Result and Observations(Inc UI)

```
Original array is [1 2 3 4 5 6 7 8 9]
dimension is (9,)
changing shape to 3,3
shape (3, 3)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

EXERCISE 1: INTRODUCTON TO NUMPY

CHANGING DIMENSION OF AN ARRAY

Program No:1 **Date:** 29-08-2022

AIM

Change dimension of an array and convert the numpy array into a list.

Theoretical Support

numpy.array() - Create an array by using the array() funcrton.

shape() – It returns a tuple with each index having the number of corresponding elements.

tolist() - changes numpy scalars to Python scalars.

Code

```
ar=np.array([1, 2, 3, 4,5,6,7,8,9])
print('Original array is',ar);
print("dimension is",ar.shape)
print('changing shape to 3,3')
ar.shape = (3, 3)
print("shape",ar.shape)
print(ar)
ls = ar.tolist()
type(ls)
print(ls)
```

Inference

In this an array is input using array() and then convert the array into list using tolist().

Result and Observations(Inc UI)

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[ 0.  0.  0.  0.  0. 11.  0.  0.  0.  0.]
```

CREATE AND UPDATE A NULL NUMPY ARRAY

Program No: 2 **Date:** 29-08-2022

AIM

Create a null numpy array of size 10 and update the sixth value to 11.

Theoretical Support

numpy.zero() - Return a new array of given shape and type, filled with zeros.

Code

```
import numpy as np
a=np.zeros(10)
print(a)
a[5]=11
print(a)
```

Inference

In this it create an array which contains zeros and convert 6 th value to 11.

Result and Observations(Inc UI)

```
Array A
[6 5 7 1 9]

Array B
[1 3 2 5 7]

A>B
[ True  True  True False  True]

A>=B
[ True  True  True False  True]

A<B
[False False False  True False]

A<=B
[False False False  True False]
```

ELEMENTWISE COMPARISON OF ARRAYS

Program No: 3 **Date:** 29-08-2022

AIM:

Write a NumPy program to create an element-wise comparison (greater, greater_equal, less and less_equal) of two given arrays.

Theoretical Support

numpy.array() - Create an array by using the array() funcrton.

numpy.greater() – To compare and return True if an array is greater than other one.

numpy.greater_equal() - checks weather the elements in a given array (first argument) is greater than or equal to a specified number(second argument).

numpy.less() – Function in python is used to check ,one by one , if the elements of array x are less than the elements of another array y that is of the same shape.

numpy.less_equal() -- checks weather the elements in a given array (first argument) is less than or equal to a specified number(second argument).

Code

```
import numpy as np

x = np.array([6,5,7,1,9])

y = np.array([1,3,2,5,7])

print("Array A")

print(x)

print("\nArray B")

print(y)

print("\nA>B")
```

```
print(np.greater(x, y))

print("\nA>=B")

print(np.greater_equal(x, y))

print("\nA<B")

print(np.less(x, y))

print("\nA<=B")

print(np.less_equal(x, y))
```

Inference

The numpy comparison functions helps to find the element-wise comparison such as equal,greater,greater than,lesser,lesser than of each elements in two arrays. Without using any loops.

Results and Observations (incl. UI)

```
[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]
```

ARRAY CREATION

Program No.: 4 **Date:** 29-08-2022

AIM:

Write a NumPy program to create an array of all the even integers from 30 to70.

Theoretical Support

numpy.arrange()-Returns an array with evenly spaced elements as per the interval.

Code

```
import numpy as np

x = np.arange(30,71,2)

print(x)
```

Inference

The numpy arrange functions helps to create an array of all even integers.

Results and Observations (incl. UI)

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

IDENTITY MATRIX

Program No.: 5 **Date:** 02-09-2022

AIM

Write a NumPy program to create a 3x3 identity matrix.

Theoretical Support

numpy.identity()- Return the identity array. The identity array is a square array with ones on the main diagonal.

Code

```
import numpy as np

ma = np.identity(3)

print(ma)
```

Inference

The numpy identity function helps to create an identity matrix.

