**Gagan Kumar Soni  
Roll No:. 19035570036/0814**

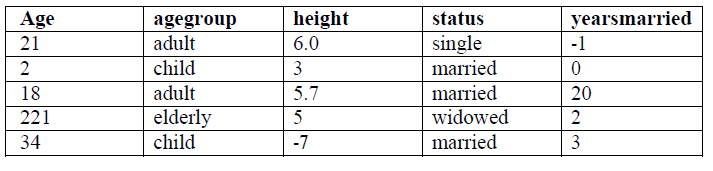
**Bsc. Hons Computer Science**

**Section A**

**Data Mining Practical  
Keshav Mahavidyalaya, University of Delhi**

Ques 1:-

Create a file “people.txt” with the following data:



i) Read the data from the file “people.txt”.

ii) Create a ruleset E that contain rules to check for the following conditions:

1. The age should be in the range 0-150.

2. The age should be greater than yearsmarried.

3. The status should be married or single or widowed.

4. If age is less than 18 the agegroup should be child, if age is between 18 and 65 the agegroup should be adult, if age is more than 65 the agegroup should be elderly.

iii) Check whether ruleset E is violated by the data in the file people.txt.

iv) Summarize the results obtained in part (iii)

v) Visualize the results obtained in part (iii)

Code:-

#Reading file people.txt

people = read.table(file = 'people.txt', sep = '\t',header = TRUE)

print(people)

#install package edit rules

#install.packages("editrules")

library(editrules)

#Create a ruleset E that contain rules to check for the following conditions:

#1. The age should be in the range 0-150.

#2. The age should be greater than yearsmarried.

#3. The status should be married or single or widowed.

#4. If age is less than 18 the agegroup should be child, if age is between 18 and 65

# the agegroup should be adult, if age is more than 65 the agegroup should be elderly.

E<-editset(expression(

Age>=0,

Age<=150,

Age>yearsmarried,

status %in% c('married','single','widowed'),

if(Age<18) agegroup=='child',

if(Age>=18 & Age<65) agegroup=='adult',

if(Age>=65) agegroup=='elderly'

))

print(E)

ve<-violatedEdits(E,people)

summary(ve)

print(ve)

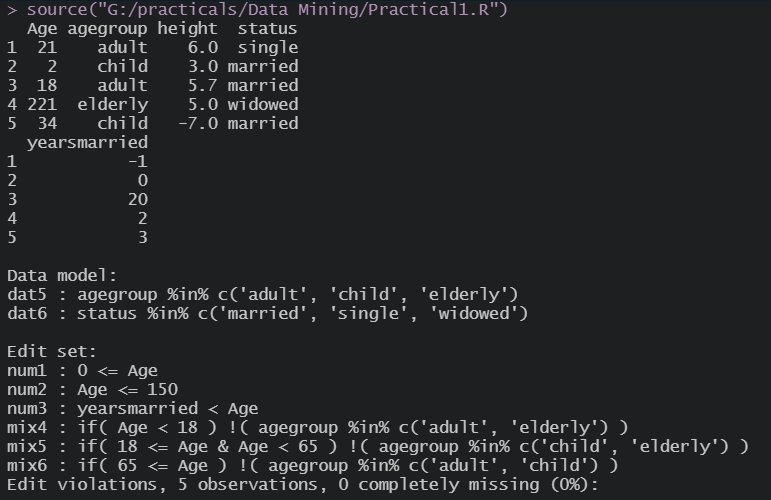
summary(ve,E)

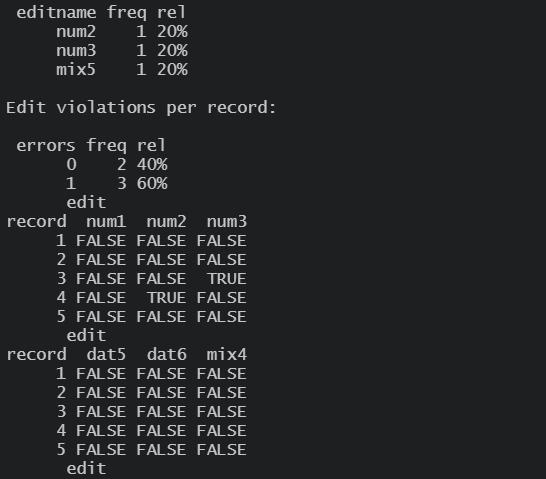
plot(E)

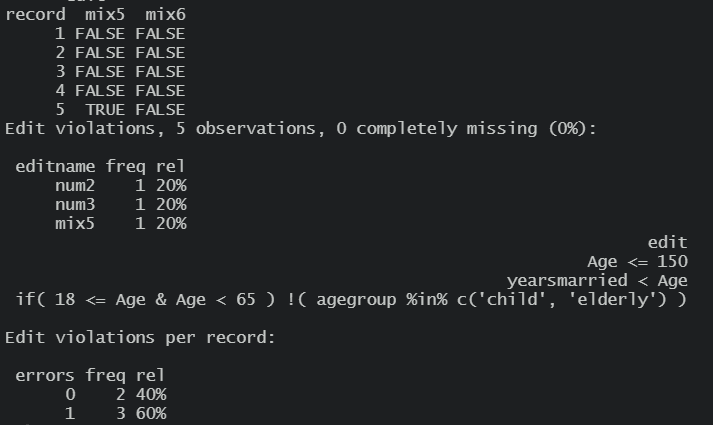
plot(ve)

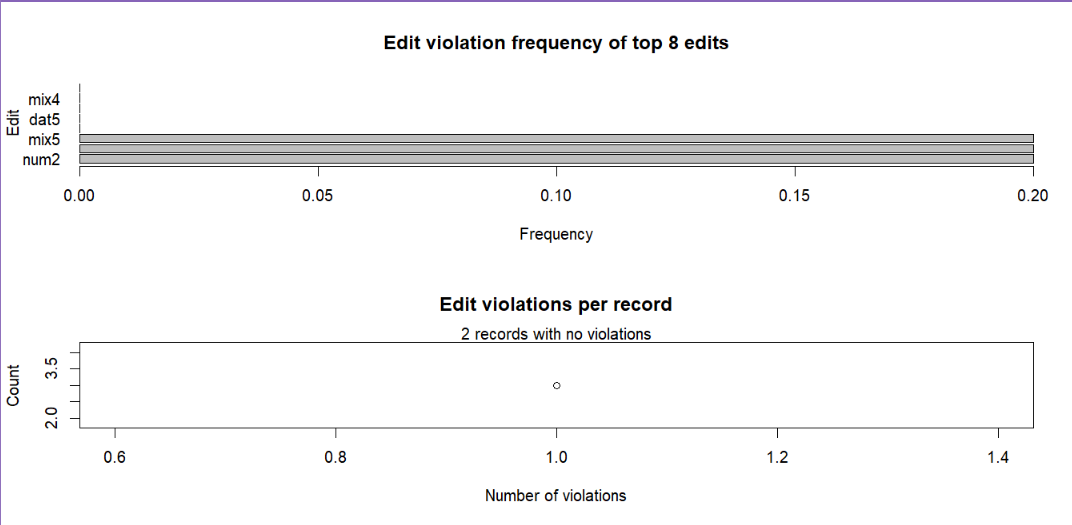
plot(E,layout=layout.circle)

Output Screen:-









Ques 2:-

Perform the following preprocessing tasks on the dirty\_iris datasetii.

i) Calculate the number and percentage of observations that are complete.

ii) Replace all the special values in data with NA.

iii) Define these rules in a separate text file and read them.

(Use editfile function in R (package editrules). Use similar function in Python).

Print the resulting constraint object.

– Species should be one of the following values: setosa, versicolor or virginica.

– All measured numerical properties of an iris should be positive.

– The petal length of an iris is at least 2 times its petal width.

– The sepal length of an iris cannot exceed 30 cm.

– The sepals of an iris are longer than its petals.

iv) Determine how often each rule is broken (violatedEdits). Also summarize and plot the

result.

v) Find outliers in sepal length using boxplot and boxplot.stats

Rules.txt:-

Species %in% c("setosa", "versicolor", "virginica")

Sepal.Length > 0

Sepal.Width > 0

Petal.Length > 0

Petal.Width > 0

Petal.Length >= 2\*Petal.Width

Sepal.Length < 30

Sepal.Length > Petal.Length

Code:-

data<-read.table(file = 'dirty\_iris.csv', sep=',', header = TRUE)

print(head(data,10))

print(summary(data))

str(data)

#i)Calculate the number and percentage of observations that are complete.

complete\_cases <- complete.cases(data)

as.numeric(complete\_cases)

n\_complete<-sum(complete\_cases)

print(n\_complete)

percent\_complete<- 100\*n\_complete/nrow(data)

print(paste("Cases in DIRTY\_IRIS dataframe that are complete: ", percent\_complete, "%"))

#ii)Replace all the special values in data with NA.

is.na(data)<- sapply(data,is.infinite)

is.na(data)

#iii)Define these rules in a separate text file and read them.

#(Use editfile function in R (package editrules).

#Print the resulting constraint object.

#-Species should be one of the following values: setosa, versicolor or virginica.

#-All measured numerical properties of an iris should be positive.

#-The petal length of an iris is at least 2 times its petal width.

#-The sepal length of an iris cannot exceed 30 cm.

#-The sepals of an iris are longer than its petals.

library(editrules)

(Rules <- editfile('ques2Rules.txt'))

#iv)Determine how often each rule is broken (violatedEdits). Also summarize and plot the result.

print(Rules)

rules\_violations <- violatedEdits(Rules,data)

print(rules\_violations)

summary(rules\_violations)

#par(mar=c(3,3,3,3))

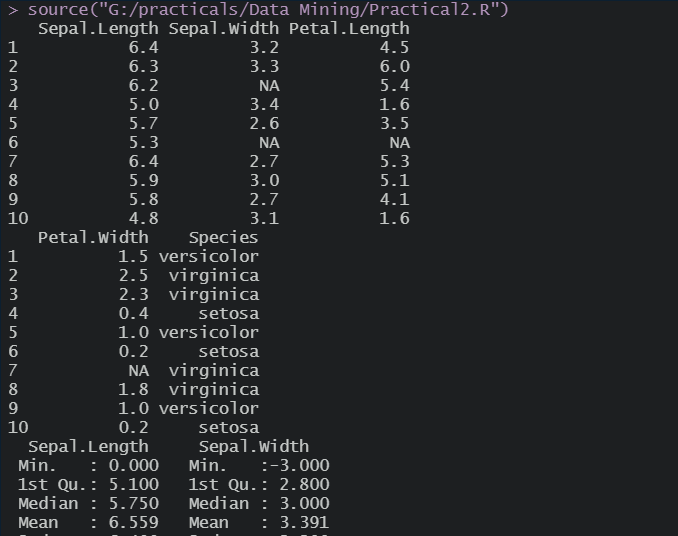
plot(rules\_violations)

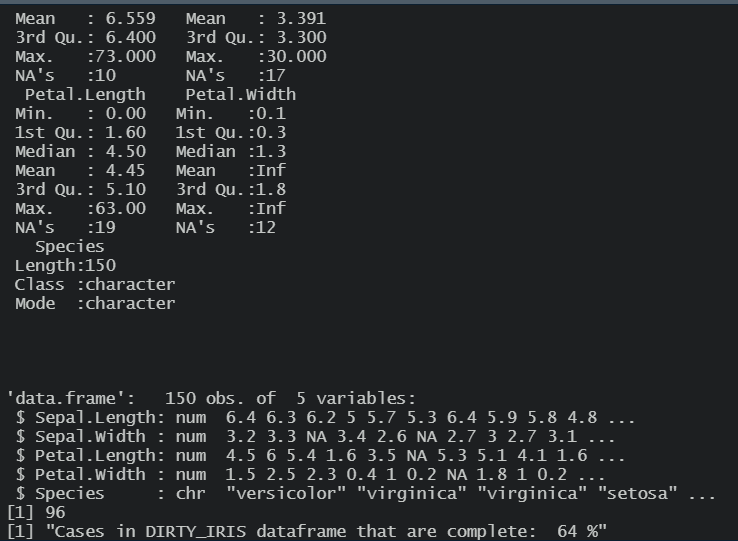
#v)Find outliers in sepal length using boxplot and boxplot.stats

