INDEX

BCA 507(C):- PRACTILE ON MACHINE LEARNING USING PYTHON

1.Write a python program to find mean , mode , median.

```
from scipy import stats
import numpy
c=[23,45,77,12,78,90,78,34,78]
x=numpy.mean(c)
y=numpy.median(c)
z=stats.mode(c)
print("mean is=",x)
print("median is=",y)
print("mode is=",z)
```

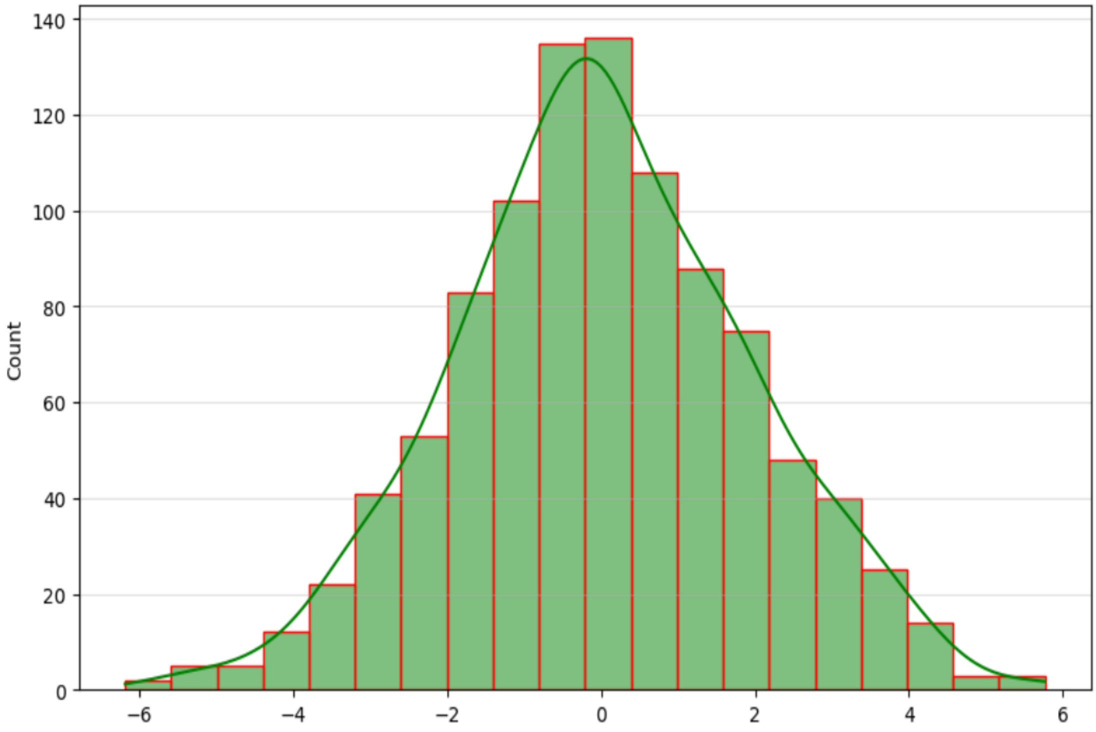
Output:-

```
mean is= 57.22222222222222
median is= 77.0
mode is= ModeResult(mode=78, count=3)
```

