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→

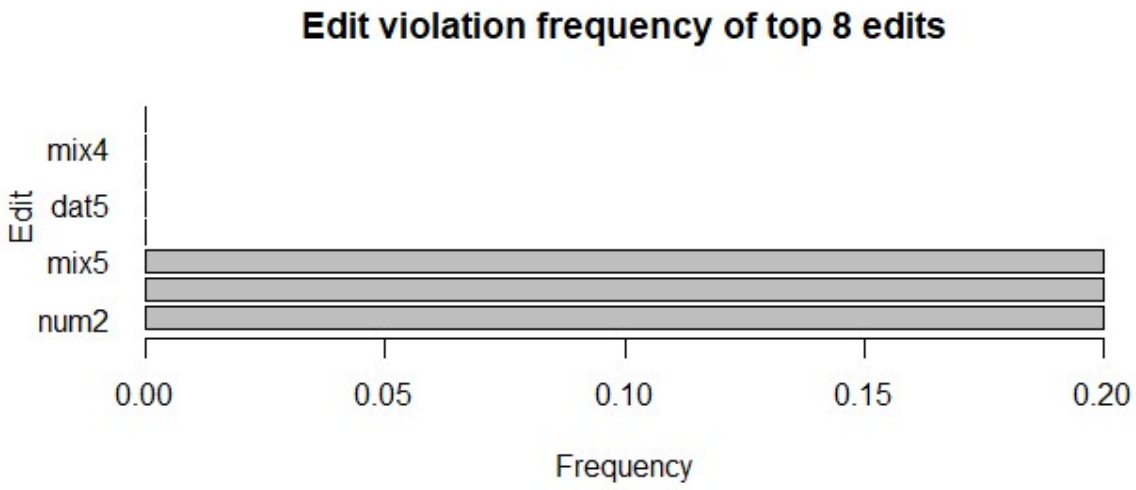
```
library("editrules")

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)
```

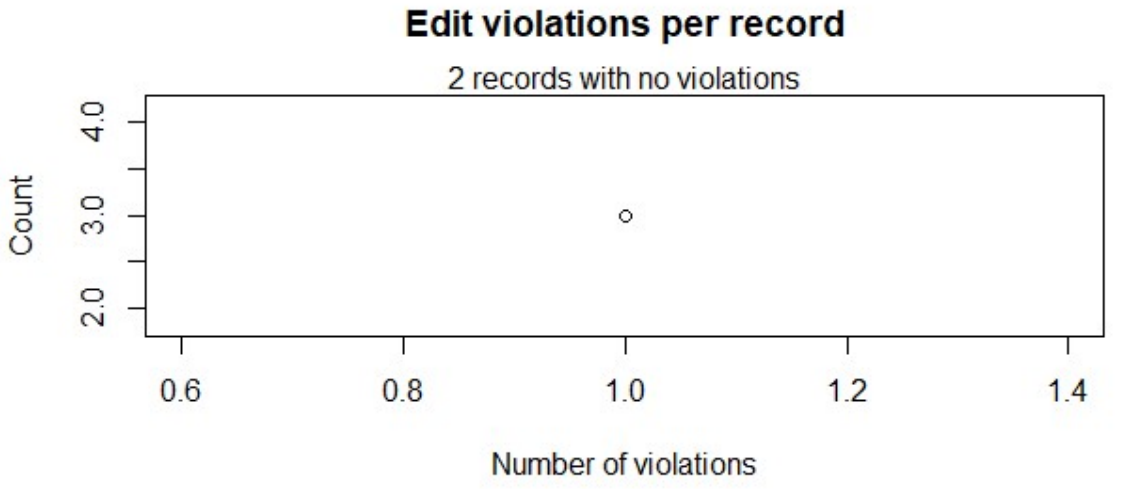
OUTPUT→

Summary:

```
       edit
record  num1  num2  num3  dat5  dat6  mix4  mix5  mix6
     1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
     2 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
     3 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
     4 FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
     5 FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
```

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)

