# Heart Disease Detection Model

Vimal Thomas Joseph

## Introduction

The objective of this project is to detect if a person has heart disease or not given the list of features. The focus is to analyze the dataset for correlation, anomalies, biases and relationships present in the data and then build an array of machine learning models using a list of suitable features that can assist predicting if a person has the heart disease or not.

## The Model generation code consists of the following modules

1) Library Management
2) Function Loading
3) Data Preparation
4) Data Analysis and Visualization
5) Scaling, PCA & Cluster Analysis
6) Creation of training and testing datasets
7) Machine Learning Model generation
8) Creation of Ensemble

## Data Preparation

For the heart disease detection project, the data is loaded from https://archive.ics.uci.edu/ml/machine-learning-databases/

The following section describes the characteristics of the feature variables.

1. age - age in years

2. sex - (1 = male; 0 = female)

3. cp -chest pain type 1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic

4. trestbps - resting blood pressure (in mm Hg on admission to the hospital)
5. ch - serum cholestoral in mg/dl

6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg - resting electrocardiographic results
   - 0: normal, 1: having ST-T wave abnormality, 2: showing probable or definite left ventricular hypertrophy
8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)

10. oldpeak - ST depression induced by exercise relative to rest
11. slope - the slope of the peak exercise ST segment -1: upsloping,2: flat, 3: downsloping
12. ca - number of major vessels (0-3) colored by fluoroscopy

13. thal - 3 = normal; 6 = fixed defect; 7 = reversable defect
14. num - num: diagnosis of heart disease (angiographic disease status) Value 0: < 50% diamep ter narrowing Value 1: > 50% diameter narrowing

# Data Analysis and Visualizations.

Let us look at the data we have downloaded and added column names to.

```
## # A tibble: 6 x 14
##     age   sex    cp trestbps  chol   fbs restecg thalach exang oldpeak slope
##   <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1    63     1     1      145   233     1       2     150     0     2.3     3
## 2    67     1     4      160   286     0       2     108     1     1.5     2
## 3    67     1     4      120   229     0       2     129     1     2.6     2
## 4    37     1     3      130   250     0       0     187     0     3.5     3
## 5    41     0     2      130   204     0       2     172     0     1.4     1
## 6    56     1     2      120   236     0       0     178     0     0.8     1
## # ... with 3 more variables: ca <chr>, thal <chr>, num <dbl>
```

The next step is to find out and replace any missing values. Based on further evaluation, there are 6 rows that contain missing values.

```
##          [,1]  [,2]  [,3]  [,4]  [,5]  [,6] 
## age      53    52    43    52    58    38   
## sex      0     1     1     1     1     1    
## cp       3     3     4     4     2     3    
## trestbps 128   138   132   128   125   138  
## chol     216   223   247   204   220   175  
## fbs      0     0     1     1     0     0    
## restecg  2     0     2     0     0     0    
## thalach  115   169   143   156   144   173  
## exang    0     0     1     1     0     0    
## oldpeak  0     0     0.1   1     0.4   0    
## slope    1     1     2     2     2     1    
## ca       "0.0" "?"   "?"   "0.0" "?"   "?"  
## thal     "?"   "3.0" "7.0" "?"   "7.0" "3.0"
## num      0     0     1     2     0     0
```

Let us replace them with NAs which would ease our further replacement options.

Now, to fill NAs with suitable replacement value, let us look at the columns with missing values. - ca and thal.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##  0.0000  0.0000  0.0000  0.6722  1.0000  3.0000       4
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##   3.000   3.000   3.000   4.734   7.000   7.000       2
```

The missing values are replaced with the median values of the impacted columns. In this case, ca column gets a value 0 an thal gets a values 3.

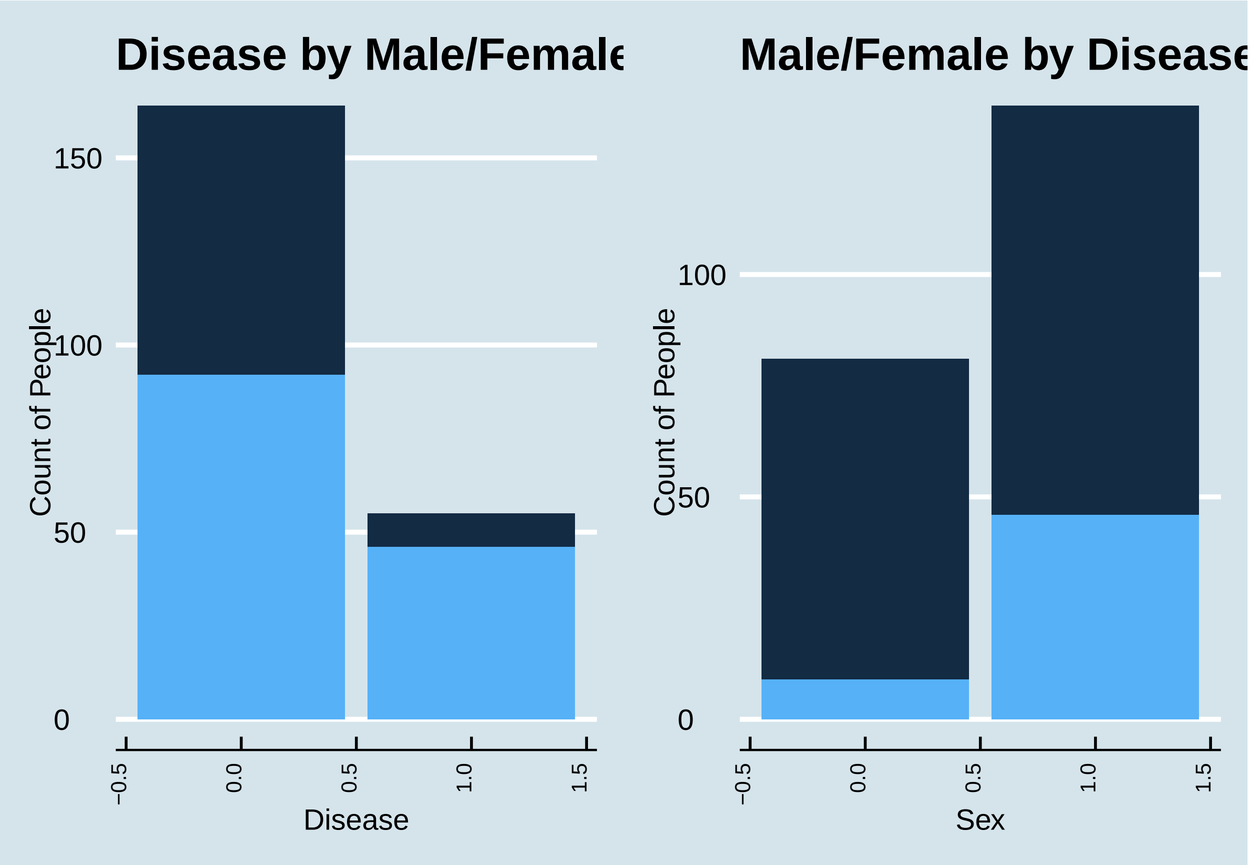
Insight Gained: After replacing the values, we should note that the median did not change for both columns as expected.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.3836  1.0000  3.0000
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   3.000   4.183   7.000   7.000
```