2.Write a python program to typical data distribution.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
D=np.random.normal(0,2,1000)
x=np.random.normal(0,2,1000)
y=np.random.normal(0,2,1000)
plt.figure(figsize=(10,6))
sns.histplot(D,bins=20,kde=True,color='g',edgecolor='r')
plt.grid(axis='y',alpha=0.30)
plt.show()
```

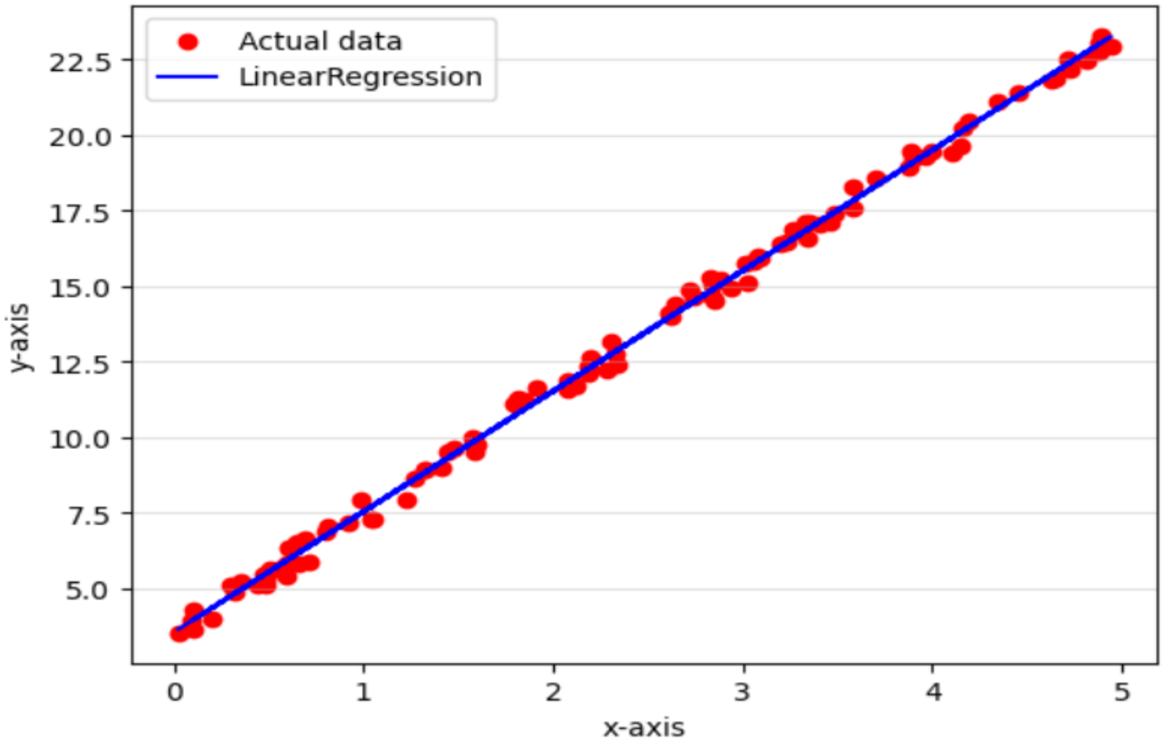
Output:-



3.Write a python program to draw scatter plot of linear regression.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
np.random.seed(0)
x=5*np.random.rand(100,1)
y=4*x+3+np.random.rand(100,1)
model=LinearRegression()
model.fit(x,y)
y_pred=model.predict(x)
plt.scatter(x,y,color='r',label='Actual data')
plt.plot(x,y_pred,color='b',label='LinearRegression')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid(axis='y',alpha=0.40)
plt.legend()
plt.show()
```

