**DATA MINING**

**PRACTICAL LIST**

**Name: Akanksha Goel**

**Examination Roll No.: 19035570011**

**College Roll No.: 0726**

**Course.: B.Sc. (Hons) Computer Science**

**Section.: A**

**Ques 1.** Create a file people.txt and then

1. Read the data from file "people.txt".
2. Create a ruleset E that contains the rules to check for the following conditions:
   1. The age should be in range 0-180
   2. The age should be in the range 0-150.2.
   3. The age should be greater than yearsmarried.
   4. The status should be married or single or widowed.
   5. The 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.
3. Check whether ruleset E is violated by the data in the file people.txt.
4. Summarize the results obtained in part (iii)
5. Visualize the results obtained in part (iii)

**. R FILE**

#install.packages("editrules")

library(editrules)

people=read.table(file="C://Users/akanksha goel/OneDrive/Documents/program/people.txt",header=TRUE)

people

str(people)

summary(people)

E<-editset(expression(

                      AGE >= 0,

                      AGE < 150,

                      AGE > YEARSMARRIED,

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

                      if(AGE < 18) AGEGROUP %in% c('child'),

                      if(AGE >= 18 && AGE <=65) AGEGROUP %in% c('adult'),

                      if(AGE >65) AGEGROUP %in% c('elderly')

                      ))

E

voilated<-violatedEdits(E,people)

people

voilated

summary(voilated)

?plot

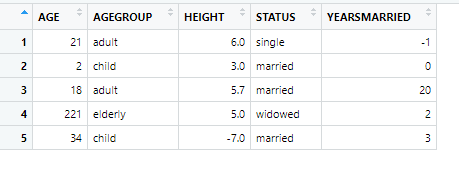
plot(voilated)

plot(E)

plot(E,layout=layout.circle,type="people")

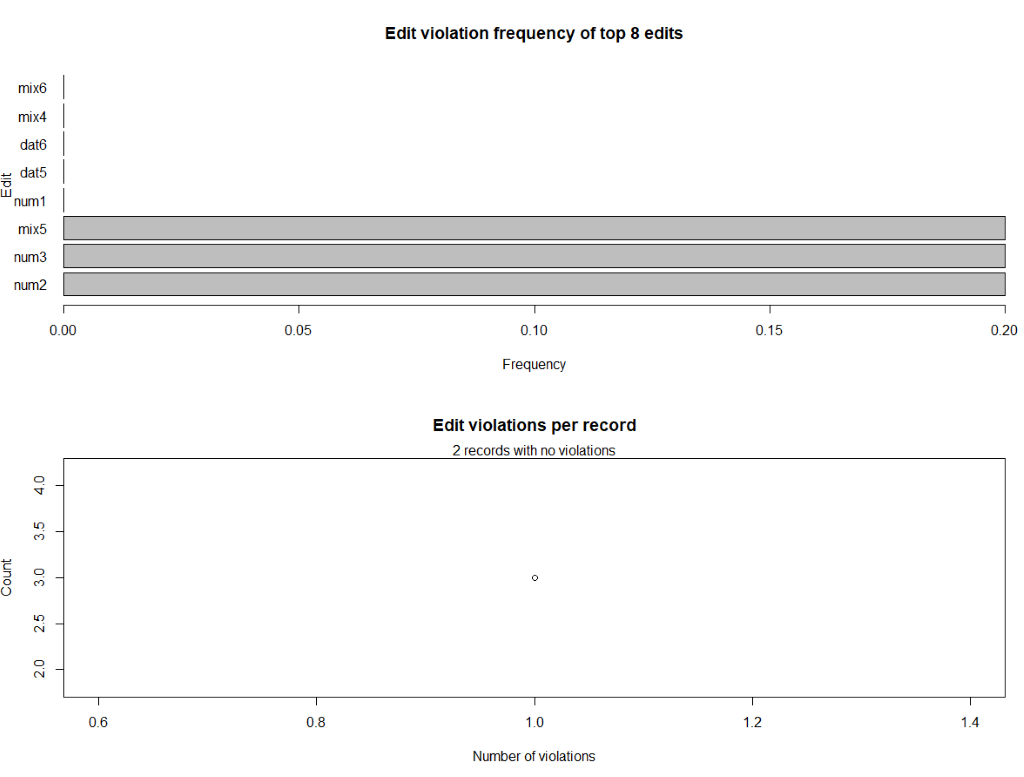
plot(E,type="l")

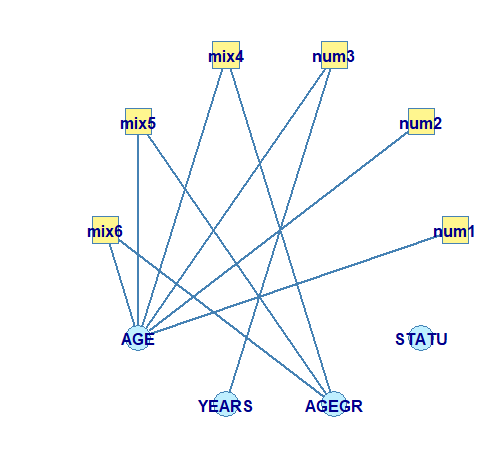
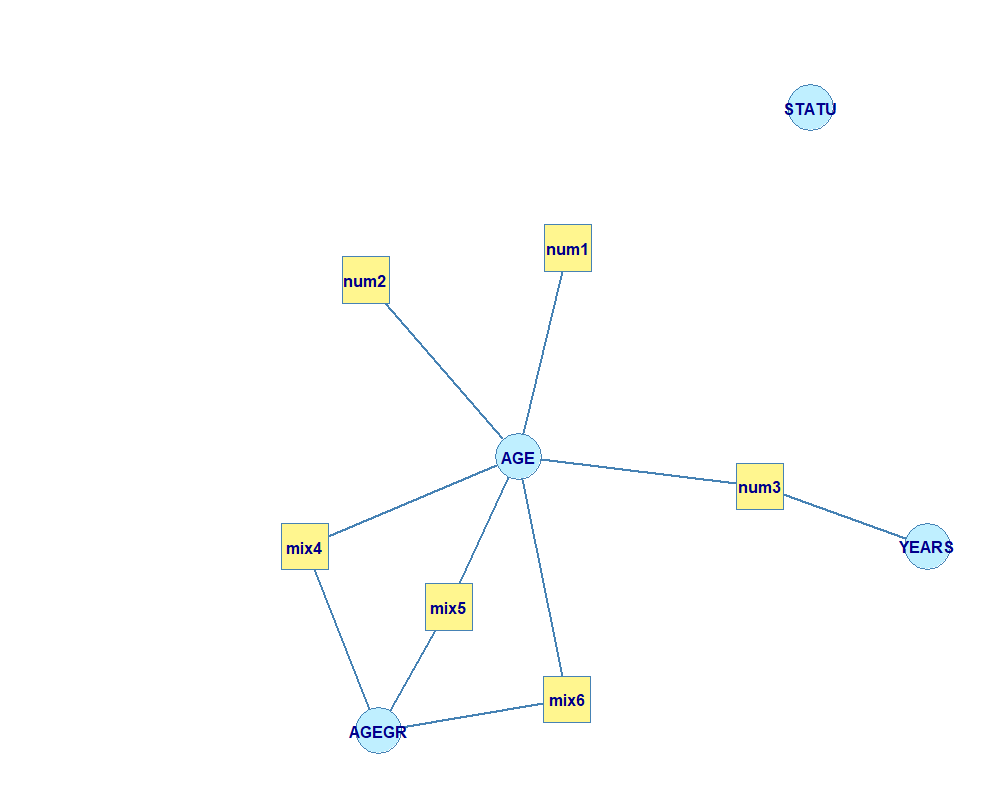
**.people.txt**



**PLOTS**

#plot(voilated)





**Ques 2**. Perform the following preprocessing tasks on the dirty\_iris dataseta

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).

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 theresult.

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

**.R FILE**

data\_iris<-read.csv(file="C://Users/akanksha goel/OneDrive/Documents/program/dirty\_iris.csv",header = TRUE)

data\_iris

summary(data\_iris)

str(data\_iris)

#information of cases that are complete i.e rows that have value for each column

complete\_cases<-complete.cases(data\_iris)

complete\_cases

as.numeric(complete\_cases)

sum\_cc<-sum(complete\_cases)#96

percent\_cc<-100\*sum\_cc/length(complete\_cases)

cat("COMPLETE CASES in dirty iris is ",percent\_cc,"%")

#Special value(infinity) is replaced with na

any(is.na(data\_iris))

which(is.na(data\_iris$Sepal.Width))

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

data\_iris

library(editrules)

(Rules<-editfile(file="C://Users/akanksha goel/OneDrive/Documents/program/efques2.txt"))

Rules

voilated<-violatedEdits(Rules,data\_iris)

voilated

summary(voilated)

plot(voilated)

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

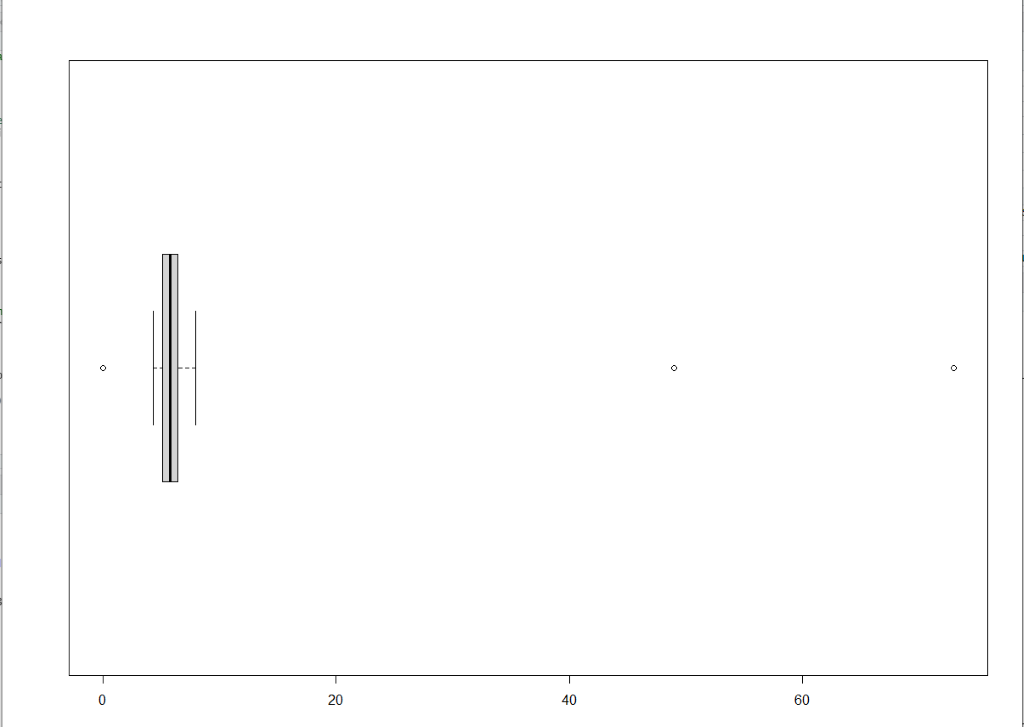
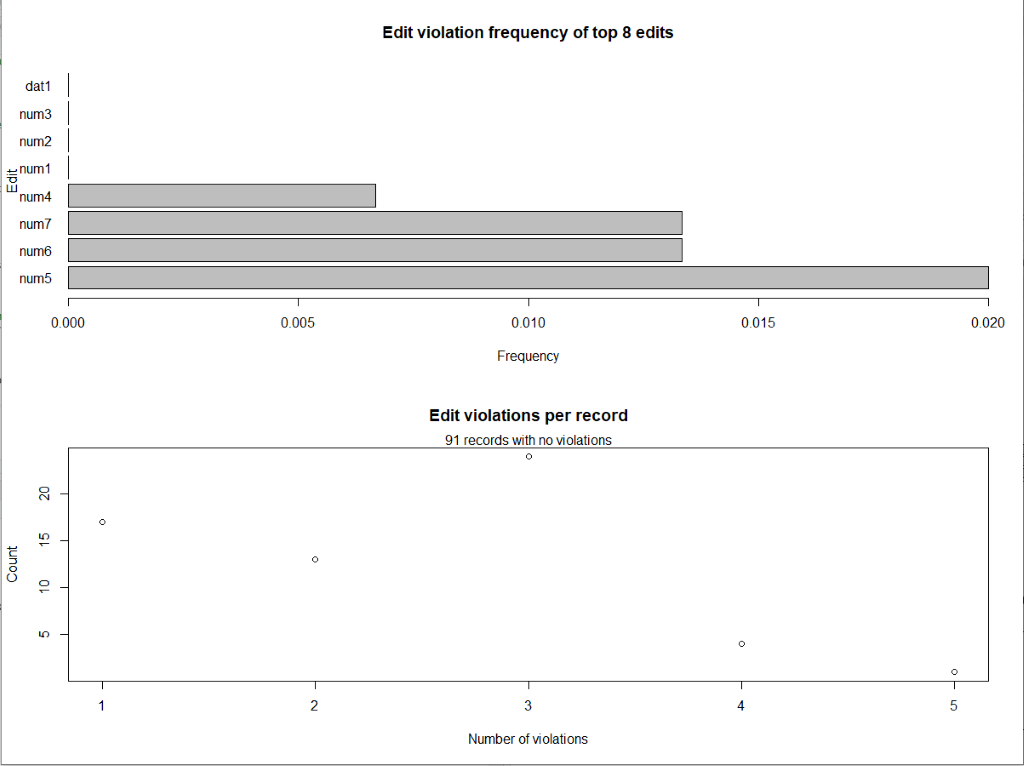
summary(data\_iris$Sepal.Length)

boxplot.stats(data\_iris$Sepal.Length)

**OUTPUT**  : **PLOTS**

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

#plot(voilated)



Q3. 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.

**.R FILE**

#IRIS DATASET

View(iris)

summary(iris)

sapply(iris[,1:4],sd)

#install.packages("caret")

library(caret)

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

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

summary(iris\_transformed\_final)

apply(iris\_transformed\_final,2,sd)

#WINE DATASET

data\_wine<-read.csv(file="C://Users/akanksha goel/OneDrive/Documents/program/wine.data",header = TRUE,sep=',')

data\_wine

View(data\_wine)

summary(data\_wine)

sapply(data\_wine,sd)

library(caret)

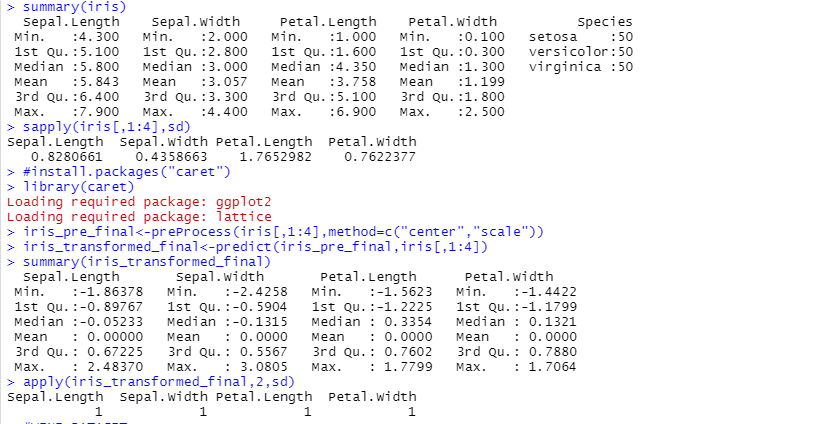
wine\_pre\_final<-preProcess(data\_wine[,],method=c("center","scale"))

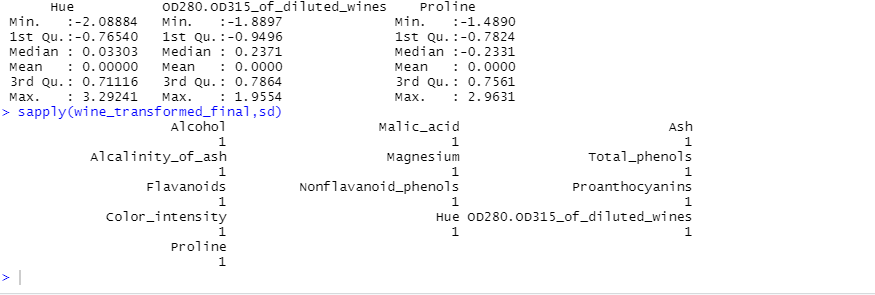
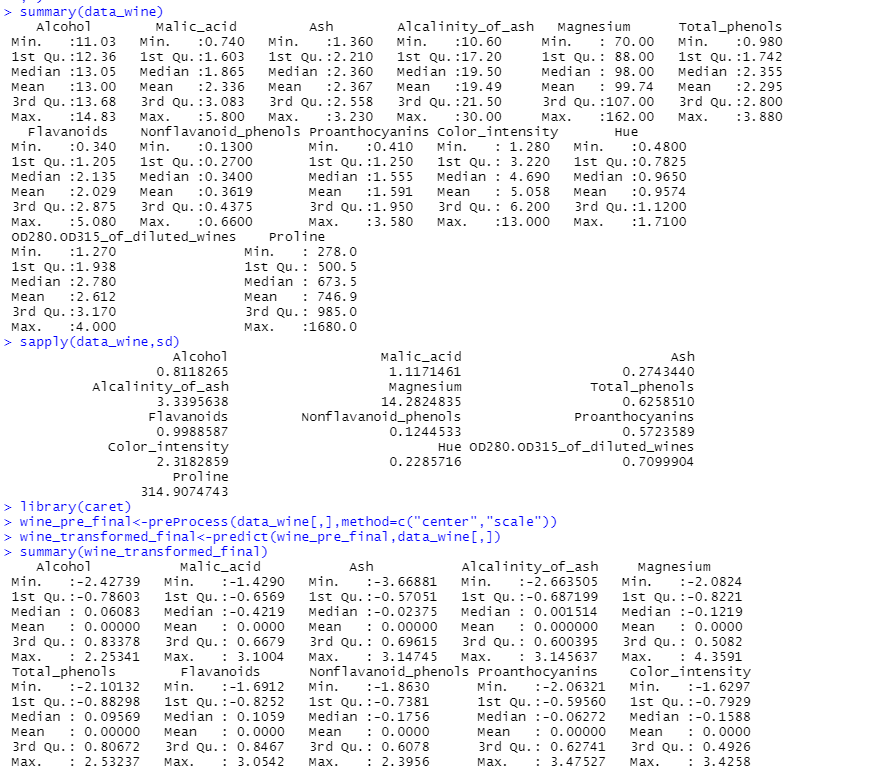
wine\_transformed\_final<-predict(wine\_pre\_final,data\_wine[,])

sapply(wine\_transformed\_final,sd)

summary(wine\_transformed\_final)

**OUTPUT**





Q3 Run Apriori algorithm to find frequent itemsets and association rules1.1Use minimum support as 50% and minimum confidence as 75%1.2Use minimum support as 60% and minimum confidence as 60 %

**.R FILE**

#install.packages('arules',dependencies = TRUE)#install.packages('arulesViz',dependencies = TRUE)

library(arules)

data("Groceries")

?Groceries

View(Groceries)

str(Groceries)

summary(Groceries)

head(Groceries)

itemFrequencyPlot(Groceries,topN=20,type="absolute")

rules<-apriori(Groceries,parameter = list(supp=0.001,conf=0.8))

inspect(rules)

inspect(head(rules))

inspect(rules[1:10])

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

rules<-apriori(Groceries,parameter = list(supp=0.001,conf=0.8))

inspect(rules[1:10])

library(arulesViz)

#There are many method, plotting engines and all of them have different control parameters. Use

# "help" to get help. List available methods for the object rules:

plot(rules, method = "graph")