Results and Observations (incl. UI)

```
Array:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

After loading, content of the text file:
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

ARRAY TO TEXT FILE

Program No.: 6 **Date:** 02-09-2022

AIM:

Write a NumPy program to save a given array to a text file and load it.

Theoretical Support

numpy.arrange()- To create evenly spaced values.

numpy.savetxt()- Save an array to a text file.

numpy.loadtxt()- Load data from a text file.

Code

```
import numpy as np
import os
x = np.arange(16).reshape(4,4)
print("Array:")
print(x)
header = 'C1 C2 C3 C4'
np.savetxt('7_array.txt', x, fmt="%d", header=header)
print("\nAfter loading, content of the text file:")
print(np.loadtxt('7_array.txt',dtype='i'))
```

Inference

The numpy function such as savetxt can be used to save array as text file and loadtxt functon used load that text file.

Results and Observations (incl. UI)

```
Array1: [1 2 3 4 5]
Array2: [1 4 5 2 5]
Equal and not equal
[ True False False False  True]
```

ARRAY EQUALITY

Program No.: 7 **Date:** 02-09-2022

AIM:

Write a NumPy program to checkwhether two arrays are equal (elementwise) or not.

Theoretical Support

numpy.array() - Create an array by using the array() funcrton.

numpy.equal() – Check whether two array are equal or not.

Code

```
import numpy as np

a1 = np.array([1,2,3,4,5])

a2 = np.array([1,4,5,2,5])

print("Array1:",a1)

print("Array2:",a2)

print("Equal and not equal")

print(np.equal(a1, a2))
```

Inference

The numpy functron equal can be used to find the arrays are equal.

Results and Observations (incl. UI)

```
Enter the elements of first matrix- 3 X 3
Enter the elements of second matrix- 3 X 3

First Matrix
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Second matrix
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

Dot product of first and second matrix

array([[ 30,  36,  42],
       [ 66,  81,  96],
       [102, 126, 150]])
```

EXERCISE 2: MATRIX OPERATION AND TRANSFORMATIONS

DOT PRODUCT OF MATRIX

Program No.: 8 **Date:** 05-09-2022

AIM:

Write a Python program to create two matrices (read values from user) and find Dot Product.

Theoretical Support

numpy.array() - Create an array by using the array() funcrton.

list() – List are used to store multiple items in a single variable.

numpy.dot() – Returns dot product of two matrices..

Code

```
import numpy as np

print("Enter the elements of first matrix- 3 X 3 ");

elements = list(map(int, input().split()));

a= np.array(elements).reshape(3,3);

print("Enter the elements of second matrix- 3 X 3 ");

elements = list(map(int, input().split()));

b= np.array(elements).reshape(3,3);

print("\nFirst Matrix\n",a)

print("Second matrix\n",b)

print("\nDot product of first and second matrix");

np.dot(a, b)
```

Inference

The numpy function such as dot() can be used to find the dot product of two matrices.

Result and Observations(Inc UI)

```
Enter the elements of first matrix- 2 X 2
Enter the elements of second matrix- 2 X 2

First Matrix
 [[1 2]
 [3 4]]

Transpose of first matrix
[[1 3]
 [2 4]]
Second matrix
 [[4 3]
 [2 1]]

Transpose of second matrix
[[4 2]
 [3 1]]
```

TRANSPOSE OF MATRIX

Program No.:9 **Date:** 05-09-2022

AIM

Write a Python program to create two matrices (read values from user) and find **Transpose**

Theoretical Support

numpy.array() - Create an array by using the array() function.

list() – List are used to store multiple items in a single variable.

numpy.traspose() – Returns transpose of matrix .

Code

```
import numpy as np
print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 2 X 2");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(2,2);
print("\nFirst Matrix\n",a);
print("\nTranspose of first matrix");
print(a.transpose())
print("Second matrix\n",b);
print("\nTranspose of second matrix");
print(b.transpose())
```

Inference

The numpy function such as transpose() can be used to find the transpose of a matrix.

Result and Observations(Inc UI)

```
Enter the elements of first matrix- 2 X 2
Enter the elements of second matrix- 2 X 2