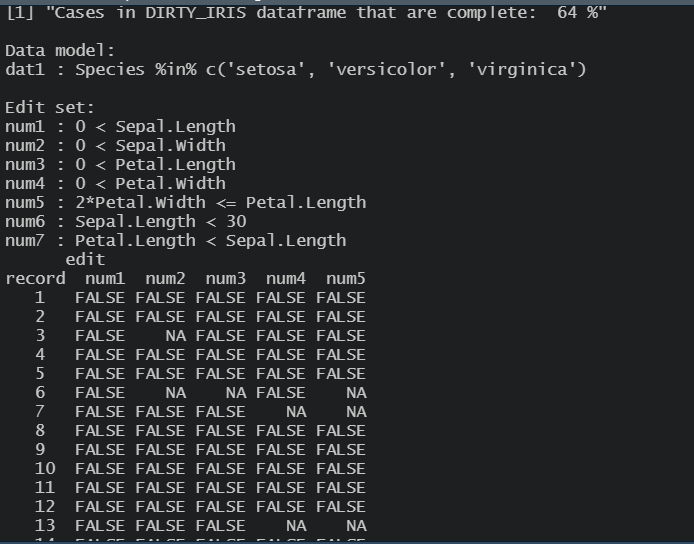
boxplot(data$Sepal.Length, horizontal=TRUE)

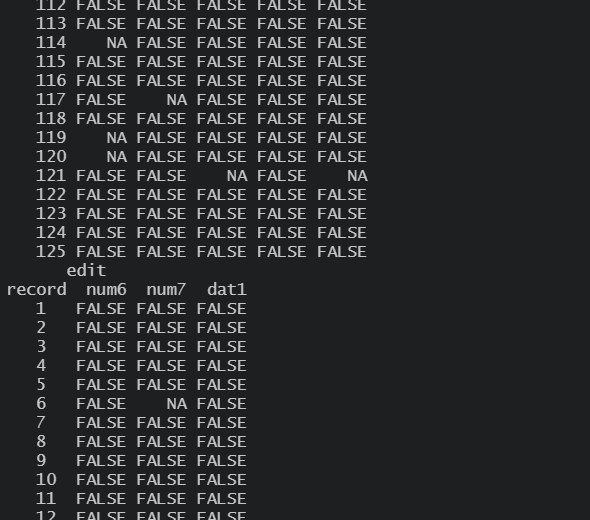
print(boxplot.stats(data$Sepal.Length))

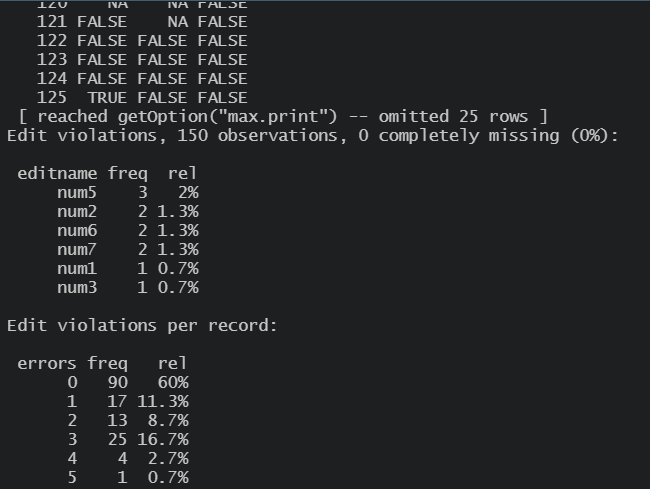
Output Screen:-

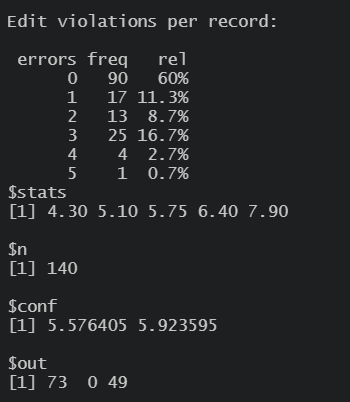


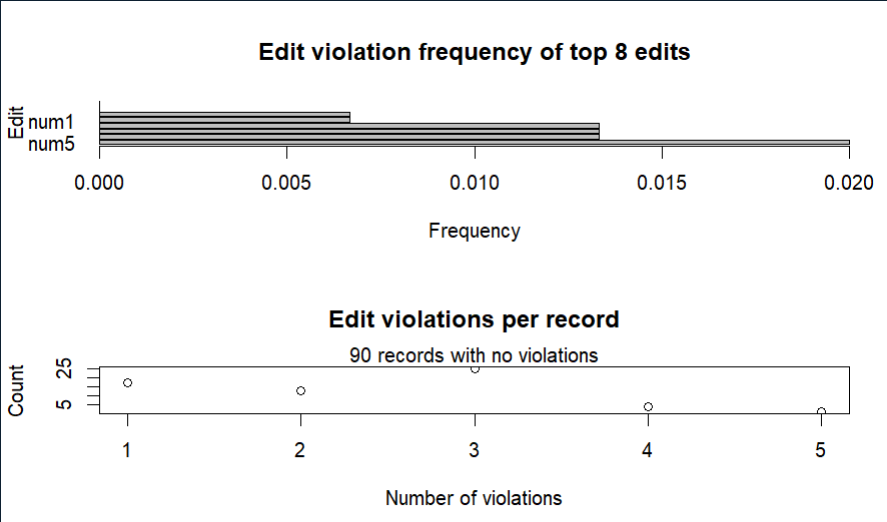


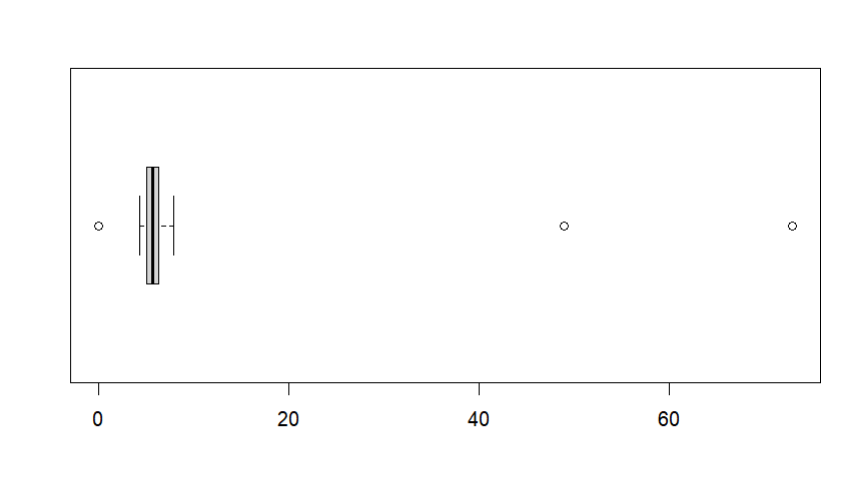












Ques 3:-

Load the data from wine dataset. Check whether all attributes are standardized or not (mean is 0 and standard deviation is 1). If not, standardize the attributes. Do the same with Iris dataset.

Code:-

library(caret)

#----WINE DATASET-----

df\_wine<-read.csv("wine.csv", header=FALSE)

View(df\_wine)

str(df\_wine)

summary(df\_wine)

apply(df\_wine,2,sd)

wine\_pre<-preProcess(df\_wine[,],method=c("center","scale"))

wine\_standard<-predict(wine\_pre,df\_wine[,])

summary(wine\_standard)

apply(wine\_standard,2,sd)

#----IRIS DATASET-----

data("iris")

View(iris)

summary(iris)

apply(iris[,1:4],2,sd)

#for making mean = 0 and standard deviation = 1 in IRIS dataset.

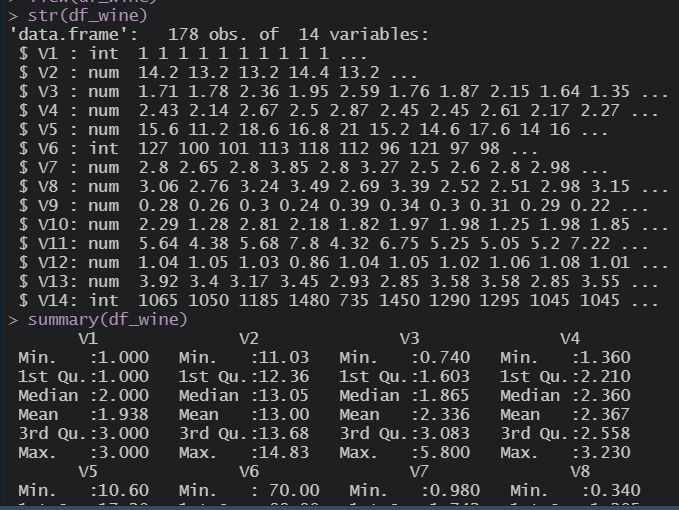
iris\_pre<-preProcess(iris[,1:4], method=c("center", "scale"))

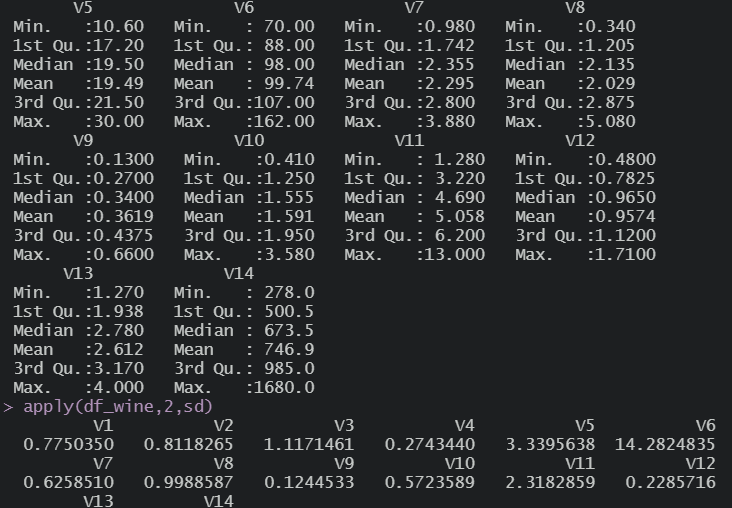
iris\_standard<-predict(iris\_pre, iris[,1:4])

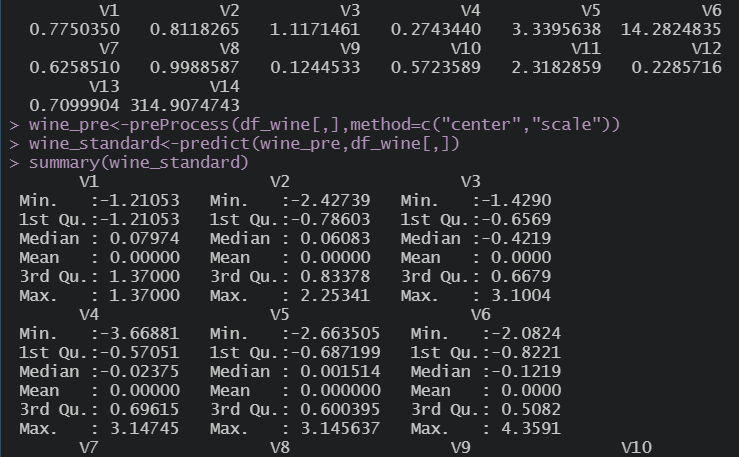
summary(iris\_standard)

