Northeastern University

College of Engineering

IE 7275 DATA MINING IN ENGINEERING

HEART DISEASE CLASSIFICATION

CASE STUDY REPORT

BY

NIKITA PAI - 001347892

&

ABHILASH HEMARAJ – 001305283

GROUP NO: 12

SUBMITTED TO PROF. XUEMIN JIN

I. Background and Introduction

Heart disease, also referred to as Cardiovascular disease, covers a range of conditions that affect your heart. These conditions include arrhythmia, cyanosis, coronary artery disease, congenital heart defects, stress, valvular heart disease, high blood pressure, etc.

Symptoms of various heart disease:
Although symptoms vary with different conditions and complications, the following are the most common:

- Chest Pain, discomfort
- Rapid or Slow heartbeat (Tachycardia and Bradycardia resp.)
- Dizziness
- Fever
- Weakness or fatigue
- Fainting

Heart Disease has been reported as the **leading** cause of death for people in the United Sates disregarding any gender or most racial groupings. There is a loss of life every 37 seconds due to various heart diseases/ conditions.

It is estimated that **1** in every **4** deaths is due to some for of heart disease/ heart condition.

Looking at the scale of the problem its is important for us to address it. As so it happens it is difficult to categorize a patient with any of the said symptoms as having heart disease or not. The major reason is that there are a lot of related factors that contribute to the same such as blood pressure, cholesterol, blood sugar level, etc.

Hence, we turn to various Data mining Techniques to help with the issue. Data mining is the process of discovering patterns in large data sets using Machine Learning. We are going to apply popular machine learning algorithms to the dataset. But first let’s have a look at the dataset.

The dataset originates from the popular UCI Repository, labelled as Heart disease dataset. The data most widely used from the repository is that of Cleveland database. The original dataset consists of 75 attributes and 303 instances. The data is multivariate, categorical and consists of real numbers and integers.

Basic Data Profiling Report

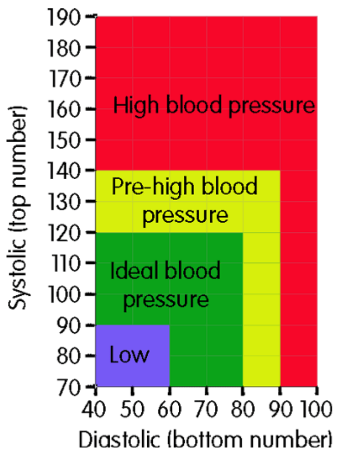
Name	**Value**
Rows	303
Columns	14
Discrete columns	0
Continuous columns	14
All missing columns	0
Missing observations	0
Complete Rows	303
Total observations	4,242
Memory allocation	36.7 Kb

The Attributes:
1] Age: Target range is between 30 to 60

2] Sex: Male =1; Female = 0

3] Cp: chest pain type (4 values) ranging from 0 to 3
- Value 0: asymptomatic
- Value 1: atypical angina
- Value 2: non-anginal pain
- Value 3: typical angina

4] Tresbps: resting blood pressure in mm Hg on admission to the hospital



5] hol: serum cholesterol in mg/dl

6] Fbs: If fasting blood sugar > 120 mg/dl, then value = 1 or value = 0

7] Restecg: resting electrocardiographic results (values 0,1,2)

- Value 0: showing probable or definite left ventricular hypertrophy by Estes' criteria
- Value 1: normal
- Value 2: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

8] Thalach: maximum heart rate achieved

9] Exang: exercise induced angina (yes =1; no = 0)

10] old peak: ST depression induced by exercise relative to rest

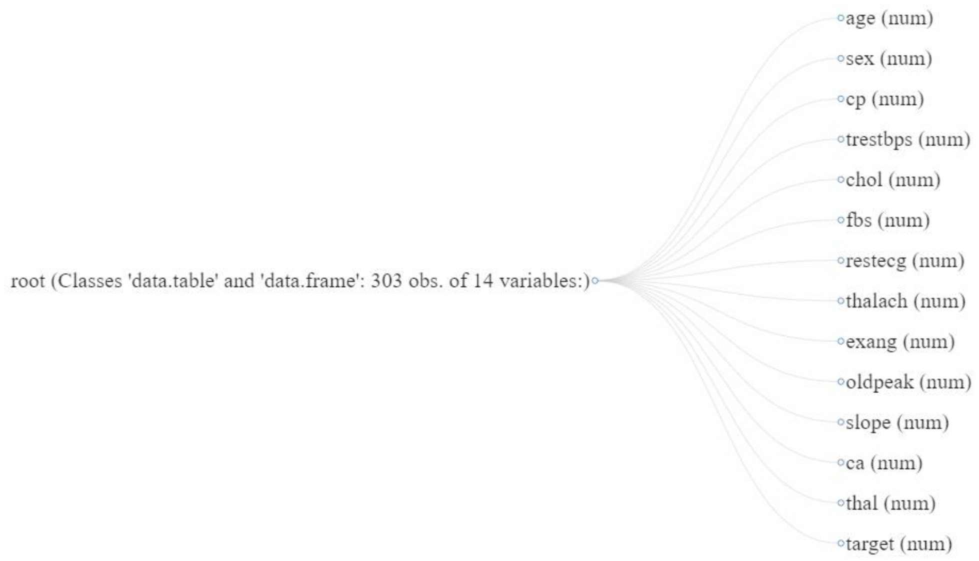
- 0: down sloping
- 1: flat
- 2: up sloping

11] slope: the slope of the peak exercise ST segment ranging from 0 to 2

12] ca: number of major vessels (0-3) colored by fluoroscopy

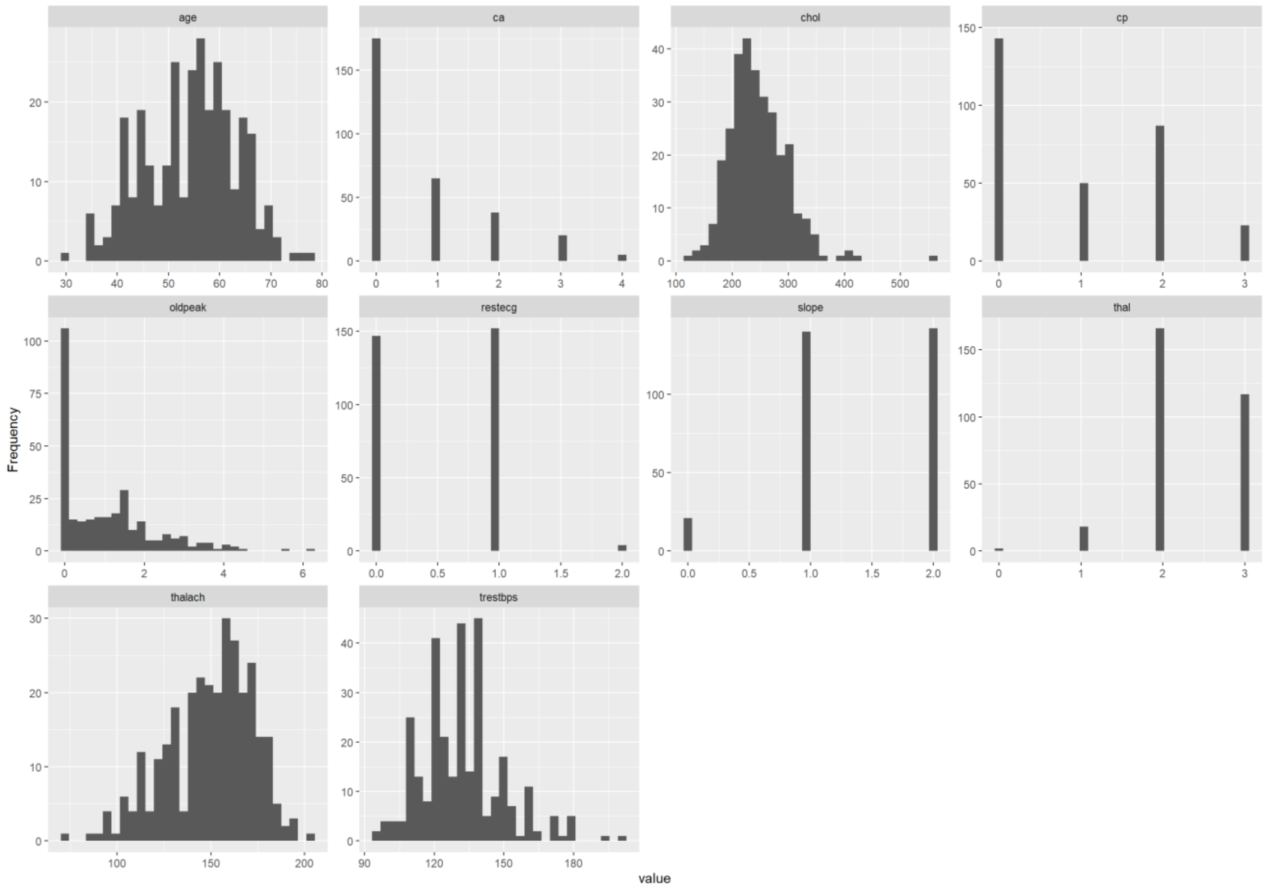
13] thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

