INDEX

BCA 507(C):- PRACTICAL ON DATA MINING USING PYTHON

SR.NO	TITLE	REMARK	SIGN
1.	Calculate the mean and standard deviation.		
2.	Read the CSV file.		
3.	Perform data filtering and calculate aggregate statistics.		
4.	Calculate total sales by month.		
5.	Implement the Clustering using K-means.		
6.	Classification using Random Forcst.		
7.	Regression Analysis using Linear Regression.		
8.	Association Rule Mining using Apriori.		
9.	Visualize the result of the clustering and compare.		
10.	Visualize the correlation matrix using a pseudocolor plot.		
11.	Use of degrees distribtution of a network.		
12.	Graph visualization of a network using maximum , minimum , median , first quartile and third quartile.		

Pract no 1: Calculate the mean and standard deviation.

```
import numpy as np
data=[10,20,30,40,50,60]
mean=np.mean(data)
print(mean)
std_dev=np.std(data)
print(std_dev)
```

Output : 35.0

17.07825127659933

Pract 2: Read the CSV file.

```
import pandas as pd

import statistics

cf=pd.read_csv('E:\screentime_analysis.csv')

print(cf)

mv=cf[['Noti','time']].mean()

mv1=cf[['Noti','time']].mode()

print("__________________________________________")

print("mean :",mv)

print("__________________________________________")

print("mode :",mv1)

print("__________________________________________")

sd=cf[['Noti','time']].std()

print( "standard deviation :",sd)
```

Output:

```
          Date        App  Usage (minutes)   Noti  time
0     8/7/2024  Instagram             81.0   24.0  57.0
1     8/8/2024  Instagram             90.0   30.0  53.0
2    8/10/2024   WhatsApp             63.0   73.0  88.0
3    8/22/2024   WhatsApp             51.0  145.0  81.0
4    8/17/2024   WhatsApp             68.0  107.0  81.0
..         ...        ...              ...    ...   ...
63         NaN        NaN              NaN    NaN   NaN
64         NaN        NaN              NaN    NaN   NaN
65         NaN        NaN              NaN    NaN   NaN
66         NaN        NaN              NaN    NaN   NaN
67         NaN        NaN              NaN    NaN   NaN

[68 rows x 5 columns]
__________________________________________
mean : Noti     34.52
time     32.92
dtype: float64
__________________________________________
mode :     Noti  time
0    2.0    1.0
1    3.0    8.0
2    NaN   10.0
3    NaN   81.0
4    NaN   88.0
__________________________________________
standard deviation : Noti     45.158334
time     33.465694
dtype: float64
```

Pract no 3: Perform data filtering and calculate aggregate statistics.

import pandas as pd

data={'name':['alice','bob','charlie','david','eve'], 'age':[20,22,32,21,19], 'salary':[3000,4000,2000,5000,3500]}

df=pd.DataFrame(data)

f_d=df[df['age']>20]

ave_sal=f_d['salary'].mean()

ave_sal1=f_d['salary'].sum()

print(f_d)

print('_________________')print(f'Averege salary of employees older than 25:{ave_sal,ave_sal1}')

Output: name age salary

1	bob	22	4000
2	charlie	32	2000
3	david	21	500

____

Averege salary of employees older than 25:(3666.6666666666665, 11000)

Pract no 4 : Calculate total sales by month.

```
month_data= { 'jan':1000,'feb':300,'march':100,'apl':350,  'may':750,

'june':400, 'jully':500,'aug':300,'sep':200,'oct':400,'nove':700,
'dec':800,

}

total_sale=sum(month_data.values())

for month,sales in month_data.items():

print(f'Sales in {month}: {sales}')

print('__________________')

print("TOTLE SALE CALCULATE FOR YEAR:",total_sale)
```

Output :