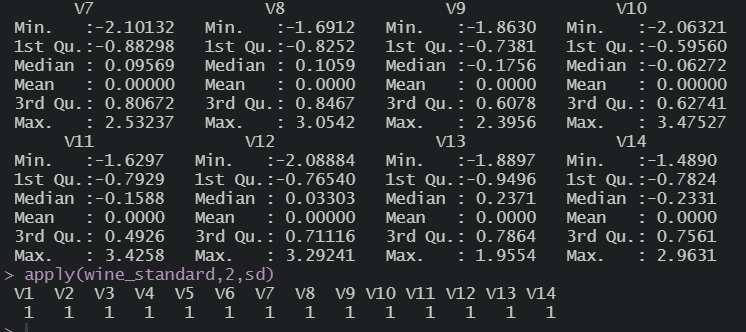
apply(iris\_standard,2,sd)

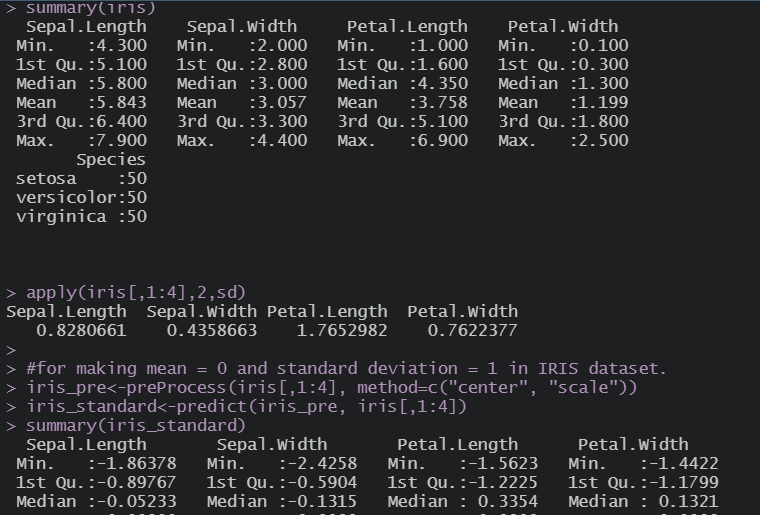
Output Screen:-

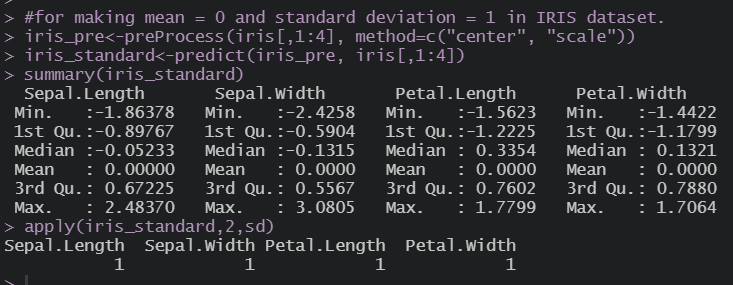












Ques 4:-

Run Apriori algorithm to find frequent itemsets and association rules

1.1 Use minimum support as 50% and minimum confidence as 75%

1.2 Use minimum support as 60% and minimum confidence as 60 %

Code:-

receipt\_df <- read.csv("datasets/Dataset for Apriori/1000/1000i.csv")

View(receipt\_df)

#Applying column names

names(receipt\_df) <- c("Receipt\_Number", "Food", "Quantity")

#Applying Data Preprocessing

id<- c(1:5)

food<- c("milk", "sugar", "chocolate", "apple", "curd")

df <- data.frame(id, food)

print(df)

#map id to its text representation

receipt\_df$Food <- df$food[match(receipt\_df$Food,df$id)]

#after Preprocessing

head(receipt\_df)

typeof(receipt\_df)

library(arules)

library(arulesViz)

#convert into basket format to run in apriori

test\_df <- receipt\_df[,c("Receipt\_Number", "Food", "Quantity")]

df\_trans <- as(split(test\_df$Food, test\_df$Receipt\_Number),"transactions")

#support and confidence

#part 1:- minimum support as 50% and minimum confidence as 75%

rules <- apriori(df\_trans,parameter = list(supp = .5, conf = .5))

plot(rules)

rules<- sort(rules, by="support", decreasing = T)

inspect(head(rules))

#part 2:- minimum support as 60% and minimum confidence as 60%

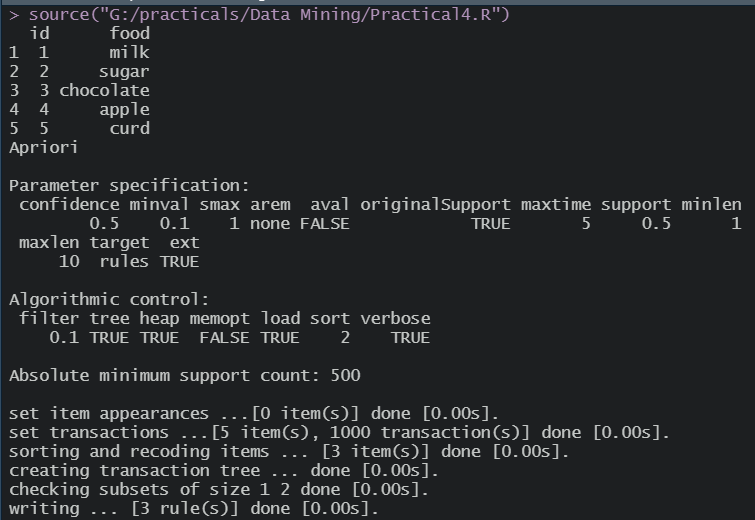
rules <- apriori(df\_trans,parameter = list(supp = .6, conf = .6))

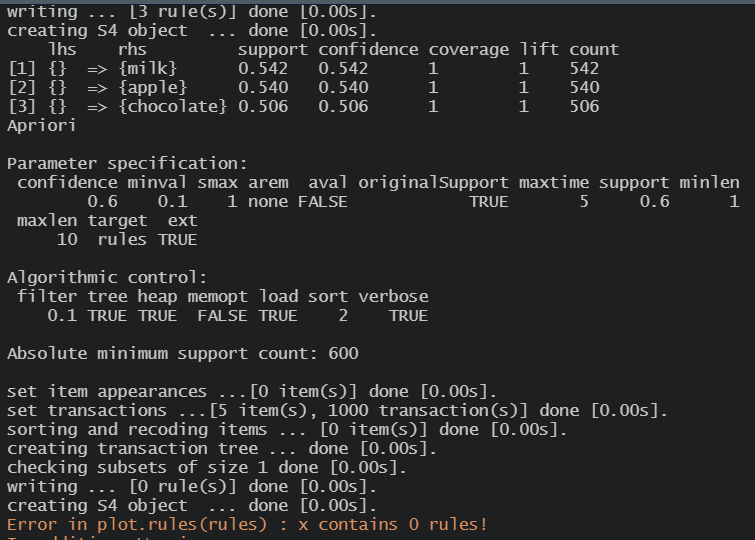
plot(rules)

rules<- sort(rules, by="support", decreasing = T)

inspect(head(rules))

Output Screen:-





Ques 5:-

Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set into training and test set. Compare the accuracy of the different classifiers under the following situations:

5.1 a) Training set = 75% Test set = 25% b) Training set = 66.6% (2/3rd of total), Test set = 33.3%

5.2 Training set is chosen by i) hold out method ii) Random subsampling iii) Cross-Validation. Compare the accuracy of the classifiers obtained.

5.3 Data is scaled to standard format.

Code:-

#Use Naive bayes, K-nearest, and Decision tree classification algorithms

#and build classifiers. Divide the data set into training and test set.

#Compare the accuracy of the different classifiers under the following

#situations:

#5.1 a) Training set = 75% Test set = 25%

#b) Training set = 66.6% (2/3rd of total), Test set = 33.3%

#5.2 Training set is chosen by i) hold out method

#ii) Random subsampling iii) Cross-Validation.

#Compare the accuracy of the classifiers obtained.

#5.3 Data is scaled to standard format.

data(iris)

print(summary(iris))

print(head(iris,5))

#library imports

library(caTools)

library(rpart)

#?rpart

library(e1071)

library(class)

library(rpart.plot)

library(caret)

#part a

print("Part a start Training set = 75% Test set = 25% ")

#Hold-out Method

print("Hold out Method")

set.seed(143)

split<-sample.split(iris$Species,SplitRatio = 0.75)

training\_set = subset(iris,split==TRUE)

testing\_set = subset(iris,split==FALSE)

#summary(training\_set)

#summary(testing\_set)

#feature scaling

train\_scale <- scale(training\_set[,1:4])

test\_scale <- scale(testing\_set[,1:4])

print(dim(training\_set))

print(dim(testing\_set))

#Naive bayes

print("Naive bayes-Hold out-part a")

classifier\_naive <- naiveBayes(Species~., data = training\_set)

print(classifier\_naive)

y\_pred <- predict(classifier\_naive, newdata = testing\_set)

print(y\_pred)

#confusion matrix

cm <- table(testing\_set$Species, y\_pred)

print(cm)

print(confusionMatrix(cm))

#K-Nearest

print("KNN-Hold out-part a")

# Fitting KNN model to traning dataset

classifier\_knn <- knn(train = train\_scale, test = test\_scale,

cl = training\_set$Species,

k=1)

print(classifier\_knn)

#confusion matrix

cm <- table(testing\_set$Species, classifier\_knn)

print(cm)

misClassError <- mean(classifier\_knn != testing\_set$Species)

print(paste("Accuracy of KNN model: ", 1 - misClassError))

#DECISION TREE

print("Decision Tree-Hold out-part a")

dtm<-rpart(Species~.,training\_set,method = "class")

#dtm #gives textual description

#plot(dtm)

#text(dtm) #adding text to plot

rpart.plot(dtm)

rpart.plot(dtm,type = 4, extra = 101)

p<-predict(dtm,testing\_set,type = "class")

#print(confusionMatrix(testing\_set[,5],p))

print(confusionMatrix(testing\_set[,5],p)$table)

print(confusionMatrix(testing\_set[,5],p)$overall["Accuracy"]\*100)

#Random subsampling

print("Random Subsampling")

dtm\_acc=list()

knn\_acc=list()

naive\_acc=list()

for(x in 1:5)

{

set.seed(123)

split<-sample.split(iris$Species,SplitRatio = 0.75)

train\_set\_rndm = subset(iris,split==TRUE)

test\_set\_rndm = subset(iris,split==FALSE)

dim(test\_set\_rndm)

dim(train\_set\_rndm)

train\_scale\_rndm <- scale(train\_set\_rndm[,1:4])

test\_scale\_rndm <- scale(test\_set\_rndm[,1:4])

dtm\_rndm = rpart(Species~., train\_set\_rndm,method = "class")

dtm\_rndm

#plot(dtm\_rndm)

#rpart.plot(dtm\_rndm)

#rpart.plot(dtm\_rndm,type = 4 , extra = 101)

p=predict(dtm\_rndm , test\_set\_rndm,type="class")

confusionMatrix(test\_set\_rndm[,5],p)$table

dtm\_acc[x] = (confusionMatrix(test\_set\_rndm[,5],p)$overall['Accuracy']\*100)

print(paste("Decision tree Accuracy: ",dtm\_acc[x]))

