COLLEGE OF ENGINEERING CHERTHALA

LAB RECORD

20MCA241 – DATA SCIENCE LAB

CERTIFICATE

This is certified to be bona fide works of Mr./Ms. …………………………….…………………., In the class………………...., Reg. No. ……………………., of College of Engineering Cherthala, during the academic year 2024-25.

Teacher In Charge **External Examiner**

Internal Examiner

Output:

```
1st array :  [1, 2, 3, 4]
2nd array :  [5, 6, 7, 8]
Addition :  [ 6  8 10 12]
Multiplication :  [ 5 12 21 32]
```

EXPERIMENT NO.1
INTRODUCTION TO NUMPY

Program No.1
Add and multiply two arrays

Aim: Write a NumPy program to add and multiply two arrays.

Algorithm:

1. Import Library: Import the numpy library.
2. Define Function: Create a function add_and_multiply_arrays with two input arrays
3. Check Shapes: Ensure array1 and array2 have the same shape. Raise an error if not.
4. Addition: Use np.add() for element-wise addition and store the result in sum_result.
5. Multiplication: Use np.multiply() for element-wise multiplication and store the result in product_result.
6. Return Results: Return sum_result and product_result from the function.
7. Create Inputs: Define two sample arrays, array_a and array_b.
8. Call Function: Pass the arrays to add_and_multiply_arrays.
9. Print Results: Display the input arrays, sum_result, and product_result.

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4])
brr = np.array([5, 4, 3, 2])
addition_result =np.add(arr, brr)
print("Addition:", addition_result)
multiplication_result = np.multiply(arr, brr)
print("Multiplication:", multiplication_result)
```

Result: Program executed successfully and output verified.

Output:

```
Enter the size of the array:4
enter  1 th array elements
5
enter  2 th array elements
8
enter  3 th array elements
3
enter  4 th array elements
1
Array:
[[5], [8], [3], [1]]
Sum of elements:
17
Sum of col:
[17]
Sum of row:
[5 8 3 1]
```

Program No.2
Sum in an Array

Aim: Write a NumPy program to compute sum of all elements, sum of each column and sum of eachrow of a given array.

Algorithm:

1. Import the numpy library and initialize an empty list arr to hold the array.
2. Input Array Size: Prompt the user to enter the size of the array (size).
3. Input Array Elements:
    a. Use a loop to collect size rows of array elements.
    b. For each row, prompt the user to input space-separated integers.
    c. Convert the input to a list of integers using map() and append it to arr.
4. Display the Array: Print the constructed 2D list arr.
5. Compute Total Sum: Use numpy.sum() to calculate the sum of all elements in the array and print it.
6. Compute Column-wise Sum: Use numpy.sum(arr, axis=0) to calculate and print the sum of each column.
7. Compute Row-wise Sum: Use numpy.sum(arr, axis=1) to calculate and print the sum of each row.
8. End the program.

Program:

```
import numpy
arr=[]
size=int(input("Enter the size of the array:"))
for i in range(size):
        print("enter ",i+1,"th array elements")
        row= list(map(int,input().split()))
        arr.append(row)
print("Array:")
print(arr)
print("Sum of elements:")
print(numpy.sum(arr))
print("Sum of each col:")
print(numpy.sum(arr,axis=0))
print("Sum of each row:")
print(numpy.sum(arr,axis=1))
```

Result: Program executed successfully and output verified.

Output:

```
enter first array :1 2 3 4 5
enter second array :1 2 3 4 5
arrays are equal



enter first array :4 5 6 2
enter second array :2 5 4 66
arrays are not equal
```

Program No.3
Check whether two arrays are equal or not

Aim: Write a NumPy program to check whether two arrays are equal or not.

Algorithm:

1. Import the numpy library.
2. Input Arrays: Take two space-separated integer inputs, convert them to NumPy arrays (array1 and array2).
3. Check Equality: Use np.array_equal() to compare array1 and array2.
4. Print "arrays are equal" if they match. Print "arrays are not equal" otherwise.
5. End the program.

Program:

```
import numpy as np
input1 = list(map(int,input("enter first array").split()))
array1 = np.asarray(input1)
input2 = list(map(int,input("enter second array").split()))
array2 = np.asarray(input2)
if np.array_equal(array1, array2):
        print("arrays are equal")
else:
        print(“arrays are not equal”)
```

Result: Program executed successfully and output verified.

Output:

```
Enter the size of the square matrix: 2
Enter each row of the matrix, with elements separated by spaces:
4 3
3 2
[[-2.  3.]
 [ 3. -4.]]
```

EXPERIMENT NO.2
MATRIX OPERATIONS

Program No.4
Inverse of the matrix

Aim: Write Python program to create two matrices (read values from user) and find inverse of thematrix.

Algorithm:

1. Import the numpy library as np.
2. Input Matrix Size: Prompt the user to input the size of the square matrix (size).
3. Input Matrix Elements:
    a. Initialize an empty list matrix.
    b. Use a loop to iterate size times to input each row of the matrix.
    c. Convert the row input into a list of floating-point numbers using map(float, input().split()).
    d. Append each row to the matrix list.
4. Compute Inverse: Use np.linalg.inv(matrix) to compute the inverse of the square matrix.
5. Output Inverse: Print the resulting inverse matrix.
6. End the program.

Program:

```
import numpy as np
size = int(input("Enter the size of the square matrix: "))
matrix = []
print("Enter each row of the matrix, with elements separated by spaces:")
for i in range(size):
    row = list(map(float, input().split()))
    matrix.append(row)
inverse=np.linalg.inv(matrix)
print(inverse)
```

Result: Program executed successfully and output verified.

Output:

```
Enter the size of the square matrix: 3
Enter each row of the matrix, with elements separated by spaces:
4 5 6
5 8 2
7 5 3
[[4. 5. 7.]
 [5. 8. 5.]
 [6. 2. 3.]]
```

Program No.5
Transpose of the matrix

Aim: Write Python program to create two matrices (read values from user) and find transpose of the matrix.

Algorithm:

```
1. Import the numpy library as np.
2. Input Matrix Size: Prompt the user to enter the size of the square matrix (size).
3. Input Matrix Elements:
    a. Initialize an empty list matrix.
    b. Use a loop to iterate size times to input each row of the matrix.
    c. Convert each row input into a list of floating-point numbers using map(float,
       input().split()).
    d. Append the row to the matrix list.
4. Compute Transpose: Use np.transpose(matrix) to compute the transpose of the matrix.
5. Output Transpose: Print the transposed matrix.
6. End the program.
```

Program:

```
import numpy as np
size = int(input("Enter the size of the square matrix: "))
matrix = []
print("Enter each row of the matrix, with elements separated by spaces:")
for i in range(size):
        row = list(map(float, input().split()))
        matrix.append(row)
trans=np.transpose(matrix)
print(trans)
```

Result: Program executed successfully and output verified.

Output:

```
Enter the size of the matrix:2
Enter each row of the matrix, with elements separated by spaces:
2 1
5 8
Determinant
11
```

Program No.6
Determinant of the matrix

Aim: Write Python program to create two matrices (read values from user) and find determinant ofthe matrix.

Algorithm:

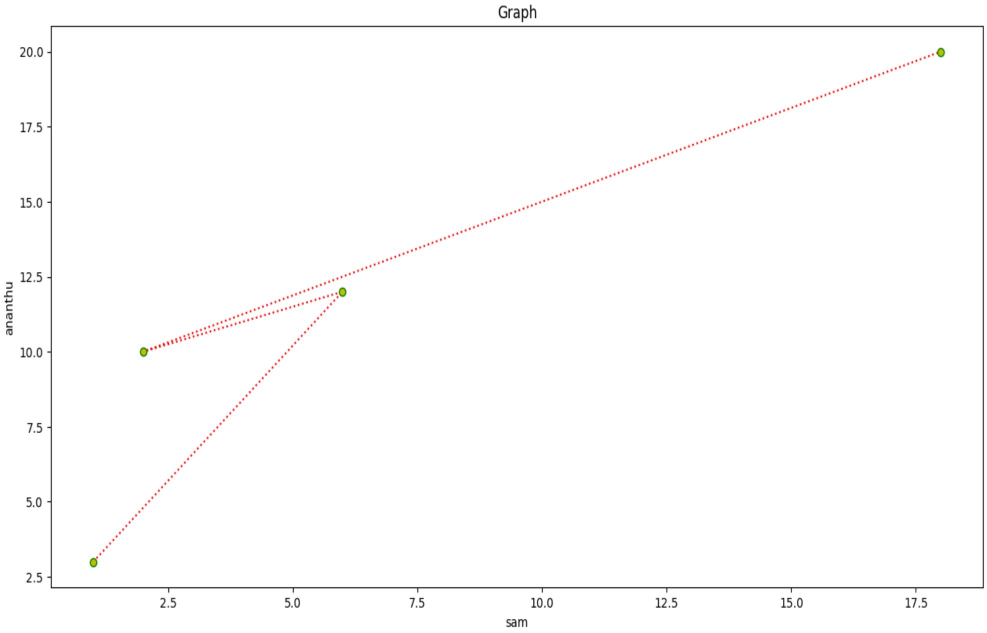
```
1. Import the numpy library as np.
2. Input Array Size: Prompt the user to enter the size of the square matrix (n).
3. Input Matrix Elements:
    a. Initialize an empty list arr.
    b. Use a loop to input n rows of the matrix.
    c. Convert each row input into a list of integers using map(int, input().split()).
    d. Append each row to the arr list.
4. Compute Determinant:
    a. Use np.linalg.det(arr) to calculate the determinant of the matrix.
    b. Round the result and convert it to an integer using int(round(d)).
5. Print the rounded determinant value.
6. End the program.
```

Program:

```
import numpy as np
n=int(input("Enter the size of an array:"));
arr=[]
for i in range (n):
    x = list(map(int,input().split()))
    arr.append(x)
d=np.linalg.det(arr)
print("Determinant")
print(int(round(d)))
```

Result: Program executed successfully and output verified.

Output:


Graph
ananthu
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
2.5
5.0
7.5
10.0
12.5
15.0
17.5
sam

EXPERIMENT NO.3
PROGRAMS USING MATPLOTLIB

Program No.7
Line Diagram

Aim: Draw a line in a diagram from position (1, 3) ,(6, 12) ,(2,10)and finally to position (18, 20). (Mark each point with a beautiful green colour and set line colour to red and line style dotted) then label x-axis and y-axis.

Algorithm:

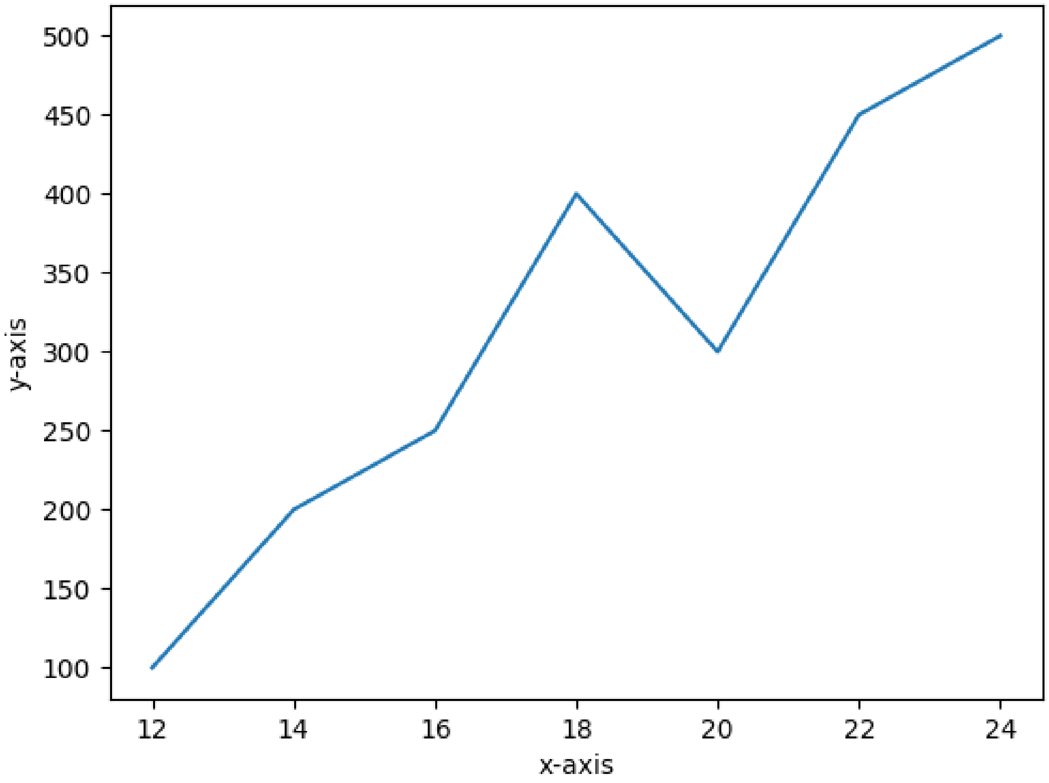
1. Import Necessary Modules: Import the required libraries, numpy as np for numerical operations and matplotlib.pyplot as plt for plotting.
2. Create Data Arrays: Define the x and y coordinates as NumPy arrays.
3. Set Up the Plot: Set up the plot by providing a title, and labeling the x and y axis.
4. Plot the Line: Use the plt.plot function to plot the line with specific styles such as color, line style, marker, and marker colors.
5. Display the Plot: Use plt.show() to display the plot on the screen. This is necessary to visualize the plot.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,6,2,18])
y=np.array([3,12,10,20])
plt.plot(x,y,color ='r',marker ='o',mfc="y",mec='g',ls=':')
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("Graph")
plt.show()
```

Result: Program executed successfully and output verified.