Sales in jan: 1000

Sales in feb: 300

Sales in march: 100

Sales in apl: 350

Sales in may: 750

Sales in june: 400

Sales in jully: 500

Sales in aug: 300

Sales in sep: 200

Sales in oct: 400

Sales in nove: 700

Sales in dec: 800

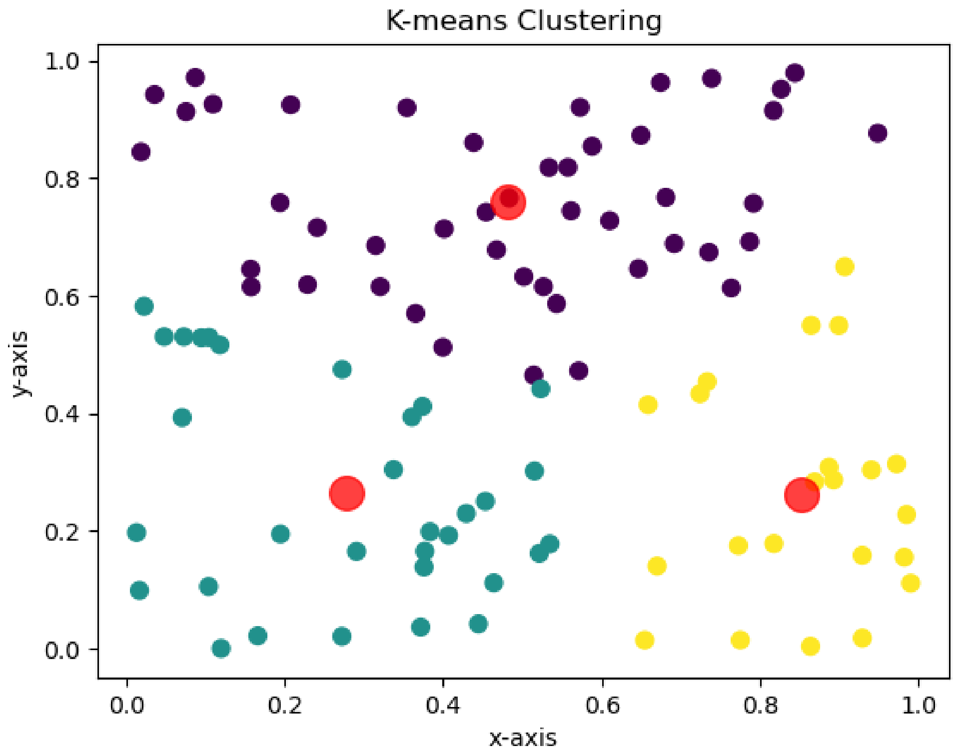
____

TOTLE SALE CALCULATE FOR YEAR: 5800

Pract no 5: Implement the clustering using K-means.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
np.random.rand(0)
x=np.random.rand(100,2)
kmeans=KMeans(n_clusters=3)
kmeans.fit(x)
center=kmeans.cluster_centers_
labels=kmeans.labels_
plt.scatter(x[:,0],x[:,1],c=labels,s=50,cmap='viridis')
plt.scatter(center[:,0],center[:,1],c='red',s=200,alpha=0.75)
plt.title('K-means Clustering')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

Output :


