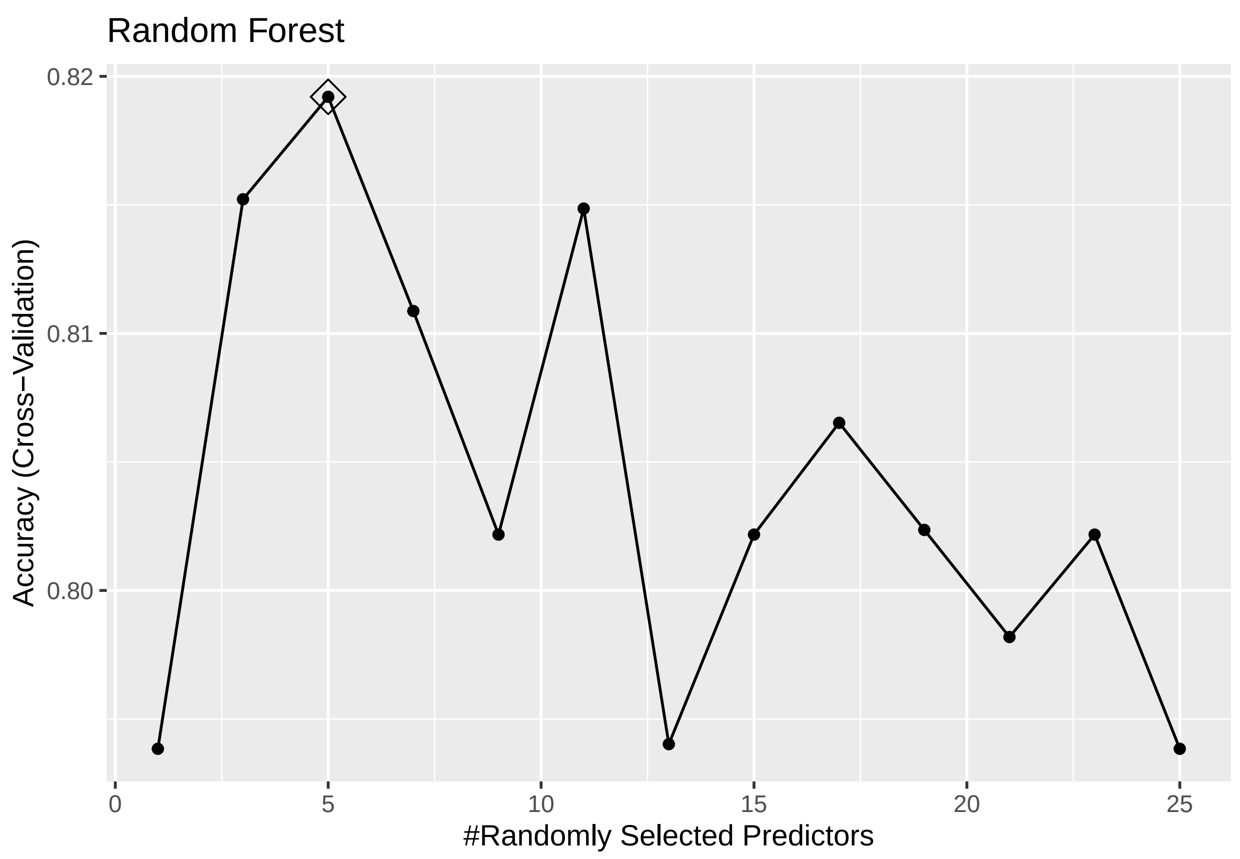
4.7 Random Forest

```
#setting the seed
set.seed(9, sample.kind = "Rounding")

#setting the tuning parameters
tuning <- data.frame(mtry = seq(1,25,2))

#training the model using the train set
train_rf <- train(train_x, as.factor(train_y),method = "rf",
                  tuneGrid = tuning,
                  trControl = control,
                  importance = TRUE)

#creating graph of tuning results
ggplot(train_rf, highlight = TRUE) +
  ggtitle("Random Forest")
```



```r
#finding the best tuning result
train_rf$bestTune
```

```
##   mtry
## 3    5
```

```r
#creating predictions
rf_preds <- predict(train_rf, test_x)

#creating a confusion matrix
Random_Forest <- confusionMatrix(rf_preds,
                                 as.factor(test_y),
                                 positive = "1")

#viewing accuracy result
Random_Forest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  3
##          1  7 23
##
##                Accuracy : 0.8333
##                    95% CI : (0.7148, 0.9171)
##       No Information Rate : 0.5667
##       P-Value [Acc > NIR] : 1.084e-05
##
##                     Kappa : 0.6667
##
##    Mcnemar's Test P-Value : 0.3428
##
##               Sensitivity : 0.8846
##               Specificity : 0.7941
##            Pos Pred Value : 0.7667
##            Neg Pred Value : 0.9000
##                Prevalence : 0.4333
##            Detection Rate : 0.3833
##      Detection Prevalence : 0.5000
##         Balanced Accuracy : 0.8394
##
##          'Positive' Class : 1
##
```

```r
#viewing importance
varImp(train_rf)
```

```
## rf variable importance
##
##          Importance
## thal         100.00
## ca            99.83
## cp            84.80
## oldpeak       75.11
## sex           52.99
## thalach       43.50
## exang         38.74
## slope         35.54
## age           34.25
## restecg       33.88
## trestbps      30.07
## fbs           15.11
## chol           0.00
```

