Heart Disease Prediction

Tuong Nguyen

March 2, 2019

Abstract

Heart disease is increasing cause of mortality globally. This project uses different modeling methods to predict the chance of having heart disease. It uses accuracy to measure how accurate the model will be. The project uses Logistic Regression, KNN, Classification Trees, and Random Forest for its modeling. Furthermore, uses K-Fold Cross Validation on all of these methods.

Introduction

As a medical doctor, more than half of our patients are at risk of having some type of heart disease. As a matter of fact, heart disease is overcoming chronic condition for common cause of mortality in adults. There are many types of heart disease; common ones that causes mortality are stroke, coronary heart disease and rheumatic heart disease. According to World Health Organization(WHO), there are 17.5 million death from cardiovascular disease globally in 2005 up from 14.4 million in 1990. More than 80 percent of these deaths are from low to middle income countries. The projected deaths by 2030 are more than 30 and 25 million for middle and low income respectively. There are many factors that contribute to heart disease. Some can be control through lifestyle modifications and medications. While others factor can't be control such as age, genetics, gender, etc.

This project will use the data that is available on kaggle to analyze the data and uses various machine learning methods to predict the chance of a person will have heart disease and it will measure the accuracies to find which method is the best.

Data

The dataset was downloaded from https://www.kaggle.com/ronitf/heart-disease-uci into folder data. The data contains 14 attributes. *Table 1* below contains the descriptions of each of the attributes.

Table 1: Description of Data Attributes

value	description
age	age in years
sex	1=male, 0=female
cp	chest pain type 1 - typical agina, 2 - atypical agina, 3 - non-anginal pain, 4 - asymptomatic
trestbps	resting blood pressure in mmHg on admission to the hospital
chol	serum cholesterol in mg/dl
fbs	fasting blood sugar $>$ 120 mg/dl, 1 - true, 0 - false
restecg	resting electrocardiographic results - 0 - normal, 1 - having ST-T wave abnormality
	(T wave inversion and/or ST elevation or depression of $>$ 0.05mv),
	2 - showing probable or definite left ventricular hypertrophy by Estes criteria
thalach	Maximum heart rate achieve
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
ca	Number of major blood vessels(0-3) colored by flourosopy
thal	3-normal,6 - fixed defect, 7 - reversable defect
target	target 1 or 0

The data is read into the data frame, heartDisease, using function, read_csv() in R. The data is in .csv file with comma delimited. The following code shows that the structure of heartDisease and its dimensions. The dimensions contains 303 rows of observation and 14 columns of attributes. The code also provide what the data looks like with the first and last few rows shown in table 2 and 3 respectively. The data is surprisingly small compared to other data used for machine learning.

```
# read in the comma delimited data.
heartDisease <- read_csv("./data/heart.csv")
```

```
## Parsed with column specification:
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trestbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalach = col_double(),
##   exang = col_double(),
##   oldpeak = col_double(),
##   slope = col_double(),
##   ca = col_double(),
##   thal = col_double(),
##   target = col_double()
## )
```

```
str(heartDisease)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 303 obs. of  14 variables:
##  $ age     : num  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex     : num  1 1 0 1 0 1 0 1 1 1 ...
##  $ cp      : num  3 2 1 1 0 0 1 1 2 2 ...
##  $ trestbps: num  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : num  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 1 0 ...
##  $ restecg : num  0 1 0 1 1 1 0 1 1 1 ...
##  $ thalach : num  150 187 172 178 163 148 153 173 162 174 ...
##  $ exang   : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slope   : num  0 0 2 2 2 1 1 2 2 2 ...
##  $ ca      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ thal    : num  1 2 2 2 2 1 2 3 3 2 ...
##  $ target  : num  1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_double(),
##   ..   cp = col_double(),
##   ..   trestbps = col_double(),
##   ..   chol = col_double(),
##   ..   fbs = col_double(),
##   ..   restecg = col_double(),
##   ..   thalach = col_double(),
##   ..   exang = col_double(),
##   ..   oldpeak = col_double(),
##   ..   slope = col_double(),
##   ..   ca = col_double(),
##   ..   thal = col_double(),
##   ..   target = col_double()
##   .. )
```

```
dim(heartDisease)
```

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

```
head(heartDisease) %>% kable(caption='First 6 Observations from Heart Disease') %>% kable_styling(boots
```

Table 2: First 6 Observations from Heart Disease

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
57	1	0	140	192	0	1	148	0	0.4	1	0	1	1

```
tail(heartDisease) %>% kable(caption='Last 6 Observations from Heart Disease') %>% kable_styling(bootst
```

Table 3: Last 6 Observations from Heart Disease

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
59	1	0	164	176	1	0	90	0	1.0	1	2	1	0
57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

The data need to be verified whether there are any NAs in them so that it could be replaced with '0'. The following code will sum up any NAs in the heartDisease data frame. From the output of the code, all of the observations do not contain any NAs in the data.

```
## Sum up any NA in each attributes.
sapply(heartDisease, function(x) sum(is.na(x)))
```

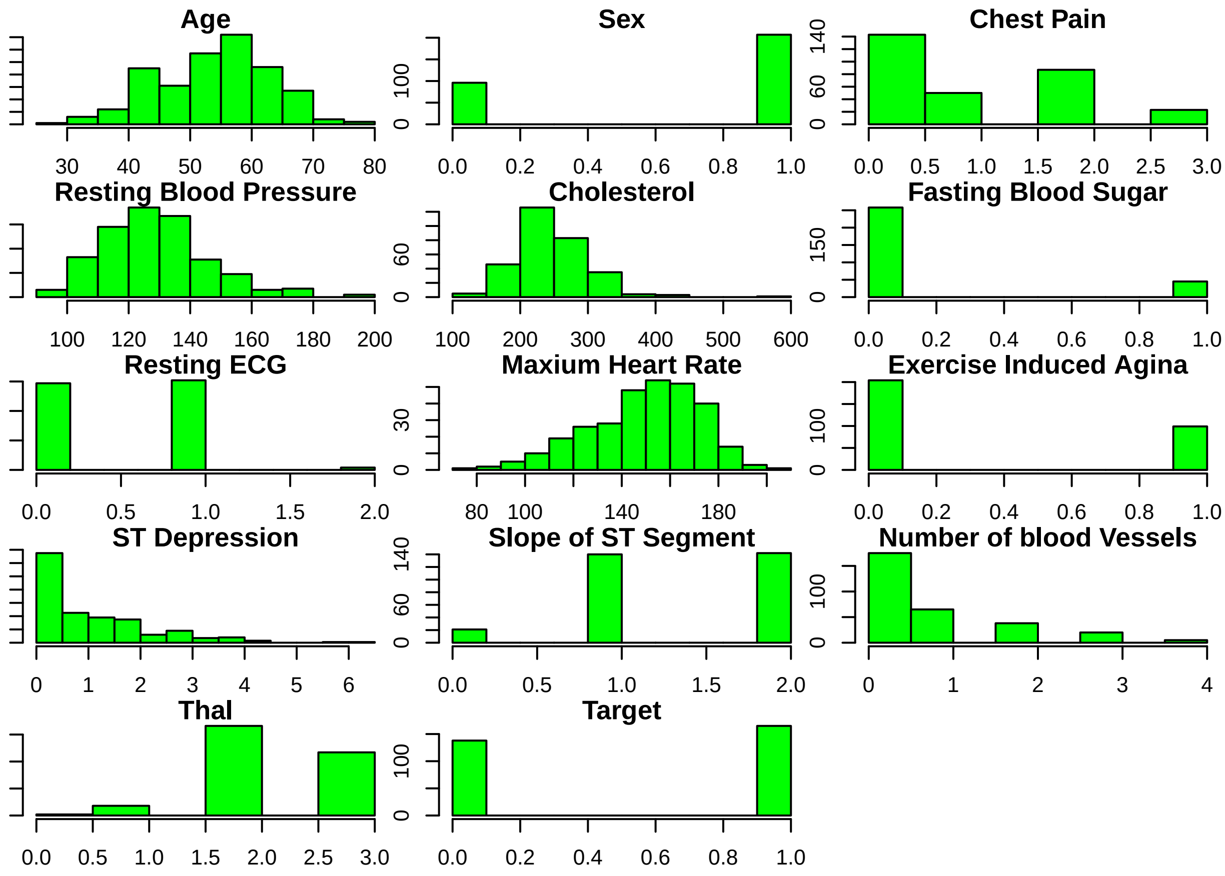
```
##      age      sex       cp trestbps     chol      fbs  restecg  thalach
##        0        0        0        0        0        0        0        0
##    exang  oldpeak    slope       ca     thal   target
##        0        0        0        0        0        0
```

The following *table 4* will show the description statistics for each of the attributes.