14] target: 0 or 1

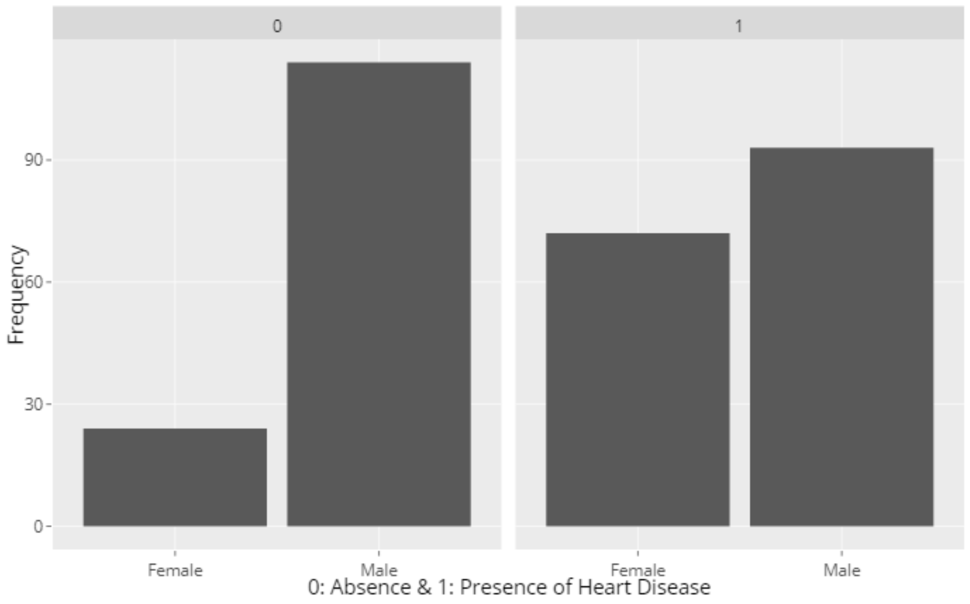


II. Data Exploration and Visualization

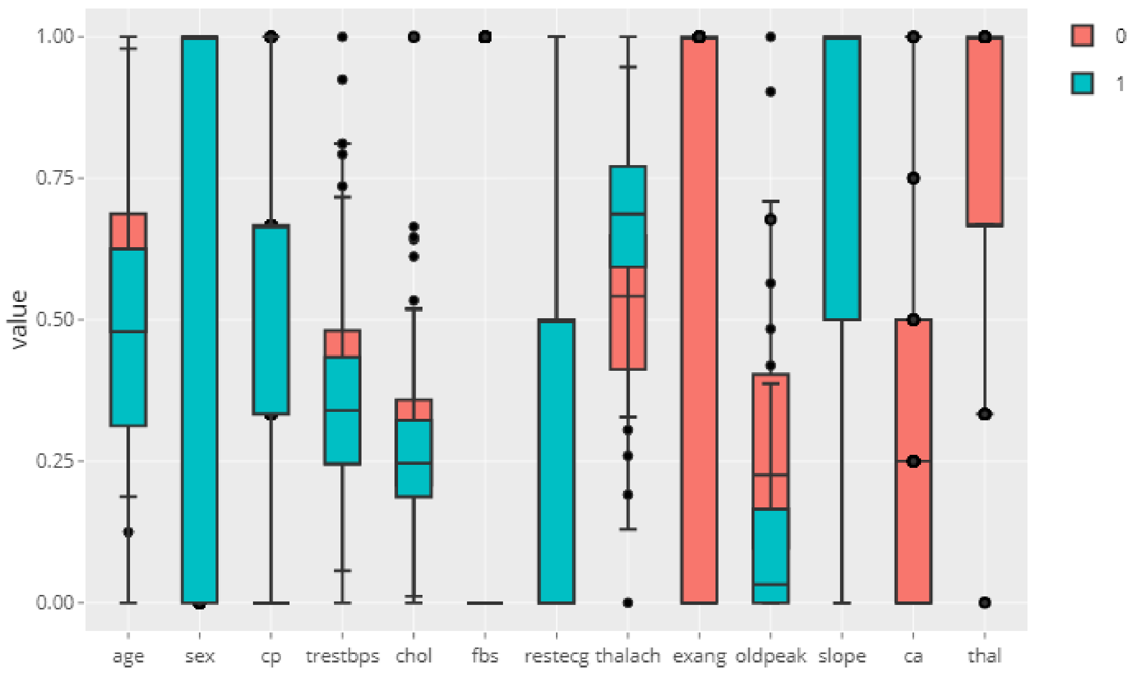
Histogram: Univariate Distribution



Sample Bias with respect to Sex:



We can see that the dataset follows a univariate distribution, wherein variables such as age, tresbps, thalach follow distributions close to normal. There are no missing values in the dataset. But we could find some sampling bias with respect to sex in the dataset wherein the number of Male patients in the sample outnumber the number of female patients.



We can also notice from the above boxplot that there are certain variables that exhibit outliers, this can be explained by the fact that these factors may be influenced by various other health related factors and that every patient's characteristics would differ.

III. Data Preparation and Preprocessing

Pre-processing performed on the dataset:

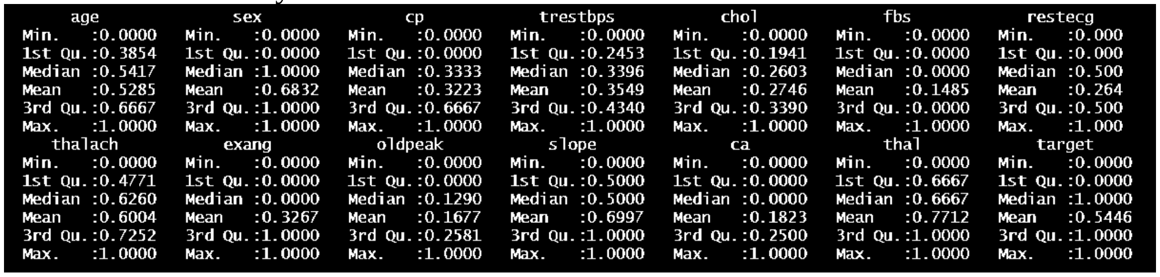
- Feature Scaling
- Correlation Analysis
- PCA
- Cluster Analysis

Feature Scaling

The original dataset consisted of 75 attributes which has been brought down to 14 essential factors that has been said to be most relevant. We would perform feature scaling specifically normalizing the dataset. Normalization is the process of scaling the variables down so that the scaled variables range from 0 to 1.

The main driving reason for us to apply normalization is that certain machine learning algorithms are sensitive to the magnitudes of the variables in the dataset. This happens because certain models are more sensitive to the distance between the data points, and hence by taking out the difference in magnitude we give a fair chance for the models to perform better at classification.

Below is the summary table for normalized variables:



```
      age              sex               cp            trestbps           chol              fbs            restecg     
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
 1st Qu.:0.3854   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.2453   1st Qu.:0.1941   1st Qu.:0.0000   1st Qu.:0.000  
 Median :0.5417   Median :1.0000   Median :0.3333   Median :0.3396   Median :0.2603   Median :0.0000   Median :0.500  
 Mean   :0.5285   Mean   :0.6832   Mean   :0.3223   Mean   :0.3549   Mean   :0.2746   Mean   :0.1485   Mean   :0.264  
 3rd Qu.:0.6667   3rd Qu.:1.0000   3rd Qu.:0.6667   3rd Qu.:0.4340   3rd Qu.:0.3390   3rd Qu.:0.0000   3rd Qu.:0.500  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.000  
    thalach           exang           oldpeak           slope              ca              thal            target      
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
 1st Qu.:0.4771   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.5000   1st Qu.:0.0000   1st Qu.:0.6667   1st Qu.:0.0000  
 Median :0.6260   Median :0.0000   Median :0.1290   Median :0.5000   Median :0.0000   Median :0.6667   Median :1.0000  
 Mean   :0.6004   Mean   :0.3267   Mean   :0.1677   Mean   :0.6997   Mean   :0.1823   Mean   :0.7712   Mean   :0.5446  
 3rd Qu.:0.7252   3rd Qu.:1.0000   3rd Qu.:0.2581   3rd Qu.:1.0000   3rd Qu.:0.2500   3rd Qu.:1.0000   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000  
```

It is also to be noted that normalization does not affect the distribution of the dataset.