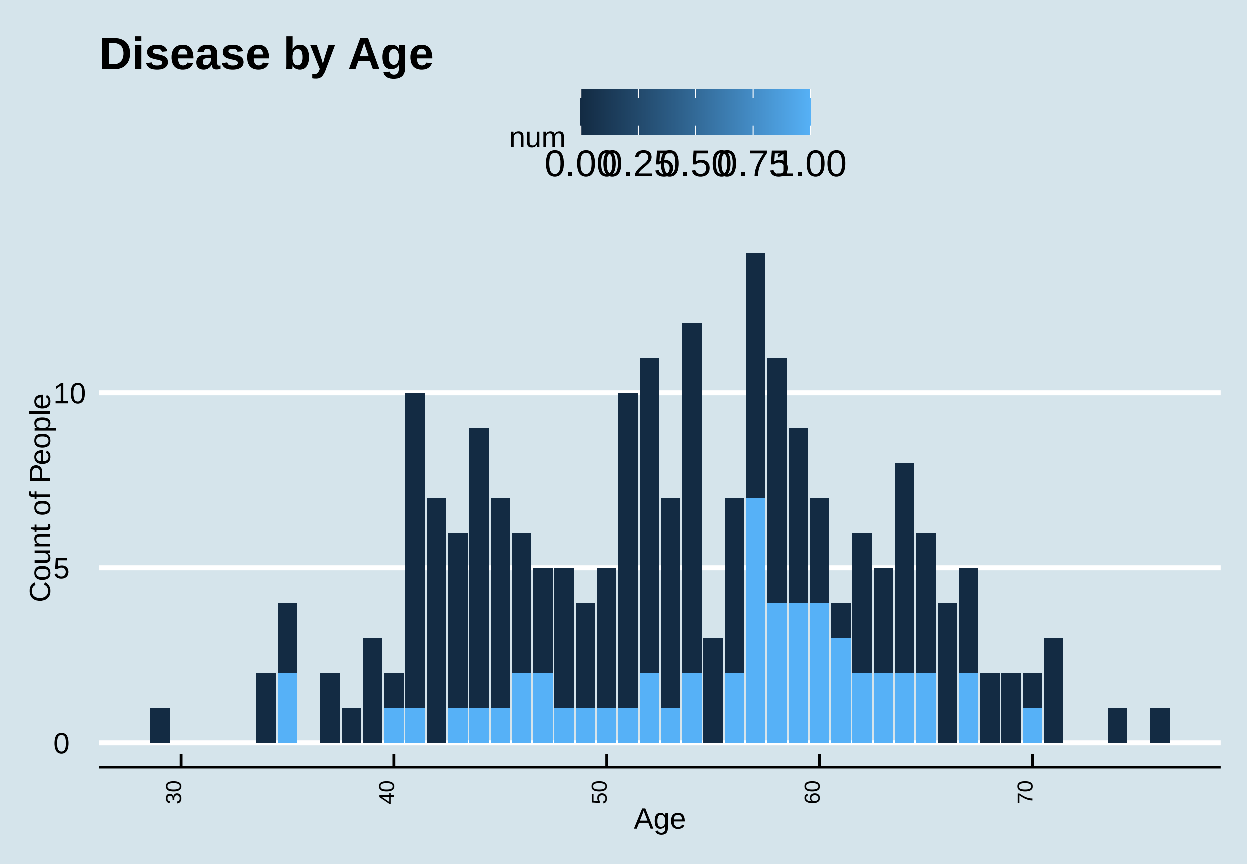
## Data Visualization

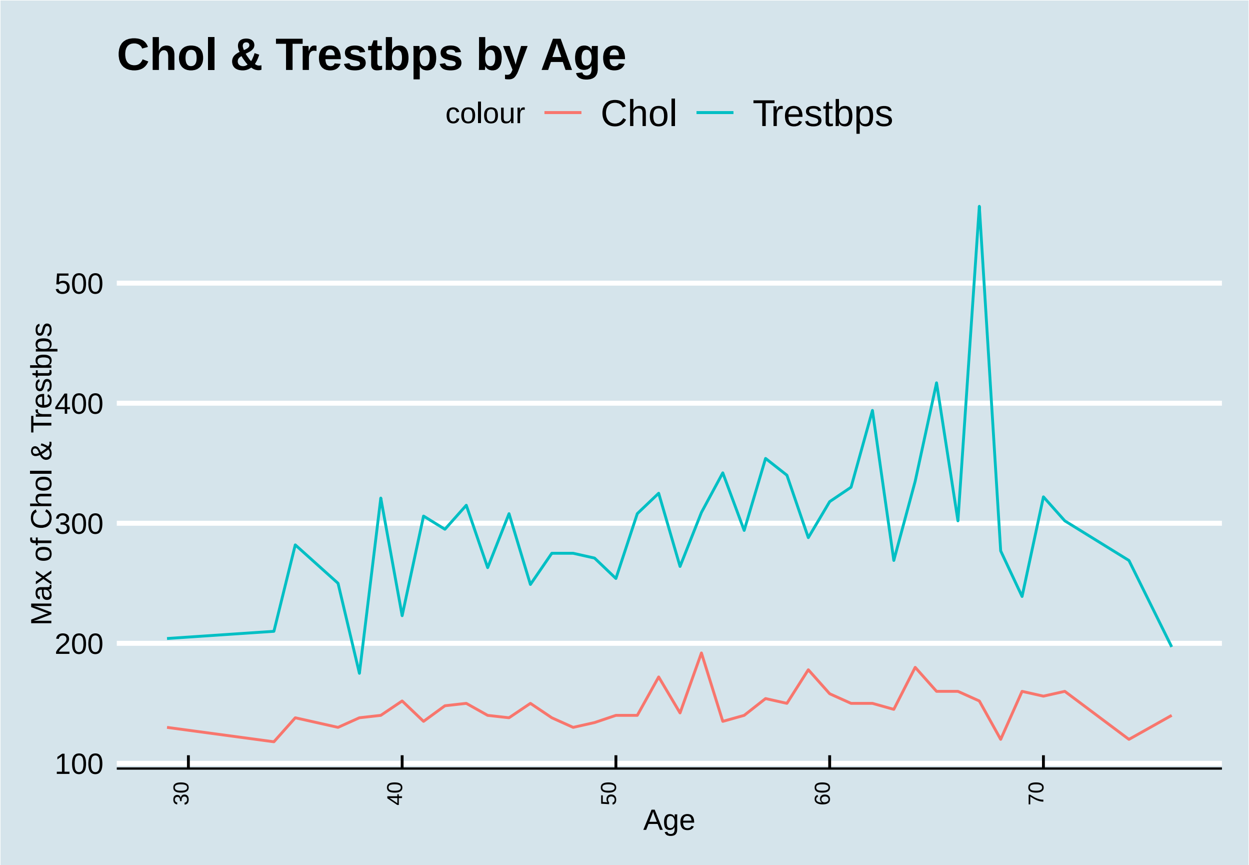
Let us create a side by side graph of disease (0 - no disease,1 - disease) and sex(Male,Female).



Insight Gained: Based on both the graphs, males seem to develop heart disease condition more than females.

Let us create a graph of disease condition based on age. Also, it would be useful to create another graph by plotting cholestoral and blood pressure by age.


Disease by Age
num
0.00
0.25
0.50
0.75
1.00
Count of People
10
5
0
30
40
50
60
70
Age



Insight Gained: It is very evident that the heart disease condition begins to develop from late thirties and is at peak between 55 and 65. Also based on the second graph, we could note that the pattern of cholestoral and blood pressure increases with age as expected.

## Scaling, PCA and Cluster Analysis

The next important step in the creation of a machine learning algorithm is to verify how the features are correlated among themselves. Do they have a feature that needs to be scaled down in order to provide an equal importance in the prediction process?

Before progressing further, the cleansed data is converted into a set of feature and predicted variables.

creation of feature (x) and outcome (y) variables to predict y_hat. As taught in the course, features are created as vector matrix and outcome data is created as a y factor

```
knitr::opts_chunk$set(echo = TRUE)

hd_m<-as.matrix(hd_wrangled)

hd_x<-hd_m[,1:13]
hd_y<-as.factor(hd_m[,14])
```

First step is to look at the features if any of the feature has higher variance proportions when compared to other features. To do that, let us look at the summary of the features to see what values we deal with.

```r