# K-NEAREST USING RANDOM SUBSAMPLING (PART A)

classifier\_knn\_rndm <- knn(train = train\_scale\_rndm[,c(1,2,3,4)], test = test\_scale\_rndm[,c(1,2,3,4)],cl = train\_set\_rndm$Species,k=1)

classifier\_knn

#confusion matrix

cm = table(test\_set\_rndm$Species, classifier\_knn\_rndm)

cm

misclasserror = mean(classifier\_knn\_rndm!=test\_set\_rndm$Species)

print(paste("Accuracy of knn model is :: ", 1 - misclasserror))

knn\_acc[x] = confusionMatrix(test\_set\_rndm[,5],classifier\_knn\_rndm)$overall['Accuracy']\*100

knn\_acc

# NAIVE BAYES USING RANDOM SUBSAMPLING (PART A)

classifier\_naive\_rndm = naiveBayes(Species~., data= train\_set\_rndm)

classifier\_naive

predicted\_y = predict(classifier\_naive\_rndm,newdata = test\_set\_rndm)

predicted\_y

table(predicted\_y)

#confusion matrix

cm = table(test\_set\_rndm$Species, predicted\_y)

print("Naive Bayes confusion matrix")

print(cm)

confusionMatrix(cm)

naive\_acc[x]=confusionMatrix(test\_set\_rndm$Species,predicted\_y)$overall['Accuracy']\*100

naive\_acc

}

#cross-validation

print("Cross Validation")

#Naive Bayes

print("Naive bayes-Cross Validation-part a")

nb\_model<-train(iris[,1:4],iris[,5],'nb',

trControl = trainControl(method = 'cv', number=4))

print(nb\_model)

#KNN

print("KNN-Cross Validation-part a")

knn\_model<-train(iris[,1:4],iris[,5],'knn',

trControl = trainControl(method = 'cv', number=4))

print(knn\_model)

#Decision Tree

print("Decision Tree-Cross Validation-part a")

Dtree\_model<-train(iris[,1:4],iris[,5],'rpart',

trControl = trainControl(method = 'cv', number=4))

print(Dtree\_model)

#part b

print("Part b start Training set = 66.6% (2/3rd of total), Test set = 33.3% ")

#Hold-out Method

print("Hold out Method")

set.seed(143)

split<-sample.split(iris$Species,SplitRatio = 0.6666)

training\_set = subset(iris,split==TRUE)

testing\_set = subset(iris,split==FALSE)

#feature scaling

train\_scale <- scale(training\_set[,1:4])

test\_scale <- scale(testing\_set[,1:4])

print(dim(training\_set))

print(dim(testing\_set))

#Naive bayes

print("Naive bayes-Hold out-part b")

classifier\_naive <- naiveBayes(Species~., data = training\_set)

print(classifier\_naive)

y\_pred <- predict(classifier\_naive, newdata = testing\_set)

print(y\_pred)

#confusion matrix

cm <- table(testing\_set$Species, y\_pred)

print(cm)

print(confusionMatrix(cm))

#K-Nearest

print("KNN-Hold out-part b")

# Fitting KNN model to traning dataset

classifier\_knn <- knn(train = train\_scale, test = test\_scale,

cl = training\_set$Species,

k=1)

print(classifier\_knn)

#confusion matrix

cm <- table(testing\_set$Species, classifier\_knn)

print(cm)

misClassError <- mean(classifier\_knn != testing\_set$Species)

print(paste("Accuracy of KNN model: ", 1 - misClassError))

#DECISION TREE

print("Decision Tree-Hold out-part b")

dtm<-rpart(Species~.,training\_set,method = "class")

#rpart.plot(dtm)

rpart.plot(dtm,type = 4, extra = 101)

p<-predict(dtm,testing\_set,type = "class")

print(confusionMatrix(testing\_set[,5],p))

#Random subsampling

print("Random Subsampling")

dtm\_acc=list()

knn\_acc=list()

naive\_acc=list()

for(x in 1:5)

{

set.seed(123)

split<-sample.split(iris$Species,SplitRatio = 0.66)

train\_set\_rndm = subset(iris,split==TRUE)

test\_set\_rndm = subset(iris,split==FALSE)

dim(test\_set\_rndm)

dim(train\_set\_rndm)

train\_scale\_rndm <- scale(train\_set\_rndm[,1:4])

test\_scale\_rndm <- scale(test\_set\_rndm[,1:4])

dtm\_rndm = rpart(Species~., train\_set\_rndm,method = "class")

dtm\_rndm

#plot(dtm\_rndm)

#rpart.plot(dtm\_rndm)

#rpart.plot(dtm\_rndm,type = 4 , extra = 101)

p=predict(dtm\_rndm , test\_set\_rndm,type="class")

confusionMatrix(test\_set\_rndm[,5],p)$table

dtm\_acc[x] = (confusionMatrix(test\_set\_rndm[,5],p)$overall['Accuracy']\*100)

print(paste("Decision tree Accuracy: ",dtm\_acc[x]))

# K-NEAREST USING RANDOM SUBSAMPLING (PART A)

classifier\_knn\_rndm <- knn(train = train\_scale\_rndm[,c(1,2,3,4)], test = test\_scale\_rndm[,c(1,2,3,4)],cl = train\_set\_rndm$Species,k=1)

classifier\_knn

#confusion matrix

cm = table(test\_set\_rndm$Species, classifier\_knn\_rndm)

cm

misclasserror = mean(classifier\_knn\_rndm!=test\_set\_rndm$Species)

print(paste("Accuracy of knn model is :: ", 1 - misclasserror))

knn\_acc[x] = confusionMatrix(test\_set\_rndm[,5],classifier\_knn\_rndm)$overall['Accuracy']\*100

knn\_acc

# NAIVE BAYES USING RANDOM SUBSAMPLING (PART A)

classifier\_naive\_rndm = naiveBayes(Species~., data= train\_set\_rndm)

classifier\_naive

predicted\_y = predict(classifier\_naive\_rndm,newdata = test\_set\_rndm)

predicted\_y

table(predicted\_y)

#confusion matrix

cm = table(test\_set\_rndm$Species, predicted\_y)

print("Naive Bayes confusion matrix")

print(cm)

confusionMatrix(cm)

naive\_acc[x]=confusionMatrix(test\_set\_rndm$Species,predicted\_y)$overall['Accuracy']\*100

naive\_acc

}

#cross-validation

print("Cross Validation")

#Naive Bayes

print("Naive bayes-Cross Validation-part b")

nb\_model<-train(iris[,1:4],iris[,5],'nb',

trControl = trainControl(method = 'cv', number=3))

print(nb\_model)

#KNN

print("KNN-Cross Validation-part b")

knn\_model<-train(iris[,1:4],iris[,5],'knn',

trControl = trainControl(method = 'cv', number=3))

print(knn\_model)

#DECISION TREE

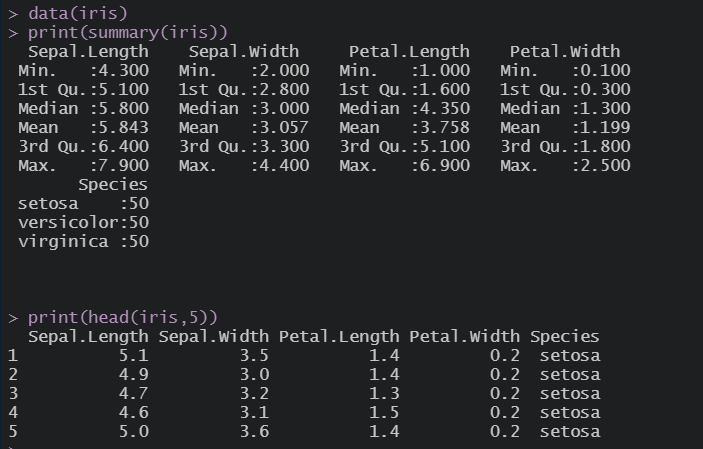
print("Decision Tree-Cross Validation-part b")

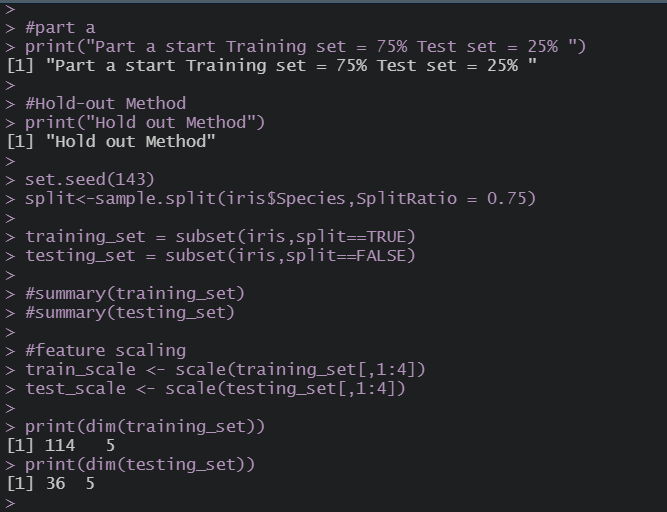
Dtree\_model<-train(iris[,1:4],iris[,5],'rpart',

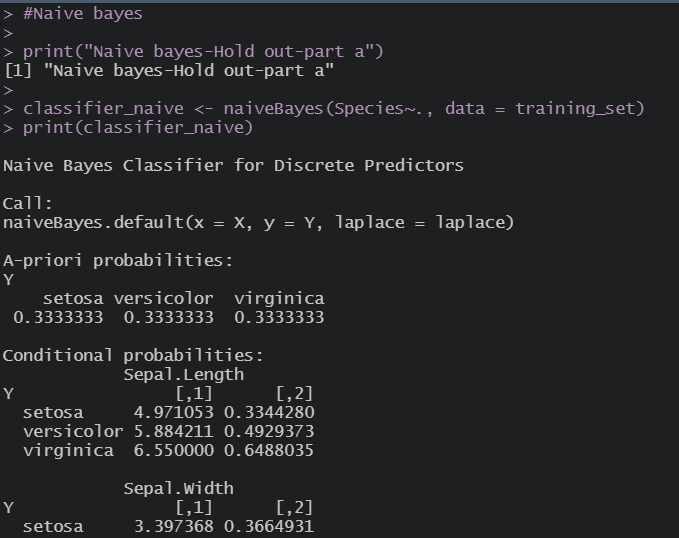
trControl = trainControl(method = 'cv', number=3))

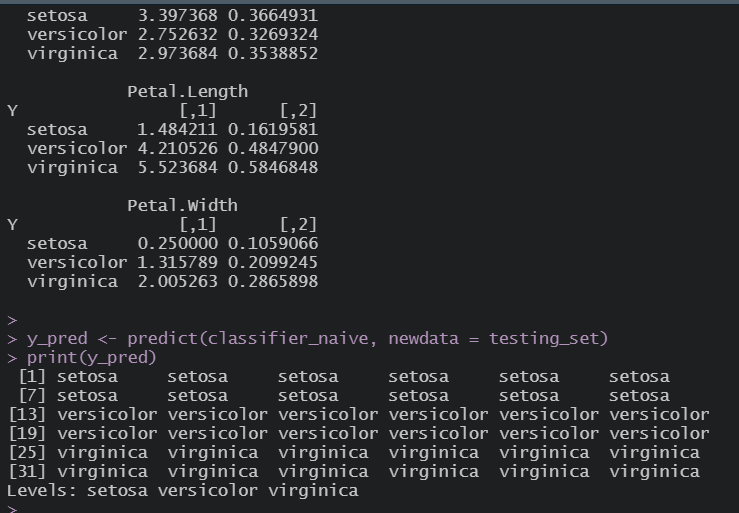
print(Dtree\_model)