According to this random forest model, the three most import factors to determining if you have heart disease or not with this data are:

- having a heart defect
- the number of major vessels that were working
- the type of chest pain

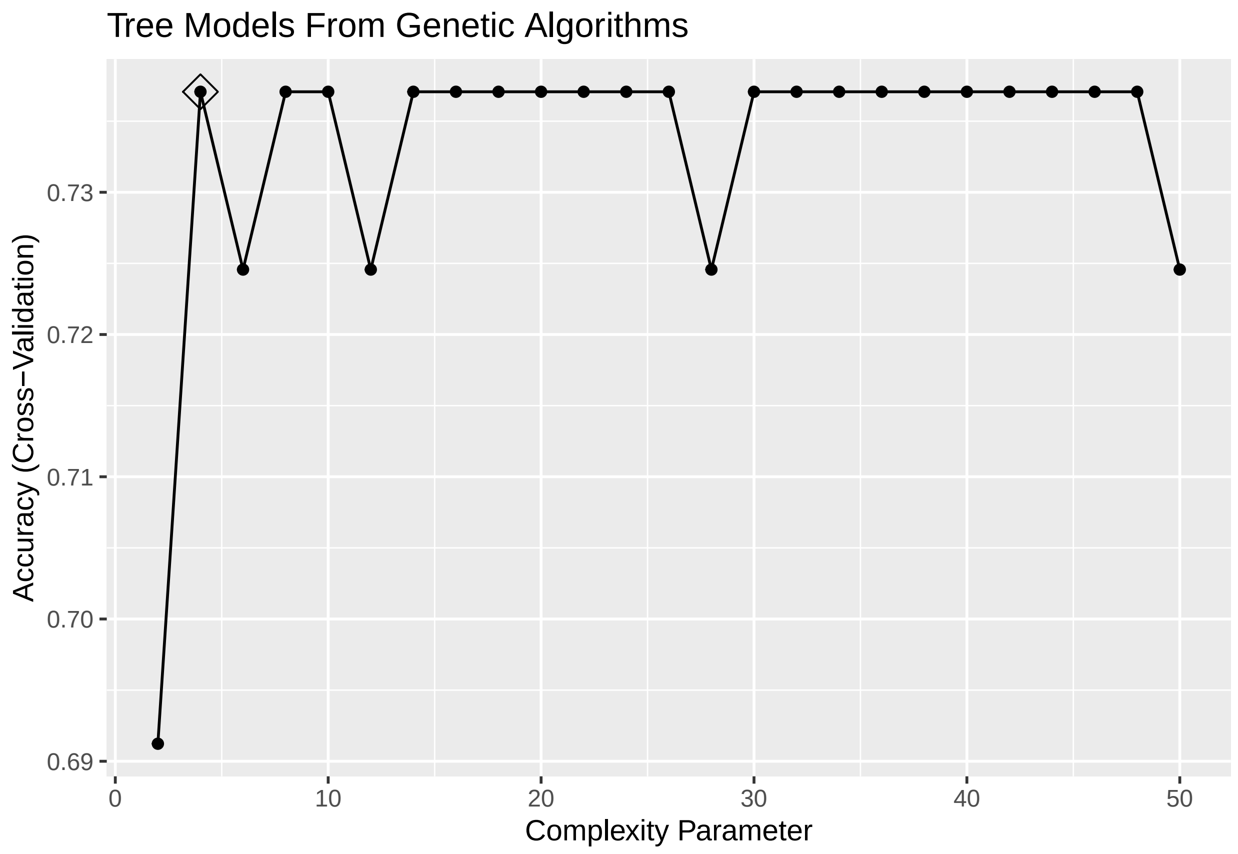
4.8 Tree Models from Genetic Algorithms

```r
#setting the seed
set.seed(7, sample.kind = "Rounding")
```

```r
#setting the tuning parameter alpha
tuning <- data.frame(alpha = seq(2,50,2))

#training the model on the train set
train_tree <- train(train_x, as.factor(train_y),method = "evtree",
                    tuneGrid = tuning,
                    trControl = control)

#creating a graph for the tuning results
ggplot(train_tree, highlight = TRUE) +
  ggtitle("Tree Models From Genetic Algorithms")
```



```r
#finding best tune
train_tree$bestTune
```

```
##   alpha
## 2     4
```

```r
#creating predictions
tree_preds <- predict(train_tree, test_x)

#creating a confusion matrix
tree_model <- confusionMatrix(tree_preds,
                              as.factor(test_y),
                              positive = "1")

#viewing accuracy results
tree_model
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  6
##          1  7 20
##
##                Accuracy : 0.7833
##                  95% CI : (0.658, 0.8793)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 0.0003781
##
##                   Kappa : 0.5608
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.7692
##             Specificity : 0.7941
##          Pos Pred Value : 0.7407
##          Neg Pred Value : 0.8182
##              Prevalence : 0.4333
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.4500
##       Balanced Accuracy : 0.7817
##
##        'Positive' Class : 1
##
```