K-means Clustering
1.0
0.8
0.6
0.4
0.2
0.0
y-axis
0.0
0.2
0.4
0.6
0.8
1.0
x-axis

Pract no 6: Classification using Random Forest.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
iris=load_iris()
X=iris.data
Y=iris.target
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
model=RandomForestClassifier(n_estimators=100,random_state=42)
model.fit(X_train,Y_train)
Y_pred=model.predict(X_test)
accuracy=accuracy_score(Y_test,Y_pred)
confusion=confusion_matrix(Y_test,Y_pred)
report=classification_report(Y_test,Y_pred)
print(f'Accuracy:{accuracy:.2f}')
print('Confusion Matrix:')
print(confusion)
print('Classification Report')
print(report)
```

Output :

Accuracy:1.00

Confusion Matrix:

[[10 0 0]

 [0 9 0]

 [0 0 11]]

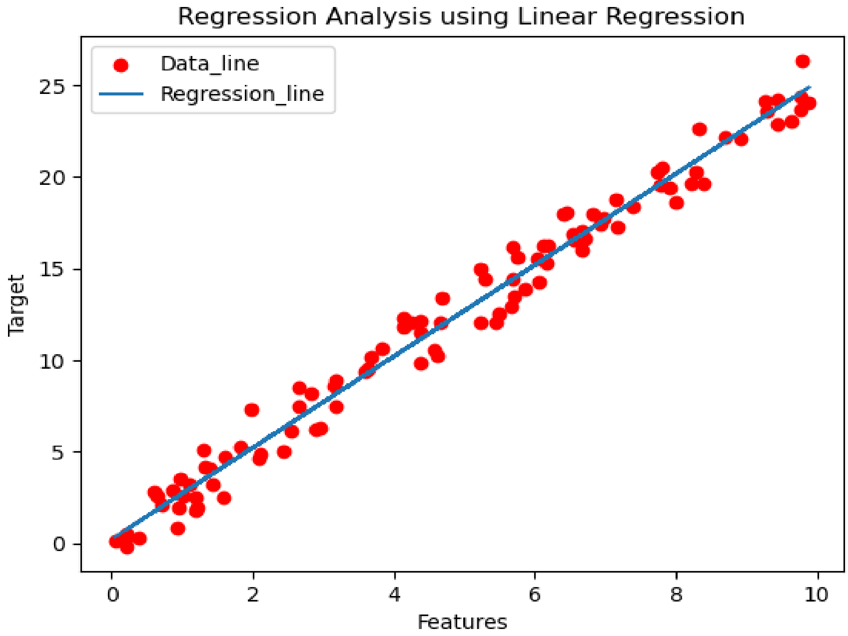
Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Pract no 7 : Regression Analysis using Linear Regression.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
np.random.seed(0)
x=np.random.rand(100,1)*10
y=2.5*x+np.random.randn(100,1)
data=pd.DataFrame(np.hstack((x,y)),
columns=['Feature','Target'])
model=LinearRegression()
model.fit(data[['Feature']],data[['Target']])
y_pred=model.predict(data[['Feature']])
plt.scatter(data['Feature'],data['Target'],color='red',label='Data_line')
plt.plot(data['Feature'],y_pred,label='Regression_line')
plt.xlabel('Features')
plt.ylabel('Target')
plt.title('Regression Analysis using Linear Regression')
plt.legend()
plt.show()
```

Output :



Regression Analysis using Linear Regression
Data_line
Regression_line
Target
Features
0
5
10
15
20
25
0
2
4
6
8
10

Pract no 8: Association rule mining using apriori.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
d_S=[['Milk','Bread'],
    ['Bread','Beer','Eggs'],
    ['Milk','Beer','Cola'],
    ['Bread','Milk','cola']]
encoder=TransactionEncoder()
onehot=encoder.fit(d_S).transform(d_S)
df=pd.DataFrame(onehot,columns=encoder.columns_)
f_i=apriori(df,min_support=0.4,use_colnames=True)
print(f_i)
rules = association_rules(f_i, num_itemsets=len(f_i), metric="lift", min_threshold=1)
print("\nAssociation Rules:")
print(rules)
```

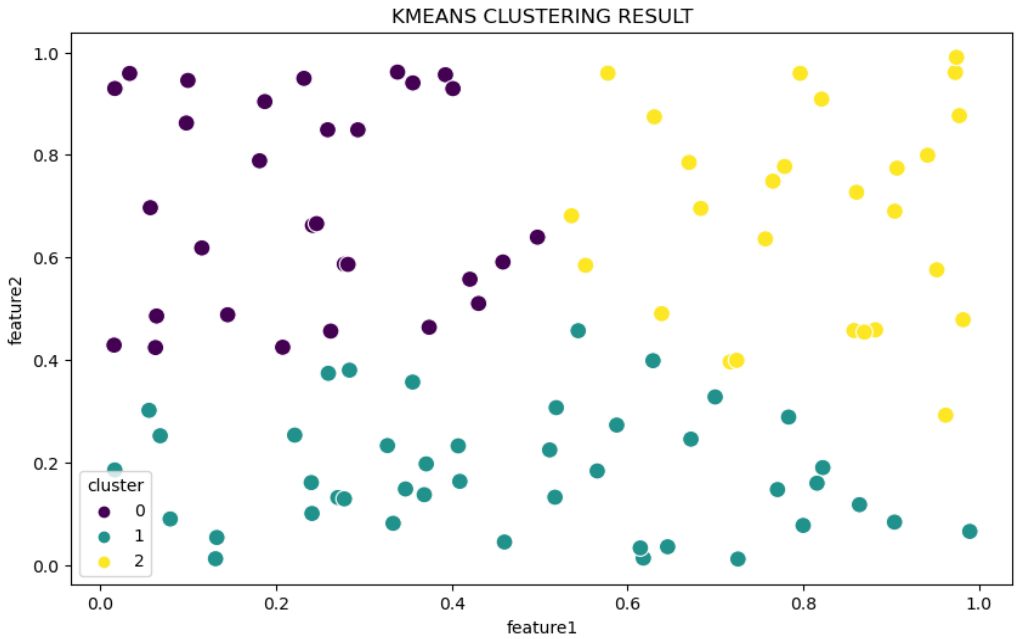
Output:-