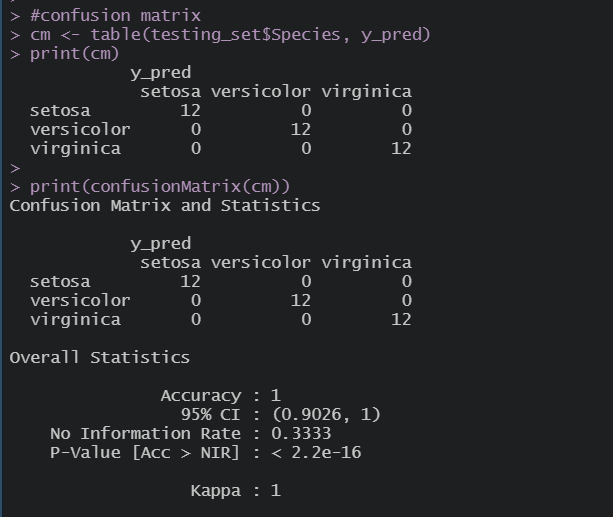
Output Screen:-

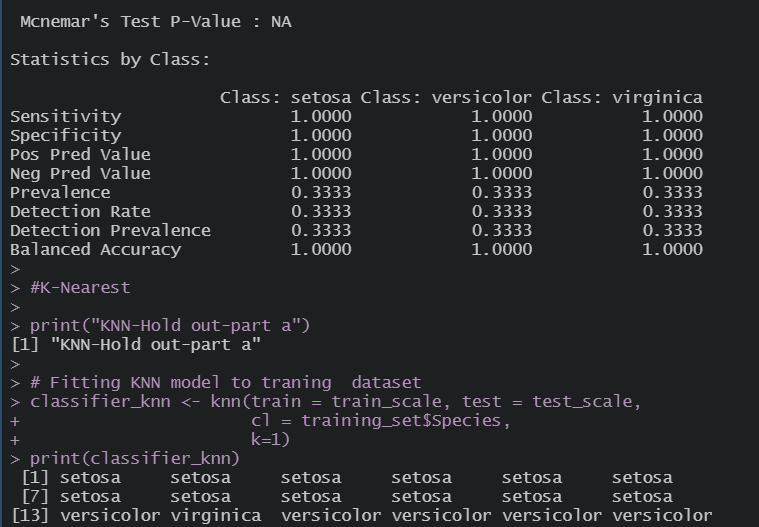


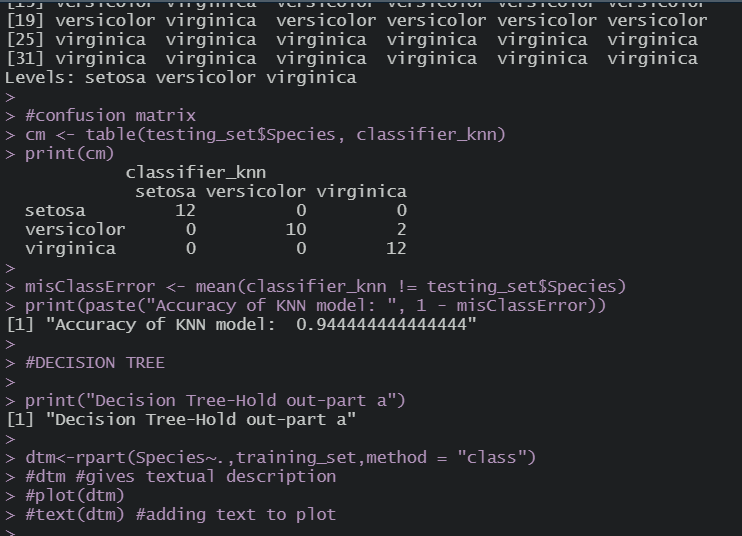


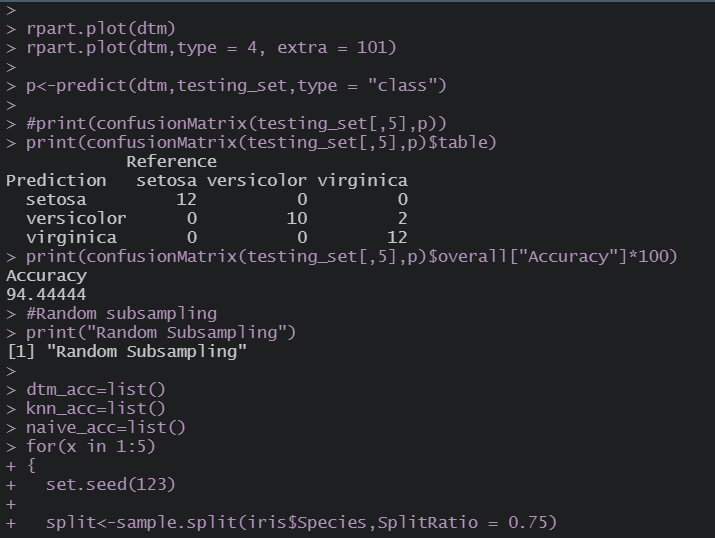


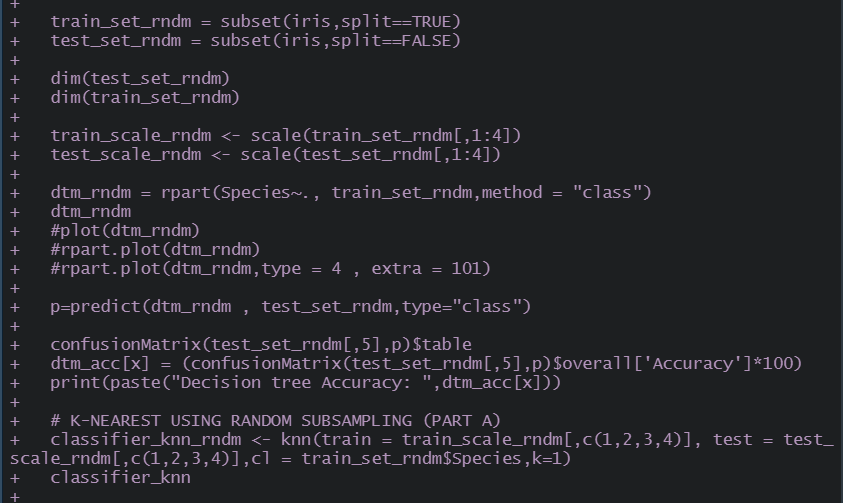


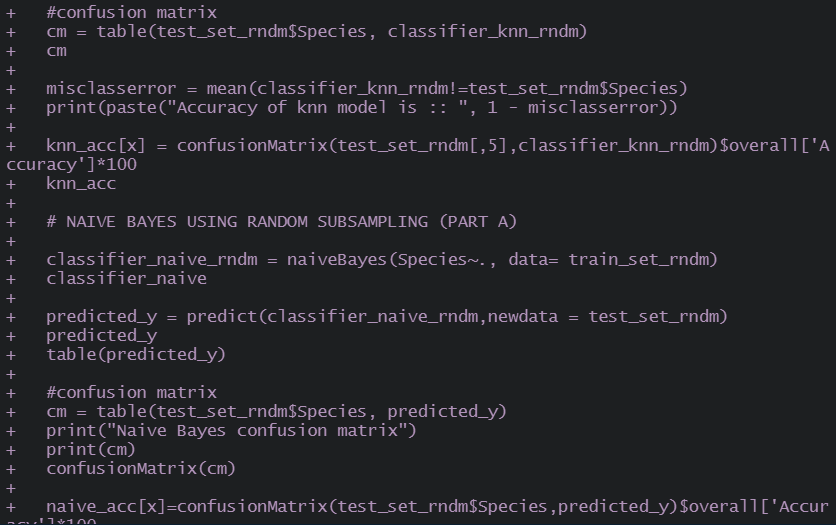


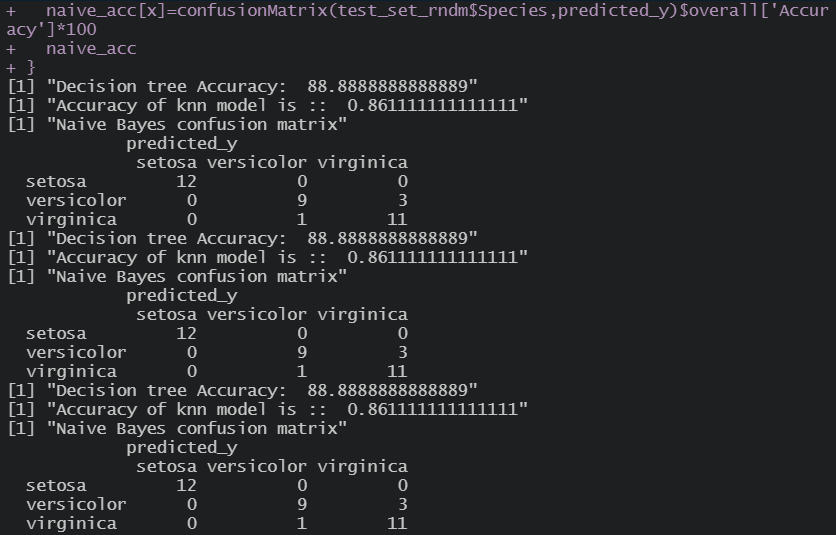


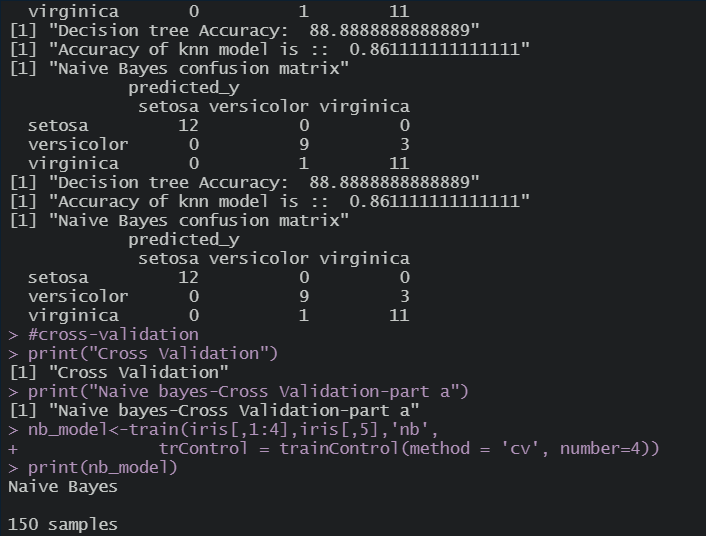


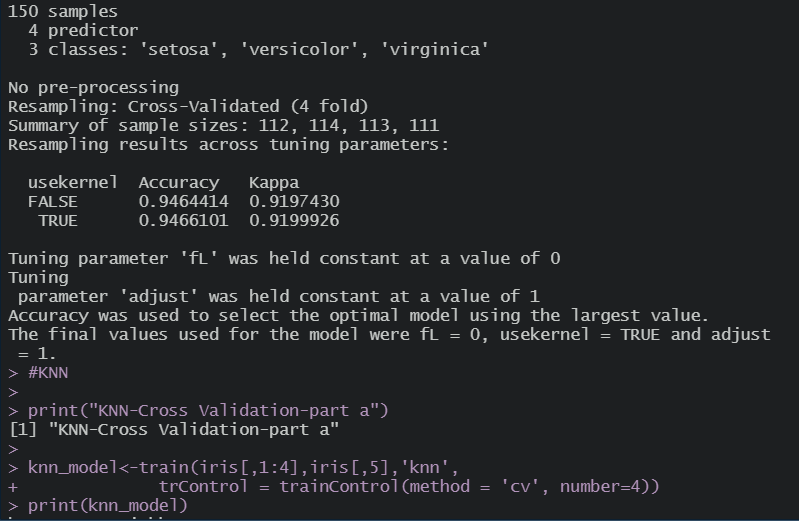


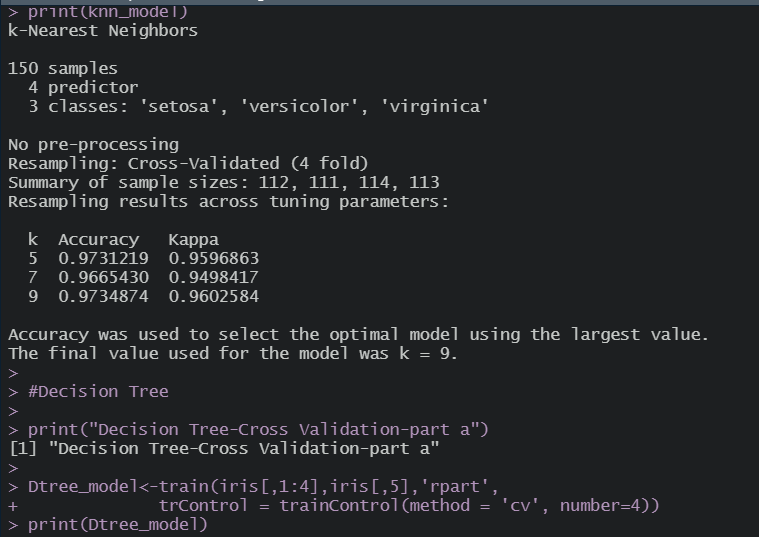


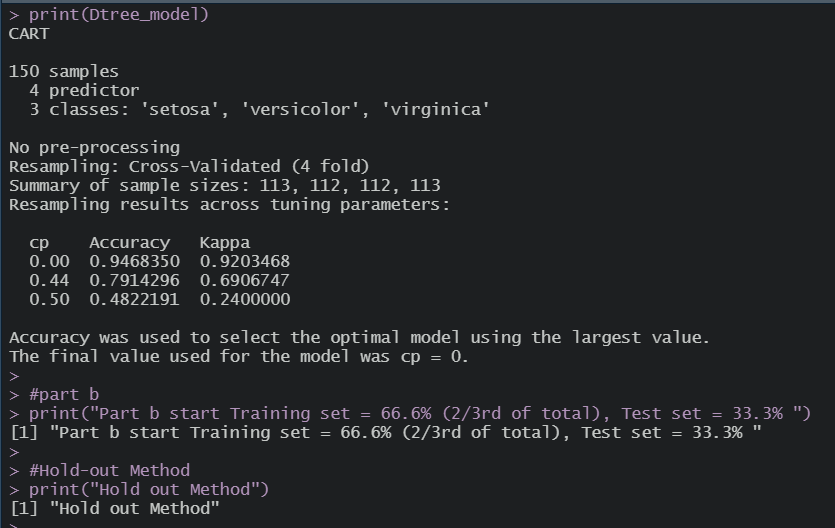