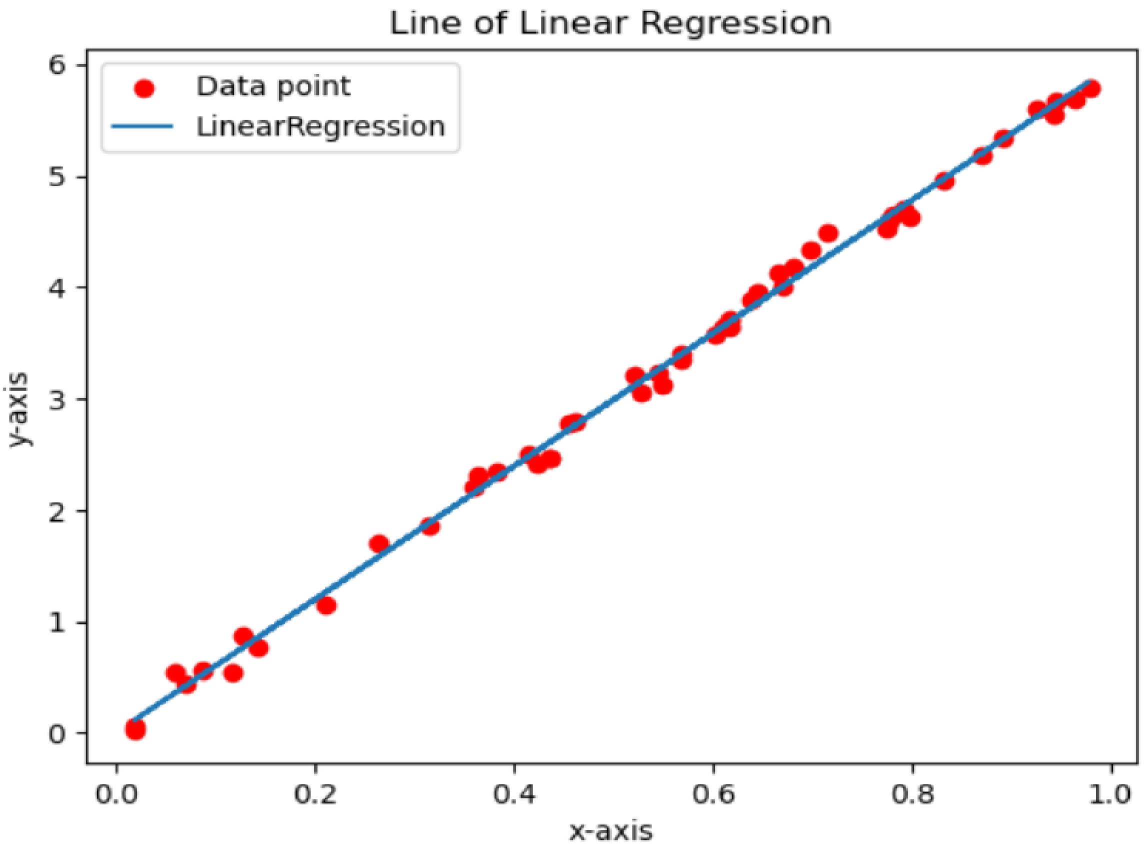
Output:-



4.Write a python program to draw the line of linear regression.

```
import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)

x=np.random.rand(50)

y=6*x+np.random.normal(0,0.1,50)

slope,intercept=np.polyfit(x,y,1)

list=slope*x+intercept

plt.scatter(x,y,color='r',label='Data point')

plt.plot(x,list,label='LinearRegression')

plt.xlabel('x-axis')

plt.ylabel('y-axis')

plt.title('Line of Linear Regression')

plt.legend()

plt.show()
```

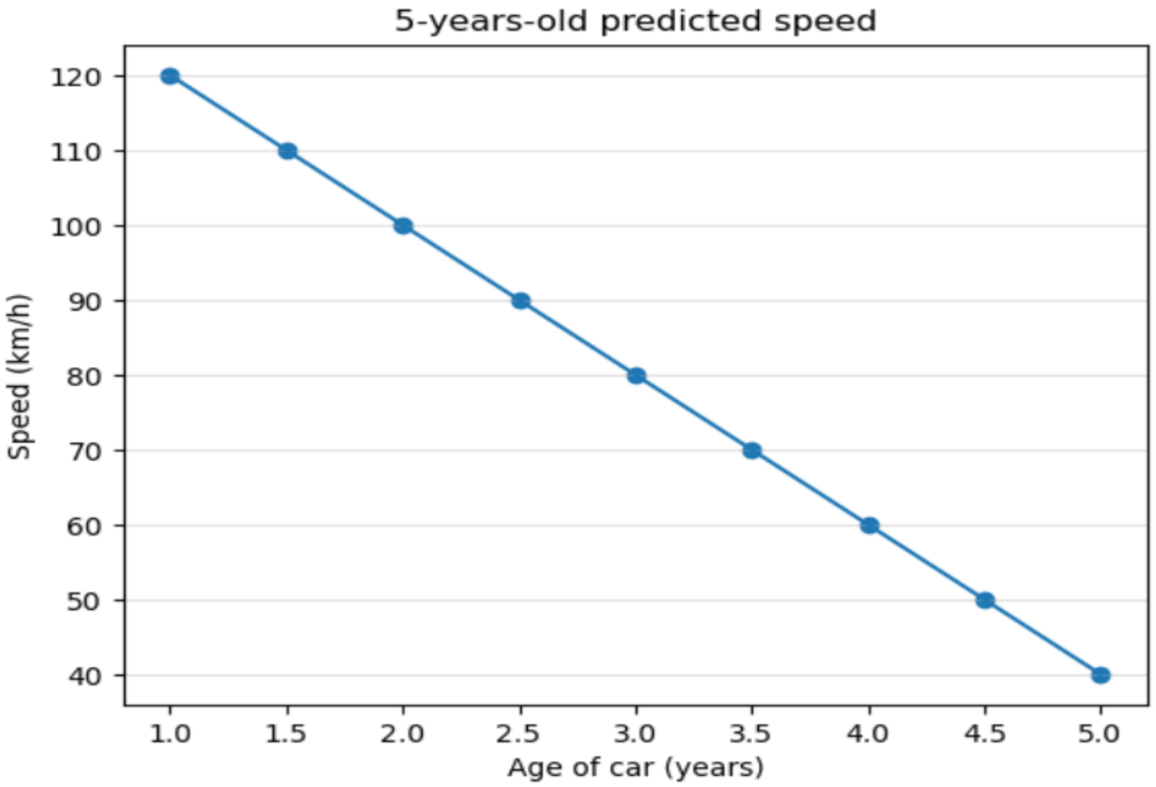
Output:-



5.Write a python Program to predict the speed of a 5 years old car.

```
import matplotlib.pyplot as plt
from scipy import stats
x = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]#year
y = [120,110,100,90,80,70,60,50,40]#speed
slp, incpt, r, p, st = stats.linregress(x, y)
def myfunc(x):
    return slp * x + incpt
mymodel = list(map(myfunc, x))
plt.xlabel('Age of car (years)')
plt.ylabel('Speed (km/h)')
plt.title('5-years-old predicted speed')
plt.scatter(x, y)
plt.grid(axis='y', alpha=0.40)
plt.plot(x, mymodel)
plt.show()
```