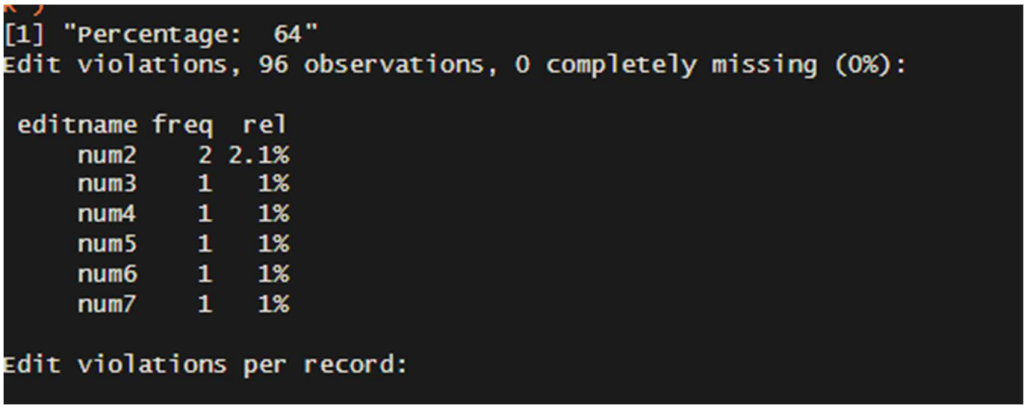
plot(res)

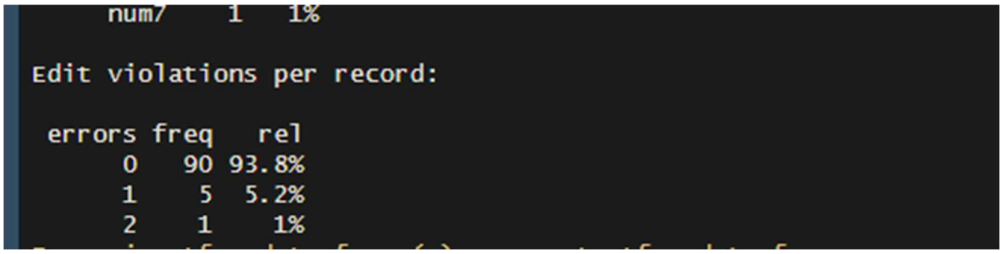
que (v)

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

boxplot(data[1:4])

OUTPUT→





```
 errors freq   rel
      0   90 93.8%
      1    5  5.2%
      2    1    1%
$stats
[1] -3.0  1.7  3.2  5.1  7.9

$n
[1] 542

$conf
[1] 2.969253 3.430747

$out
[1]  73  49  29  30  63  23  14 Inf

> |
```

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→

0	-0.900681	1.019004	-1.340227	-1.315444
1	-1.143017	-0.131979	-1.340227	-1.315444
2	-1.385353	0.328414	-1.397064	-1.315444
3	-1.506521	0.098217	-1.283389	-1.315444
4	-1.021849	1.249201	-1.340227	-1.315444
...	...	...	...	...
145	1.038005	-0.131979	0.819596	1.448832
146	0.553333	-1.282963	0.705921	0.922303
147	0.795669	-0.131979	0.819596	1.053935
148	0.432165	0.788808	0.933271	1.448832
149	0.068662	-0.131979	0.762758	0.790671

