Name : Gaurang Tyagi

Roll Number : 4065

Paper : Data Mining

PRACTICAL FILE

Ques1.) Q1. Create a file “people.txt” with the following data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Age | Agegroup | Height | Status | yearsmarried |
| 21 | Adult | 6.0 | single | -1 |
| 2 | Child | 3 | Married | 0 |
| 18 | Adult | 5.7 | married | 20 |
| 221 | Elderly | 5 | widowed | 2 |
| 34 | Child | -7 | Married | 3 |

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)

Ans.) CODE🡪

library("editrules")

people <- read.csv('C:/Users/COMP43/Desktop/GaurangTyagi/Rstudio/people.txt',sep='\t')

E <- editset(expression(

Age > 0, Age < 150,

Age > yearsmarried,

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

if(Age < 18) agegroup %in% 'child',

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

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

))

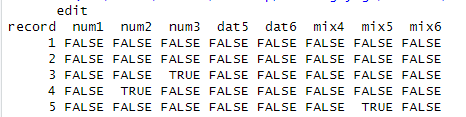
V <- violatedEdits(E,people)

summary(V)

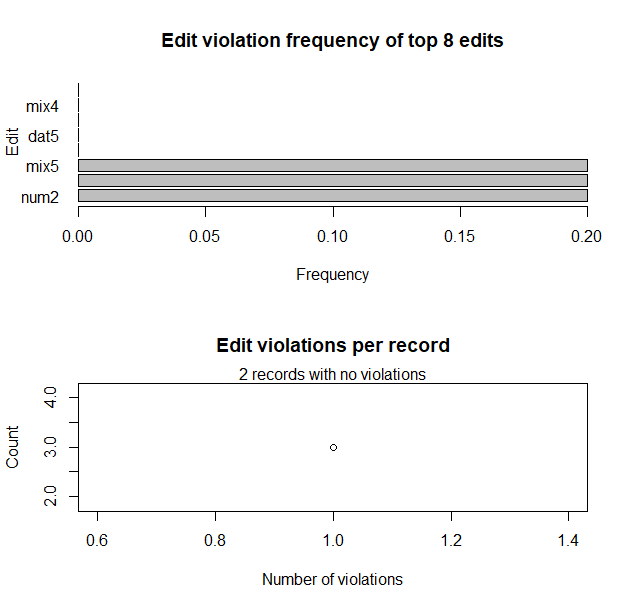
plot(V)

OUTPUT🡪

Summary:



Plot:



Ques2.) 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

Ans.) CODE🡪

library(editrules)

data <- read.csv("C:/sem6/vs study/r/dirty-iris-data.csv")

total\_count = nrow(data)

print(paste("Total count : ",total\_count,sep = ""))

# que (i)

complete\_count = nrow(data[complete.cases(data),])

print(paste(

"Complete values count : ",complete\_count,

" (",(complete\_count\*100/total\_count),"%)",

sep = "")

)

# que (ii)

data <- replace(data,data<0,NA)

data <- replace(data,data==Inf,NA)

# que (iii)

rules <- editfile("C:/sem6/vs study/r/rules2.txt")

# que (iv)

res <- violatedEdits(rules,data)

summary(res)

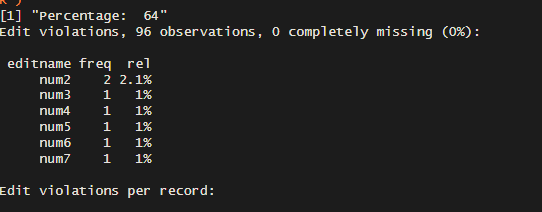
plot(res)

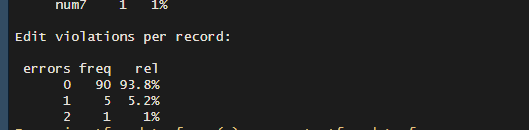
# que (v)

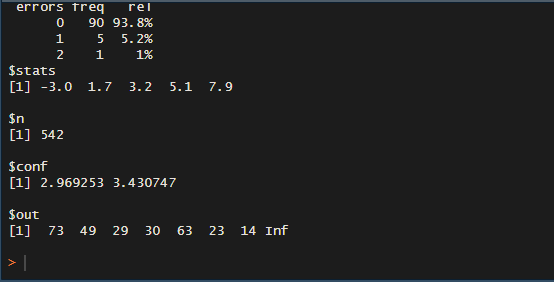
data <- replace(data,data==NA,0)

boxplot(data[1:4])

OUTPUT🡪







Ques3.) 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.

