****

**DATA MINING PRACTICAL FILE**

Submitted to: - Mrs. Preeti Ma’am

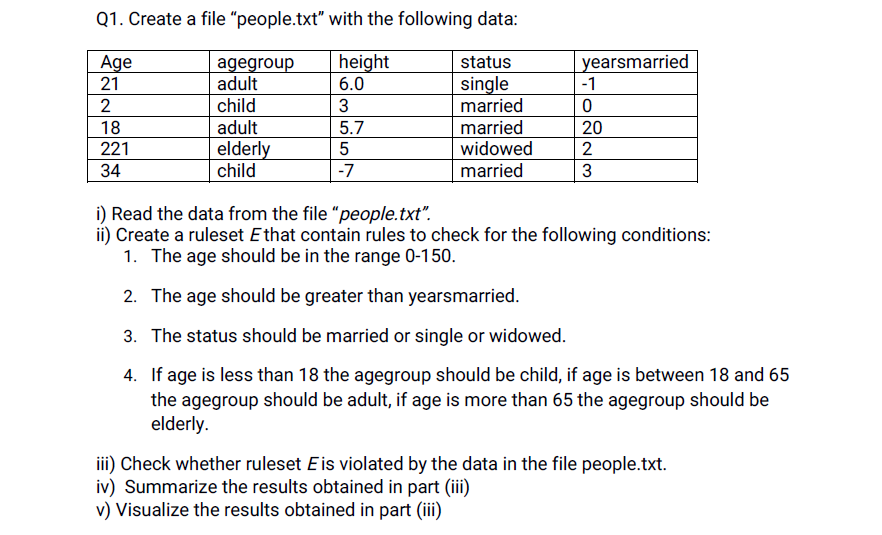
Submitted by: - Harsh Rai 16HCS4420

Mohd Monish 16HCS4431

Divyanshu Tyagi 16HCS4417

Roll Number: -

B.Sc. (Hons) Computer Science



Code: -

library(dplyr)

library(editrules)

library(readxl)

myDataSet =read\_excel ('C:/Users/4/Desktop/AUTUMN/DM/Programs/New folder/Book.xlsx')

print(myDataSet)

E <- editfile ('C:/Users/4/Desktop/AUTUMN/DM/Programs/New folder/rules.txt')

result <- violated Edits (E, myDataSet)

print(result)

summary(result)

plot(E)

OUTPUT: -

A tibble: 5 x 5

age agegroup height status yearsmarried

*<dbl>* *<chr>* *<dbl>* *<chr>* *<dbl>*

1 21 adult  6 single -1

2 2 child 3 married 0

3 18 adult 5.7 married 20

4 221 elderly 5 widowed 2

5 34 child -7 married 3

> E <- editfile('C:/Users/4/Desktop/AUTUMN/DM/Programs/New folder/rules.txt')

> result <- violatedEdits(E,myDataSet)

> print(result)

edit

record num1 num2 num3 dat1

1 FALSE FALSE FALSE FALSE

2 FALSE FALSE FALSE FALSE

3 FALSE FALSE TRUE FALSE

4 FALSE TRUE FALSE FALSE

5 FALSE FALSE FALSE FALSE

> summary(result)

Edit violations, 5 observations, 0 completely missing (0%):

editname freq rel

num2 1 20%

num3 1 20%

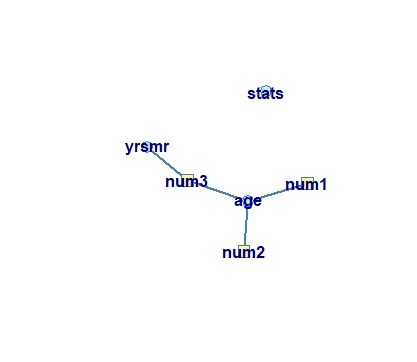
Edit violations per record:

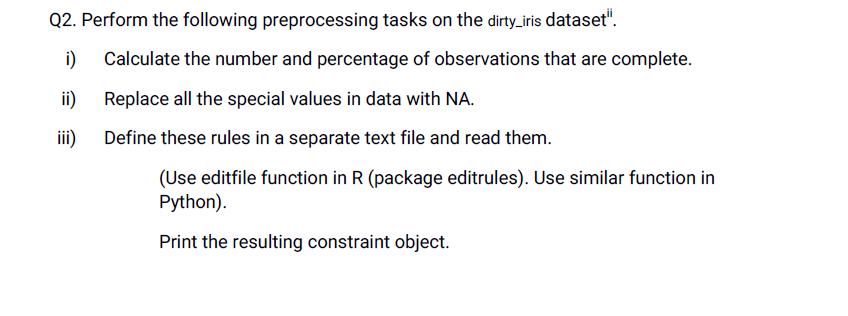
errors freq rel

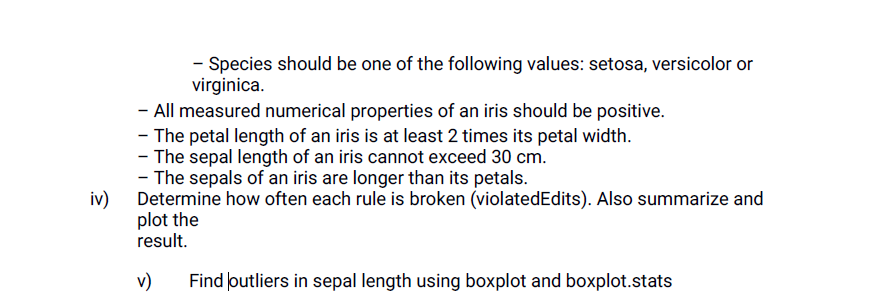
0 3 60%

1 2 40%

> plot(E)



****



Code:

library(editrules)

myDataFrame<-read.csv ('C:/Users/4/Desktop/AUTUMN/DM/Programs/New folder/dirty\_iris.csv')

attach(myDataFrame)

c = sum(complete. cases(myDataFrame))

cat("\n Number of complete observations : ", c)

cat("\n Number of complete observations in % : ",

c/(dim(myDataFrame)[1])\*100)

columnNames=names(myDataFrame)

print(columnNames)

for(columnName in columnNames){

if (class (myDataFrame [, columnName]) =='integer' || class (myDataFrame [, columnName])=='numeric'){

myDataFrame [! is.na (myDataFrame [, columnName]) & (myDataFrame [, columnName]>20 |

myDataFrame [, columnName] <0), columnName] =NA

}

}

myRules<-editfile (file='C:/Users/4/Desktop/AUTUMN/DM/Programs/New folder/ruleset.txt')

E<-violated Edits (myRules, myDataFrame)

print(E)

plot(E)

boxplot stat=boxplot. stats (myDataFrame$Sepal.Length)

out = boxplot\_stat$out

cat ('List of outliers: ‘, out)

boxplot (boxplot stat)

Number of complete observations : 96

Number of complete observations in % : 64

edit

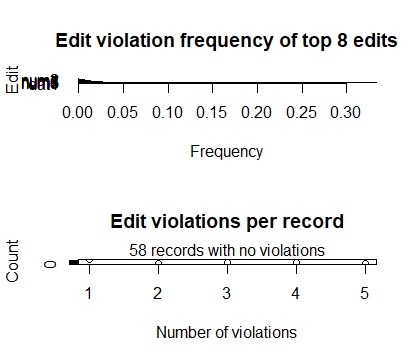
record num1 num2 num3 num4 num5 num6 num7 dat1

1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

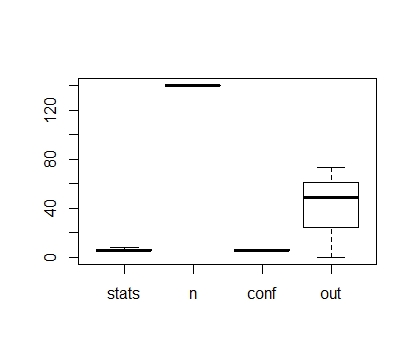
2 FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE

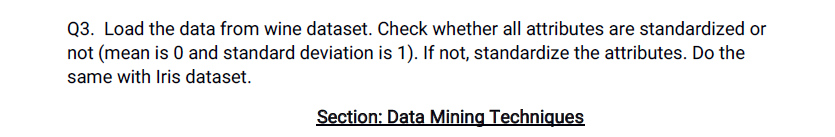
3 FALSE FALSE NA FALSE FALSE FALSE FALSE TRUE

[ reached getOption("max.print") -- omitted other rows ]



List of outliers : 73 0 49





Code: -

library (BBmisc)

wineData=read.csv("C:/Users/4/Desktop/AUTUMN/DM/Programs/wine.csv")

normalizeData=normalize(wineData)

print(normalizeData)

getMean=colMeans(normalizeData)

print(getMean)

irisData=as.data. frame(iris)

normalizeData1=normalize(irisData)

print(normalizeData1)

getMean1=colMeans(normalizeData1)

print(getMean1)