First Matrix
 [[1 2]
 [3 4]]

Trace of first matrix 5
Second matrix
 [[3 2]
 [1 5]]

Trace of second matrix 8
```

TRACE OF MATRIX

Program No.: 10 **Date:** 09-09-2022

AIM

Write a Python program to create two matrices (read values from user) and find **Trace.**

Theoretical Support

numpy.array() - Create an array by using the array() function.

list() – List are used to store multiple items in a single variable.

numpy.trace() – Returns sum along diagonals of a matrix .

Code

```
import numpy as np
print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 2 X 2");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(2,2);
print("\nFirst Matrix\n",a);
print("\nTrace of first matrix",a.trace());
print("Second matrix\n",b);
print("\nTrace of second matrix",b.trace());
```

Inference

The numpy function such as trace() can be used to find the sum along diagonals of a matrix.

Result and Observations(Inc UI)

```
Enter the elements of first matrix- 2 X 2
Enter the elements of second matrix- 3 X 3

First Matrix
 [[1 2]
 [3 4]]

Rank of first matrix 2
Second matrix
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

Rank of second matrix 2
```

RANK OF MATRIX

Program No.: 11 **Date:** 09-09-2022

AIM

Write a Python program to create two matrices (read values from user) and find **Rank.**

Theoretical Support

numpy.array() - Create an array by using the array() function.

list() – List are used to store multiple items in a single variable.

numpy. linalg.matrix_rank () – Returns rank of a matrix .

Code

```
import numpy as np
print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 3 X 3");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(3,3);
print("\nFirst Matrix\n",a);
print("\nRank of first matrix",np.linalg.matrix_rank(a));
print("Second matrix\n",b);
print("\nRank of second matrix",np.linalg.matrix_rank(b));
```

Inference

The numpy function such as linalg.matrix_rank () can be used to find the rank of a matrix.

Result and Observations(Inc UI)

```
Enter the elements of first matrix- 2 X 2
Enter the elements of second matrix- 2 X 2

First Matrix
 [[1 1]
 [1 1]]

Determinant of first matrix 0.0
Second matrix
 [[1 2]
 [3 1]]

Determinant of second matrix -5.000000000000001
```

DETERMINANT OF MATRIX

Program No.: 12 **Date:** 09-09-2022

AIM

Write a Python program to create two matrices (read values from user) and find **Determinant.**

Theoretical Support

numpy.array() - Create an array by using the array() function.

list() – List are used to store multiple items in a single variable.

numpy.linalg.det () – Returns determinant of a matrix .

Code

```
import numpy as np
print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 2 X 2");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(2,2);
print("\nFirst Matrix\n",a);
print("\nDeterminant of first matrix",np.linalg.det(a));
print("Second matrix\n",b);
print("\nDeterminant of second matrix",np.linalg.det(b));
```

Inference

The numpy function such as numpy.linalg.det () can be used to find the determinant of a matrix.

Result and Observations(Inc UI)

```
Enter the elements of first matrix- 2 X 2
Enter the elements of second matrix- 2 X 2

First Matrix
 [[1 2]
 [4 5]]

Inverse of first matrix
[[-1.66666667  0.66666667]
 [ 1.33333333 -0.33333333]]
Second matrix
 [[1 2]
 [3 2]]

Inverse of second matrix
[[-0.5   0.5 ]
 [ 0.75 -0.25]]
```

INVERSE OF MATRIX

Program No.: 13 **Date:** 12-09-2022

AIM

Write a Python program to create two matrices (read values from user) and find **Inverse.**

Theoretical Support

numpy.array() - Create an array by using the array() function.

list() – List are used to store multiple items in a single variable.

numpy.linalg.inv() – Returns inverse of a matrix .

Code

```
import numpy as np
print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 2 X 2");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(2,2);
print("\nFirst Matrix\n",a);
print("\nInverse of first matrix");
print(np.linalg.inv(a));
print("Second matrix\n",b);
print("\nInverse of second matrix");
print(np.linalg.inv(b));
```

Inference

The numpy function such as numpy.linalg.inv() can be used to find the inverse of a matrix.

Result and Observations(Inc UI)