Ans.) CODE🡪

import pandas as pd

from sklearn.datasets import load\_iris

from sklearn.preprocessing import StandardScaler

pd.options.display.float\_format = '{:.2f}'.format

data = load\_iris()

iris = pd.DataFrame(data.data,columns=data.feature\_names)

iris

wine = pd.read\_csv("wine.csv")

wine = wine.drop('Wine',axis='columns')

wine

scaler = StandardScaler()

wine = pd.DataFrame(scaler.fit\_transform(wine),columns=wine.columns)

sum\_wine = wine.describe()

sum\_wine

iris = pd.DataFrame(scaler.fit\_transform(iris),columns=iris.columns)

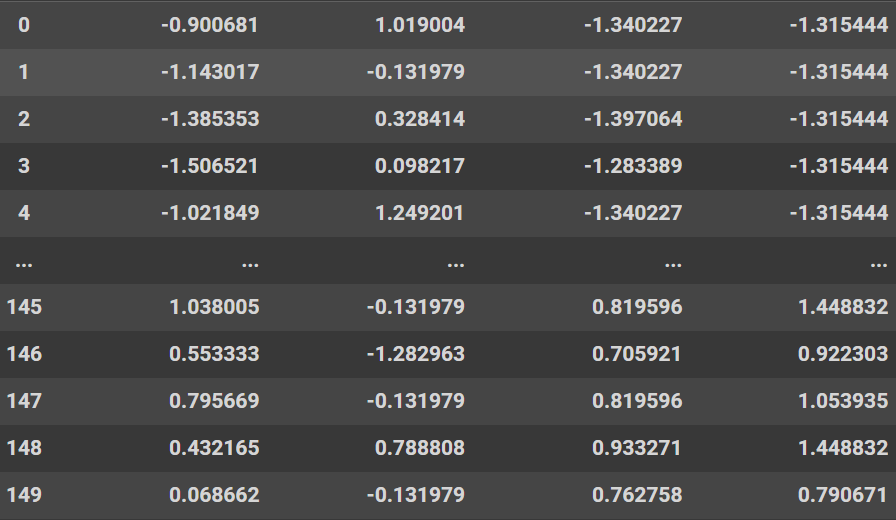
sum\_iris = iris.describe()

sum\_iris

pd.options.display.float\_format = None

iris

OUTPUT🡪



Ques4.) 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 %

Ans.) CODE🡪

df = pd.read\_csv("GroceryStoreDataSet.csv",names=['products'])

df.head()

df= df['products'].str.split(",")

df.head()

enc = TransactionEncoder()

\_ = enc.fit\_transform(df)

df = pd.DataFrame(\_,columns=enc.columns\_)

df.head()

# min\_support - 50% & min\_confidence - 75%

df = apriori(df,min\_support=.5,use\_colnames=True,verbose=1)

df

df\_assoc\_rules = association\_rules(df,metric='confidence',min\_threshold=.75)

df\_assoc\_rules

# min-support 60% & min-confidence 60%

df = pd.DataFrame(\_,columns=enc.columns\_)

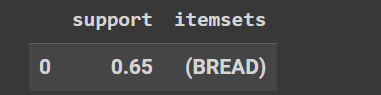
df = apriori(df,min\_support=.6,use\_colnames=True,verbose=1)

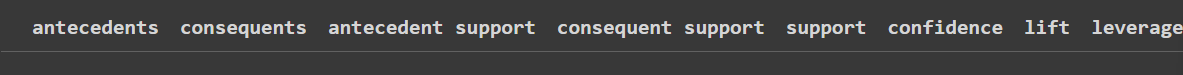
df

df\_assoc\_rules = association\_rules(df,metric='confidence',min\_threshold=.6)

df\_assoc\_rules

OUTPUT🡪





Ques5.) 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.

Ans.) CODE🡪

report\_arr = []

for i,j in zip(train\_splits\_arr\_leave,test\_splits\_arr\_leave):

fit = NB.fit(data.iloc[i,1:-2],data.iloc[i,-1])

y\_pred = fit.predict(data.iloc[j,1:-2])

report = classification\_report(data.iloc[j,-1],y\_pred,output\_dict=True)

report\_arr.append(pd.DataFrame(report))

sum\_acc=sum\_pre=sum\_rec=sum\_f1 = 0

for i in report\_arr:

sum\_acc+= i.iloc[0,2]

sum\_pre+= i.iloc[0,3]

sum\_rec+= i.iloc[1,3]

sum\_f1+= i.iloc[2,3]

sum\_acc/len(report\_arr)

sum\_pre/len(report\_arr)

sum\_f1/len(report\_arr)