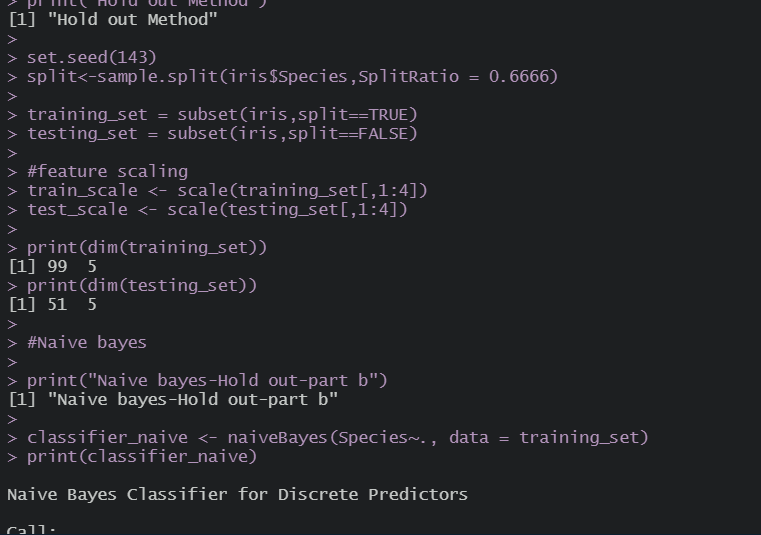


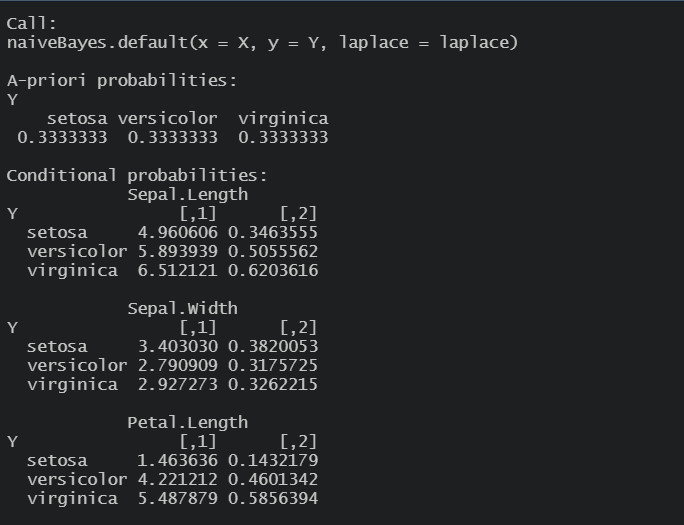


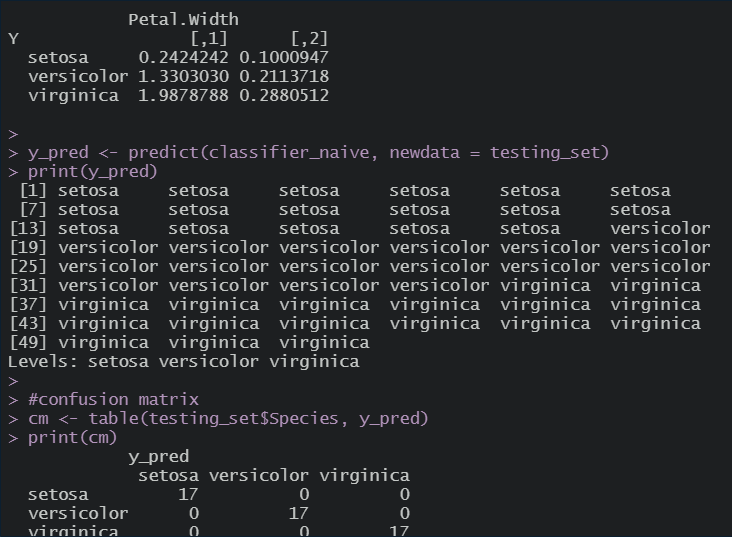


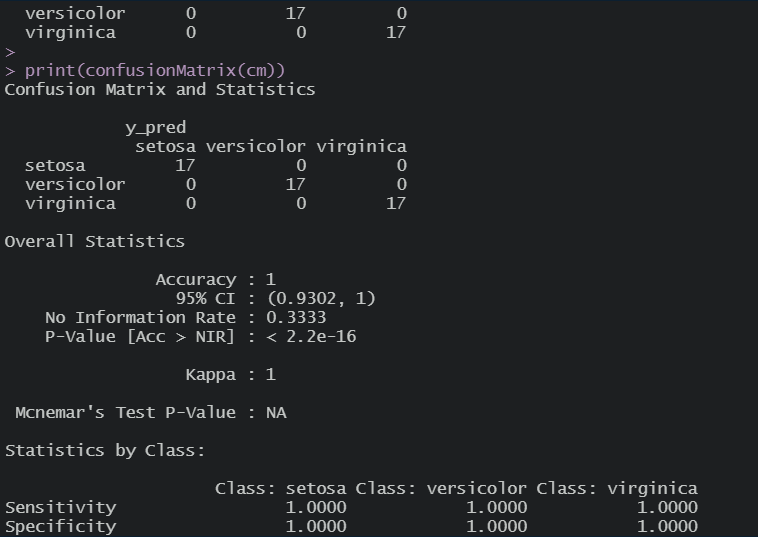


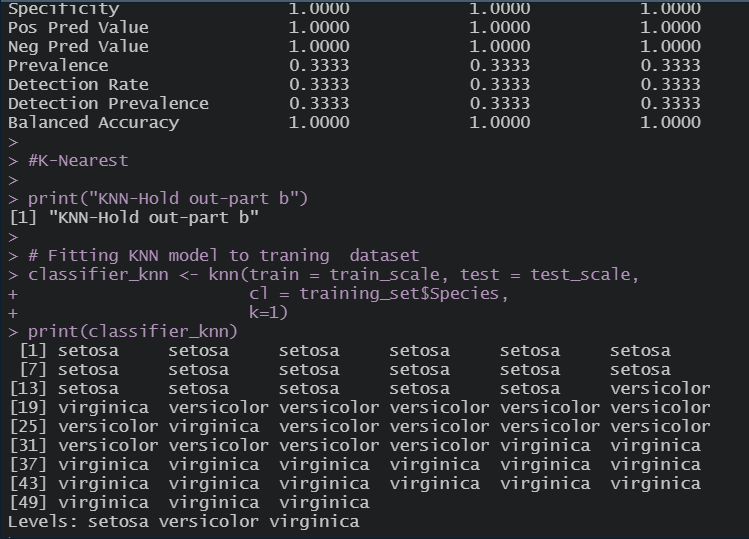


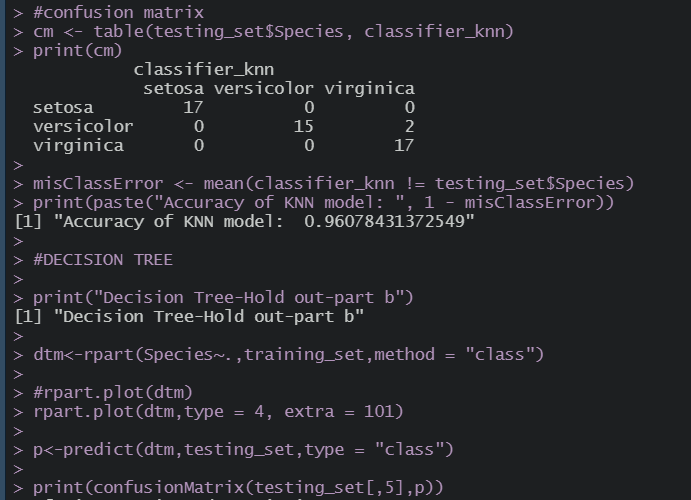


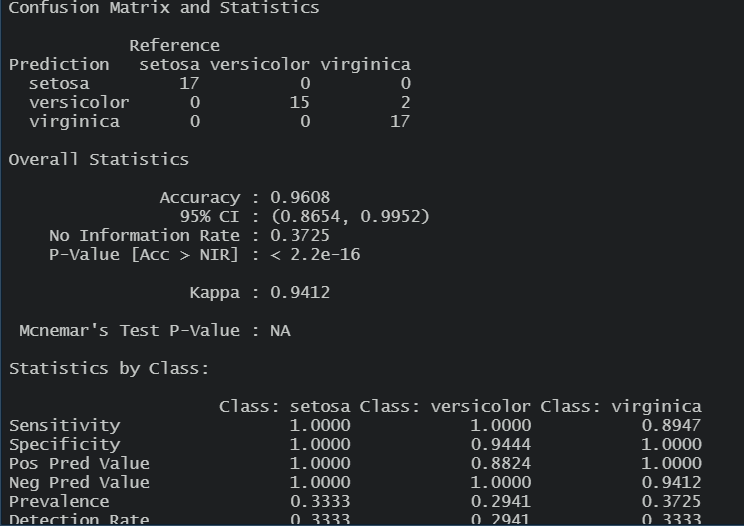




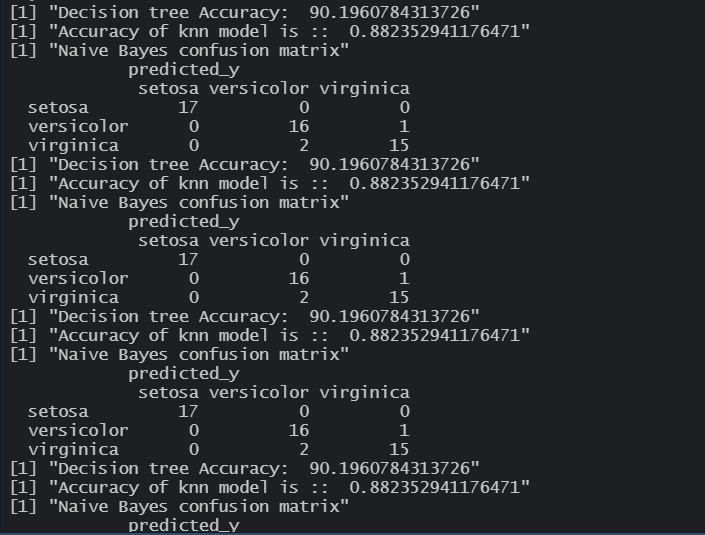


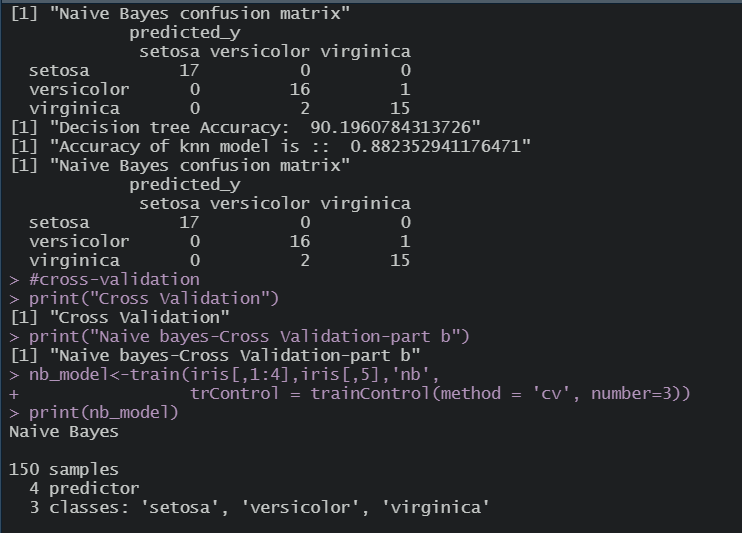


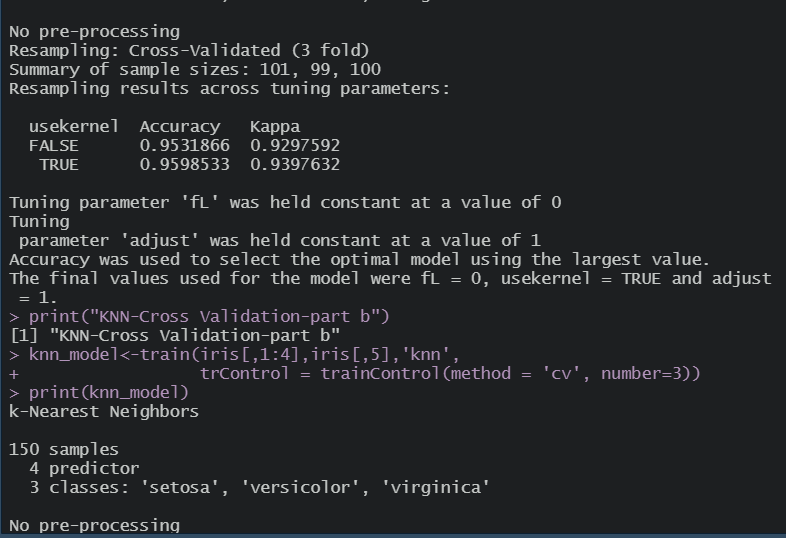


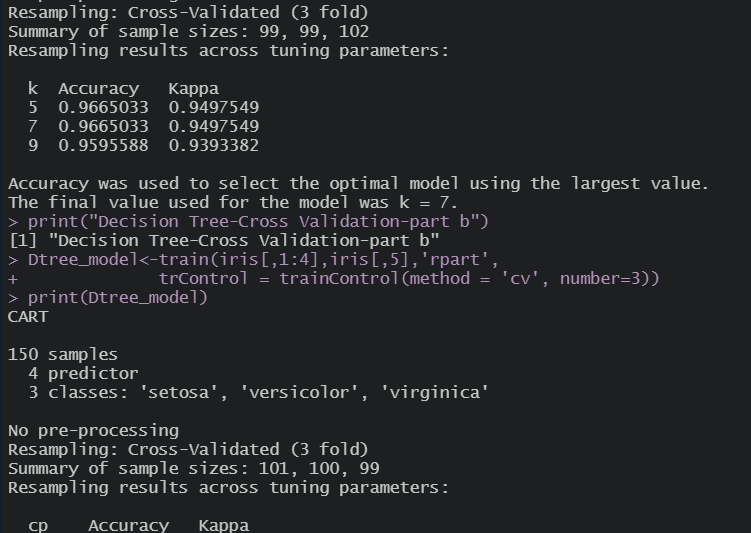