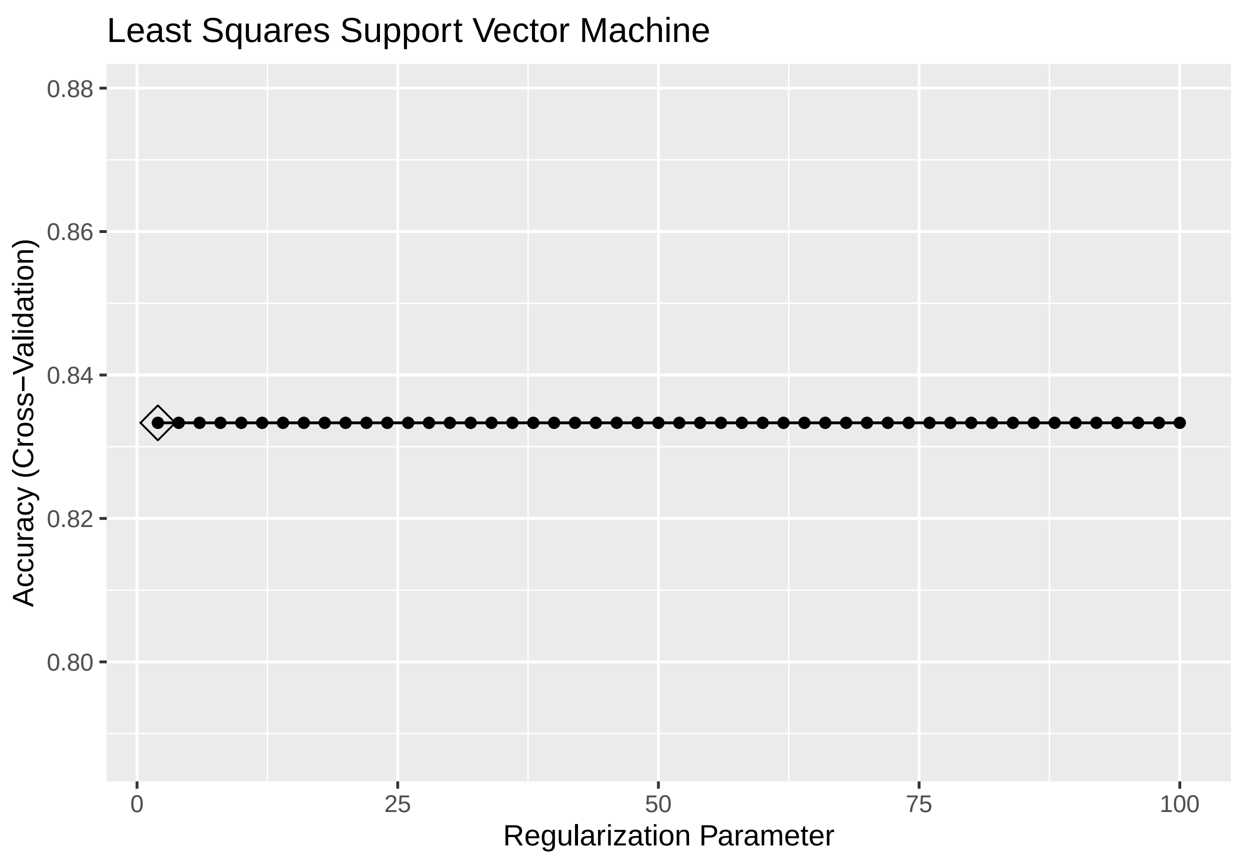
4.9 Least Squares Support Vector Machine

```
#setting the seed
set.seed(7, sample.kind = "Rounding")

#setting the tuning parameter alpha
tuning <- data.frame(tau = seq(2,100,2))

#training the model on the train set
train_SVM <- train(train_x, as.factor(train_y),
                   method = "lssvmLinear",
                   tuneGrid = tuning,
                   trControl = control)

#creating a graph for the tuning results
ggplot(train_SVM, highlight = TRUE) +
  ggtitle("Least Squares Support Vector Machine")
```



```
#finding best tune
train_SVM$bestTune
```

```
##   tau
## 1   2
```

```
#creating predictions
SVM_preds <- predict(train_SVM, test_x)

#creating a confusion matrix
SVM <- confusionMatrix(SVM_preds,
                       as.factor(test_y),
                       positive = "1")

#viewing accuracy results
SVM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  3
##          1  5 23
##
##                Accuracy : 0.8667
##                   95% CI : (0.7541, 0.9406)
##      No Information Rate : 0.5667
##      P-Value [Acc > NIR] : 5.858e-07
##
##                    Kappa : 0.7309
##
##   Mcnemar's Test P-Value : 0.7237
##
##              Sensitivity : 0.8846
##              Specificity : 0.8529
##           Pos Pred Value : 0.8214
##           Neg Pred Value : 0.9062
##               Prevalence : 0.4333
##           Detection Rate : 0.3833
##     Detection Prevalence : 0.4667
##        Balanced Accuracy : 0.8688
##
##         'Positive' Class : 1
##
```