```
   support       itemsets
0     0.50         (Beer)
1     0.75        (Bread)
2     0.75         (Milk)
3     0.50  (Bread, Milk)

Association Rules:
Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, representativity, leverage, conviction, zhangs_met
ric, jaccard, certainty, kulczynski]
Index: []
```

Pract no 9: Visualize the result of the clustering and compare.

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

from sklearn.cluster import KMeans

data=np.random.rand(100,2)

df=pd.DataFrame(data,

columns=['feature1','feature2'])

kmeans=KMeans(n_clusters=3)

df['cluster']=kmeans.fit_predict(df[['feature1','feature2']])

plt.figure(figsize=(10,6))

sns.scatterplot(data=df,x='feature1',y='feature2',palette='viridis',hue
='cluster',s=100)

plt.title('KMEANS CLUSTERING RESULT')

plt.legend(title='cluster')

plt.show()
```

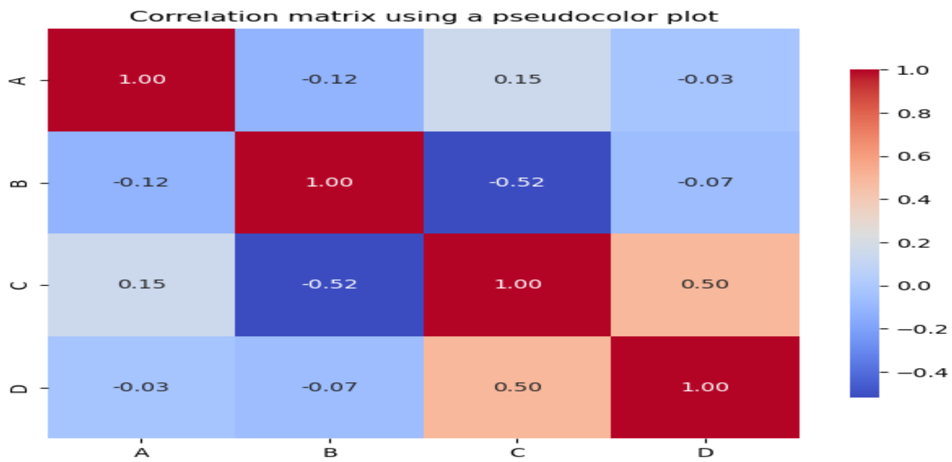
Output :


KMEANS CLUSTERING RESULT
feature1
feature2
cluster
0
1
2
0.0
0.2
0.4
0.6
0.8
1.0

Pract no 10: Visualize the Correlation matrix using a pseudocolor plot.

