Part B - Assignment No: 2

Perform the following operations using R/Python on the Air quality and Heart Diseases data sets

e) Data model building

```
#Read HeartDisease.CSV

hdata=read.csv(file="/home/hduser/Desktop/heartdisease.csv",
header=TRUE,sep=",")
names(hdata)
str(hdata)
dim(hdata)

# dumifying variable num, if num!-0means,
hdata$num[hdata$num>0]<-1
summary(hdata$num)

# barplot fate(i.e. hear disease 1 or not 0)
barplot(table(hdata$num), main="Fate", col="black")

#plot sex vs fate using mosaicplot
mosaicplot(hdata$sex ~ hdata$num,main="Fate by Gender",
shade=FALSE,color=TRUE,xlab="Gender", ylab="Heart disease")

#fate by age using boxplot
boxplot(hdata$age ~ hdata$num,main="Fate by Age",ylab="Age",xlab="Heart
disease")

#replacing ? by NA
levels(hdata$thal)[levels(hdata$thal)=="?"]<-NA
table(hdata$thal)

# 3   6   7
# 166  18 117
#replacing NA with max factor
hdata$thal[is.na(hdata$thal)]<-3
table(hdata$thal)

# 3   6   7
# 168  18 117

levels(hdata$ca)[levels(hdata$ca)=="?"]<-NA

table(hdata$ca)

# 0   1   2   3
# 176  65  38  20
h$ca[is.na(hdata$ca)]<-0
table(hdata$ca)

# 0   1   2   3
# 180  65  38  20

dim(hdata)
# [1] 303  14

# Data Model Building
```

```
# Step1 : Divide the dataset into taining and Testing
library(caTools)
hdata[, c(1)] <- sapply(hdata[, c(1)], as.numeric)
set.seed(123)
split = sample.split(hdata$num, SplitRatio = 2/3)
train_hdata = subset(hdata, split == TRUE)
test_hdata = subset(hdata, split == FALSE)

#You can use following code for creating training and testing samples,
#but you cannot get random samples which is possible with above split and subset
function
# train_hdata=hdata[1:212,]
# test_hdata=hdata[213:303,]
dim(train_hdata)
#[1] 212  14
dim(test_hdata)
#[1] 91 14


# Step 2: Use prediction Model using any of the technique-like regression,
classification and clustering
# here I have used Technique 1-Linear regression, 2-Multiple regression, 3-kNN,
4-Naive Bayes technique for prediction

# Technique 1: Linear regression
# Here for hear disease dataset, Variable age is IV and num is IV for linear
regression model
# fitting simple linear Regression to the training set
library(caTools)
regressor=lm(formula = num~age, data=train_hdata)

#predicting the test set result using regressor
hd_age_predict=predict(regressor, newdata=test_hdata)

# As the result is not whole number, rounding the result
round_age=hd_age_predict
rage=round(round_age)

# Displaying the accuracy using confusion Matrix
library(e1071)
library(caret)
df=confusionMatrix(rage,test_hdata$num)


# Confusion Matrix and Statistics

# Reference
# Prediction  0  1
# 0 35 20
# 1 20 26
#
# Accuracy : 0.604
# 95% CI : (0.5017, 0.6999)
# No Information Rate : 0.5446
# P-Value [Acc > NIR] : 0.1357
#
# Kappa : 0.2016
# Mcnemar's Test P-Value : 1.0000
#
#              Sensitivity : 0.6364
#              Specificity : 0.5652
#           Pos Pred Value : 0.6364
#           Neg Pred Value : 0.5652
#               Prevalence : 0.5446
```

```
#          Detection Rate : 0.3465
#    Detection Prevalence : 0.5446
#       Balanced Accuracy : 0.6008
#
#        'Positive' Class : 0


#----------------------------------------------------------
# Technique 2: Multiple regression
# prediction using multiple linear regression
# Here DV=num, and IV= all reset of the variables in dataset
# fitting multiple Regression to the training set
regressor=lm(formula =
num~age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope,
data=train_hdata)

# predicting the test set result
hd_age_predict=predict(regressor, newdata=test_hdata)

# As the result is not whole number, rounding the result
round_age=hd_age_predict
rage=round(round_age)


library(e1071)
library(caret)
df=confusionMatrix(rage,test_hdata$num)

# Confusion Matrix and Statistics
#
# Reference
# Prediction  0  1
# 0 45 13
# 1 10 33
#
# Accuracy : 0.7723
# 95% CI : (0.6782, 0.8498)
# No Information Rate : 0.5446
# P-Value [Acc > NIR] : 1.749e-06
#
# Kappa : 0.5384
# Mcnemar's Test P-Value : 0.6767
#
#             Sensitivity : 0.8182
#             Specificity : 0.7174
#          Pos Pred Value : 0.7759
#          Neg Pred Value : 0.7674
#              Prevalence : 0.5446
#          Detection Rate : 0.4455
#    Detection Prevalence : 0.5743
#       Balanced Accuracy : 0.7678
#
#        'Positive' Class : 0
#

#--------------------------------------------------------------------------------
