Use Case Study Report: Classification of Cardiac Arrhythmia Disease

Group No.: Group 20

Student Names: Athul Sony, Aakash Atnoorkar

Executive Summary:

The study aims at classifying the Cardiac Arrhythmia disease into two classes – Normal ECG and Heart Disease. We have obtained this data from University of California at Irvine Machine Learning Data Repository. Each record contains clinical measurements from ECG signals and intervals and some other information such as sex, age, weight, along with the decision of a cardiologist. The dataset has 452 records of patients and 279 attributes which mostly has numerical columns. We have performed PCA on 198 numerical columns and selected 41 principal components for further analysis. We applied several models on this dataset including K-Nearest Neighbors, Logistic Regression, Decision Trees, Random Forests, Boosted Trees, Linear Discriminant Analysis, Neural Networks and Support Vector Machines. Out of the models built using these algorithms, Boosted Trees gave the maximum accuracy.

I. Background and Introduction:

In this era of fast-paced and stressful lifestyle, people are unknowingly causing heart diseases. As it is extremely important to keep our heart and accordingly our health at the optimum level, one should take routine heart tests. Cardiac Arrhythmia is known to be a root cause for many serious heart diseases including heart failure.

Arrhythmia is a form of irregularity in heart rhythms. The motive behind choosing dataset related to Cardiac Arrhythmia is to speed up the diagnosis of the disease and saving time for the cardiologist which can be utilized in treating the patients. After some research about the attributes of the dataset, we found that the data consists of the values recorded from the electrocardiogram (ECG). The ECG recordings are captured by placing the electrodes on the body parts and understanding the signals. The ECG signals consist of P waves, T waves, QRS Complex and R-R duration. These parameters help us understand and determine whether the patient has disease or not. The below figure shows an ECG signal with a description of its key features.

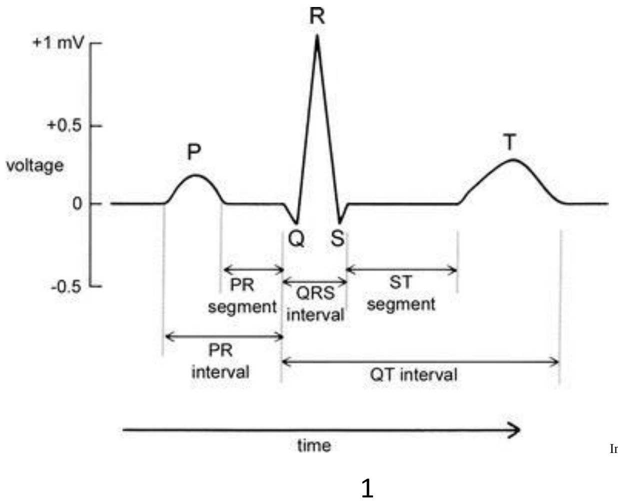
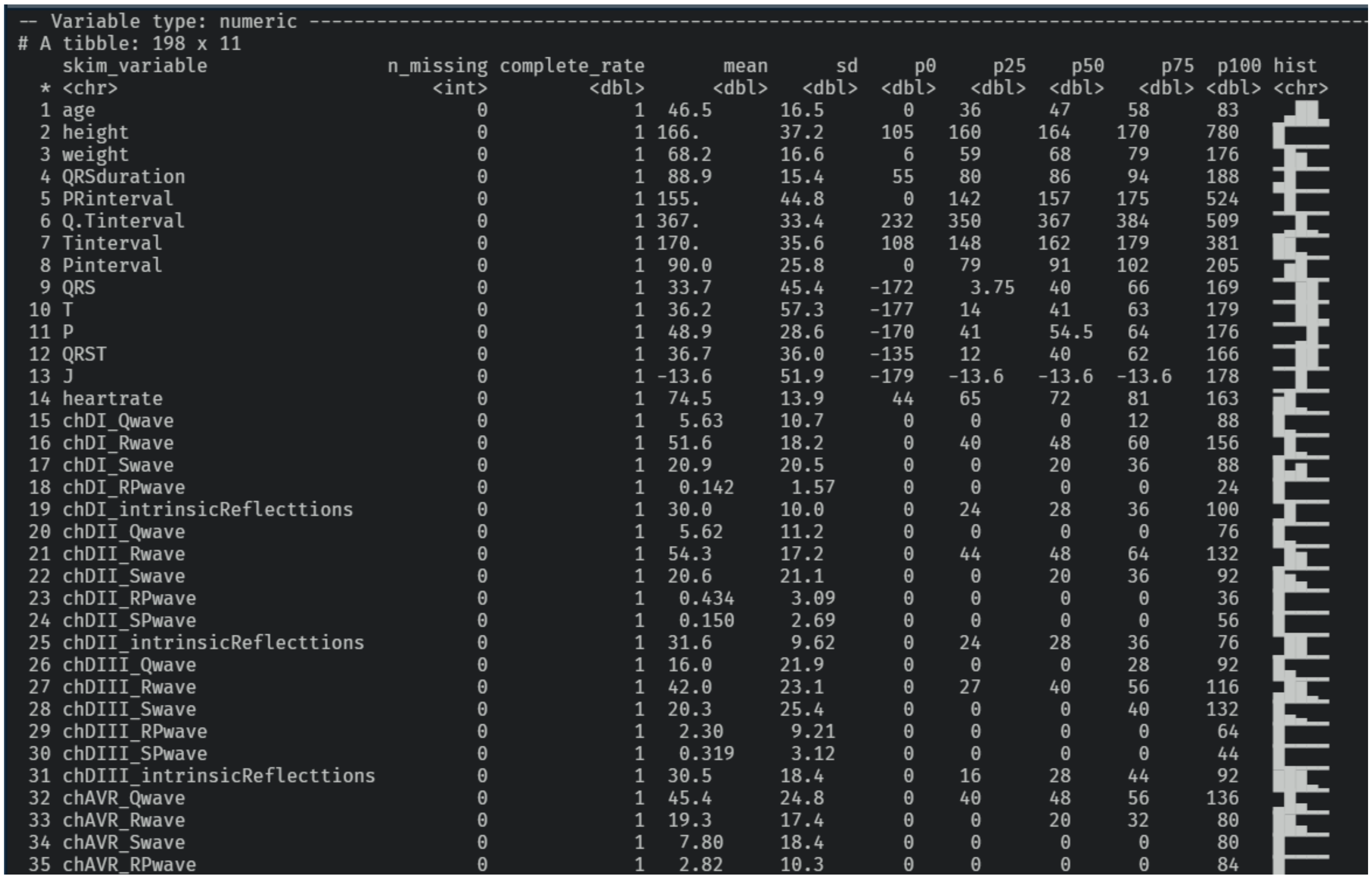


Image Courtesy:Handbook of Cardiac Anatomy, Physiology, and Devices

The dataset has 198 numerical variables and 81 categorical variables which includes some of the other important measurements such as weight, height, gender. There are total 13 types in which the diagnosis can be classified. Out of those, one tells us if the patient's heart condition is perfectly normal, and the others tell us the different types of Cardiac Arrhythmia. Each ECG signal in the dataset is 10s long and contains one rhythm class.

Even though there were 12 different arrythmia classifications in this dataset like ventricular tachycardia, atrial flutter, atrial fibrillation, malignant ventricular, ventricular bigeminy etc., for this project, we have converted the dataset into a binary classification problem as the algorithms that we learnt during this course would work best for binary classification problems. So, for this project, we are considering only 2 types of result – Normal ECG and Heart Disease. To identify whether a patient has a Heart Disease or has Normal ECG, we have built some models based on algorithms, which are K-Nearest Neighbors, Naïve Bayes, Random Forest, Decision Trees, Boosted Trees, Linear Discriminant Analysis, Logistic Regression, Artificial Neural Networks and Support Vector Machines. We plotted a confusion matrix and further compared the models based on several factors – Accuracy, Misclassification rate, Lift Charts and ROC charts.

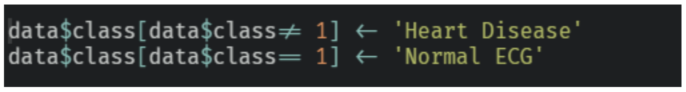
As we cannot accommodate summary of each and every column, we have included a snippet of summary of some of the important numerical variables.



```
-- Variable type: numeric ------------------------------------------------------------------------------------------
# A tibble: 198 x 11
   skim_variable                n_missing complete_rate    mean    sd    p0    p25   p50   p75  p100 hist
 * <chr>                            <int>         <dbl>   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <chr>
 1 age                                  0             1  46.5   16.5      0  36     47    58      83
 2 height                               0             1 166.    37.2    105 160    164   170     780
 3 weight                               0             1  68.2   16.6      6  59     68    79     176
 4 QRSduration                          0             1  88.9   15.4     55  80     86    94     188
 5 PRinterval                           0             1 155.    44.8      0 142    157   175     524
 6 Q.Tinterval                          0             1 367.    33.4    232 350    367   384     509
 7 Tinterval                            0             1 170.    35.6    108 148    162   179     381
 8 Pinterval                            0             1  90.0   25.8      0  79     91   102     205
 9 QRS                                  0             1  33.7   45.4   -172   3.75  40    66     169
10 T                                    0             1  36.2   57.3   -177  14     41    63     179
11 P                                    0             1  48.9   28.6   -170  41     54.5  64     176
12 QRST                                 0             1  36.7   36.0   -135  12     40    62     166
13 J                                    0             1 -13.6   51.9   -179 -13.6  -13.6 -13.6   178
14 heartrate                            0             1  74.5   13.9     44  65     72    81     163
15 chDI_Qwave                           0             1   5.63  10.7      0   0      0    12      88
16 chDI_Rwave                           0             1  51.6   18.2      0  40     48    60     156
17 chDI_Swave                           0             1  20.9   20.5      0   0     20    36      88
18 chDI_RPwave                          0             1   0.142  1.57     0   0      0     0      24
19 chDI_intrinsicReflecttions           0             1  30.0   10.0      0  24     28    36     100
20 chDII_Qwave                          0             1   5.62  11.2      0   0      0     0      76
21 chDII_Rwave                          0             1  54.3   17.2      0  44     48    64     132
22 chDII_Swave                          0             1  20.6   21.1      0   0     20    36      92
23 chDII_RPwave                         0             1   0.434  3.09     0   0      0     0      36
24 chDII_SPwave                         0             1   0.150  2.69     0   0      0     0      56
25 chDII_intrinsicReflecttions          0             1  31.6    9.62     0  24     28    36      76
26 chDIII_Qwave                         0             1  16.0   21.9      0   0      0    28      92
27 chDIII_Rwave                         0             1  42.0   23.1      0  27     40    56     116
28 chDIII_Swave                         0             1  20.3   25.4      0   0      0    40     132
29 chDIII_RPwave                        0             1   2.30   9.21     0   0      0     0      64
30 chDIII_SPwave                        0             1   0.319  3.12     0   0      0     0      44
31 chDIII_intrinsicReflecttions         0             1  30.5   18.4      0  16     28    44      92
32 chAVR_Qwave                          0             1  45.4   24.8      0  40     48    56     136
33 chAVR_Rwave                          0             1  19.3   17.4      0   0     20    32      80
34 chAVR_Swave                          0             1   7.80  18.4      0   0      0     0      80
35 chAVR_RPwave                         0             1   2.82  10.3      0   0      0     0      84
```

II. Data Exploration and Visualization

The dataset is converted to a binary classification problem by running the following code



```
data$class[data$class!= 1] <- 'Heart Disease'
data$class[data$class== 1] <- 'Normal ECG'
```

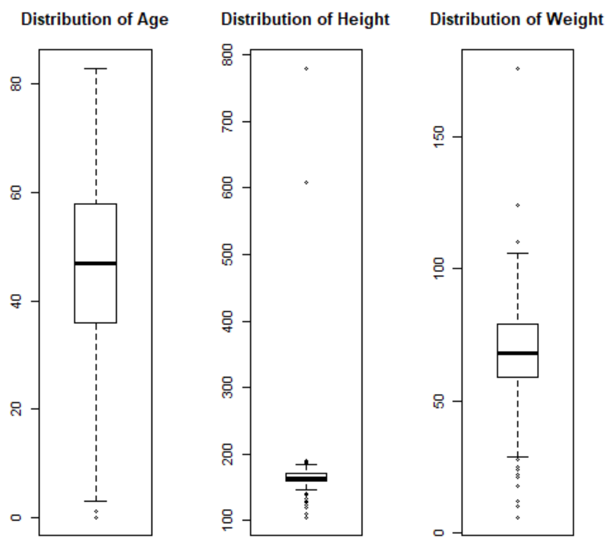
Data visualization techniques are applied on the dataset to explore the dataset further. The techniques used are:

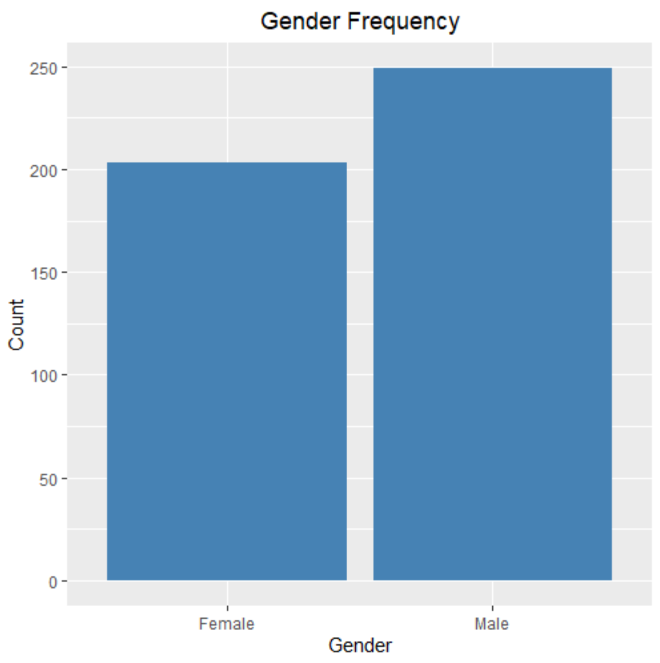
Boxplots- Boxplots of predictor variables helped us determine the outliers. Some of the input variables contained outliers and these require further investigation.

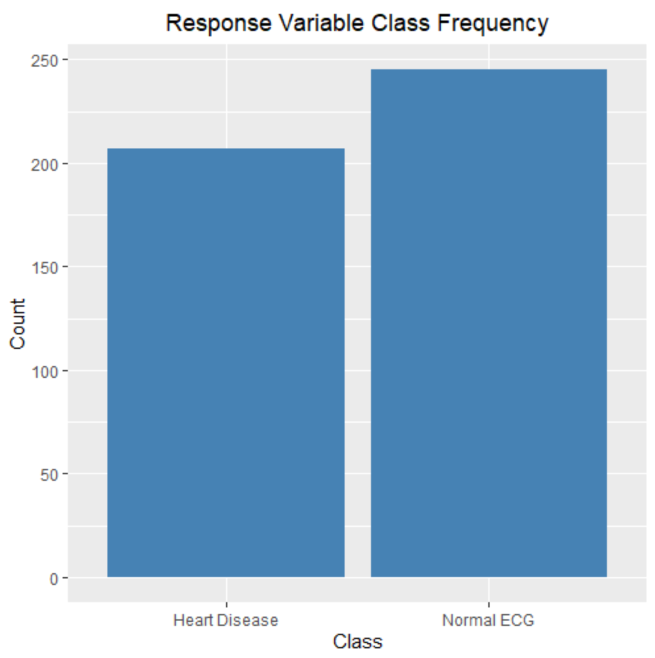
Histogram- Histogram of predictor variables are plotted to help us understand whether the variables are normally distributed or has some skewness.

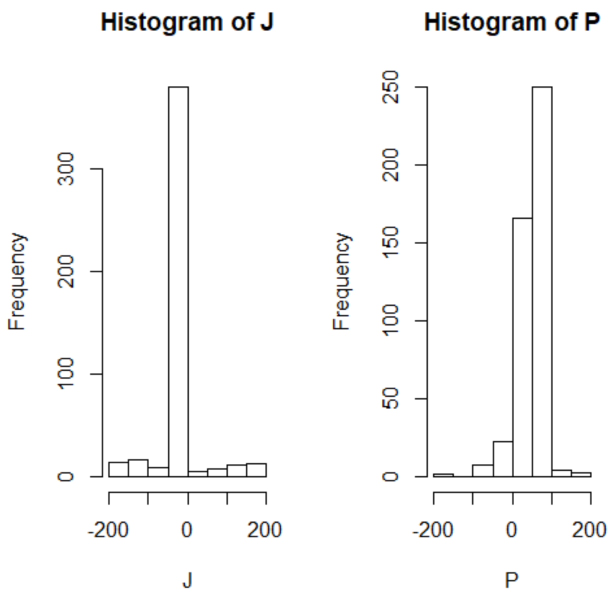
Scatterplots- Scatterplots of predictor variables against each other are plotted to help us understand how change in one variable affects another.

First, boxplots, histograms and bar plots of few input predictors are plotted.

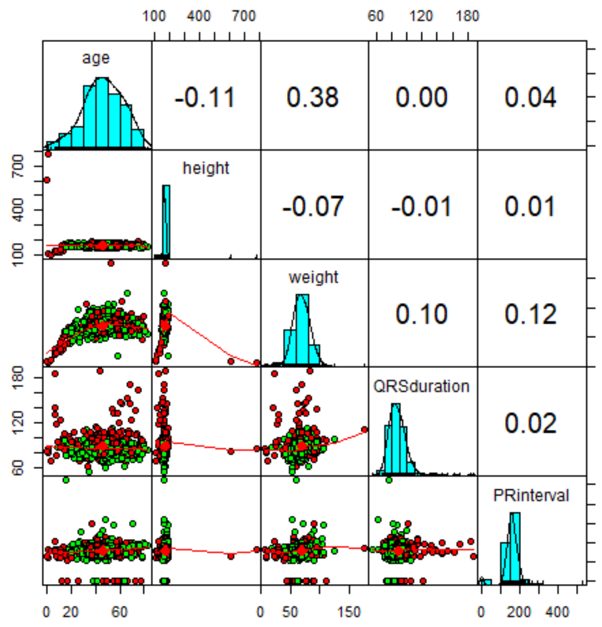


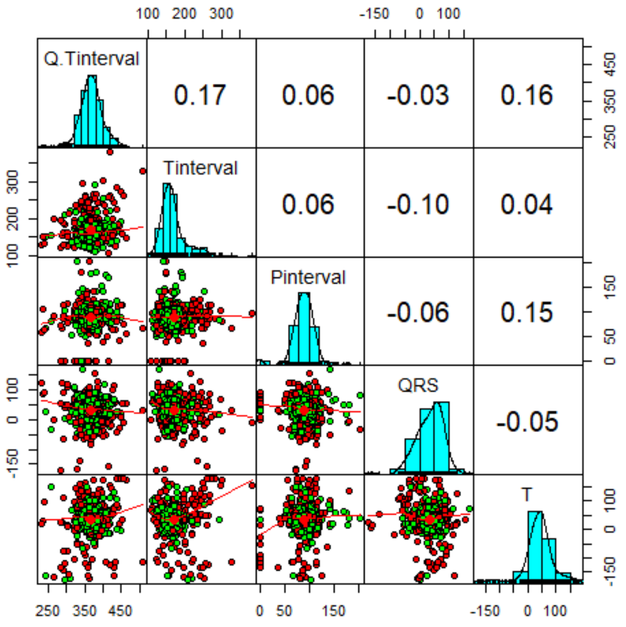


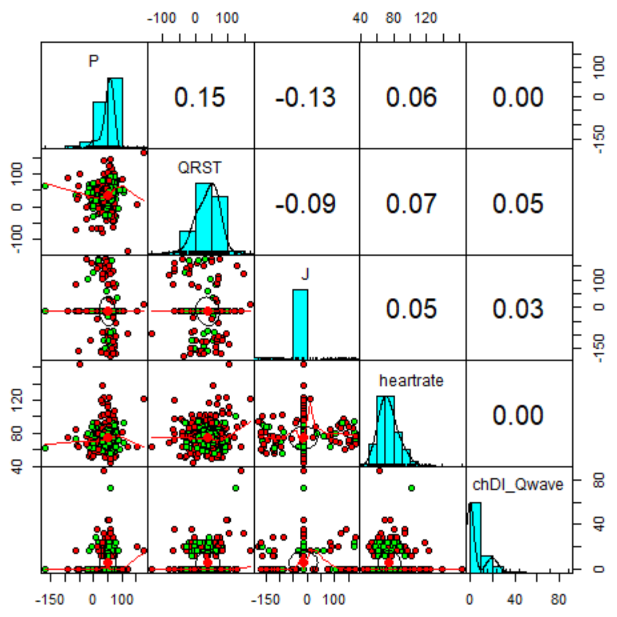


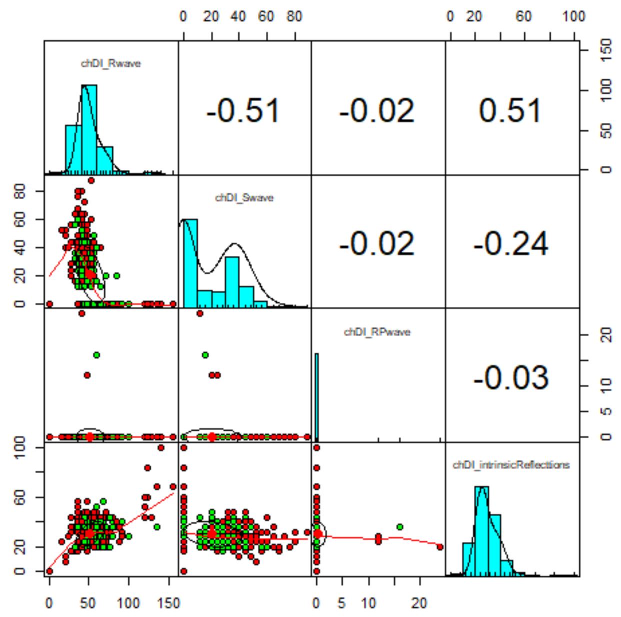


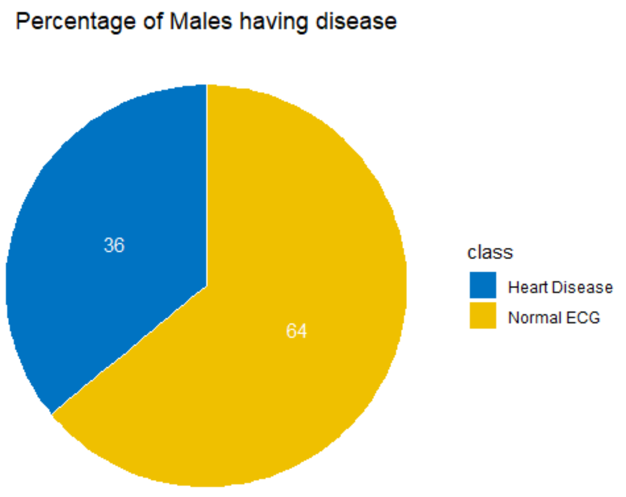
Now, we have plotted scatter plots of some variables to see how much one variable is affected by the other.