summary(hd_x)
```

```
##       age             sex               cp           trestbps    
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0  
##  1st Qu.:46.00   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:120.0  
##  Median :54.00   Median :1.0000   Median :3.000   Median :130.0  
##  Mean   :53.29   Mean   :0.6301   Mean   :2.932   Mean   :130.3  
##  3rd Qu.:60.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0  
##  Max.   :76.00   Max.   :1.0000   Max.   :4.000   Max.   :192.0  
##       chol            fbs            restecg          thalach     
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 88.0  
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:144.0  
##  Median :239.0   Median :0.0000   Median :0.0000   Median :159.0  
##  Mean   :244.3   Mean   :0.1233   Mean   :0.9178   Mean   :155.3  
##  3rd Qu.:269.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:170.5  
##  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.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.4000   Median :1.000   Median :0.0000  
##  Mean   :0.2192   Mean   :0.6918   Mean   :1.466   Mean   :0.3836  
##  3rd Qu.:0.0000   3rd Qu.:1.2000   3rd Qu.:2.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :4.2000   Max.   :3.000   Max.   :3.0000  
##       thal      
##  Min.   :3.000  
##  1st Qu.:3.000  
##  Median :3.000  
##  Mean   :4.183  
##  3rd Qu.:7.000  
##  Max.   :7.000  
```

It looks like age, trestbps, chol and thalach have values in 100s while the rest of the features have values in 1s.

This could create a broader variance differences when preparing the data for model training. As taught in the course, let us see if scaling this dataset makes any difference in their variance proportions.