Correlation Analysis

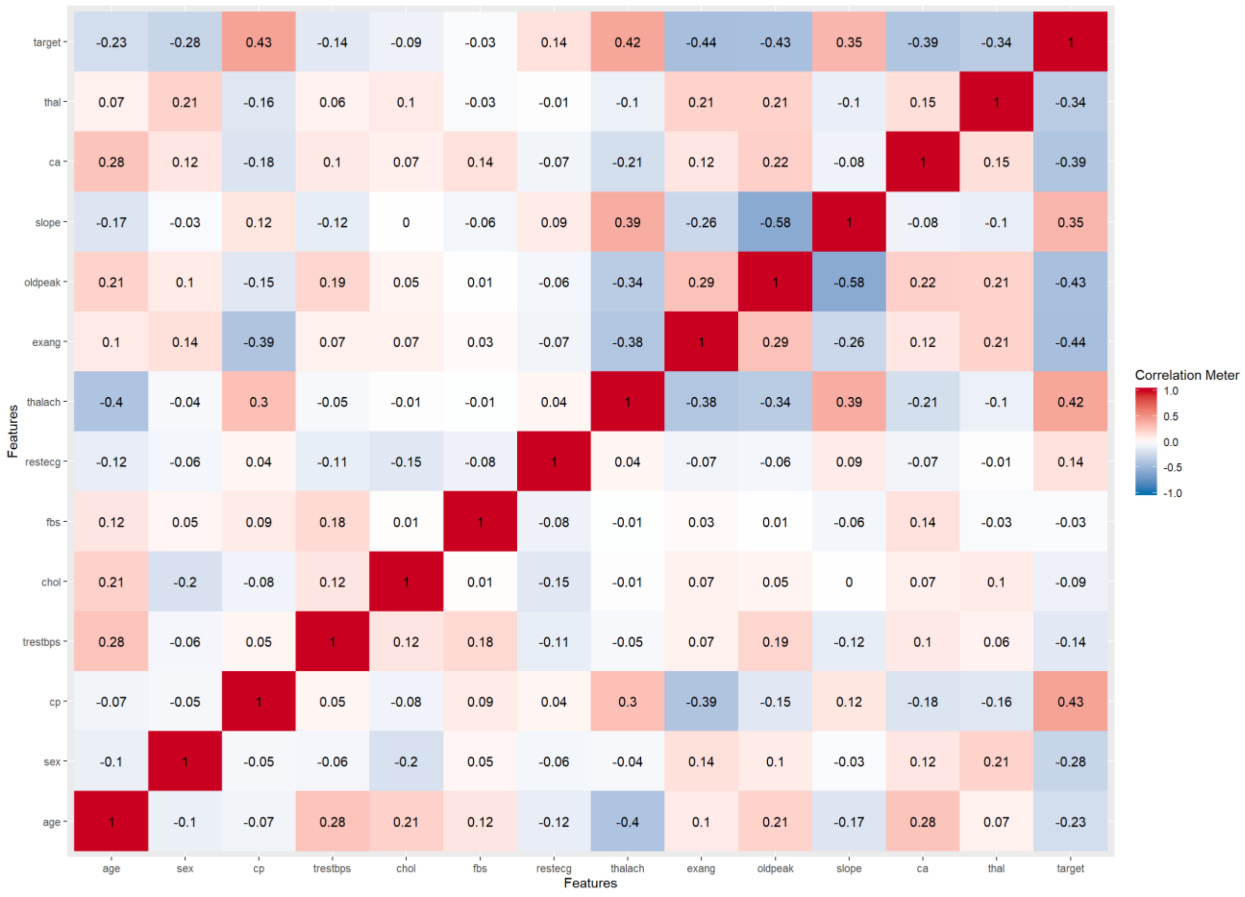
We have used Pearson correlation for our correlation analysis,

$$\rho \quad = \quad \text{Pearson correlation} \quad = \quad \frac{\text{covariance}}{\text{(std of x) (std of y)}}$$

A Pearson correlation ranges from -1 to 1 that indicates the extent to which two variables are linearly related. Looking at the correlation plot we can come to say about the extent to which our features are correlated to each other.

Number that tends to positive 1 means that there is high positive correlation between the variables. If ρ is around zero then we can state that there is little to no correlation between the variables. If ρ tends to negative 1 then there is high negative correlation between the variables.1

Pearson's Correlation is usually a measure that data scientist use to understand how the attributes are inter related and can decide on how one can use the relation on further modeling.



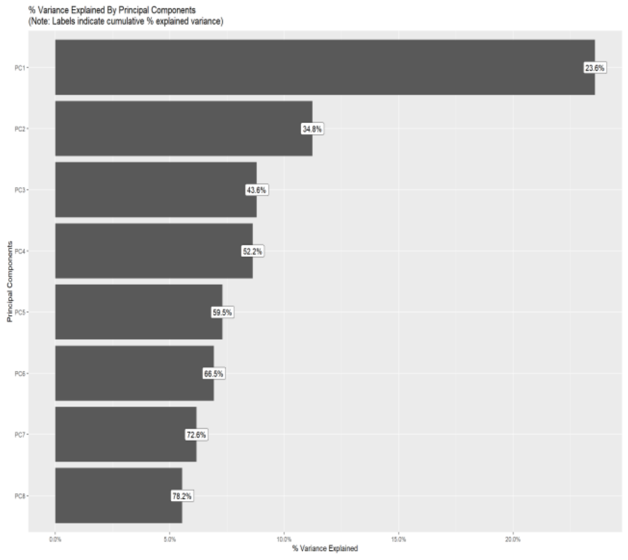
From the above figure we can say that generally there is not much correlation between the variables in hand. The maximum positive correlation is exhibited by cp ~ target variables. The maximum negative correlation is between slope and old peak. Barring these weak correlations, we can postulate that our data is highly independent and that all variables must be equally considered for the upcoming modeling.

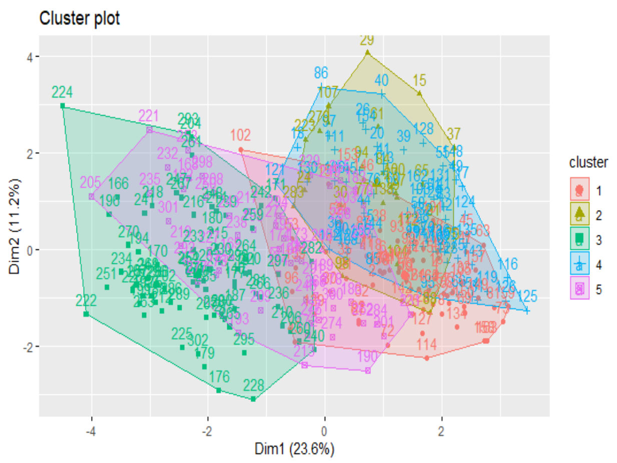
We can comment on attributes independency by looking performing PCA;

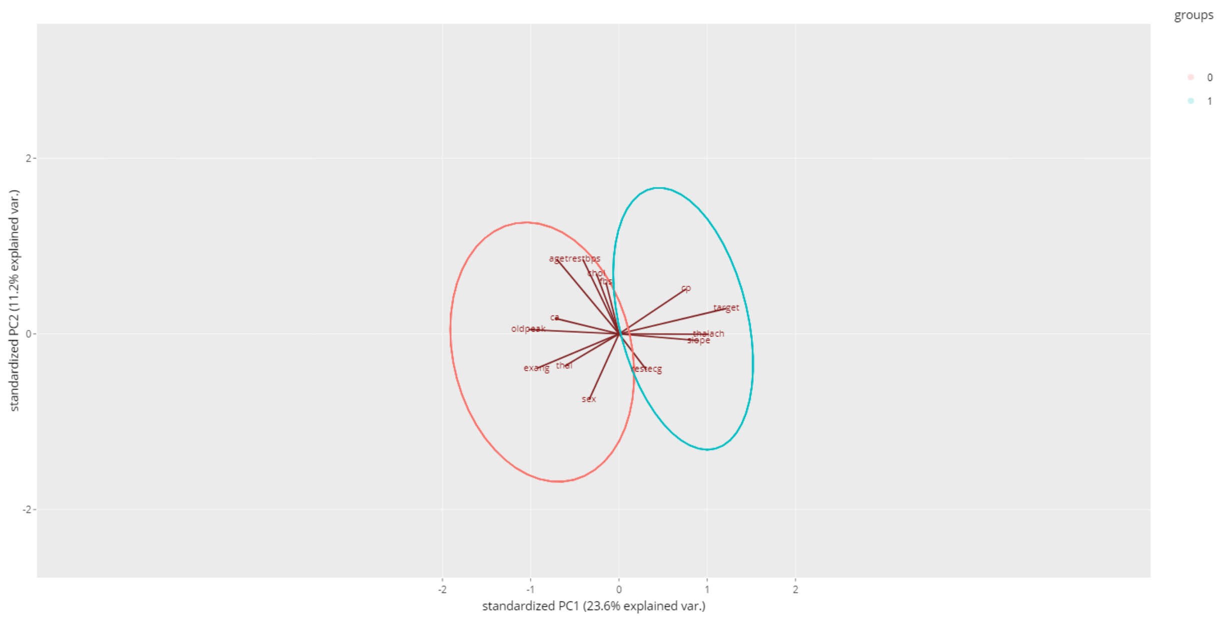
Principal Component Analysis and Cluster Analysis

When there are large number of attributes, PCA helps us to summarize the dataset with a scaled number of representative variables, known as principal components, that preserves most of the variability of the original dataset.