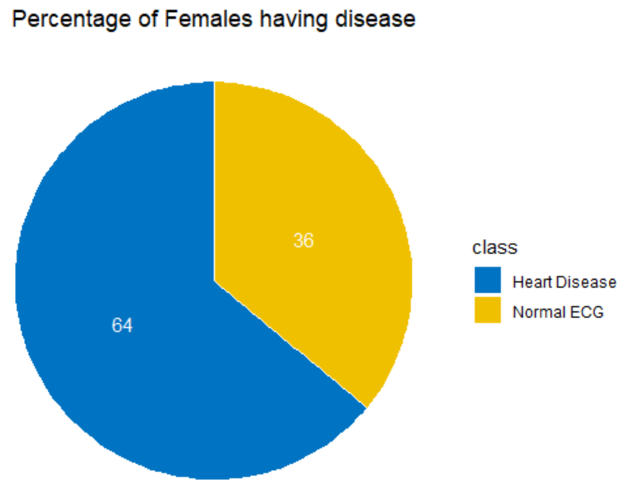








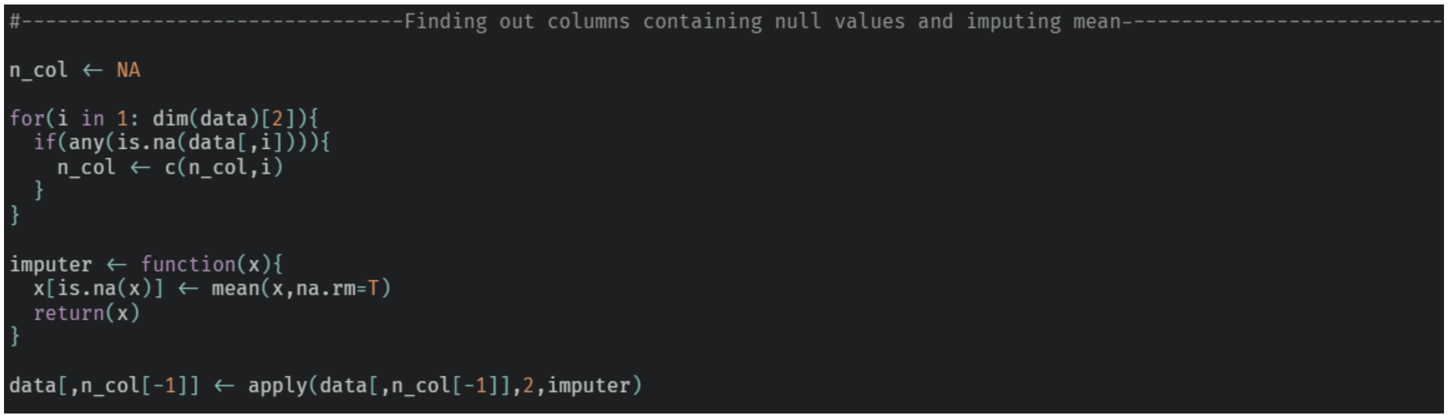




- Male population is more than Females in this dataset
- As seen from the above results, in the dataset, higher proportion of females have arrythmia diseases than males
- chDI_Rwave and chDI_Swave are moderately negatively correlated
- Age and Weight are positively correlated

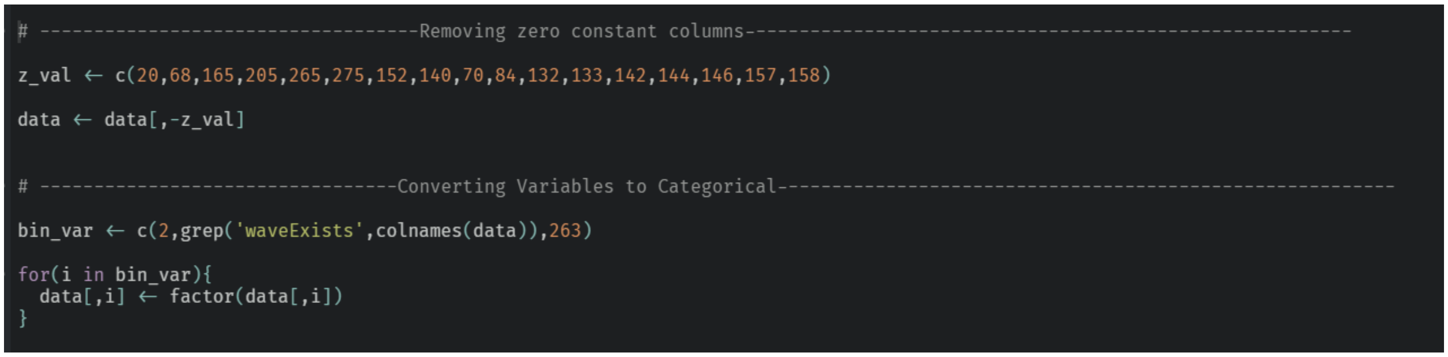
III. Data Preparation and Preprocessing

Some of the variables contain NA values and these NA values will be imputed with the mean by running the following commands



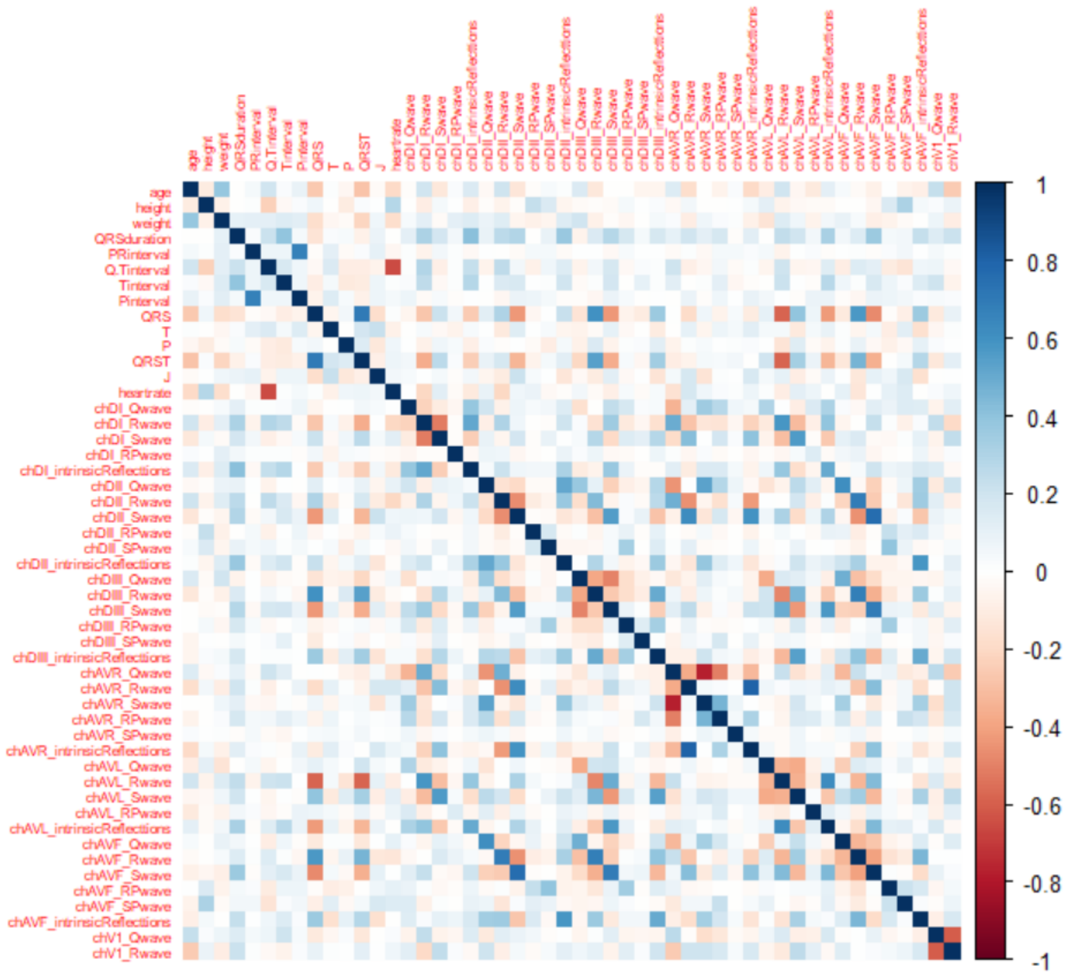
```
#--------------------------------Finding out columns containing null values and imputing mean--------------------------

n_col ← NA

for(i in 1: dim(data)[2]){
  if(any(is.na(data[,i]))){
    n_col ← c(n_col,i)
  }
}

imputer ← function(x){
  x[is.na(x)] ← mean(x,na.rm=T)
  return(x)
}

data[,n_col[-1]] ← apply(data[,n_col[-1]],2,imputer)
```

Also, some variables contain constant values. We have removed those variables from analysis. Also required variables were converted to factors.



```
# ---------------------------------Removing zero constant columns-------------------------------------------------------

z_val ← c(20,68,165,205,265,275,152,140,70,84,132,133,142,144,146,157,158)

data ← data[,-z_val]


# --------------------------------Converting Variables to Categorical-------------------------------------------------------

bin_var ← c(2,grep('waveExists',colnames(data)),263)

for(i in bin_var){
  data[,i] ← factor(data[,i])
}
```

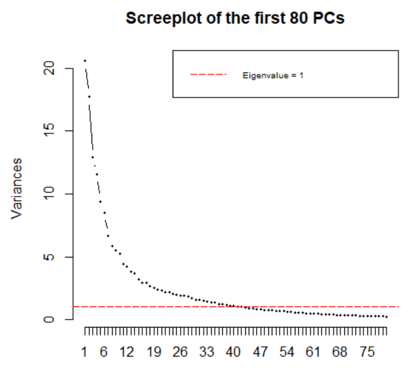
Now that we have performed Exploratory Data Analysis (EDA) on this dataset by checking if the dataset has any missing values and by performing mean imputation on the values which are unusual when compared with its distribution, the next step would be to check the correlation between each of the numerical variables. As the number of numerical variables are large in the dataset, shown below is the correlation plot of the first 50 numerical variables. From the below correlation plots it can be observed that there are many features which are highly correlated with each other.

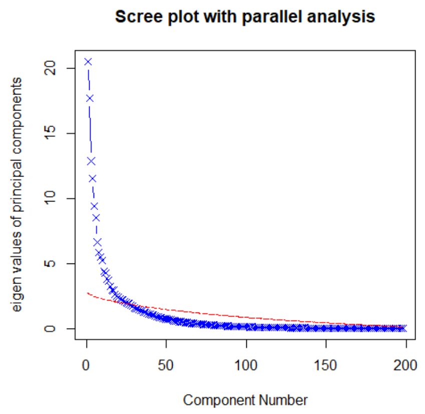


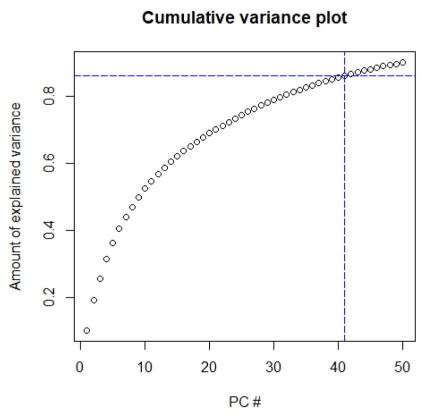
The high correlation between some of the features may hurt the interpretability of the model. Hence, it is necessary to deal with such features.

To deal with this issue it is required to perform the Principal Component Analysis (PCA) on the data and then choose the significant principal component that are required to build and train the model. Parallel Analysis was used to determine the number of principal components to retain. Following plots show the proportionality of variance for all the 198 principal components generated.

Out of 198 principal components we are choosing the first 41 principal components to build and train the model. These components capture a cumulative variance of 86.2% of the total data.