# List the available engines for method "scatterplot"

plot(rules, method = "scatterplot", engine = "ggplot2")

plot(rules[1:5],method="graph",shading=NULL)

plot(rules[1:5],method="graph",engine='interactive',shading=NULL)

#b part

receipt\_df<-read.csv("C:/Users/akanksha goel/OneDrive/Documents/program/1000i.csv",header = F)

receipt\_df

#Applying Column Names

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

#Applying data preprocessing ,creating a dataframe each containing each item and its corresponding id

id<-c(1:5)

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

df<-data.frame(id,food)

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

#After preprocessing

head(receipt\_df)

typeof(receipt\_df)

#Converting list type to 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")

rules<-apriori(df\_trans,parameter = list(supp=0.002,conf=.1))

inspect(head(rules))

#PLOTS

plot(rules, method = "graph")

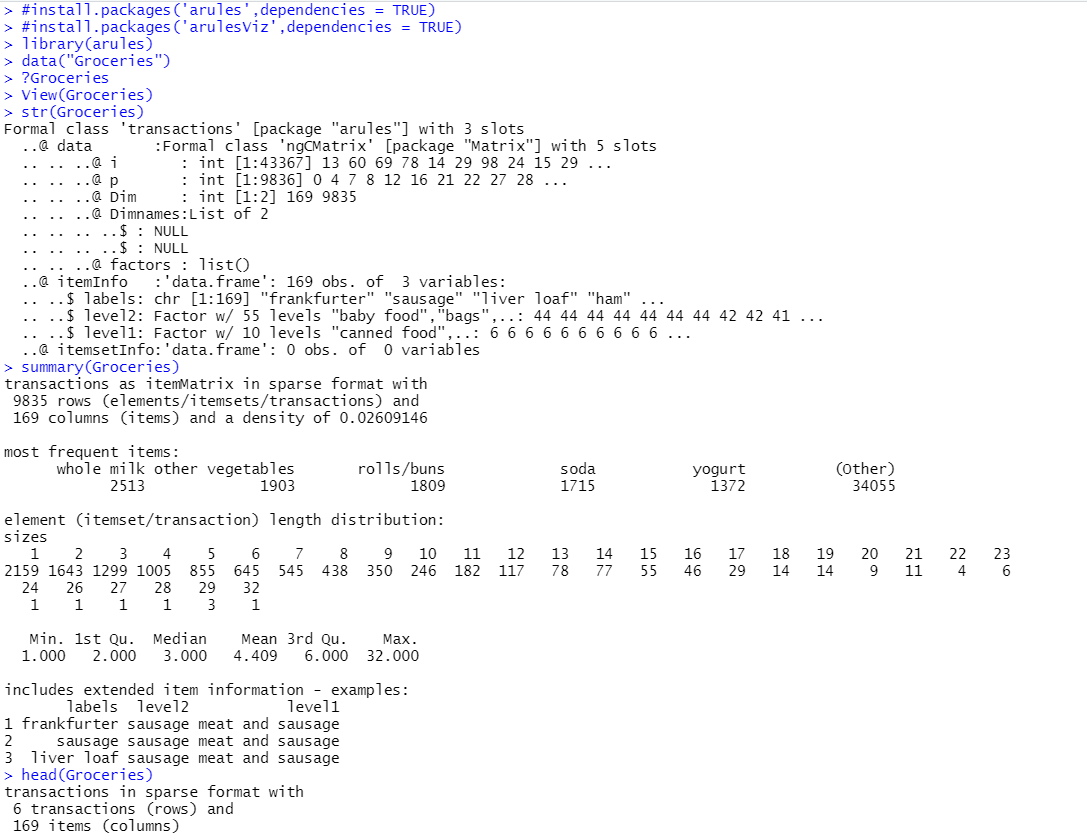
# List the available engines for method "scatterplot"

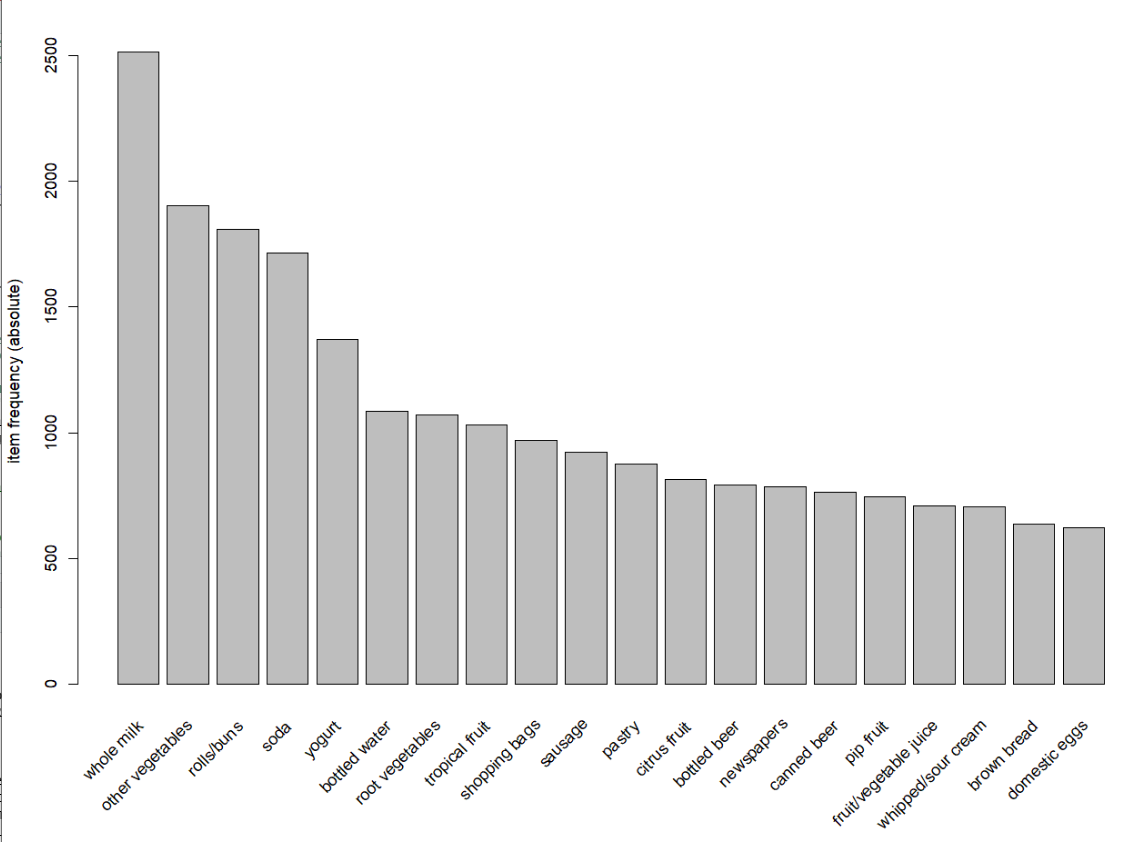
plot(rules, method = "scatterplot", engine = "ggplot2")

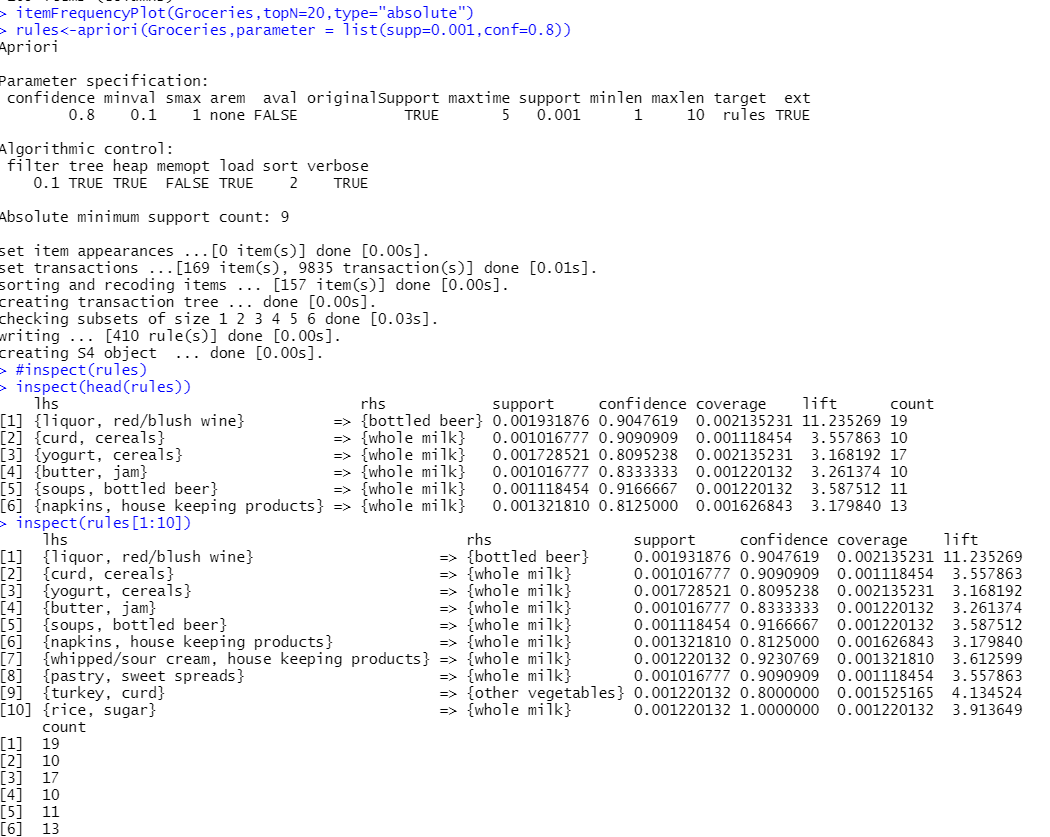
plot(rules[1:5],method="graph",shading=NULL)

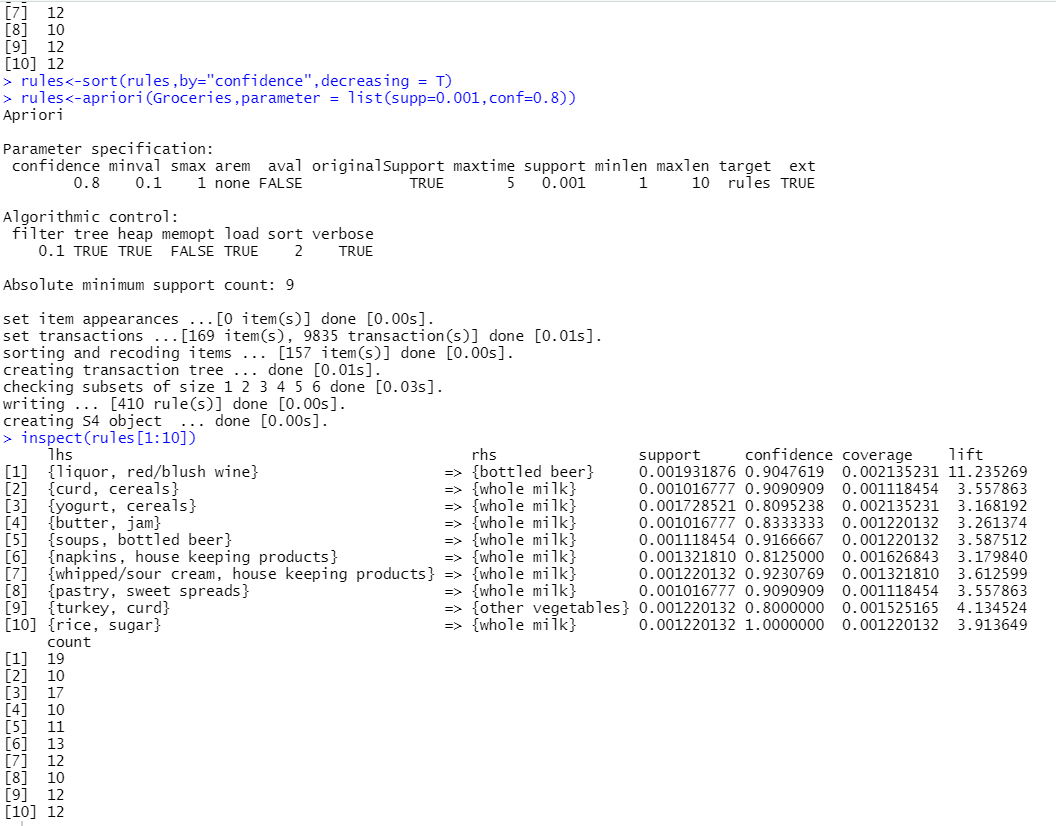
plot(rules[1:5],method="graph",engine='interactive',shading=NULL)