4.10 Bayesian Generalized Linear Model

```r
#setting the seed
set.seed(7, sample.kind = "Rounding")

#training the model on the train set
train_nb <- train(train_x, as.factor(train_y),method = "bayesglm",
                   trControl = control)

#creating predictions
nb_preds <- predict(train_nb, test_x)

#creating a confusion matrix
nb <- confusionMatrix(nb_preds,
                      as.factor(test_y),
                      positive = "1")

#viewing accuracy results
nb
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 28  3
##          1  6 23
##
##                 Accuracy : 0.85
##                   95% CI : (0.7343, 0.929)
##      No Information Rate : 0.5667
##      P-Value [Acc > NIR] : 2.679e-06
##
```

```
##                    Kappa : 0.6987
##
##   Mcnemar's Test P-Value : 0.505
##
##              Sensitivity : 0.8846
##              Specificity : 0.8235
##           Pos Pred Value : 0.7931
##           Neg Pred Value : 0.9032
##               Prevalence : 0.4333
##           Detection Rate : 0.3833
##     Detection Prevalence : 0.4833
##        Balanced Accuracy : 0.8541
##
##         'Positive' Class : 1
##
```

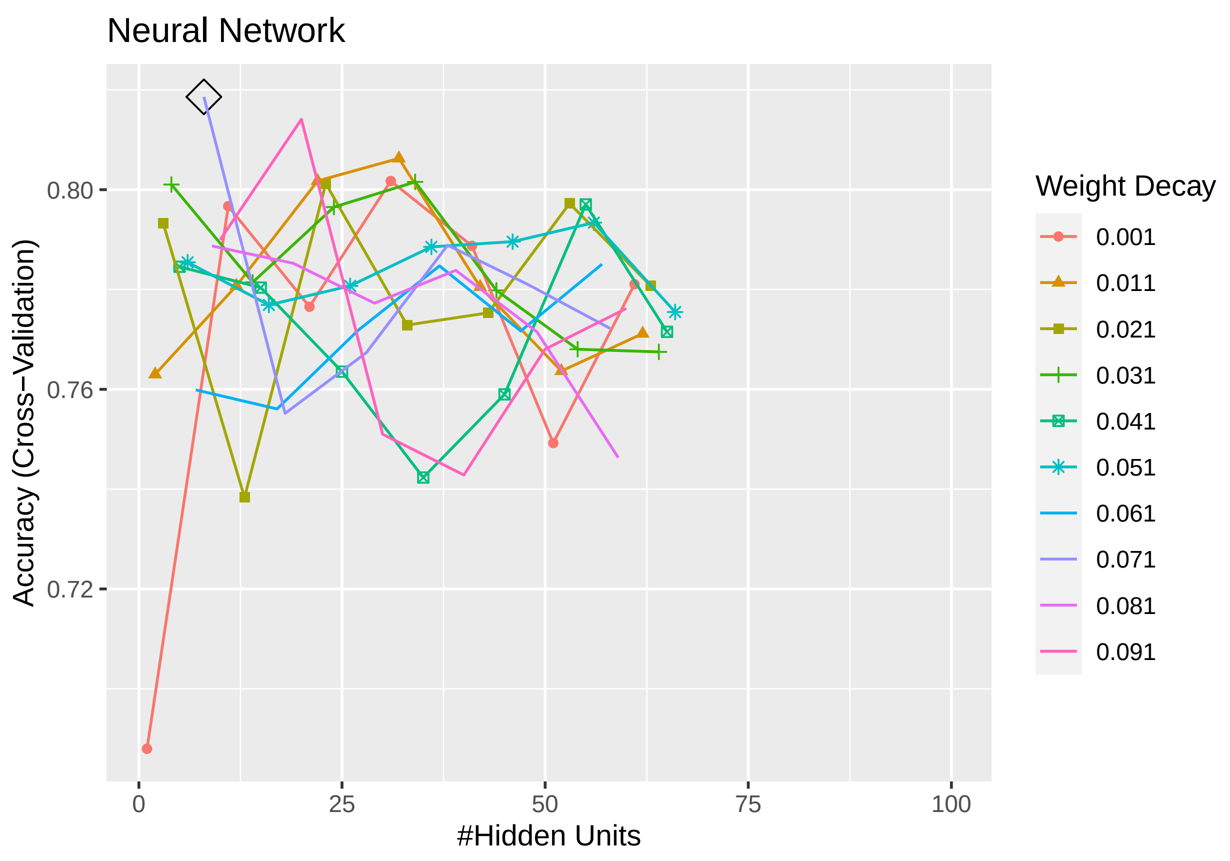
4.11 Neural Network

```
#setting the seed
set.seed(7, sample.kind = "Rounding")

#setting the tuning parameter alpha