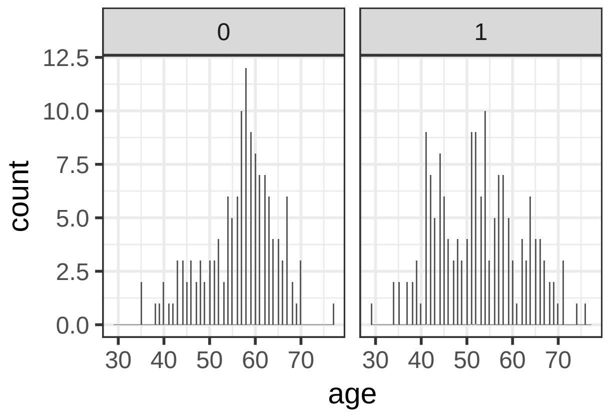
The histograms for each of attributes is shown below. From the graphs, there are more male than female in the study and the median age is between 50-60.

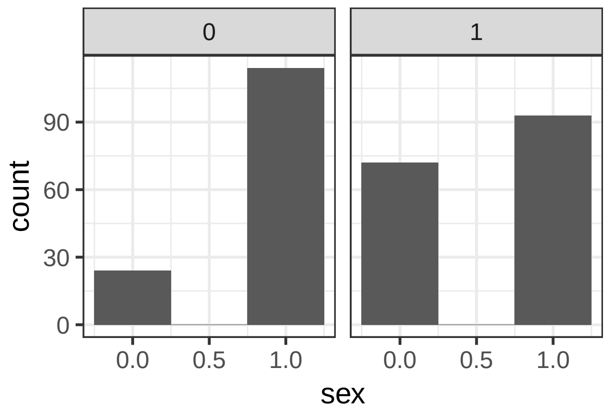
Table 4: Description Statistics for Heart Disease

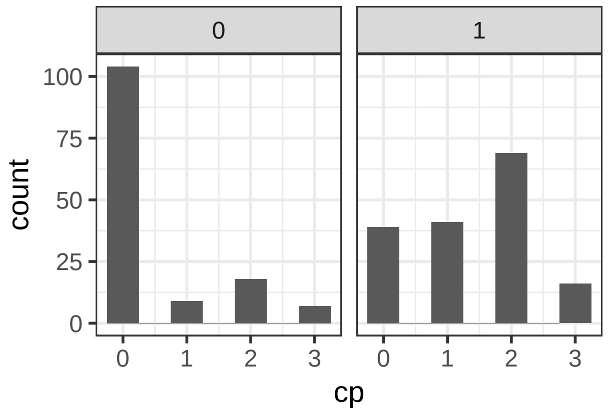
	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
age	1	303	54.3663366	9.0821010	55.0	54.5432099	10.37820	29	77.0	48.0	-0.2004632	-0.5691237	0.5217531
sex	2	303	0.6831683	0.4660108	1.0	0.7283951	0.00000	0	1.0	1.0	-0.7835174	-1.3906574	0.0267716
cp	3	303	0.9669967	1.0320525	1.0	0.8641975	1.48260	0	3.0	3.0	0.4799436	-1.2051173	0.0592899
trestbps	4	303	131.6237624	17.5381428	130.0	130.4362140	14.82600	94	200.0	106.0	0.7067170	0.8683960	1.0075400
chol	5	303	246.2640264	51.8307510	240.0	243.4855967	47.44320	126	564.0	438.0	1.1321049	4.3628409	2.9775988
fbs	6	303	0.1485149	0.3561979	0.0	0.0617284	0.00000	0	1.0	1.0	1.9670254	1.8754110	0.0204630
restecg	7	303	0.5280528	0.5258596	1.0	0.5185185	0.00000	0	2.0	2.0	0.1609167	-1.3708345	0.0302098
thalach	8	303	149.6468647	22.9051611	153.0	150.9753086	22.23900	71	202.0	131.0	-0.5321005	-0.0999265	1.3158671
exang	9	303	0.3267327	0.4697945	0.0	0.2839506	0.00000	0	1.0	1.0	0.7351959	-1.4642869	0.0269890
oldpeak	10	303	1.0396040	1.1610750	0.8	0.8555556	1.18608	0	6.2	6.2	1.2571761	1.5003397	0.0667020
slope	11	303	1.3993399	0.6162261	1.0	1.4609053	1.48260	0	2.0	2.0	-0.5032939	-0.6525221	0.0354013
ca	12	303	0.7293729	1.0226064	0.0	0.5390947	0.00000	0	4.0	4.0	1.2974762	0.7806522	0.0587472
thal	13	303	2.3135314	0.6122765	2.0	2.3580247	0.00000	0	3.0	3.0	-0.4720126	0.2517144	0.0351744
target	14	303	0.5445545	0.4988348	1.0	0.5555556	0.00000	0	1.0	1.0	-0.1780446	-1.9747849	0.0286573

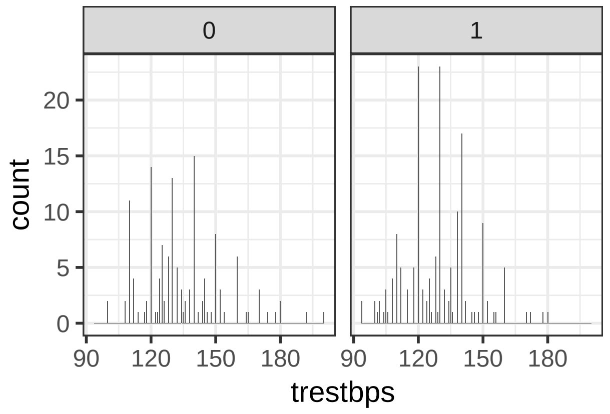


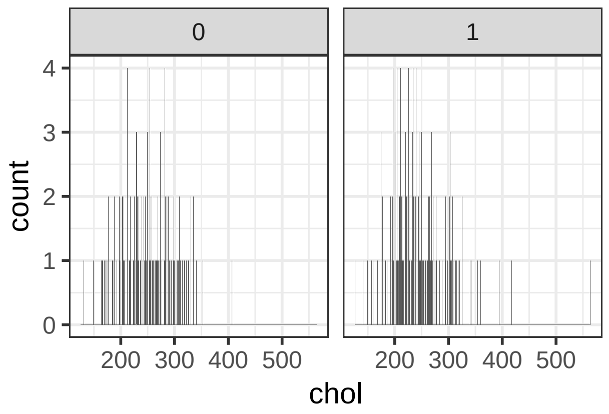
The histograms of each attributes are categorized by the presence of heart disease(1) or not(0) are also shown below. The charts can be shown that the presence of heart disease occur with same equivalent for both sexes, while without disease, there's more male than female. In addition, resting blood pressure is higher with the presence of heart disease than without. Chest pain also shown to have higher indication with disease than without. Maximum heart rate achieve is much higher in presence of heart disease as well.