One way to find out if scaling is required, is by conducting PCA. Let us compute Principal Component Analysis - PCA for scaled vs unscaled dataset of same features.

```r
unscaled <- prcomp(hd_x)
scaled <- prcomp(hd_x, scale = TRUE)
```

PCAs before scaling:

```r
summary(unscaled)
```

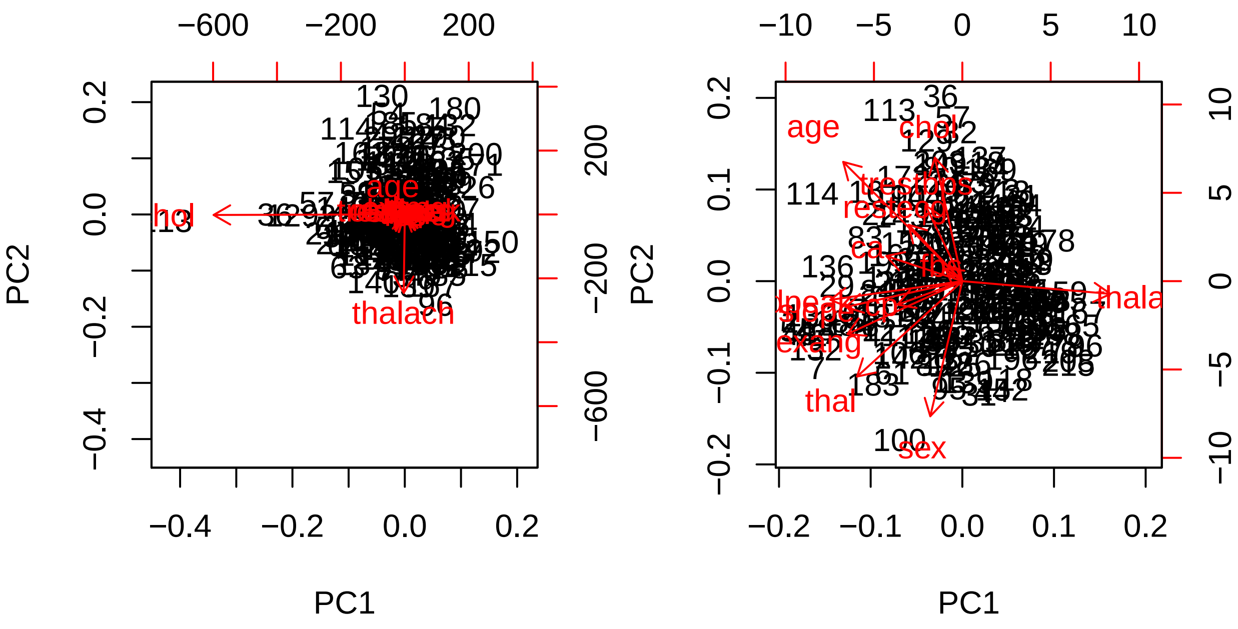
```
## Importance of components:
##                            PC1     PC2      PC3     PC4     PC5    PC6     PC7
## Standard deviation     50.6663 21.3047 16.77631 7.27992 1.76935 0.9966 0.93374
## Proportion of Variance  0.7635  0.1350  0.08371 0.01576 0.00093 0.0003 0.00026
## Cumulative Proportion   0.7635  0.8985  0.98222 0.99798 0.99891 0.9992 0.99947
##                            PC8     PC9    PC10    PC11    PC12    PC13
## Standard deviation     0.84260 0.69265 0.46179 0.39738 0.36114 0.31307
## Proportion of Variance 0.00021 0.00014 0.00006 0.00005 0.00004 0.00003
## Cumulative Proportion  0.99968 0.99982 0.99989 0.99993 0.99997 1.00000
```

PCAs after scaling

```
summary(scaled)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.5682 1.2744 1.2082 1.10794 1.03024 0.97732 0.93431
## Proportion of Variance 0.1892 0.1249 0.1123 0.09443 0.08165 0.07347 0.06715
## Cumulative Proportion  0.1892 0.3141 0.4264 0.52083 0.60247 0.67595 0.74310
##                            PC8     PC9    PC10    PC11   PC12    PC13
## Standard deviation     0.88262 0.83565 0.78237 0.70580 0.6161 0.61037
## Proportion of Variance 0.05992 0.05372 0.04708 0.03832 0.0292 0.02866
## Cumulative Proportion  0.80302 0.85674 0.90382 0.94214 0.9713 1.00000
```

Insight Gained: Proportion of the variance for PC1 reduced from 70% to 18%. This scenario is observed for other features as well. Hence, let us scale the data before we proceed further. Also, the diagrams below show how the data is centered before and after the scaling.

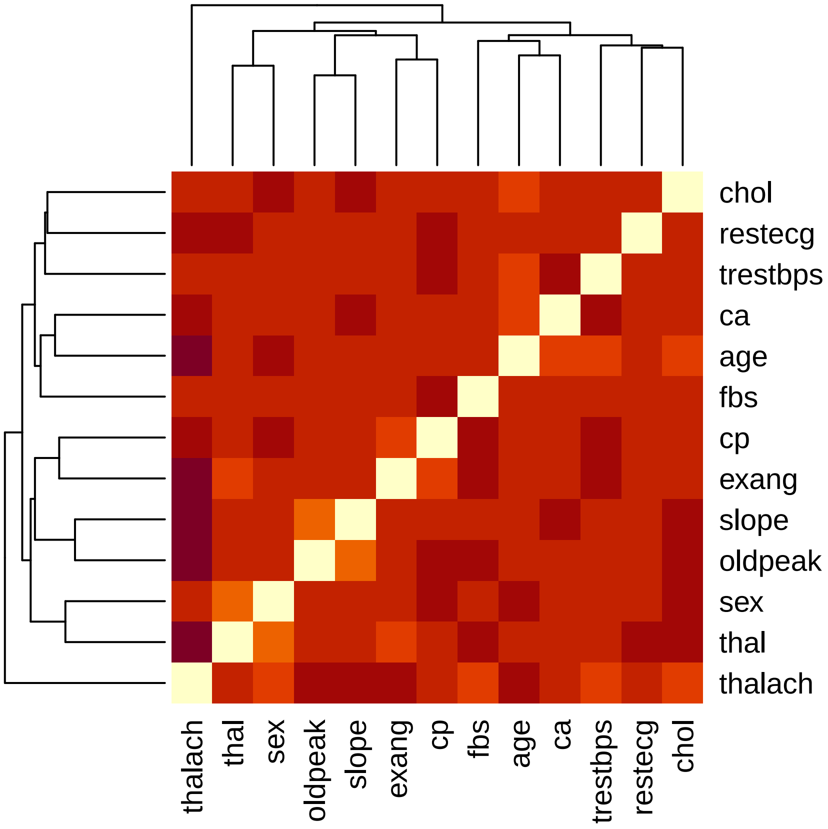


Let us scale data set by subtracting the column average from each column values and dividing that by overall column standard deviation.

```
hd_x_minus<-sweep(hd_x,2,colMeans(hd_x),"-")

hd_xdiv<-sweep(hd_x_minus,2,colSds(hd_x_minus),"/")
```