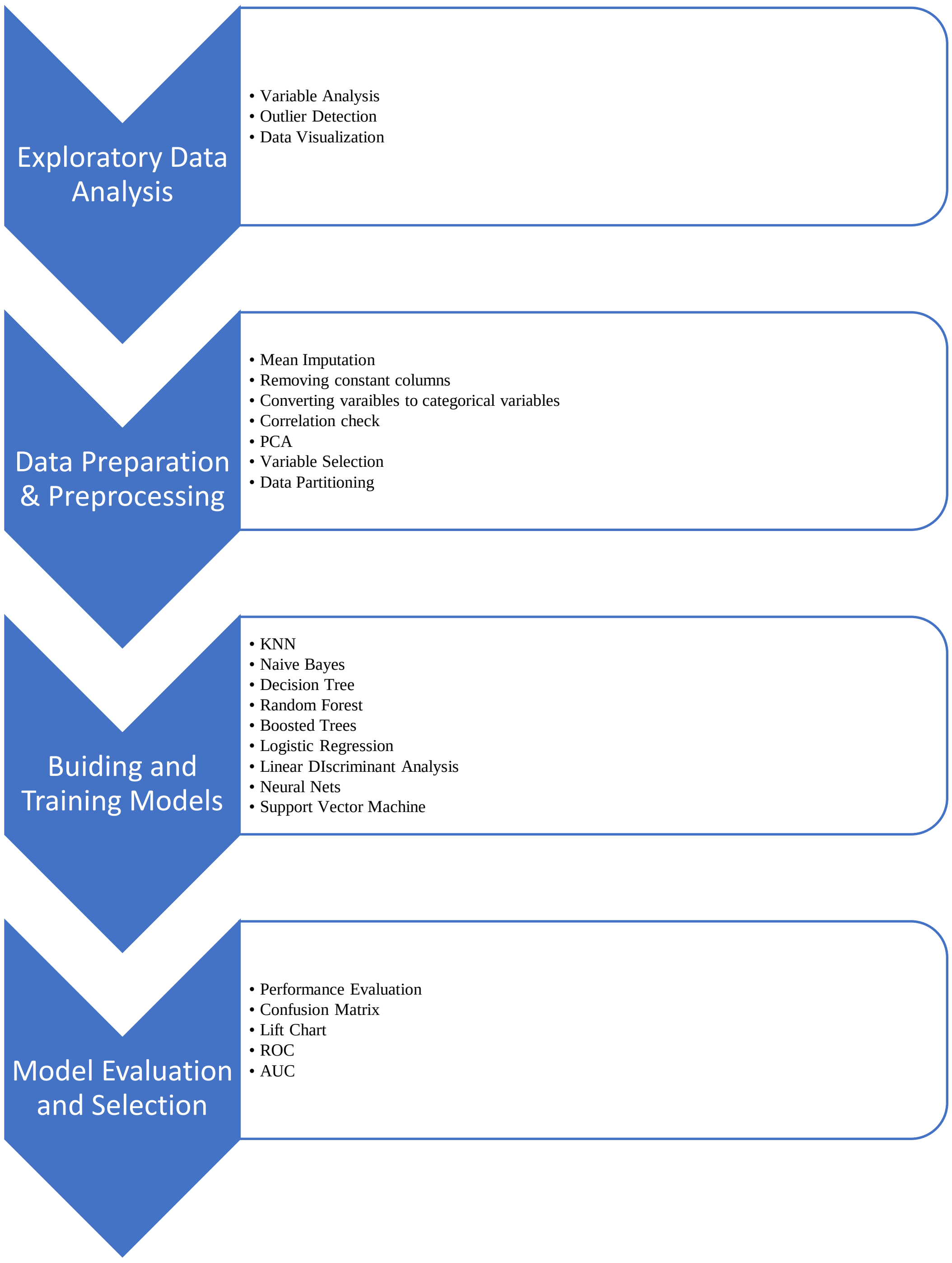


The next step would be to split the dataset into training and validation data. We have split the data into 7:3 ratio, that means we have randomly assigned 70% of data for training and 30% of data for validation.

IV. Data Mining Techniques and Implementation

We worked on 9 different classification algorithms on this dataset. The final required outcome is a variable named "Class". The "Class" variable is set to 1 if there is a "Heart Disease" and 0 if the patient has normal ECG. For this project, important class is set as "Heart Disease".

- KNN
- Naïve Bayes
- Classification Trees
- Random Forest
- Boosted Trees
- Logistic Regression
- Linear Discriminant Analysis
- Neural Nets
- Support Vector Machine



KNN Algorithm

- Categorical variables were converted to m dummy variables
- Data after PCA was partitioned into train and test sets with 70:30 ratio
- 10-fold cross validation was performed to obtain the best value of k that gave maximum accuracy
- Resampling method chosen was “repeatedcv” and number of resampling iterations was set to 3
- Based on the results of cross validation, a plot of K v/s Accuracy was graphed
- K = 5 was shown to give maximum accuracy based on cross validation results

- The trained model was applied on test set
- Confusion Matrix and Lift Chart were determined and plotted

Classification Trees Algorithm

- Data after PCA was partitioned into train, test sets with 70:30 ratio
- 10-fold cross validation was performed to obtain the optimal value of cp, "tuneLength" set to 100
- Resampling method chosen was "repeatedcv" and number of resampling iterations was set to 3
- Using the optimal cp value, best pruned tree model was built
- Best pruned tree model was plotted using prp function
- Best Pruned tree model was then applied to test set
- Confusion Matrix, Lift Chart, ROC and AUC were determined and plotted

Random Forest Algorithm

- Data after PCA was partitioned into train and test sets with 70:30 ratio
- 10-fold cross validation was performed to find the optimal value of mtry
- Out of Bag Error Estimate was noted for the trained model
- A plot of mtry v/s accuracy was plotted based on cross validation results
- Using the best value of mtry, the trained model was then applied on test set
- Confusion Matrix, Lift Chart, ROC and AUC were determined and plotted

Boosted Trees Algorithm

- Data after PCA was partitioned into train and test sets with 70:30 ratio
- Model was trained on the training set
- The trained model was then applied on test set
- Confusion Matrix, Lift Chart, ROC and AUC were determined and plotted

Logistic Regression Algorithm

- Data after PCA solved correlation problems associated with numerical variables
- Chi Square Test of Independence was used to check for dependence among categorical variables and those having high dependency were eliminated from analysis
- First, we took linear combination of predictor variables to build the logistic regression model and calculated accuracy of the model by applying model on test set and got 22% accuracy
- This was an indication that our response variable was not linearly separable
- Through some trial and error, we introduced polynomial predictor terms of second degree one by one and was able to get an accuracy of 28.89%
- Confusion Matrix and Lift Chart were determined and plotted

Linear Discriminant Analysis Algorithm

- Data after PCA solved correlation problems associated with numerical variables
- Chi Square Test of Independence was used to check for dependence among categorical variables and those having high dependency were eliminated from analysis
- For categorical variables, m-1 dummy variables were created

- Data was then partitioned into Train and Test set with 70:30 ratio
- The trained model was then applied on test set
- Confusion Matrix and Lift Chart were determined and plotted

Artificial Neural Network Algorithm

- All numerical predictors were normalized to have values between [0,1]
- For nominal categorical variables, m-1 dummy variables were created
- Number of hidden layers was chosen as 1 and number of nodes as 1
- Data was then partitioned into Train and Test set with 70:30 ratio
- The model was trained with the training data
- The trained model was then applied on test set to get and accuracy of 70.37%
- Confusion Matrix and Lift Chart were determined and plotted

Support Vector Machine Algorithm

- Data after PCA was partitioned into train and test sets with 70:30 ratio
- Model was trained on the training set
- The trained model was then applied on test set
- Confusion Matrix, Lift Chart, ROC and AUC were determined and plotted

Naïve Bayes Algorithm

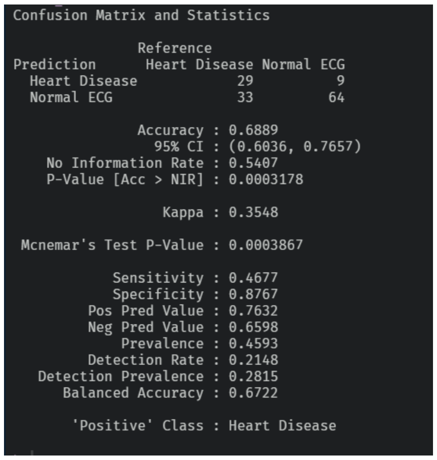
- All numerical variables were binned and converted to categorical variables
- This data was partitioned into train and test sets with 70:30 ratio
- Model was trained on the training set
- The trained model was then applied on test set
- Confusion Matrix, Lift Chart, ROC and AUC were determined and plotted

V. Performance Evaluation

Based on the consideration of accuracy of our model, we picked Boosted Trees Model to be the best classifier that could separate our response variable.

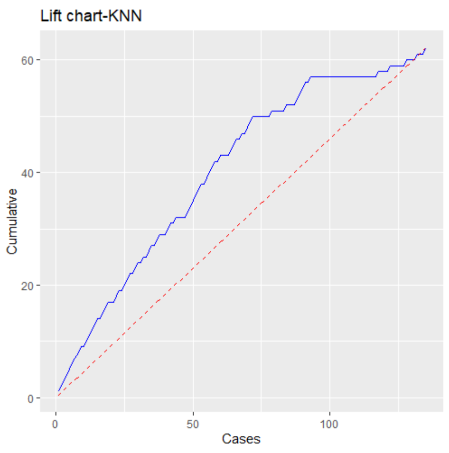
Algorithm	**Model Accuracy**
KNN	68.89%
Naïve Bayes	68.15%
Decision Tree	73.33%
Random Forest	77.04%
Boosted Trees	**80.74%**
Logistic Regression	28.89%
Linear Discriminant Analysis	74.81%
Artificial Neural Network	70.37%
Support Vector Machine	78.52%

Output Metrics of KNN Algorithm on Test Data

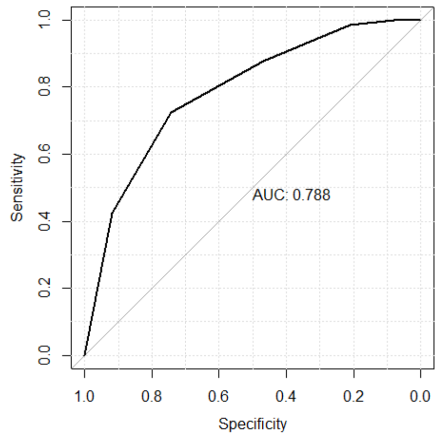


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            29          9
  Normal ECG               33         64

               Accuracy : 0.6889
                 95% CI : (0.6036, 0.7657)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 0.0003178

                  Kappa : 0.3548

 Mcnemar's Test P-Value : 0.0003867

            Sensitivity : 0.4677
            Specificity : 0.8767
         Pos Pred Value : 0.7632
         Neg Pred Value : 0.6598
             Prevalence : 0.4593
         Detection Rate : 0.2148
   Detection Prevalence : 0.2815
      Balanced Accuracy : 0.6722

       'Positive' Class : Heart Disease
```

Confusion Matrix

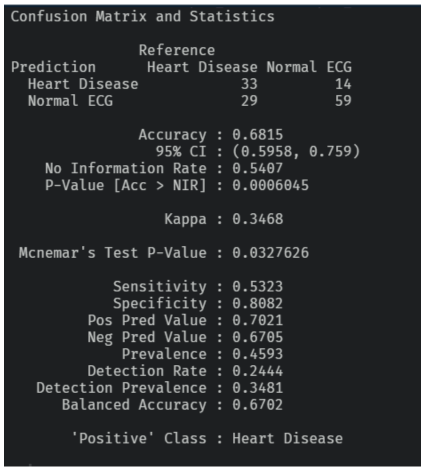


Lift Chart



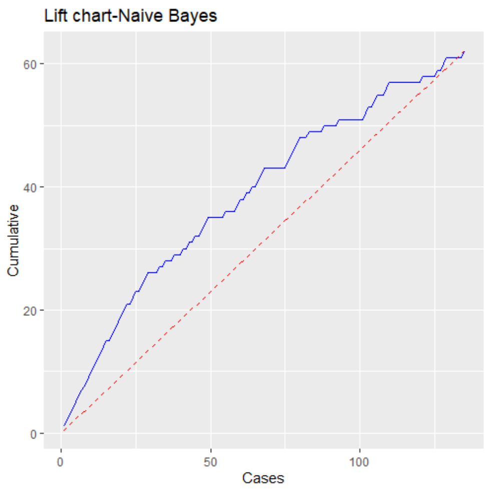
ROC Chart

Output Metrics of Naïve Bayes Model on Test Data

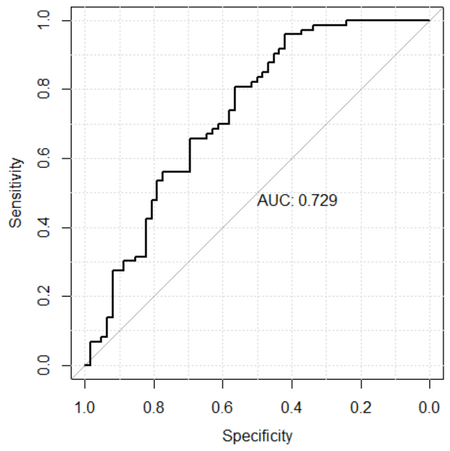


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            33         14
  Normal ECG               29         59

               Accuracy : 0.6815
                 95% CI : (0.5958, 0.759)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 0.0006045

                  Kappa : 0.3468

 Mcnemar's Test P-Value : 0.0327626

            Sensitivity : 0.5323
            Specificity : 0.8082
         Pos Pred Value : 0.7021
         Neg Pred Value : 0.6705
             Prevalence : 0.4593
         Detection Rate : 0.2444
   Detection Prevalence : 0.3481
      Balanced Accuracy : 0.6702

       'Positive' Class : Heart Disease
```