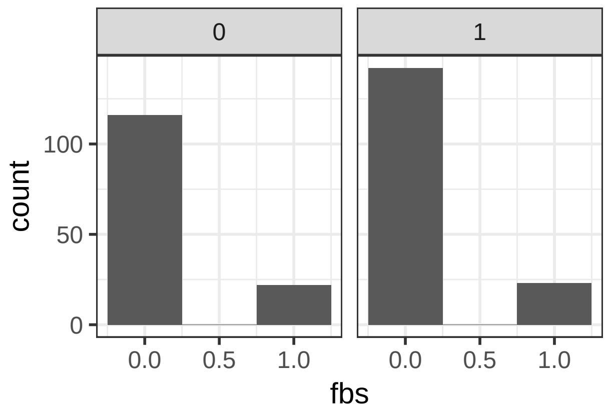

0
1
count
age

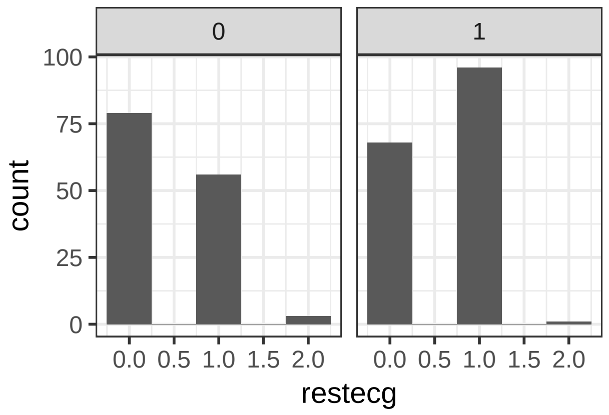

0
1
count
sex

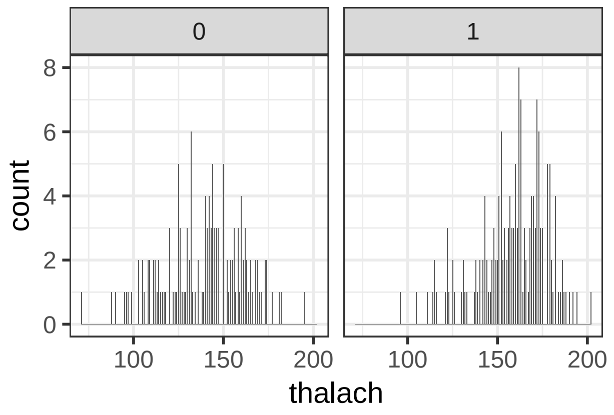

0
1
count
cp

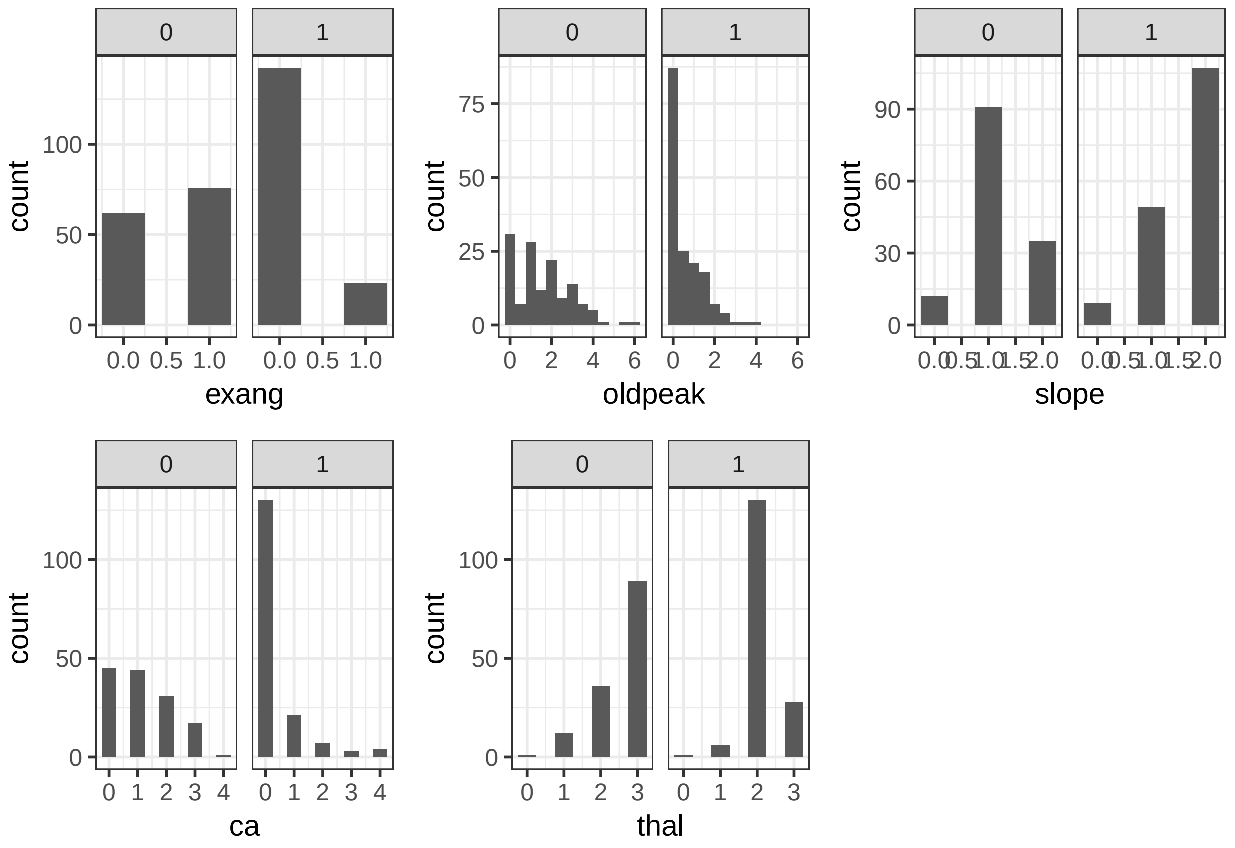

0
1
count
trestbps


0
1
count
0
1
2
3
4
200
300
400
500
chol

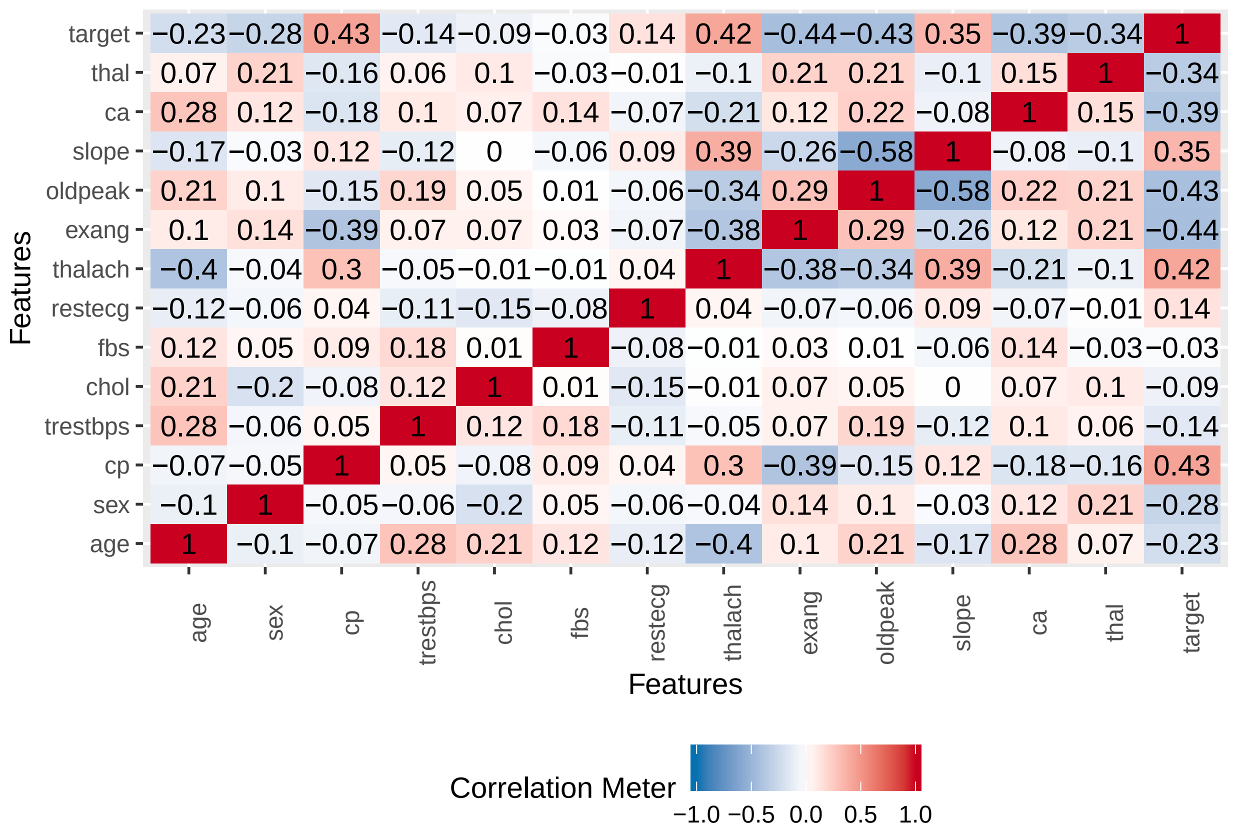

0
1
count
0
50
100
0.0
0.5
1.0
fbs

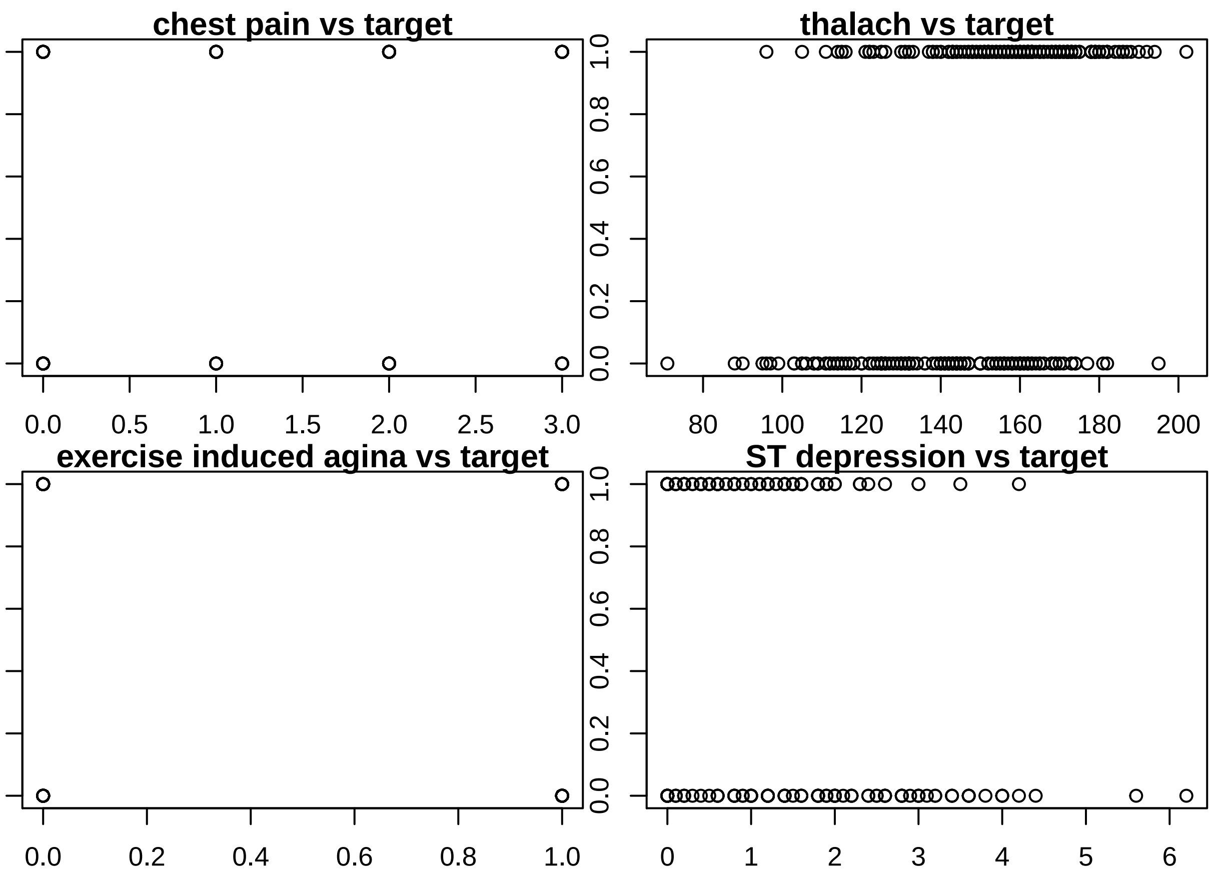

0
1
count
0
25
50
75
100
0.0
0.5
1.0
1.5
2.0
restecg


0
1
count
0
2
4
6
8
100
150
200
thalach

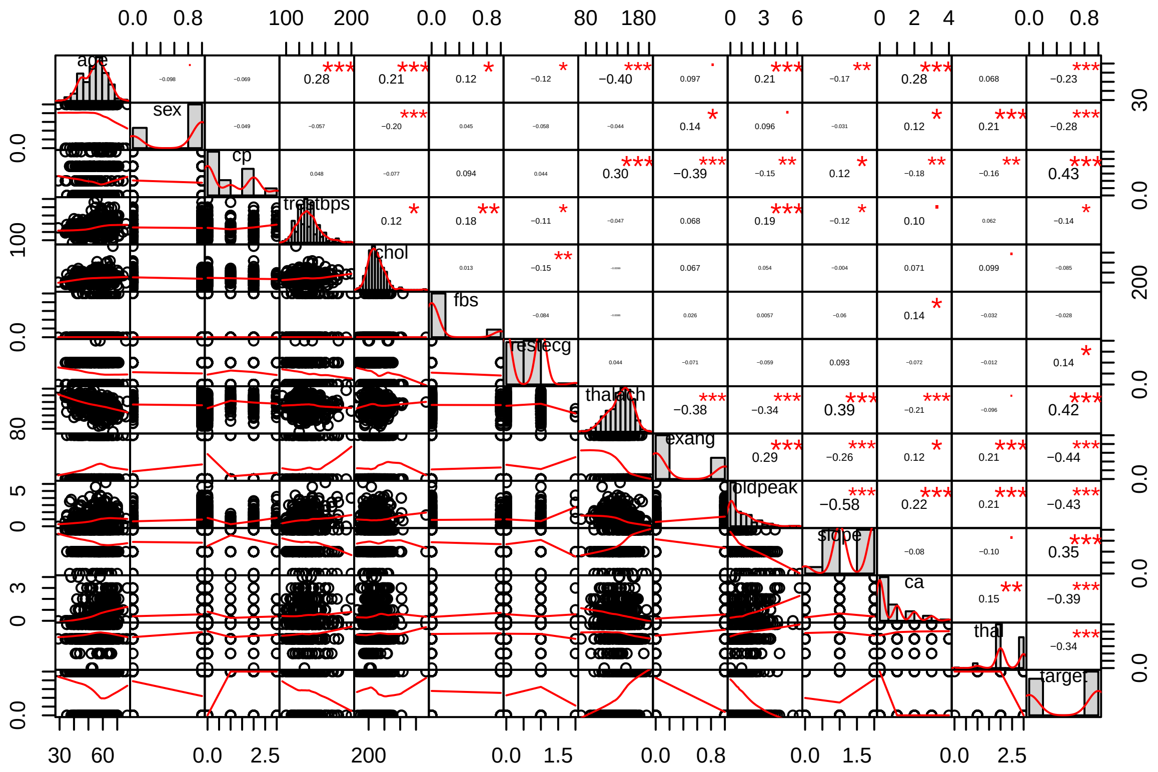


The next task is to find whether there are correlations between these variables. The plot of correlations below shows that chest pain, maximum heart rate, exercise induced angina, ST depression induced by exercise relative to rest has the highest correlation with target. Each of these attributes are plotted against target are also shown below. A summary of correlation of all the attributes are plotted and their histograms are also available.


target −0.23 −0.28 0.43 −0.14 −0.09 −0.03 0.14 0.42 −0.44 −0.43 0.35 −0.39 −0.34 1
thal 0.07 0.21 −0.16 0.06 0.1 −0.03 −0.01 −0.1 0.21 0.21 −0.1 0.15 1 −0.34
ca 0.28 0.12 −0.18 0.1 0.07 0.14 −0.07 −0.21 0.12 0.22 −0.08 1 0.15 −0.39