--
# Technique 3: k-Nearest Neighbour claasifer
# Prediction using KNN
# Use data transformation technique such as scaling and normalization for
normalizing dataset
# Writting the function for normalizing the values of all variables
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
```

```
h1data<-hdata
# using Normalize function and applying it on dataset
h1data_n <- as.data.frame(lapply(h1data[1:11], normalize))
#h1data_n[12:13]<-h1data[12:13]
# Dividing dataset into training and testing
traink_hdata=h1data_n[1:212,]
testk_hdata=h1data_n[213:303,]

library(class)
h1data_train_labels <- hdata[1:212, 14]
h1data_test_labels <- hdata[213:303, 14]

# Applying Knn function on dataset
h1data_test_pred <- knn(train = traink_hdata, test = testk_hdata,cl =
h1data_train_labels, k=17)

# Another method of getting confusion matrix using CrossTable
library(gmodels)
CrossTable(x=h1data_test_labels,y=h1data_test_pred,prop.chisq = FALSE)

table(h1data_test_labels,h1data_test_pred)
#              h1data_test_pred
# h1data_test_labels  0  1
#               0 39  9
#               1 17 26

# round_age=hd_age_predict
# rage=round(round_age)

# Displaying the accuracy using confusion Matrix
library(e1071)
library(caret)
df=confusionMatrix(h1data_test_labels,h1data_test_pred)

# Confusion Matrix and Statistics
#
# Reference
# Prediction  0  1
# 0 39  9
# 1 17 26
#
# Accuracy : 0.7143
# 95% CI : (0.61, 0.8041)
# No Information Rate : 0.6154
# P-Value [Acc > NIR] : 0.0317
#
# Kappa : 0.4212
# Mcnemar's Test P-Value : 0.1698
#
#             Sensitivity : 0.6964
#             Specificity : 0.7429
#          Pos Pred Value : 0.8125
#          Neg Pred Value : 0.6047
#              Prevalence : 0.6154
#          Detection Rate : 0.4286
#    Detection Prevalence : 0.5275
#       Balanced Accuracy : 0.7196
#
#        'Positive' Class : 0
#---------------------------------------------------------------------------
# Technique 4:  Naive bayes classifyer
# Naive bayes classifyer needs catagorical data for prediction
# Preparing data for Naive Bayes
```

```
h1data<-hdata
h1data$age=factor(h1data$age)
h1data$sex=factor(h1data$sex)
h1data$cp=factor(h1data$cp)
h1data$trestbps=factor(h1data$trestbps)
h1data$chol=factor(h1data$chol)
h1data$fbs=factor(h1data$fbs)
h1data$restecg=factor(h1data$restecg)
h1data$thalach=factor(h1data$thalach)
h1data$exang=factor$exang
h1data$exang=factor(h1data$exang)
h1data$oldpeak=factor(h1data$oldpeak)
h1data$slope=factor(h1data$slope)
h1data$num=factor(h1data$num)

# Dividing dataset into training and testing
trainnb_hdata=h1data[1:212,]
testnb_hdata=h1data[213:303,-14]

# Applying Naive Bayes claasifier on dataset
library(e1071)
classifier <- naiveBayes(num
~age+sex+cp+trestbps+chol+fbs+restecg+thalach+exang+oldpeak+slope,trainnb_hdata)
prediction <- predict(classifier, testnb_hdata ,type="class")


prediction
# [1] 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0
0 1 1 1 0 0 0
# [45] 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 1 0 1 0 0 1 1 0 0 0 1 0 1
1 1 0 1 0 0 1
# [89] 1 0 0
# Levels: 0 1
table(prediction, h1data[213:303,14])

# prediction  0  1
# 0 41 18
# 1  7 25

# Displaying the accuracy using confusion Matrix
library(e1071)
library(caret)
df=confusionMatrix(h1data[213:303,14],prediction)

# Confusion Matrix and Statistics
#
# Reference
# Prediction  0  1
# 0 41  7
# 1 18 25
#
# Accuracy : 0.7253
# 95% CI : (0.6217, 0.8137)
# No Information Rate : 0.6484
# P-Value [Acc > NIR] : 0.07486
#
# Kappa : 0.4414
# Mcnemar's Test P-Value : 0.04550
#
#              Sensitivity : 0.6949
#              Specificity : 0.7812
#           Pos Pred Value : 0.8542
#           Neg Pred Value : 0.5814
#               Prevalence : 0.6484
```

```
#          Detection Rate : 0.4505
#    Detection Prevalence : 0.5275
#       Balanced Accuracy : 0.7381
#
#        'Positive' Class : 0
#
```