Output:-



6.Write a python Program to print the coefficient values of the regression object.

```
from sklearn.linear_model import LinearRegression

import numpy as np

np.random.seed(0)

x=np.random.rand(50,1)

y=2*x.squeeze()+np.random.normal(0,0.1,50)

regression_model=LinearRegression()

regression_model.fit(x,y)

print('coefficent value',regression_model.coef_)
```

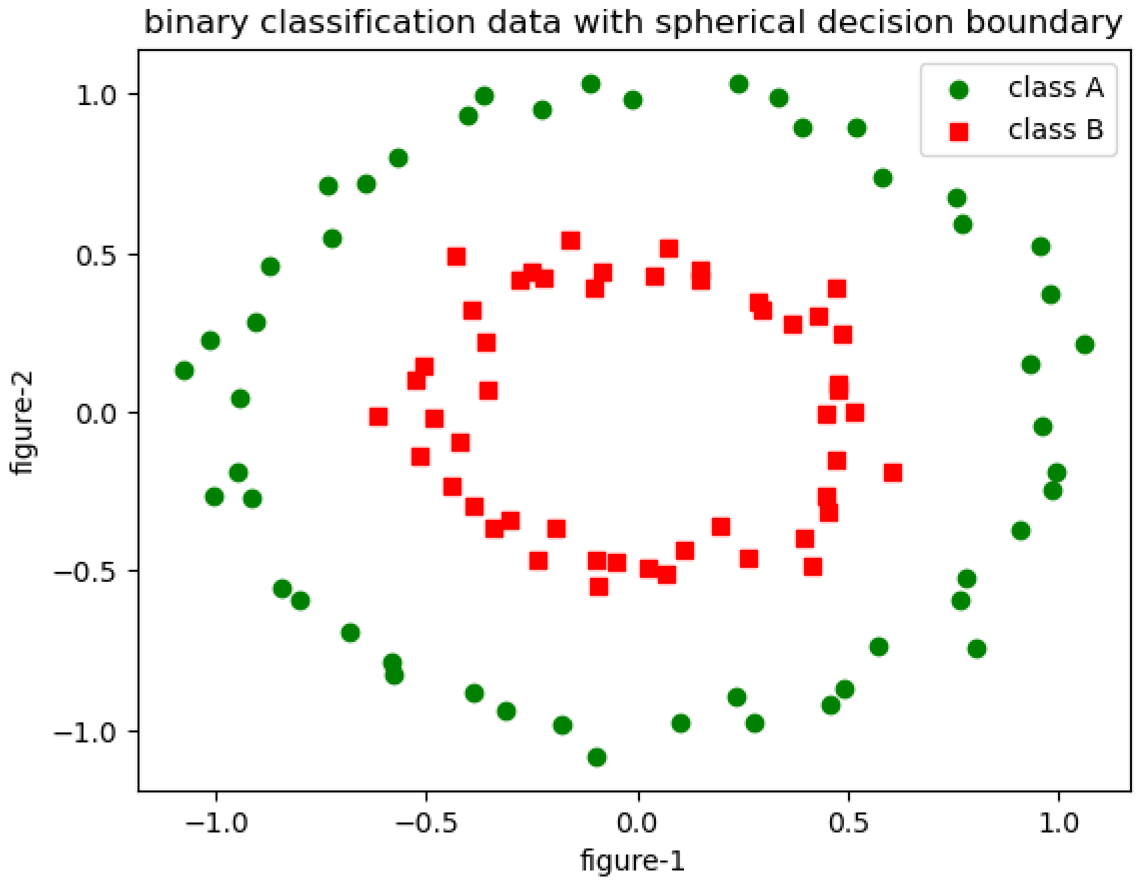
Output:-

```
coefficent value [1.96927329]
```

