Northeastern University
College of Engineering

IE7275: Data Mining in Engineering

Prediction of Heart Disease Factors with Machine Learning

Submitted By:

Group: 01

Mayank Mehta: 001401159

Aashka Shah: 001081495

ACKNOWLEDGEMENT

We as a team would like to firstly thank Prof. Xuemin Jin, for assigning us such a challenging and holistic project. It is through this project we were able to apply our theoretical concepts and knowledge learned in the course and see its wide scale industrial application in problem solving. Through this medium we would also like to acknowledge the fact that this complied project report and the work we are submitting is our original work. We hold the ideals of our school Northeastern University in high regards and we abide by the code of honor.

TABLE OF CONTENTS

Prediction of Heart Disease Factors with Machine Learning

Background Information:

In today's world the rate of heart disease is increasing tremendously because of our eating and drinking habits. On an average there are 12 million deaths per annum due to heart disease, reported by World health organization. In order to decrease this rate, we need to take early precautions by changing our way of living.

Problem Statement:

The factors such as BP, diabetics, cholesterol, current smoker, etc are the causes for heart disease. The objective of this project is to analyze the data and build a model to predict whether a person can get heart disease in the coming 10 years or not. If we have the efficient model than a person can take precautions to avoid the heart disease at an early stage.

Data source:

https://www.kaggle.com/dileep070/heart-disease-prediction-using-logistic-regression#framingham.csv

Dataset Description:

There is a total of 4238 observations and 16 attribute variables and 1 is a response variable with binary class.

Attribute 1	Male
Attribute 2	Age
Attribute 3	Education
Attribute 4	Current smoker
Attribute 5	Cig per day
Attribute 6	BP medicines
Attribute 7	Prevalent stroke
Attribute 8	Prevalent hypertension
Attribute 9	Diabetes
Attribute 10	Total Cholesterol
Attribute 11	Systolic blood pressure
Attribute 12	Diastolic blood pressure
Attribute 13	BMI
Attribute 14	Heart rate
Attribute 15	Glucose
Attribute 16	Coronary heart disease

##Import dataset -

```
system("java -version")

library("readxl")
heart_data <- read_excel("C:/Users/mayan/OneDrive/Documents/heart_data.xlsx")
#View(heart_data)
```

- **Data Exploration:**

Checking the Dimensions:

```
dim(heart_data)

## [1] 4238   16
```

Data: 16 columns(Variables) and 4238 observations

```
str(heart_data)

## Classes 'tbl_df', 'tbl' and 'data.frame':    4238 obs. of  16 variables:
##  $ male           : num  1 0 1 0 0 0 0 0 1 1 ...
##  $ age            : num  39 46 48 61 46 43 63 45 52 43 ...
##  $ education      : num  4 2 1 3 3 2 1 2 1 1 ...
##  $ currentSmoker  : num  0 0 1 1 1 0 0 1 0 1 ...
##  $ cigsPerDay     : num  0 0 20 30 23 0 0 20 0 30 ...
##  $ BPMeds         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ prevalentStroke: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ prevalentHyp   : num  0 0 0 1 0 1 0 0 1 1 ...
##  $ diabetes       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ totChol        : num  195 250 245 225 285 228 205 313 260 225 ...
##  $ sysBP          : num  106 121 128 150 130 ...
##  $ diaBP          : num  70 81 80 95 84 110 71 71 89 107 ...
##  $ BMI            : num  27 28.7 25.3 28.6 23.1 ...
##  $ heartRate      : num  80 95 75 65 85 77 60 79 76 93 ...
##  $ glucose        : num  77 76 70 103 85 99 85 78 79 88 ...
##  $ TenYearCHD     : num  0 0 0 1 0 0 1 0 0 0 ...

heart_data$TenYearCHD <- factor(heart_data$TenYearCHD)
summary(heart_data)

##       male             age          education     currentSmoker
##  Min.   :0.0000   Min.   :32.00   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:42.00   1st Qu.:1.000   1st Qu.:0.0000
##  Median :0.0000   Median :49.00   Median :2.000   Median :0.0000
##  Mean   :0.4292   Mean   :49.58   Mean   :1.979   Mean   :0.4941
##  3rd Qu.:1.0000   3rd Qu.:56.00   3rd Qu.:3.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :70.00   Max.   :4.000   Max.   :1.0000
```

```
##    cigsPerDay         BPMeds         prevalentStroke     prevalentHyp
##  Min.   : 0.000   Min.   :0.00000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median : 0.000   Median :0.00000   Median :0.000000   Median :0.0000
##  Mean   : 9.003   Mean   :0.03587   Mean   :0.005899   Mean   :0.3105
##  3rd Qu.:20.000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:1.0000
##  Max.   :70.000   Max.   :1.00000   Max.   :1.000000   Max.   :1.0000
##     diabetes          totChol          sysBP             diaBP
##  Min.   :0.00000   Min.   :107.0   Min.   : 83.5   Min.   : 48.00
##  1st Qu.:0.00000   1st Qu.:206.0   1st Qu.:117.0   1st Qu.: 75.00
##  Median :0.00000   Median :234.0   Median :128.0   Median : 82.00
##  Mean   :0.02572   Mean   :236.7   Mean   :132.4   Mean   : 82.89
##  3rd Qu.:0.00000   3rd Qu.:262.0   3rd Qu.:144.0   3rd Qu.: 89.88
##  Max.   :1.00000   Max.   :696.0   Max.   :295.0   Max.   :142.50
##       BMI          heartRate         glucose        TenYearCHD
##  Min.   :15.54   Min.   : 44.00   Min.   : 40.00   0:3594
##  1st Qu.:23.08   1st Qu.: 68.00   1st Qu.: 72.00   1: 644
##  Median :25.41   Median : 75.00   Median : 80.00
##  Mean   :25.80   Mean   : 75.88   Mean   : 82.69
##  3rd Qu.:28.04   3rd Qu.: 83.00   3rd Qu.: 89.90
##  Max.   :56.80   Max.   :143.00   Max.   :394.00
```

```r
levels(heart_data$TenYearCHD)
```

```
## [1] "0" "1"
```

```r
sum(is.na(heart_data))
```

```
## [1] 0
```

Cheking for Missing values

```r
heart_data <- na.omit(heart_data)
sum(is.na(heart_data))
```

```
## [1] 0
```

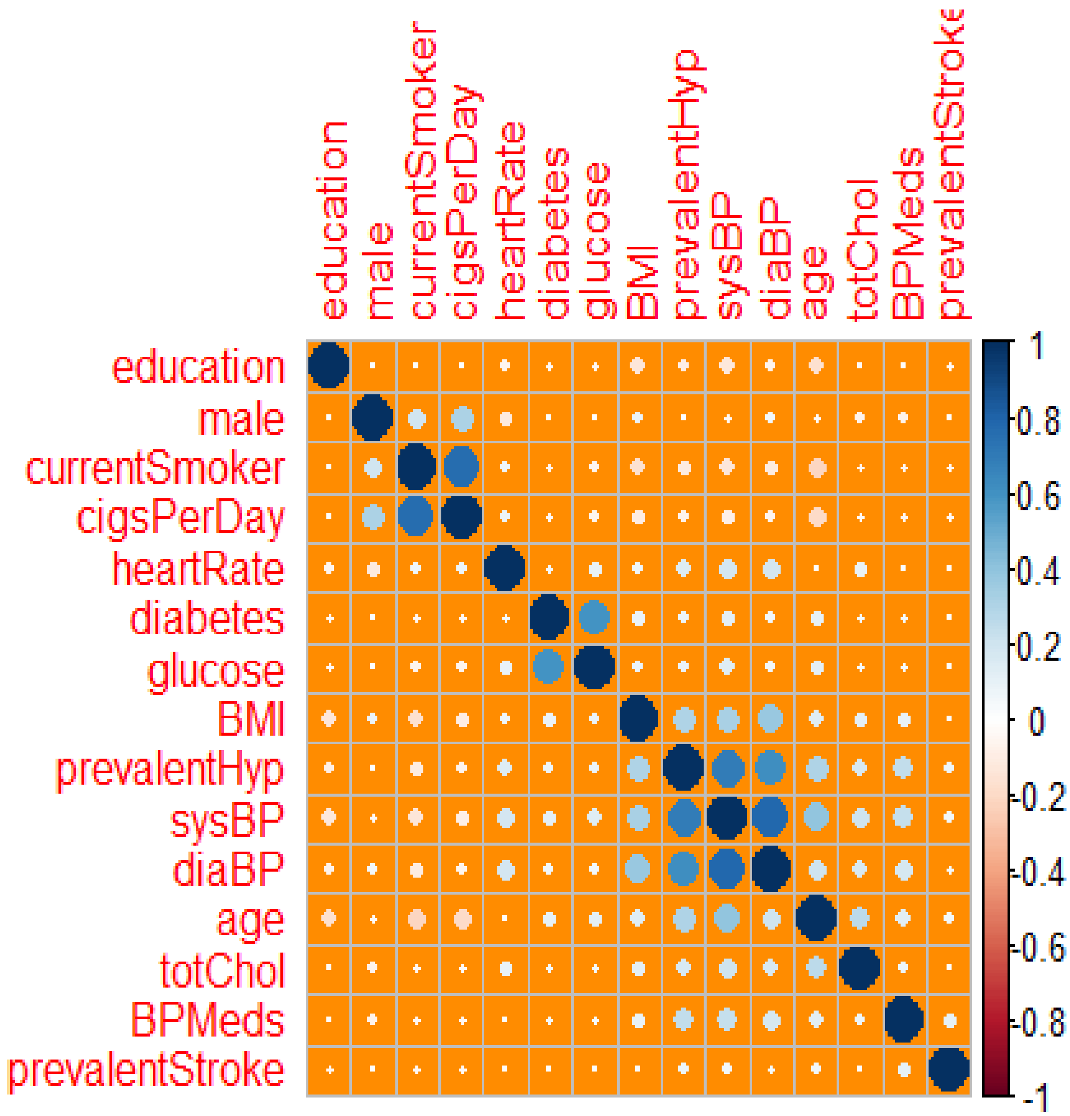
```r
normalize <- function(x) { (x - min(x)) / (max(x) - min(x))}
heart_data$totChol <- normalize(heart_data$totChol)
heart_data$sysBP <- normalize(heart_data$sysBP)
heart_data$diaBP <- normalize(heart_data$diaBP)
heart_data$BMI <- normalize(heart_data$BMI)
heart_data$heartRate <- normalize(heart_data$heartRate)
heart_data$glucose <- normalize(heart_data$glucose)
```

