**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()

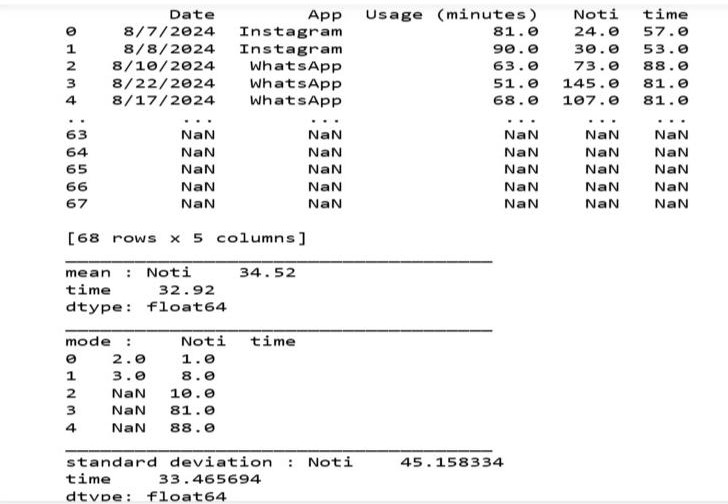
print(" ") print("mean :",mv)

print(" ") print("mode :",mv1)

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

print( "standard deviation :",sd)

**Output:**

****

# 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

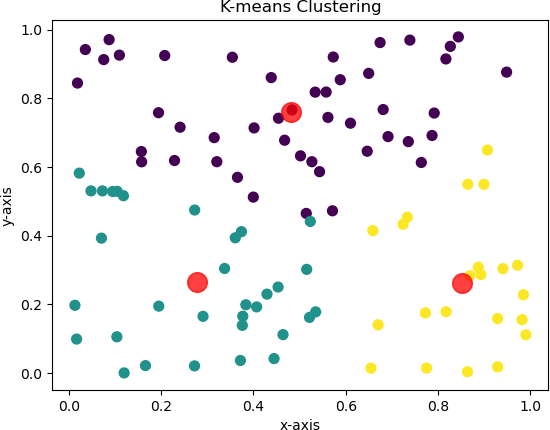
# Pract no 5: Implement the clustering using K-means.

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 :**



# 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,rando m\_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']])

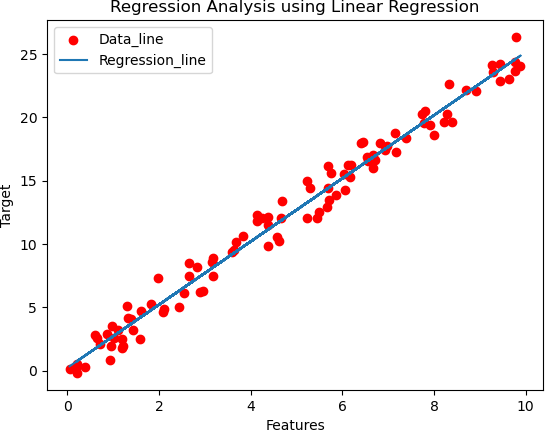
plt.scatter(data['Feature'],data['Target'],color='red',label='Data\_line') plt.plot(data['Feature'],y\_pred,label='Regression\_line') plt.xlabel('Features')

plt.ylabel('Target')

plt.title('Regression Analysis using Linear Regression') plt.legend()

plt.show()