Confusion Matrix

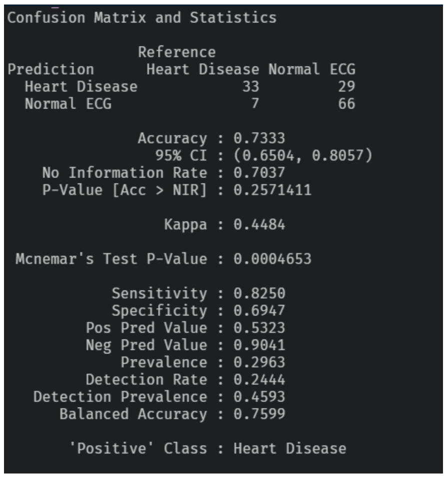


Lift Chart



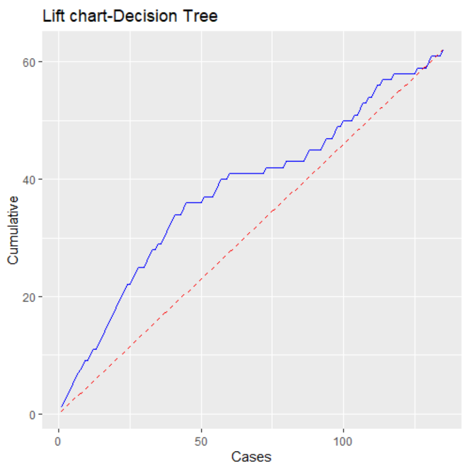
ROC Chart

Output Metrics of Decision Tree Algorithm on Test Data

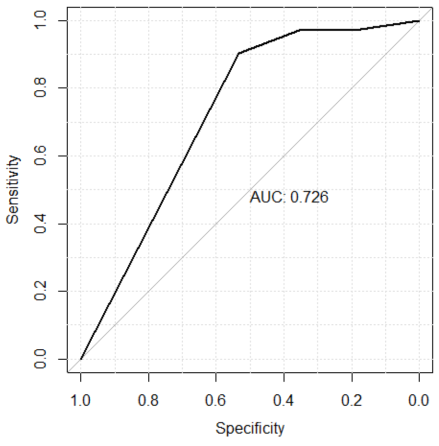


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            33         29
  Normal ECG                7         66

               Accuracy : 0.7333
                 95% CI : (0.6504, 0.8057)
    No Information Rate : 0.7037
    P-Value [Acc > NIR] : 0.2571411

                  Kappa : 0.4484

 Mcnemar's Test P-Value : 0.0004653

            Sensitivity : 0.8250
            Specificity : 0.6947
         Pos Pred Value : 0.5323
         Neg Pred Value : 0.9041
             Prevalence : 0.2963
         Detection Rate : 0.2444
   Detection Prevalence : 0.4593
      Balanced Accuracy : 0.7599

       'Positive' Class : Heart Disease
```

Confusion Matrix

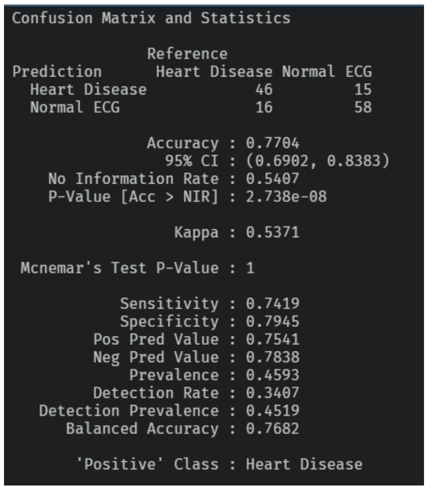


Lift Chart



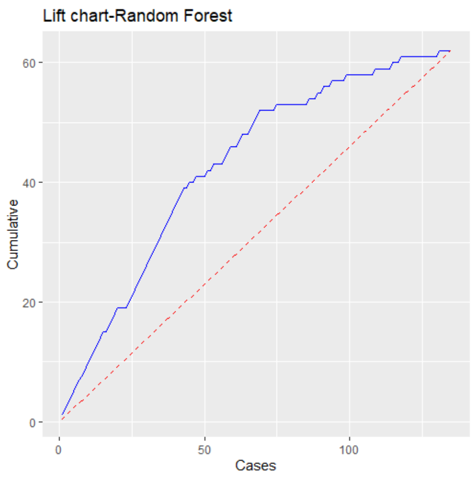
ROC Chart

Output Metrics of Random Forest Algorithm on Test Data

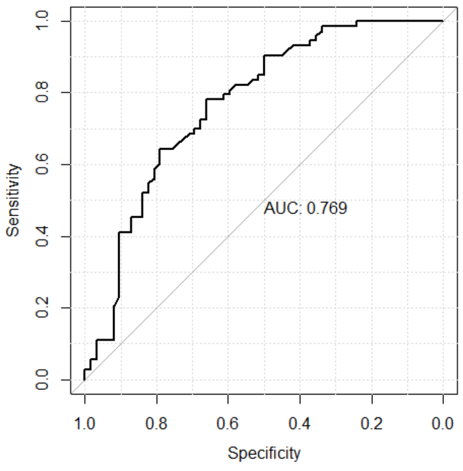


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            46         15
  Normal ECG               16         58

               Accuracy : 0.7704
                 95% CI : (0.6902, 0.8383)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 2.738e-08

                  Kappa : 0.5371

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7419
            Specificity : 0.7945
         Pos Pred Value : 0.7541
         Neg Pred Value : 0.7838
             Prevalence : 0.4593
         Detection Rate : 0.3407
   Detection Prevalence : 0.4519
      Balanced Accuracy : 0.7682

       'Positive' Class : Heart Disease
```

Confusion Matrix

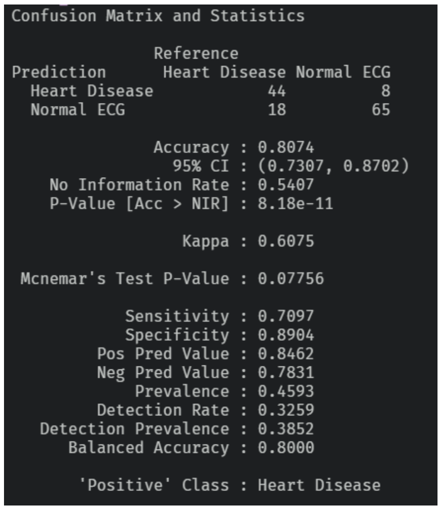


Lift Chart



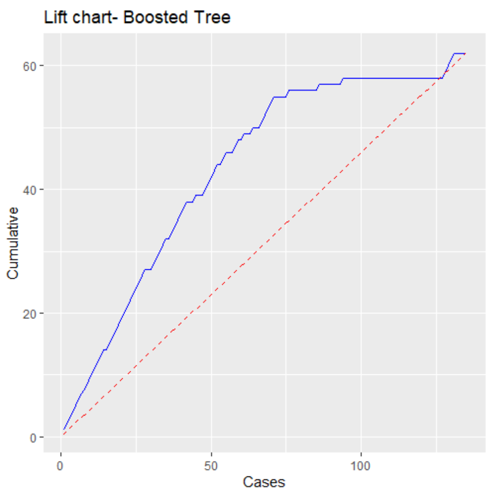
ROC Chart

Output Metrics of Boosted Trees Algorithm on Test Data

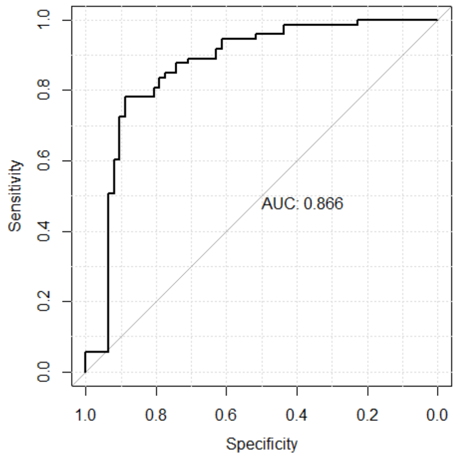


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            44          8
  Normal ECG               18         65

               Accuracy : 0.8074
                 95% CI : (0.7307, 0.8702)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 8.18e-11

                  Kappa : 0.6075

 Mcnemar's Test P-Value : 0.07756

            Sensitivity : 0.7097
            Specificity : 0.8904
         Pos Pred Value : 0.8462
         Neg Pred Value : 0.7831
             Prevalence : 0.4593
         Detection Rate : 0.3259
   Detection Prevalence : 0.3852
      Balanced Accuracy : 0.8000

       'Positive' Class : Heart Disease
```

Confusion Matrix

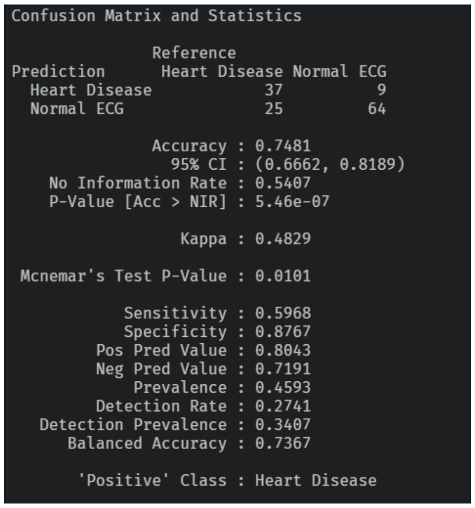


Lift Chart



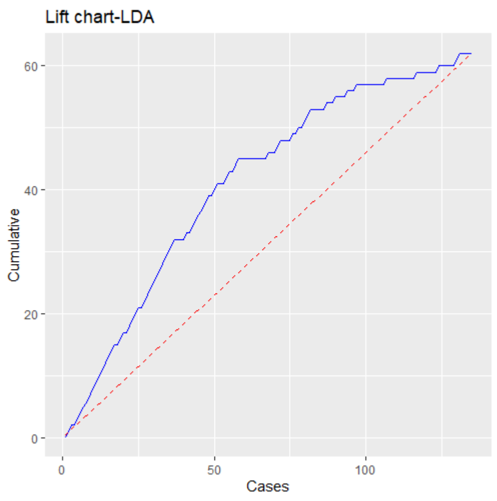
ROC Chart

Output Metrics of LDA Algorithm on Test Data

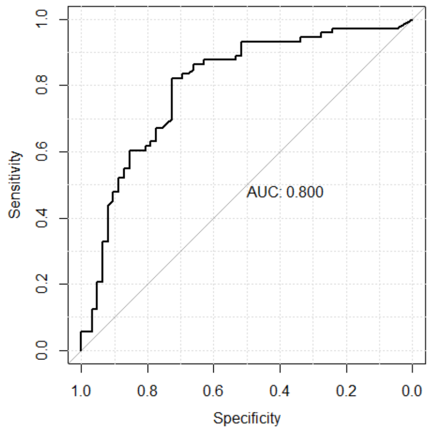


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            37          9
  Normal ECG               25         64

               Accuracy : 0.7481
                 95% CI : (0.6662, 0.8189)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 5.46e-07

                  Kappa : 0.4829

 Mcnemar's Test P-Value : 0.0101

            Sensitivity : 0.5968
            Specificity : 0.8767
         Pos Pred Value : 0.8043
         Neg Pred Value : 0.7191
             Prevalence : 0.4593
         Detection Rate : 0.2741
   Detection Prevalence : 0.3407
      Balanced Accuracy : 0.7367

       'Positive' Class : Heart Disease
```