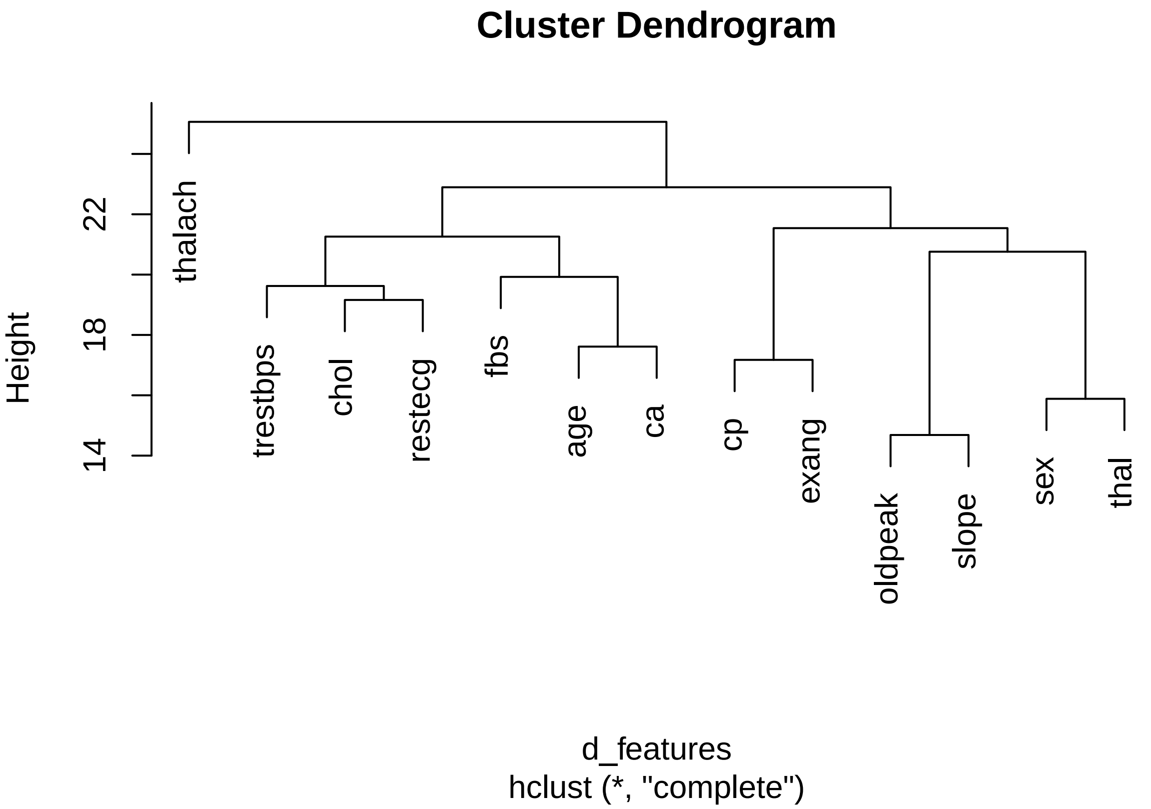
The next step is to perform cluster analysis to understand feature grouping and their distances between them. In order to perform cluster analysis, let us calculate the distance between the matrix's features. Following heatmap shows the features after calculating distances between the scaled data.

```
d_features <- dist(t(hd_xdiv))

heatmap(as.matrix(d_features))
```



The next step is analyzing the clusters present in the dataset. This will enhance our ability to explain why certain model choose a specific set of features leaving out the rest. Especially models like classification trees and Random Forest (even though random forest model reduces the human interpretability of the prediction).

Let us try to perform hierarchical clustering as taught in the course on the 14 features and then cut the tree into 3 groups. this analysis clearly shows prominent cluster groups like thalach. we can use this information in the clustering models



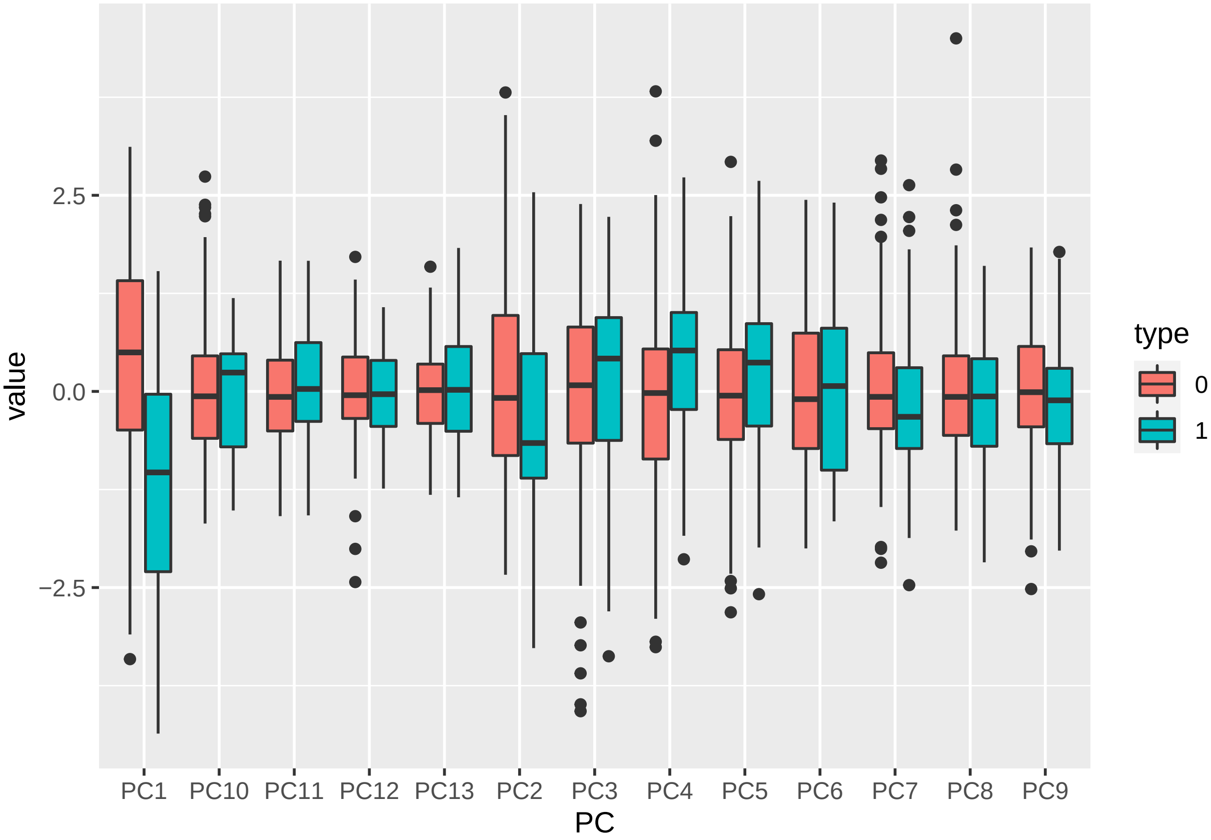
Insight Gained: The clusters after grouped into their prominent groups are shown below.