➢ Data Visualization:

```
#install.packages("ggplot2")
#install.packages("GGally")
#install.packages("corrplot")

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3

library(GGally)

library(tidyverse)

library(dplyr)
correlation <- cor(heart_data [,-16], method = "pearson" , use = "complete.ob
s")
library(corrplot)

round(correlation,2)

##                  male   age education currentSmoker cigsPerDay BPMeds
## male             1.00 -0.03      0.02          0.20       0.32  -0.06
## age             -0.03  1.00     -0.16         -0.21      -0.19   0.13
## education        0.02 -0.16      1.00          0.02       0.01  -0.01
## currentSmoker    0.20 -0.21      0.02          1.00       0.77  -0.05
## cigsPerDay       0.32 -0.19      0.01          0.77       1.00  -0.04
## BPMeds          -0.06  0.13     -0.01         -0.05      -0.04   1.00
## prevalentStroke  0.00  0.06     -0.04         -0.03      -0.03   0.10
## prevalentHyp     0.01  0.31     -0.08         -0.10      -0.07   0.24
## diabetes         0.02  0.10     -0.04         -0.04      -0.04   0.05
## totChol         -0.07  0.26     -0.02         -0.05      -0.03   0.08
## sysBP           -0.04  0.39     -0.13         -0.13      -0.09   0.24
## diaBP            0.06  0.21     -0.06         -0.11      -0.06   0.17
## BMI              0.08  0.14     -0.14         -0.17      -0.09   0.09
## heartRate       -0.12 -0.01     -0.05          0.06       0.07   0.02
## glucose          0.00  0.11     -0.03         -0.05      -0.06   0.04
##                 prevalentStroke prevalentHyp diabetes totChol sysBP diaBP
BMI
## male                       0.00         0.01     0.02   -0.07 -0.04  0.06
0.08
## age                        0.06         0.31     0.10    0.26  0.39  0.21
0.14
## education                 -0.04        -0.08    -0.04   -0.02 -0.13 -0.06
-0.14
## currentSmoker             -0.03        -0.10    -0.04   -0.05 -0.13 -0.11
-0.17
## cigsPerDay                -0.03        -0.07    -0.04   -0.03 -0.09 -0.06
-0.09
## BPMeds                     0.10         0.24     0.05    0.08  0.24  0.17
0.09
```

```
## prevalentStroke            1.00          0.07      0.01     0.00  0.06  0.05
0.02
## prevalentHyp               0.07          1.00      0.08     0.16  0.70  0.62
0.30
## diabetes                   0.01          0.08      1.00     0.04  0.11  0.05
0.09
## totChol                    0.00          0.16      0.04     1.00  0.21  0.16
0.11
## sysBP                      0.06          0.70      0.11     0.21  1.00  0.78
0.33
## diaBP                      0.05          0.62      0.05     0.16  0.78  1.00
0.38
## BMI                        0.02          0.30      0.09     0.11  0.33  0.38
1.00
## heartRate                 -0.02          0.15      0.05     0.09  0.18  0.18
0.07
## glucose                    0.02          0.08      0.60     0.04  0.13  0.06
0.08
##                   heartRate glucose
## male                  -0.12    0.00
## age                   -0.01    0.11
## education             -0.05   -0.03
## currentSmoker          0.06   -0.05
## cigsPerDay             0.07   -0.06
## BPMeds                 0.02    0.04
## prevalentStroke       -0.02    0.02
## prevalentHyp           0.15    0.08
## diabetes               0.05    0.60
## totChol                0.09    0.04
## sysBP                  0.18    0.13
## diaBP                  0.18    0.06
## BMI                    0.07    0.08
## heartRate              1.00    0.09
## glucose                0.09    1.00

whiteblack <- c("white", "black")
corrplot(correlation, order = "hclust",  bg = "darkorange")
```

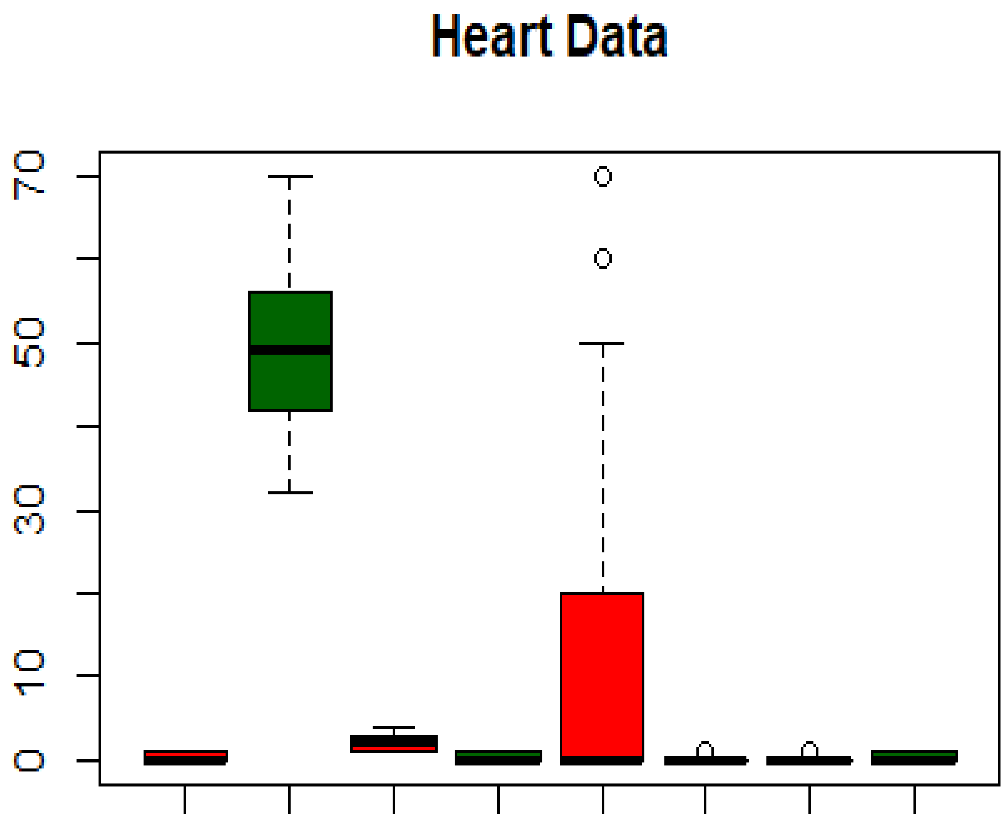


Normalized the data set.