Confusion Matrix

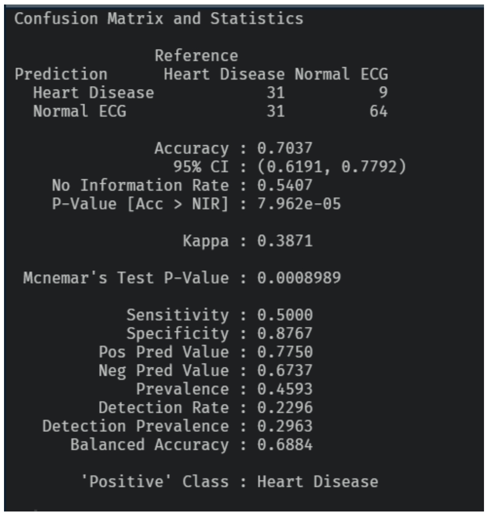


Lift Chart



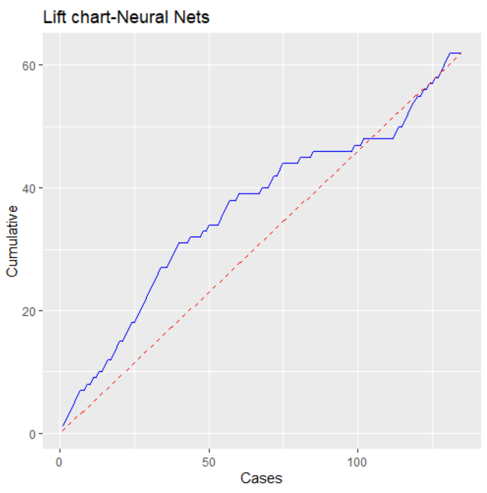
ROC Chart

Output Metrics of Artificial Neural Nets Algorithm on Test Data

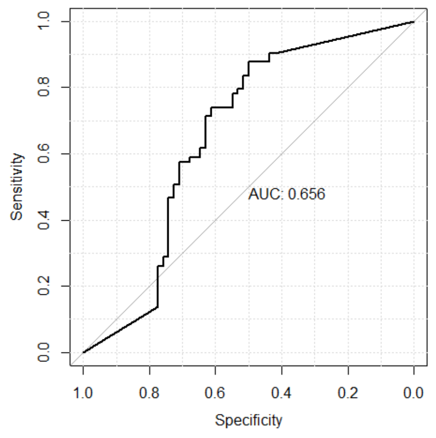


```
Confusion Matrix and Statistics

                 Reference
Prediction        Heart Disease Normal ECG
  Heart Disease              31          9
  Normal ECG                 31         64

               Accuracy : 0.7037
                 95% CI : (0.6191, 0.7792)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 7.962e-05

                  Kappa : 0.3871

 Mcnemar's Test P-Value : 0.0008989

            Sensitivity : 0.5000
            Specificity : 0.8767
         Pos Pred Value : 0.7750
         Neg Pred Value : 0.6737
             Prevalence : 0.4593
         Detection Rate : 0.2296
   Detection Prevalence : 0.2963
      Balanced Accuracy : 0.6884

       'Positive' Class : Heart Disease
```

Confusion Matrix

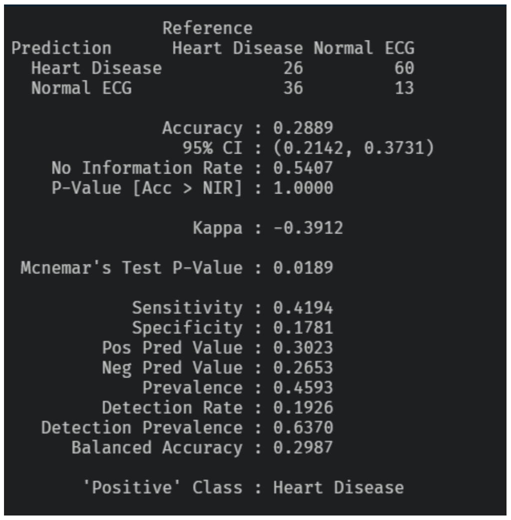


Lift Chart



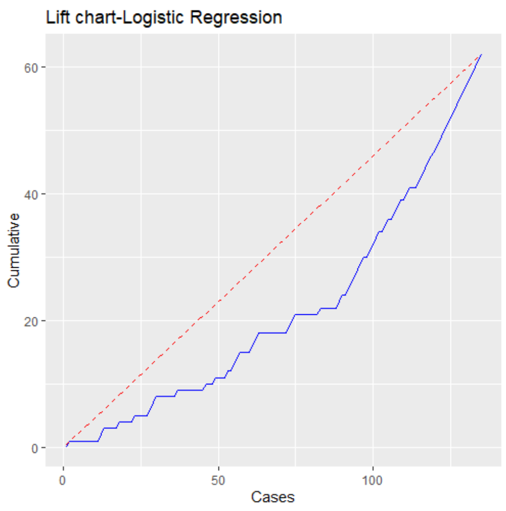
ROC Chart

Output Metrics of Logistic Regression on Test Data



```
                 Reference
Prediction        Heart Disease Normal ECG
  Heart Disease              26         60
  Normal ECG                 36         13

               Accuracy : 0.2889
                 95% CI : (0.2142, 0.3731)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 1.0000

                  Kappa : -0.3912

 Mcnemar's Test P-Value : 0.0189

            Sensitivity : 0.4194
            Specificity : 0.1781
         Pos Pred Value : 0.3023
         Neg Pred Value : 0.2653
             Prevalence : 0.4593
         Detection Rate : 0.1926
   Detection Prevalence : 0.6370
      Balanced Accuracy : 0.2987

       'Positive' Class : Heart Disease
```

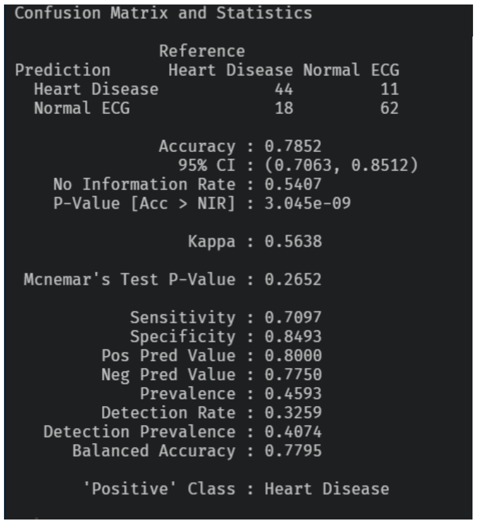
Confusion Matrix



Lift Chart

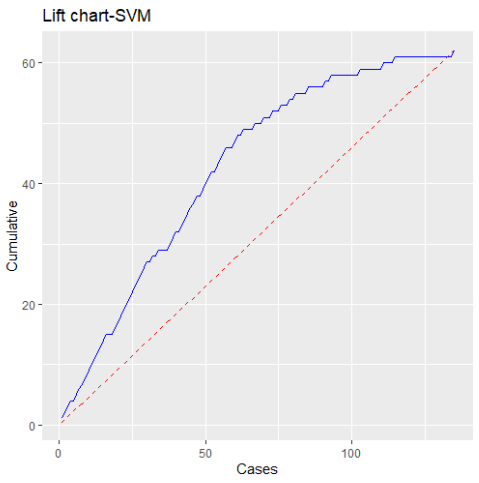
*As seen from lift charts results, Logistic Regression model performs worse than a Random model. This might be because of the inaccuracy in the degree of the polynomial we selected for each predictor. When we took the linear combination of predictors, the model performed the worst. So, detailed study taking into consideration what polynomial terms is needed for each predictor needs to be done.

Output Metrics of Support Vector Machine on Test Data

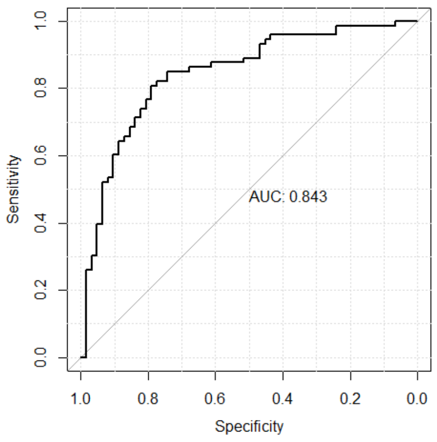


```
Confusion Matrix and Statistics

               Reference
Prediction      Heart Disease Normal ECG
  Heart Disease            44         11
  Normal ECG               18         62

               Accuracy : 0.7852
                 95% CI : (0.7063, 0.8512)
    No Information Rate : 0.5407
    P-Value [Acc > NIR] : 3.045e-09

                  Kappa : 0.5638

 Mcnemar's Test P-Value : 0.2652

            Sensitivity : 0.7097
            Specificity : 0.8493
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.7750
             Prevalence : 0.4593
         Detection Rate : 0.3259
   Detection Prevalence : 0.4074
      Balanced Accuracy : 0.7795

       'Positive' Class : Heart Disease
```

Confusion Matrix



Lift Chart



ROC Chart

VI. Discussion and Recommendation

Of all the models, we recommend using Boosted Trees model as it gave the highest accuracy in predictions. Logistic Regression Model gave the least accuracy when a polynomial combination of predictors were considered. A linear combination of predictor terms were giving predictions even less accurate. A detailed trial and error analysis using different combinations of polynomial terms of predictors needs to be explored in order to get a complex nonlinear hypothesis and only then can the accuracy be increased. The dataset initially had 13 different classes in the response variable. We converted this dataset to a binary classification problem. Deep Learning techniques could be used for training models to give better accuracy for multi class classification.

VII. Summary

The case study helped us understand the different techniques used for prediction analysis and their performance in general and for this dataset in particular. Boosted Trees model after some tweaking can be used to categorize the output as having Heart Disease or having a Normal ECG.