7.Write a python program to 2nd binary classification data generated by make_circles() have a spherical decision boundary.

```
from sklearn.datasets import make_circles

import matplotlib.pyplot as plt

x,y=make_circles(n_samples=100,noise=0.05,factor=0.5,random_state=42)

plt.figure()

plt.scatter(x[y==0][:,0],x[y==0][:,1],color='g',label='class A')

plt.scatter(x[y==1][:,0],x[y==1][:,1],color='r', marker='s',label='class B')

plt.title('binary classification data with spherical decision boundary')

plt.xlabel('figure-1')

plt.ylabel('figure-2')

plt.legend()

plt.show()
```

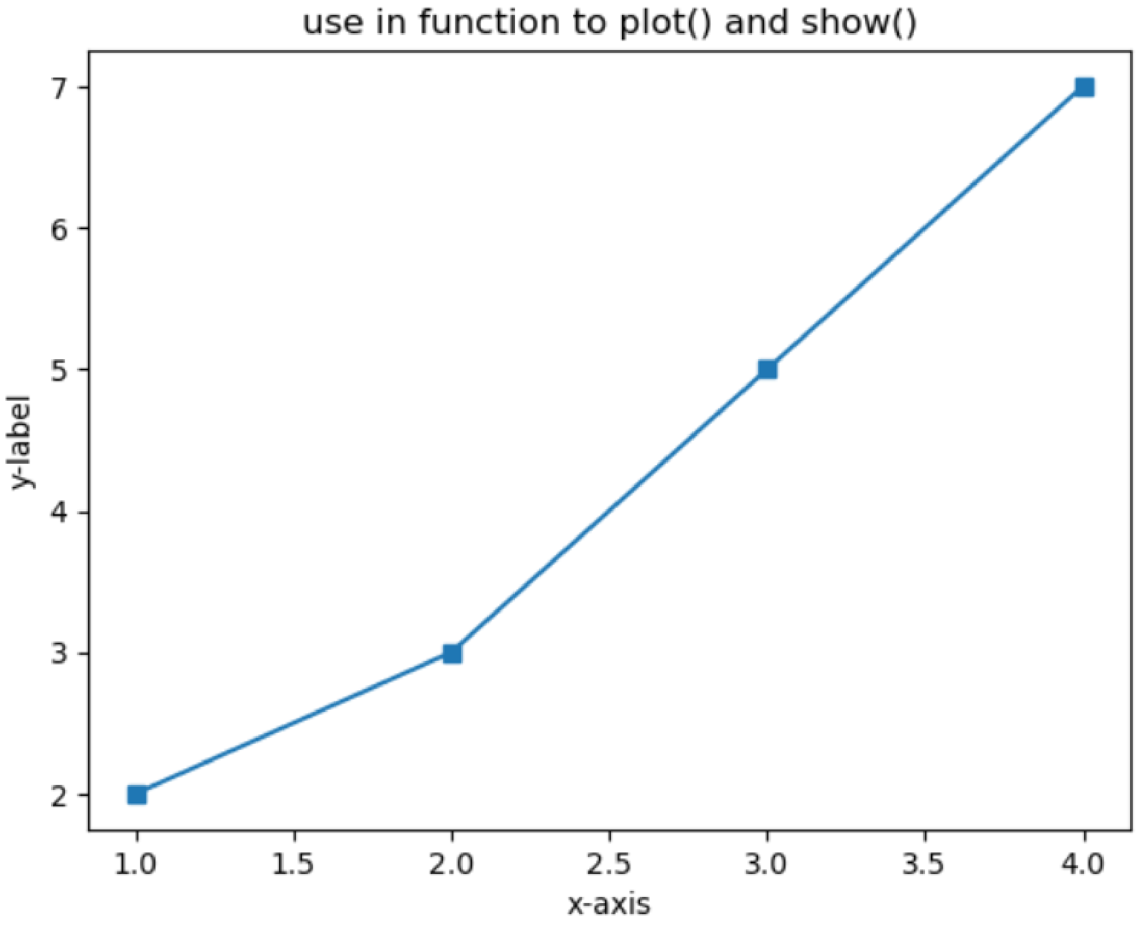
Output:-



8.Write a python program to display the plot we can use the functions plot() and show() from pyplot.

```
import matplotlib.pyplot as plt
x=([1],[2],[3],[4])
y=([2,3,5,7])
plt.plot(x,y,marker='s')
plt.xlabel('x-axis')
plt.ylabel('y-label')
plt.title('use in function to plot() and show()')
plt.show()
```