1) thalach being in its own cluster.
2) a group of clusters formed by cp,exang, oldpeak, slope, sex and thal.
3) Another group of clusters formed by rest of the features.

Now that we have scaled the data, let us apply PCA technique to actual dataset.

Insight Gained: An interesting observation is, even after scaling the data, by performing PCAs, we understand how the first few PCAs account for most of the variance and data distribution among the features.

Insight Gained: To see the results of the PCA analysis, let us plot PCA1 vs PCA2 and then all PCAs with a box plot to show how the first few PCAs account for all most all the feature importance and cumulated variance.



Insight Gained: Next diagram clearly showes, even after scaling, the composition of predictabiilty of 0 and 1 based on PCA1 is higher than all the other PCAs combined.

We have seen a similar approach before scaling however, the variance between the features were so much that there could have been a predictive bias if we hadn't scaled the data.

## Creation of training and testing datasets.

Now that we have cleaned, analyzed and visualized the features, let us try to start writing machine learning models to predict the presence of heart disease.

Before beginning the process, let us break the dataset into training and testing so that, majority of the data set is used to train the model and then apply the training on test data to predict y_hat of y - in this case, the presence (1) or absence (0) of heart disease.

```
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(hd_y, times = 1, p = 0.15, list = FALSE)
test_x <- hd_xdiv[test_index,]
test_y <- hd_y[test_index]
train_x <- hd_xdiv[-test_index,]
train_y <- hd_y[-test_index]
```

Once the data is split into test and train data, let us look at their matrix distribution ( number of columns and rows)

```
dim(test_x)
```

```
## [1] 34 13
```

```
dim(train_x)
```

```
## [1] 185  13
```

# Model Approach

Since this is a classification problem of DISEASE or NOT DISEASE, let us try k-means, logistic regression, classification trees and random forest

# Model 1 : k-means clustering model

similar to an exercise taught in the course, let us write a function for kmeans.

```
k<-kmeans(train_x,centers=2)
#predicting
pred_k<-predict_kmeans(test_x,k)

#loading actual test y outcomes
act_val_k<-test_y
#converting predicted values to disease or not disease status
pred_val_k<-ifelse(pred_k==’1’,’1’,’0’)

#using actual values and the predicted values, generating a confusion matrix. Only the overall accuracy
kmeans_acc<-confusionMatrix(data=factor(pred_val_k),reference = factor(act_val_k))$overall["Accuracy"]
kmeans_acc
```

```
##  Accuracy
## 0.7058824
```

# Modell 2: logistic regression model

```
#training
train_glm<-train(y=factor(train_y),x=train_x, method="glm")

#predicting
predict_glm<-predict(train_glm,test_x)

#using actual values and the predicted values, generating a confusion matrix. Only the overall accuracy
glm_acc<-confusionMatrix(data=factor(predict_glm),reference = factor(test_y))$overall["Accuracy"]
glm_acc
```

```
##  Accuracy
## 0.9117647
```

## Model 3: KNN Model

```r
#setting up tunegrid parameter
k_seq<-seq(3, 21, 2)

#training
train_knn<-train(y=factor(train_y),x=train_x, method="knn",tuneGrid = data.frame(k=k_seq))

#predicting
predict_knn<-predict(train_knn,test_x)

#using actual values and the predicted values, generating a confusion matrix. Only the overall accuracy
knn_acc<-confusionMatrix(data=predict_knn,reference = test_y)$overall["Accuracy"]
knn_acc
```