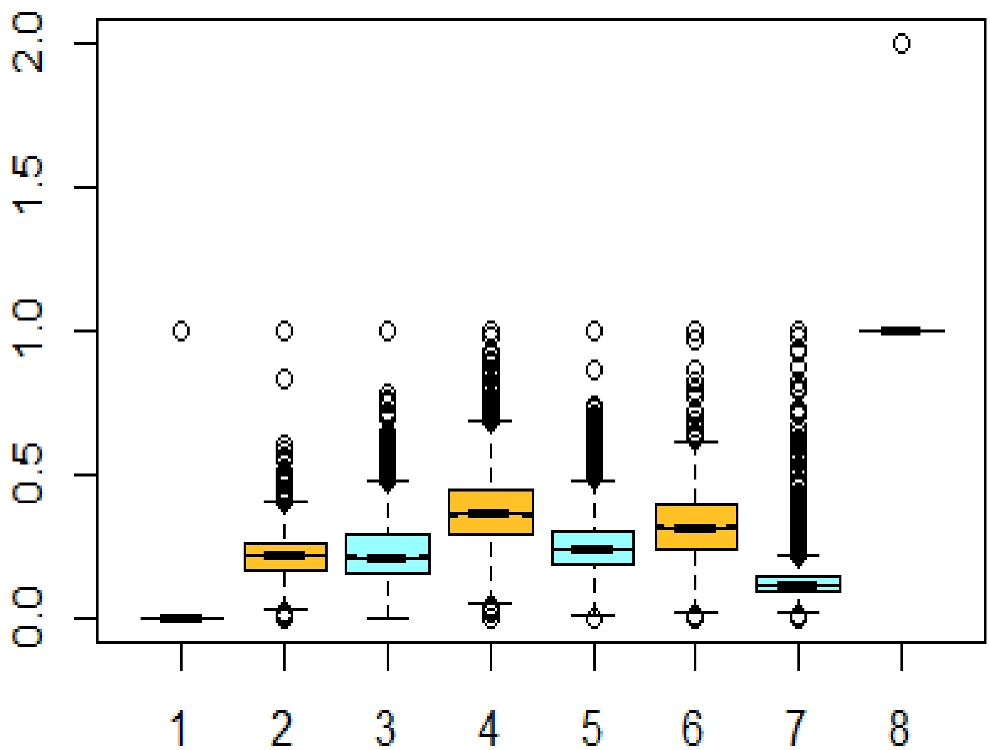
By looking at the below correlation plot, there's moderately high correlation between pelvic incidence and sacral slope.

#Here is presentation of box plot-graphs to check the outliers corresponding each predictor variable.

```
boxplot(heart_data$male, heart_data$age, heart_data$education,heart_data$curr
entSmoker, heart_data$cigsPerDay, heart_data$BPMeds, heart_data$prevalentStro
ke, heart_data$prevalentHyp, notch=FALSE,col=(c("red","darkgreen")),main="Hea
rt Data")$out
```



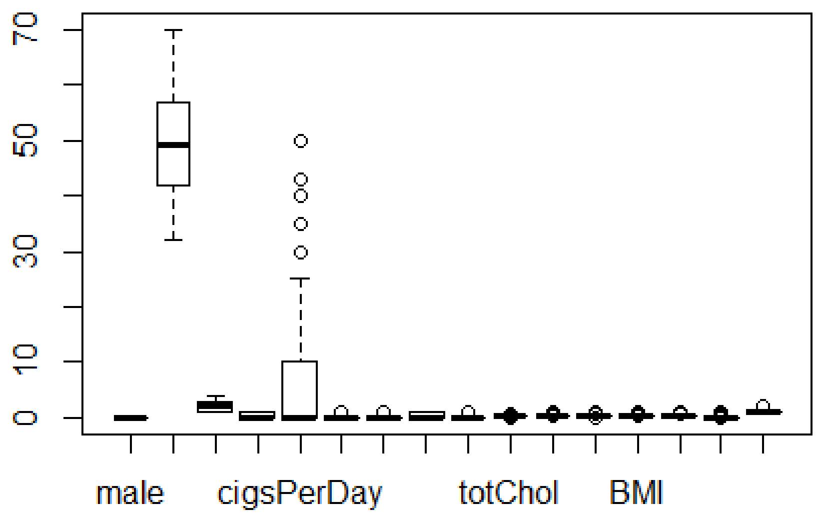

```
boxplot(heart_data$diabetes, heart_data$totChol, heart_data$sysBP, heart_data
$diaBP, heart_data$BMI, heart_data$heartRate, heart_data$glucose, heart_data$
TenYearCHD, notch=TRUE,col=(c("darkslategray1","goldenrod1")))$out
```




```r
# Load the dataset
data("heart_data")

# Create a boxplot of the dataset, outliers are shown as two distinct points
boxplot(heart_data)$out
```

```r
#detecting outliers
boxplot(heart_data$male, heart_data$age, heart_data$education,heart_data$currentSmoker, heart_data$cigsPerDay, heart_data$BPMeds, heart_data$prevalentStroke, heart_data$prevalentHyp, plot=FALSE)$out
```

```
#Defining outliers as a vector
outliers <- boxplot(heart_data$male, heart_data$age, heart_data$education,hea
rt_data$currentSmoker, heart_data$cigsPerDay, heart_data$BPMeds, heart_data$p
revalentStroke, heart_data$prevalentHyp, plot=FALSE)$out

# Removing Outliers
heart_data <- heart_data[-which(c(heart_data$male, heart_data$age, heart_data
$education,heart_data$currentSmoker, heart_data$cigsPerDay, heart_data$BPMeds
, heart_data$prevalentStroke, heart_data$prevalentHyp) %in% outliers),]

boxplot(heart_data)
```



Performing Data Mining Techniques

Preparing Data Set into Train Data (80%) and Test Data (20%)

```
set.seed(123)
library(caTools)
sample <- sample.split(heart_data, SplitRatio = 0.80)
train <- subset(heart_data, sample == TRUE)

test1 <- subset(heart_data, sample == FALSE)

test <- test1[,-16]
train_knn <- train[,-16]
```

```r
actual.results <- as.vector(test1$TenYearCHD)
train.result <- as.vector(train$TenYearCHD)
```

(1) K-nearest neighbors algorithms:

```r
library(class)
library(gmodels)

#For k=1
knn_pred.result1 <- knn(train_knn, test, train.result, k=1 )
table(knn_pred.result1, actual.results)

##                   actual.results
## knn_pred.result1   0   1
##                  0 474  59
##                  1  54  17

misClassificError <- mean(knn_pred.result1 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.812913907284768"

#For k=2
knn_pred.result2 <- knn(train_knn, test, train.result, k=2 )
table(knn_pred.result2, actual.results)

##                   actual.results
## knn_pred.result2   0   1
##                  0 476  59
##                  1  52  17

misClassificError <- mean(knn_pred.result2 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.816225165562914"

#For k=3
knn_pred.result3 <- knn(train_knn, test, train.result, k=3 )
table(knn_pred.result3, actual.results)

##                   actual.results
## knn_pred.result3   0   1
##                  0 513  70
##                  1  15   6

misClassificError <- mean(knn_pred.result3 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.859271523178808"

#For k=4
knn_pred.result4 <- knn(train_knn, test, train.result, k=4 )
table(knn_pred.result4, actual.results)
```

```
##                  actual.results
## knn_pred.result4   0   1
##                 0 508  72
##                 1  20   4

misClassificError <- mean(knn_pred.result4 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.847682119205298"

#For k=5
knn_pred.result5 <- knn(train_knn, test, train.result, k=5 )
table(knn_pred.result5, actual.results)

##                  actual.results
## knn_pred.result5   0   1
##                 0 517  71
##                 1  11   5

misClassificError <- mean(knn_pred.result5 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.864238410596026"

#For k=6
knn_pred.result6 <- knn(train_knn, test, train.result, k=6 )
table(knn_pred.result6, actual.results)

##                  actual.results
## knn_pred.result6   0   1
##                 0 515  72
##                 1  13   4

misClassificError <- mean(knn_pred.result6 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.859271523178808"

#For k=7
knn_pred.result7 <- knn(train_knn, test, train.result, k=7 )
table(knn_pred.result7, actual.results)

##                  actual.results
## knn_pred.result7   0   1
##                 0 518  72
##                 1  10   4

misClassificError <- mean(knn_pred.result7 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.864238410596026"
```

```r
#For k=8
knn_pred.result8 <- knn(train_knn, test, train.result, k=8 )
table(knn_pred.result8, actual.results)

##                  actual.results
## knn_pred.result8   0   1
##                 0 518  71
##                 1  10   5

misClassificError <- mean(knn_pred.result8 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.865894039735099"

#For k=9
knn_pred.result9 <- knn(train_knn, test, train.result, k=9 )
table(knn_pred.result9, actual.results)

##                  actual.results
## knn_pred.result9   0   1
##                 0 521  74
##                 1   7   2

misClassificError <- mean(knn_pred.result9 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.865894039735099"

#For k=10
knn_pred.result10 <- knn(train_knn, test, train.result, k=10 )
table(knn_pred.result10, actual.results)

##                   actual.results
## knn_pred.result10   0   1
##                  0 519  74
##                  1   9   2

misClassificError <- mean(knn_pred.result10 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.862582781456954"

#For k=10, we are getting the best accuracy which is 84.23%. Hence, we can se
lect k=10 as the best value of k.
```

For K = 6 we have got best results 86.58% accuracy, according to model, we can select K = 6 as best K value.