tuning <- data.frame(size = seq(100),
                     decay = seq(.001,.1,.01))

#training the model on the train set
train_nn <- train(train_x, as.factor(train_y),method = "nnet",
                   tuneGrid = tuning,
                   trControl = control)
```

```
#creating a graph for the tuning results
ggplot(train_nn, highlight = TRUE) +
  ggtitle("Neural Network")
```



```
#finding best tune
train_nn$bestTune

#creating predictions
nn_preds <- predict(train_nn, test_x)

#creating a confusion matrix
nn <- confusionMatrix(nn_preds,
                      as.factor(test_y),
                      positive = "1")
```

```
#viewing confusion matrix
nn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 25  4
##          1  9 22
##
##                Accuracy : 0.7833
##                  95% CI : (0.658, 0.8793)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 0.0003781
```

```
##
##                   Kappa : 0.5686
##
##  Mcnemar's Test P-Value : 0.2672575
##
##             Sensitivity : 0.8462
##             Specificity : 0.7353
##          Pos Pred Value : 0.7097
##          Neg Pred Value : 0.8621
##              Prevalence : 0.4333
##          Detection Rate : 0.3667
##    Detection Prevalence : 0.5167
##       Balanced Accuracy : 0.7907
##
##        'Positive' Class : 1
##
```