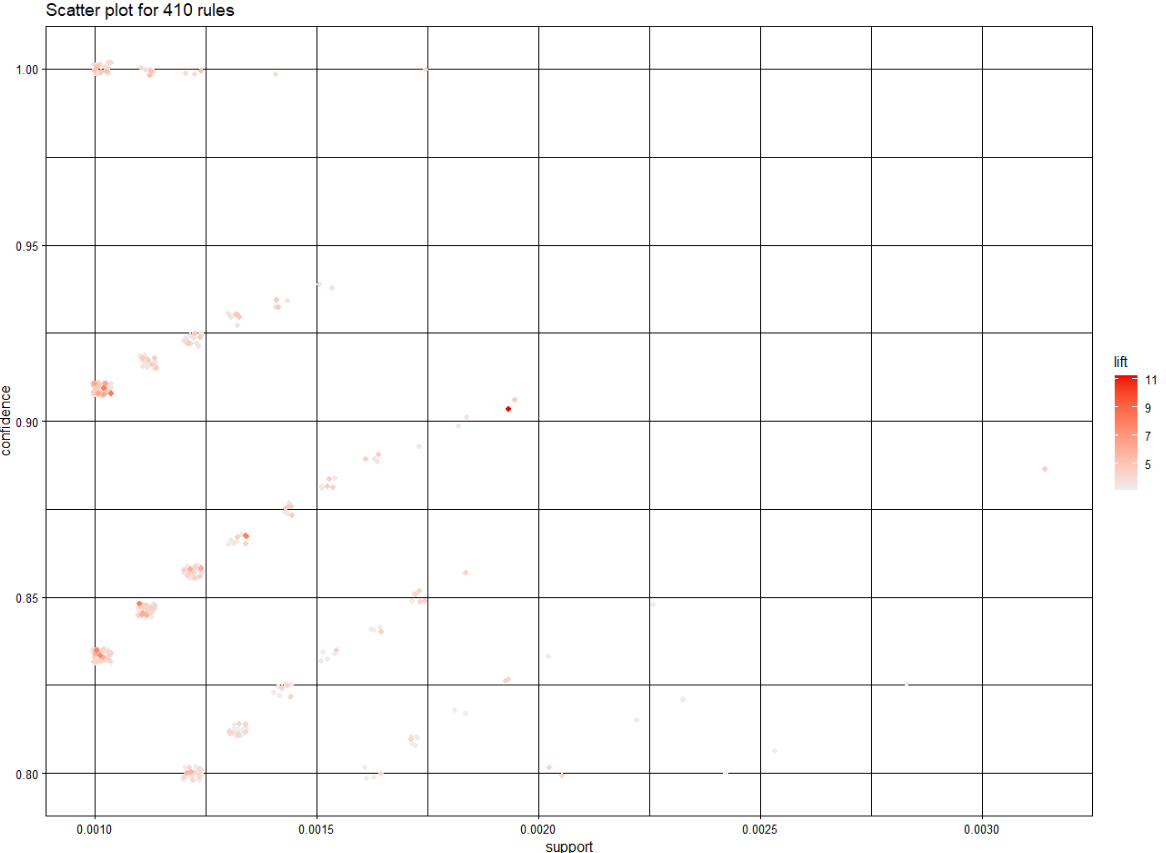
**OUTPUT**



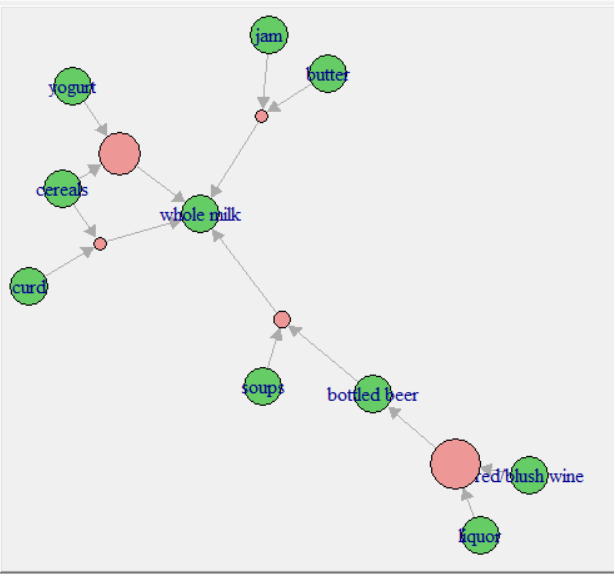


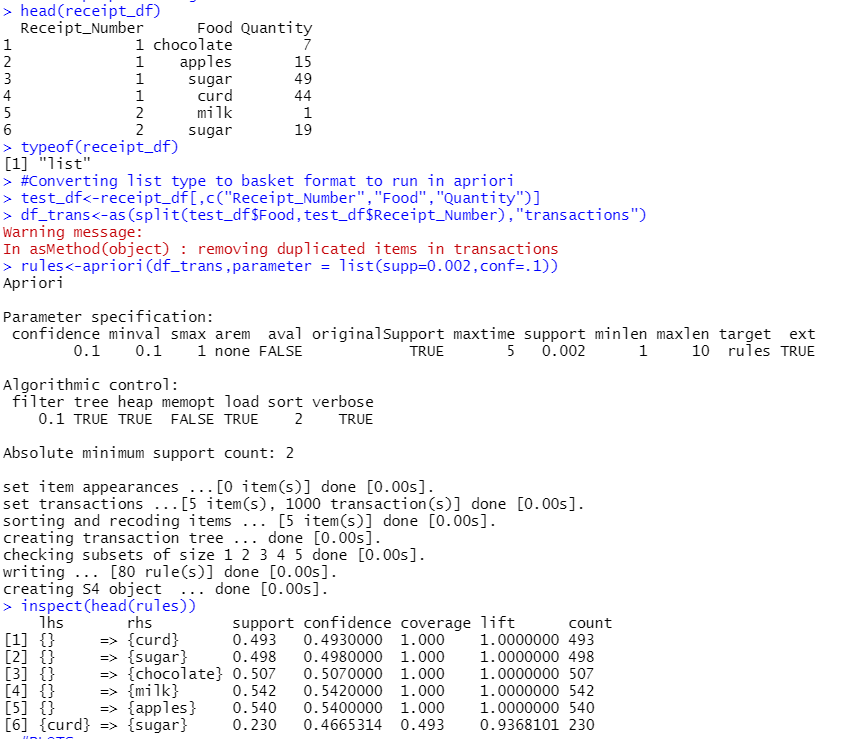


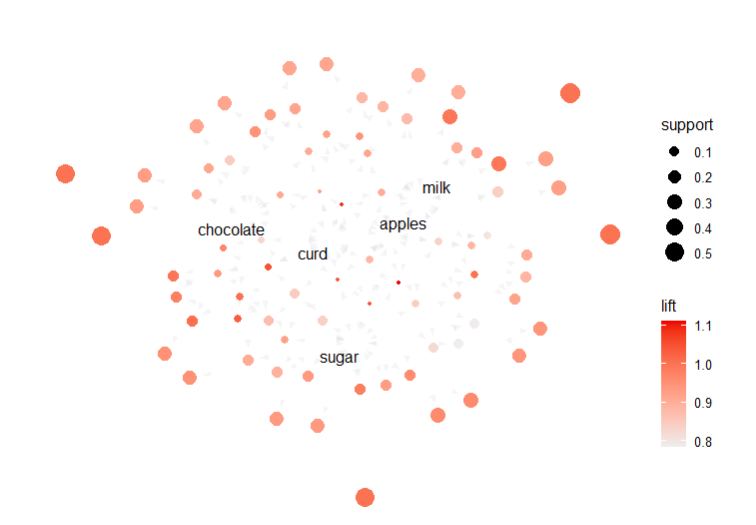


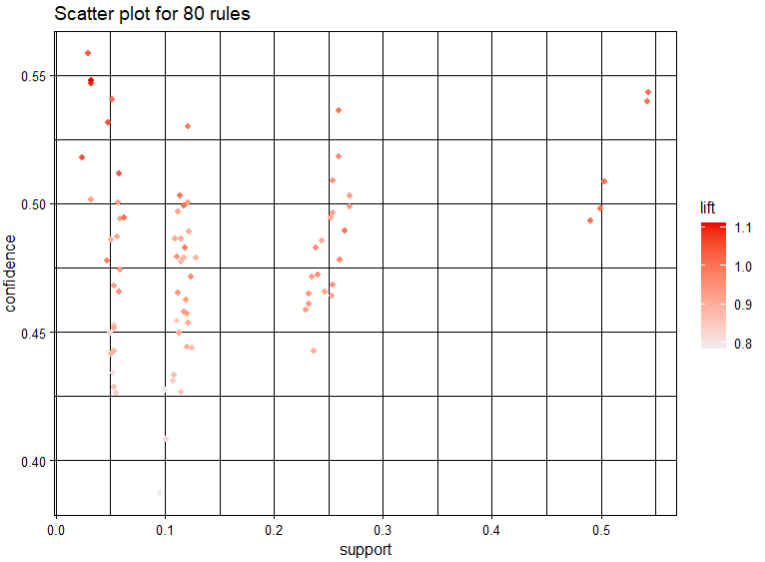


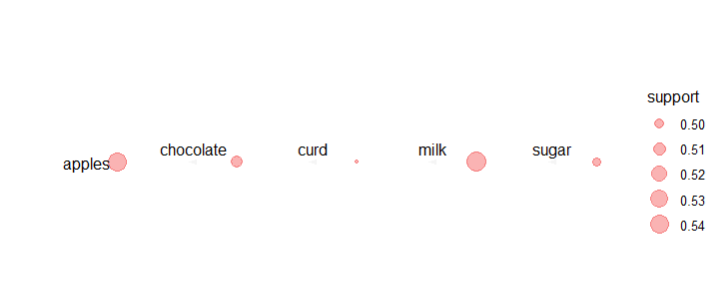


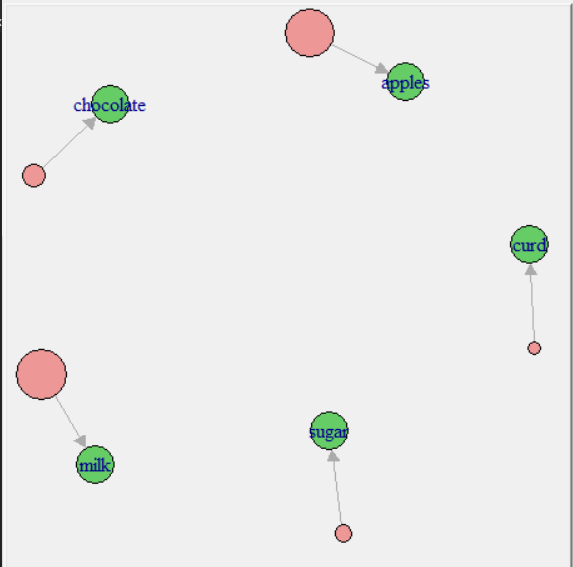












Q5. Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers.

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%

5.2Training set is chosen by i) hold out method ii) Random subsampling iii) Cross-Validation.

Compare the accuracy of the classifiers obtained.

5.3Data is scaled to standard format.

**.R FILE : IRIS**

View(iris)

n=nrow(iris)

n

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

#install.packages("rpart.plot",dependencies=TRUE)

#install.packages("e1071",dependencies=TRUE)

library(caret)

library(rpart.plot)

library(rpart)

library(caTools)

library(e1071)

#for(x in 1:10){

#PART 1

#Training set = 75%

#HOLD OUT METHOD

set.seed(123)

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

iris\_train = subset(iris,split==TRUE)

iris\_test = subset(iris,split==FALSE)

dim(iris\_test)

dim(iris\_train)

dtm<-rpart(Species~., iris\_train, method='class')

print(dtm)

plot(dtm)

text(dtm)

print(dtm)

rpart.plot(dtm)

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

p<-predict(dtm,iris\_test,type='class')

p

confusionMatrix(iris\_test[,5],p)

print(confusionMatrix(iris\_test[,5],p)$overall['Accuracy']\*100)

#SAMPLE RESAMPLING

for (x in 1:10){

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

iris\_train = subset(iris,split==TRUE)

iris\_test = subset(iris,split==FALSE)

dtm<-rpart(Species~., iris\_train, method='class')

print(dtm)

rpart.plot(dtm)

p<-predict(dtm,iris\_test, type='class')

print(confusionMatrix(iris\_test[,5],p))

print(confusionMatrix(iris\_test[,5],p)$overall['Accuracy']\*100)

}

#CROSS VALIDATION creating a model with cross validation 10 folds

model = train(iris\_train[,c(1,2,3,4)],iris\_train$Species,'rpart',trControl=trainControl(method='cv', number=10))

p<-predict(model,iris\_test, type='class')

print(confusionMatrix(iris\_test[,5],p))

print(confusionMatrix(iris\_test[,5],p)$overall['Accuracy']\*100)

#NORMALISE

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

iris\_n<-as.data.frame(lapply(iris[,c(1,2,3,4)],normalise))

str(iris\_n)

summary(iris\_n)

#K NEAREST NEIGHBOURS

iris\_train\_target<-iris\_train[,5]

iris\_test\_target<-iris\_test[,5]

k=sqrt(nrow(iris))

modelKNN<-knn(iris\_train[,c(1,2,3,4)],iris\_test[,c(1,2,3,4)],iris\_train\_target,k)

print(confusionMatrix(iris\_test[,5],modelKNN))

print(confusionMatrix(iris\_test[,5],modelKNN)$overall['Accuracy']\*100)

#NAIVE BAYES THEOREM

modelNB<-naiveBayes(Species~.,data=iris\_train)

pNB<-predict(modelNB,iris\_test)

print(confusionMatrix(iris\_test[,5],pNB))

print(confusionMatrix(iris\_test[,5],pNB)$overall['Accuracy']\*100)

#PART 2

#Training set = 66.6%

#HOLD OUT METHOD

set.seed(123)

split<-sample.split(iris$Species,SplitRatio = 0.666)

iris\_train = subset(iris,split==TRUE)

iris\_test = subset(iris,split==FALSE)

dim(iris\_test)

dim(iris\_train)

dtm<-rpart(Species~., iris\_train, method='class')

print(dtm)

plot(dtm)

text(dtm)

print(dtm)

rpart.plot(dtm)

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

p<-predict(dtm,iris\_test,type='class')

p

confusionMatrix(iris\_test[,5],p)

print(confusionMatrix(iris\_test[,5],p)$overall['Accuracy']\*100)

#SAMPLE RESAMPLING

for (x in 1:10){

split<-sample.split(iris$Species,SplitRatio = 0.666)

iris\_train = subset(iris,split==TRUE)

iris\_test = subset(iris,split==FALSE)

dtm<-rpart(Species~., iris\_train, method='class')

print(dtm)

rpart.plot(dtm)

p<-predict(dtm,iris\_test, type='class')

print(confusionMatrix(iris\_test[,5],p))

print(confusionMatrix(iris\_test[,5],p)$overall['Accuracy']\*100)

}

#CROSS VALIDATION creating a model with cross validation 10 folds

model = train(iris\_train[,c(1,2,3,4)],iris\_train$Species,'rpart',trControl=trainControl(method='cv', number=10))

p<-predict(model,iris\_test, type='class')

print(confusionMatrix(iris\_test[,5],p))

print(confusionMatrix(iris\_test[,5],p)$overall['Accuracy']\*100)

#NORMALISE

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

iris\_n<-as.data.frame(lapply(iris[,c(1,2,3,4)],normalise))

str(iris\_n)

summary(iris\_n)

#K NEAREST NEIGHBOURS

iris\_train\_target<-iris\_train[,5]

iris\_test\_target<-iris\_test[,5]

k=sqrt(nrow(iris))

modelKNN<-knn(iris\_train[,c(1,2,3,4)],iris\_test[,c(1,2,3,4)],iris\_train\_target,k)

print(confusionMatrix(iris\_test[,5],modelKNN))

print(confusionMatrix(iris\_test[,5],modelKNN)$overall['Accuracy']\*100)

#NAIVE BAYES THEOREM

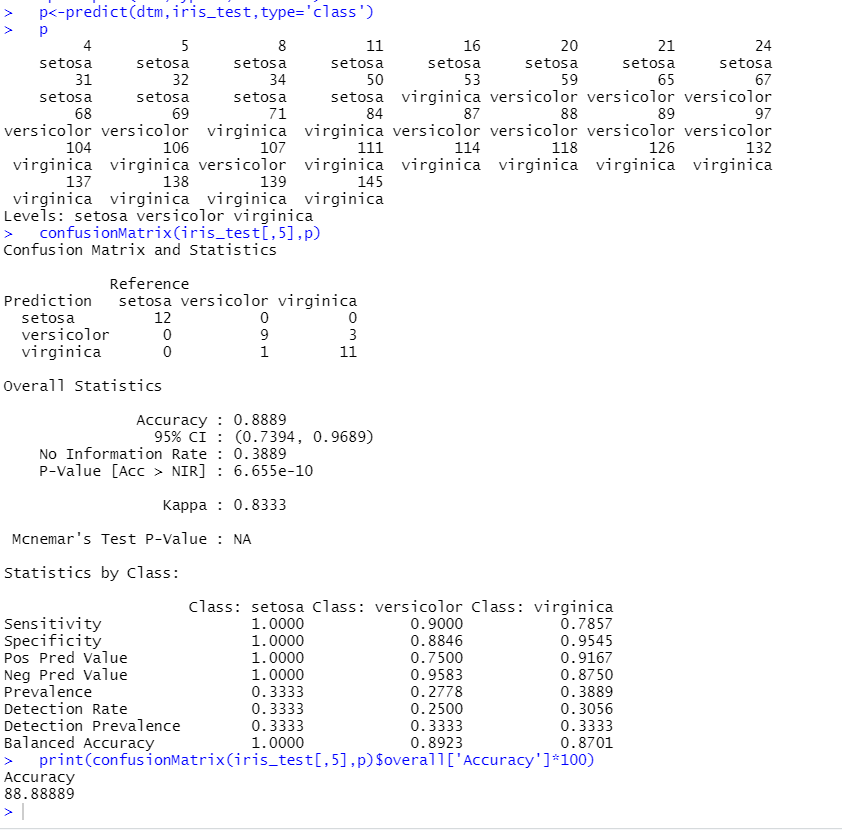
modelNB<-naiveBayes(Species~.,data=iris\_train)

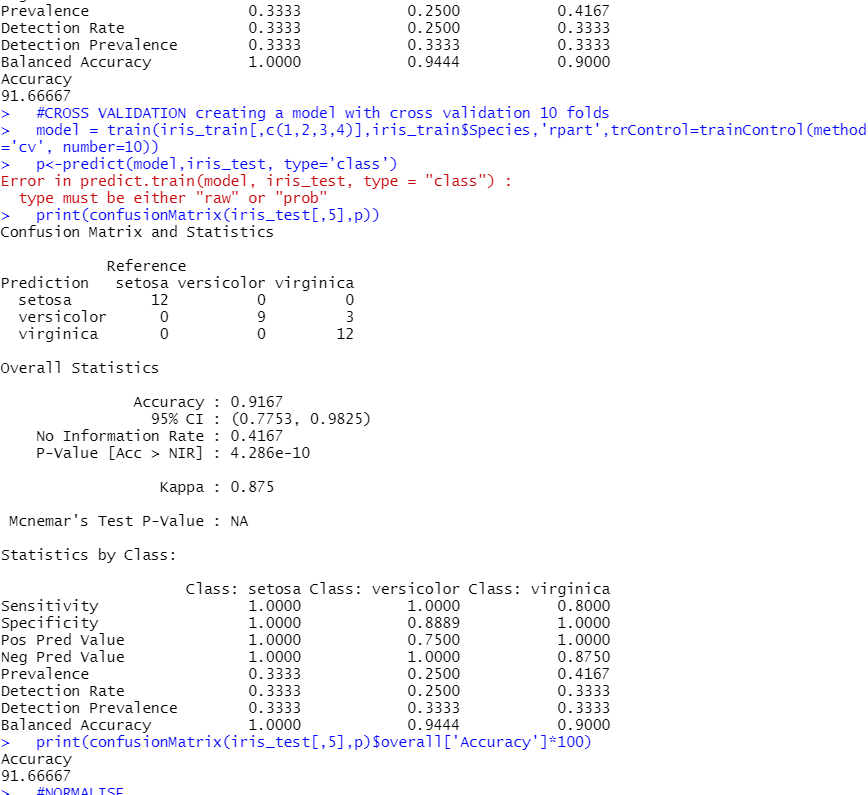
pNB<-predict(modelNB,iris\_test)

print(confusionMatrix(iris\_test[,5],pNB))

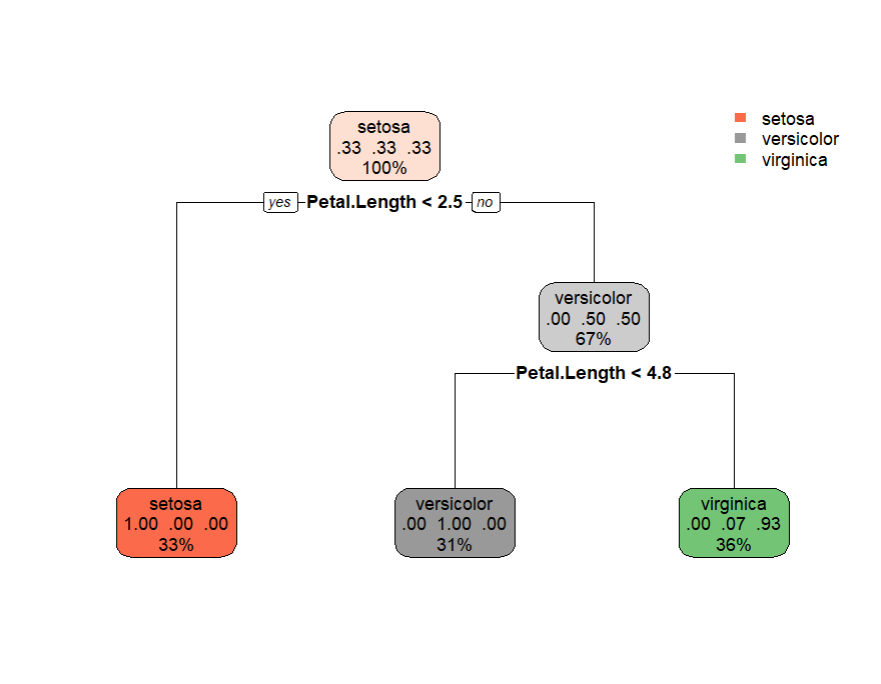
print(confusionMatrix(iris\_test[,5],pNB)$overall['Accuracy']\*100)

**OUTPUT** :**IRIS**





**PLOTS :IRIS**



**.R FILE:BREAST CANCER WINCONSIN**

#---------------------BREAST CANCER----------------

cc= read.csv("C:/Users/akanksha goel/OneDrive/Documents/program/breast-cancer-wisconsin.data", header=TRUE)

names(cc)=c("code\_number","clump thickness","cell size","cell shape","marginal adhesion","epithelial cell size","bare nuclei","bland chromatin","normal nucleoli","mitoses","Class")

cc <- na.omit(cc)

cc

cancer = as.data.frame(lapply(cc, function(x){

if(any(is.infinite(x))){

which(is.infinite(x))

x[is.infinite((x))] = 0

}

return(as.numeric(x))

}))

cancer = na.omit(cancer)

library(caret)

library(rpart.plot)

library(rpart)

library(caTools)

library(e1071)

#for(x in 1:10){

#PART 1

#Training set = 75%

#HOLD OUT METHOD

set.seed(124)

split<-sample.split(factor(cancer[,11]),SplitRatio = 0.75)

b\_train = subset(cancer,split==TRUE)

b\_test = subset(cancer,split==FALSE)

dim(b\_test)

dim(b\_train)

dtm<-rpart(Class~., b\_train, method='class')

print(dtm)

plot(dtm)

text(dtm)

print(dtm)

rpart.plot(dtm)

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

pb<-predict(dtm,b\_test,type='class')

pb

confusionMatrix(factor(b\_test[,11]),pb)

confusionMatrix(factor(b\_test[,11]),pb)$overall['Accuracy']\*100

#SAMPLE RESAMPLING

for (x in 1:10){

split<-sample.split(factor(cancer[,11]),SplitRatio = 0.75)

b\_train = subset(cancer,split==TRUE)

b\_test = subset(cancer,split==FALSE)

dim(b\_test)

dim(b\_train)

dtm<-rpart(Class~., b\_train, method='class')

print(dtm)

rpart.plot(dtm)

pb<-predict(dtm,b\_test,type='class')

pb

confusionMatrix(factor(b\_test[,11]),pb)

confusionMatrix(factor(b\_test[,11]),pb)$overall['Accuracy']\*100

}

#CROSS VALIDATION creating a model with cross validation 10 folds

model = train(b\_train[,c(2:10)],factor(b\_train[,11]),'rpart',trControl=trainControl(method='cv', number=10))

p<-predict(model,b\_test[,c(2:10)])

confusionMatrix(factor(b\_test[,11],c('2','4')),p)$overall['Accuracy']\*100

#NORMALISE

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

cancer\_n<-as.data.frame(lapply(cancer[,2:10],normalise))

str(cancer\_n)

summary(cancer\_n)

#K NEAREST NEIGHBOURS

b\_train\_target<-b\_train[,11]

b\_test\_target<-b\_test[,11]

k=sqrt(nrow(cc))

knn\_model<-knn(b\_train[,c(2:10)],b\_test[,c(2:10)],b\_train\_target,k)

confusionMatrix(factor(b\_test[,11],c('2','4')),knn\_p)$overall['Accuracy']\*100

#NAIVE BAYES THEOREM

modelNB<-naiveBayes(Class~.,data=b\_train)

pNB<-predict(modelNB,b\_test)

print(confusionMatrix(factor(b\_test[,11],c('2','4')),pNB))

print(confusionMatrix(factor(b\_test[,11],c('2','4')),pNB)$overall['Accuracy']\*100)

#PART 2

#Training set = 66.6%

#HOLD OUT METHOD

set.seed(124)

split<-sample.split(factor(cancer[,11]),SplitRatio = 0.666)

b\_train = subset(cancer,split==TRUE)

b\_test = subset(cancer,split==FALSE)

dim(b\_test)

dim(b\_train)

dtm<-rpart(Class~., b\_train, method='class')

print(dtm)

plot(dtm)

text(dtm)

print(dtm)

rpart.plot(dtm)