Output: -

print(getMean)

Wine Alcohol Malic.acid Ash

8.194132e-17 -8.594093e-16 -6.734236e-17 8.046486e-16

Acl Mg Phenols Flavanoids

-7.684922e-17 -4.095117e-17 -1.391677e-17 6.950589e-17

Nonflavanoid.phenols Proanth Color.int Hue

-1.041614e-16 -1.223272e-16 3.676054e-17 2.100087e-16

OD Proline

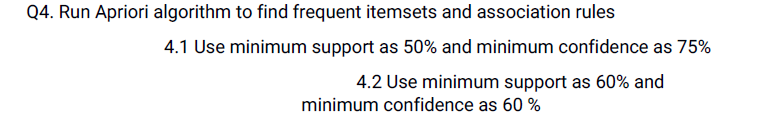
3.009648e-16 -1.037131e-16

#for iris data set

> print(getMean1)

Sepal. Length Sepal. Width Petal. Length Petal. Width

-4.480675e-16 2.035409e-16 -2.844947e-17 -3.714621e-17



library(arules)

MyData = read.csv('C:/Users/4/Desktop/DM/groceries.csv')

#when minsup= 0.5 and confidence= 0.75

print('For minimum support : 0.5 and confidence : 0.75')

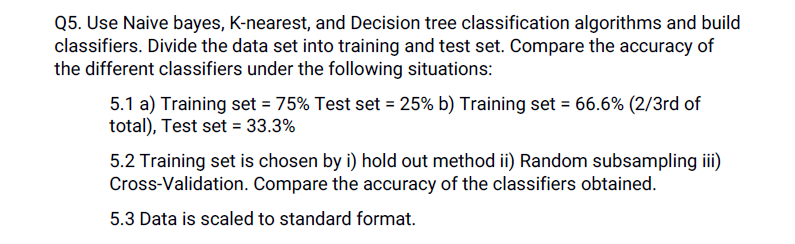
rules= apriori(MyData, parameter= list(supp=0.5,conf=0.75,target="rules") ) #function for generating rules using apriori algorithm

summary(rules)

print('For minimum support : 0.6 and confidence : 0.6')

rules= apriori(MyData, parameter= list(supp=0.6,conf=0.6,target="rules") ) #function for generating rules using apriori algorithm

summary(rules)

  
Code: -

library(e1071)

library(caret)

data(iris)

size=nrow(iris)\*(0.75) #size of the training data is 75%

train=iris[1:size,]

test=iris[-(1:size),] # 25% is the testing data

#Holdout method

model=naiveBayes(Species~.,data=train) # creates a naive bayes mode

pre=predict(model,test[,-5]) # predict the value to the test data

count <- 0

for( i in 1:nrow(test)) {

if(pre[i] == test[i,5])

count= count + 1}

prob <- (count/nrow(test))\* 100

print('Accuracy using holdout method in naive bayes : ')

print(prob)

View(confusionMatrix(pre,test[,5])$table,'Naive Bayes Holdout')

[1] "Accuracy using holdout method in naive bayes : "

[1] 86.84211

#Sampling

t=sample(1:nrow(iris),size)

training=iris[t,]

testing=iris[-(t),]

model=naiveBayes(Species~.,data=training)

pre=predict(model,testing[,-5])

count <- 0

for( i in 1:nrow(testing)) {

if(pre[i] == testing[i,5])

count= count + 1}

prob <- (count/nrow(testing))\* 100

print('Accuracy using random subsampling in NB : ')

print(prob)

View(confusionMatrix(pre,testing[,5])$table,'Naive Bayes Sampling')

[1] "Accuracy using random subsampling in naive bayes: "

[1] 92.10526

#K Fold method

controller= trainControl(method='cv',number=10)

p= train(Species~.,data= training, trControl=controller,method='nb')

pre=predict(p,testing[,-5])

print(p$result)

View(confusionMatrix(pre,testing[,5])$table,'Naive Bayes Kfold')

usekernel fL adjust Accuracy Kappa AccuracySD KappaSD

1 FALSE 0 1 0.9643939 0.9464198 0.04601922 0.06925908

2 TRUE 0 1 0.9553030 0.9328395 0.04716885 0.07088081

library(e1071)

library(rpart)

library(rpart.plot)

library(caret)

data(iris)

size=nrow(iris)\*(0.75)

training=iris[1:size,]

testing=iris[-(1:size),]