4.12 Ensemble

```r
preds <- data.frame(log_r = glm_preds,
                    lda = lda_preds,
                    qda = qda_preds,
                    loess = loess_preds,
                    knn = knn_preds,
                    rf = rf_preds,
                    tree = tree_preds,
                    svm = SVM_preds,
                    nb = nb_preds,
                    nn = nn_preds)
preds
```

```
##    log_r lda qda loess knn rf tree svm nb nn
## 1      0   0   0     0   0  1    0   0  0  0
## 2      0   0   0     0   0  0    0   0  0  0
## 3      0   0   1     0   1  0    1   0  0  0
## 4      1   1   1     1   0  1    1   1  1  1
## 5      0   0   0     0   0  0    0   0  0  0
## 6      0   0   0     0   0  0    0   0  0  0
## 7      1   1   1     1   1  1    1   1  1  1
## 8      0   0   0     0   0  0    0   0  0  0
## 9      0   0   0     0   1  0    0   0  0  0
## 10     1   1   0     1   1  0    0   1  1  1
## 11     1   1   1     1   0  1    1   1  1  1
## 12     0   0   0     0   0  0    0   0  0  0
## 13     1   1   1     1   1  1    1   1  1  1
## 14     1   1   1     1   1  1    1   1  1  1
## 15     0   0   0     0   1  0    0   0  0  0
## 16     1   1   1     1   1  1    1   1  1  1
## 17     0   0   0     0   0  0    0   0  0  0
## 18     1   1   1     1   1  1    1   1  1  1
## 19     0   0   0     0   0  0    0   0  0  0
## 20     1   1   1     1   1  1    1   1  1  1
## 21      1    1    1      1    1   1     1    1   1   1
## 22      0    0    0      0    0   0     0    0   0   0
## 23      0    0    0      0    0   1     0    0   0   1
## 24      1    1    1      1    1   1     0    1   1   1
## 25      0    0    0      0    0   0     1    0   0   0
## 26      1    1    1      1    1   1     1    1   1   0
## 27      0    0    0      0    0   0     0    0   0   0
## 28      1    1    1      1    0   1     1    1   1   1
## 29      0    0    1      0    0   0     0    0   0   0
## 30      0    0    0      0    0   0     0    0   0   0
## 31      1    1    1      1    1   1     1    1   1   1
## 32      1    1    1      1    1   1     1    1   1   1
## 33      0    0    0      0    0   0     0    0   0   0
## 34      1    1    1      1    0   1     1    1   1   1
## 35      0    0    0      0    1   0     0    0   0   0
## 36      0    0    0      0    1   0     0    0   0   0
## 37      1    1    1      1    1   1     1    1   1   1
## 38      1    0    1      1    0   0     0    0   1   1
## 39      1    1    1      1    1   1     1    1   1   1
## 40      1    1    1      1    1   1     1    1   1   1
## 41      1    1    1      1    1   1     0    1   1   1
## 42      0    0    0      0    0   0     0    0   0   0
## 43      0    0    1      0    1   1     0    0   0   1
## 44      1    1    1      1    1   1     1    1   1   1
## 45      0    0    0      0    0   0     0    0   0   0
## 46      0    0    0      0    0   0     0    0   0   0
## 47      0    0    0      0    1   0     0    0   0   0
## 48      0    0    0      0    0   0     0    0   0   0
## 49      0    0    0      0    1   0     0    0   0   0
## 50      1    1    1      1    1   1     0    1   1   1
## 51      0    0    0      0    1   0     0    0   0   0
## 52      0    0    0      0    0   0     0    0   0   0
## 53      1    1    1      1    1   1     1    1   1   1
## 54      1    1    1      1    1   1     1    1   1   1
## 55      0    0    0      0    0   0     0    0   0   1
## 56      1    1    1      1    0   1     1    1   1   1
## 57      1    1    1      1    0   1     1    1   1   1
## 58      0    0    1      0    0   0     1    0   0   0
## 59      1    1    1      1    1   1     1    1   1   1
## 60      1    1    1      1    1   1     1    1   1   1
```