Appendix: R Code for use case study

```r
library(psych)
library(caTools)
library(corrplot)
library(caret)
library(class)
library(rpart)
library(rpart.plot)
library(forecast)
library(dummies)
library(ggplot2)
library(scales)
library(randomForest)
library(dplyr)
library(MASS)
library(adabag)
library(neuralnet)
library(pROC)
library(e1071)
library(skimr)

# -------------------------------------------Reading Data---------------------------------------------------

setwd("C:/Users/athul/Downloads")

data <- read.csv('data.csv',stringsAsFactors = F,na.strings = c(' ','?',''))

data$class[data$class!= 1] <- 'Heart Disease'
data$class[data$class== 1] <- 'Normal ECG'

skim(data)


# -----------------------------------Removing zero constant columns---------------------------------------------

z_val <- c(20,68,165,205,265,275,152,140,70,84,132,133,142,144,146,157,158)

data <- data[,-z_val]


# ---------------------------------Converting Variables to Categorical------------------------------------------

bin_var <- c(2,grep('waveExists',colnames(data)),263)

for(i in bin_var){
  data[,i] <- factor(data[,i])
}


#--------------------------------Finding out columns containing null values and imputing mean-------------------

n_col <- NA

for(i in 1: dim(data)[2]){
  if(any(is.na(data[,i]))){
    n_col <- c(n_col,i)
  }
}

imputer <- function(x){
  x[is.na(x)] <- mean(x,na.rm=T)
  return(x)
}

data[,n_col[-1]] <- apply(data[,n_col[-1]],2,imputer)


#---------------------------------------------------------EDA---------------------------------------------------
```

```
par(mfrow=c(1,3))
boxplot(data$age, main="Distribution of Age")
boxplot(data$height, main="Distribution of Height")
boxplot(data$weight, main="Distribution of Weight")

par(mfrow=c(1,3))
boxplot(data$PRinterval, main="Distribution of PR interval")
boxplot(data$Q.Tinterval, main="Distribution of Q.T interval")
boxplot(data$Pinterval, main="Distribution of P interval")

par(mfrow=c(1,3))
boxplot(data$J, main="Distribution of J")
boxplot(data$P, main="Distribution of P")
boxplot(data$T, main="Distribution of T")

par(mfrow = c(1,2))
hist(data$J,main = 'Histogram of J',xlab = 'J')
hist(data$P,main = 'Histogram of P',xlab = 'P')

par(mfrow = c(1,2))
hist(data$T,main = 'Histogram of T',xlab = 'T')
hist(data$heartrate,main = 'Histogram of Heart Rate',xlab = 'Heart Rate')


pairs.panels(data[,c(1,3:6)],gap = 0,bg = c('red','green')[data$class],pch = 21)

pairs.panels(data[,c(7:11)],gap = 0,bg = c('red','green')[data$class],pch = 21)

pairs.panels(data[,c(12:16)],gap = 0,bg = c('red','green')[data$class],pch = 21)

pairs.panels(data[,c(17:20)],gap = 0,bg = c('red','green')[data$class],pch = 21)


data$gender <- ifelse(data$sex==1,'Male','Female')

data %>%
  ggplot(aes(x = gender))+
  geom_bar(fill='steelblue')+
  xlab('Gender')+
  ylab('Count')+
  ggtitle('Gender Frequency')+
  theme(plot.title = element_text(hjust = 0.5))

data %>%
  ggplot(aes(x = class))+
  geom_bar(fill='steelblue')+
  xlab('Class')+
  ylab('Count')+
  ggtitle('Response Variable Class Frequency')+
  theme(plot.title = element_text(hjust = 0.5))
```

```
x %>%
  ggplot(aes(x = "", y = Prop, fill = class)) +
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = lab.ypos, label = Prop), color = "white")+
  scale_fill_manual(values = c("#0073C2FF", "#EFC000FF")) +
  theme_void()+
  ggtitle('Percentage of Males having disease')+
  theme(plot.title = element_text(hjust = 0.5))


y <- data[data$sex==0,c(2,280)] %>%
  group_by(sex,class) %>%
  summarise(Count = n())
y$Prop <- round((x$Count/sum(x$Count)) * 100,0)

y <- y %>%
  arrange(desc(class)) %>%
  mutate(lab.ypos = cumsum(Prop) - 0.5*Prop)
```

```
y %>%
  ggplot(aes(x = "", y = Prop, fill = class)) +
  geom_bar(width = 1, stat = "identity", color = "white") +
  coord_polar("y", start = 0)+
  geom_text(aes(y = lab.ypos, label = Prop), color = "white")+
  scale_fill_manual(values = c("#0073C2FF", "#EFC000FF")) +
  theme_void()+
  ggtitle('Percentage of Females having disease')+
  theme(plot.title = element_text(hjust = 0.5))
```

```
# -----------------------------------------------------PCA-----------------------------------------------------

data <- data[,-264]

# Parallel Analysis to determine number of principal components

fa.parallel(data[,-bin_var],fa='pc',n.iter = 100,show.legend=F,main = 'Scree plot with parallel analysis')

# PCA

pca <- prcomp(data[,-bin_var],center=T,scale=T)

summary(pca)

# ScreePlot

screeplot(pca, type = "l", npcs = 80, main = "Screeplot of the first 80 PCs",pch = 16,cex = 0.4)
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"),col=c("red"), lty=5, cex=0.6)

# Cumulative Variance Plot

cumpro <- cumsum(pca$sdev^2 / sum(pca$sdev^2))
plot(cumpro[0:50], xlab = "PC #", ylab = "Amount of explained variance", main = "Cumulative variance plot")
abline(v = 41, col="blue", lty=5)
abline(h = 0.86179, col="blue", lty=5)

# Selecting 41 principal components

data_after_pca <- data.frame(data[,bin_var],pca$x[,1:41])


#---------------------------------------------------KNN Algorithm-----------------------------------------------


# creating m dummy variables

knn_dum <- data.frame(y = rep(NA,nrow(data_after_pca)))

for(i in colnames(data_after_pca[,1:64])){
  knn_dum <- cbind(knn_dum,dummy(i ,data = data_after_pca))
}

knn_dum <- knn_dum[-1]

data_knn <- data.frame(knn_dum,data_after_pca[,65:106])

# Partitioning the data

set.seed(100)

split_knn <- sample.split(data_knn$class,SplitRatio = 0.7)

train_set_knn <- data_knn[split_knn,]

test_set_knn <- data_knn[!split_knn,]

# Performing Cross Validation

ctrl <- trainControl(method="repeatedcv",repeats = 3,number = 10)

model_knn <- train(class ~ ., data = train_set_knn, method = "knn", trControl = ctrl, tuneLength = 20)
```

```
model_knn

# K v/s Accuracy Plot

plot(model_knn)

paste("Best value of k chosen is ",as.character(model_knn$bestTune))

# Applying the model on test data

output_knn <- data.frame(Actual = test_set_knn$class, Pred = predict(model_knn,newdata = test_set_knn),
           Prob = round(predict(model_knn,newdata = test_set_knn,type = 'prob'),2))

output_knn$Actual_Binary <- ifelse(output_knn$Actual == 'Heart Disease',1,0)

# Confusion Matrix

cfm_knn <- confusionMatrix(output_knn$Pred,output_knn$Actual)

cfm_knn

ggplotConfusionMatrix <- function(m){
  mytitle <- paste("Accuracy", percent_format()(m$overall[1]))
  p <-
    ggplot(data = as.data.frame(m$table) ,aes(x = Reference, y = Prediction)) +
    geom_tile(aes(fill = log(Freq)), colour = "white") +
    scale_fill_gradient(low = "white", high = "steelblue") +
    geom_text(aes(x = Reference, y = Prediction, label = Freq)) +
    theme(legend.position = "none") +
    ggtitle(mytitle)
  return(p)
}

ggplotConfusionMatrix(cfm_knn)

# Plotting Lift Chart

prop_knn <- round(table(test_set_knn$class)[1]/(table(test_set_knn$class)[1] + table(test_set_knn$class)[2]),2)

# Plotting Lift Chart

rlift.df.knn <- data.frame(x = c(1:135),
                          y = cumsum(output_knn$Actual_Binary[order(output_knn$Prob.Heart.Disease, decreasing =
T)]),
                          bench = c(1:135)*prop_knn)

ggplot(rlift.df.knn, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-KNN") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and finding AUC

roc_knn <- roc(output_knn$Actual,output_knn$Prob.Heart.Disease)

plot.roc(roc_knn,print.auc = T,grid = T)
```

```
#-----------------------------------------------Naive Bayes--------------------------------------------------

data_nb <- data_after_pca

# Binning numerical variables to convert them into categorical

for(i in 66:106){
  bin <- seq(min(data_nb[,i]),max(data_nb[,i]),(max(data_nb[,i])-min(data_nb[,i]))/3)
  data_nb[,i]<- cut(data_nb[,i],bin,labels = 1:3,include.lowest=TRUE, right=FALSE)
}

# Partitioning the data into train test

set.seed(100)
```

```
split_nb <- sample.split(data_nb$class,SplitRatio = 0.7)

train_set_nb <- data_nb[split_nb,]

test_set_nb <- data_nb[!split_nb,]

model_nb <- naiveBayes(class~.,data = train_set_nb)

model_nb

# Applying the model on test data

output_nb <- data.frame(Actual = test_set_nb$class,
                        Actual_Binary = ifelse(test_set_nb$class=='Heart Disease',1,0),
                        Prob = predict(model_nb,newdata = test_set_nb,type='raw')[,1] )

output_nb$Pred <- ifelse(output_nb$Prob >=0.5,'Heart Disease','Normal ECG')

output_nb$Pred <- factor(output_nb$Pred)

# Confusion Matrix

cfm_nb <- confusionMatrix(output_nb$Pred,output_nb$Actual)

cfm_svm

ggplotConfusionMatrix(cfm_nn)

prop_nb <- round(table(test_set_nb$class)[1]/(table(test_set_nb$class)[1] + table(test_set_nb$class)[2]),2)

# Lift Chart

rlift.df.nb <- data.frame(x = c(1:135),
                          y = cumsum(output_nb$Actual_Binary[order(output_nb$Prob, decreasing = T)]),
                          bench = c(1:135)*prop_nb)

ggplot(rlift.df.nb, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") + ggtitle("Lift chart-SVM") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and determining AUC

roc_nb <- roc(output_nb$Actual,output_nb$Prob)

plot.roc(roc_nb,print.auc = T,grid = T)
```

```
 #-------------------------------------------Classification Trees-----------------------------------------------

# Partitioning data into train and test

set.seed(100)

split_dt <- sample.split(data_after_pca$class,SplitRatio = 0.7)

train_set_dt <- data_after_pca[split_dt,]

test_set_dt <- data_after_pca[!split_dt,]

# Performing 10-fold cross validation

control_dt <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 3)


tGrid_dt <- expand.grid(cp = seq(0, .025, .0001))

model_dt <- train(class~., data = train_set_dt,method = "rpart", metric = "Accuracy", trControl=control_dt,
tuneLength = 100)

# Results of 10-fold cross validation
```

```
model_dt

# Optimal value of cp

paste('Optimal Value of cp after cross validation is ',as.character(model_dt$bestTune$cp))

# Model built using the best parameters

model_dt_best <- model_dt$finalModel

# Plotting the best pruned tree model

prp(model_dt_best,split.font = 2,type = 1,extra = 2,varlen = -15)

# Applying the model on test data

output_dt <- data.frame(Actual = test_set_dt$class, Pred = predict(model_dt,newdata = test_set_dt),
                     Prob = round(predict(model_dt,newdata = test_set_dt,type = 'prob'),2))

output_dt$Actual_Binary <- ifelse(output_dt$Actual == 'Heart Disease',1,0)

# Confusion Matrix

cfm_dt <- confusionMatrix(output_dt$Actual,output_dt$Pred)

cfm_dt

ggplotConfusionMatrix(cfm_dt)

prop_dt <- round(table(test_set_dt$class)[1]/(table(test_set_dt$class)[1] + table(test_set_dt$class)[2]),2)

# Plotting Lift Chart

rlift.df.dt <- data.frame(x = c(1:135),
                          y = cumsum(output_dt$Actual_Binary[order(output_dt$Prob.Heart.Disease, decreasing =
T)]),
                          bench = c(1:135)*prop_dt)

ggplot(rlift.df.dt, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-Decision Tree") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and finding AUC

roc_dt<- roc(output_dt$Actual,output_dt$Prob.Heart.Disease)

plot.roc(roc_dt,print.auc = T,grid = T)

#--------------------------------------------Random forest------------------------------------------------------


# Partioning data into train and test

split_rf <- sample.split(data_after_pca$class,SplitRatio = 0.7)

train_set_rf <- data_after_pca[split_rf,]

test_set_rf <- data_after_pca[!split_rf,]

# Performing 10-fold cross validation

control_rf <- trainControl(method="repeatedcv", number=10, repeats=3,search = 'grid')

set.seed(100)

tunegrid <- expand.grid(.mtry=c(1:15))

model_rf <- train(class~., data=train_set_rf, method="rf", metric="Accuracy", tuneGrid=tunegrid,
trControl=control_rf)

# Results of 10-fold cross validation
```

```r
model_rf

# plotting mtry v/s Accuracy

plot(model_rf)

# Applying the best model to test data

output_rf <- data.frame(Actual = test_set_rf$class,Prob = predict(model_rf,test_set_rf,type = 'prob'))

output_rf$Pred <- ifelse(output_rf$Prob.Heart.Disease >= 0.5,'Heart Disease','Normal ECG')