In the below figures we can see the results of PCA and K-means Clustering as an attempt to draw insights from the data.



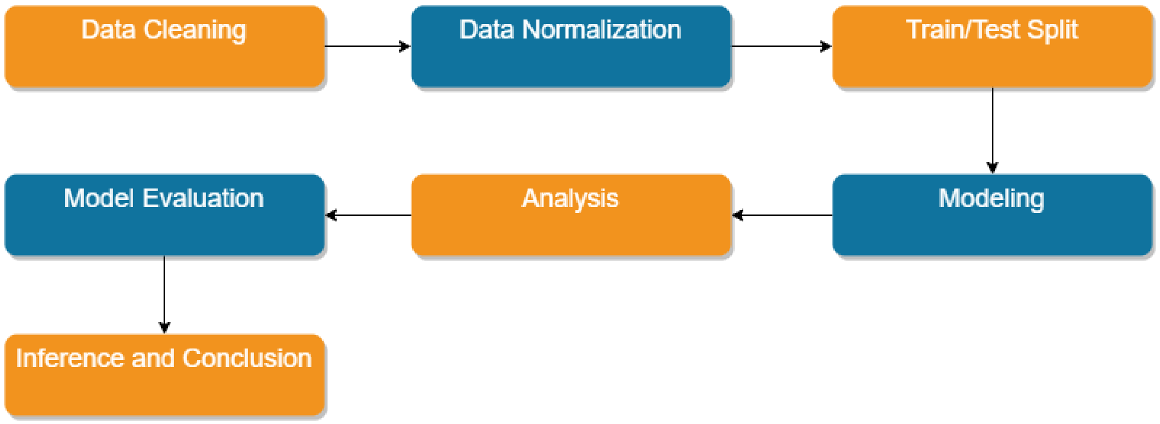




The first two principal components only capture 34.8% of the data. As a matter of fact, even if we consider the first 8 principal components, we end up only covering about 78.2% of the total variance, using which could lead to significant data loss. This is a clear indicator that all our variables are highly independent and hence must contribute to the modeling process without eliminations. This is also evident from cluster plot, wherein there is no clear grouping of the variables due to lot of overlapping characteristics. We achieved clear grouping in the above bi-plot, where target variable was used to group the principal components.

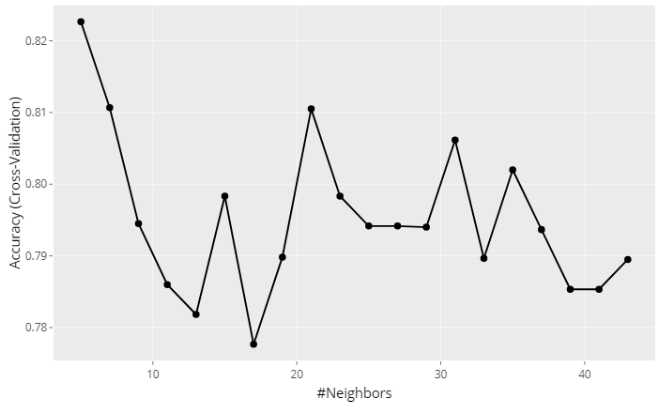
IV. Data Mining Techniques and Implementation

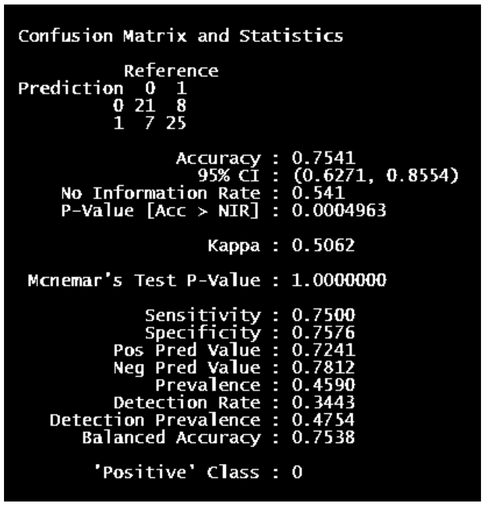
Since the dataset consisted of little to no missing values, we could employ advanced exploratory data analysis and draw insights from the dataset.
We employed data normalization before splitting the dataset into training and testing. The train/ test ratio used for splitting was 80:20.



We implemented the following Data Mining Techniques to cater to our classification problem:

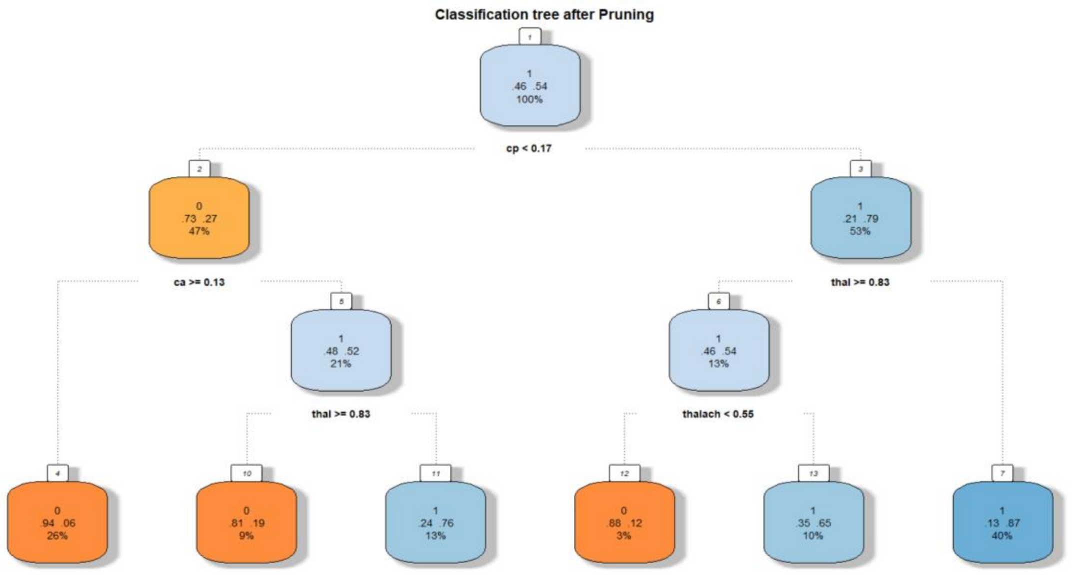
1. K-Nearest Neighbors
   Being a non-parametric learning algorithm, kNN keeps all training examples in memory. Once a new, unseen example is introduced, the algorithm finds k training examples closest to the data and returns the majority label. 10-fold cross validation was performed to achieve k as '5'. The output of our kNN model is as follows: Accuracy achieved: 0.7541



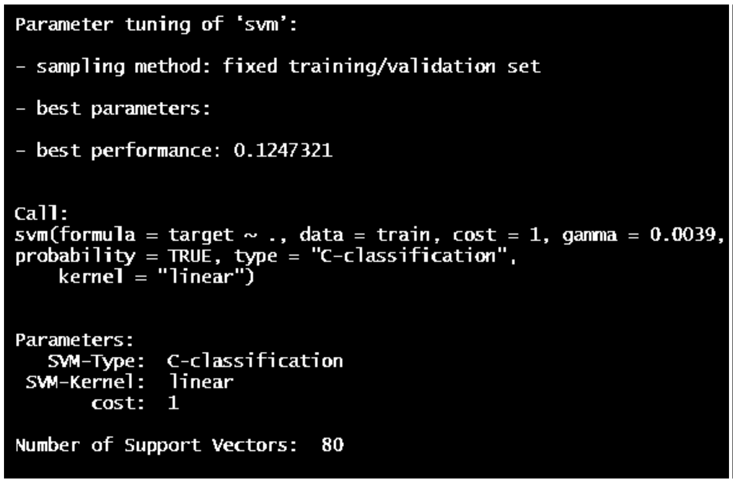


```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 21  8
         1  7 25

               Accuracy : 0.7541
                 95% CI : (0.6271, 0.8554)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 0.0004963

                  Kappa : 0.5062

 Mcnemar's Test P-Value : 1.0000000

            Sensitivity : 0.7500
            Specificity : 0.7576
         Pos Pred Value : 0.7241
         Neg Pred Value : 0.7812
             Prevalence : 0.4590
         Detection Rate : 0.3443
   Detection Prevalence : 0.4754
      Balanced Accuracy : 0.7538

       'Positive' Class : 0
```