Now that we have a data frame with all the predictions, we will take the mode of each sample and use that result as the ensemble's prediction for each sample.

```r
ensemble <- apply(preds,1,function(x) names(which.max(table(x))))

#factoring the results
ensemble <- as.factor(ensemble)
```

creating a confusion matrix

```r
ensemble <- confusionMatrix(ensemble,
                                as.factor(test_y),
                                positive = "1")
ensemble
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  3
##          1  5 23
##
##                Accuracy : 0.8667
##                  95% CI : (0.7541, 0.9406)
##     No Information Rate : 0.5667
##     P-Value [Acc > NIR] : 5.858e-07
##
##                   Kappa : 0.7309
##
##  Mcnemar's Test P-Value : 0.7237
##
##             Sensitivity : 0.8846
##             Specificity : 0.8529
##          Pos Pred Value : 0.8214
##          Neg Pred Value : 0.9062
##              Prevalence : 0.4333
##          Detection Rate : 0.3833
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.8688
##
##        'Positive' Class : 1
##
```

5. Results

5.1 Results

creating a results table

```
cm_list <- list(log_r = logistic_regression,
                    lda = LDA,
                    qda = QDA,
                    loess = Loess,
                    knn = Knearest_neighbors,
                    rf = Random_Forest,
                    tree = tree_model,
                    svm = SVM,
                    nb = nb,
                    nn = nn,
                all = ensemble)

cm_results <- sapply(cm_list, function(x) x$byClass)
```

```r
options(digits = 2)

results_table <- kable(cm_results)
results_table
```