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

pb<-predict(dtm,b\_test,type='class')

pb

confusionMatrix(factor(b\_test[,11]),pb)

confusionMatrix(factor(b\_test[,11]),pb)$overall['Accuracy']\*100

#SAMPLE RESAMPLING

for (x in 1:10){

split<-sample.split(factor(cancer[,11]),SplitRatio = 0.666)

b\_train = subset(cancer,split==TRUE)

b\_test = subset(cancer,split==FALSE)

dim(b\_test)

dim(b\_train)

dtm<-rpart(Class~., b\_train, method='class')

print(dtm)

rpart.plot(dtm)

pb<-predict(dtm,b\_test,type='class')

pb

confusionMatrix(factor(b\_test[,11]),pb)

confusionMatrix(factor(b\_test[,11]),pb)$overall['Accuracy']\*100

}

#CROSS VALIDATION creating a model with cross validation 10 folds

model = train(b\_train[,c(2:10)],factor(b\_train[,11]),'rpart',trControl=trainControl(method='cv', number=10))

p<-predict(model,b\_test[,c(2:10)])

confusionMatrix(factor(b\_test[,11],c('2','4')),p)$overall['Accuracy']\*100

#NORMALISE

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

cancer\_n<-as.data.frame(lapply(cancer[,2:10],normalise))

str(cancer\_n)

summary(cancer\_n)

#K NEAREST NEIGHBOURS

b\_train\_target<-b\_train[,11]

b\_test\_target<-b\_test[,11]

k=sqrt(nrow(cc))

knn\_model<-knn(b\_train[,c(2:10)],b\_test[,c(2:10)],b\_train\_target,k)

confusionMatrix(factor(b\_test[,11],c('2','4')),knn\_model)$overall['Accuracy']\*100

#NAIVE BAYES THEOREM

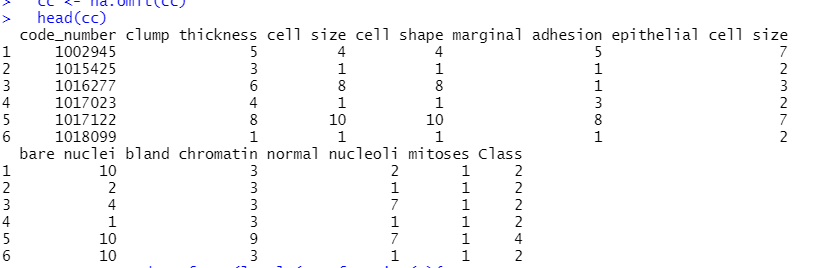
modelNB<-naiveBayes(Class~.,data=b\_train)

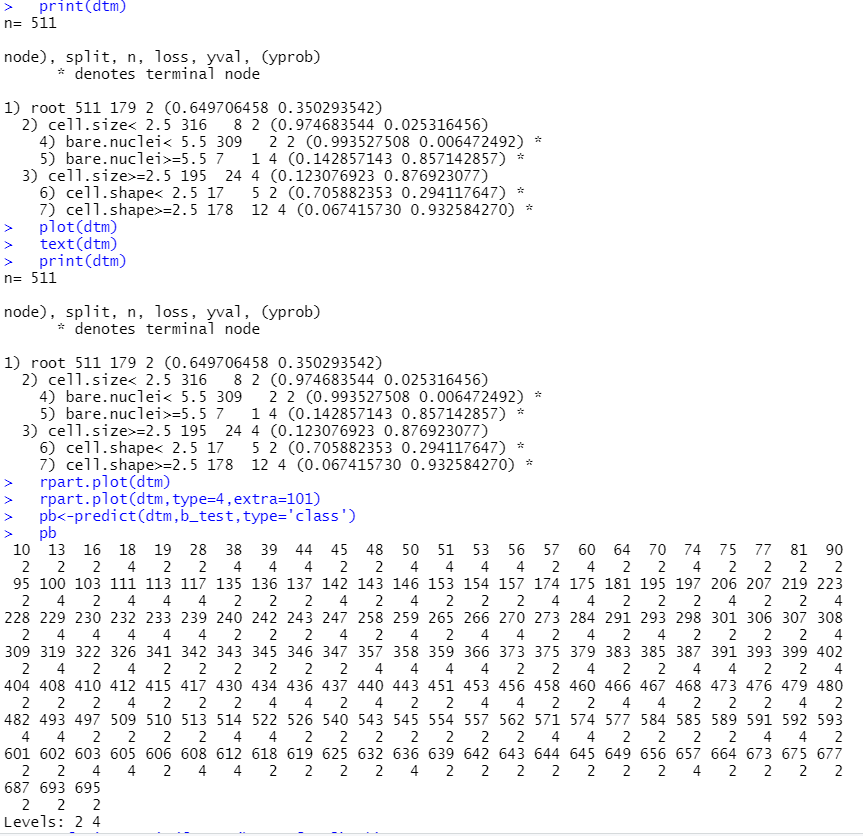
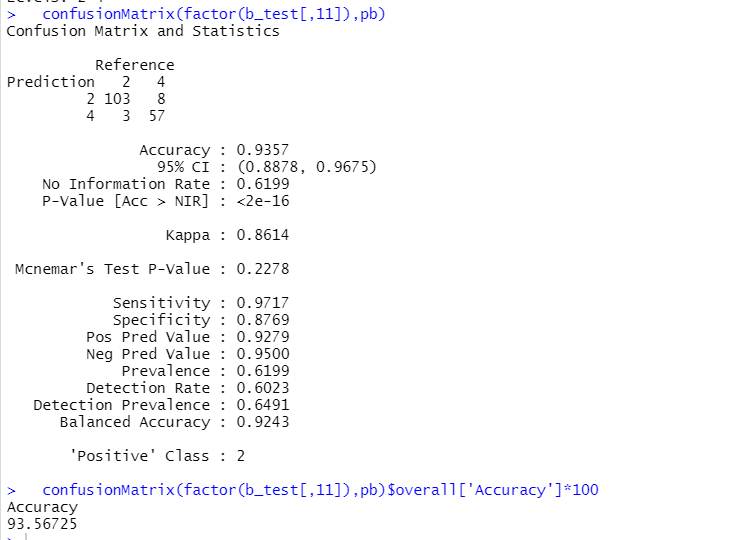
pNB<-predict(modelNB,b\_test)

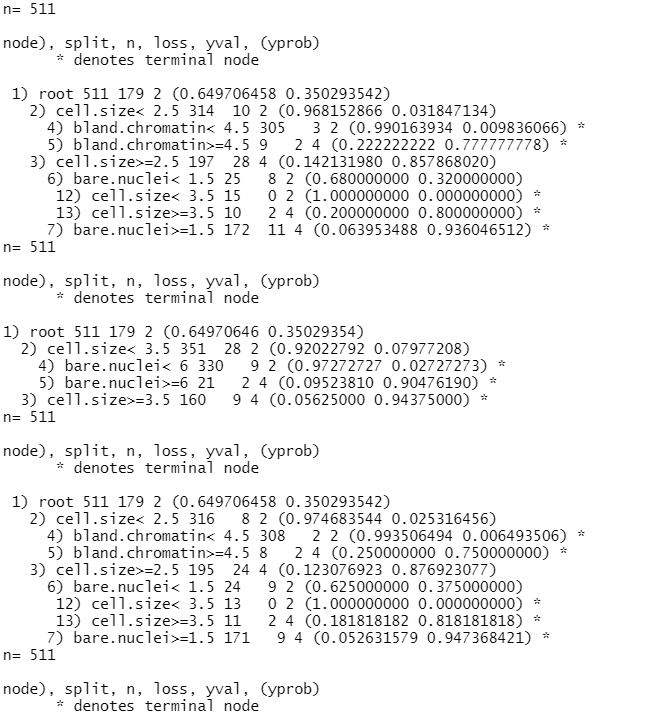
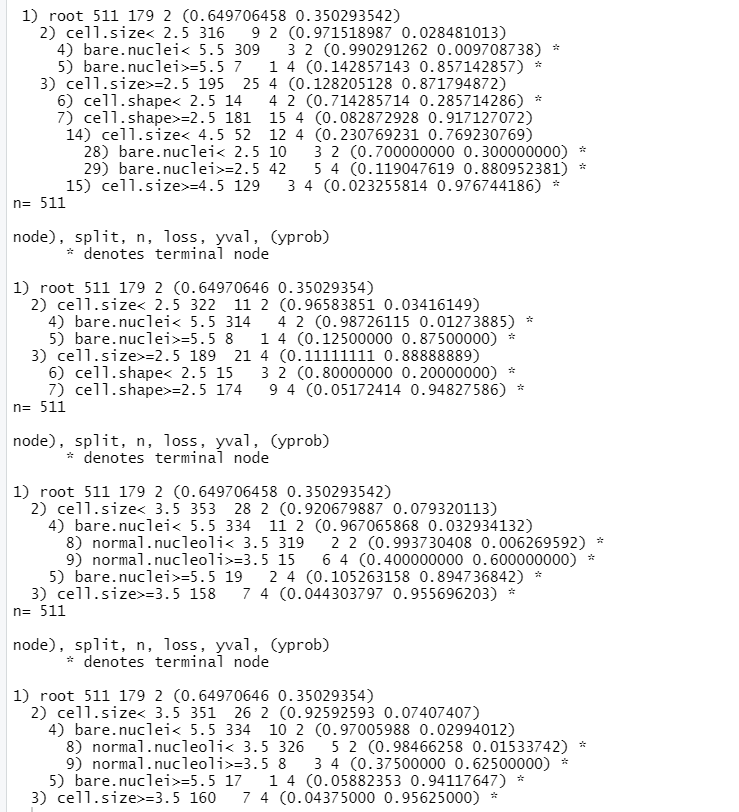
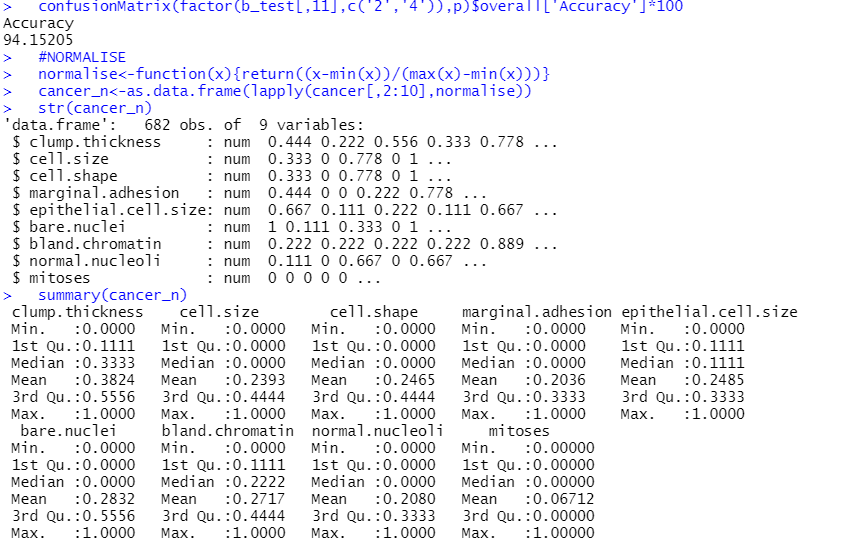
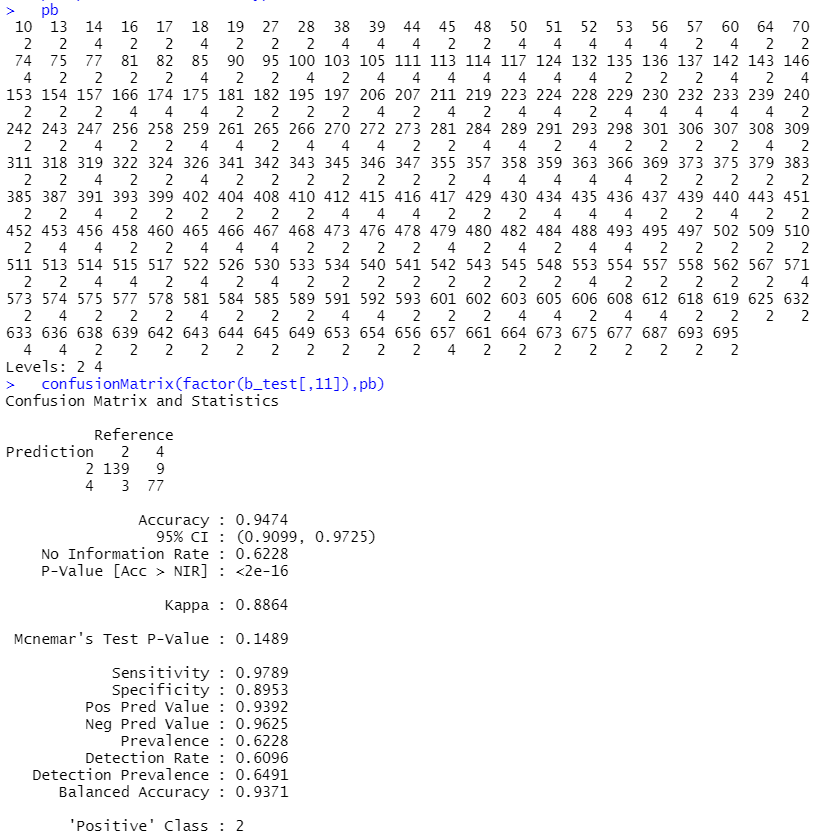
print(confusionMatrix(factor(b\_test[,11],c('2','4')),pNB))

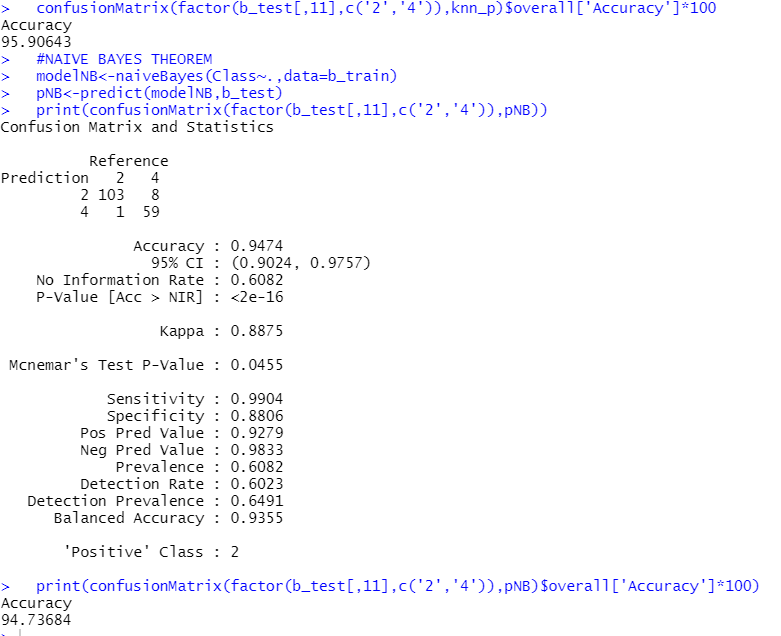
print(confusionMatrix(factor(b\_test[,11],c('2','4')),pNB)$overall['Accuracy']\*100)

**OUTPUT**  **BREAST CANCER WINCONSIN**



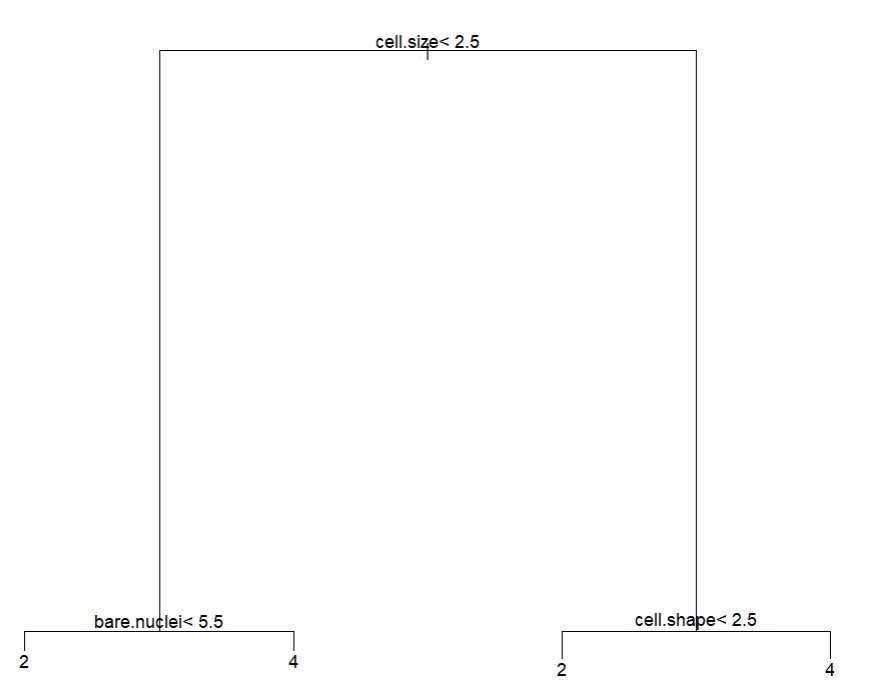
 

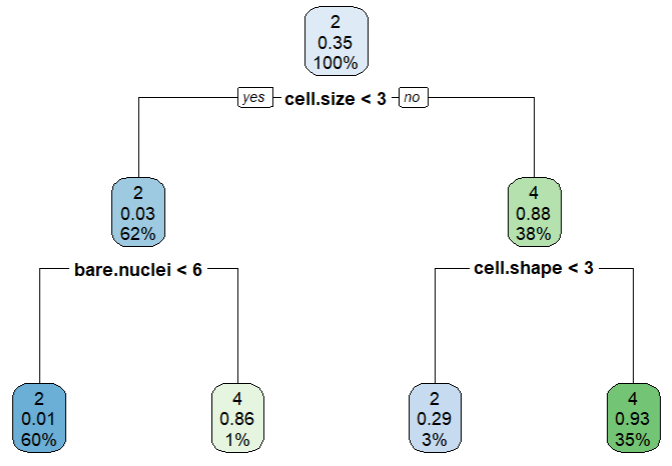


**PLOTS BREAST CANCER WINCONSIN**

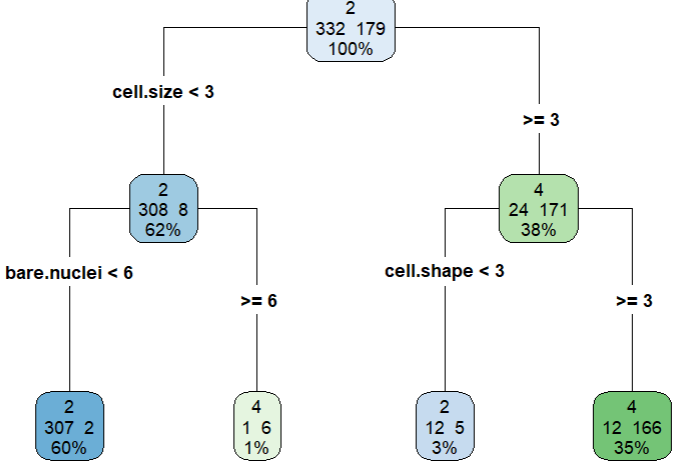
plot(dtm)



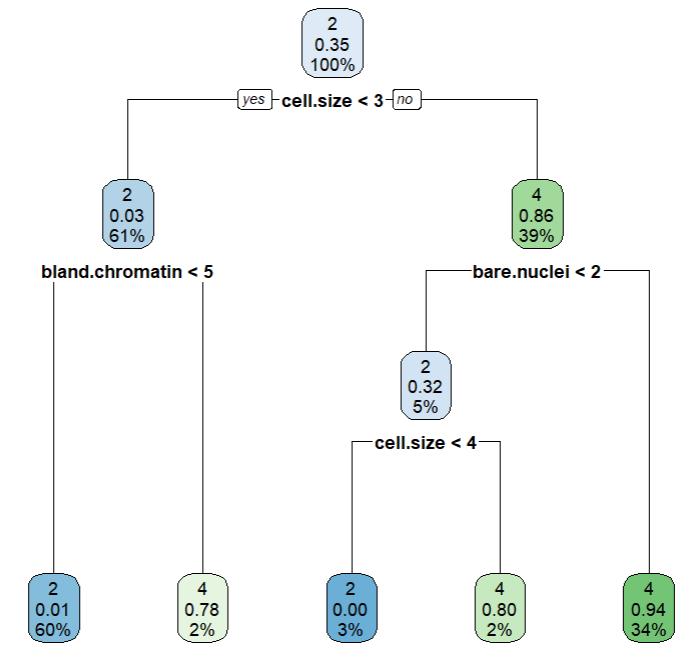
rpart.plot(dtm)