slope −0.17 −0.03 0.12 −0.12 0 −0.06 0.09 0.39 −0.26 −0.58 1 −0.08 −0.1 0.35
oldpeak 0.21 0.1 −0.15 0.19 0.05 0.01 −0.06 −0.34 0.29 1 −0.58 0.22 0.21 −0.43
exang 0.1 0.14 −0.39 0.07 0.07 0.03 −0.07 −0.38 1 0.29 −0.26 0.12 0.21 −0.44
thalach −0.4 −0.04 0.3 −0.05 −0.01 −0.01 0.04 1 −0.38 −0.34 0.39 −0.21 −0.1 0.42
restecg −0.12 −0.06 0.04 −0.11 −0.15 −0.08 1 0.04 −0.07 −0.06 0.09 −0.07 −0.01 0.14
fbs 0.12 0.05 0.09 0.18 0.01 1 −0.08 −0.01 0.03 0.01 −0.06 0.14 −0.03 −0.03
chol 0.21 −0.2 −0.08 0.12 1 0.01 −0.15 −0.01 0.07 0.05 0 0.07 0.1 −0.09
trestbps 0.28 −0.06 0.05 1 0.12 0.18 −0.11 −0.05 0.07 0.19 −0.12 0.1 0.06 −0.14
cp −0.07 −0.05 1 0.05 −0.08 0.09 0.04 0.3 −0.39 −0.15 0.12 −0.18 −0.16 0.43
sex −0.1 1 −0.05 −0.06 −0.2 0.05 −0.06 −0.04 0.14 0.1 −0.03 0.12 0.21 −0.28
age 1 −0.1 −0.07 0.28 0.21 0.12 −0.12 −0.4 0.1 0.21 −0.17 0.28 0.07 −0.23
age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal target
Features
Features
Correlation Meter
−1.0 −0.5 0.0 0.5 1.0



chest pain vs target
thalach vs target
exercise induced agina vs target
ST depression vs target
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1.5
2.0
2.5
3.0
80
100
120
140
160
180
200
0
1
2
3
4
5
6