**Output :**



# 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'],

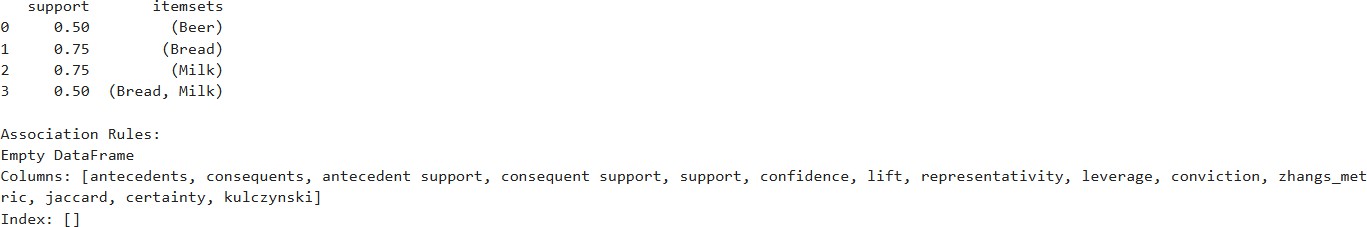
['Milk','Beer','Cola'],

['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:-**

****

# 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))

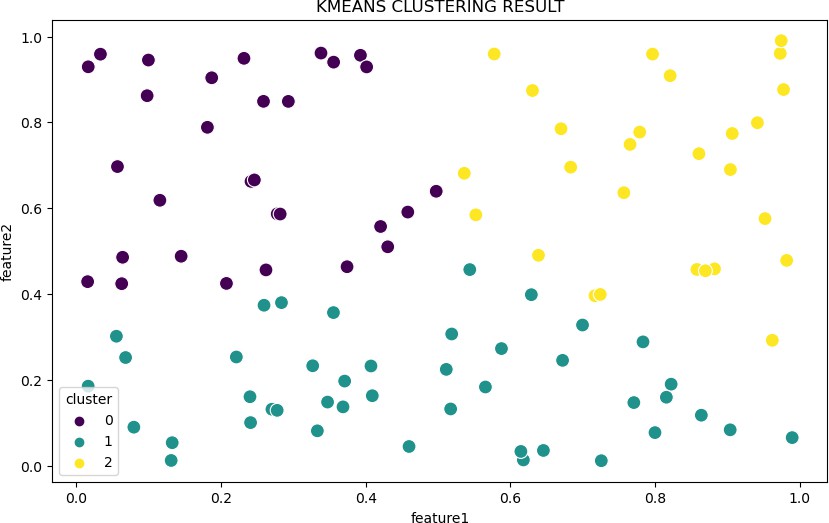
sns.scatterplot(data=df,x='feature1',y='feature2',palette='viridis',hue

='cluster',s=100)

plt.title('KMEANS CLUSTERING RESULT')

plt.legend(title='cluster') plt.show()

**Output :**



# 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)

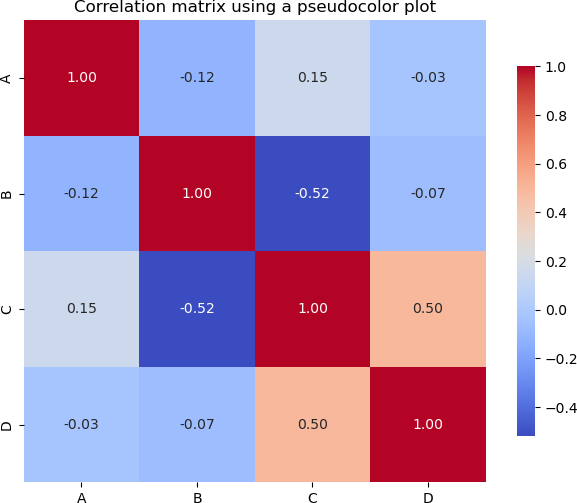
corr=df.corr() plt.figure(figsize=(8,6))

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

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

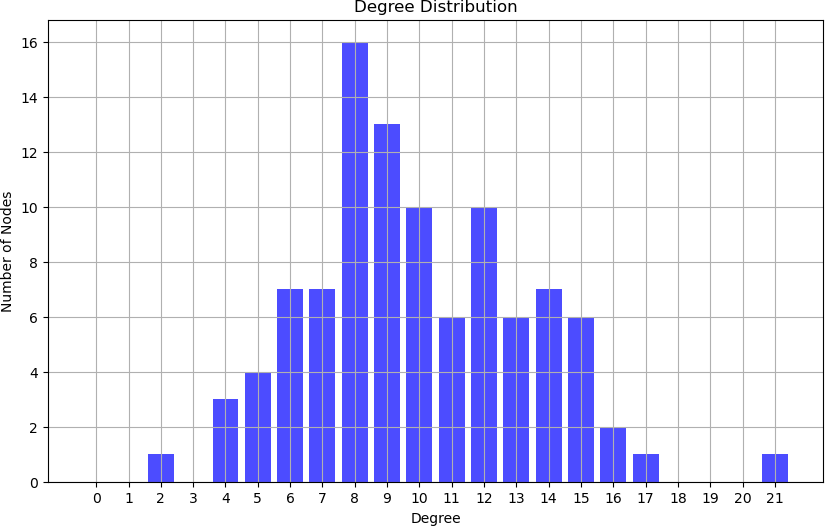
n=100 p=0.1

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 :**



# 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(f acecolor='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