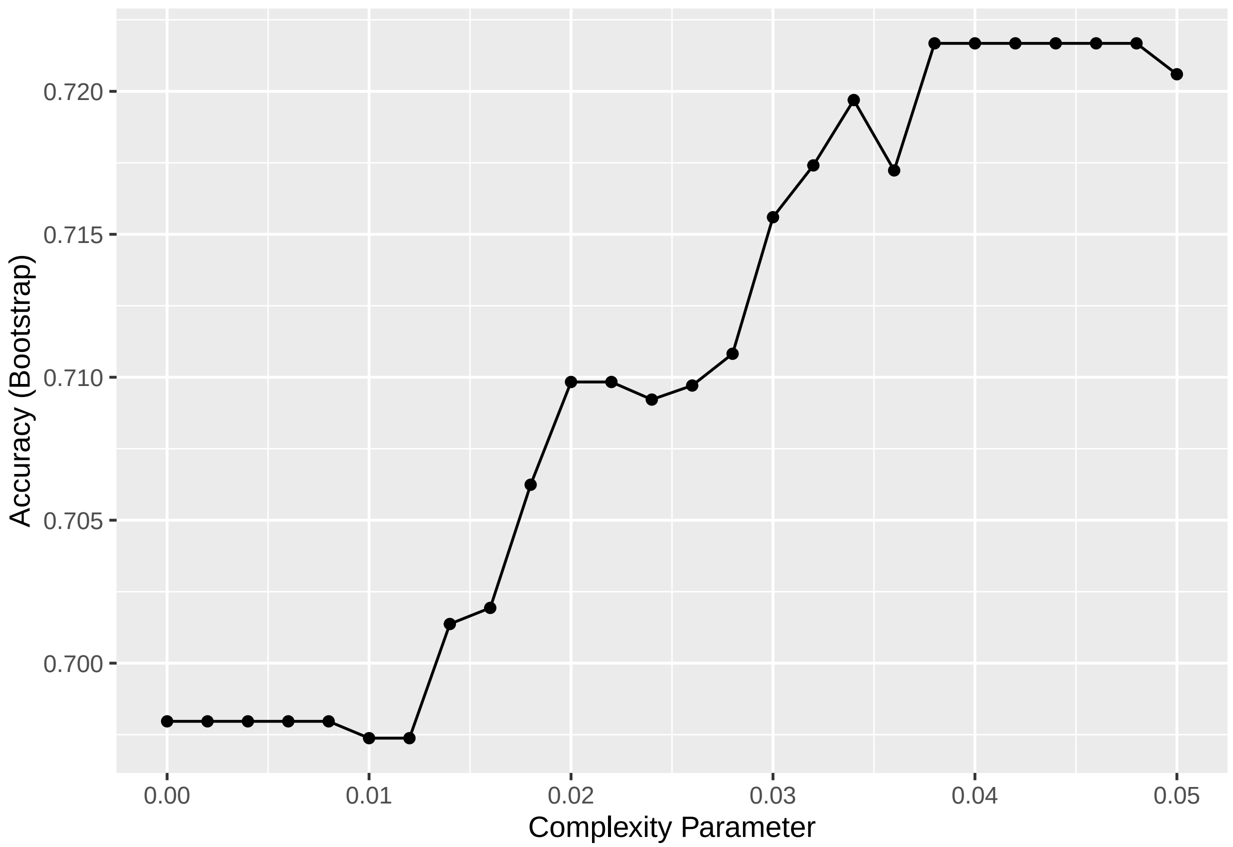
```
##  Accuracy
## 0.8823529
```

## Model 4: classification Tree Model

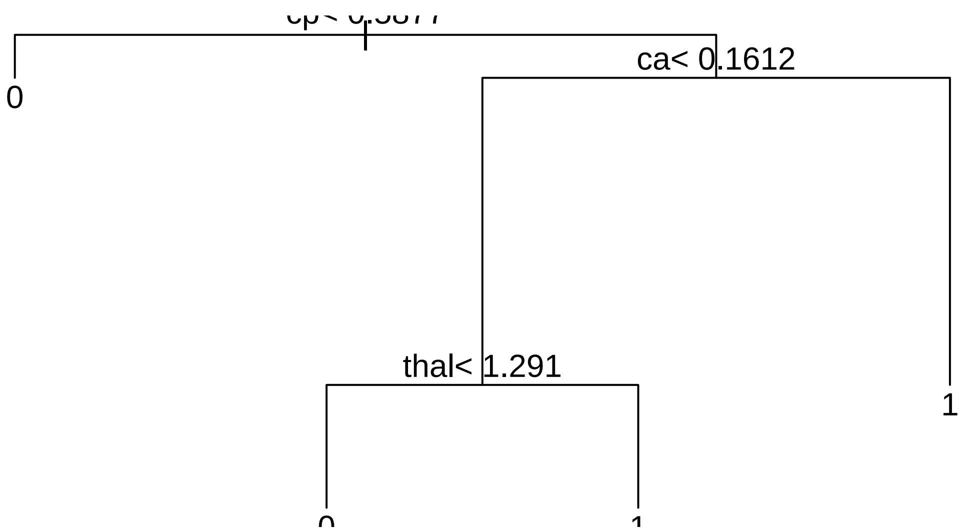
```r
#training
train_cls_tree<-train(y=factor(train_y),x=train_x, method="rpart",
                      tuneGrid = data.frame(cp = seq(0, 0.05, 0.002)))


#predicting
predict_cls_tree<-predict(train_cls_tree,test_x)

#using actual values and the predicted values, generating a confusion matrix. Only the overall accuracy
cls_tree_acc<-confusionMatrix(data=predict_cls_tree,reference = test_y)$overall["Accuracy"]
cls_tree_acc
```

```
##  Accuracy
## 0.9117647
```

For classification tree machine algorithm, we can try to create the tree structure and see if the cluster analysis helped in determining the features predicting the disease. The following diagram shows what complex parameter is