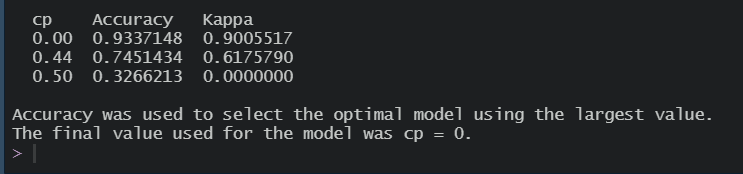




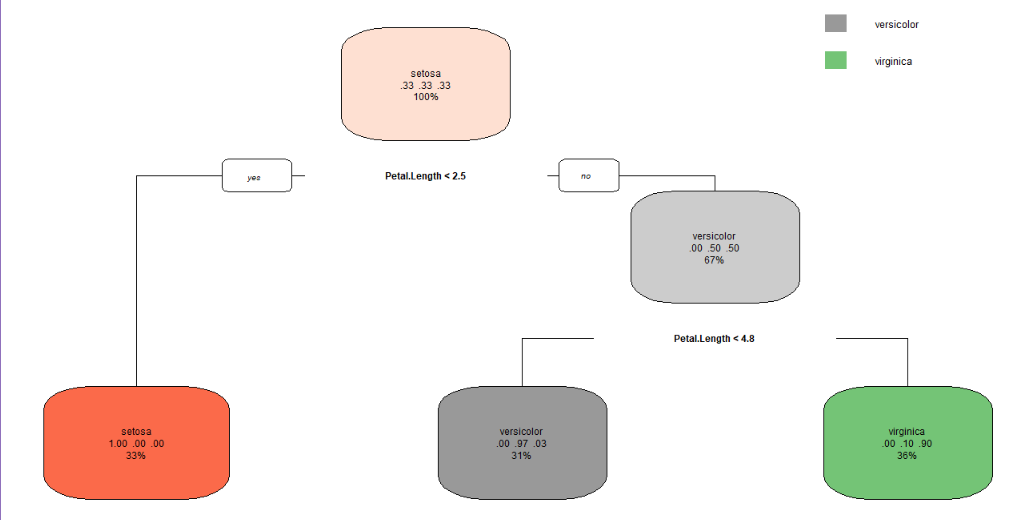


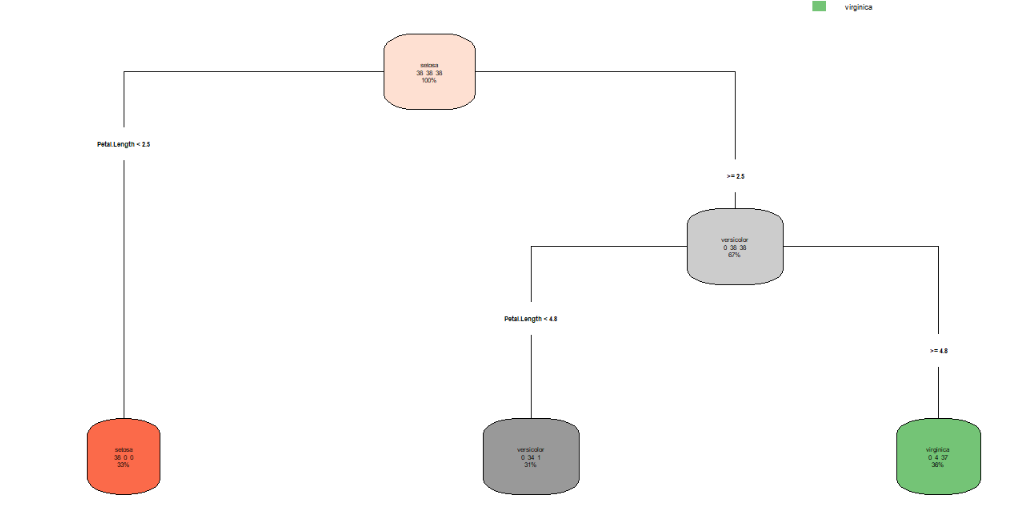


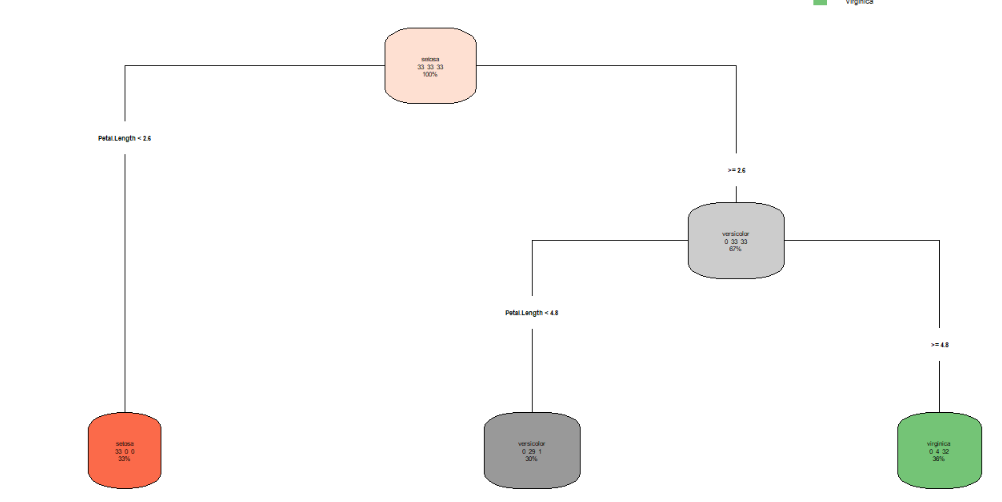




PLOTS:-







Ques 6:-

Use Simple Kmeans, DBScan, Hierachical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

Code:-

#Q6. Use Simple Kmeans, DBScan, Hierachical clustering algorithms

#for clustering. Compare the performance of clusters by changing

#the parameters involved in the algorithms.