(2) Classification Tree Regression:

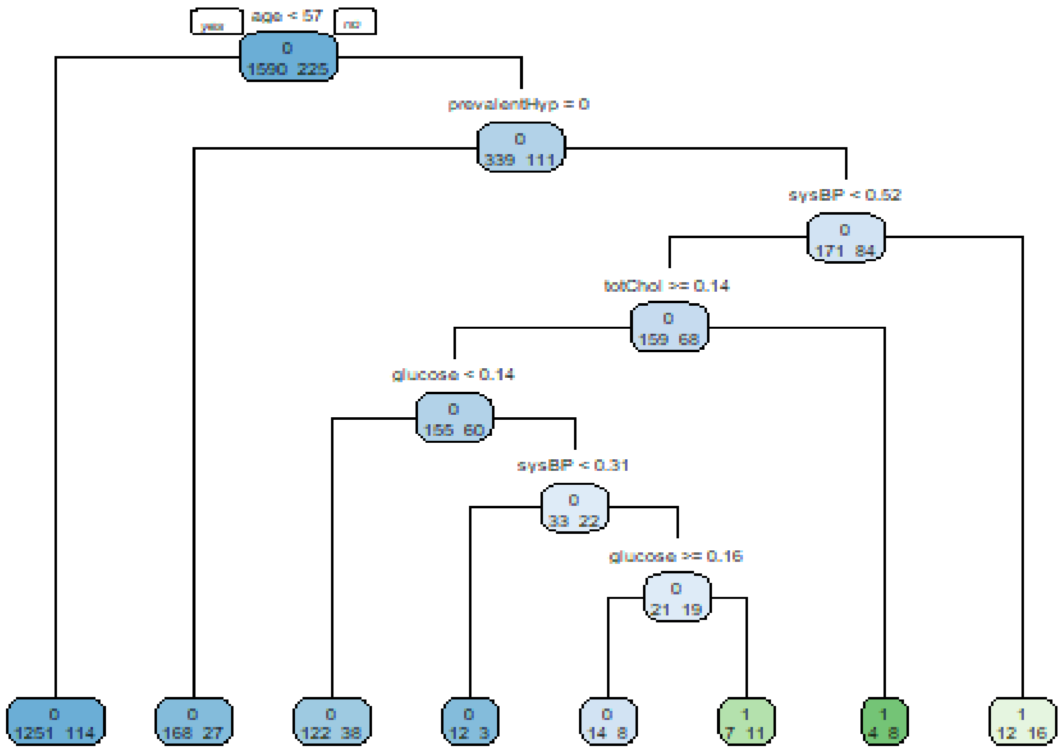
```
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.6.3

CT_model1 <- rpart(TenYearCHD ~ . , data = train, method = 'class', control =
                    rpart.control(minsplit = 30, cp=0.0055))
printcp(CT_model1)

##
## Classification tree:
## rpart(formula = TenYearCHD ~ ., data = train, method = "class",
##     control = rpart.control(minsplit = 30, cp = 0.0055))
##
## Variables actually used in tree construction:
## [1] age          glucose      prevalentHyp sysBP        totChol
##
## Root node error: 225/1815 = 0.12397
##
## n= 1815
##
##          CP nsplit rel error xerror     xstd
## 1 0.0088889      0   1.00000 1.0000 0.062398
## 2 0.0059259      4   0.96444 1.1111 0.065254
## 3 0.0055000      7   0.94667 1.1022 0.065034

rpart.plot(CT_model1, type = 1, extra = 1, split.font = 1, varlen = -20)
```




```
summary(CT_model1)

## Call:
## rpart(formula = TenYearCHD ~ ., data = train, method = "class",
##     control = rpart.control(minsplit = 30, cp = 0.0055))
##   n= 1815
##
##            CP nsplit rel error   xerror       xstd
## 1 0.008888889      0 1.0000000 1.000000 0.06239776
## 2 0.005925926      4 0.9644444 1.111111 0.06525388
## 3 0.005500000      7 0.9466667 1.102222 0.06503386
##
## Variable importance
##          age        sysBP prevalentHyp        diaBP      glucose      totC
hol
##           33           23           15            9            7
7
##          BMI    heartRate     diabetes
##            3            1            1
##
## Node number 1: 1815 observations,    complexity param=0.008888889
##   predicted class=0  expected loss=0.1239669  P(node) =1
##     class counts:  1590   225
##    probabilities: 0.876 0.124
```

```
##   left son=2 (1365 obs) right son=3 (450 obs)
##   Primary splits:
##       age          < 56.5      to the left,  improve=18.016630, (0 missing
)
##       prevalentHyp < 0.5       to the left,  improve=17.489410, (0 missing
)
##       sysBP        < 0.4314421 to the left,  improve=17.268750, (0 missing
)
##       glucose      < 0.2189266 to the left,  improve= 9.855757, (0 missing
)
##       diaBP        < 0.4312169 to the left,  improve= 7.791471, (0 missing
)
##   Surrogate splits:
##       sysBP    < 0.3877069 to the left,  agree=0.774, adj=0.089, (0 split)
##       glucose  < 0.2584746 to the left,  agree=0.755, adj=0.013, (0 split)
##       diabetes < 0.5       to the left,  agree=0.754, adj=0.009, (0 split)
##       BPMeds   < 0.5       to the left,  agree=0.753, adj=0.004, (0 split)
##       totChol  < 0.4151104 to the left,  agree=0.753, adj=0.004, (0 split)
##
## Node number 2: 1365 observations
##   predicted class=0  expected loss=0.08351648  P(node) =0.7520661
##     class counts:  1251   114
##    probabilities: 0.916 0.084
##
## Node number 3: 450 observations,    complexity param=0.008888889
##   predicted class=0  expected loss=0.2466667  P(node) =0.2479339
##     class counts:   339   111
##    probabilities: 0.753 0.247
##   left son=6 (195 obs) right son=7 (255 obs)
##   Primary splits:
##       prevalentHyp < 0.5       to the left,  improve=8.058100, (0 missing)
##       sysBP        < 0.4314421 to the left,  improve=7.034135, (0 missing)
##       age          < 64.5      to the left,  improve=4.412990, (0 missing)
##       totChol      < 0.1400679 to the right, improve=4.380554, (0 missing)
##       heartRate    < 0.4292929 to the left,  improve=2.902740, (0 missing)
##   Surrogate splits:
##       sysBP     < 0.2777778 to the left,  agree=0.880, adj=0.723, (0 split
)
##       diaBP     < 0.3783069 to the left,  agree=0.789, adj=0.513, (0 split
)
##       BMI       < 0.1999515 to the left,  agree=0.633, adj=0.154, (0 split
)
##       heartRate < 0.2777778 to the left,  agree=0.607, adj=0.092, (0 split
)
##       age       < 57.5      to the left,  agree=0.580, adj=0.031, (0 split
)
##
## Node number 6: 195 observations
##   predicted class=0  expected loss=0.1384615  P(node) =0.107438
##     class counts:   168    27
```

```
##    probabilities: 0.862 0.138
##
## Node number 7: 255 observations,    complexity param=0.008888889
##   predicted class=0  expected loss=0.3294118  P(node) =0.1404959
##     class counts:   171    84
##    probabilities: 0.671 0.329
##   left son=14 (227 obs) right son=15 (28 obs)
##   Primary splits:
##       sysBP   < 0.5153664 to the left,  improve=3.684626, (0 missing)
##       totChol < 0.1341256 to the right, improve=3.639449, (0 missing)
##       glucose < 0.2189266 to the left,  improve=2.983091, (0 missing)
##       age     < 64.5      to the left,  improve=1.884314, (0 missing)
##       diaBP   < 0.7513228 to the left,  improve=1.524130, (0 missing)
##   Surrogate splits:
##       diaBP   < 0.7275132 to the left,  agree=0.910, adj=0.179, (0 split)
##       BMI     < 0.6805623 to the left,  agree=0.898, adj=0.071, (0 split)
##       glucose < 0.7669492 to the left,  agree=0.898, adj=0.071, (0 split)
##
## Node number 14: 227 observations,    complexity param=0.008888889
##   predicted class=0  expected loss=0.2995595  P(node) =0.1250689
##     class counts:   159    68
##    probabilities: 0.700 0.300
##   left son=28 (215 obs) right son=29 (12 obs)
##   Primary splits:
##       totChol   < 0.1400679 to the right, improve=3.414951, (0 missing)
##       heartRate < 0.2878788 to the right, improve=2.069950, (0 missing)
##       glucose   < 0.1341808 to the left,  improve=1.607551, (0 missing)
##       diaBP     < 0.3544974 to the right, improve=1.470640, (0 missing)
##       BMI       < 0.5510179 to the left,  improve=1.397959, (0 missing)
##
## Node number 15: 28 observations
##   predicted class=1  expected loss=0.4285714  P(node) =0.015427
##     class counts:    12    16
##    probabilities: 0.429 0.571
##
## Node number 28: 215 observations,    complexity param=0.005925926
##   predicted class=0  expected loss=0.2790698  P(node) =0.1184573
##     class counts:   155    60
##    probabilities: 0.721 0.279
##   left son=56 (160 obs) right son=57 (55 obs)
##   Primary splits:
##       glucose   < 0.1411017 to the left,  improve=2.161628, (0 missing)
##       diaBP     < 0.3544974 to the right, improve=1.919488, (0 missing)
##       BMI       < 0.5510179 to the left,  improve=1.645318, (0 missing)
##       sysBP     < 0.4314421 to the left,  improve=1.453488, (0 missing)
##       heartRate < 0.4393939 to the left,  improve=1.298862, (0 missing)
##   Surrogate splits:
##       diabetes  < 0.5       to the left,  agree=0.772, adj=0.109, (0 split
)
##       totChol   < 0.1544992 to the right, agree=0.749, adj=0.018, (0 split
```

```
)
##       heartRate < 0.5909091 to the left,  agree=0.749, adj=0.018, (0 split
)
##
## Node number 29: 12 observations
##   predicted class=1  expected loss=0.3333333  P(node) =0.00661157
##     class counts:     4     8
##    probabilities: 0.333 0.667
##
## Node number 56: 160 observations
##   predicted class=0  expected loss=0.2375  P(node) =0.08815427
##     class counts:   122    38
##    probabilities: 0.762 0.238
##
## Node number 57: 55 observations,    complexity param=0.005925926
##   predicted class=0  expected loss=0.4  P(node) =0.03030303
##     class counts:    33    22
##    probabilities: 0.600 0.400
##   left son=114 (15 obs) right son=115 (40 obs)
##   Primary splits:
##       sysBP     < 0.3096927 to the left,  improve=1.650000, (0 missing)
##       diaBP     < 0.4814815 to the left,  improve=1.650000, (0 missing)
##       totChol   < 0.2113752 to the left,  improve=1.295470, (0 missing)
##       heartRate < 0.2979798 to the right, improve=1.294895, (0 missing)
##       glucose   < 0.1567797 to the right, improve=1.171739, (0 missing)
##   Surrogate splits:
##       diaBP    < 0.2698413 to the left,  agree=0.764, adj=0.133, (0 split)
##       diabetes < 0.5       to the right, agree=0.745, adj=0.067, (0 split)
##       glucose  < 0.2782486 to the right, agree=0.745, adj=0.067, (0 split)
##
## Node number 114: 15 observations
##   predicted class=0  expected loss=0.2  P(node) =0.008264463
##     class counts:    12     3
##    probabilities: 0.800 0.200
##
## Node number 115: 40 observations,    complexity param=0.005925926
##   predicted class=0  expected loss=0.475  P(node) =0.02203857
##     class counts:    21    19
##    probabilities: 0.525 0.475
##   left son=230 (22 obs) right son=231 (18 obs)
##   Primary splits:
##       glucose   < 0.1567797 to the right, improve=1.2126260, (0 missing)
##       age       < 59.5      to the right, improve=0.7901254, (0 missing)
##       sysBP     < 0.356974  to the right, improve=0.7820802, (0 missing)
##       diaBP     < 0.4920635 to the left,  improve=0.7591168, (0 missing)
##       heartRate < 0.4747475 to the right, improve=0.6880952, (0 missing)
##   Surrogate splits:
##       diaBP   < 0.3862434 to the right, agree=0.625, adj=0.167, (0 split)
##       age     < 62.5      to the left,  agree=0.600, adj=0.111, (0 split)
##       totChol < 0.2741935 to the left,  agree=0.600, adj=0.111, (0 split)
```

```
##       sysBP   < 0.3640662 to the right, agree=0.600, adj=0.111, (0 split)
##       BMI     < 0.1649297 to the right, agree=0.600, adj=0.111, (0 split)
##
## Node number 230: 22 observations
##   predicted class=0  expected loss=0.3636364  P(node) =0.01212121
##     class counts:    14     8
##    probabilities: 0.636 0.364
##
## Node number 231: 18 observations
##   predicted class=1  expected loss=0.3888889  P(node) =0.009917355
##     class counts:     7    11
##    probabilities: 0.389 0.611
```

```
CT_pred.result1 <-predict(CT_model1, test, type = 'class')

table(CT_pred.result1, actual.results)
```

```
##                actual.results
## CT_pred.result1   0   1
##               0 516  70
##               1  12   6
```

```
misClassificError <- mean(CT_pred.result1 != actual.results)
print(paste('Accuracy', 1-misClassificError))
```