# Decision Tree

dtree = DecisionTreeClassifier()

fit = dtree.fit(x\_train,y\_train)

y\_pred = fit.predict(x\_test)

report = classification\_report(y\_test,y\_pred,output\_dict=True)

pd.DataFrame(report)

report\_arr = list()

for i,j in zip(train\_splits\_arr\_kfold,test\_splits\_arr\_kfold):

fit = dtree.fit(data.iloc[i,1:-2],data.iloc[i,-1])

y\_pred = fit.predict(data.iloc[j,1:-2])

report = classification\_report(data.iloc[j,-1],y\_pred,output\_dict=True)

report = pd.DataFrame(report)

report\_arr.append(report)

len(report\_arr)

sum\_acc=sum\_pre=sum\_rec=sum\_f1 = 0

for i in report\_arr:

sum\_acc+= i.iloc[0,3]

sum\_pre+= i.iloc[0,4]

sum\_rec+= i.iloc[1,4]

sum\_f1+= i.iloc[2,4]

sum\_acc/5,sum\_pre/5,sum\_f1/5

report\_arr = []

for i,j in zip(train\_splits\_arr\_leave,test\_splits\_arr\_leave):

fit = NB.fit(data.iloc[i,1:-2],data.iloc[i,-1])

y\_pred = fit.predict(data.iloc[j,1:-2])

report = classification\_report(data.iloc[j,-1],y\_pred,output\_dict=True)

report\_arr.append(pd.DataFrame(report))

sum\_acc=sum\_pre=sum\_rec=sum\_f1 = 0

for i in report\_arr:

sum\_acc+= i.iloc[0,2]

sum\_pre+= i.iloc[0,3]

sum\_rec+= i.iloc[1,3]

sum\_f1+= i.iloc[2,3]

sum\_acc/len(report\_arr),sum\_pre/len(report\_arr),sum\_f1/len(report\_arr)

# KNN

knn = KNeighborsClassifier()

fit = knn.fit(x\_train,y\_train)

y\_pred = fit.predict(x\_test)

report = classification\_report(y\_test,y\_pred,output\_dict=True)

report = pd.DataFrame(report)

report

report\_arr = list()

for i,j in zip(train\_splits\_arr\_kfold,test\_splits\_arr\_kfold):

fit = dtree.fit(data.iloc[i,1:-2],data.iloc[i,-1])

y\_pred = fit.predict(data.iloc[j,1:-2])

report = classification\_report(data.iloc[j,-1],y\_pred,output\_dict=True)

report = pd.DataFrame(report)

report\_arr.append(report)

len(report\_arr)

sum\_acc=sum\_pre=sum\_rec=sum\_f1 = 0

for i in report\_arr:

sum\_acc+= i.iloc[0,3]

sum\_pre+= i.iloc[0,4]

sum\_rec+= i.iloc[1,4]

sum\_f1+= i.iloc[2,4]

sum\_acc/5,sum\_pre/5,sum\_f1/5

report\_arr = []

for i,j in zip(train\_splits\_arr\_leave,test\_splits\_arr\_leave):

fit = NB.fit(data.iloc[i,1:-2],data.iloc[i,-1])

y\_pred = fit.predict(data.iloc[j,1:-2])

report = classification\_report(data.iloc[j,-1],y\_pred,output\_dict=True)

report\_arr.append(pd.DataFrame(report))

sum\_acc=sum\_pre=sum\_rec=sum\_f1 = 0

for i in report\_arr:

sum\_acc+= i.iloc[0,2]

sum\_pre+= i.iloc[0,3]

sum\_rec+= i.iloc[1,3]

sum\_f1+= i.iloc[2,3]

sum\_acc/len(report\_arr),sum\_pre/len(report\_arr),sum\_f1/len(report\_arr)

Ques6.) Use Simple Kmeans, DBScan, Hierachical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

Ans.) CODE🡪

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy.cluster.hierarchy import linkage,dendrogram

from sklearn.metrics import accuracy\_score

from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN

from sklearn.model\_selection import train\_test\_split,cross\_validate

data = pd.read\_csv('HTRU\_2.csv')

data.head()

X,Y = data.iloc[:,:-1],data.iloc[:,-1]

x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,Y,test\_size=.30,random\_state=0)

sse = dict()

for i in range(2,10):

KM = KMeans(n\_clusters=i)

fit = KM.fit\_transform(X,Y)

sse[i] = KM.inertia\_

plt.plot(sse.keys(),sse.values(),marker='o')

plt.xlabel('clusters')

plt.ylabel('sse')

plt.show()

KM = KMeans(n\_clusters=5)

fit = KM.fit\_transform(X,Y)

data['cluster'] = KM.labels\_

data.head()

HC = AgglomerativeClustering(n\_clusters=5,compute\_distances=True)

fit = HC.fit(X.iloc[:100],Y)

l = linkage(HC.children\_,'single')

figure = plt.figure(figsize=(20,20))

dn = dendrogram(l)

plt.savefig(fname='dendrogram')

x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,Y,test\_size=.3)

import numpy as np

db = DBSCAN(eps=15.5, min\_samples=5).fit(data)

core\_samples\_mask = np.zeros\_like(db.labels\_, dtype=bool)

core\_samples\_mask[db.core\_sample\_indices\_] = True

labels = pd.DataFrame(db.labels\_,columns=['Cluster ID'])

result = pd.concat((data,labels), axis=1)

labels.value\_counts()

OUTPUT🡪