```
Enter the elements of first matrix- 2 X 2
Enter the elements of second matrix- 2 X 2

First Matrix
 [[1 2]
 [3 4]]

Eigen value of first matrix [-0.37228132  5.37228132]

Eigen vector of first matrix
 [[-0.82456484 -0.41597356]
 [ 0.56576746 -0.90937671]]
Second matrix
 [[4 3]
 [2 1]]

Eigen value of second matrix [ 5.37228132 -0.37228132]

Eigen vector of second matrix
 [[ 0.90937671 -0.56576746]
 [ 0.41597356  0.82456484]]
```

EIGEN VALUES AND EIGEN VECTORS

Program No.: 14 **Date:** 12-09-2022

AIM

Write a Python program to create two matrices (read values from user) and Eigen value and Eigen vector.

Theoretical Support

numpy.array() - Create an array by using the array() function.

list() – List are used to store multiple items in a single variable.

numpy.linalg.eig() -Returns eigen value and eigen vector of matrix.

Code

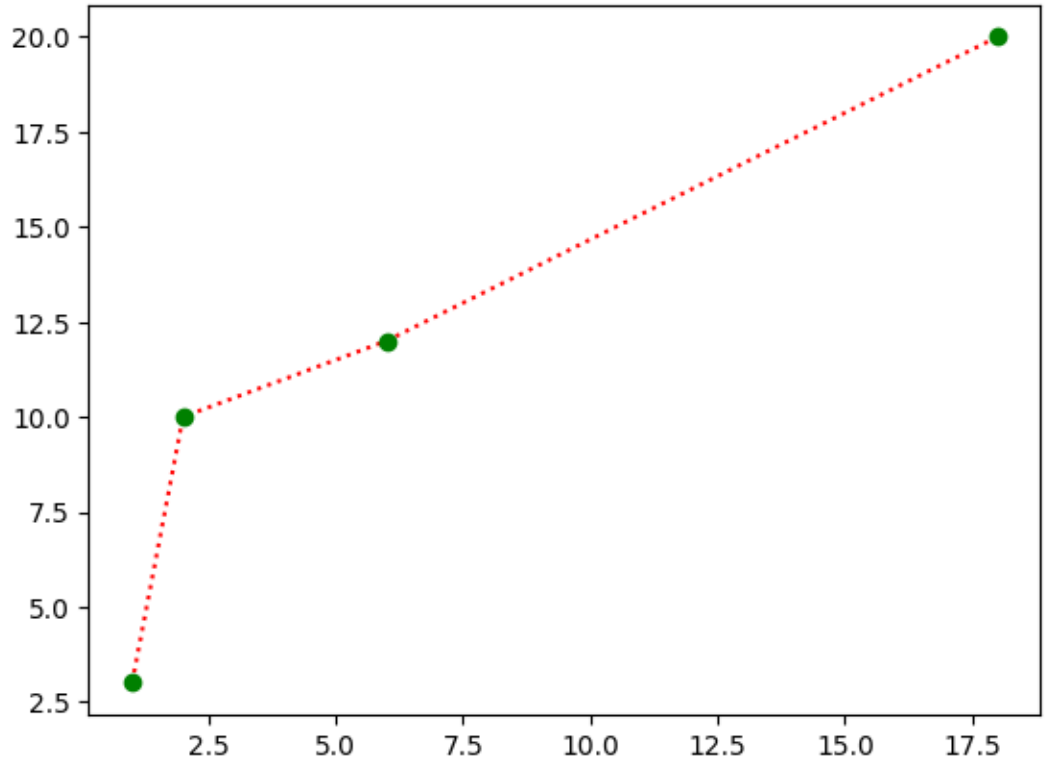
```
import numpy as np
print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 2 X 2");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(2,2);
print("\nFirst Matrix\n",a);
eval1,evect1 = np.linalg.eig(a)
eval2,evect2 = np.linalg.eig(b)
print("\nEigen value of first matrix",eval1);
```