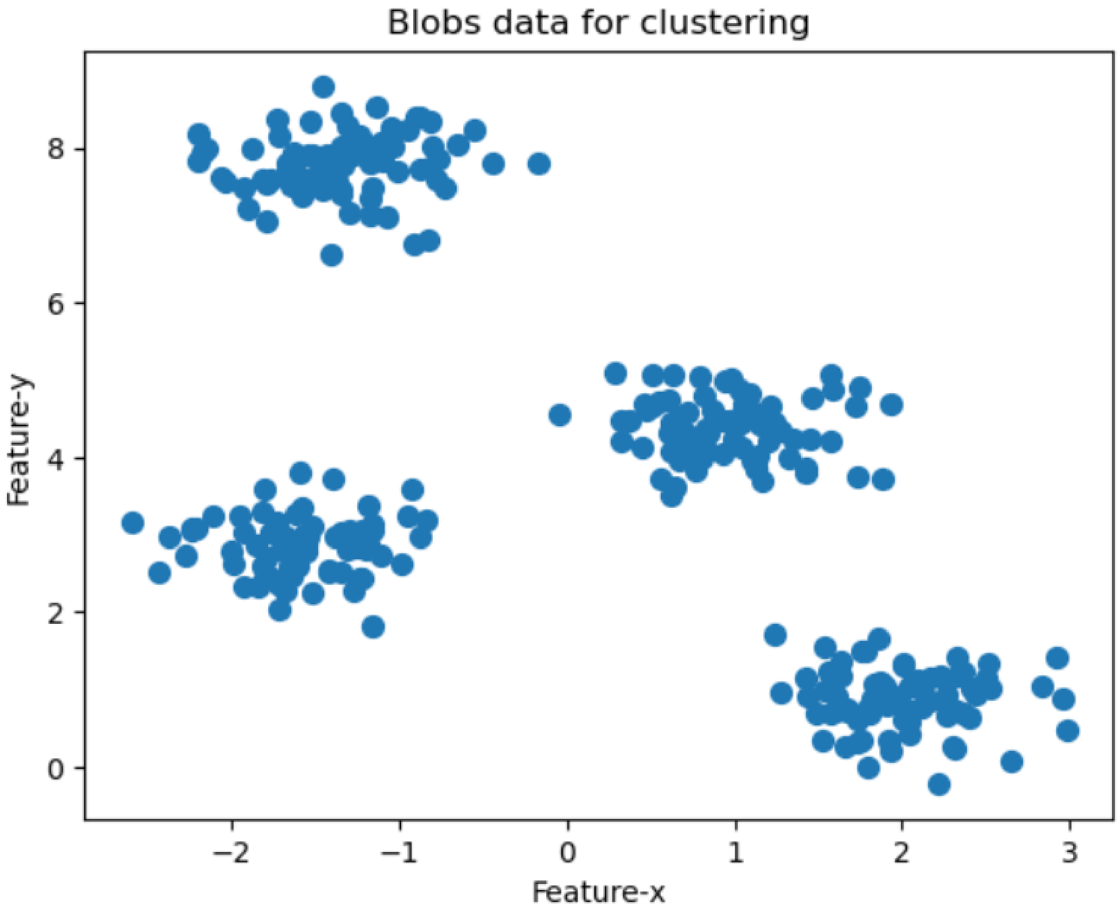
Output:-



9.Write a python Program to data generated by the function make_blobs() are blobs that can be utilized for clustering.

```
import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs

x,y=make_blobs(n_samples=300,centers=4,cluster_std=0.40,random_state=0)

plt.scatter(x[:,0],x[:,1],s=50)

plt.xlabel('Feature-x')

plt.ylabel('Feature-y')

plt.title('Blobs data for clustering')

plt.show()
```

Output:-



10.Write a python program to random multi-label classification data is created by the function make make_multilabel_classification().

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make_multilabel_classification

from sklearn.multioutput import MultiOutputClassifier

from sklearn.ensemble import RandomForestClassifier

X,y=make_multilabel_classification(n_samples=100,n_features=20,n_classes=5, n_labels=2, random_state=0)

clf=MultiOutputClassifier(RandomForestClassifier(n_estimators=100))

clf.fit(X,y)

plt.scatter(X[:,0],X[:,1], c=[np.argmax(i) for i in y])

print(X,y)