Method & Analysis

The data is partitioned into training set and test set. The training set consists of 151 observations while the test set has 152. The training set will be used to train the learning machine and will use test set to predict the chance of having heart disease. From the data set, if there is a random guess that the person will have a heart disease, that guess will be *55.6 percent* correct with the squared loss of *0.2494654*. This project will use different models to improve the chances of predicting correctly. The first model will be Logistic Regression.

```
dim(train_set)
```

```
## [1] 151  14
```

```
dim(test_set)
```

```
## [1] 152  14
```

```
m<-mean(train_set$target)
m
```

```
## [1] 0.5562914
```

```
mean((m - test_set$target)^2)
```

```
## [1] 0.2494653
```

1. Logistic Regression

Logistic Regression is used to model binary dependent variable with two possible outcomes. In heart disease project, the outcome of 1 denotes heart disease presence and zero otherwise. It is a form of binomial regression.

In R programming, the use of glm() and predict can be used for logistic modeling. All thirteen predictors will be used to predict the person having heart disease. In Logistic Regression, the predictor variables can be continuous or categorical. The following code demonstrates the use of glm and predict. After the glm was run, the coefficients and intercept of the linear model are listed. If $\widehat{y}$ is greater than *0.5*, a 1(having heart disease) is assigned to y otherwise assigned a zero. The confusionMatrix showed the accuracy to be *0.7961* with confidence interval of *(0.7232, 0.857)*. Thus, our Logistic Regression model is much better than regular guesses.

```
fit <- glm(factor(target) ~ age + sex + factor(cp) + trestbps +
               chol + fbs + factor(restecg) + thalach + exang +
               oldpeak + factor(slope) + factor(ca) + factor(thal), family=binomial(link='logit'), data=
summary(fit)
```

```
##
## Call:
## glm(formula = factor(target) ~ age + sex + factor(cp) + trestbps +
##     chol + fbs + factor(restecg) + thalach + exang + oldpeak +
##     factor(slope) + factor(ca) + factor(thal), family = binomial(link = "logit"),
##     data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4416  -0.1804   0.1093   0.3778   2.9413
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.572e+00  6.194e+00  -0.415  0.67792
## age               3.334e-02  4.492e-02   0.742  0.45790
## sex              -5.583e-01  9.314e-01  -0.599  0.54892
## factor(cp)1       9.538e-01  9.742e-01   0.979  0.32753
## factor(cp)2       2.228e+00  8.987e-01   2.479  0.01317 *
## factor(cp)3       2.866e+00  1.310e+00   2.187  0.02872 *
## trestbps          1.021e-02  2.002e-02   0.510  0.61016
## chol             -8.453e-05  6.303e-03  -0.013  0.98930
## fbs              -1.449e+00  1.202e+00  -1.205  0.22801
## factor(restecg)1 -8.765e-02  6.913e-01  -0.127  0.89910
## factor(restecg)2 -2.354e+00  2.898e+00  -0.812  0.41669
## thalach           5.697e-04  1.906e-02   0.030  0.97615
## exang            -8.895e-01  7.855e-01  -1.132  0.25744
## oldpeak          -6.256e-01  4.704e-01  -1.330  0.18353
## factor(slope)1   -5.939e-01  1.231e+00  -0.482  0.62954
## factor(slope)2    8.883e-01  1.320e+00   0.673  0.50101
## factor(ca)1      -2.485e+00  7.953e-01  -3.124  0.00178 **
## factor(ca)2      -5.013e+00  1.566e+00  -3.202  0.00136 **
## factor(ca)3      -1.947e+01  1.695e+03  -0.011  0.99083
## factor(ca)4      -3.974e-01  5.802e+00  -0.068  0.94540
## factor(thal)1     2.308e+00  4.704e+00   0.491  0.62370
## factor(thal)2     2.397e+00  4.609e+00   0.520  0.60305
## factor(thal)3    -4.786e-01  4.630e+00  -0.103  0.91768
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 207.41  on 150  degrees of freedom
## Residual deviance:  73.90  on 128  degrees of freedom
## AIC: 119.9
##
## Number of Fisher Scoring iterations: 16
```

```
fit$coef
```

```
##      (Intercept)              age              sex      factor(cp)1
##    -2.572445e+00     3.334204e-02    -5.582627e-01     9.538223e-01
##      factor(cp)2      factor(cp)3         trestbps             chol
##     2.227896e+00     2.866261e+00     1.020778e-02    -8.453105e-05
##              fbs factor(restecg)1 factor(restecg)2          thalach
##    -1.449398e+00    -8.765385e-02    -2.353597e+00     5.696586e-04
##            exang          oldpeak   factor(slope)1   factor(slope)2
##    -8.895041e-01    -6.255798e-01    -5.939072e-01     8.883184e-01
##      factor(ca)1      factor(ca)2      factor(ca)3      factor(ca)4
##    -2.484864e+00    -5.013202e+00    -1.946965e+01    -3.973577e-01
##    factor(thal)1    factor(thal)2    factor(thal)3
##     2.308023e+00     2.396940e+00    -4.785797e-01
```

```
y_hat <- predict(fit, test_set)

y <- ifelse(y_hat > 0.5, 1, 0)
confusionMatrix(data=factor(y), factor(test_set$target))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 59 19
##          1 12 62
##
##                Accuracy : 0.7961
##                  95% CI : (0.7232, 0.857)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : 1.306e-11
##
##                   Kappa : 0.5928
##  Mcnemar's Test P-Value : 0.2812
##
##             Sensitivity : 0.8310
##             Specificity : 0.7654
##          Pos Pred Value : 0.7564
##          Neg Pred Value : 0.8378
##              Prevalence : 0.4671
##          Detection Rate : 0.3882
##    Detection Prevalence : 0.5132
##       Balanced Accuracy : 0.7982
##
##        'Positive' Class : 0
##
```

2. Logistic Regression with K-fold Cross Validation

Since regular logistic regression can produce the accuracy of 0.7961, the question to ask is 'Is this accuracy accurate?' K-Fold Cross Validation is applied to Logistic Regression to to validate the accuracy of the model.