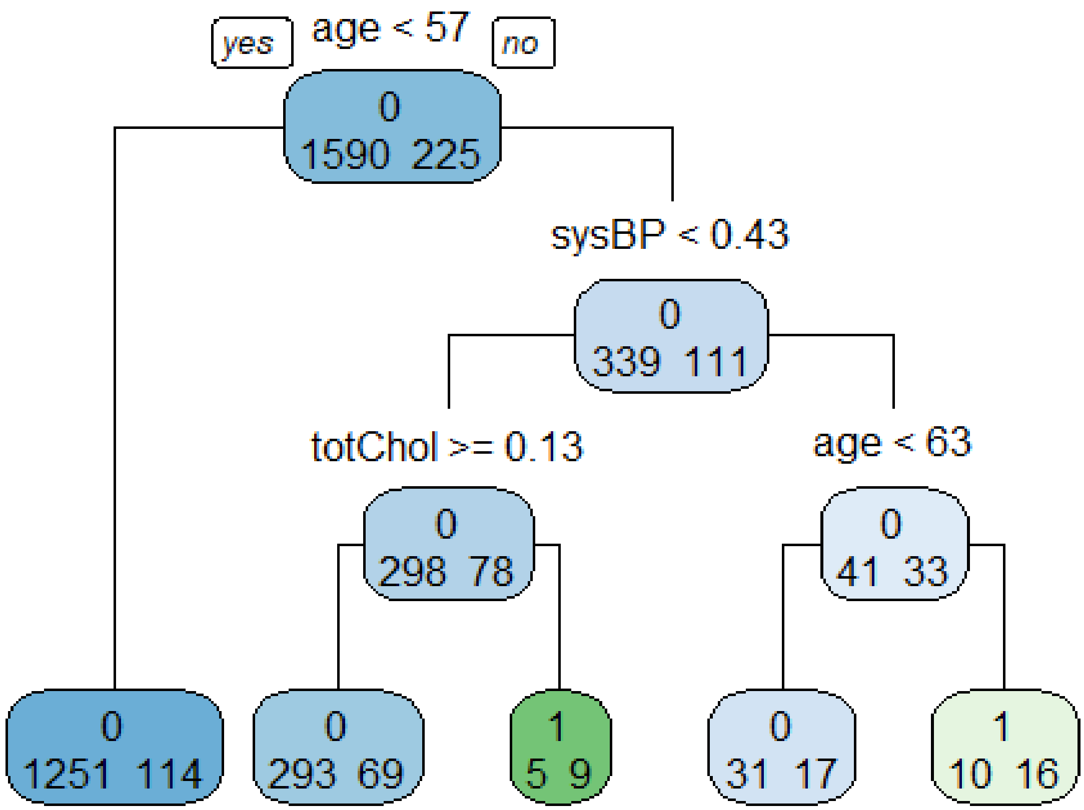
```
## [1] "Accuracy 0.864238410596026"
```