#working on HTRU\_2 dataset

df <- read.csv("datasets/HTRU2/HTRU\_2.csv")

print(summary(df))

#View(df)

str(df)

library(ggplot2)

#normalize data for variables measured on different scale

dfScaled <- scale(df[,-9])

dfScaled

summary(dfScaled)

set.seed(234)

kmeans\_model <- kmeans(x = df[,-9], centers = 2, nstart = 20)

print(kmeans\_model)

print(kmeans\_model$cluster)

cm <- table(df[,9], kmeans\_model$cluster)

print(cm)

#library(caret)

#print(confusionMatrix(cm))

# ----------------- HIERACHICAL CLUSTERING --------------------

#distance matrix - matrix of distance b/w every point to any other

d= dist(dfScaled)

fitH = hclust(d,"ward.D2")

print(fitH)

plot(fitH)

rect.hclust(fitH,k=3,border = "blue")

cluster= cutree(fitH,3) #we can cut off the tree at the desired number of clusters using cutree

cluster

table(df[,9],cluster)

# ---------------------- DBSCAN ---------------------------

#install.packages("dbscan",dependencies = TRUE)

library(dbscan)

kNNdistplot(dfScaled,k=3) #to decide value of eps

abline(h=0.7,col="red",Ity = 2)

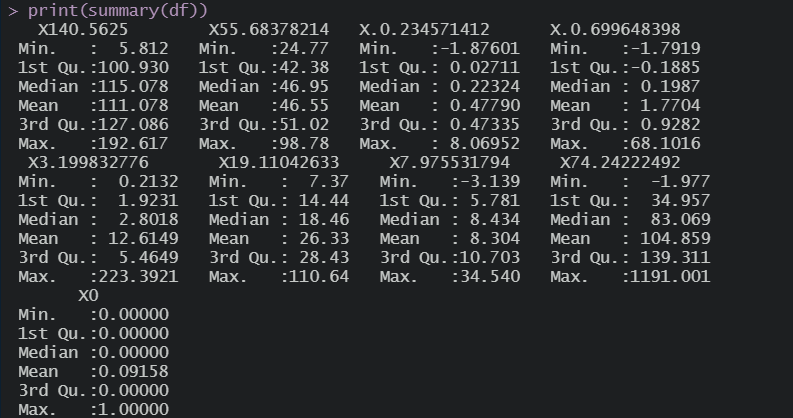
fitD<-dbscan(dfScaled,eps=0.7,minPts = 5)

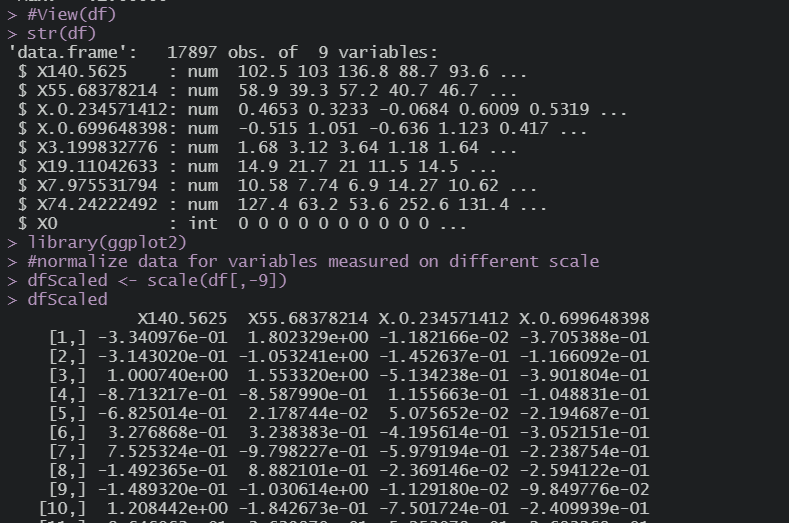
print(fitD)

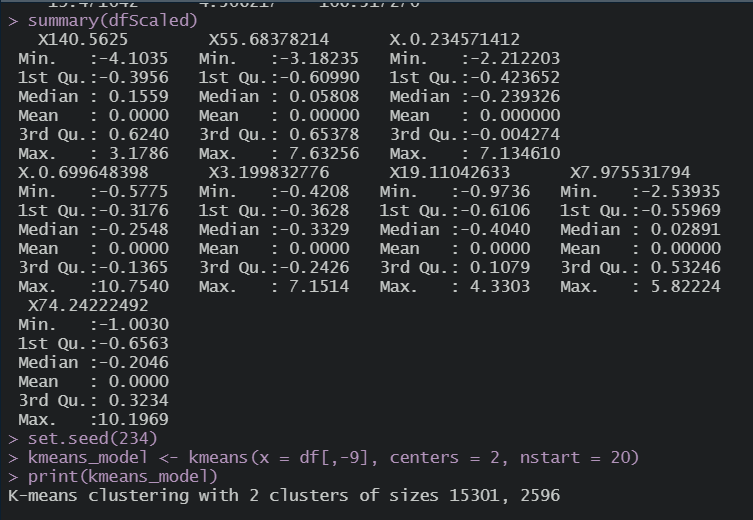
plot(df,col=fitD$cluster)

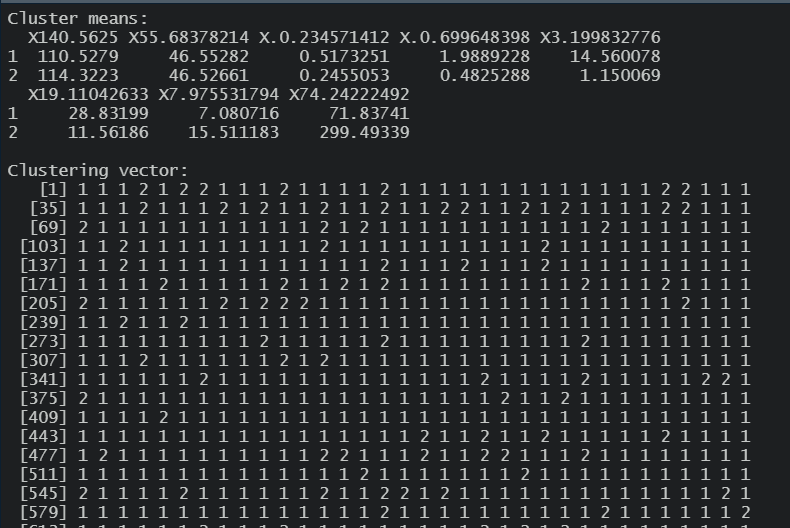
table(df[,9],fitD$cluster)

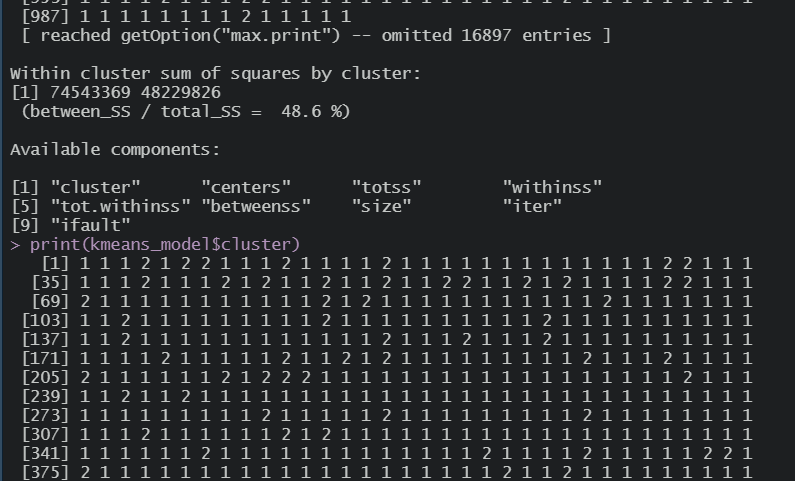
Output Screen:-

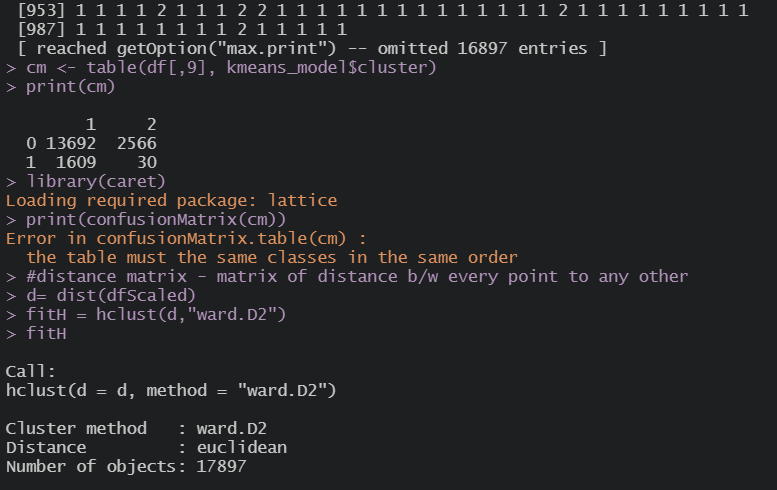


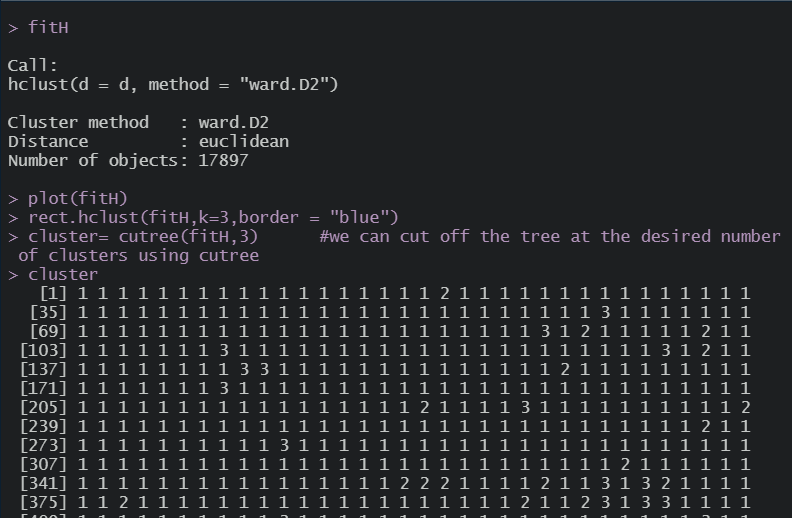


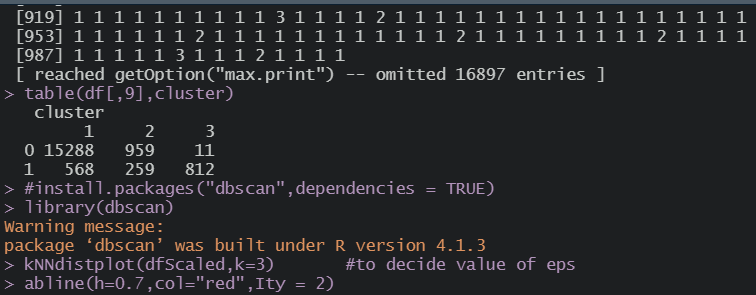


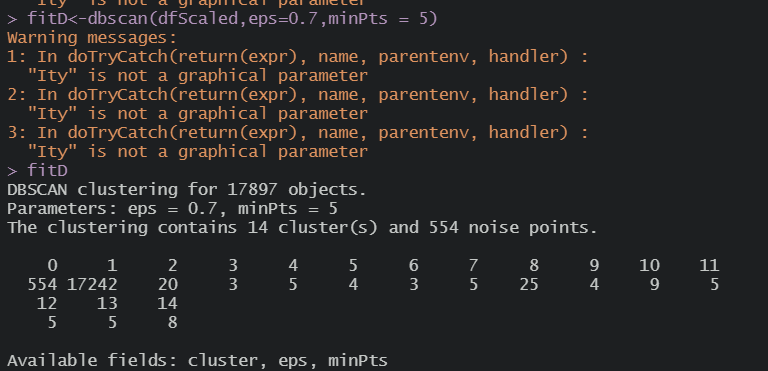


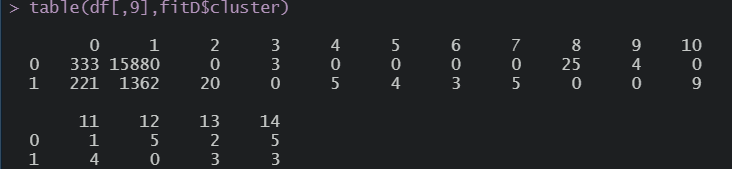












Plots:-