Cross validation is used in machine learning to estimate the skill of a machine learning model on unseen data. K-Fold Cross-Validation is a resampling procedure used to evaluate machine learning models. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. In this project, k is set at 10, thus becoming 10 fold cross validation. This means that there will be 10 samples using 10% of the observation each. The code and its output to perform on K-Fold Cross Validation is shown below. Notice that the accuracy is now *.8882* with a smaller confidence interval, which is an improvement and more reliable.

```r
control <- trainControl(method='cv', number=10, savePredictions=T)

train_glm_cv <- train(factor(target) ~ age + sex + factor(cp) + trestbps + chol + fbs
               + factor(restecg) + thalach + exang + oldpeak + factor(slope)
               + factor(ca) + factor(thal), data=heartDisease, trControl=control, method='glm', family=

print(train_glm_cv)
```

```
## Generalized Linear Model
##
## 303 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 274, 272, 273, 273, 273, 273, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8302484  0.6536535
```

```r
glm_pred <- predict(train_glm_cv, test_set, type = "raw")
cm_lr <- confusionMatrix(data=factor(glm_pred), factor(test_set$target))
cm_lr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 59  5
##          1 12 76
##
##                Accuracy : 0.8882
##                  95% CI : (0.827, 0.9335)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.774
##  Mcnemar's Test P-Value : 0.1456
##
##             Sensitivity : 0.8310
##             Specificity : 0.9383
##          Pos Pred Value : 0.9219
##          Neg Pred Value : 0.8636
##              Prevalence : 0.4671
##          Detection Rate : 0.3882
##    Detection Prevalence : 0.4211
##       Balanced Accuracy : 0.8846
##
##        'Positive' Class : 0
##
```

```
###confusionMatrix(data=factor(glm_pred), factor(test_set$target))

varImp(train_glm_cv)
```

```
## glm variable importance
##
##   only 20 most important variables shown (out of 22)
##
##                     Overall
## `factor(ca)1`       100.00
## `factor(ca)2`        96.42
## `factor(cp)2`        84.30
## `factor(cp)3`        74.17
## sex                  71.82
## `factor(ca)3`        51.13
## trestbps             46.20
## exang                35.05
## thalach              34.26
## oldpeak              33.01
## `factor(cp)1`        29.60
## `factor(restecg)1`   21.39
## age                  19.99
## chol                 18.00
## `factor(thal)1`      17.32
## `factor(thal)2`      15.64
## `factor(slope)1`     14.88
## fbs                  11.95
## `factor(ca)4`        11.46
## `factor(slope)2`     11.24
```

Notice the variable importance in the previous logistic regression model. The output from the code showed that ca(number of blood vessels), chest pain(cp) and sex are the most important variables.

3. K-nearest Neighbors(KNN)

K-Nearest Neighbors, or KNN, is the most used algorithm in machine learning. Its purpose is to separate the data points into several classes to predict the classification of the new sample point. KNN is mostly use in real-life scenarios because of its non-parametric characteristics. It implies that it does not make any assumptions about the distribution of the data. In R programming, the functions *knn3* and *predict* to train the model and make the prediction of heart disease using the default value of *k=1*. We notice that the accuracy is of *0.6382*, which is worse than our Logistic Regression. We need to find the appropriate *k* so that it will provide an optimum accuracy.

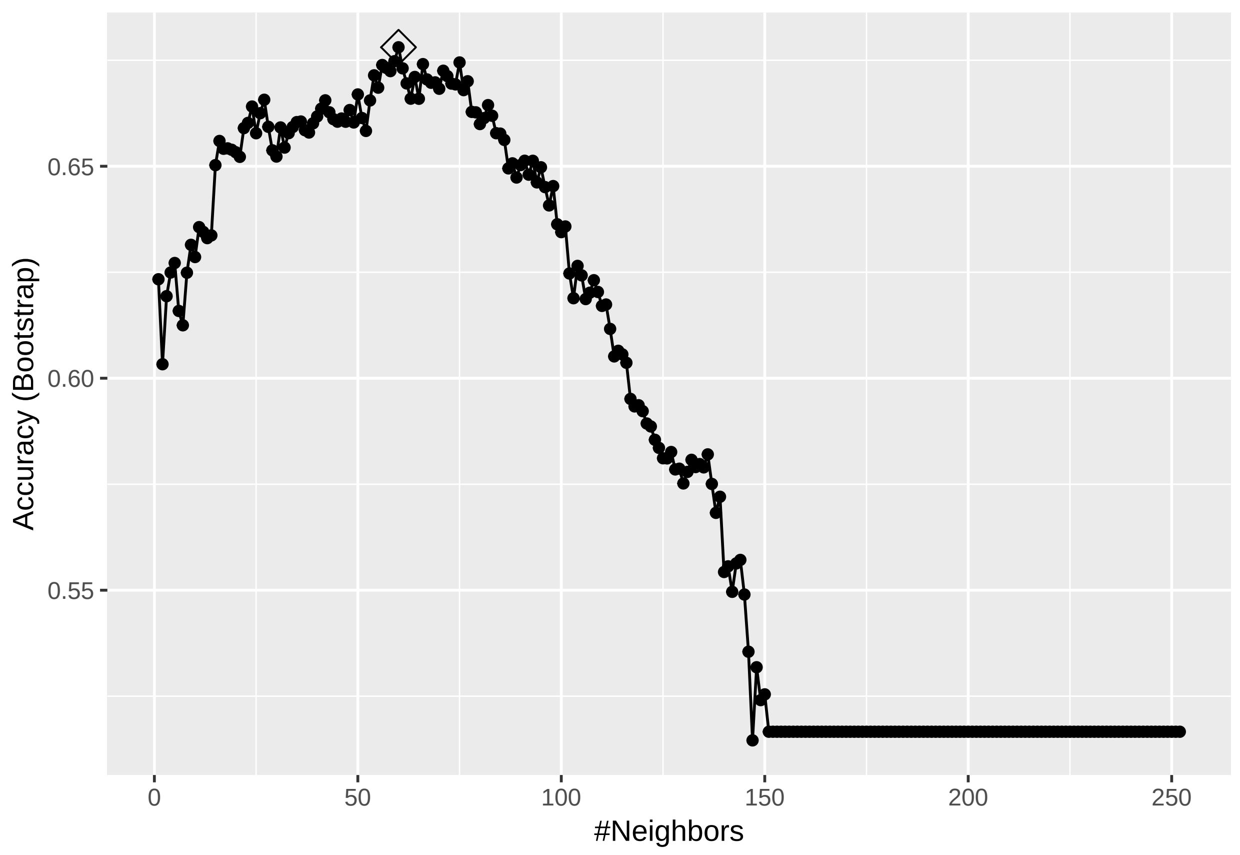
```
knn_fit <- knn3(factor(target) ~ age + sex + factor(cp) + trestbps +
                  chol + fbs + factor(restecg) + thalach + exang +
                  oldpeak + factor(slope) + factor(ca) + factor(thal), data = train_set)
y_hat_knn <- predict(knn_fit,test_set, type = "class")
confusionMatrix(data=factor(y_hat_knn), factor(test_set$target))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  0  1
##          0 41 25
##          1 30 56
##
##                Accuracy : 0.6382
##                  95% CI : (0.5564, 0.7144)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : 0.005586
##
##                   Kappa : 0.27
##  Mcnemar's Test P-Value : 0.589639
##
##             Sensitivity : 0.5775
##             Specificity : 0.6914
##          Pos Pred Value : 0.6212
##          Neg Pred Value : 0.6512
##              Prevalence : 0.4671
##          Detection Rate : 0.2697
##    Detection Prevalence : 0.4342
##       Balanced Accuracy : 0.6344
##
##        'Positive' Class : 0
##
```

The following code that will find the maximum k which provide the best model, using tuneGrid parameter in the train function. To make the prediction more accurate, a reduction in the predictor variables because the correlations table noted that chest pain, maximum heart rate, exercise induced angina, and old ST peak on ECG have the highest correlation with target. The code is shown below.

```r
train_knn <- train(factor(target) ~ sex + factor(cp) + thalach + exang +
                     oldpeak, method = "knn",
                   data = train_set, tuneGrid = data.frame(k = seq(1,252)))
ggplot(train_knn, highlight= TRUE)
```