For Model 1, we have got 86.92% of the accuracy Considering predictors with importance more 7.

```
CT_model2 <- rpart(TenYearCHD ~ age + sysBP + glucose + totChol, data = train
, method = 'class', control=rpart.control(minsplit=30, cp=0.0055))
print(CT_model2)
```

```
## n= 1815
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1815 225 0 (0.87603306 0.12396694)
##    2) age< 56.5 1365 114 0 (0.91648352 0.08351648) *
##    3) age>=56.5 450 111 0 (0.75333333 0.24666667)
##      6) sysBP< 0.4314421 376  78 0 (0.79255319 0.20744681)
##       12) totChol>=0.1298812 362  69 0 (0.80939227 0.19060773) *
##       13) totChol< 0.1298812 14   5 1 (0.35714286 0.64285714) *
##      7) sysBP>=0.4314421 74  33 0 (0.55405405 0.44594595)
##       14) age< 62.5 48  17 0 (0.64583333 0.35416667) *
##       15) age>=62.5 26  10 1 (0.38461538 0.61538462) *
```

```
rpart.plot(CT_model2, type = 1, extra = 1, split.font = 1, varlen = -20)
```




```
summary(CT_model2)

## Call:
## rpart(formula = TenYearCHD ~ age + sysBP + glucose + totChol,
##     data = train, method = "class", control = rpart.control(minsplit = 30,
##         cp = 0.0055))
##   n= 1815
##
##           CP nsplit rel error   xerror       xstd
## 1 0.01111111      0 1.0000000 1.000000 0.06239776
## 2 0.00550000      4 0.9555556 1.066667 0.06413943
##
## Variable importance
##     age   sysBP totChol glucose
##      57      24      17       2
##
## Node number 1: 1815 observations,    complexity param=0.01111111
##   predicted class=0  expected loss=0.1239669  P(node) =1
##     class counts:  1590   225
##    probabilities: 0.876 0.124
##   left son=2 (1365 obs) right son=3 (450 obs)
##   Primary splits:
```

```
##       age     < 56.5      to the left,  improve=18.016630, (0 missing)
##       sysBP   < 0.4314421 to the left,  improve=17.268750, (0 missing)
##       glucose < 0.2189266 to the left,  improve= 9.855757, (0 missing)
##       totChol < 0.3302207 to the left,  improve= 2.687579, (0 missing)
##   Surrogate splits:
##       sysBP   < 0.3877069 to the left,  agree=0.774, adj=0.089, (0 split)
##       glucose < 0.2584746 to the left,  agree=0.755, adj=0.013, (0 split)
##       totChol < 0.4151104 to the left,  agree=0.753, adj=0.004, (0 split)
##
## Node number 2: 1365 observations
##   predicted class=0  expected loss=0.08351648  P(node) =0.7520661
##     class counts:  1251   114
##    probabilities: 0.916 0.084
##
## Node number 3: 450 observations,    complexity param=0.01111111
##   predicted class=0  expected loss=0.2466667  P(node) =0.2479339
##     class counts:   339   111
##    probabilities: 0.753 0.247
##   left son=6 (376 obs) right son=7 (74 obs)
##   Primary splits:
##       sysBP   < 0.4314421 to the left,  improve=7.034135, (0 missing)
##       age     < 64.5      to the left,  improve=4.412990, (0 missing)
##       totChol < 0.1400679 to the right, improve=4.380554, (0 missing)
##       glucose < 0.2189266 to the left,  improve=2.600147, (0 missing)
##   Surrogate splits:
##       glucose < 0.7669492 to the left,  agree=0.838, adj=0.014, (0 split)
##
## Node number 6: 376 observations,    complexity param=0.01111111
##   predicted class=0  expected loss=0.2074468  P(node) =0.2071625
##     class counts:   298    78
##    probabilities: 0.793 0.207
##   left son=12 (362 obs) right son=13 (14 obs)
##   Primary splits:
##       totChol < 0.1298812 to the right, improve=5.513594, (0 missing)
##       age     < 64.5      to the left,  improve=2.764295, (0 missing)
##       sysBP   < 0.2801418 to the left,  improve=2.643850, (0 missing)
##       glucose < 0.1313559 to the left,  improve=1.244436, (0 missing)
##
## Node number 7: 74 observations,    complexity param=0.01111111
##   predicted class=0  expected loss=0.4459459  P(node) =0.04077135
##     class counts:    41    33
##    probabilities: 0.554 0.446
##   left son=14 (48 obs) right son=15 (26 obs)
##   Primary splits:
##       age     < 62.5      to the left,  improve=2.301542, (0 missing)
##       sysBP   < 0.5851064 to the left,  improve=2.045201, (0 missing)
##       glucose < 0.1031073 to the left,  improve=1.345345, (0 missing)
##       totChol < 0.2419355 to the left,  improve=1.279689, (0 missing)
##   Surrogate splits:
##       glucose < 0.1426554 to the left,  agree=0.703, adj=0.154, (0 split)
```

```
##       totChol < 0.1842105 to the right, agree=0.689, adj=0.115, (0 split)
##
## Node number 12: 362 observations
##   predicted class=0  expected loss=0.1906077  P(node) =0.199449
##     class counts:   293    69
##    probabilities: 0.809 0.191
##
## Node number 13: 14 observations
##   predicted class=1  expected loss=0.3571429  P(node) =0.007713499
##     class counts:     5     9
##    probabilities: 0.357 0.643
##
## Node number 14: 48 observations
##   predicted class=0  expected loss=0.3541667  P(node) =0.02644628
##     class counts:    31    17
##    probabilities: 0.646 0.354
##
## Node number 15: 26 observations
##   predicted class=1  expected loss=0.3846154  P(node) =0.01432507
##     class counts:    10    16
##    probabilities: 0.385 0.615

CT_pred.result2 <- predict(CT_model2, test, type = 'class')

table(CT_pred.result2, actual.results)

##                  actual.results
## CT_pred.result2   0   1
##                 0 518  73
##                 1  10   3

misClassificError <- mean(CT_pred.result2 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.862582781456954"
```

For model 2 we observed accuracy 86.25% with considering predictors with importance more than 10.

Performing another regression model to check accuracy:

(3) Logistic Regression Analysis:

```
logistic_model1 <- glm(TenYearCHD ~ ., data = train, family = binomial)
summary(logistic_model1)

##
## Call:
## glm(formula = TenYearCHD ~ ., family = binomial, data = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6743  -0.5331  -0.3806  -0.2809   2.7800
##
## Coefficients: (1 not defined because of singularities)
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -5.560635   0.710507  -7.826 5.02e-15 ***
## male                    NA         NA      NA       NA
## age               0.059984   0.011041   5.433 5.54e-08 ***
## education        -0.088297   0.085923  -1.028  0.30413
## currentSmoker    -0.330993   0.256939  -1.288  0.19767
## cigsPerDay        0.034400   0.013034   2.639  0.00831 **
## BPMeds            0.183607   0.294902   0.623  0.53354
## prevalentStroke  1.464501   0.673171   2.176  0.02959 *
## prevalentHyp      0.472125   0.221148   2.135  0.03277 *
## diabetes         -0.002629   0.485778  -0.005  0.99568
## totChol          -0.091011   0.999733  -0.091  0.92746
## sysBP             2.637351   1.145090   2.303  0.02127 *
## diaBP            -0.761296   0.906057  -0.840  0.40078
## BMI               0.077761   0.721167   0.108  0.91413
## heartRate        -0.970351   0.654215  -1.483  0.13801
## glucose           2.371081   1.178310   2.012  0.04419 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1360.4  on 1814  degrees of freedom
## Residual deviance: 1205.2  on 1800  degrees of freedom
## AIC: 1235.2
##
## Number of Fisher Scoring iterations: 5
```

```
logistic_pred.results1 <- predict(logistic_model1, test, type= 'response')
```

```
logistic_pred.results1 <- ifelse(logistic_pred.results1 > 0.5,1,0)
```

```
table(logistic_pred.results1, actual.results)
```

```
##                        actual.results
## logistic_pred.results1   0   1
##                       0 524  75
##                       1   4   1
```

```
misClassificError <- mean(logistic_pred.results1 != actual.results)
print(paste('Accuracy',1-misClassificError))
```

```
## [1] "Accuracy 0.869205298013245"
```

From model 1 of Logistic Regression we can observe that accuracy measures 85.74%

For P values we are getting more than 0.5, trying to remove them and for better result creating second model:

```
logistic_model2 <- glm(TenYearCHD ~ male + age + cigsPerDay + BPMeds + preval
entStroke + prevalentHyp + totChol + sysBP + glucose, data = train, family =
binomial)
summary(logistic_model2)

##
## Call:
## glm(formula = TenYearCHD ~ male + age + cigsPerDay + BPMeds +
##     prevalentStroke + prevalentHyp + totChol + sysBP + glucose,
##     family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6542  -0.5350  -0.3834  -0.2868   2.7389
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -6.447045   0.565129 -11.408  < 2e-16 ***
## male                   NA         NA      NA       NA
## age              0.065705   0.010662   6.163 7.16e-10 ***
## cigsPerDay       0.020515   0.008449   2.428   0.0152 *
## BPMeds           0.192988   0.292030   0.661   0.5087
## prevalentStroke  1.498205   0.665387   2.252   0.0243 *
## prevalentHyp     0.430237   0.216720   1.985   0.0471 *
## totChol         -0.181291   0.992651  -0.183   0.8551
## sysBP            1.940010   0.865570   2.241   0.0250 *
## glucose          2.331279   0.936806   2.489   0.0128 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1360.4  on 1814  degrees of freedom
## Residual deviance: 1211.0  on 1806  degrees of freedom
## AIC: 1229
##
## Number of Fisher Scoring iterations: 5

logistic_pred.results2 <- predict(logistic_model2, test, type='response')

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from a rank-deficient fit may be misleading
```

```
logistic_pred.results2 <- ifelse(logistic_pred.results2 > 0.4,1,0)

table(logistic_pred.results2, actual.results)

##                         actual.results
## logistic_pred.results2   0   1
##                        0 521  71
##                        1   7   5

misClassificError <- mean(logistic_pred.results2 != actual.results)
print(paste('Accuracy', 1-misClassificError))

## [1] "Accuracy 0.870860927152318"
```

For second model we have got 87.09% accuracy with lesser AIC value comparision of model 1.