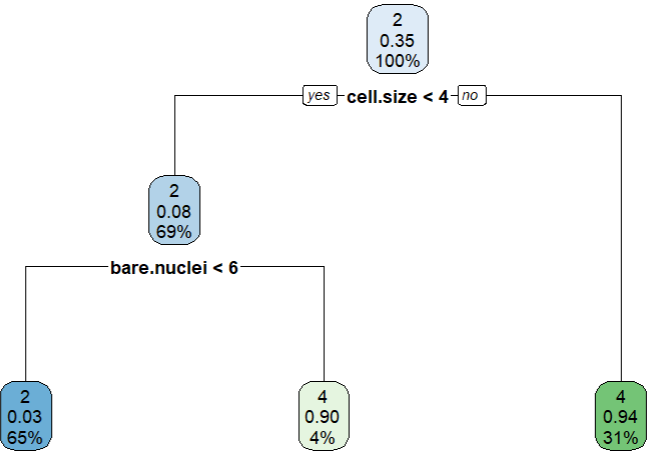
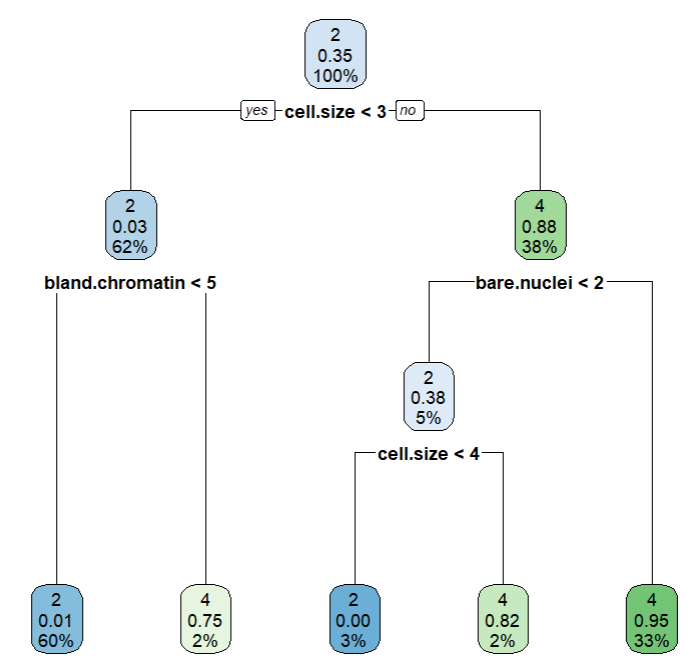
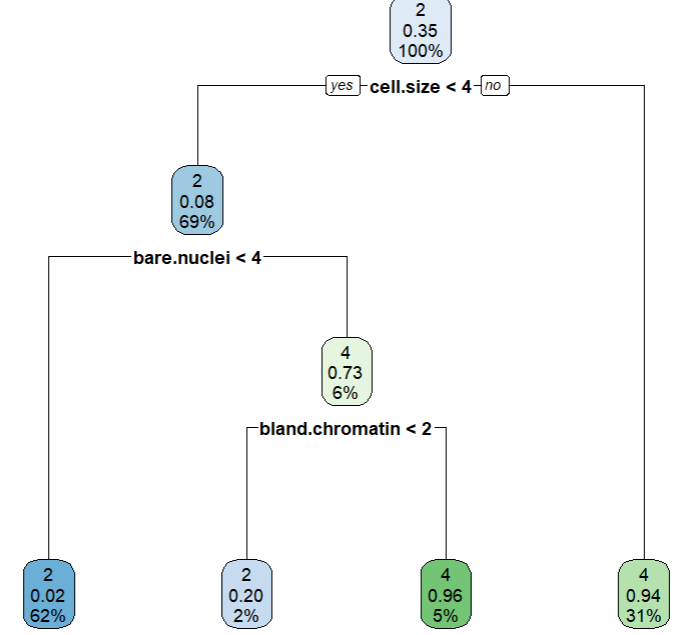


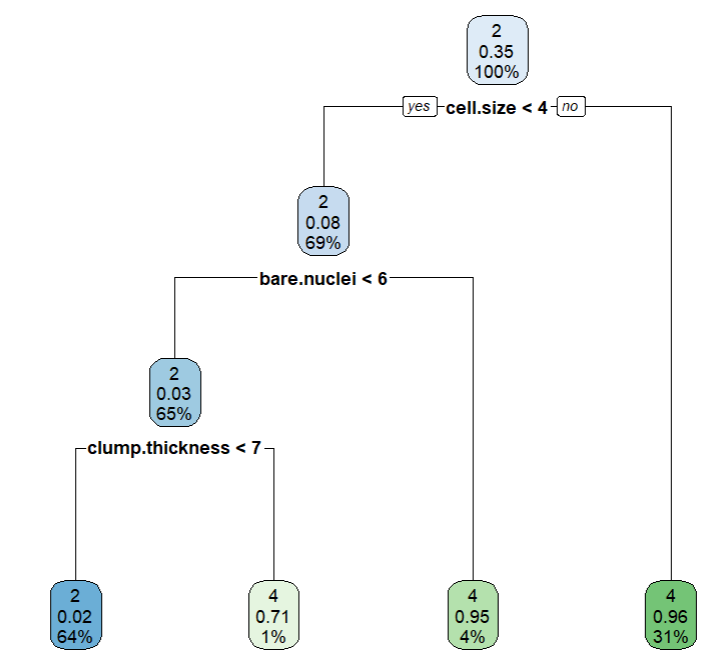
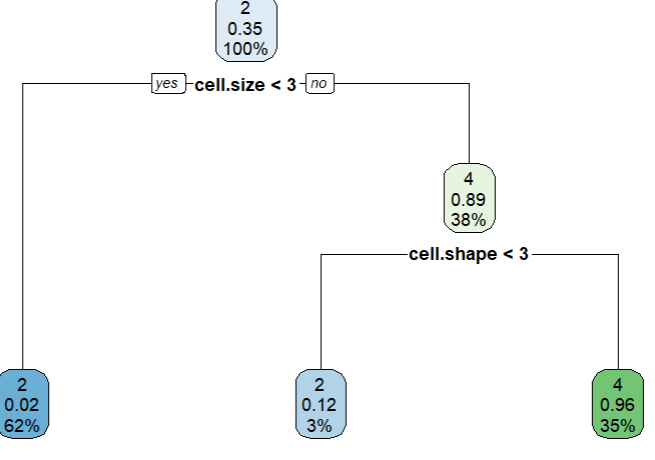
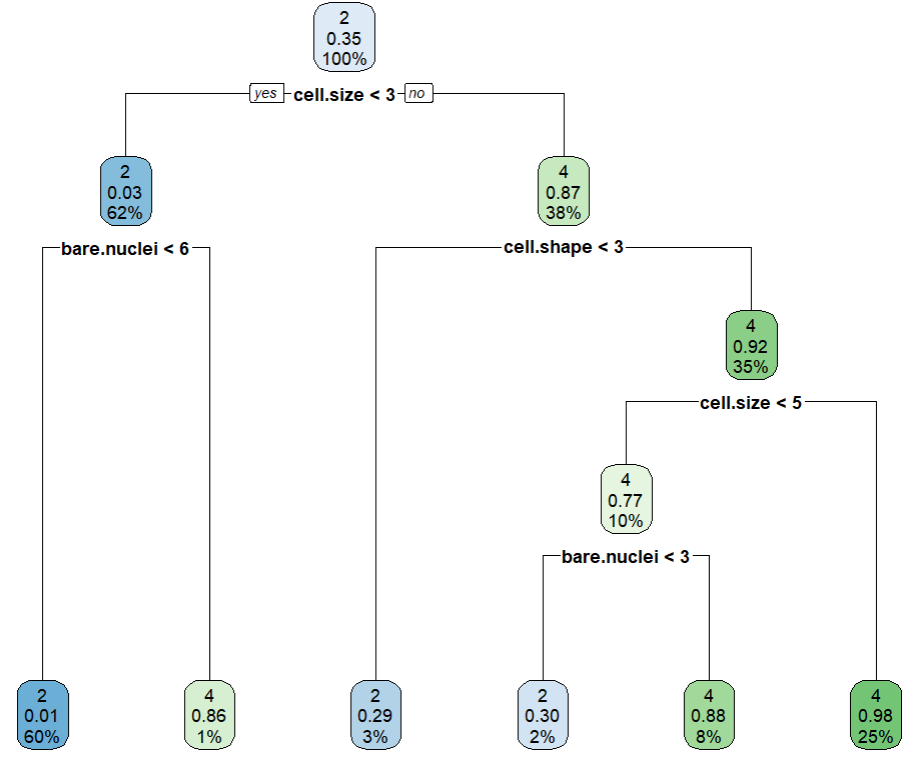
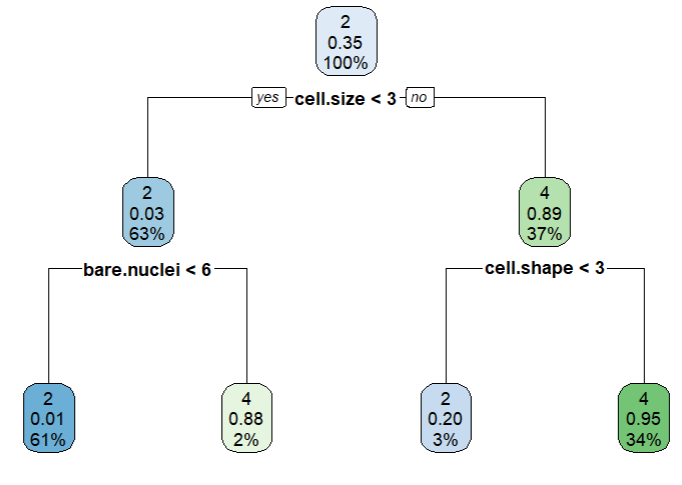
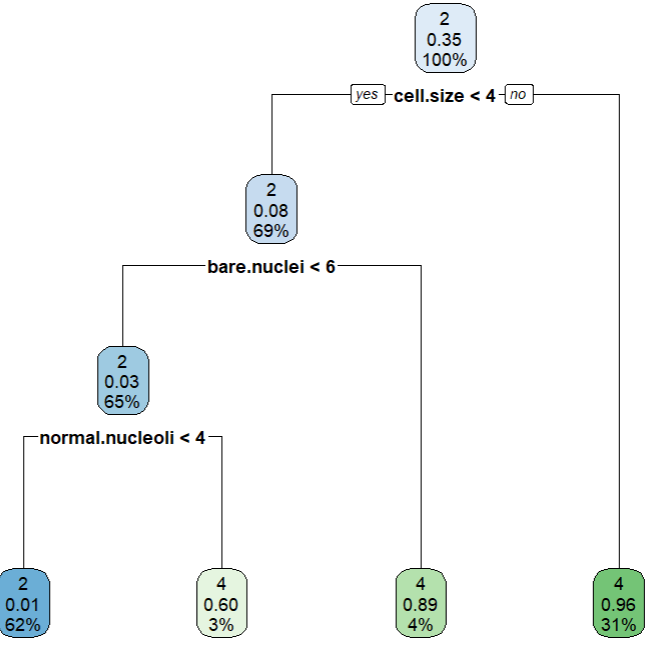
rpart.plot(dtm,type=4,extra=101)



rpart.plot(dtm)



Q6. Use Simple Kmeans, DBScan, Hierachical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

**.R FILE**

K MEANS ALGORITHM

data(iris)

plot(iris)

plot(iris$Sepal.Length,iris$Sepal.Width)

str(iris)

irisScaled<-scale(iris[,-5])

head(irisScaled)

summary(irisScaled)

#K Means Clustering

fitK<-kmeans(irisScaled,3)

fitK

fitK$size

fitK$cluster

str(fitK)

plot(iris,col=fitK$cluster)

#Choosing k

k<-list()

for (i in 1:10) {

k[[i]] <- kmeans(irisScaled,i)

}

#will iterate through 1 to 10 and fit kmeans model for each value and will be saved to a list called k

k

# observe Within cluster sum of squares by cluster value , it is 0 when cluster k =1 and increases with value of k

#for every k mean ,calculate between sum of square / total sum of square and then plot

betweenss\_totss <-list()

for (i in 1:10) {

betweenss\_totss[[i]] <- k[[i]]$betweenss/k[[i]]$totss

}

plot(1:10,betweenss\_totss,type = "b",ylab = "betweenss ss/total ss",xlab = "Cluster(k)")

for (i in 1:4) {

plot(iris,col=k[[i]]$cluster)

}

table(iris$Species,fitK$cluster) #compare the clusters with the species

#can see overlaps

library(ggplot2)

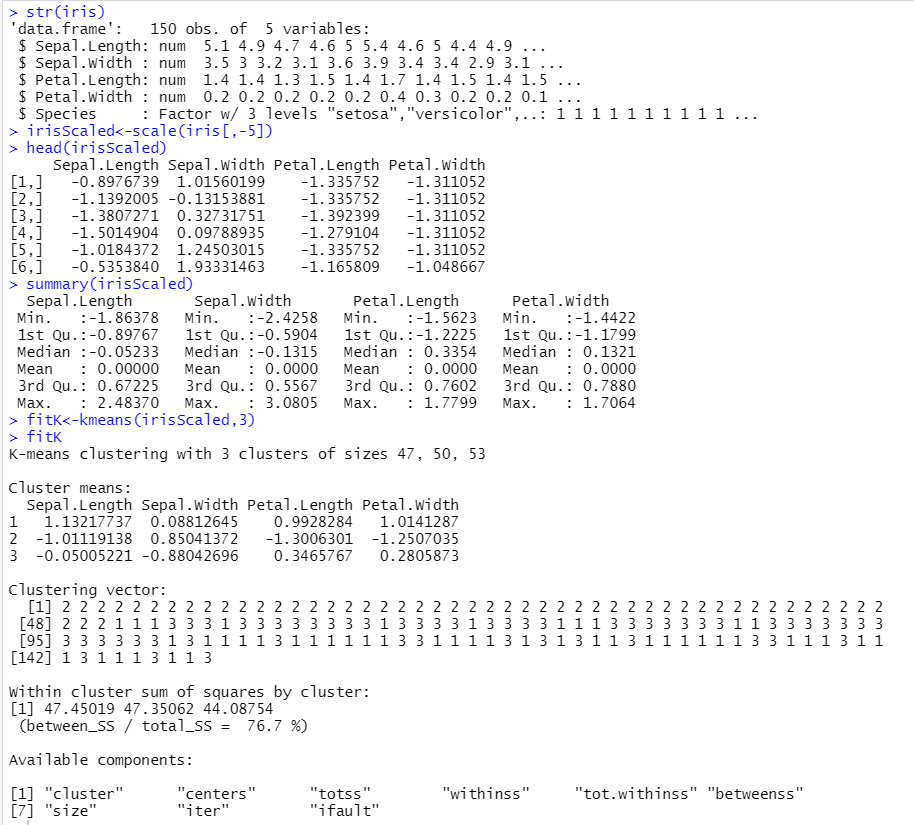
ggplot(iris, aes(Petal.Length, Petal.Width,color = Species)) + geom\_point()

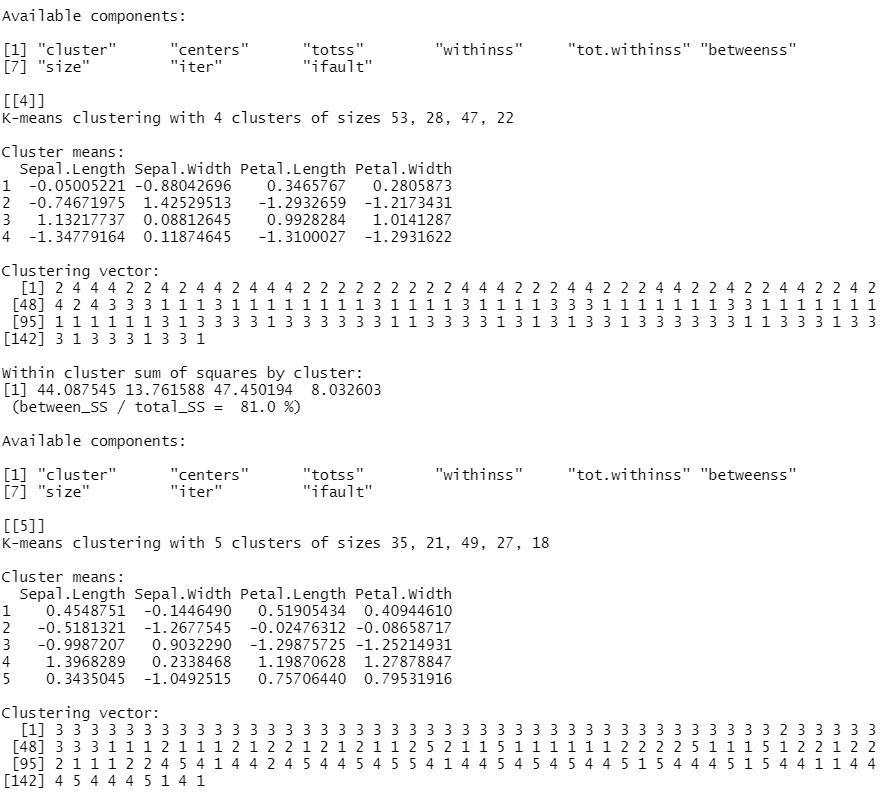
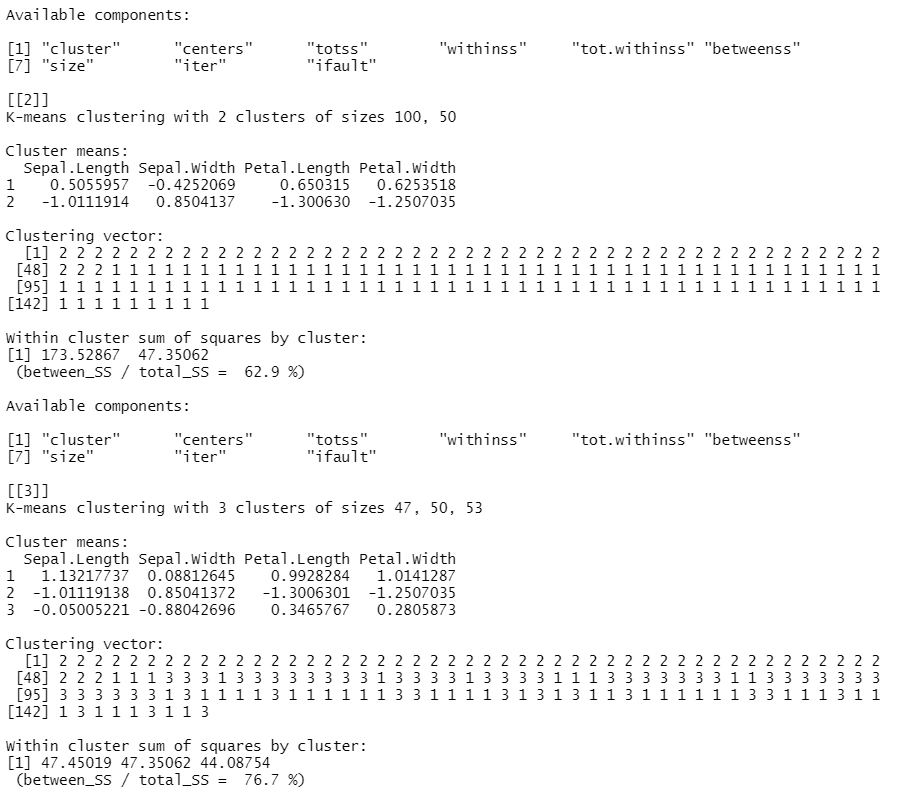
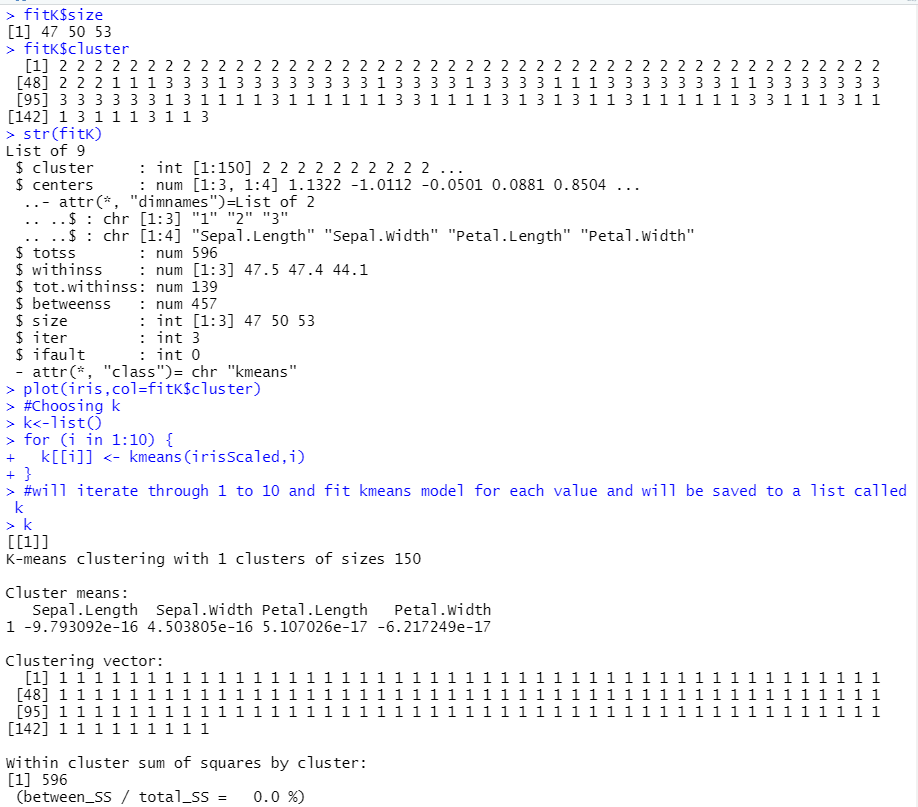
ggplot(iris, aes(Petal.Length, Petal.Width,color = fitK$cluster)) + geom\_point()