Output:


500
450
400
350
300
250
200
150
100
y-axis
12
14
16
18
20
22
24
x-axis

Program No.8
Line diagram values from a text file

Aim: Write a Python program to draw a line using given axis values taken from a text file, withsuitable label in the x axis, y axis and a title.

Temperature in degree Celsius	Sales
12	100
14	200
16	250
18	400
20	300
22	450
24	500

Algorithm:

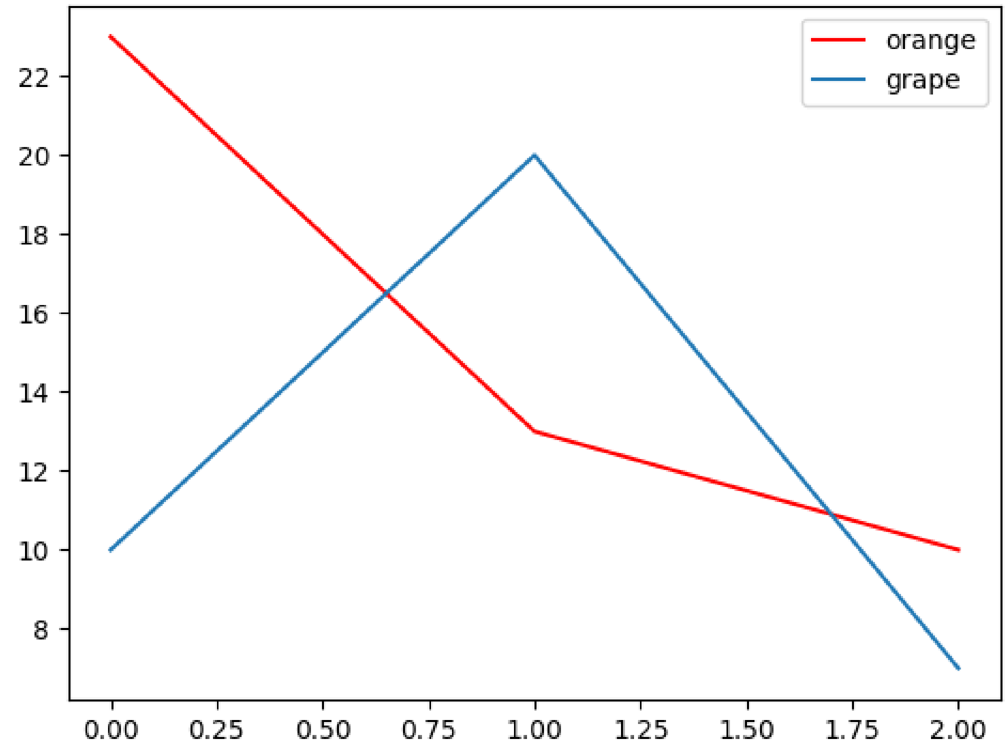
1. Import Necessary Modules: Import the required libraries, matplotlib.pyplot as plt for plotting.
2. Read Data from the Text File: Read the temperature and sales data from the text file. You can use the numpy.loadtxt function or other methods to read data from the file.
3. Set Up the Plot: Set up the plot by providing a title, and labeling the x and y axis.
4. Plot the Line: Use the plt.plot function to plot the line with specific styles such as marker, linestyle, and color.
5. Display Labels and Title: Use plt.legend() to display the legend.
6. Show the Plot: Use plt.show() to display the plot on the screen.

Program:

```
import matplotlib.pyplot as plt
x=[]
y=[]
f=open("sales.txt")
next(f)
for row in f:
        row=row.split(' ')
        x.append(int(row[0]))
        y.append(int(row[1]))
plt.xlabel("Temperature in degree celsius")
plt.ylabel("Sales")
plt.title(“Sales vs Temperature”)
plt.plot(x,y)
plt.show()
```

Result: Program executed successfully and output verified.

Output:



orange
grape
22
20
18
16
14
12
10
8
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Program No.9
Two or more lines on same plot with suitable legends

Aim: Write a Python program to plot two or more lines on same plot with suitable legends of eachline.

Algorithm:

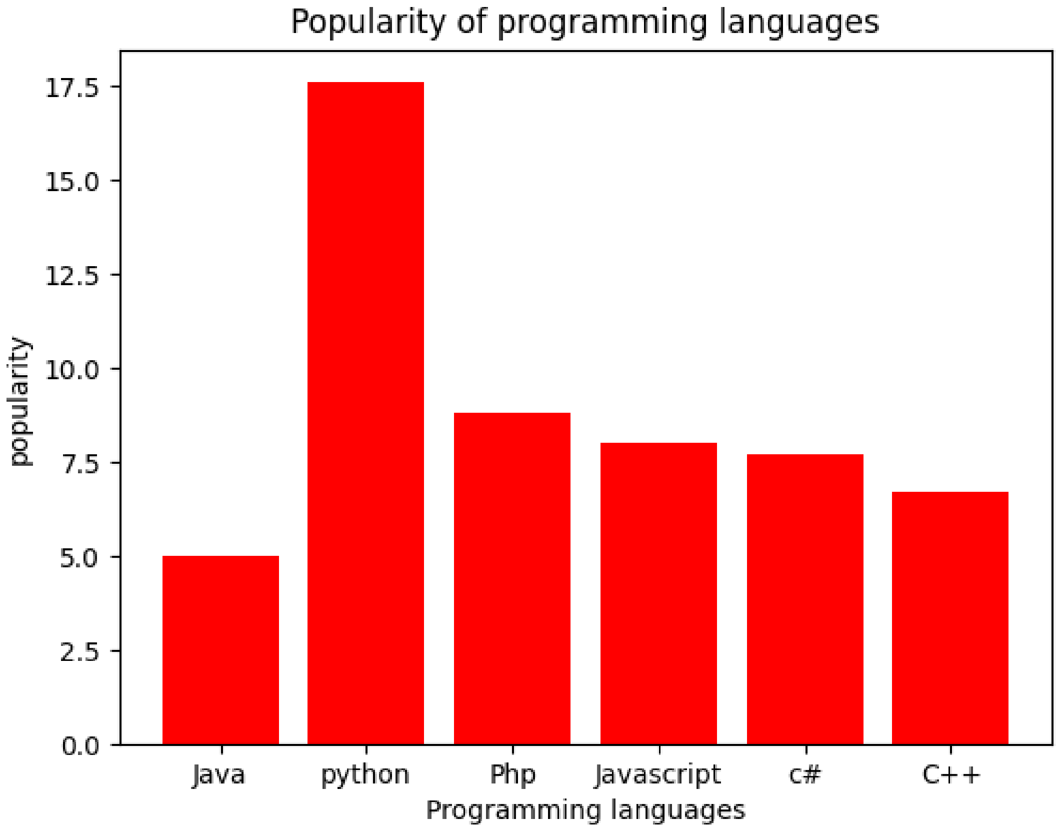
1. Import Necessary Modules: Import the required library, matplotlib.pyplot as plt for plotting.
2. Set Up the Plot: Set up the plot by providing a title, and labeling the x and y axes.
3. Plot the Lines: Use the plt.plot function to plot each line with specific x and y coordinates.
4. Add Legends: Use plt.legend() to add legends for each line.
5. Display the Plot: Use plt.show() to display the plot on the screen.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([23,13,10])
y=np.array([10,20,7])
plt.plot(x)
plt.plot(y)
plt.legend(["orange", "grape"], loc="upper right")
plt.title(“FRUIT SALES”)
plt.show()
```

Result: Program executed successfully and output verified.

Output:


Popularity of programming languages
popularity
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Java
python
Php
Javascript
c#
C++
Programming languages

Program No.10
Bar chart

Aim: Write a Python programming to display a bar chart of the popularity of programming Languages.

Programming languages:	Java	Python	PHP	JavaScript	C#	C++
Popularity	22.2	17.6	8.8	8	7.7	6.7

Algorithm:

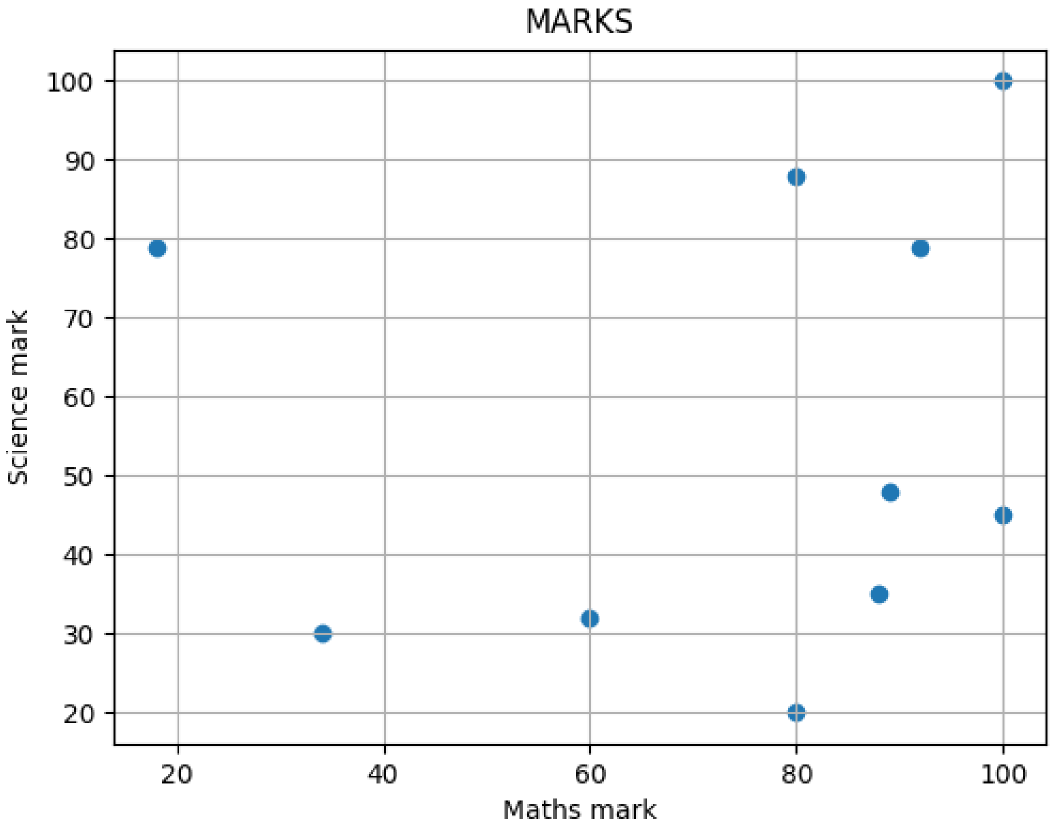
1. Import Necessary Modules: Import the required libraries, matplotlib.pyplot as plt for plotting and numpy as np for numerical operations.
2. Set Up the Data: Define arrays for programming languages and their popularity values.
3. Set Up the Plot: Use plt.bar to create a bar chart with programming languages on the x-axis and popularity percentages on the y-axis.
4. Set Up Labels and Title: Provide labels for the x-axis, y-axis, and a title for the plot.
5. Display the Plot: Use plt.show() to display the plot on the screen.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array(["Java","python","Php","Javascript","c#","Cpp"])
y=np.array([5,17.6,8.8,8,7.7,6.7])
plt.bar(x,y,color="r")
plt.title("Languages vs Popularity")
plt.xlabel("Programming languages")
plt.ylabel("popularity")
plt.show()
```

Result: Program executed successfully and output verified.

Output:


MARKS
100
90
80
70
60
50
40
30
20
Science mark
20
40
60
80
100
Maths mark

Program No.11
Scatter plot

Aim: Write a Python program to draw a scatter plot comparing two subject marks of Mathematicsand Science. Use marks of 10 students.
math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]

Algorithm:

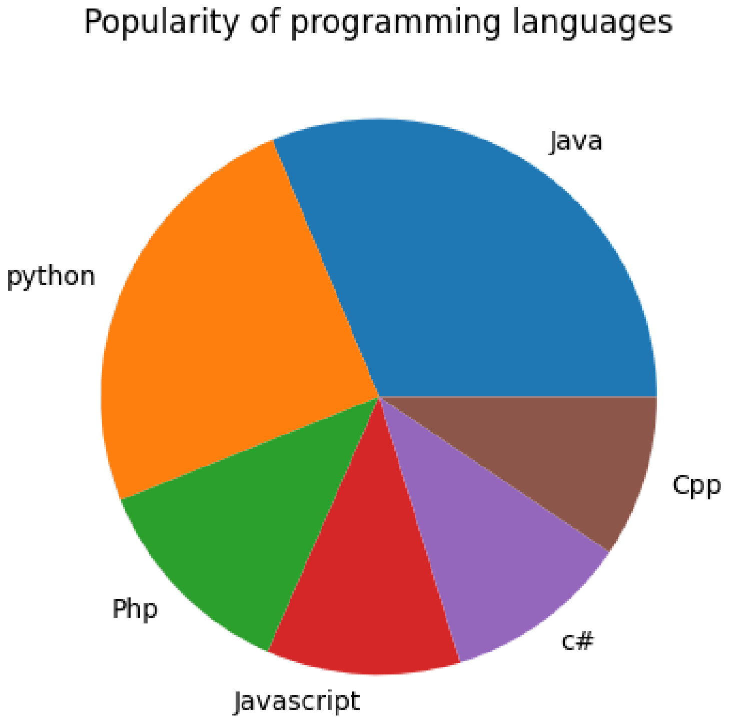
1. Import Necessary Modules: Import the required library, matplotlib.pyplot as plt for plotting.
2. Set Up the Data: Define the math_marks, science_marks, and marks_range lists.
3. Set Up the Plot: Use plt.scatter to create a scatter plot with math_marks on the x-axis and science_marks on the y-axis.
4. Set Up Labels and Title: Provide labels for the x-axis, y-axis, and a title for the plot.
5. Customize the Plot: Add grid lines, set ticks on both axes using plt.grid, plt.xticks, and plt.yticks.
6. Display the Plot: Use plt.show() to display the plot on the screen.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([88,92,18,89,100,80,60,100,80,34])
y=np.array([35,79,79,48,100,88,32,45,20,30])
plt.title('Scatter Plot: Mathematics vs Science')
plt.grid(True)
plt.scatter(x,y)
plt.xlabel("Maths mark")
plt.ylabel("Science mark")
plt.title("MARKS")
plt.show()
```

Result: Program executed successfully and output verified.

Output:


Popularity of programming languages
Java
python
Cpp
Php
c#
Javascript

Program No.12
Pie chart

Aim: Write a Python programming to create a pie chart of the popularity of programmingLanguages.

Programming languages:	Java	Python	PHP	JavaScript	C#	C++
Popularity	22.2	17.6	8.8	8	7.7	6.7

Algorithm:

1. Import Necessary Modules: Import the required library, matplotlib.pyplot as plt for plotting.
2. Set Up the Data: Define the programming languages and their popularity percentages.
3. Set Up the Plot: Use plt.pie to create a pie chart with popularity percentages, labels, and additional formatting parameters like autopct (to display percentage labels) and startangle.
4. Set Up Title: Provide a title for the pie chart.
5. Display the Plot: Use plt.show() to display the pie chart on the screen.

Program:

```
import numpy as np
importmatplotlib.pyplot as plt
x=["Java","Python","Php","Javascript","c#","Cpp"]
y=np.array([22.2,17.6,8.8,8,7.7,6.7])
plt.pie(y,labels=x)
plt.title("Popularity of programming languages")
plt.show()
```

Result: Program executed successfully and output verified.

Output:

```
0    a
1    b
2    c
dtype: object
```

EXPERIMENT NO.4
INTRODUCTION TO PANDAS

Program No.13
List-to-Series Conversion

Aim: Write a python program to implement List-to-Series Conversion

Algorithm:

1. Import Library: Import the pandas library as pd.
2. Prepare Data: Define a list of values.
3. Create Series: Use pd.Series() to convert the list into a pandas Series.
4. Display Series: Print the Series using the print() function.

Program:

```
import pandas as pd
names = ['a','b','c']
x = pd.Series(names)
print(x)
```

Result: Program executed successfully and output verified.

Output:

```
 Name  Age
0    a   24
1    b   25
2    c   26
3    d   27
```

Program No.14
Dictionary into corresponding dataframe

Aim: Write a python program to convert the given a dictionary into corresponding dataframe anddisplay it.

Algorithm:

1. Import Library: Import the pandas library as pd.
2. Prepare Data: Define a dictionary with column names as keys and lists of values as their corresponding data.
3. Create DataFrame: Use pd.DataFrame() to convert the dictionary into a pandas DataFrame.
4. Display DataFrame: Print the DataFrame using the print() function.

Program:

```
import pandas as pd
details = {
'Name' : ['a','b','c','d'],
'Age' : [24,25,26,27],
}
df = pd.DataFrame(details)
print(df)
```

Result: Program executed successfully and output verified.

Output:

```
  Name  Age
0    a   24
1    b   25
```

Program No.15
Select first 2 rows and output from a given a dataframe

Aim: Write a python program to select first 2 rows and output them from a given a dataframe.

Algorithm:

1. Import Library: Import the pandas library as pd.
2. Prepare Data: Define a dictionary with keys as column names and values as lists of data.
3. Create DataFrame: Use pd.DataFrame(data) to convert the dictionary into a pandas DataFrame.
4. Slice Rows: Use slicing syntax df[:] to select specific rows.
5. Display Result: Print the sliced DataFrame using the print() function.

Program:

```
import pandas as pd
details = {
'Name' : ['a','b','c','d'],
'Age' : [24,25,26,27],
}
df = pd.DataFrame(details)
print(df[0:2])
```

Result: Program executed successfully and output verified.

Output:

```
name\tclass
0       a\t1
1       b\t2
2       c\t3
```

Program No.16
Read the given CSV file, and convert it into a dataframe

Aim: Write a python program to read the given CSV file, and convert it into a dataframe and display it.

Algorithm:

1. Import Library: Import the pandas library as pd.
2. Read CSV File: Use pd.read_csv('filename.csv') to read the contents of a CSV file into a DataFrame.
3. Display Data: Print the DataFrame using the print() function.

Program:

```
import pandas as pd
df = pd.read_csv('pandas.csv')
print(df)

CSV:
name   class
a      1
b      2
c      3
```

Result: Program executed successfully and output verified.

Output:

```
Accuracy :  0.9777777777777777
Enter The Sample Data
Enter Sepal Length In CM : 4
Enter Sepal width In CM : 2
Enter Petal Length In CM : 1
Enter Petal width In CM : .2
[0]
['setosa']


Accuracy :  0.9777777777777777
Enter The Sample Data
Enter Sepal Length In CM : 6
Enter Sepal width In CM : 2
Enter Petal Length In CM : 4
Enter Petal width In CM : 1
[1]
['versicolor']



Accuracy :  0.9777777777777777
Enter The Sample Data
Enter Sepal Length In CM : 6
Enter Sepal width In CM : 3
Enter Petal Length In CM : 5
Enter Petal width In CM : 2
[2]
['virginica']
```

EXPERIMENT NO.5
PROGRAMS USING DATASCIENCE

Program No.17
K-NN classification using any standard dataset

Aim: Write a python program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.

Algorithm:

1. Import required libraries.
2. Load Iris dataset.
3. Extract features (x) and target labels (y) from the dataset.
4. Use train_test_split() to split the dataset into training (x_train, y_train) and testing sets (x_test, y_test).
5. Set the test size to 30% (test_size=0.3) and random_state=1.
6. Create an instance of KNeighborsClassifier with n_neighbors=3.
7. Train the classifier using the training data (x_train, y_train) with the fit() method.
8. Use the predict() method on x_test to make predictions.
9. Calculate and display the accuracy using metrics.accuracy_score().
10. Prompt the user to enter the features of a flower sample
11. Use the predict() method to classify the input sample.
12. Map the numeric prediction to the corresponding class name using iris.target_names.
13. Display Results.

Program:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
c_knn=KNeighborsClassifier(n_neighbors=3)
c_knn.fit(x_train,y_train)
y_pred=c_knn.predict(x_test)
print(y_pred)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
print("enter the sample data:")
a=float(input("enter the sepal length in cm:"))
b=float(input("enter the sepal width in cm:"))
c=float(input("enter the petal length in cm:"))
d=float(input("enter the petal width in cm:"))
sample=[[a,b,c,d]]
pred=c_knn.predict(sample)
pred_v=[iris.target_names[p] for p in pred]
print(pred)
print(pred_v)
```

Result: Program executed successfully and output verified.