	log_r	lda	qda	loess	knn	rf	tree	svm	nb	nn	all
Sensitivity	0.88	0.88	0.88	0.88	0.69	0.88	0.77	0.88	0.88	0.85	0.88
Specificity	0.82	0.85	0.74	0.82	0.62	0.79	0.79	0.85	0.82	0.74	0.85
Pos Pred Value	0.79	0.82	0.72	0.79	0.58	0.77	0.74	0.82	0.79	0.71	0.82
Neg Pred Value	0.90	0.91	0.89	0.90	0.72	0.90	0.82	0.91	0.90	0.86	0.91
Precision	0.79	0.82	0.72	0.79	0.58	0.77	0.74	0.82	0.79	0.71	0.82
Recall	0.88	0.88	0.88	0.88	0.69	0.88	0.77	0.88	0.88	0.85	0.88
F1	0.84	0.85	0.79	0.84	0.63	0.82	0.75	0.85	0.84	0.77	0.85
Prevalence	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
Detection Rate	0.38	0.38	0.38	0.38	0.30	0.38	0.33	0.38	0.38	0.37	0.38
Detection Prevalence	0.48	0.47	0.53	0.48	0.52	0.50	0.45	0.47	0.48	0.52	0.47
Balanced Accuracy	0.85	0.87	0.81	0.85	0.65	0.84	0.78	0.87	0.85	0.79	0.87

5.2 Best model

Three models had the highest F1 score of .85: Linear discriminant analysis, Least Squares Support Vector Machine, Ensemble

In fact, they had the same metric performances as you can see below, suggesting they had the same predictions. So it looks like we have a three way tie!

```r
kable(cm_results[,c(2,8,11)])
```

	lda	svm	all
Sensitivity	0.88	0.88	0.88
Specificity	0.85	0.85	0.85
Pos Pred Value	0.82	0.82	0.82
Neg Pred Value	0.91	0.91	0.91
Precision	0.82	0.82	0.82
Recall	0.88	0.88	0.88
F1	0.85	0.85	0.85
Prevalence	0.43	0.43	0.43
Detection Rate	0.38	0.38	0.38
Detection Prevalence	0.47	0.47	0.47
Balanced Accuracy	0.87	0.87	0.87

5.3 Brief thoughts about the results

We set out with a goal of training a model that could predict the correct diagnosis with a F1 score of 85%. Not only did we achieve this goal but 3 out of the 11 models were able to predict at least 85%. I was a bit surprised by the results, I was expected the ensemble to have a higher score then the rest or maybe the neural network. But I was please to see that more than one model was able to achieve a F1 score of .85.

4. Conclusion

4.1 Summary

We set out to use the UCI data set on Heart Disease to create a model that could correctly predict Heart Disease diagnoses. We set a goal of achieving an F1 score of .85. We started by downloading the UCI data set on Heart Disease. We then cleaned the data set and prepared it for analysis. We split the data set into training and test sets. We trained 10 algorithms using the train set and applied the k-fold cross validation technique with a k of 10. We found that having a heart defect, the number of major vessels that were working, and the type of chest pain we the most important factors in determining if you have heart disease or not, using this data set. We achieved our goal of creating a model with a F1 score of .85 with 3 models: Linear discriminant analysis, Least Squares Support Vector Machine,Ensemble. They each had a F1 score of .85.

4.2 Limitations

For me the biggest limitation in this project is the size of the data set. With only 303 observations this is a very small sample size. The other limitation is the data within the data set. 14 features is enough to achieve a high prediction accuracy, as we proved, but I think with more features we could achieve a score over 90%.

4.3 Future Work

For the future, I would be curious to see how these algorithms preform on a much larger data set, say 10 million plus observation data set. Along with a data set that has more features such as: height, weight, if parents had heart disease, use of drugs and alcohol, exercise amount, etc. I would be curious to see which algorithms preform better and if any preform worse. One final thing that I would include is adding more algorithms to this project. These 10 algorithms are not the only algorithms that work well with classification and they may produce a higher F1 score.