#HoldOut Method

mod=rpart(Species~.,data= training)

rpart.plot(mod)

pre=predict(mod,testing,type='class')

count <- 0

for( i in 1:nrow(testing)) {

if(pre[i] == testing[i,5])

count= count + 1}

prob <- (count/nrow(testing))\* 100

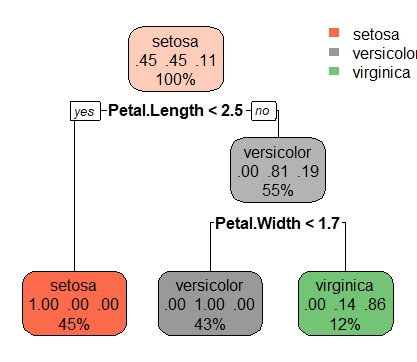
print('Accuracy using holdout method in Decision Tree: ')

print(prob)

View(confusionMatrix(pre,testing[,5])$table,'Decision Tree Holdout')

[1] "Accuracy using holdout method in Decision Tree: "

[1] 89.47368



#Sampling

t=sample(1:nrow(iris),size)

training=iris[t,]

testing=iris[-(t),]

model=rpart(Species~.,data=training)

rpart.plot(model)

pre=predict(model,testing,type='class')

print(table(pre))

count <- 0

for( i in 1:nrow(testing)) {

if(pre[i] == testing[i,5])

count= count + 1}

prob <- (count/nrow(testing))\* 100

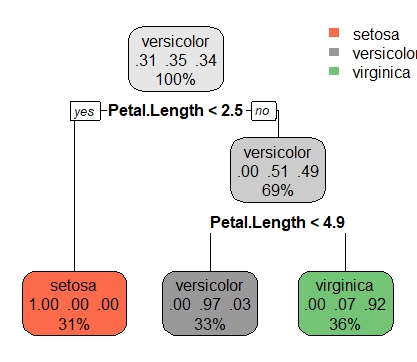
print('Accuracy using random subsampling in Decision Tree : ')

print(prob)

View(confusionMatrix(pre,testing[,5])$table,'Decision Tree Sampling')

[1] "Accuracy using random subsampling in Decision Tree : "

[1] 92.10526



#K Fold method

controller= trainControl(method='cv',number=10)

p= train(Species~.,data= training, trControl=controller,method='nb')

pre=predict(p,testinging[,-5])

print(p$result)

View(confusionMatrix(pre,testing[,5])$table,'Naive Bayes Kfold')

usekernel fL adjust Accuracy Kappa AccuracySD KappaSD

1 FALSE 0 1 0.9568182 0.93500 0.04560161 0.06866788

2 TRUE 0 1 0.9560606 0.93375 0.04639872 0.06998264

library(e1071)

library(class)

data(iris)

size=nrow(iris)\*(0.75)

training=iris[1:size,]

testing=iris[-(1:size),]

#HoldOut Method

mod=knn(training[,-5], testing[,-5], training$Species, k=3, prob= TRUE)

print(mod)

count <- 0

for( i in 1:nrow(testing)) {

if(mod[i] == testing[i,5])

count= count + 1}

prob <- (count/nrow(testing))\* 100

print('Accuracy using holdout method in knn: ')

print(prob)

View(confusionMatrix(pre,testing[,5])$table,'KNN Holdout')

[1] "Accuracy using holdout method in knn: "

[1] 71.05263

#Sampling

t=sample(1:nrow(iris),size)

training=iris[t,]

testing=iris[-(t),]

mod=knn(training[,-5], testing[,-5], training$Species, k=3, prob= TRUE)

count <- 0

for( i in 1:nrow(testing)) {

if(mod[i] == testing[i,5])

count= count + 1}

prob <- (count/nrow(testing))\* 100

print('Accuracy using random subsampling in knn : ')

print(prob)

View(confusionMatrix(pre,testing[,5])$table,'KNN Sampling')

[1] "Accuracy using random subsampling in knn : "

[1] 94.73684

#K Fold method

controller= trainControl(method='cv',number=10)

p= train(Species~.,data= training, trControl=controller,method='knn')

pre=predict(p,testing[,-5])

print(p$result)