Output:

```
Accuracy :  0.9333333333333333
Enter The Sample Data
Enter Sepal Length In CM : 5
Enter Sepal width In CM : 4
Enter Petal Length In CM : 1
Enter Petal width In CM : .2
[0]
['setosa']
```

```
Accuracy :  0.9333333333333333
Enter The Sample Data
Enter Sepal Length In CM : 6
Enter Sepal width In CM : 2
Enter Petal Length In CM : 4
Enter Petal width In CM : 1
[1]
['versicolor']
```

```
Accuracy :  0.9333333333333333
Enter The Sample Data
Enter Sepal Length In CM : 6
Enter Sepal width In CM : 3
Enter Petal Length In CM : 5
Enter Petal width In CM : 2
[2]
['virginica']
```

Program No.18
Naïve Bayes Algorithm using any standard dataset

Aim: Write a python program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

Algorithm:

1. Import required libraries, including load_iris from sklearn.datasets, train_test_split from sklearn.model_selection, GaussianNB from sklearn.naive_bayes, and metrics from sklearn.
2. Load Iris dataset.
3. Extract features (x) and target labels (y) from the dataset.
4. Use train_test_split() to split the dataset into training (x_train, y_train) and testing sets (x_test, y_test).
5. Set the test size to 30% (test_size=0.3) and random_state=1.
6. Create an instance of the GaussianNB classifier.
7. Train the classifier using the fit() method on the training data (x_train, y_train).
8. Use the predict() method on x_test to make predictions.
9. Calculate and display the accuracy using metrics.accuracy_score().
10. Prompt the user to enter the features of a flower sample (sepal length, sepal width, petal length, petal width).
11. Store the input as a feature array.
12. Use the predict() method to classify the input sample.
13. Map the numeric prediction to the corresponding class name using iris.target_names.
14. Print the numeric prediction.
15. Print the corresponding class name.

Program:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
iris = load_iris()
x = iris.data
y = iris.target
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
gnb=GaussianNB()
gnb.fit(x_train,y_train)
y_pred = gnb.predict(x_test)
print("Naive bayes iris prediction")
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))
print("Enter the sample data")
a=float(input("Enter sepal length in cm = "))
b=float(input("Enter sepal width in cm = "))
c=float(input("Enter petal length in cm = "))
d=float(input("Enter petal width in cm = "))
sample = [[a,b,c,d]]
pred = gnb.predict(sample)
pred_v = [iris.target_names[p] for p in pred]
print(pred_v)
```

Result: Program executed successfully and output verified.

Output:

```
Enter BMI :24
[22670.62753186]
Coefficients:
 [938.23786125]
Coeficient of determination:0.47
```

Program No.19
Linear regression techniques using any standard dataset

Aim: Write a python program to implement linear regression techniques using any standard dataset available in the public domain and evaluate its performance.

Algorithm:

1. Import necessary libraries.
2. Load the Iris dataset using load_iris().
3. Extract the feature data (x) and target labels (y).
4. Use train_test_split() to divide the dataset into training and testing sets.
5. Set test_size=0.3 (30% data for testing) and random_state=1 for reproducibility.
6. Create an instance of the LinearRegression model.
7. Train the model using the fit() method on the training data (x_train, y_train).
8. Use the predict() method on the test data (x_test) to generate predictions (y_pred).
9. Calculate the Mean Squared Error (MSE) using mean_squared_error(y_test, y_pred).
10. Calculate the R² Score using r2_score(y_test, y_pred).
11. Print the MSE and R² score to evaluate the model's performance.

Program:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error,r2_score
df=datasets.load_diabetes()
df['feature_names']
diabetes_X,diabetes_y=datasets.load_diabetes(return_X_y=True)
diabetes_X
diabetes_X=diabetes_X[:,np.newaxis,2]
diabetes_X.shape
diabetes_X
diabetes_X_train=diabetes_X[:-20]
diabetes_X_test=diabetes_X[-20:]
diabetes_X_test
diabetes_y_train=diabetes_y[:-20]
diabetes_y_test=diabetes_y[-20:]
diabetes_y_test
model=linear_model.LinearRegression()
model.fit(diabetes_X_train,diabetes_y_train)
diabetes_y_pred=model.predict(diabetes_X_test)
bmi=float(input("Enter BMI :"))
s6=[[bmi]]
result=model.predict(s6)
print(result)
print("Coefficients:\n",model.coef_)
print("Coeficient of determination:%.2f"%r2_score(diabetes_y_test,diabetes_y_pred))
```

Result: Program executed successfully and output verified.

Program No.20
Multiple regression techniques using any standard dataset

Aim: Write a python program to implement multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

Algorithm:

1. Import matplotlib.pyplot as plt, numpy as np, datasets, linear_model from sklearn, and mean_squared_error, r2_score from sklearn.metrics.
2. Load and inspect the diabetes dataset:
3. Use datasets.load_diabetes() to load the diabetes dataset.
4. Display the feature names in the dataset.
5. Extract features and target variable:
6. Use datasets.load_diabetes(return_X_y=True) to obtain both the features (diabetes_X) and the target variable (diabetes_y).
7. Select a single feature (BMI):
8. Choose a specific feature (e.g., BMI) by selecting the corresponding column of diabetes_X.
9. Shape of the feature matrix:
10. Display the shape of the feature matrix (diabetes_X).
11. Split the dataset into training and testing sets:
12. Use array slicing to split diabetes_X and diabetes_y into training and testing sets.
13. Create a Linear Regression model:
14. Initialize a Linear Regression model (model) using linear_model.LinearRegression().
15. Train the model on the training set:
16. Use fit() to train the model on the training data (diabetes_X_train and diabetes_y_train).
17. Make predictions on the test set:
18. Use predict() to make predictions on the test set (diabetes_X_test).
19. Take user input for feature(s):
20. Prompt the user to enter values for the chosen feature(s).
21. Make prediction for user-entered data:
22. Create a sample (user_sample) with the user-entered feature values and use the trained model to predict the target variable.
23. Print the prediction for user-entered data:
24. Print the predicted target variable for the user-entered data.
25. Print model coefficients:
26. Print the coefficients of the linear regression model.
27. Print the coefficient of determination (R-squared):
28. Print the coefficient of determination using r2_score().