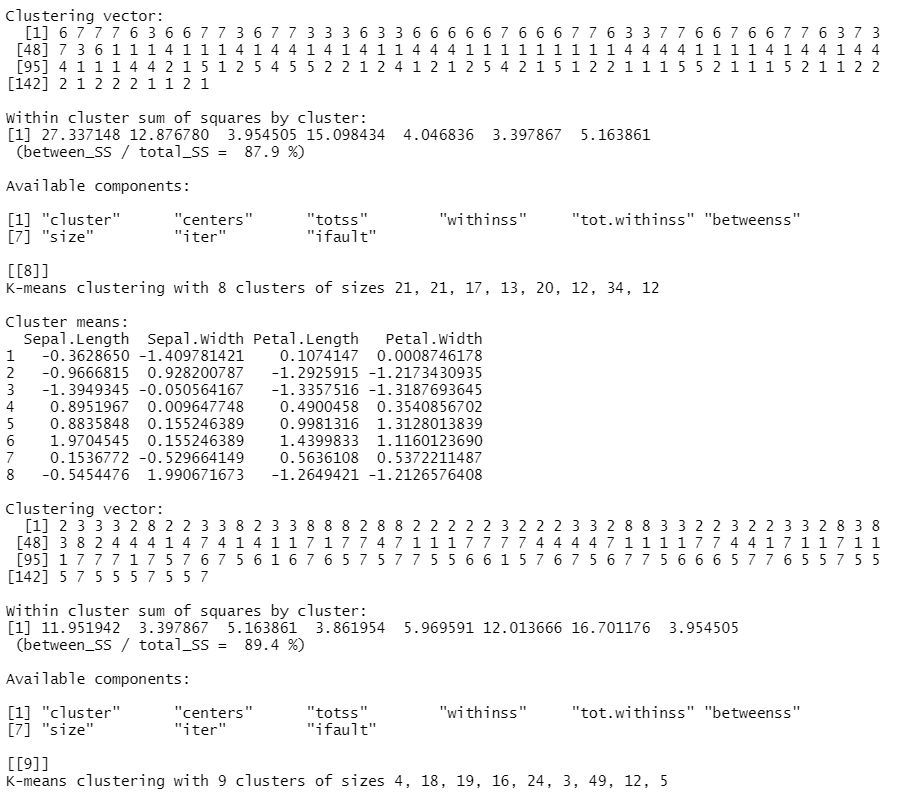
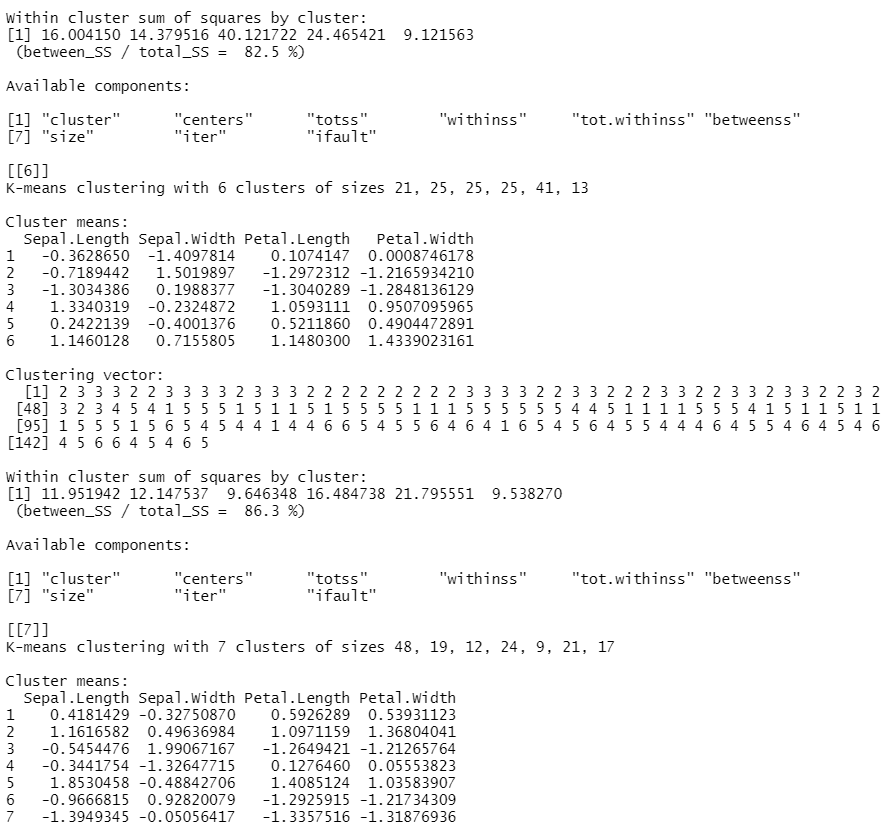
#can visualize and compare similarly for Sepal width and length

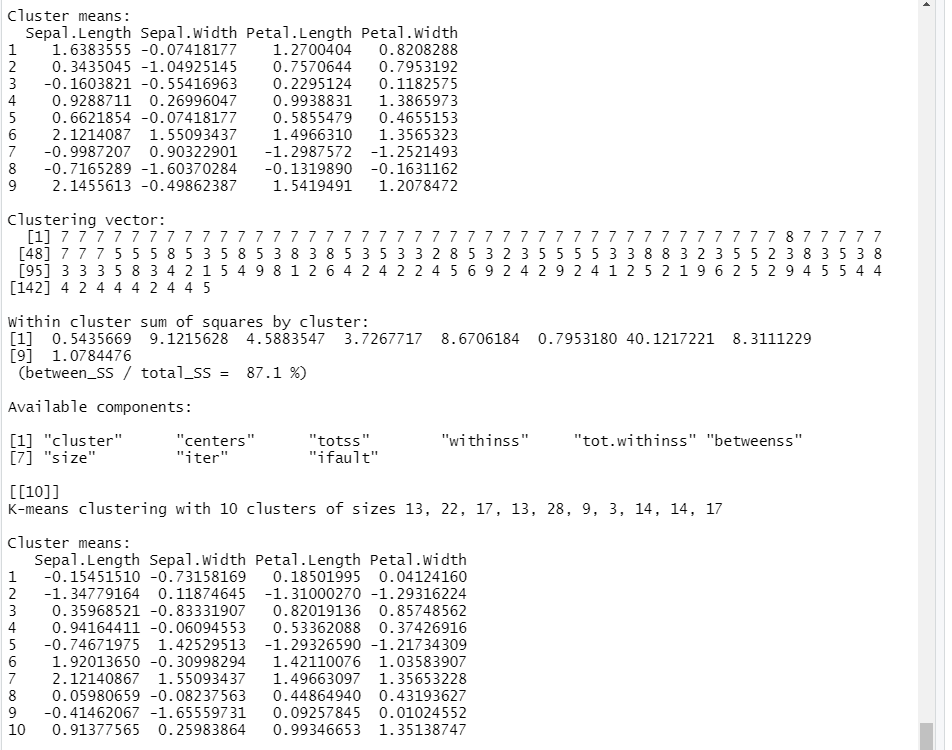
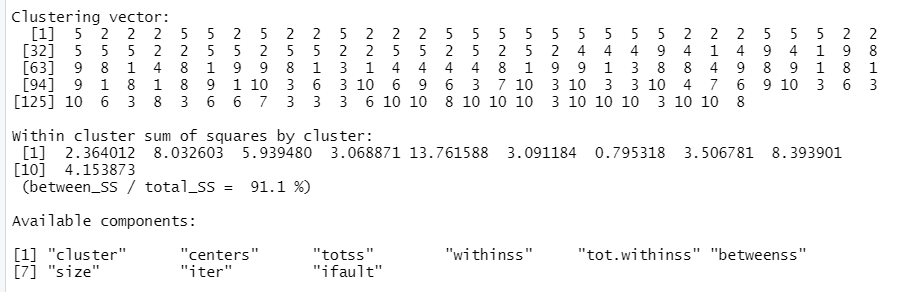
**OUTPUT**

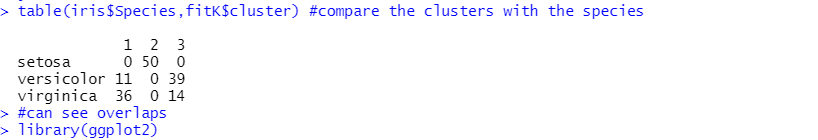
K MEANS ALGORITHM





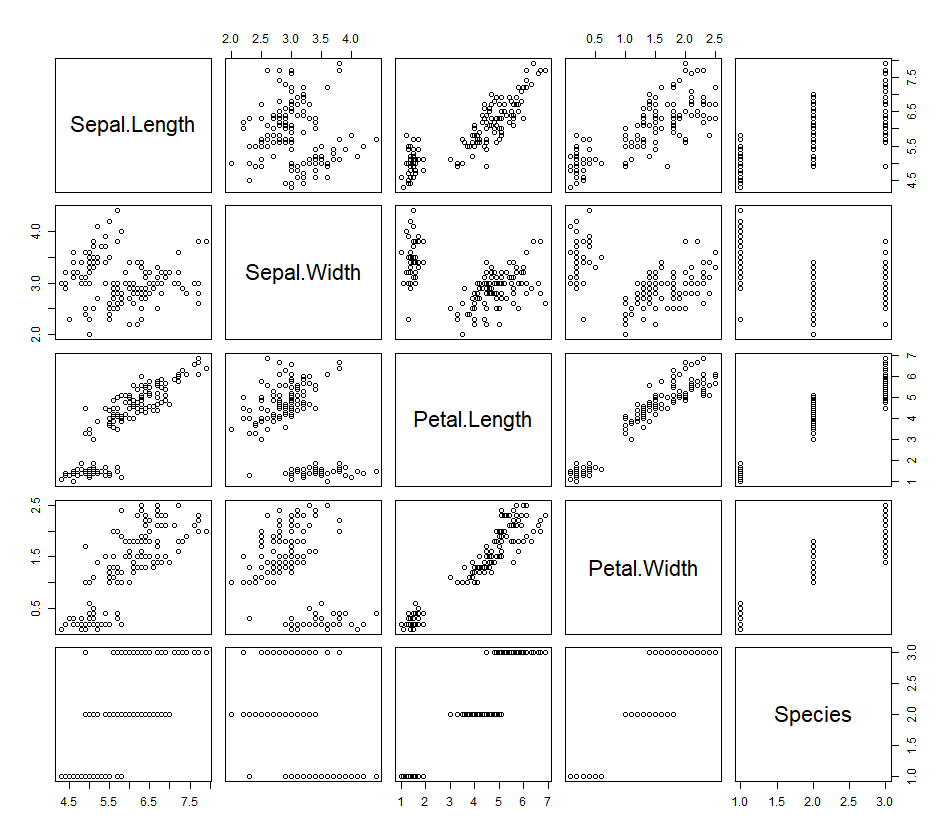




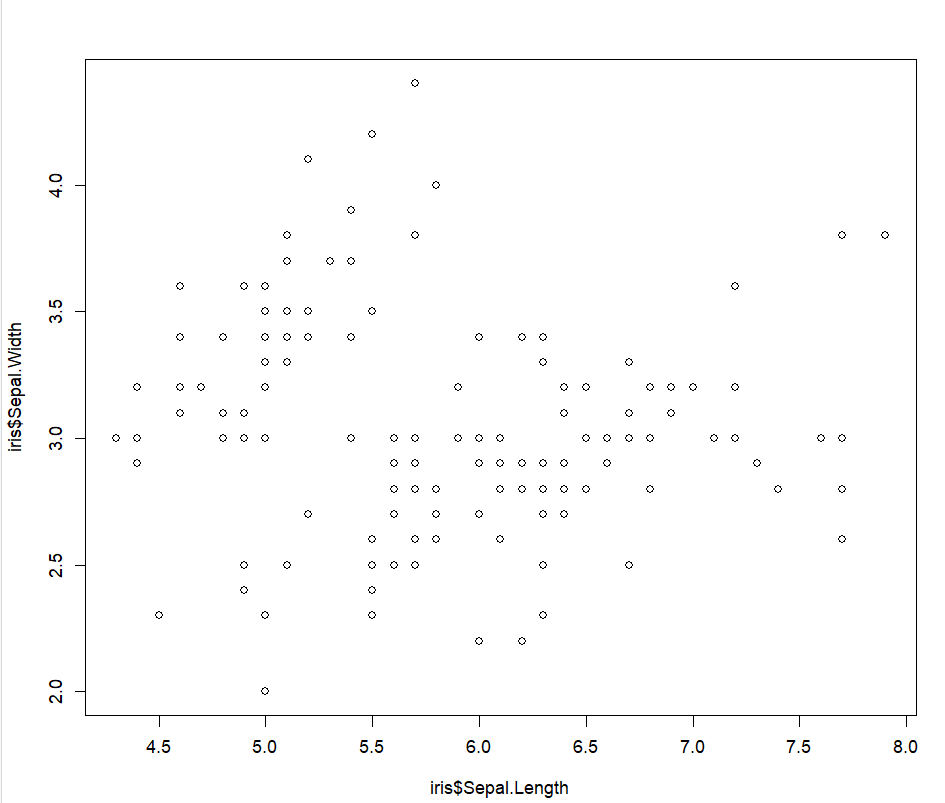


**PLOTS** : K MEANS

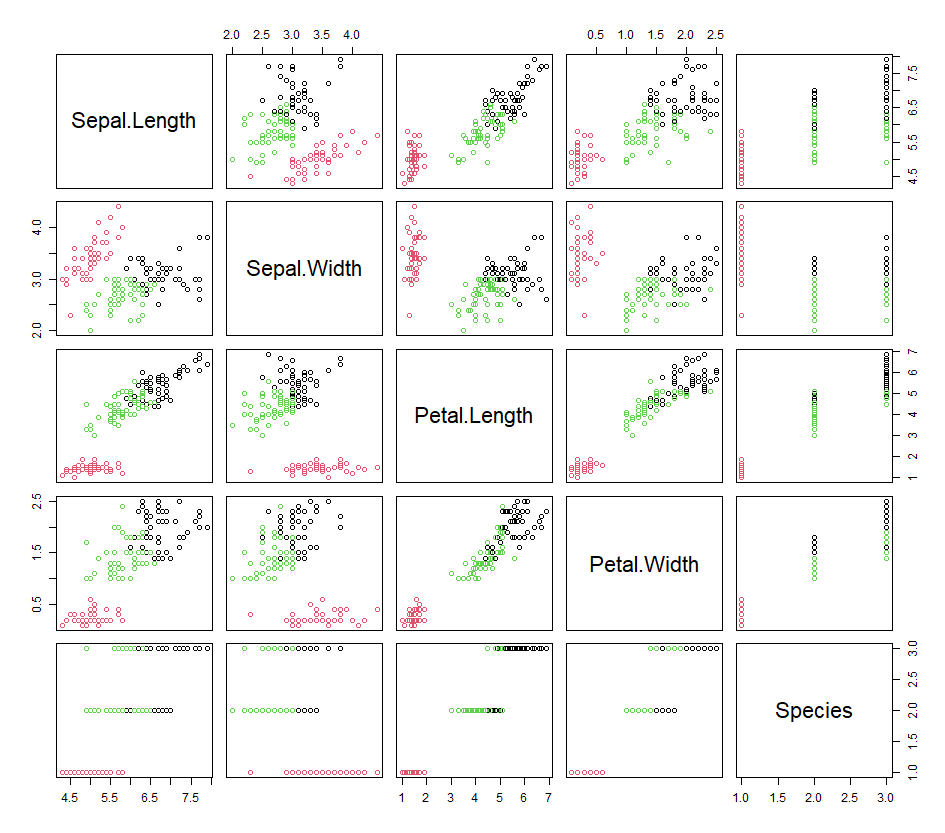
plot(iris)