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

data={ 'A':np.random.rand(10), 'B':np.random.rand(10),
'C':np.random.rand(10), 'D':np.random.rand(10)}

df=pd.DataFrame(data)

corr=df.corr()

plt.figure(figsize=(8,6))

sns.heatmap(corr,annot=True,fmt='.2f',cmap='coolwarm',square=True,

cbar_kws={"shrink":.8})

print(data)

plt.title('Correlation matrix using a pseudocolor plot')

plt.show()
```

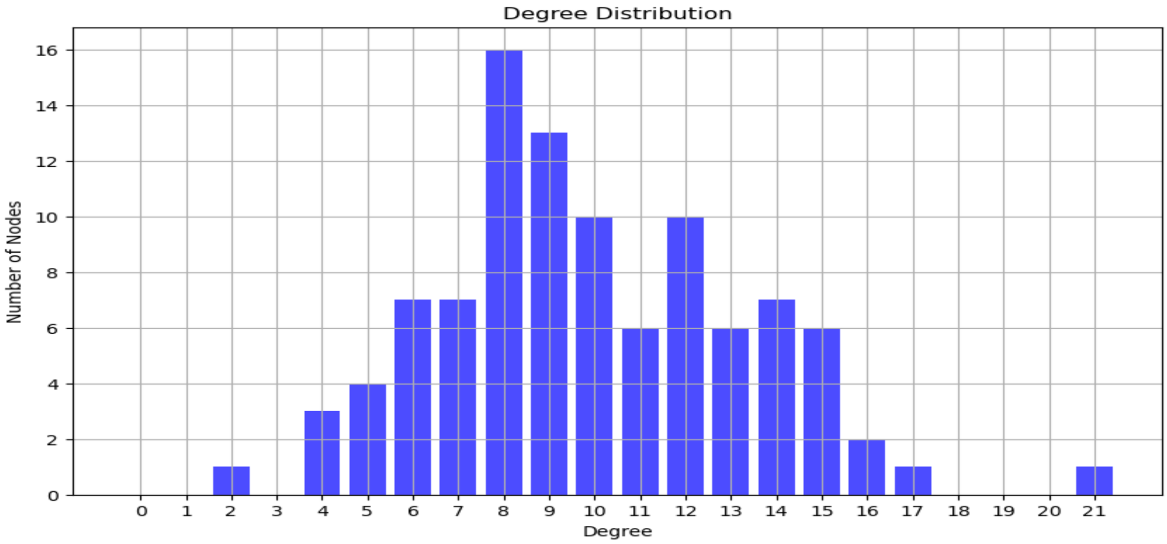
Output :



Practical 11: Use of degree distributon of a network .

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
n=100
p=0.1
g=nx.erdos_renyi_graph(n,p)
degree_sequence=[d for n, d in g.degree()]
degree_count=np.bincount(degree_sequence)
degrees=np.arange(len(degree_count))
plt.figure(figsize=(10,6))
plt.bar(degrees,degree_count,width=0.8,color='b',alpha=0.7)
plt.title('Degree Distribution')
plt.xlabel('Degree')
plt.ylabel('Number of Nodes')
plt.xticks(degrees)
plt.grid()
plt.show()
```

Output :



Degree Distribution
Number of Nodes
Degree
0
2
4
6
8
10
12
14
16
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

Practical 12 :Graph visulization of a network using maximum,minimum,median,first qurtile and third qurtile.

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
n=100
p=0.1
G=nx.erdos_renyi_graph(n,p)
degree_sequence=[d for n,d in G.degree()]
degree_min=np.min(degree_sequence)
degree_max=np.min(degree_sequence)
degree_median=np.median(degree_sequence)
degree_q1=np.percentile(degree_sequence,25)
degree_q3=np.percentile(degree_sequence,75)
print(f"Minimum Degree:{degree_min}")
print(f"Maximum Degree:{degree_max}")
print(f"Median Degree:{degree_median}")
print(f"First Quartile(Q1):{degree_q1}")
print(f"Third Quartile(Q3):{degree_q3}")
plt.figure(figsize=(12,8))
pos=nx.spring_layout(G)
nx.draw(G,pos,node_size=50,with_labels=False,alpha=0.7)
plt.title('Network Visualization with Degree Statistics')
plt.text(-
1.5,1.5,f'Min:{degree_min}\nMax:{degree_max}\nMedian:{degree_
```

```
median}\nQ1:{degree_q1}\nQ3:{degree_q3}',fontsize=10,bbox=dict(facecolor='white',alpha=0.5))

plt.show()
```

Output :

Minimum Degree:4

Maximum Degree:4

Median Degree:10.0

First Quartile(Q1):8.0

Third Quartile(Q3):11.0