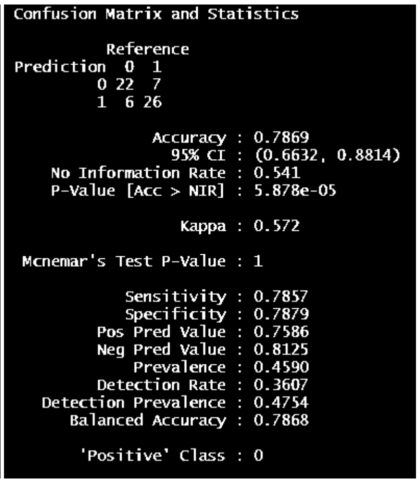
2. Decision Trees (Classification Trees):
   A decision tree is an acyclic graph that can be used to make classifications. In each node of the tree, a specific feature of the feature vector is examined against a threshold. If the value of the feature is below the threshold, then the tree grows through a left branch; otherwise, the right branch is followed. As the leaf node is reached, the decision is made about the class to which the example belongs. The output of the pruned decision tree is as follows. Accuracy achieved: 0.7705



3. Support Vector Machines
   The algorithm builds an SVM model using the labeled data. The model classifies the datapoints present in an multi-dimensional space using hyperplanes. The output of our SVM model is as follows:



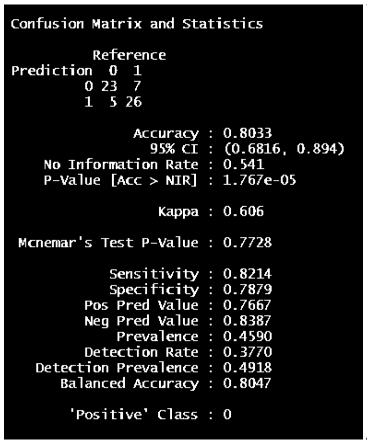
```
Parameter tuning of 'svm':

- sampling method: fixed training/validation set

- best parameters:

- best performance: 0.1247321


Call:
svm(formula = target ~ ., data = train, cost = 1, gamma = 0.0039,
probability = TRUE, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  80
```



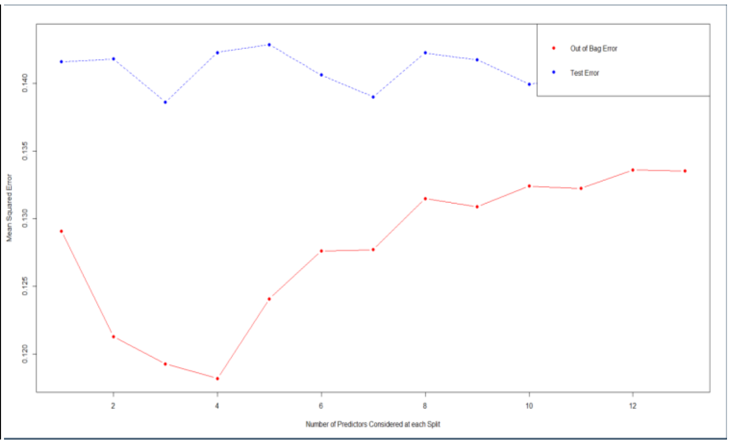
```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 22  7
         1  6 26

               Accuracy : 0.7869
                 95% CI : (0.6632, 0.8814)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 5.878e-05

                  Kappa : 0.572

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.7857
            Specificity : 0.7879
         Pos Pred Value : 0.7586
         Neg Pred Value : 0.8125
             Prevalence : 0.4590
         Detection Rate : 0.3607
   Detection Prevalence : 0.4754
      Balanced Accuracy : 0.7868

       'Positive' Class : 0
```

4. Random Forests

   In Random Forests the idea is to decorrelate the several trees which are generated by the different bootstrapped samples from training Data. It uses a modified decision tree learning algorithm that inspects, at each split, a random subset of the features. The reason for doing this is to avoid the correlation of the trees. And at the end reducing the variance in the trees.

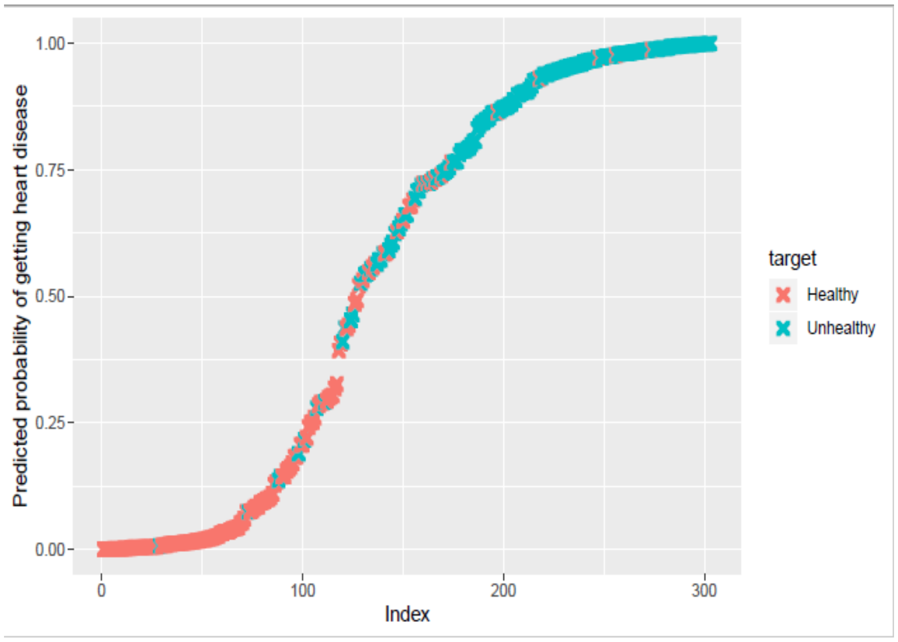


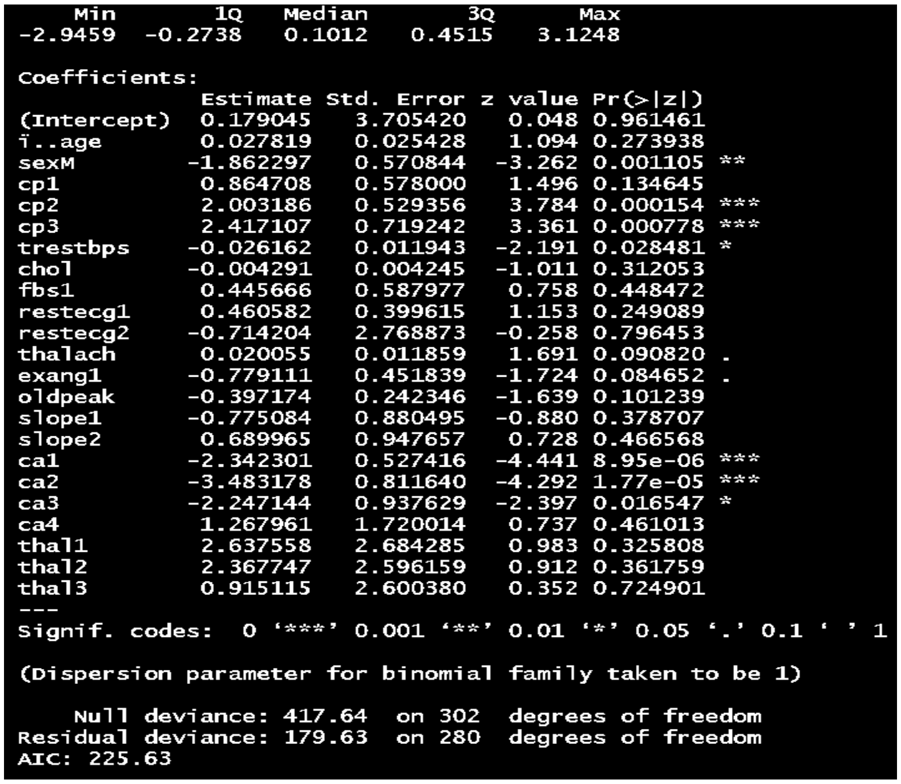
```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 23  7
         1  5 26

               Accuracy : 0.8033
                 95% CI : (0.6816, 0.894)
    No Information Rate : 0.541
    P-Value [Acc > NIR] : 1.767e-05

                  Kappa : 0.606

 Mcnemar's Test P-Value : 0.7728

            Sensitivity : 0.8214
            Specificity : 0.7879
         Pos Pred Value : 0.7667
         Neg Pred Value : 0.8387
             Prevalence : 0.4590
         Detection Rate : 0.3770
   Detection Prevalence : 0.4918
      Balanced Accuracy : 0.8047

       'Positive' Class : 0
```