Program:

```
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
diabetes = datasets.load_diabetes()
diabetes_X, diabetes_y = diabetes.data, diabetes.target
diabetes_X = diabetes_X[:, [0, 2, 3]]
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
```

Output:

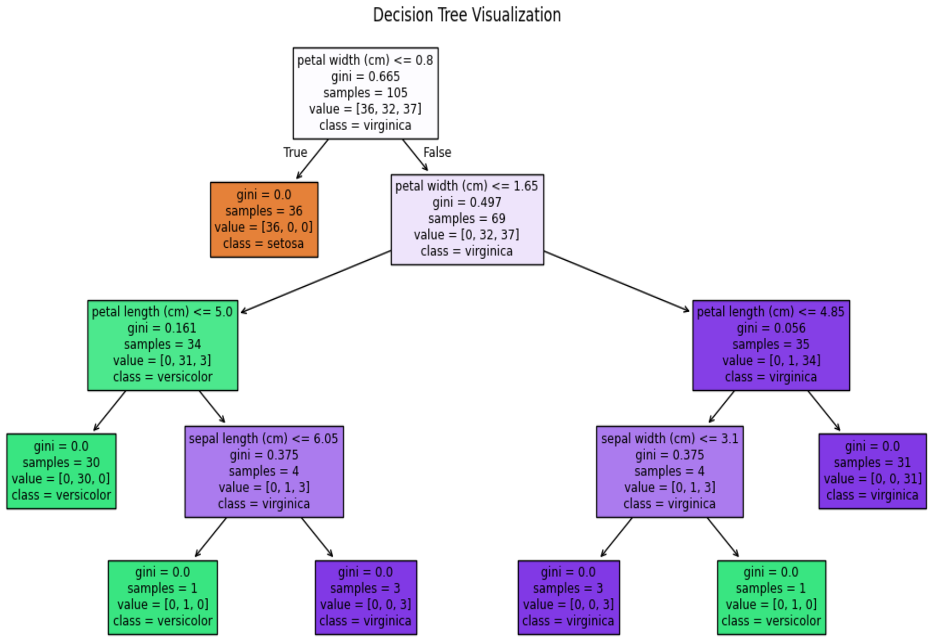
```
Coefficients:
 [ 34.06320066 778.7780595  399.92057897]
Intercept:
 152.85391932400353
Mean squared error: 2605.59
Coefficient of determination: 0.46
Enter BMI: 24
Enter Blood Pressure: 110
Enter Cholesterol: 150
Predicted diabetes progression: [146624.04412508]
```

```
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
print("Coefficients: \n", regr.coef_)
print("Intercept: \n", regr.intercept_)
diabetes_y_pred = regr.predict(diabetes_X_test)
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test,
diabetes_y_pred))
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test, diabetes_y_pred))
bmi = float(input("Enter BMI: "))
bp = float(input("Enter Blood Pressure: "))
ldl = float(input("Enter Cholesterol: "))
s6 = np.array([[bmi, bp, ldl]])
result = regr.predict(s6)
print("Predicted diabetes progression:", result)
```

Result: Program executed successfully and output verified.

Output:

Accuracy : 0.9555555555555556



Program No.21

Decision trees using any standard dataset

Aim: Write a python program to implement decision trees using any standard dataset available inthe public domain and find the accuracy of the algorithm.

Algorithm:

```
1. Use load_iris() to load the Iris dataset.
2. Extract features (x) and target labels (y).
3. Divide the dataset into training and testing sets using train_test_split().
4. Set test_size=0.3 (30% of data for testing) and random_state=1 for reproducibility.
5. Create a DecisionTreeClassifier instance.
6. Train the classifier using the fit() method on the training data (x_train, y_train).
7. Use the trained model's predict() method to predict the labels for the test data (x_test).
   Calculate the accuracy using metrics.accuracy_score(y_test, y_pred).
8. Print the accuracy.
9. Prompt the user to input sample data (sepal length, sepal width, petal length, petal width).
10. Combine the input values into a feature array (sample).
11. Use the trained model to predict the class of the sample.
12. Map the numeric prediction to the class name using iris.target_names.
13. Use tree.plot_tree() to visualize the decision tree.
14. Display feature names and class names for clarity.
15. Show the visualization using plt.show().
```

Program:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))

from sklearn import tree
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
tree.plot_tree(dt, filled=True, feature_names=iris.feature_names,
class_names=iris.target_names)
plt.title("Decision Tree Visualization")
plt.show()
```

Result: Program executed successfully and output verified.

Program No.22
Image classification using Support vector machine

Aim: Write a python program to implement image classification using Support vector machine.

Algorithm:

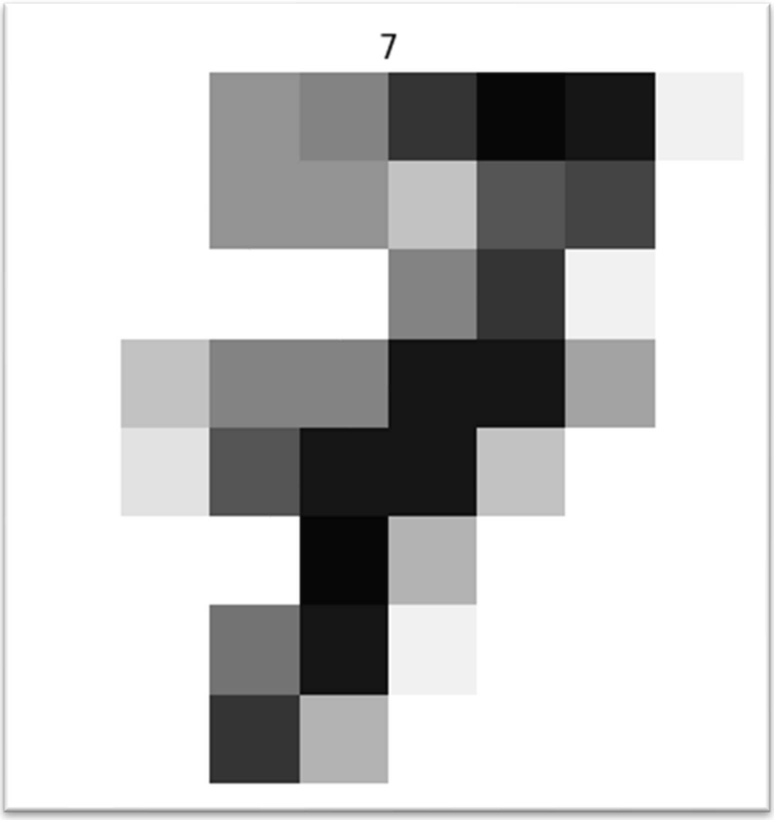
1. Import necessary libraries:
2. Import numpy as np, load_digits from sklearn.datasets, matplotlib.pyplot as plt, train_test_split from sklearn.model_selection, svm from sklearn, and accuracy_score fromsklearn.metrics.
3. Load the digits dataset:
4. Use load_digits() to load the digits dataset.
5. Display dataset information:
6. Print the data, target, shape of data, shape of images, and the length of the images inthe dataset.
7. Reshape images for processing:
8. Reshape the images into a 2D array.
9. Split the dataset into training and testing sets:
10. Use train_test_split to split the dataset into training and testing sets.
11. Create a Support Vector Machine (SVM) model:
12. Initialize an SVM classifier (model) using svm.SVC().
13. Train the model on the training set:
14. Use fit() to train the model on the training data.
15. Make predictions on the test set:
16. Use predict() to make predictions on the test set.
17. Calculate and print the accuracy of the model:
18. Use accuracy_score() to calculate and print the accuracy of the model on the testset.
19. Take user input for a specific digit:
20. Prompt the user to enter a value corresponding to a digit in the dataset.
21. Make predictions for the user-entered digit:
22. Use the trained model to predict the class of the user-entered digit.
23. Display the user-entered digit and its prediction:
24. Display the image of the user-entered digit using plt.imshow().
25. Print the predicted result and visualize it using plt.title() and plt.show().

Program:

```
import numpy as np
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
datasets=load_digits()
print(datasets.data)
print(datasets.target)
print(datasets.data.shape)
print(datasets.images.shape)
dataimageLength=len(datasets.images)
print(dataimageLength)
x=datasets.images.reshape((dataimageLength,-1))
y=datasets.target
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=0)
print(x_train.shape)
print(x_test.shape)
```

Output:

```
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
[0 1 2 ... 8 9 8]
(1797, 64)
(1797, 8, 8)
1797
(1347, 64)
(450, 64)
Accuracy of the model:99.11111111111111%
Enter the value 7
[7]
```



```
from sklearn import svm
model=svm.SVC()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
from sklearn.metrics import accuracy_score
print("Accuracy of the model:{0}%".format(accuracy_score(y_test,y_pred)*100))
n=int(input("Enter the value "))
result = model.predict(datasets.images[n].reshape((1, -1)))
plt.imshow(datasets.images[n], cmap=plt.cm.gray_r, interpolation='nearest')
print(result)
plt.axis('off')
plt.title('%i' % result[0])
plt.show()
```

Result: Program executed successfully and output verified.

Output:

```
Centroids for Cluster 1:
[5.9016129  2.7483871  4.39354839 1.43387097]
Centroids for Cluster 2:
[5.006 3.428 1.462 0.246]
Centroids for Cluster 3:
[6.85       3.07368421 5.74210526 2.07105263]
Enter the sample data
Enter sepal length in cm: 5
Enter sepal width in cm: 3
Enter petal length in cm: 1
Enter petal width in cm: .2
The input belongs to Cluster: 1
```

Program No.23
K-means clustering technique using standard dataset

Aim: Write a python program to implement k-means clustering technique using any standard dataset available in the public domain.

Algorithm:

```
1. Load the Iris dataset using load_iris().
2. Extract the feature data (x) from the dataset.
3. Create a KMeans instance with:
    a. n_clusters=3 (since there are 3 species in the Iris dataset).
    b. init='k-means++' (to initialize centroids efficiently).
    c. random_state=42 (for reproducibility).
    d. n_init=10 (number of initializations runs).
4. Use the fit() method on the feature data (x) to perform clustering.
5. Retrieve the cluster centroids using the cluster_centers_ attribute.
6. Print the centroids for each of the 3 clusters.
7. Prompt the user to input values for sepal length, sepal width, petal length, and petal width.
8. Combine the input values into a NumPy array (sample).
9. Use the predict() method to determine which cluster the input data belongs to.
10. Print the predicted cluster.
```

Program:

```
from sklearn.datasets import load_iris
import numpy as np
from sklearn.cluster import KMeans
iris = load_iris()
x = iris.data
# Set n_clusters to 3 since the Iris dataset has 3 species
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42, n_init=10)
kmeans.fit(x)
centroids = kmeans.cluster_centers_
print("Centroids for Cluster 1:")
print(centroids[0])
print("Centroids for Cluster 2:")
print(centroids[1])
print("Centroids for Cluster 3:")
print(centroids[2])
print("Enter the sample data")
a = float(input("Enter sepal length in cm: "))
b = float(input("Enter sepal width in cm: "))
c = float(input("Enter petal length in cm: "))
d = float(input("Enter petal width in cm: "))
sample = np.array([[a, b, c, d]])
# Predict the cluster for the input values
predicted_cluster = kmeans.predict(sample)
print("The input belongs to Cluster:", predicted_cluster[0])
```

Result: Program executed successfully and output verified.

Output:

```
POS Tags: [('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sample', 'JJ'), ('sentence', 'NN'), ('for', 'IN'), ('POS',
'NNP'), ('tagging', 'NN'), ('.', '.')]
```

EXPERIMENT NO.6
PROGRAMS USING NATURAL LANGUAGE PROCESSING

Program No.24
Part of Speech tagging

Aim: Write a python program to implement Part of Speech tagging.

Algorithm:

1. Import the nltk library.
2. Download the 'punkt' resource from NLTK.
3. Download the 'stopwords' resource from NLTK.
4. Download the 'wordnet' resource from NLTK.
5. Download the 'omw-1.4' resource from NLTK.
6. Import the word_tokenize function from nltk.tokenize.
7. Import the ngrams function from nltk.
8. Define the sample text to be processed.
9. Tokenize the text into words using word_tokenize.
10. Define a function named generate_ngrams to generate n-grams from a list of tokens.
11. Specify the value of 'n' for n-grams (e.g., 5 grams in this case).
12. Generate n-grams using the defined function generate_ngrams.
13. Print the generated n-grams

Program:

```
import nltk
nltk.download('punkt')
 nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
text = "This is a sample sentence for generating n-grams."
tokenized_text = word_tokenize(text)
print(tokenized_text)
from nltk import ngrams
def generate_ngrams(tokens, n):
return list(ngrams(tokens, n))
n_value = 3 # Change this value to generate different n-grams (e.g., bigrams, trigrams)
ngrams_list = generate_ngrams(tokenized_text, n_value)
print(f"{n_value}-grams:", ngrams_list)
```

Result: Program executed successfully and output verified.

Output:

```
['This', 'is', 'a', 'sample', 'sentence', 'for', 'generating', 'n-grams', '.']
5-grams: [('This', 'is', 'a', 'sample', 'sentence'), ('is', 'a', 'sample', 'sentence', 'for'), ('a', 'sample', 'sentence', 'for', 'generating'), ('sample', 'sentence', 'for', 'generating', 'n-grams'), ('sentence', 'for', 'generating', 'n-grams', '.')]
```

Program No.25
N-gram

Aim: Write a python program to implement N-gram.

Algorithm:

1. Import the nltk library.
2. Download the 'punkt' resource from NLTK.
3. Download the 'stopwords' resource from NLTK.
4. Download the 'wordnet' resource from NLTK.
5. Download the 'omw-1.4' resource from NLTK.
6. Import the word_tokenize function from nltk.tokenize.
7. Import the ngrams function from nltk.
8. Define the sample text to be processed.
9. Tokenize the text into words using word_tokenize.
10. Define a function named generate_ngrams to generate n-grams from a list of tokens.
11. Specify the value of 'n' for n-grams (e.g., 5 grams in this case).
12. Generate n-grams using the defined function generate_ngrams.
13. Print the generated n-grams

Program:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
text = "This is a sample sentence for generating n-grams."
tokenized_text = word_tokenize(text)
print(tokenized_text)
from nltk import ngrams
def generate_ngrams(tokens, n):
 return list(ngrams(tokens, n))
n_value = 3  # Change this value to generate different n-grams (e.g., bigrams, trigrams)
ngrams_list = generate_ngrams(tokenized_text, n_value)
print(f"{n_value}-grams:", ngrams_list)
```

Result: Program executed successfully and output verified.