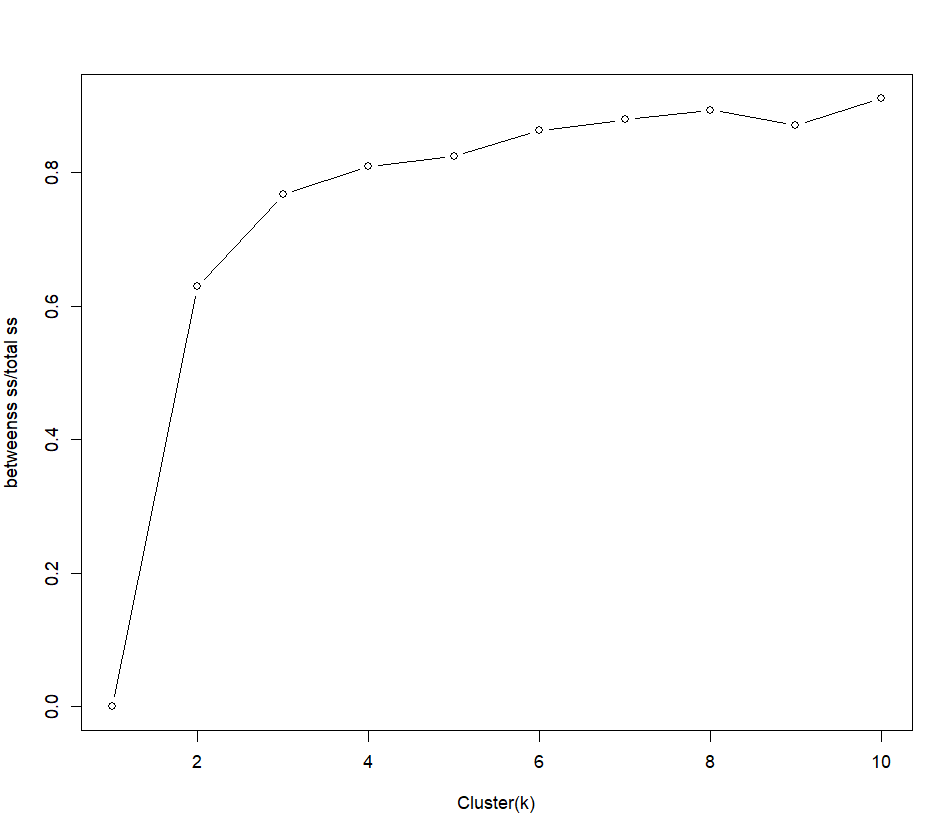
plot(iris$Sepal.Length,iris$Sepal.Width)



plot(iris,col=fitK$cluster)

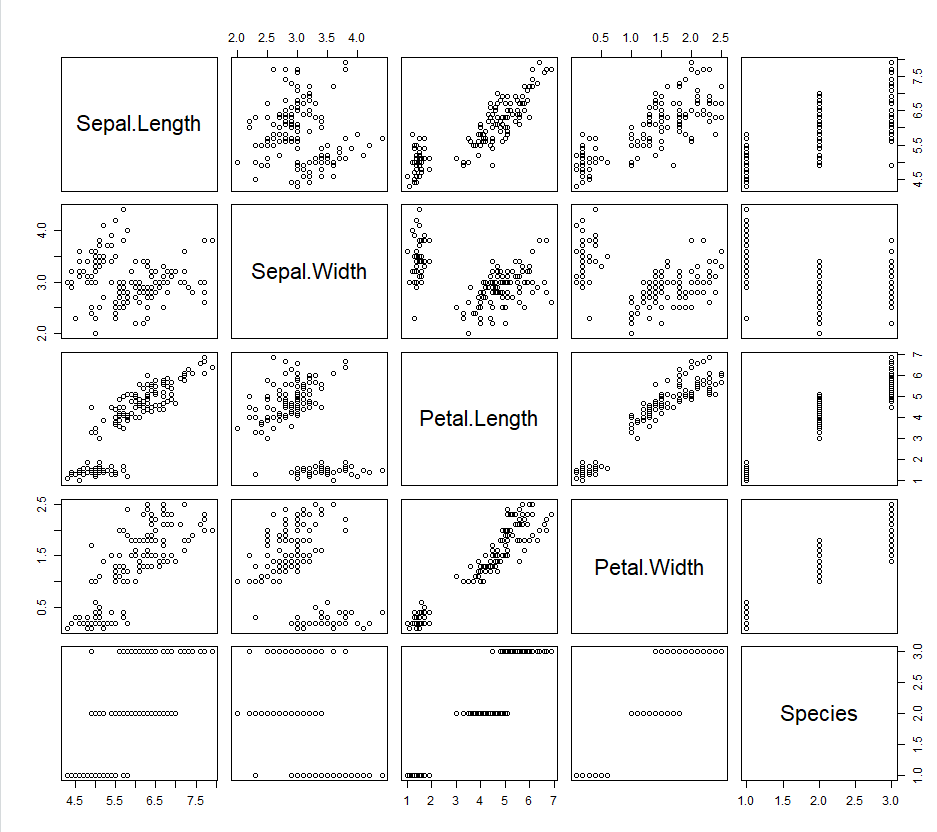


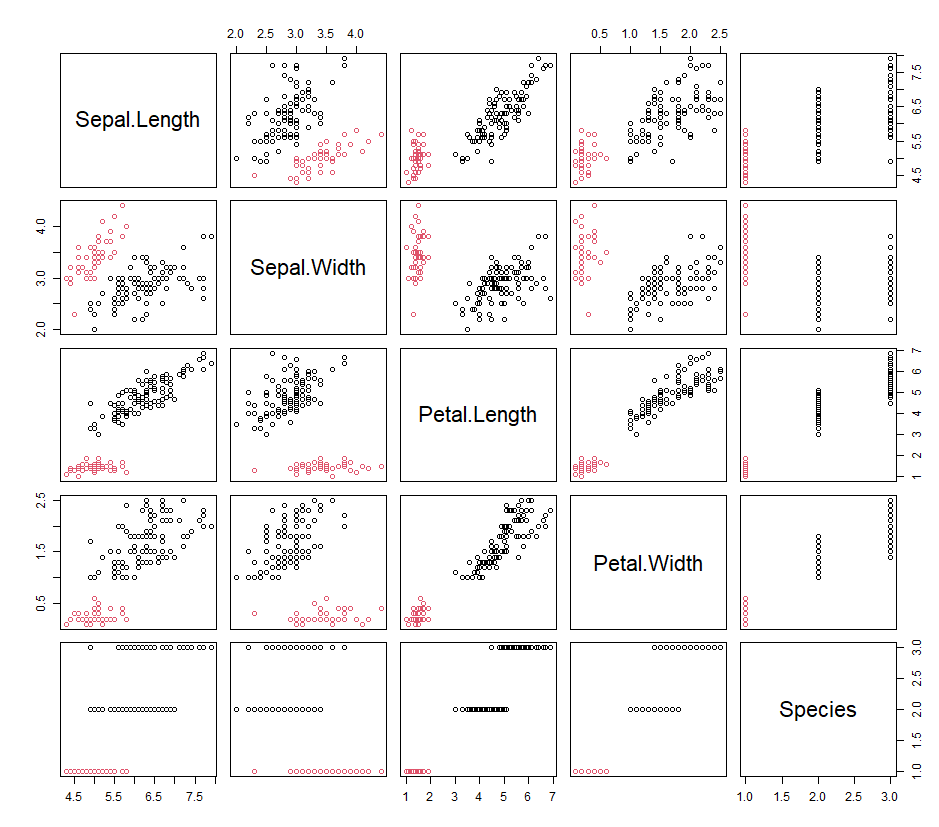
plot(1:10,betweenss\_totss,type = "b",ylab = "betweenss ss/total ss",xlab = "Cluster(k)")

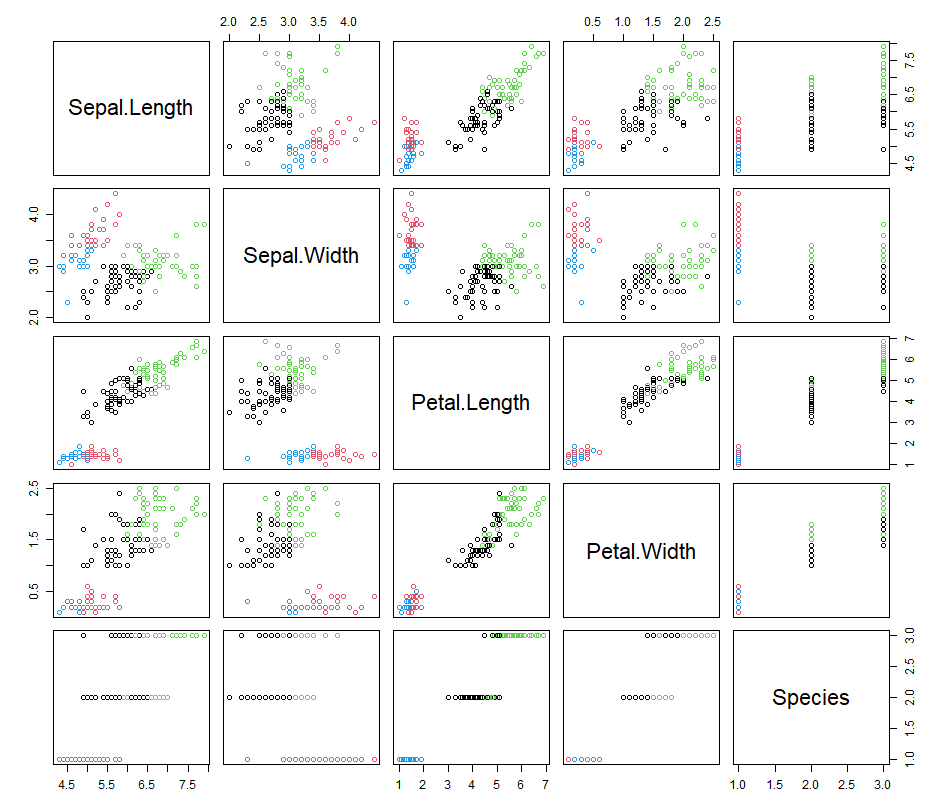
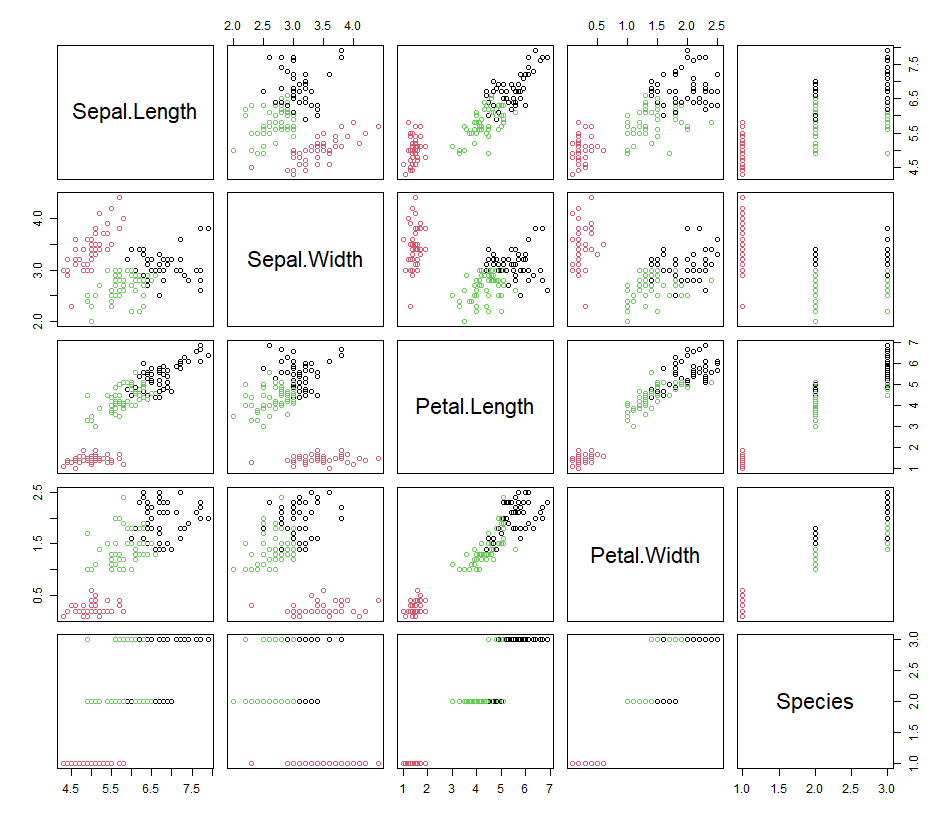


for (i in 1:4) {

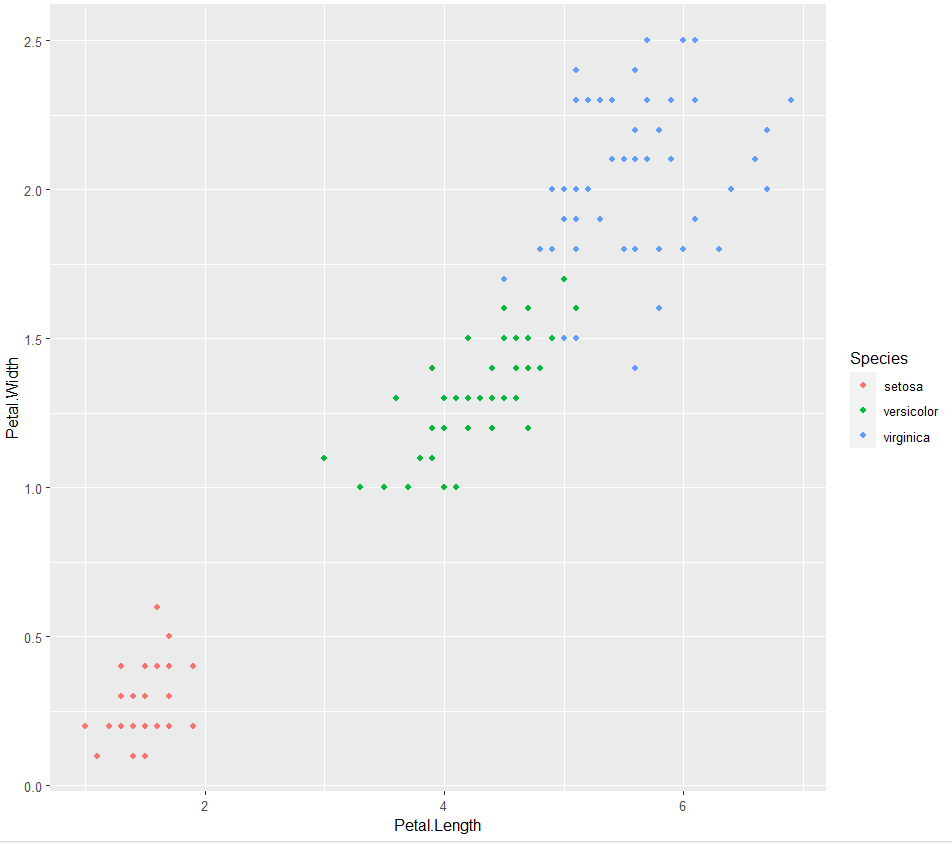
plot(iris,col=k[[i]]$cluster)

}

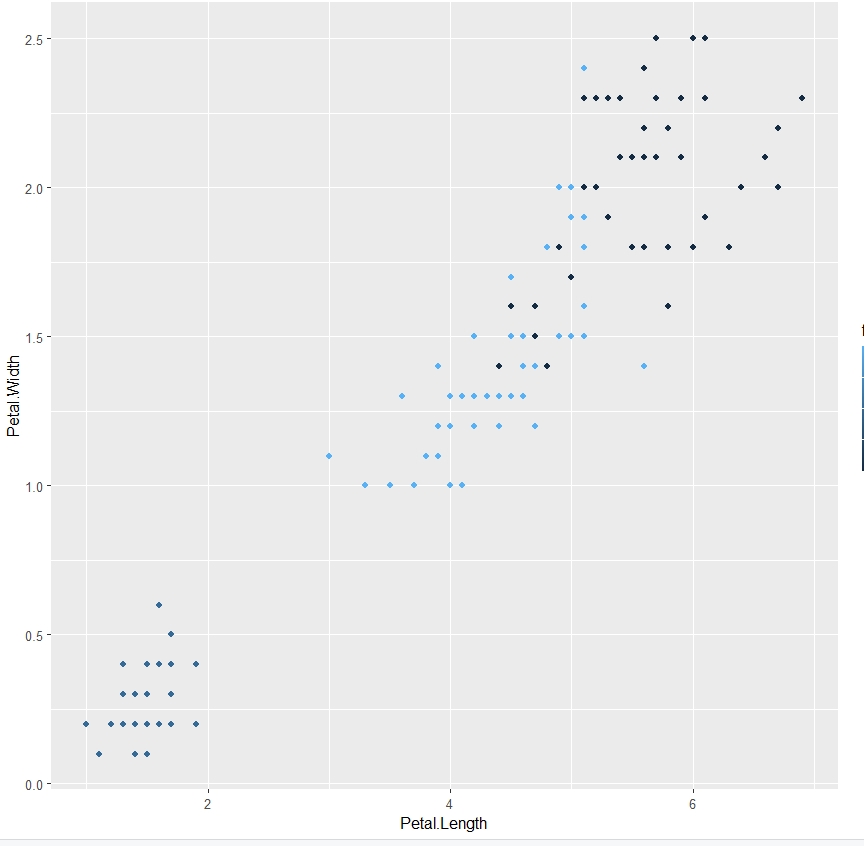




ggplot(iris, aes(Petal.Length, Petal.Width,color = Species)) + geom\_point()



ggplot(iris, aes(Petal.Length, Petal.Width,color = fitK$cluster)) + geom\_point()



**.R FILE :HIERARCHIAL CLUSTERING**

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

d <- dist(irisScaled)

fitH<- hclust(d,"ward.D2") # scaled dataset and algorithm, go through ?hclust

?hclust #to check different algorithms

plot(fitH)

#dendrogram - every observation labelled and clustered with most similar observation,higher up the tree is larger and larger clusters

#upto data scientist to decide where to cut the tree and get cluster

#to visualize cutting of tree

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

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

cluster

plot(iris,col=cluster)

table(iris$Species,cluster)

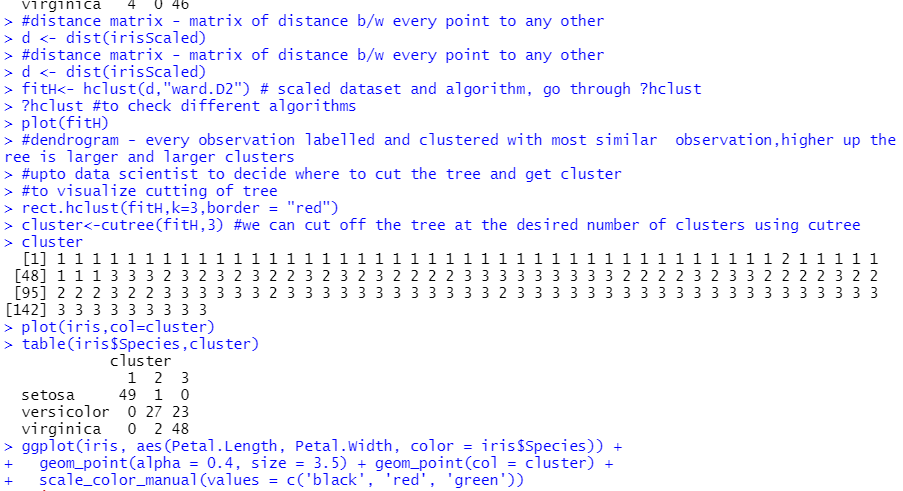
ggplot(iris, aes(Petal.Length, Petal.Width, color = iris$Species)) +

geom\_point(alpha = 0.4, size = 3.5) + geom\_point(col = cluster) +

scale\_color\_manual(values = c('black', 'red', 'green'))

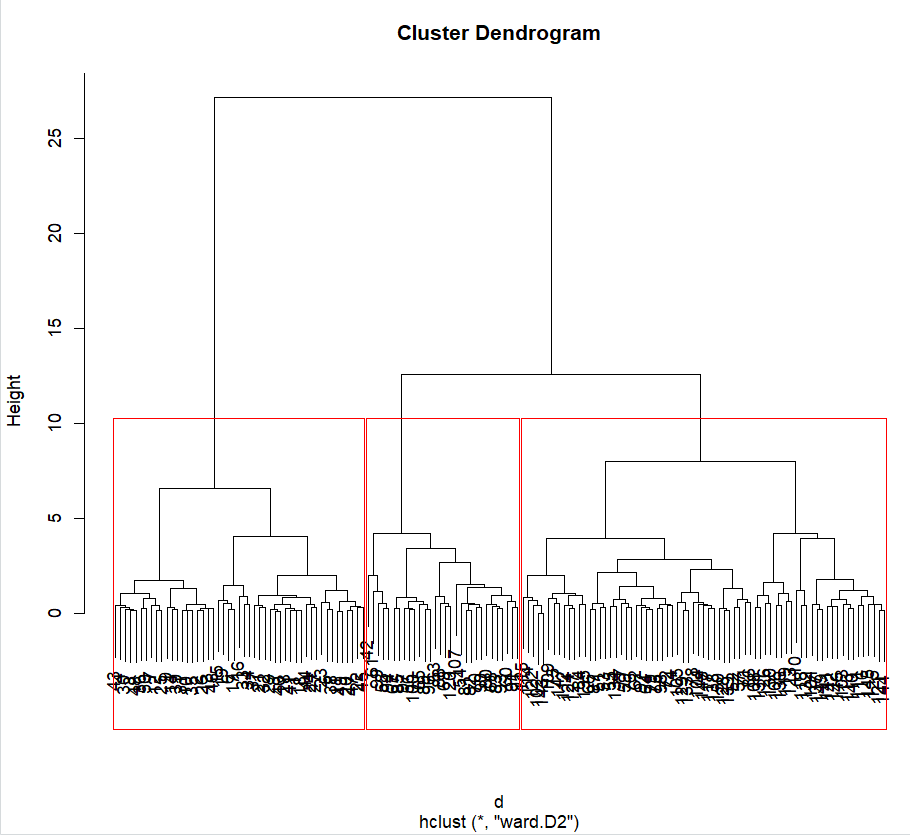
#All the points where the inner color doesn’t match the outer color are the ones whichwere clustered incorrectly.