5. Logistic Regression

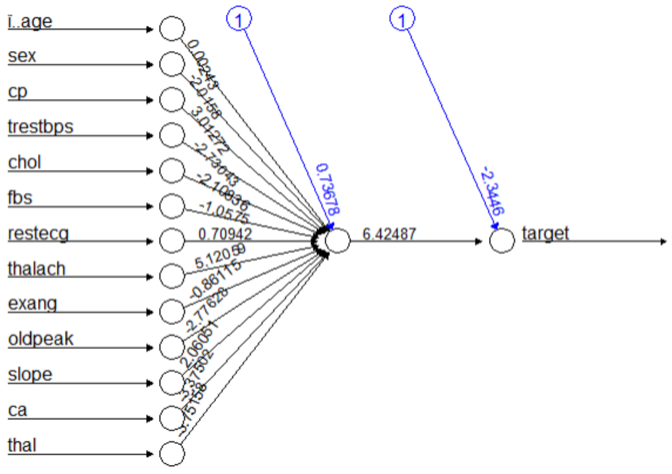
   Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

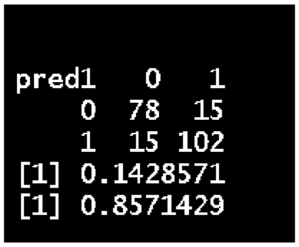




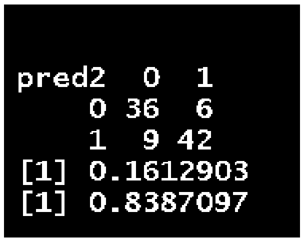
```
    Min       1Q    Median       3Q      Max
-2.9459  -0.2738   0.1012   0.4515   3.1248

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.179045   3.705420   0.048 0.961461
ï..age       0.027819   0.025428   1.094 0.273938
sexM        -1.862297   0.570844  -3.262 0.001105 **
cp1          0.864708   0.578000   1.496 0.134645
cp2          2.003186   0.529356   3.784 0.000154 ***
cp3          2.417107   0.719242   3.361 0.000778 ***
trestbps    -0.026162   0.011943  -2.191 0.028481 *
chol        -0.004291   0.004245  -1.011 0.312053
fbs1         0.445666   0.587977   0.758 0.448472
restecg1     0.460582   0.399615   1.153 0.249089
restecg2    -0.714204   2.768873  -0.258 0.796453
thalach      0.020055   0.011859   1.691 0.090820 .
exang1      -0.779111   0.451839  -1.724 0.084652 .
oldpeak     -0.397174   0.242346  -1.639 0.101239
slope1      -0.775084   0.880495  -0.880 0.378707
slope2       0.689965   0.947657   0.728 0.466568
ca1         -2.342301   0.527416  -4.441 8.95e-06 ***
ca2         -3.483178   0.811640  -4.292 1.77e-05 ***
ca3         -2.247144   0.937629  -2.397 0.016547 *
ca4          1.267961   1.720014   0.737 0.461013
thal1        2.637558   2.684285   0.983 0.325808
thal2        2.367747   2.596159   0.912 0.361759
thal3        0.915115   2.600380   0.352 0.724901
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 417.64  on 302  degrees of freedom
Residual deviance: 179.63  on 280  degrees of freedom
AIC: 225.63
```

6. Neural Networks
   Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input.





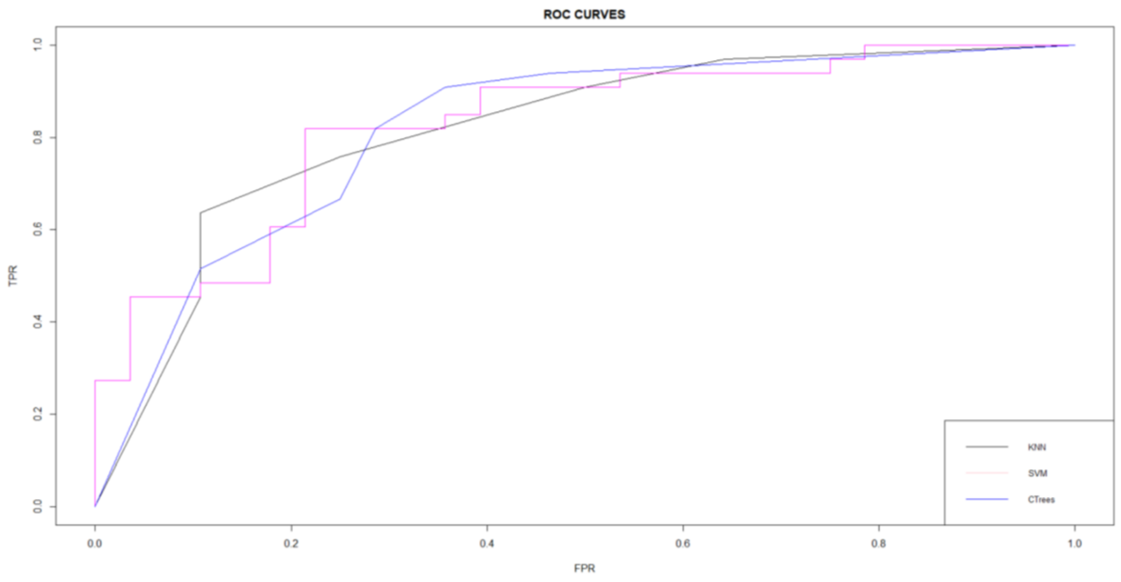
```
pred1   0   1
    0  78  15
    1  15 102
[1] 0.1428571
[1] 0.8571429
```

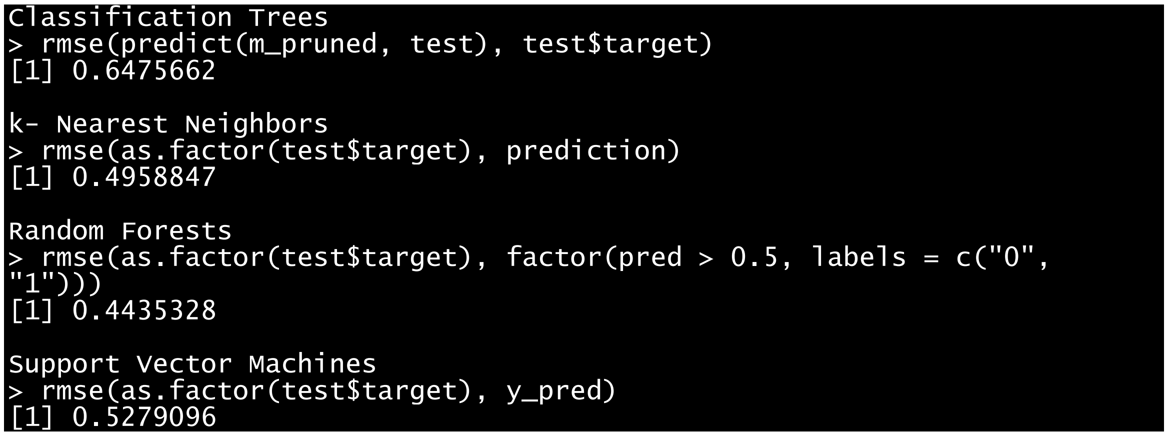


```
pred2  0  1
    0 36  6
    1  9 42
[1] 0.1612903
[1] 0.8387097
```

V. Performance Evaluation

As presented, below is the Roc curve for knn, svm and classification trees. We can clearly see that there is considerable overlapping characteristics in the curve. Since the accuracy of all the said classifiers range between 0.75 ~0.81. Random Forests turned out to be the best performing algorithm, given that its out of bag and test error where least at 3 predictors takes into consideration. We have included the confusion matrices in the respective sections of the models.





```
Classification Trees
> rmse(predict(m_pruned, test), test$target)
[1] 0.6475662

k- Nearest Neighbors
> rmse(as.factor(test$target), prediction)
[1] 0.4958847

Random Forests
> rmse(as.factor(test$target), factor(pred > 0.5, labels = c("0",
"1")))
[1] 0.4435328

Support Vector Machines
> rmse(as.factor(test$target), y_pred)
[1] 0.5279096
```

VI. Discussion and Recommendation

After performing most of the Supervised machine learning algorithms we come to know that the best models are that of random forests and neural networks. Still achieving a maximum accuracy of about 0.8387 from neural nets. This calls for methods such as bagging and boosting for random forests or using dummy variables and replacing all the present categorical variables. Even though Neural networks give better accuracy, it is to be noted that they take significantly more time to predict on new unseen datapoints.