Second Model would be more significant to select.

(4) Naive Bayes:

```
#install.packages("e1071"")
library(e1071)

## Warning: package 'e1071' was built under R version 3.6.3

Naive_model = train(train_knn,train.result,'nb',trControl=trainControl(method
='cv',number=10))

Naive_model

## Naive Bayes
##
## 1815 samples
##   15 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1633, 1634, 1633, 1633, 1634, 1634, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE            NaN        NaN
##    TRUE      0.8777002  0.0531613
##
## Tuning parameter 'fL' was held constant at a value of 0
```

```
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = TRUE and adju
st
##  = 1.

predict_Naive <- predict(Naive_model,newdata = test )

table(predict_Naive, actual.results)

##                actual.results
## predict_Naive   0   1
##              0 523  73
##              1   5   3

misClassificError <- mean(predict_Naive != actual.results)
print(paste('Accuracy',1-misClassificError))

## [1] "Accuracy 0.870860927152318"
```

(5) Neural Netowrk:

```
#nstall.packages("neuralnet")
library(neuralnet)

## Warning: package 'neuralnet' was built under R version 3.6.3

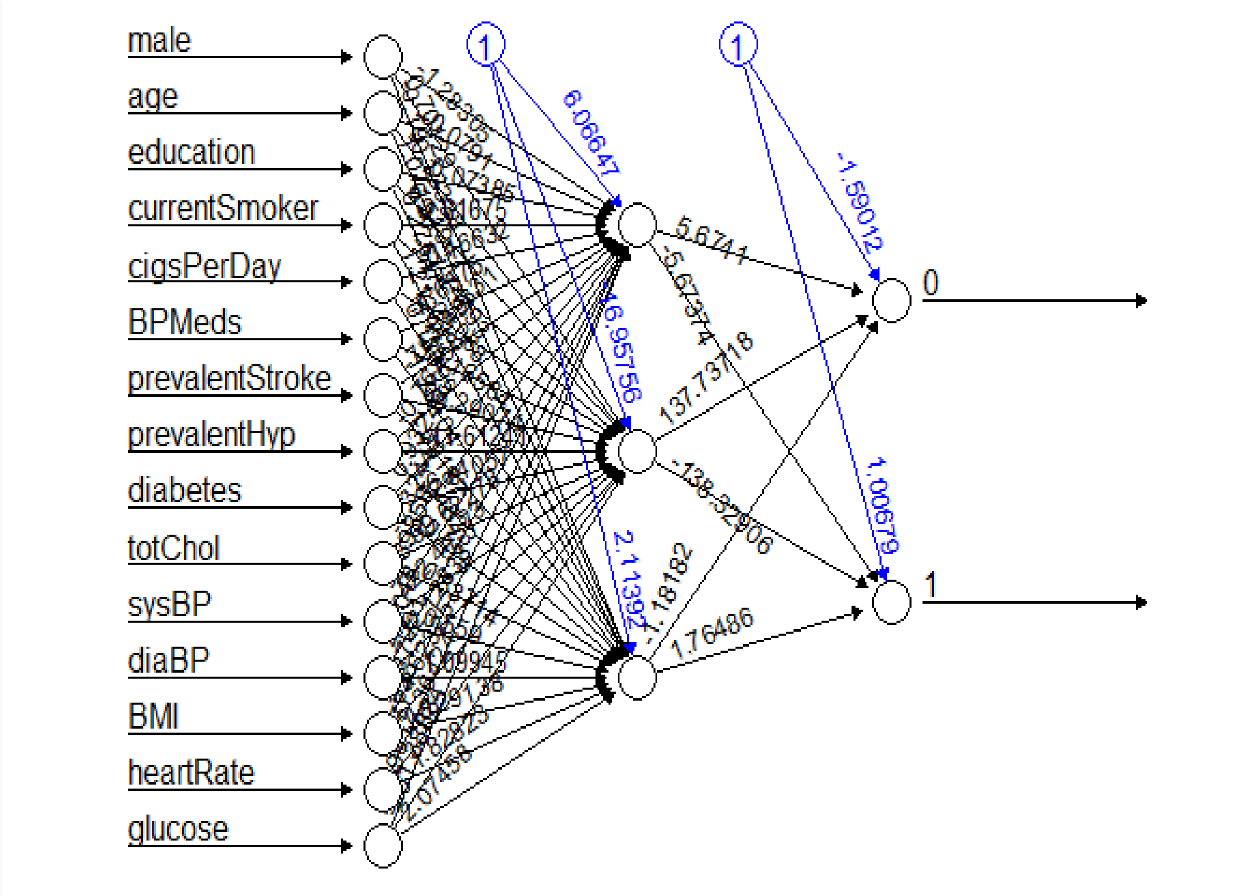
##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##
##     compute

nn = neuralnet(TenYearCHD ~ .,data=train, hidden=3,act.fct = "logistic", line
ar.output = FALSE)

## Warning: Algorithm did not converge in 1 of 1 repetition(s) within the ste
pmax.

plot(nn)
```




```
pred.nn = compute(nn,test)
#pred.nn$net.result

prob.nn <- pred.nn$net.result
pred.nn <- ifelse(prob.nn>0.5, 0, 1)
#pred.nn

table(pred.nn[,1], actual.results)
```

```
##          actual.results
## pred.svm   0   1
##          0 517  67
##          1  11   9
```

```
misClassificError <- mean(pred.nn[,1] != actual.results)
print(paste('Accuracy',1-misClassificError))
```

```
## [1] "Accuracy 0.870860927152318"
```

Here we have got 87.08% accuracy by Neural Network

(6) Support Vector Machine:

```
#install.packages("caret")
library(caret)

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

svm_Linear <- train(TenYearCHD ~ ., data = train, method = "svmLinear", trCon
trol=trctrl, tuneLength = 10)

svm_Linear

## Support Vector Machines with Linear Kernel
##
## 1815 samples
##   15 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1634, 1633, 1634, 1633, 1634, 1634, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8746087  -0.0001524295
##
## Tuning parameter 'C' was held constant at a value of 1

pred.svm <- predict(svm_Linear, newdata = test)
#pred.svm

table(pred.svm, actual.results)

##           actual.results
## pred.svm   0   1
##          0 526  76
##          1   2   0

misClassificError <- mean(pred.svm != actual.results)
print(paste('Accuracy',1-misClassificError))

## [1] "Accuracy 0.870860927152318"
```

Here we have got 87.08% accuracy by Support Vector Machine

Appendix

Codes

```
word_document: default
author: "Mayank And Aashka"
output: word_document
title: 'Project: Prediction of Heart Disease Factors with Machine Learning'
---
```{r}
system("java -version")
library("readxl")
heart_data <- read_excel("C:/Users/mayan/OneDrive/Documents/heart_data.xlsx")
#View(heart_data)
```
```{r}
dim(heart_data)
```
```{r}
str(heart_data)
```
```{r}
heart_data$TenYearCHD <- factor(heart_data$TenYearCHD)
summary(heart_data)
```
```{r}
levels(heart_data$TenYearCHD)
```
```{r}
sum(is.na(heart_data))
```
```{r}
heart_data <- na.omit(heart_data)
sum(is.na(heart_data))
```
```{r}
normalize <- function(x) { (x - min(x)) / (max(x) - min(x))}
heart_data$totChol <- normalize(heart_data$totChol)
heart_data$sysBP <- normalize(heart_data$sysBP)
heart_data$diaBP <- normalize(heart_data$diaBP)
heart_data$BMI <- normalize(heart_data$BMI)
heart_data$heartRate <- normalize(heart_data$heartRate)
heart_data$glucose <- normalize(heart_data$glucose)