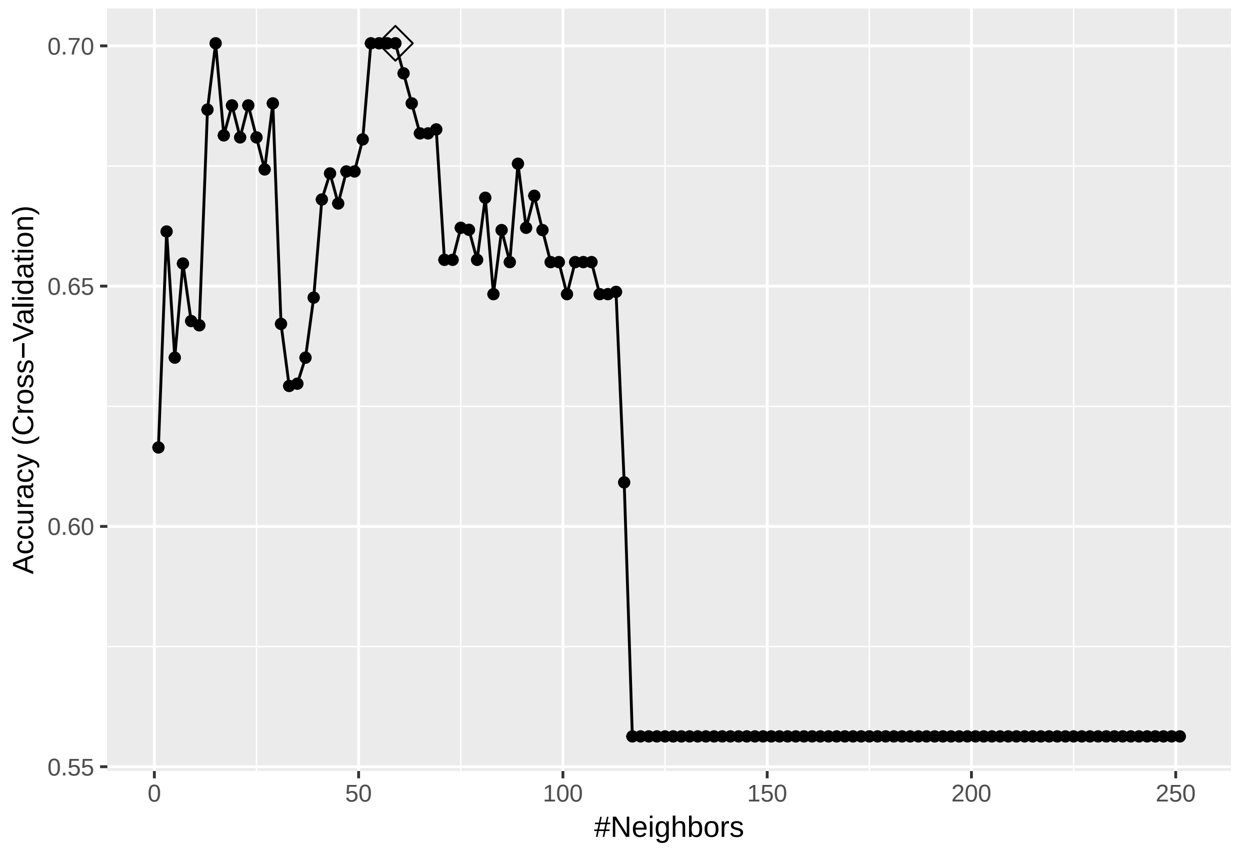
The best accuracy that it could achieve is *0.7105* with confidence interval of (0.6315, 0.7811). In addition, both the sensitivity and specificity are much lower than logistic model previously.

```
knn_pred <- predict(train_knn, test_set, type = "raw")
confusionMatrix(data=factor(knn_pred), factor(test_set$target))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47 20
##          1 24 61
##
##                Accuracy : 0.7105
##                  95% CI : (0.6315, 0.7811)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : 5.776e-06
##
##                   Kappa : 0.4165
##  Mcnemar's Test P-Value : 0.6511
##
##             Sensitivity : 0.6620
##             Specificity : 0.7531
##          Pos Pred Value : 0.7015
##          Neg Pred Value : 0.7176
##              Prevalence : 0.4671
##          Detection Rate : 0.3092
##    Detection Prevalence : 0.4408
##       Balanced Accuracy : 0.7075
##
##        'Positive' Class : 0
##
```

If K-Fold cross validation is applied on KNN, the accuracy is about the same with 0.7171, which validates our KNN model.

```
control <-trainControl(method="cv", number = 10, savePredictions=T)
train_knn_cv <- train(factor(target) ~ sex + factor(cp) + thalach + exang +
                      oldpeak, method = "knn",
                      data = train_set, tuneGrid = data.frame(k = seq(1,252,2)),
                      trControl = control)
ggplot(train_knn_cv, highlight=TRUE)
```



```
knn_pred <- predict(train_knn_cv, test_set, type = "raw")
cm_knn <- confusionMatrix(data=factor(knn_pred), factor(test_set$target))
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47 19
##          1 24 62
##
##                Accuracy : 0.7171
##                  95% CI : (0.6384, 0.7871)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : 2.595e-06
##
##                   Kappa : 0.4293
##  Mcnemar's Test P-Value : 0.5419
##
##             Sensitivity : 0.6620
##             Specificity : 0.7654
##          Pos Pred Value : 0.7121
##          Neg Pred Value : 0.7209
##              Prevalence : 0.4671
##          Detection Rate : 0.3092
##    Detection Prevalence : 0.4342
##       Balanced Accuracy : 0.7137
##
##        'Positive' Class : 0
##
```

4. Classification(decision) trees with Cross Validation

Classification(Decision) tree is commonly used in machine learning and data mining. It is used in where the outcome is categorical. It is a simple representation for classifying examples. The tree can be learned by splitting the source into subsets based on the attribute value test. The process then repeated on each derived subset, which is called recursive partitioning. The model predict the value of the target based on various variables inputs. Each interior node corresponds to one of the variables. Each leaf represents a value of the target variable.

Classification tree uses *Gini Index* and *Entropy* to decide where to partition the tree. Gini Index is defined as

$$Gini = \sum_{k=1} \widehat{p}_{m,k}(1 - \widehat{p}_{m,k})$$

and Entropy is defined as

$$Entropy = -\sum_{k=1} \widehat{p}_{m,k} log(\widehat{p}_{m,k})$$

with 0 x log(0) defined as 0 and $\widehat{p}_{m,k}$ as the proportion of observations in partition m that are of class k. If K=0, then both the Gini Index and Entropy are 0.

Let's examine how classification performs in R programming compared to other learning models. The code start to train the model using the train function with method "rpart". The accuracy is shown to be 0.7829, much higher than KNN model, but still lower than our Logistic Regression.

```r
control <- trainControl(method = "cv",
             number = 10, savePredictions=T)

train_rpart <- train(factor(target) ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = c(0.01)),
                     data = train_set, trControl = control)

#plot(train_rpart)

rpart_pred <- predict(train_rpart, test_set)
cm_ct <- confusionMatrix(data=factor(rpart_pred), factor(test_set$target))
cm_ct
```

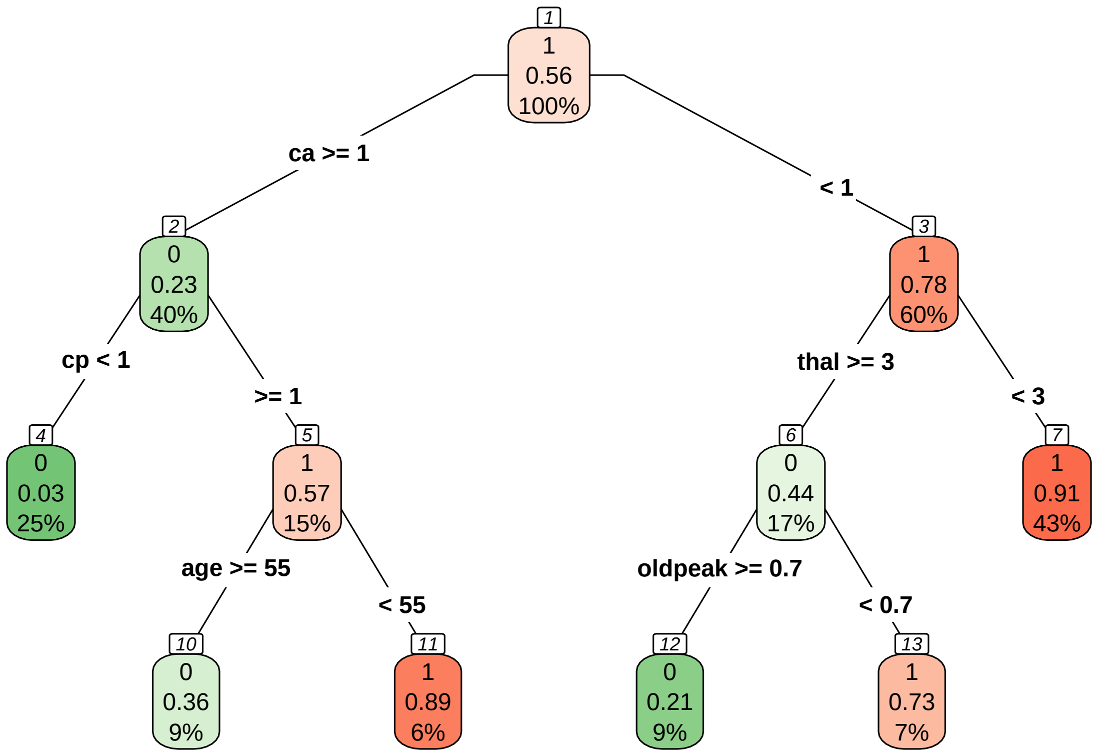
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 55 17
##          1 16 64
##
##                Accuracy : 0.7829
##                  95% CI : (0.7088, 0.8456)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : 1.426e-10
##
##                   Kappa : 0.5643
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7746
##             Specificity : 0.7901
##          Pos Pred Value : 0.7639
##          Neg Pred Value : 0.8000
##              Prevalence : 0.4671
##          Detection Rate : 0.3618
##    Detection Prevalence : 0.4737
##       Balanced Accuracy : 0.7824
##
##        'Positive' Class : 0
##
```

```r
varImp(train_rpart)
```

```
## rpart variable importance
##
##          Overall
## thal     100.000
## cp        92.253
## oldpeak   91.672
## exang     77.645
## ca        76.058
## thalach   37.507
## slope     29.838
## age       28.972
## trestbps   7.841
## chol       5.968
## restecg    0.000
## fbs        0.000
## sex        0.000
```

The decision tree is shown below. From the tree, it is much easier to interpret and easer to visualize. The tree first split with predictor *ca*. In addition, the variable importance showed that Thal, chest pain, and old peak on ECG are the most important variables.