VII. Summary

We have extensively studied the Heart disease dataset and addressed the need for the intervention of data mining techniques for the issue. As for the current study we have tackled a typical classification problem that helps in deciding whether a patient has heart disease or not. Due to contributing risk factors, it becomes difficult for such a classification if done manually; Modern data mining algorithms such as Neural Networks, Random Forests allow us to address the same. Over the course of the study we have fitted six of such algorithms to the dataset and scrutinized its outcomes by means of various metrics such as Accuracy, RMSE and ROC curves.

VIII. References

1. https://towardsdatascience.com/heart-disease-prediction-73468d630cfc
2. https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118
3. https://www.kaggle.com/ronitf/heart-disease-uci
4. https://datascienceplus.com/random-forests-in-r?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com
5. https://www.cdc.gov/heartdisease/about.htm

Appendix: R Code for use case study

~~~
```{r}
library(readr)
ht <- read_csv("C:/Users/abhil/Downloads/heart.csv")
View(ht)

colnames(ht)
```
```{r}

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

ht[,-14] <- as.data.frame(lapply(ht[,-14], normalize))

```


```{r}
knitr::kable(head(heart))

str(heart)

install.packages("DataExplorer")
library(DataExplorer)
DataExplorer::create_report(df)
```



```{r}
boxp <-
melt(ht, id.var = "target") %>%
ggplot(aes(x=variable, y=value)) +
  geom_boxplot(aes(fill=as.factor(target))) +
  xlab("") +
  theme(legend.title = element_blank())

ggplotly(boxp)

```
```{r}

# Train, Test split
spec = c(train = 0.80, test = 0.20)

g = sample(cut(
~~~

```
seq(nrow(ht)),
nrow(ht)*cumsum(c(0,spec)),
labels = names(spec)
))

res = split(ht, g)
sapply(res, nrow)/nrow(ht)

train <- res$train
#valid <- res$validate
test  <- res$test

```
```{r}
library(tidyverse)
library(magrittr)
library(reshape2)
library(ggplot2)
library(plotly)

Pearsons_Corr_test <- ht %>%
  dplyr::select(-target) %>%
  as.matrix() %>%
  na.omit() %>%
  cor()

View(Pearsons_Corr_test)
Pearsons_Corr_test <- round(Pearsons_Corr_test,2)
melted_matrix <- melt(Pearsons_Corr_test)
head(melted_matrix)


ggplot(data = melted_matrix, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
    name="Pearson\nCorrelation") +
  xlab("") +
  ylab("")

# Interactive Plot for Correlation
value <- Pearsons_Corr_test
layout <- list(title = "Features Correlation Matrix")
p <- plot_ly()
```

```
p <- add_trace(p, type = "heatmap", x = unique(melted_matrix$Var1),
y=unique(melted_matrix$Var2), z = value)
p <- layout(p, title=layout$title)
p
```
~~~
# Sample bias
```{r}

b <-
ht %>%
  ggplot(aes(factor(sex, labels = c("Female", "Male")))) +
  geom_bar(aes(fill = as.factor(sex))) +
  facet_wrap(~ target) +
  xlab("0: Absence & 1: Presence of Heart Disease") +
  ylab("Frequency")



ggplotly(b)

~~~

```{r}
reshape2::melt(ht)

h <- ggplot(reshape2::melt(ht), aes(value)) +
    geom_histogram(bins = 10) +
    facet_wrap(~variable, scales = 'free_x')
ggplotly(h)

```

# Check distribution
```{r}

library(fitdistrplus)
par(mfrow = c(2,3))
descdist(ht$age)
descdist(ht$trestbps)
descdist(ht$thalach)
descdist(ht$oldpeak)


```
```
```

~~~
```{r}

color <- c("#00AFBB", "#E7B800")
pairs(ht[,1:13], pch = 19,  cex = 0.5,
      col = color[as.factor(ht$target)],
      lower.panel=NULL)
library(GGally)
ggplotly(ggpairs(ht[,1:13], ggplot2::aes(colour = as.factor(sex))))


```
```{r}

# Missing Data
# install.packages("naniar")
library(naniar, quietly = TRUE)
vis_miss(ht)

```





# PCA
```{r}
full.pca <- prcomp(ht, center = TRUE,scale. = TRUE)
summary(full.pca)
```
```{r}
library(devtools)
install_github("vqv/ggbiplot")

```
```{r}
library(ggbiplot)
library(plotly)
ggplotly(ggbiplot(full.pca, alpha = 0))
```
```{r}
ggplotly(ggbiplot(full.pca, ellipse=TRUE,alpha = 0, groups=as.factor(ht$target),
labels.size = 5))

```
```{r}
~~~

~~~~
ggplotly(ggbiplot(full.pca, ellipse=TRUE,alpha = 0, groups=as.factor(ht$sex)))

```
```{r}
library(pca3d)
pca3d(full.pca, group=as.factor(ht$target))
#https://towardsdatascience.com/heart-disease-prediction-73468d630cfc
```
# k means
```{r}
k2 <- kmeans(ht, centers = 5, nstart = 25)
library(factoextra)
fviz_cluster(k2, data = ht)

```
```{r}
set.seed(123)
library(purrr)
# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(ht, k, nstart = 10 )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)


#plot(k.values, wss_values,type="b", pch = 19, frame = FALSE, xlab="Number of
clusters K", ylab="Total within-clusters sum of squares")

ggplotly(
ggplot() +
  geom_line(aes(x = k.values, y = wss_values)))
```

```{r}
set.seed(123)

fviz_nbclust(ht, kmeans, method = "wss")
```

# kNN
~~~~

~~~
```{r}

library(caret)
library(ggplot2)

# Fit the model on the training data

model <- train(
  as.factor(target) ~., data = train, method = "knn",
  trControl = trainControl("cv", number = 10),
  tuneLength = 20
  )

model

```

```{r}
# Plot model accuracy vs different values of k
p<- model %>%
  ggplot(aes(k, Accuracy)) +
  geom_line()

ggplotly(p)

prediction <- model %>% predict(test)
head(prediction)

```
```{r}

library(gmodels)
# Compute model accuracy rate
mean(prediction == test$target)
CrossTable(test$target, prediction, prop.chisq = FALSE)

confusionMatrix(prediction, as.factor(test$target))

```
```{r}

library(pROC)

```

```{r}
~~~

```
knn_prediction_prob <- predict(model, test, type='prob')[2]

roc_knn <- roc(as.numeric(test$target),as.numeric(as.matrix((knn_prediction_prob))))

FPR = 1 - roc_knn$specificities
TPR = roc_knn$sensitivities
ROC <- data.frame(FPR, TPR)

p <- ggplot(ROC) +
   geom_line(aes(x = FPR, y = TPR)) +
   geom_point(aes(x = FPR, y = TPR)) +
   geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1)) +
   xlab("1 - Specificity") +
   ylab("Sensitivity") +
   ggtitle("R.O.C Curve")
ggplotly(p)
```

~~~
# Classification Trees
```{r}

library(caret)
library(rpart)
set.seed(123)
# Grow a tree using all of the available predictor data
heart_model <- rpart(target~., data = train, method = "class")

# Make predictions on the test dataset
heart_test_pred <- predict(heart_model, test, type = "class")

# Examine the confusion matrix and Compute the accuracy on the test dataset
table(heart_test_pred, test$target)
confusionMatrix(heart_test_pred, as.factor(test$target))
mean(heart_test_pred == test$target)
~~~
```{r}

# pre-pruning with rpart
library(rpart)
library(rattle)
#prune_control <- rpart.control(maxdepth = 30, minsplit = 20)
m <- rpart(target ~.,
data = ht,
method = "class")
```

~~~~
plotcp(m)

# post-pruning with rpart
plotcp(m)
m_pruned <- prune(m, cp = 0.018)


fancyRpartPlot(heart_model, sub = " " , main = "Classification tree for Training data",
yesno = 0, palettes = c("YlOrRd"), branch.lwd = 2)

fancyRpartPlot(m, sub = " " , main = "Classification tree for Training data", yesno = 0,
palettes = c("YlOrRd"), branch.lwd = 2)

fancyRpartPlot(m_pruned, sub = " " , main = "Classification tree after Pruning", yesno =
0, palettes = c("YlOrRd"), branch.lwd = 2)

```
```{r}

# Make predictions on the test dataset
pruned_heart_test_pred <- predict(m_pruned, test, type = "class")

# Examine the confusion matrix and Compute the accuracy on the test dataset
table(pruned_heart_test_pred, test$target)
confusionMatrix(heart_test_pred, as.factor(test$target))
mean(pruned_heart_test_pred == test$target)