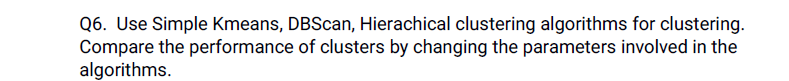
View(confusionMatrix(pre,testing[,5])$table,'Naive Bayes Kfold')

k Accuracy Kappa AccuracySD KappaSD

1 5 0.9734848 0.9601698 0.04274321 0.06421239

2 7 0.9825758 0.9739198 0.03677690 0.05504091

3 9 0.9825758 0.9739198 0.03677690 0.05504091



library(dbscan)

data(iris)

#kmeans

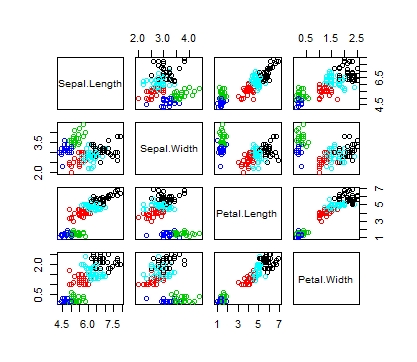
#centers = 5

k= kmeans(iris[,-5],5) #kmeans function creates clusters of the data provided and the number of centroids

cat('SSE of the model with 5 : ',k$tot.withinss,'\n')

plot(iris[,-5],col= k$cluster)

of the model with 5 : 49.85942 SSE of the model with 5 : 49.85942



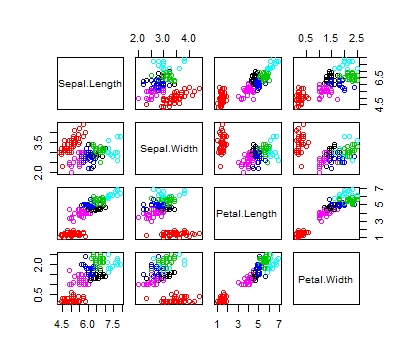
#centers = 6

k= kmeans(iris[,-5],6) #kmeans function creates clusters of the data provided and the number of centroids

cat('SSE of the model with 6 : ',k$tot.withinss,'\n')

plot(iris[,-5],col= k$cluster)

SSE of the model with 6 : 42.16294



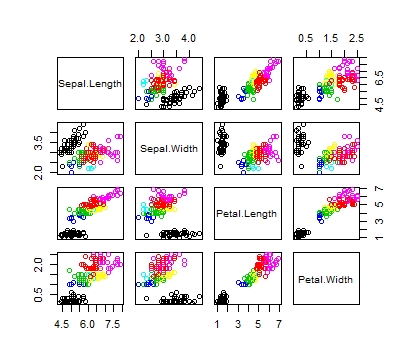
#centers = 7

k= kmeans(iris[,-5],7) #kmeans function creates clusters of the data provided and the number of centroids

cat('SSE of the model with 7 : ',k$tot.withinss,'\n')

plot(iris[,-5],col= k$cluster)

SSE of the model with 7 : 38.11226



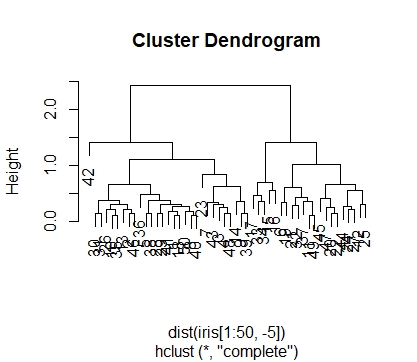
#hierarchical method

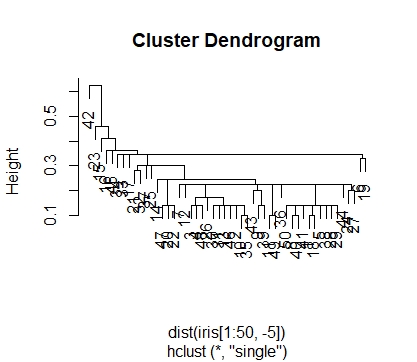
h1 = hclust(dist(iris[1:50,-5]),method = 'complete') #Produces the hierarchical clusters of the data on the basis of their minimum distance

h2 = hclust(dist(iris[1:50,-5]),method = 'single') #Produces the hierarchical clusters of the data on the basis of the minimum distance distance

plot(h1)

plot(h2)





#DBscan method

db = dbscan(iris[,-5], eps = .5, minPts = 5) #It produces density based clustering where radius is .5 and minimum closest points are 5

plot(iris[,-5],col=db$cluster)