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→

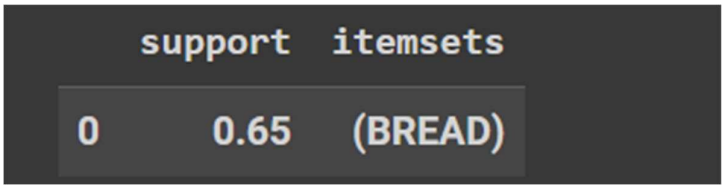
```
df = pd.read_csv("GroceryStoreDataSet.csv",names=['products'])

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→



	support	itemsets
0	0.65	(BREAD)

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage

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→

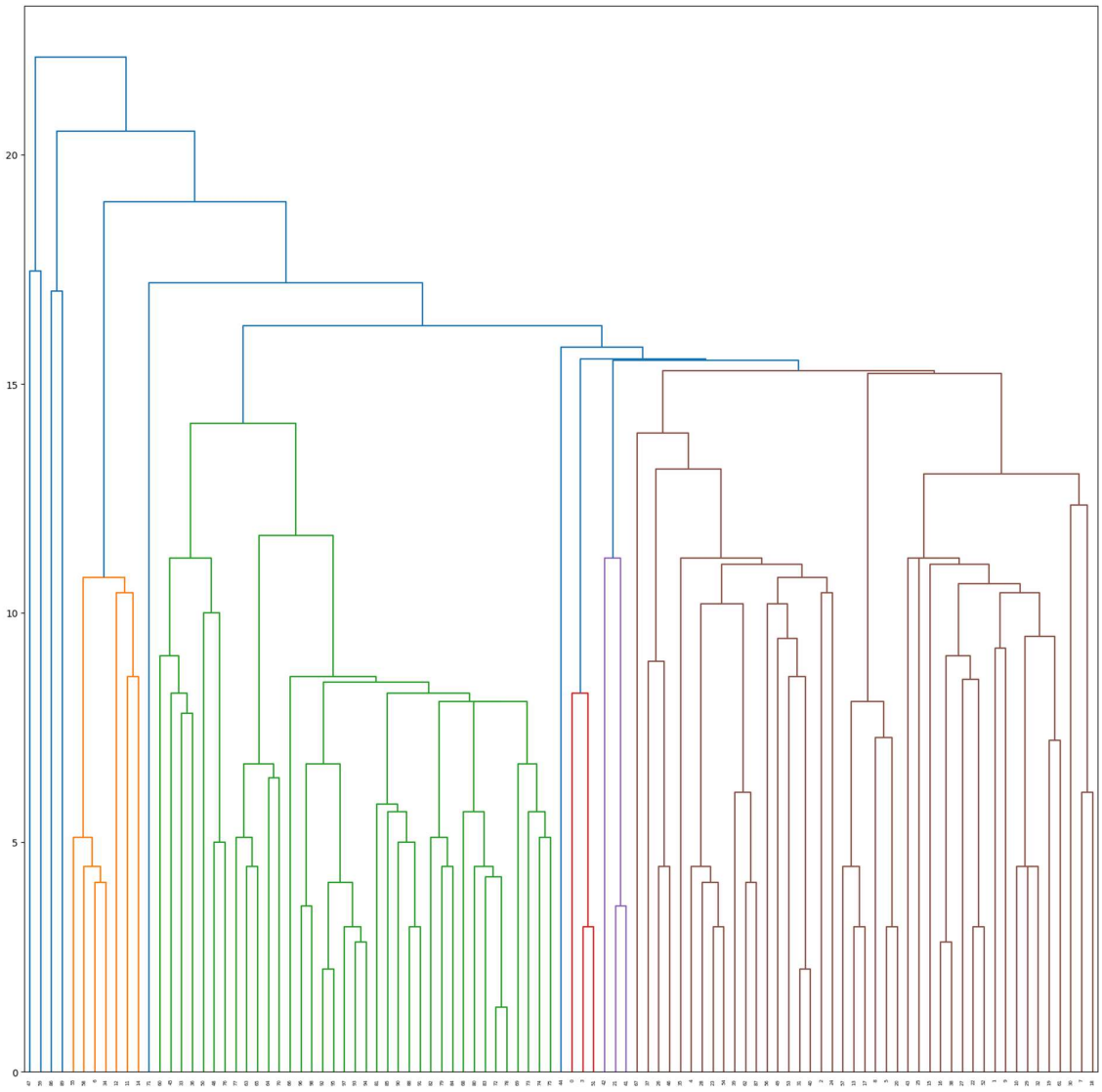
```
import pandas as pd

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→

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	cluster
0	140.562500	55.683782	-0.234571	-0.699648	3.199833	19.110426	7.975532	74.242225	0	0
1	102.507812	58.882430	0.465318	-0.515088	1.677258	14.860146	10.576487	127.393580	0	1
2	103.015625	39.341649	0.323328	1.051164	3.121237	21.744669	7.735822	63.171909	0	0
3	136.750000	57.178449	-0.068415	-0.636238	3.642977	20.959280	6.896499	53.593661	0	0
4	88.726562	40.672225	0.600866	1.123492	1.178930	11.468720	14.269573	252.567306	0	2


20
15
10
5
0