plt.title("multi-label Classification")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()
```

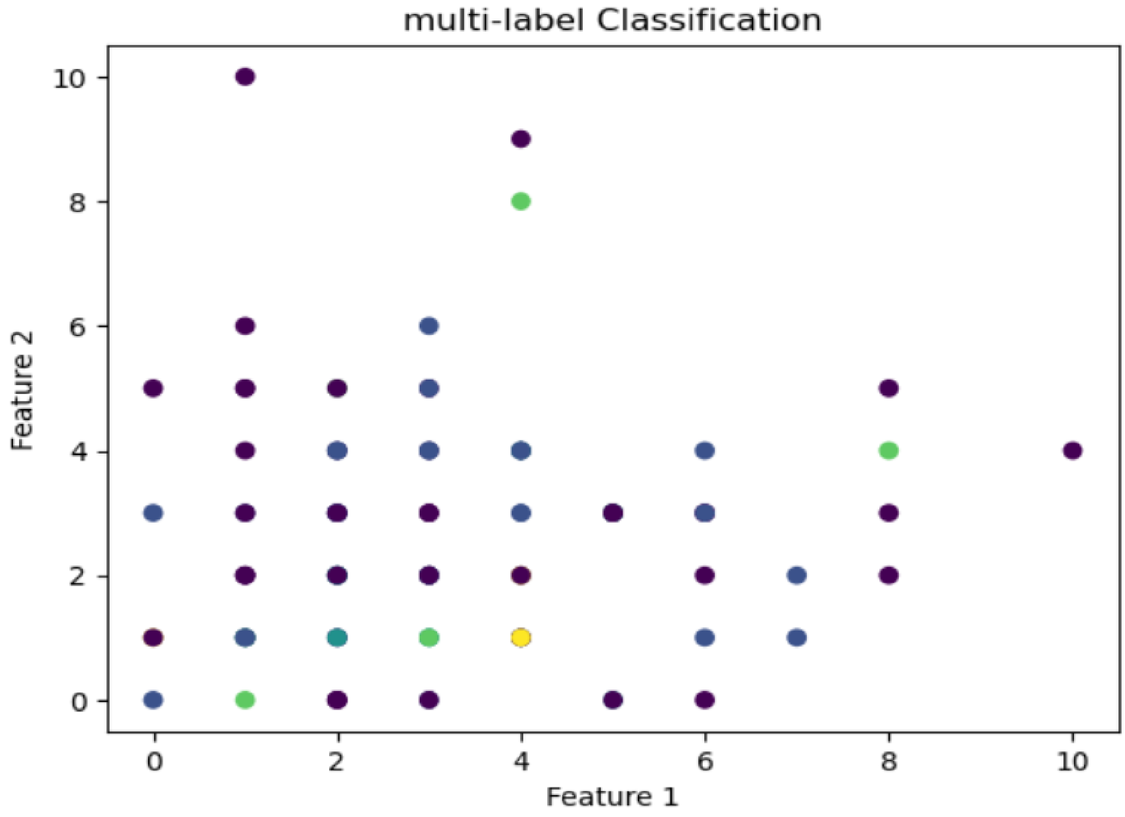
Output:-

```
[[3. 1. 4. ... 4. 1. 3.]
 [5. 0. 6. ... 0. 0. 3.]
 [3. 4. 1. ... 3. 2. 5.]
 ...
 [2. 1. 2. ... 1. 0. 3.]
 [6. 4. 1. ... 1. 3. 5.]
 [2. 4. 2. ... 5. 4. 2.]] [[0 0 1 1 1]
 [0 0 1 0 0]
 [1 1 0 1 0]
 [1 1 1 1 1]
 [1 1 1 0 0]
 [1 1 1 0 0]
 [0 1 0 0 1]
```

```
[0 1 1 1 1]

[1 1 0 0 1]

[1 1 1 1 1]

[0 0 0 0 0]

[0 0 1 0 1]

[0 0 0 1 1]

[1 1 0 1 1]

[0 0 1 0 0]

[1 0 1 1 0]

[1 0 0 1 1]

[0 0 0 1 1]

[0 0 1 0 1]

[1 1 1 1 0]

[0 1 0 1 1]

[0 0 0 0 0]

[1 1 0 0 0]

[1 0 0 0 0]

[1 0 0 1 0]

[1 0 0 0 1]

[0 0 0 0 1]

[0 0 0 0 0]

[1 1 0 0 0]

[1 0 1 0 0]

[0 1 0 0 0]

[0 0 0 0 1]

[1 1 0 1 1]

[0 1 0 1 0]

[0 1 0 0 0]

[0 0 1 0 0]

[1 1 0 1 0]

[1 0 0 1 0]
```

```
[0 1 0 1 1]
[0 0 1 0 1]
[0 0 1 0 0]
[0 0 0 1 0]
[1 1 1 0 1]
[0 0 1 0 1]
[0 0 0 0 0]
[1 1 1 1 1]
[0 1 0 0 1]
[0 0 0 0 0]
[1 0 1 0 1]
[0 1 0 1 0]
[0 1 1 0 1]
[1 0 1 1 1]
[1 0 1 0 0]
[0 1 1 0 0]
[0 0 0 1 0]
[0 1 0 0 0]
[0 0 0 0 0]
[0 1 1 1 1]
[1 1 1 1 0]
[1 0 0 1 0]
[0 1 1 0 1]
[0 0 0 1 1]
[0 0 0 0 0]
[0 1 1 0 0]
[0 1 1 1 0]
[0 1 1 1 0]
[1 0 1 1 1]
[0 1 0 0 0]
[0 0 0 0 0]
```