```r
##plot(train_rpart$finalModel, margin = 0.1)
##text(train_rpart$finalModel, cex = 0.65)
rpart.plot(train_rpart$finalModel, type = 4, fallen.leaves = FALSE, box.palette = "GnRd", nn=TRUE)
```



5. Random Forest with Cross Validation

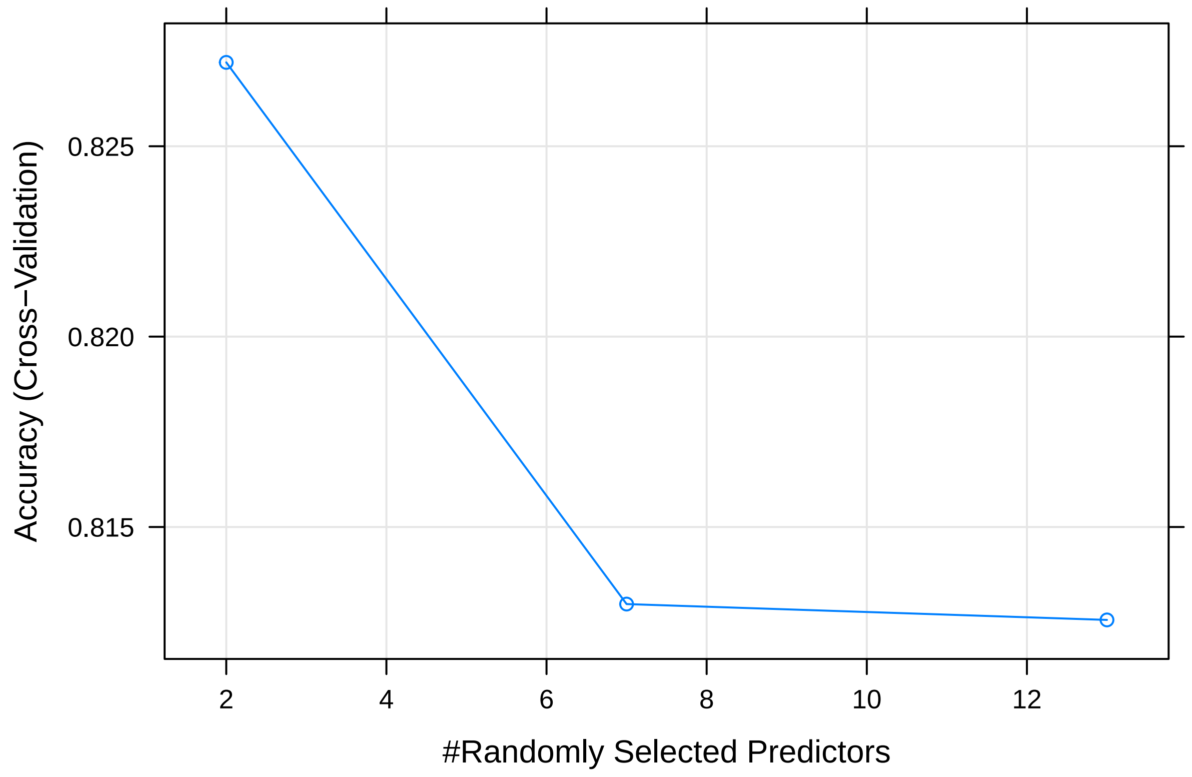
Decision tree has its limitations. It is harder to train than KNN or regression model. It can also lead to over-train due to its recursive properties. To overcome these limitations, Random Forest is introduced. The goal of Random Forest is to improve prediction performance and instability by averaging the decision trees. The trick is to use bragging. This can be done by building decision tress $T_1, T_2, .., T_B$ using the training set. For every observation in test set, form a prediction $\widehat{y}_j$ using T_j.

First, create a bootstrap training by sampling N observation from training set with replacement. Then create a decision tree from bootstrap training set. Below is the code to achieve that and its accuracy of *0.8421* with a 95% confidence interval of *(0.7742, 0.8961)*. This accuracy is very close to logistic regression. The plot of the accuracy and the list of variable importance is also displayed. Notice that the number of blood vessels(ca), old peak and maximum heart rate are the most important variables.

```
control <- trainControl(method = "cv",
                        number = 10, savePredictions=T)

train_rf <- train(factor(target) ~ .,
                  method = "rf",
                  data = train_set, trControl = control)

plot(train_rf)
```



```r
pred_rf <- predict(train_rf, test_set)
cm_rf <- confusionMatrix(data = factor(pred_rf),factor(test_set$target))
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 55  8
##          1 16 73
##
##                Accuracy : 0.8421
##                  95% CI : (0.7742, 0.8961)
##     No Information Rate : 0.5329
##     P-Value [Acc > NIR] : 8.261e-16
##
##                   Kappa : 0.6806
##  Mcnemar's Test P-Value : 0.153
##
##             Sensitivity : 0.7746
##             Specificity : 0.9012
##          Pos Pred Value : 0.8730
##          Neg Pred Value : 0.8202
##              Prevalence : 0.4671
##          Detection Rate : 0.3618
##    Detection Prevalence : 0.4145
##       Balanced Accuracy : 0.8379
##
##        'Positive' Class : 0
##
```

```
## display the variable importance
varImp(train_rf)
```

```
## rf variable importance
##
##          Overall
## ca       100.000
## oldpeak   80.651
## thalach   72.122
## thal      70.069
## cp        67.028
## age       59.602
## chol      57.276
## trestbps  49.702
## slope     36.528
## exang     30.542
## sex        4.936
## restecg    4.277
## fbs        0.000
```

Results

From the table 5 shown below, Logistic Regression achieve the highest accuracy followed by Random Forest. All of the modeling methods beat out the regular guesses. Furthermore, the sensitivity and specificity are high with logistic regression as compared to other models.

Table 5: Accuracy Results From Various Methods.

methods	Accuracy.Values	sensitivity	specificity
Regular Guesses	0.5562914	–	–
Logistic Regression with K-Fold CV	0.8881579	0.830985915492958	0.938271604938272
KNN with K-Fold CV	0.7171053	0.661971830985915	0.765432098765432
Classification Tree with K-Fold CV	0.7828947	0.774647887323944	0.790123456790123
Random Forest with K-Fold CV	0.8421053	0.774647887323944	0.901234567901235

Part of the issue with low accuracies in KNN could be due to small amount of dataset. It only contained 303 observations, as compared to other data for machine learning which could range from thousands to millions. If there was more data available, the KNN model, classification tree and random forest accuracies, sensitivities, and specificities will be much more accurate.

Conclusion

This project touched on multiple model for machine learning. It ended up with Logistic Regression being the best model to predict heart disease. It was interesting project from a medical point of view because health care providers could use this modeling to help patients in predicting their chances of having heart disease in the future. This will allow both patients and doctors to discuss their options of treatments and preventions. The set back of this project is that it does not predict the type of heart disease the patient will have, since each treatment is slightly different.

For future works, if there is more data available, the modeling methods would be run again to see if KNN will beat out logistic regression. In addition, if there are data that categorize the type of heart disease, it will be valuable to health care providers to come up with treatment plan for the patients.