**OUTPUT : HIERARCHIAL CLUSTERING**

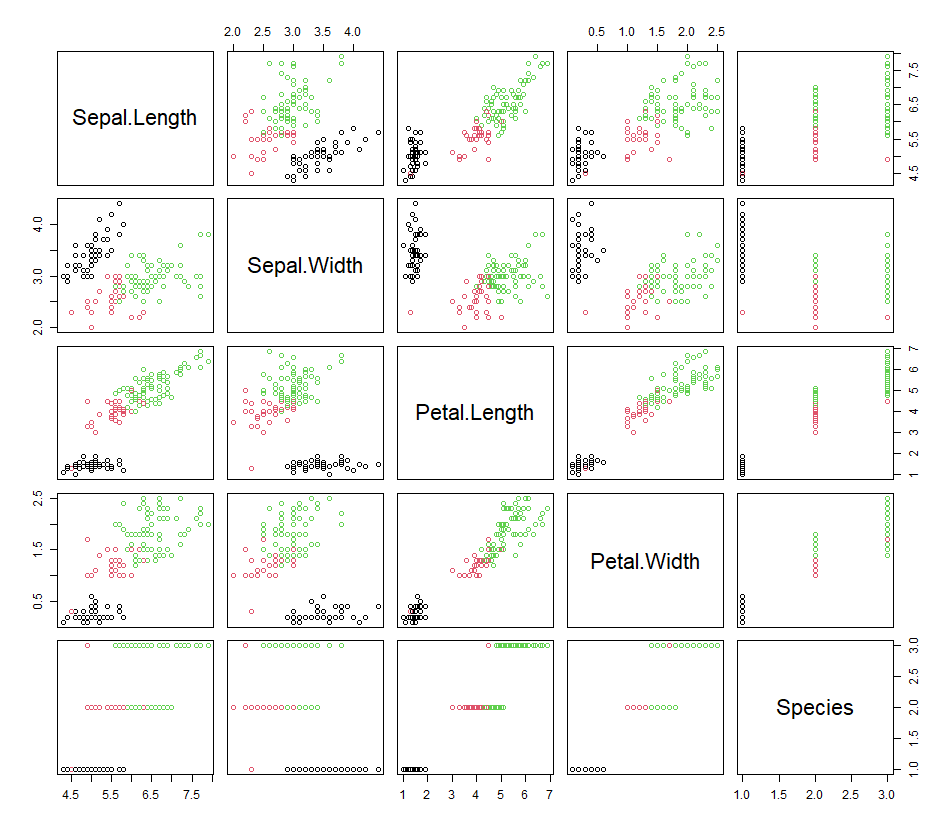


**PLOTS: HIERARCHIAL CLUSTERING**

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



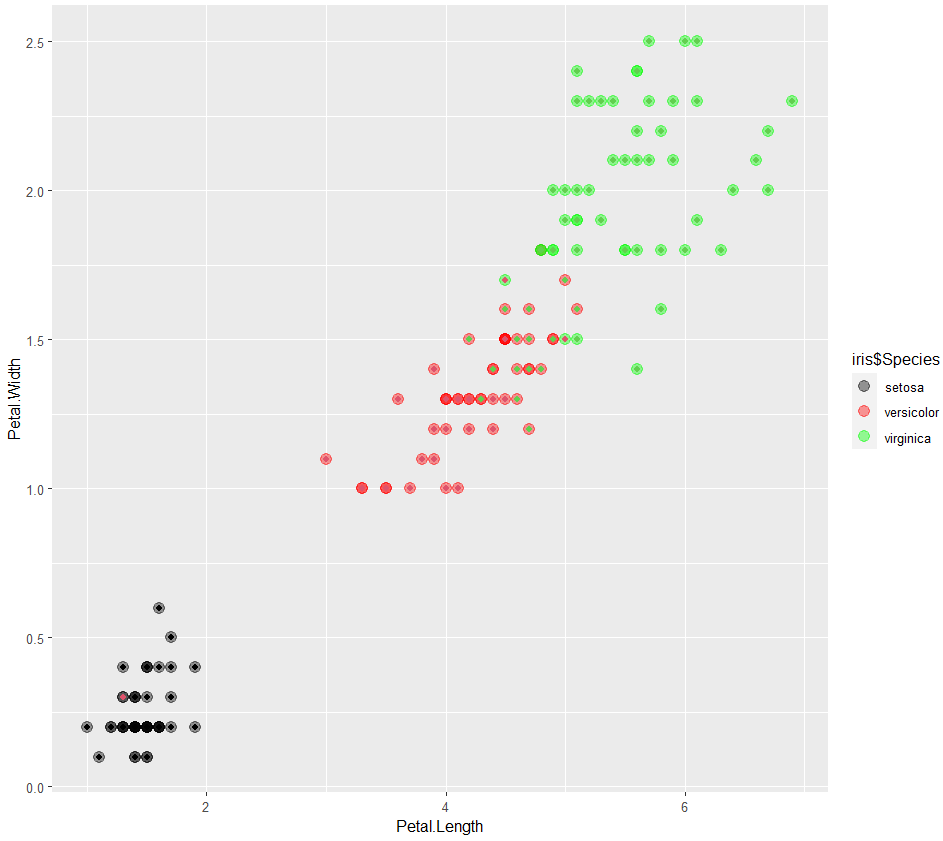
plot(iris,col=cluster)



ggplot(iris, aes(Petal.Length, Petal.Width, color = iris$Species)) +

geom\_point(alpha = 0.4, size = 3.5) + geom\_point(col = cluster) +

scale\_color\_manual(values = c('black', 'red', 'green'))



**.R FILE : DBSCAN**

#DBSCAN #install.packages("dbscan")

library(dbscan)

#Method for determining the optimal eps valueThe method proposed here consists of computing the he k-nearest neighbor distances in a matrix of points.

#The idea is to calculate, the average of the distances of every point to its k nearest neighbors.

#The value of k will be specified by the user and corresponds to MinPts.

#Next, these k-distances are plotted in an ascending order. The aim is to determine the “knee”, which corresponds to the optimal eps parameter.

#A knee corresponds to a threshold where a sharp change occurs along the k-distance curve.

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

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

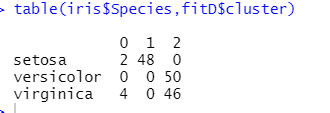
fitD<-dbscan(irisScaled,eps =0.7 ,minPts =5 )

# applying on scaled data and giving eps (knee) value as observed and mentioning minimum points

plot(iris,col=fitD$cluster)

table(iris$Species,fitD$cluster)

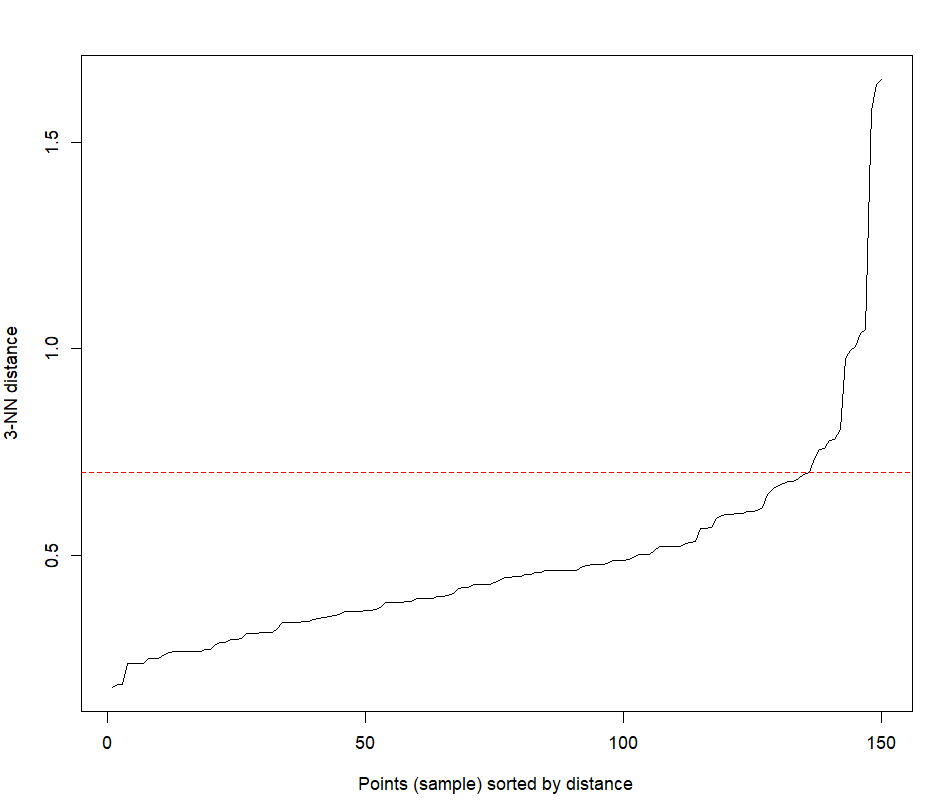
**OUTPUT : DBSCAN**



**PLOTS : DBSCAN**

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

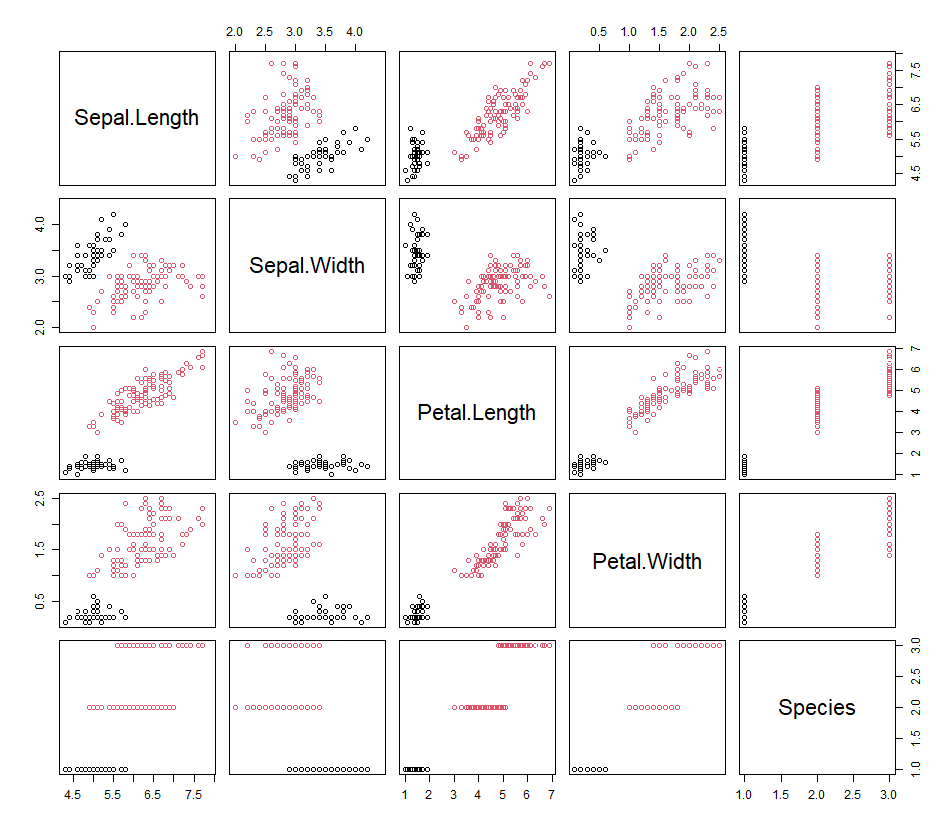
abline(h=0.7,col="red",lty=2)



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

# applying on scaled data and giving eps (knee) value as observed and mentioning minimum points

plot(iris,col=fitD$cluster)



**.HTRU DATASET**

**.R FILE:K MEANS**

df = read.csv(file = "C:/Users/akanksha goel/OneDrive/Documents/program/HTRU\_2.csv",header=FALSE) View(df)

plot(df)

str(df)

dfScaled <- scale(df[,-9])

dfScaled

fitK = kmeans(dfScaled,3)

fitK

fitK$size

fitK$cluster

str(fitK)

plot(df,col=fitK$cluster)

table(df$Class,fitK$cluster)

**.R FILE :HIERARCHIAL CLUSTERING**

 d= dist(dfScaled)

fitH = hclust(d,"ward.D2")

fitH

plot(fitH)

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

cluster= cutree(fitH,3)

cluster

table(df$Class,cluster)

**.R FILE: DBSCAN**

library(dbscan)

kNNdistplot(dfScaled,k=3)

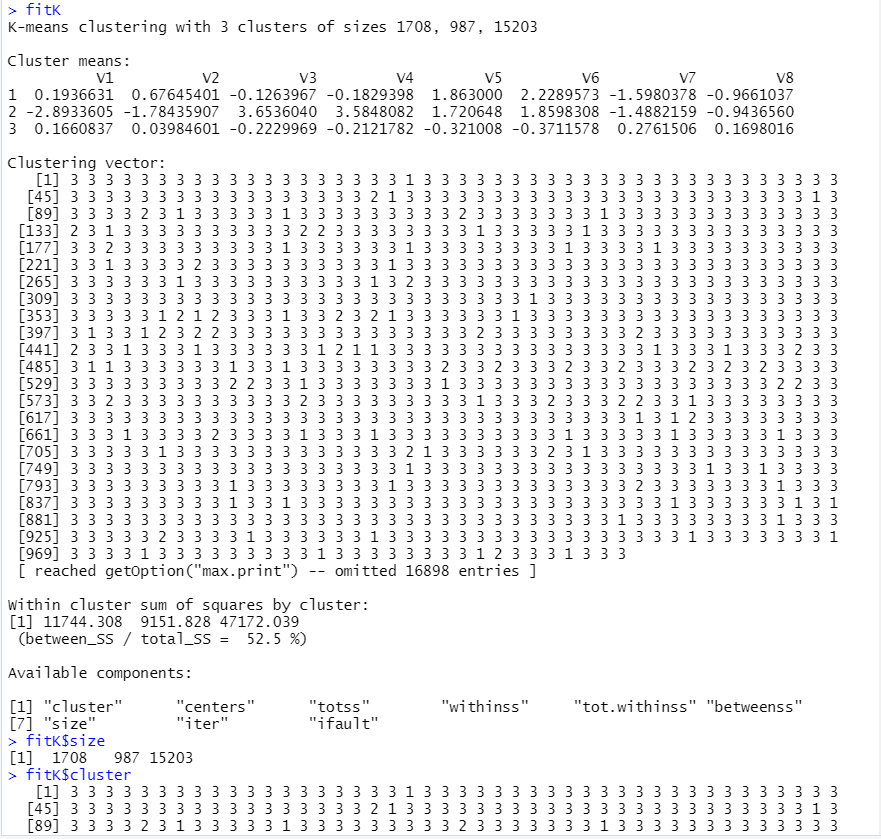
abline(h=0.7,col="red",Ity = 2)

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

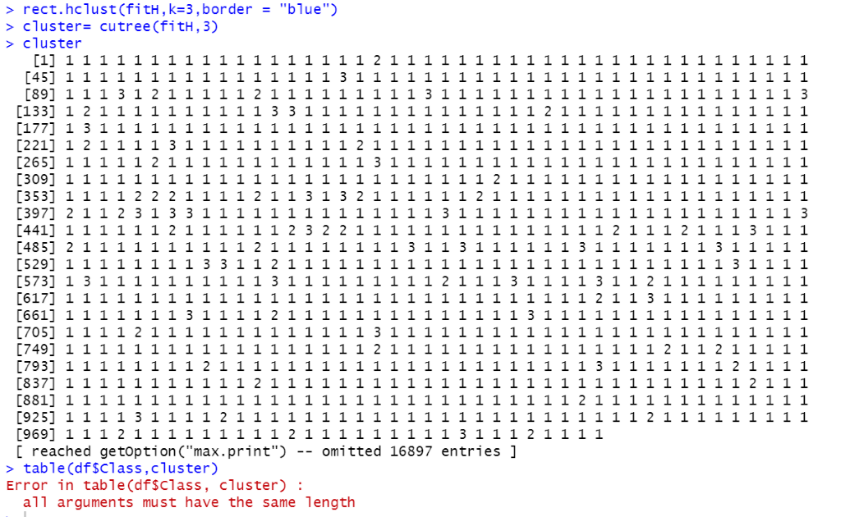
fitD

plot(df,col=fitD$cluster)

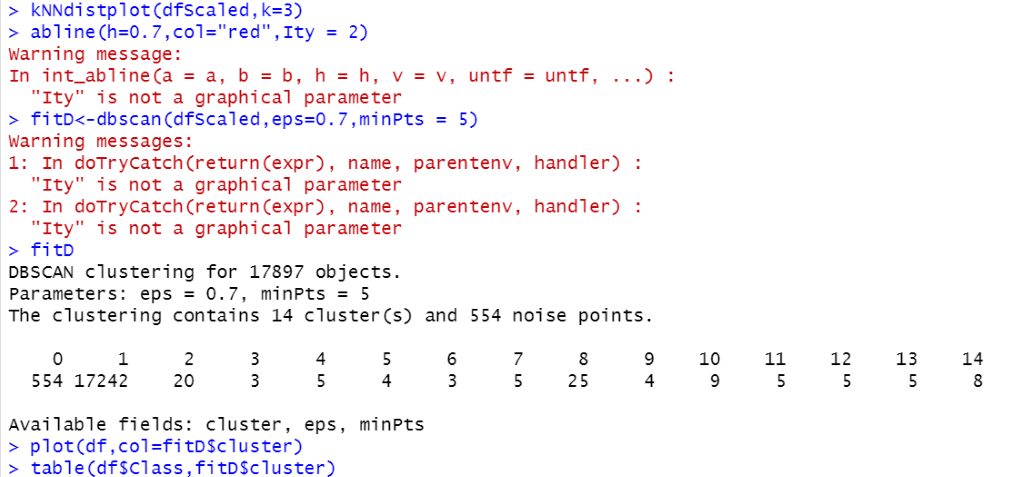
table(df$Class,fitD$cluster)

**OUTPUT: K MEANS**

**OUTPUT : HIERARCHIAL CLUSTERING**



**OUTPUT : DBSCAN**



**PLOTS**