```
[0 0 0 0 0]
[0 1 1 1 1]
[0 0 0 0 0]
[0 0 0 1 0]
[0 0 0 0 0]
[0 1 0 1 0]
[0 0 1 1 1]
[0 1 0 0 0]
[1 0 1 0 0]
[0 0 0 1 0]
[0 1 0 1 0]
[0 1 0 0 0]
[1 1 1 1 1]
[0 1 1 0 0]
[1 1 0 1 0]
[1 1 1 1 0]
[0 0 1 0 0]
[0 1 1 0 0]
[0 0 0 0 0]
[1 0 0 0 0]
[0 1 1 1 0]
[0 0 0 0 0]
[0 1 1 1 1]
[0 0 0 0 1]
[0 1 1 0 0]
[1 1 1 1 0]
[0 0 0 0 0]
[1 0 0 0 1]
[0 0 1 0 0]
[0 1 1 0 0]
[0 1 0 1 1]]
```


multi-label Classification
Feature 2
Feature 1
0
2
4
6
8
10
0
2
4
6
8
10

11.Write a python program to implement the KNN algorithm.

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
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)
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
def knn(X_train,y_train,X_test,k):
    predictions=[]
    for x in X_test:
        distances=[np.linalg.norm(x -x_train) for x_train in X_train]
        k_indices=np.argsort(distances)[:k]
        k_labels=[y_train[i] for i in k_indices]
        prediction=max(set(k_labels),key=k_labels.count)
        predictions.append(prediction)
    return predictions
k=5
predictions=knn(X_train,y_train,X_test,k)
accuracy=np.mean(predictions==y_test)
print("Accuracy:",accuracy)
```

Output:-

```
Accuracy: 1.0
```

12.Write a python program to creating a dataframe to implement one hot encoding from CSV file.

```
import pandas as pd

data=pd.read_csv(r"C:\Users\BCA PC11\Documents\ab\harsha.csv")

print(data)

one_hot_encoded_data=pd.get_dummies(data,columns=['user_id','age'])

print(one_hot_encoded_data)

one_hot_encoded_data.to_csv(r"C:\Users\BCAPC11\Documents\ab\one_hot_encoded_harsha.csv",index=True)
```

Output:-

```
   user_id  age  annual_income  purchase_amount  loyalty_score region  \
0        1   25          45000              200            4.5  North
1        2   34          55000              350            7.0  South
2        3   45          65000              500            8.0   West
3        4   22          30000              150            3.0   East
4        5   29          47000              220            4.8  North
5        6   41          61000              480            7.8  South
6        7   36          54000              400            6.5   West
7        8   27          43000              230            4.2   East

   purchase_frequency
0                  12
1                  18
2                  22
3                  10
4                  13
5                  21
6                  19
7                  14
```

```
   annual_income  purchase_amount  loyalty_score region  purchase_frequency  \
0          45000              200            4.5  North                  12
1          55000              350            7.0  South                  18
2          65000              500            8.0   West                  22
3          30000              150            3.0   East                10
4          47000              220            4.8  North                  13
5          61000              480            7.8  South                  21
6          54000              400            6.5   West                  19
7          43000              230            4.2   East                14

   user_id_1  user_id_2  user_id_3  user_id_4  user_id_5  ...  user_id_7  \
0       True      False      False      False      False  ...      False
1      False       True      False      False      False  ...      False
2      False      False       True      False      False  ...      False
3      False      False      False       True      False  ...      False
4      False      False      False      False       True  ...      False
5      False      False      False      False      False  ...      False
6      False      False      False      False      False  ...       True
7      False      False      False      False      False  ...      False

   user_id_8  age_22  age_25  age_27  age_29  age_34  age_36  age_41  age_45
0      False   False    True   False   False   False   False   False   False
1      False   False   False   False   False    True   False   False   False
2      False   False   False   False   False   False   False   False    True
3      False    True   False   False   False   False   False   False   False
4      False   False   False   False    True   False   False   False   False
5      False   False   False   False   False   False   False    True   False
6      False   False   False   False   False   False    True   False   False
7       True   False   False    True   False   False   False   False   False

[8 rows x 21 columns]
```