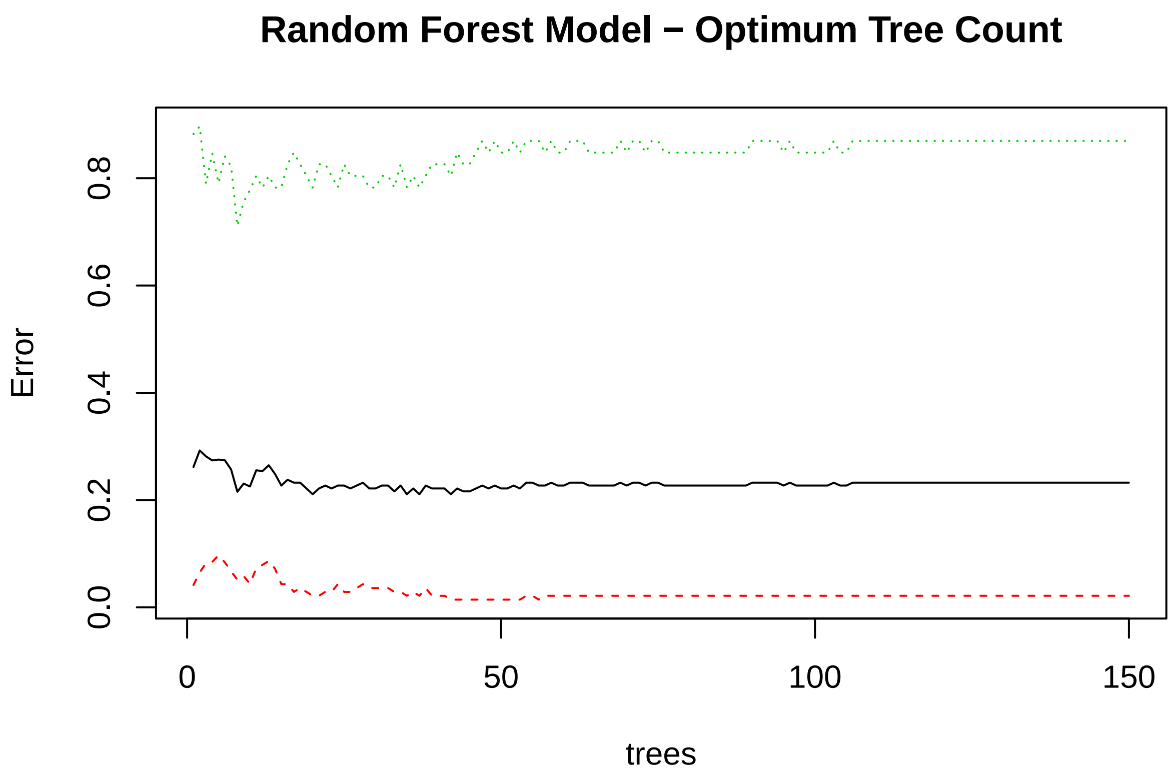
The following diagram shows the tree structure as explained above.



## Model 5: Randomg Forest Model

```
#training
train_rf<-train(y=factor(train_y),x=train_x, method="rf",
                ntree=150,
                tuneGrid = data.frame(mtry = seq(1,7,1)))

#plotting the predictors against the booostrap samples
#ggplot(train_rf)

#predicting
predict_rf<-predict(train_rf,test_x)

#using actual values and the predicted values, generating a confusion matrix. Only the overall accuracy
rf_acc<-confusionMatrix(data=factor(predict_rf),reference = factor(test_y))$overall["Accuracy"]
```

Following diagram shows the optimum tree count to be chosen for the prediction.



Listing all Accuracies

```
rf_acc
```

```
##  Accuracy
## 0.8235294
```

```
cls_tree_acc
```

```
##  Accuracy
## 0.9117647
```

```
knn_acc
```

```
##  Accuracy
## 0.8823529
```

```
glm_acc
```

```
##  Accuracy
## 0.9117647
```

```
kmeans_acc
```

```
##  Accuracy
## 0.7058824
```

## Creation of an Ensemble Model

Since there is variation in accuracy across different models, let us try to create an ensemble of all these models and see if we can improve the accuracy. What strategy could possibility be best fitting for this situation?

One possible argument is, the model should try to focus on the specificity rate. That is, the model should be able to predict the disease correctly for those who have the disease. In other words, the model should try to reduce false negatives. Hence, I have ensured that the ensemble will predict that a person has the heart disease even if one of the models predicted the person to have heart disease. Let us see if this strategy improves the accuracy of the overall model especially with respect to specificity.

```
#Based on the collection of all model values, let us predict the overall y_hat,
#fine-tuned by the ifelse condition to increase specificity.

result<-result%>%mutate(ens =
                          as.numeric(as.character(cls_tree))+
                 as.numeric(as.character(kmeans))+
                  as.numeric(as.character(glm))+
                  as.numeric(as.character(rf))
                  +as.numeric(as.character(knn))
                   )%>%
                    mutate(ensamble_y=ifelse(ens>=1,1,0))


#Accuracy of Ensemble Model.
Ens_Acc<-confusionMatrix(data=as.factor(result$ensamble_y),reference = test_y)$overall["Accuracy"]
Ens_Acc
```

```
##  Accuracy
## 0.8235294
```

```
#confusionMatrix(data=as.factor(result$ensamble_y),reference = test_y)
```

## Model Performance Analysis

Individual models like classification tree and logistic offer 90 plus over all accuracies, however their specificity rate is not that impressive. As noted above, my approach on this project is to enhance specificity so that the real patients are truly identified.

## Why did I choose overall accuracy of 82 than 94?

Even in the ensemble model, if the condition is changed from ifelse(ens>=1,1,0) to ifelse(ens>1,1,0), the overall accuracy is greatly increased to 94% - Meaning, a patient is considered having disease only if at least two models predict the person having a disease. However, the goal here is not to increase overall accuracy but to increase specificity rate. The confusion matrix below shows how best the specificity rate is for the ensemble model.

This ensures that when a person has heart disease, he or she is truly identified to be having heart disease and any posible false negatives are avoided.

```
confusionMatrix(data=as.factor(result$ensamble_y),reference = test_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20  1
##          1  5  8
##
##                Accuracy : 0.8235
##                  95% CI : (0.6547, 0.9324)
##     No Information Rate : 0.7353
##     P-Value [Acc > NIR] : 0.1658
##
##                   Kappa : 0.6031
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8000
##             Specificity : 0.8889
##          Pos Pred Value : 0.9524
##          Neg Pred Value : 0.6154
##              Prevalence : 0.7353
##          Detection Rate : 0.5882
##    Detection Prevalence : 0.6176
##       Balanced Accuracy : 0.8444
##
##        'Positive' Class : 0
##
```

As an added measure, the RMSE is calculated for the model.

```
#Finale RMSE of Ensemble Model

RMSE(as.numeric(result$ensamble_y),as.numeric(as.character(test_y)))
```

```
## [1] 0.420084
```

# Conclusion

As stated in the introduction, the goal of this project is to create a list of models that enhances the prediction of heart disease present, which has been met as explained in the sections above. However, the work is not done yet. The future work or the pending work in this model is to ingest more data available from different countries and see how model performs.

This is due to the limitation that there are only a few hundred data set available for the prediction. This could very well explain a presence of overfitting or overtraining of the model data.