```
# SVM
```{r}

library(e1071)
library(ROCR, quietly = TRUE)
set.seed(2)

tune_ <- tune(svm, target~., data = train,
              ranges = list(gamma = 2^(-8:1), cost = 2^(0:4)),
              tunecontrol = tune.control(sampling = "fix"))
tune_

svm_model <- svm(formula = target ~ .,
             data = train,
             cost=1, gamma=0.0039,
             probability = TRUE,
             type = 'C-classification',
~~~~

~~~
           kernel = 'linear')

svm_model

```
```{r}

# Predicting the Test set results
y_pred = predict(svm_model, newdata = test)
y_pred

confusionMatrix(y_pred, as.factor(test$target))

svm_model_prob <- predict(svm_model, type="prob", newdata=test, probability =
TRUE)

# svm
svm_model_prob_rocr <- prediction(attr(svm_model_prob, "probabilities")[,1],
as.factor(test$target))
svm_perf <- performance(svm_model_prob_rocr, "tpr","fpr")
plot(svm_perf, col=4)


```


```{r}

heart_model_prob <- predict(heart_model, type = "prob", test, probability = TRUE)
heart_model_prob_rocr <- prediction(heart_model_prob[,2], as.factor(test$target))
tree_perf <- performance(heart_model_prob_rocr, "tpr","fpr")
plot(tree_perf, col=4)

```


# Random Forests
```{r}
# install.packages("randomForest")
library(randomForest)

```
```{r}

ht.rf=randomForest(target ~ . , data = train)
ht.rf

```
~~~

~~~
```{r}
oob.err=double(13)
test.err=double(13)

#mtry is no of Variables randomly chosen at each split
for(mtry in 1:13)
{
  rf=randomForest(target ~ . , data = train , mtry=mtry, ntree=400)
  oob.err[mtry] = rf$mse[400] #Error of all Trees fitted

  pred<-predict(rf,test) #Predictions on Test Set for each Tree
  test.err[mtry]= with(test, mean( (target - pred)^2)) #Mean Squared Test Error

  cat(mtry," ") #printing the output to the console

}
```

```{r}
test.err
oob.err
```



```{r}

matplot(1:mtry , cbind(oob.err,test.err), pch=19 ,
col=c("red","blue"),type="b",ylab="Mean Squared Error",xlab="Number of Predictors
Considered at each Split")
legend("topright",legend=c("Out of Bag Error","Test Error"),pch=19,
col=c("red","blue"))


confusionMatrix(factor(pred > 0.5, labels = c("0", "1")), as.factor(test$target))
factor(pred > 0.5, labels = c("0", "1"))
```

```{r}
# Predicting the Test set results
y_pred = predict(heart_model, newdata = test)
y_pred

confusionMatrix(y_pred, as.factor(test$target))

heart_model_prob <- predict(heart_model, type="prob", newdata=test, probability =
TRUE)
~~~

~~~
confusionMatrix(heart_model_prob, as.factor(test$target))
# svm
heart_model_prob_rocr <- prediction(heart_model_prob[,2], as.factor(test$target))
tree_perf <- performance(heart_model_prob_rocr, "tpr","fpr")
plot(x = ROC$FPR, y = ROC$TPR, type = "l", main = "ROC CURVES", xlab = "FPR",
ylab = "TPR")
plot(svm_perf, col = 6, add = TRUE)
plot(tree_perf, col=4, add = TRUE )
legend("bottomright", legend=c("KNN", "SVM", "CTrees"),
       col=c("black","pink", "blue"), lty = 1, cex=0.8)

```
```{r}
library(ModelMetrics)
rmse(predict(m_pruned, test), test$target)

rmse(as.factor(test$target), prediction)

rmse(as.factor(test$target), factor(pred > 0.5, labels = c("0", "1")))

rmse(as.factor(test$target), y_pred)

```
```{r Calling for Libraries}
library(tidyverse)
library(ROCR)
library(PerformanceAnalytics)
library(e1071)
library(caret)
library(gbm)
library(corrplot)
library(ggcorrplot)
library(MASS)
library(rpart)
library(caTools)
library(naivebayes)
library(class)
library(ISLR)
library(glmnet)
library(Hmisc)
library(funModeling)
library(pROC)
library(randomForest)
library(klaR)
library(scales)
~~~

```
library(cluster)
library(factoextra)
library(DataExplorer)
library(ClustOfVar)
library(GGally)


```

```{r Neural Network}
library(neuralnet)

#min-max normalization
m2$cp <- (m2$cp - min(m2$cp))/(max(m2$cp) - min(m2$cp))
m2$trestbps <- (m2$trestbps - min(m2$trestbps))/(max(m2$trestbps) -min(m2$trestbps))
m2$chol <- (m2$chol - min(m2$chol))/(max(m2$chol) -min(m2$chol))
m2$restecg <- (m2$restecg - min(m2$restecg))/(max(m2$restecg) -min(m2$restecg))
m2$thalach <- (m2$thalach - min(m2$thalach))/(max(m2$thalach) - min(m2$thalach))
m2$oldpeak <- m2$oldpeak <- (m2$oldpeak - min(m2$oldpeak))/(max(m2$oldpeak) -
min(m2$oldpeak))
m2$slope <- (m2$slope - min(m2$slope))/(max(m2$slope) - min(m2$slope))
m2$ca <- (m2$ca - min(m2$ca))/(max(m2$ca) - min(m2$ca))
m2$thal <- (m2$thal - min(m2$thal))/(max(m2$thal) - min(m2$thal))
```

```{r testing and training}
set.seed(100)
seed <- sample(2, nrow(m2), replace = TRUE, prob = c(0.7, 0.3))
training <- m2[seed==1,]
testing <- m2[seed==2,]
```

```{r}
summary(training)
plot(training)
```

```{r Applying Neural Network}
set.seed(222)
n <- neuralnet(target~.,
            data = training,
            hidden = c(2,2),
            err.fct = "ce",
            linear.output = FALSE)
plot(n)
```

~~~
```
```{r}
output <- neuralnet::compute(n, training[,-14])
head(output$net.result)
head(training[1,])
```

```{r}
output <- neuralnet::compute(n, training[,-14], rep = 1)
p1 <- output$net.result
pred1 <- ifelse(p1>0.5, 1, 0)
tab1 <- table(pred1, training$target)
tab1
Errorrate <- 1-sum(diag(tab1))/sum(tab1)
Errorrate
Accuracy <- 1- Errorrate
Accuracy
```

```{r}
output <- neuralnet::compute(n, testing[,-14], rep = 1)
p2 <- output$net.result
pred2 <- ifelse(p2>0.5, 1, 0)
tab2 <- table(pred2, testing$target)
tab2
1-sum(diag(tab2))/sum(tab2)
Errorrate2 <- 1-sum(diag(tab2))/sum(tab2)
Accuracy2 <- 1- Errorrate2
Accuracy2
```

#Logistic
```{r}
library(ggplot2)
library(cowplot)
```

```{r}
head(data)
str(data)
```

```{r}
data <- read.csv("heart(AutoRecovered).csv")
data[data$sex == 0,]$sex <- "F"
data[data$sex == 1,]$sex <- "M"
~~~

~~~
data$sex <- as.factor(data$sex)

data$cp <- as.factor(data$cp)
data$fbs <- as.factor(data$fbs)
data$restecg <- as.factor(data$restecg)
data$exang <- as.factor(data$exang)
data$slope <- as.factor(data$slope)
data$ca <- as.factor(data$ca)
data$thal <- as.factor(data$thal)
data$target <- ifelse(test=data$target == 0, yes="Healthy", no="Unhealthy")
data$target <- as.factor(data$target)
```

```{r}
str(data)
```

```{r}
xtabs(~ target + sex, data=data)
xtabs(~ target + cp, data=data)
xtabs(~ target + fbs, data=data)
xtabs(~ target + restecg, data=data)
xtabs(~ target + exang, data=data)
xtabs(~ target + slope, data=data)
xtabs(~ target + ca, data=data)
xtabs(~ target + thal, data=data)

```



```{r}

logistic <- glm(target ~., data=data, family="binomial")
summary(logistic)
```



```{r}
null <- logistic$null.deviance/-2
proposed <- logistic$deviance/-2
```

```{r}
(null - proposed) /null
```
~~~

~~~
```{r}
1 - pchisq(2*(proposed - null), df=(length(logistic$coefficients)-1))
```

```{r}
predicted.data <- data.frame(
  probability.of.target=logistic$fitted.values,
  target=data$target)
```

```{r}
predicted.data <- predicted.data[
  order(predicted.data$probability.of.target, decreasing=FALSE),]
predicted.data$rank <- 1:nrow(predicted.data)
```

```{r}
ggplot(data=predicted.data, aes(x=rank, y=probability.of.target)) +
  geom_point(aes(color=target), alpha=1, shape=4, stroke=2) +
  xlab("Index") +
  ylab("Predicted probability of getting heart disease")

ggsave("heart_disease_probabilities.pdf")
```
~~~