#install.packages("ggplot2")
```
```

````
#install.packages("GGally")
#install.packages("corrplot")

library(ggplot2)
library(GGally)
library(tidyverse)
library(dplyr)
correlation <- cor(heart_data [,-16], method = "pearson" , use = "complete.obs")
library(corrplot)
round(correlation,2)
whiteblack <- c("white", "black")
corrplot(correlation, order = "hclust",  bg = "darkorange")
```
```{r}
boxplot(heart_data$male, heart_data$age, heart_data$education,heart_data$currentSmoker,
heart_data$cigsPerDay, heart_data$BPMeds, heart_data$prevalentStroke,
heart_data$prevalentHyp, notch=FALSE,col=(c("red","darkgreen")),main="Heart Data")

boxplot(heart_data$diabetes, heart_data$totChol, heart_data$sysBP, heart_data$diaBP,
heart_data$BMI, heart_data$heartRate, heart_data$glucose, heart_data$TenYearCHD,
notch=TRUE,col=(c("darkslategray1","goldenrod1")))

boxplot(heart_data$male, heart_data$age, heart_data$education,heart_data$currentSmoker,
heart_data$cigsPerDay, heart_data$BPMeds, heart_data$prevalentStroke,
heart_data$prevalentHyp, plot=FALSE)$out

outliers <- boxplot(heart_data$male, heart_data$age,
heart_data$education,heart_data$currentSmoker, heart_data$cigsPerDay, heart_data$BPMeds,
heart_data$prevalentStroke, heart_data$prevalentHyp, plot=FALSE)$out

heart_data <- heart_data[-which(c(heart_data$male, heart_data$age,
heart_data$education,heart_data$currentSmoker, heart_data$cigsPerDay, heart_data$BPMeds,
heart_data$prevalentStroke, heart_data$prevalentHyp) %in% outliers),]

boxplot(heart_data)
```
```{r}
set.seed(123)
library(caTools)
sample <- sample.split(heart_data, SplitRatio = 0.80)
train <- subset(heart_data, sample == TRUE)
test1 <- subset(heart_data, sample == FALSE)
test <- test1[,-16]
train_knn <- train[,-16]
actual.results <- as.vector(test1$TenYearCHD)
train.result <- as.vector(train$TenYearCHD)
```
````
````

````
```{r}
# (1) K-nearest neighbours algorithms:
library(class)
library(gmodels)

#For k=1
knn_pred.result1 <- knn(train_knn, test, train.result, k=1 )
table(knn_pred.result1, actual.results)
misClassificError <- mean(knn_pred.result1 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=2
knn_pred.result2 <- knn(train_knn, test, train.result, k=2 )
table(knn_pred.result2, actual.results)
misClassificError <- mean(knn_pred.result2 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=3
knn_pred.result3 <- knn(train_knn, test, train.result, k=3 )
table(knn_pred.result3, actual.results)
misClassificError <- mean(knn_pred.result3 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=4
knn_pred.result4 <- knn(train_knn, test, train.result, k=4 )
table(knn_pred.result4, actual.results)
misClassificError <- mean(knn_pred.result4 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=5
knn_pred.result5 <- knn(train_knn, test, train.result, k=5 )
table(knn_pred.result5, actual.results)
misClassificError <- mean(knn_pred.result5 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=6
knn_pred.result6 <- knn(train_knn, test, train.result, k=6 )
table(knn_pred.result6, actual.results)
misClassificError <- mean(knn_pred.result6 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=7
knn_pred.result7 <- knn(train_knn, test, train.result, k=7 )
table(knn_pred.result7, actual.results)
misClassificError <- mean(knn_pred.result7 != actual.results)
print(paste('Accuracy', 1-misClassificError))
````
````

~~~~
#For k=8
knn_pred.result8 <- knn(train_knn, test, train.result, k=8 )
table(knn_pred.result8, actual.results)
misClassificError <- mean(knn_pred.result8 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=9
knn_pred.result9 <- knn(train_knn, test, train.result, k=9 )
table(knn_pred.result9, actual.results)
misClassificError <- mean(knn_pred.result9 != actual.results)
print(paste('Accuracy', 1-misClassificError))

#For k=10
knn_pred.result10 <- knn(train_knn, test, train.result, k=10 )
table(knn_pred.result10, actual.results)
misClassificError <- mean(knn_pred.result10 != actual.results)
print(paste('Accuracy', 1-misClassificError))

```
```{r}
# (2) CART : Classification and Regression Tree:
library(rpart)
library(rpart.plot)

CT_model1 <- rpart(TenYearCHD ~ . , data = train, method = 'class', control =
                rpart.control(minsplit = 30, cp=0.0055))
printcp(CT_model1)
rpart.plot(CT_model1, type = 1, extra = 1, split.font = 1, varlen = -20)
summary(CT_model1)

CT_pred.result1 <-predict(CT_model1, test, type = 'class')

table(CT_pred.result1, actual.results)
misClassificError <- mean(CT_pred.result1 != actual.results)
print(paste('Accuracy', 1-misClassificError))
```
```{r}
CT_model2 <- rpart(TenYearCHD ~ age + sysBP + glucose + totChol, data = train, method =
'class', control=rpart.control(minsplit=30, cp=0.0055))
print(CT_model2)
rpart.plot(CT_model2, type = 1, extra = 1, split.font = 1, varlen = -20)
summary(CT_model2)

CT_pred.result2 <- predict(CT_model2, test, type = 'class')

table(CT_pred.result2, actual.results)
misClassificError <- mean(CT_pred.result2 != actual.results)
~~~~

````
print(paste('Accuracy', 1-misClassificError))
```
```{r}
#(3) Logistic Regression Analysis:
logistic_model1 <- glm(TenYearCHD ~ ., data = train, family = binomial)
summary(logistic_model1)

logistic_pred.results1 <- predict(logistic_model1, test, type= 'response')
logistic_pred.results1 <- ifelse(logistic_pred.results1 > 0.5,1,0)

table(logistic_pred.results1, actual.results)
misClassificError <- mean(logistic_pred.results1 != actual.results)
print(paste('Accuracy',1-misClassificError))
```
```{r}
logistic_model2 <- glm(TenYearCHD ~ male + age + cigsPerDay + BPMeds + prevalentStroke
+ prevalentHyp + totChol + sysBP + glucose, data = train, family = binomial)
summary(logistic_model2)

logistic_pred.results2 <- predict(logistic_model2, test, type='response')
logistic_pred.results2 <- ifelse(logistic_pred.results2 > 0.4,1,0)

table(logistic_pred.results2, actual.results)
misClassificError <- mean(logistic_pred.results2 != actual.results)
print(paste('Accuracy', 1-misClassificError))
```
```{r}
# (4) Naive Bayes:
#install.packages('ellipse')

#install.packages('e1071')
library(e1071)

Naive_model = train(train_knn,train.result,'nb',trControl=trainControl(method='cv',number=10))

Naive_model

predict_Naive <- predict(Naive_model,newdata = test )

table(predict_Naive, actual.results)
misClassificError <- mean(predict_Naive != actual.results)
print(paste('Accuracy',1-misClassificError))
```
```{r}
# (5) Neural Netowrk:
#install.packages('neuralnet')
library(neuralnet)
````

````
nn = neuralnet(TenYearCHD ~ .,data=train, hidden=3,act.fct = "logistic", linear.output = TRUE)

plot(nn)

pred.nn = compute(nn,test)

prob.nn <- pred.nn$net.result
pred.nn <- ifelse(prob.nn>0.5, 0, 1)
pred.nn

table(pred.nn[,1], actual.results)
misClassificError <- mean(pred.nn[,1] != actual.results)
print(paste('Accuracy',1-misClassificError))
```
```{r}
# (6) Support Vector Machine:
#install.packages("caret")
library(caret)

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 5)

svm_Linear <- train(TenYearCHD ~ ., data = train, method = "svmLinear", trControl=trctrl,
tuneLength = 10)

svm_Linear

pred.svm <- predict(svm_Linear, newdata = test)
#pred.svm

table(pred.svm, actual.results)
misClassificError <- mean(pred.svm != actual.results)
print(paste('Accuracy',1-misClassificError))
```
````
````

V. Performance Evaluation:

As we can see in results above, efficiency of different models are as follows:
1) For K-NN, efficiency is 86.58% for k=6.
2) For logistic regression, we got 87.09% efficiency
3) For CART, we got 86.92%
4) For Naïve Bayes, it is 87.08%
5) For Neural Networks, it is it is 87.58%
6) For Support Vector Machine, we have got 87.08%

VI. Discussion and Recommendation:

- We have created models using six different supervised learning algorithms and generated confusion matrix and errors for each model. We have selected the model which predicts value with the highest efficiency.
- Since, it is binary classification problem, performing logistic regression was a starting point. Then, we performed CART and K-NN algorithms respectively and selected the best possible models for them. Afterwards, we have performed Naïve Bayes, Neural Networks, Support Vector Machine respectively.
- Out of these six models, gave us the highest accuracy as stated in Performance evaluation.
- From analysis and observation, we can say that Neural Network gives the highest efficiency for this model with 87.58% amongst six models.
- We also recommend using random forest and boosting algorithm for potential improvement.