output_rf$Pred <- factor(output_rf$Pred)

output_rf$Actual_Binary <- ifelse(output_rf$Actual == 'Heart Disease',1,0)

# Confusion Matrix

cfm_rf <- confusionMatrix(output_rf$Pred,output_rf$Actual)

cfm_rf

ggplotConfusionMatrix(cfm_rf)

prop_rf <- round(table(test_set_rf$class)[1]/(table(test_set_rf$class)[1] + table(test_set_rf$class)[2]),2)

# Lift Chart


rlift.df.rf <- data.frame(x = c(1:135),
                       y = cumsum(output_rf$Actual_Binary[order(output_rf$Prob.Heart.Disease, decreasing = T)]),
                       bench = c(1:135)*prop_rf)

ggplot(rlift.df.rf, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-Random Forest") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and determining AUC

roc_rf <- roc(output_rf$Actual,output_rf$Prob.Heart.Disease)

plot.roc(roc_rf,print.auc = T,grid = T)

#--------------------------------------------------Boosted Trees---------------------------------------------

# Partitioning the data into train and test

split_bt <- sample.split(data_after_pca$class,SplitRatio = 0.7)

train_set_bt <- data_after_pca[split_bt,]

test_set_bt <- data_after_pca[!split_bt,]

# Model Training

model_bt <- boosting(class ~ ., data = train_set_bt)

# Applying model to test data

output_bt <- data.frame(Actual = test_set_bt$class,Prob = predict(model_bt,test_set_bt)$prob, Pred
=predict(model_bt,test_set_bt)$class)

output_bt$Actual_Binary <- ifelse(output_bt$Actual=='Heart Disease',1,0)

# Confusion Matrix

cfm_bt <- confusionMatrix(output_bt$Pred,output_bt$Actual)

cfm_bt

ggplotConfusionMatrix(cfm_bt)
```

```
prop_bt <- round(table(test_set_bt$class)[1]/(table(test_set_bt$class)[1] + table(test_set_bt$class)[2]),2)

# Lift Chart

rlift.df.bt <- data.frame(x = c(1:135),
                          y = cumsum(output_bt$Actual_Binary[order(output_bt$Prob.1, decreasing = T)]),
                          bench = c(1:135)*prop_bt)

ggplot(rlift.df.bt, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart- Boosted Tree") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and determining AUC

roc_bt <- roc(output_bt$Actual,output_bt$Prob.1)

plot.roc(roc_bt,print.auc = T,grid = T)


#---------------------------------------------Logistic Regression------------------------------------------------

# Chi Square Test of Independence

chi_test_result <- data.frame(Pair = rep(NA,4096),P_Value = rep(NA,4096))

p = 0
for(i in 1:64){
  for(j in 1:64){
    tab <- chisq.test(table(data_after_pca[,i],data_after_pca[,j]))
    chi_test_result[j+p,1] <- paste(as.character(i),'-',as.character(j))
    chi_test_result[j+p,2] <- round(tab$p.value,5)
    q = j+p
  }
  p = q
}

# Displaying results having p value less than 0.05. If p is less than 0.05, we reject the null hypothesis that
# two variables are independent

head(chi_test_result[chi_test_result$P_Value <0.05,])

# Removing collinear categorical columns based on Chi Square Test of Independence

data_lr <- data_after_pca[,c(1:6,10,14:16,18,19,21,23,26,28,31,33,34,36,37,38,41,53,55,61,62,65:106)]

# Removing categorical constant columns

data_lr <- data_lr[,-c(7,10,16)]

# Partitioning data into train and test

set.seed(100)

split_lr <- sample.split(data_lr$class,SplitRatio = 0.7)

train_set_lr <- data_lr[split_lr,]

test_set_lr <- data_lr[!split_lr,]

model_lr <- glm(class~.,data = train_set_lr,family = 'binomial')

summary(model_lr)

model_lr1 <-
glm(class~sex+chDI_RRwaveExists+chDI_DD_RRwaveExists+chDI_RPwaveExists+chDI_DD_RPwaveExists+chDI_RTwaveExists+

chDIII_RRwaveExists+chDIII_DD_RRwaveExists+chDIII_RTwaveExists+chDIII_DD_RTwaveExists+chAVR_DD_RRwaveExists+

chAVR_DD_RPwaveExists+chAVL_DD_RRwaveExists+chAVF_RRwaveExists+chAVF_DD_RPwaveExists+chAVF_RTwaveExists+

chV1_RRwaveExists+chV1_DD_RRwaveExists+chV1_RPwaveExists+chV1_DD_RTwaveExists+chV3_DD_RTwaveExists+
                   chV4_DD_RRwaveExists+chV6_RRwaveExists+chV6_DD_RRwaveExists+poly( PC1 ,2 )+poly( PC2 ,2
)+poly( PC3 ,2 )+
                   poly( PC4 ,2 )+poly( PC5 ,2 )+poly( PC6 ,2 )+poly( PC7 ,2 )+poly( PC8 ,2 )+poly( PC9 ,2
)+poly( PC10 ,2 )+
                   poly( PC11 ,2 )+poly( PC12 ,2 )+poly( PC13 ,2 )+poly( PC14 ,2 )+poly( PC15 ,2 )+poly( PC16 ,2
)+
                   poly( PC17 ,2 )+poly( PC18 ,2 )+poly( PC19 ,2 )+poly( PC20 ,2 )+poly( PC21 ,2 )+poly( PC22 ,2
)+
                   poly( PC23 ,2 )+poly( PC24 ,2 )+poly( PC25 ,2 )+poly( PC26 ,2 )+poly( PC27 ,2 )+poly( PC28 ,2
)+
                   poly( PC29 ,2 )+poly( PC30 ,2 )+poly( PC31 ,2 )+poly( PC32 ,2 )+poly( PC33 ,2 )+poly( PC34 ,2
)+
                   poly( PC35 ,2 )+poly( PC36 ,2 )+poly( PC37 ,2 )+poly( PC38 ,2 )+poly( PC39 ,2 )+poly( PC40 ,2
)+
                   poly( PC41 ,2 ),data = train_set_lr,family = 'binomial')

summary(model_lr1)

stat_significant_cols <-  data.frame(summary(model_lr1)$coef[summary(model_lr1)$coef[,4] <= .05, 4])

model_lr1 <- glm(class~sex+chDI_RRwaveExists+poly(PC1, 2)+poly(PC2, 2)+poly(PC3, 2)+poly(PC4, 2)+poly(PC6, 2)+
                   poly(PC9, 2)+poly(PC10, 2)+poly(PC11, 2)+poly(PC14, 2)+poly(PC16, 2)+poly(PC19, 2)+poly(PC21,
2)+
                   poly(PC21, 2)+poly(PC23, 2)+poly(PC25, 2)+poly(PC32, 2)+poly(PC33, 2)+poly(PC35,
2)+poly(PC36, 2)+
                   poly(PC38, 2)+poly(PC39, 2)+poly(PC40, 2),data = train_set_lr,family = 'binomial')


output_lr <-  data.frame(Actual = test_set_lr$class, Pred.Prob =
round(predict(model_lr1,test_set_lr[,c(1,2,25,26:29,31,34,35,36,39,41,44,46,48,50,57,58,60,61,63,64,65)],type =
'response'),3))

output_lr$Pred <- ifelse(output_lr$Pred.Prob >=0.5,'Heart Disease','Normal ECG')

output_lr$Pred <- factor(output_lr$Pred)

output_lr$Actual_Binary <- ifelse(output_lr$Actual=='Heart Disease',1,0)

# Confusion Matrix

cfm_lr <- confusionMatrix(output_lr$Pred,output_lr$Actual)

cfm_lr

ggplotConfusionMatrix(cfm_lr)

prop_lr <- round(table(test_set_lr$class)[1]/(table(test_set_lr$class)[1] + table(test_set_lr$class)[2]),2)

# Lift Chart

rlift.df.lr <- data.frame(x = c(1:135),
                       y = cumsum(output_lr$Actual_Binary[order(output_lr$Pred.Prob, decreasing = T)]),
                       bench = c(1:135)*prop_lr)

ggplot(rlift.df.lr, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-Logistic Regression") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and determining AUC

roc_lr <- roc(output_lr$Actual,output_lr$Pred.Prob)

plot.roc(roc_lr,print.auc = T,grid = T)

#----------------------------------------LDA-------------------------------------------------------------

# Chi Square Test of Independence

chi_test_result <- data.frame(Pair = rep(NA,4096),P_Value = rep(NA,4096))

p = 0
for(i in 1:64){
  for(j in 1:64){
    tab <- chisq.test(table(data_after_pca[,i],data_after_pca[,j]))
    chi_test_result[j+p,1] <- paste(as.character(i),'-',as.character(j))
    chi_test_result[j+p,2] <- tab$p.value
    q = j+p
  }
  p = q
}

head(chi_test_result)

# Removing collinear categorical columns based on Chi Square Test of Independence

data_lda <- data_after_pca[,c(1:6,10,14:16,18,19,21,23,26,28,31,33,34,36,37,38,41,53,55,61,62,65:106)]

# Removing categorical constant columns

data_lda <- data_lda[,-c(7,10,16)]

# Convering to m-1 dummy binary varaibles

for(i in 1:24){
  data_lda[,i] <- as.numeric(as.character(data_lda[,i]))
}

# Partitioning data into train and test

set.seed(100)

split_lda <- sample.split(data_lda$class,SplitRatio = 0.7)

train_set_lda <- data_lda[split_lda,]

test_set_lda <- data_lda[!split_lda,]

model_lda <- lda(class~.,data = train_set_lda)

output_lda <- data.frame(Actual=test_set_lda$class,Pred = predict(model_lda,test_set_lda)$class, Prob =
round(predict(model_lda,test_set_lda)$posterior,3))

output_lda$Actual_Binary <- ifelse(output_lda$Actual=='Heart Disease',1,0)

# Confusion Matrix

cfm_lda <- confusionMatrix(output_lda$Pred,output_lda$Actual)

cfm_lda

ggplotConfusionMatrix(cfm_lda)

prop_lda <- round(table(test_set_lda$class)[1]/(table(test_set_lda$class)[1] + table(test_set_lda$class)[2]),2)

# Lift Chart

rlift.df.lda <- data.frame(x = c(1:135),
                       y = cumsum(output_lda$Actual_Binary[order(output_lda$Prob.Heart.Disease, decreasing =
T)]),
                       bench = c(1:135)*prop_lda)

ggplot(rlift.df.lda, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-LDA") + xlab('Cases') + ylab('Cumulative')


# Plotting ROC and determining AUC

roc_lda <- roc(output_lda$Actual,output_lda$Prob.Heart.Disease)

plot.roc(roc_lda,print.auc = T,grid = T)

#---------------------------------------------Neural Nets-----------------------------------------------------
```

```r
data_nn <- data_after_pca

for(i in 1:64){
  data_nn[,i] <- as.numeric(as.character(data_nn[,i]))
}

data_nn$class <- ifelse(data$class=='Heart Disease',1,0)

# Normalizing the data

normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}

data_nn[,66:106] <- apply(data_nn[,66:106],2,normalize)

set.seed(100)

split_nn <- sample.split(data_nn$class,SplitRatio = 0.7)

train_set_nn <- data_nn[split_nn,]

test_set_nn <- data_nn[!split_nn,]

model_nn <- neuralnet(class~.,data = train_set_nn,err.fct = "ce",linear.output = FALSE,stepmax = 2e+06,hidden =
1)

output_nn <- data.frame(Actual = ifelse(test_set_nn$class ==1,'Heart Disease','Normal ECG'),
                        Actual_Binary = test_set_nn$class,
                        Pred.Prob = compute(model_nn,test_set_nn[,-65])$net.result)

output_nn$Pred <- ifelse(output_nn$Pred.Prob >=0.5,'Heart Disease','Normal ECG')

output_nn$Pred <- factor(output_nn$Pred)

# Confusion Matrix

cfm_nn <- confusionMatrix(output_nn$Pred,output_nn$Actual)

cfm_nn

ggplotConfusionMatrix(cfm_nn)

prop_nn <- round(table(test_set_nn$class)[2]/(table(test_set_nn$class)[1] + table(test_set_nn$class)[2]),2)

# Lift Chart

rlift.df.nn <- data.frame(x = c(1:135),
                           y = cumsum(output_nn$Actual_Binary[order(output_nn$Pred.Prob, decreasing = T)]),
                           bench = c(1:135)*prop_nn)

ggplot(rlift.df.nn, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-Neural Nets") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and determining AUC

roc_nn <- roc(output_nn$Actual,output_nn$Pred.Prob)

plot.roc(roc_nn,print.auc = T,grid = T)


# ---------------------------------------------------SVM-------------------------------------------------------

set.seed(100)

split_svm <- sample.split(data_after_pca$class,SplitRatio = 0.7)

train_set_svm <- data_after_pca[split_svm,]

test_set_svm <- data_after_pca[!split_svm,]
```

```r
model_svm <- svm(class~.,data = train_set_svm,probability = TRUE)

summary(model_svm)

pred_svm <- predict(model_svm,test_set_svm,probability = TRUE)

output_svm <- data.frame(Actual = test_set_svm$class,
                        Actual_Binary = ifelse(test_set_svm$class=='Heart Disease',1,0),
                        Prob = attr(pred_svm,"probabilities")[,1] )

output_svm$Pred <- ifelse(output_svm$Prob>=0.5,'Heart Disease','Normal ECG')

output_svm$Pred <- factor(output_svm$Pred)

# Confusion Matrix

cfm_svm <- confusionMatrix(output_svm$Pred,output_svm$Actual)

cfm_svm

ggplotConfusionMatrix(cfm_nn)

prop_svm <- round(table(test_set_svm$class)[1]/(table(test_set_svm$class)[1] + table(test_set_svm$class)[2]),2)

# Lift Chart

rlift.df.svm <- data.frame(x = c(1:135),
                          y = cumsum(output_svm$Actual_Binary[order(output_svm$Prob, decreasing = T)]),
                          bench = c(1:135)*prop_svm)

ggplot(rlift.df.svm, aes(x = x)) + geom_line(aes(y= y), color = "blue") + geom_line(aes(y = bench), color =
"red", lty = "dashed") +
  ggtitle("Lift chart-SVM") + xlab('Cases') + ylab('Cumulative')

# Plotting ROC and determining AUC

roc_svm <- roc(output_svm$Actual,output_svm$Prob)

plot.roc(roc_svm,print.auc = T,grid = T)
```

Extra Resources used for this project:

- https://geekymedics.com/understanding-an-ecg/
- https://link.springer.com/chapter/10.1007/978-1-60327-372-5_17
- http://www.sthda.com/english/wiki/chi-square-test-of-independence-in-r
- https://www.r-bloggers.com/to-eat-or-not-to-eat-thats-the-question-measuring-the-association-between-categorical-variables/