```
print("\nEigen vector of first matrix\n",evect1);

print("Second matrix\n",b);

print("\nEigen value of second matrix",eval2);

print("\nEigen vector of second matrix\n",evect2);
```

Inference

The numpy functon linalg.eig() can be used to find the eigen value and eigen vector of the matrix.

Result and Observations(Inc UI)



EXERCISE 3: PROGRAMS USING MATPLOTLIB

LINE DIAGRAM

Program No.: 15 **Date:** 12-09-2022

AIM

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

Theoretical Support

numpy.array() - Create an array by using the array() function.

plot() -This function is used to draw points(markers) in a diagram.

show() – This function is used to display all diagrams.

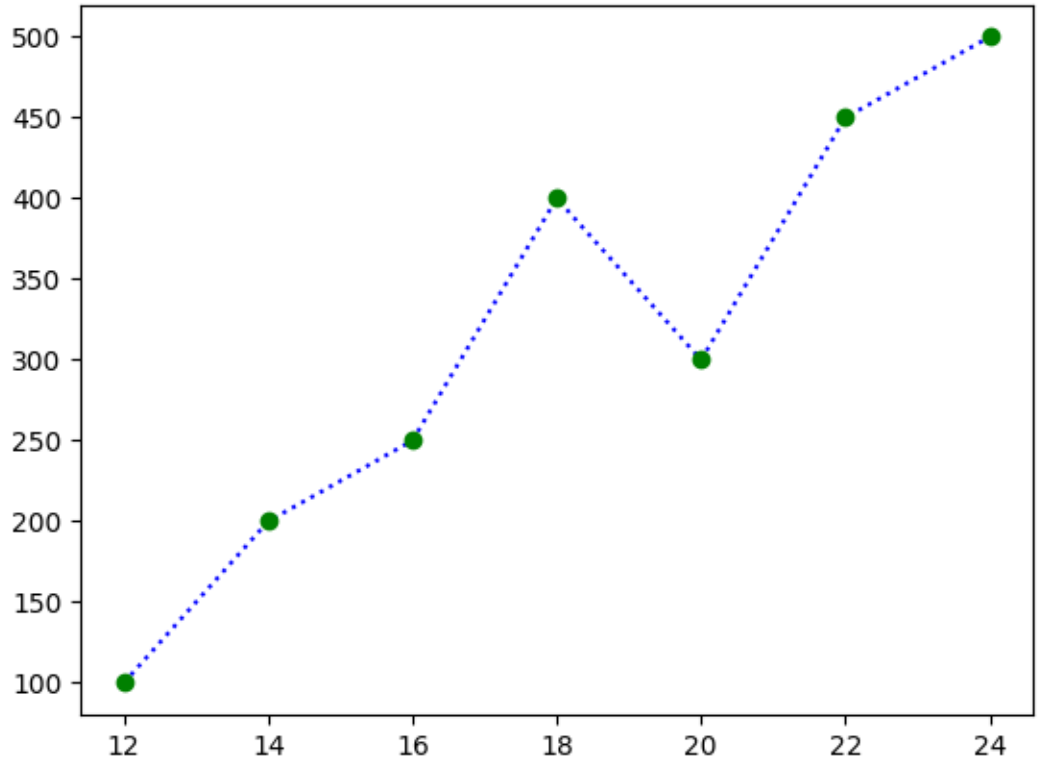
Code

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 2, 6, 18])
ypoints = np.array([3, 10, 12, 20])
plt.plot(xpoints, ypoints,marker = 'o',color="red",mec = 'g', mfc = 'g',linestyle = 'dotted')
plt.show()
```

Inference

We can create diagrams using the matplotlib function plot() and the diagram can be displayed using show().

Result and Observations(Inc UI)



PLOT FOR THE GIVEN DATA

Program No: 16 **Date:** 16-09-2022

Draw a plot for the following data:

Temperature in degree Celsius , Sales

12	100
14	200
16	250
18	400
20	300
22	450

Theoretical Support

numpy.array() - Create an array by using the array() function.

plot() -This function is used to draw points(markers) in a diagram.

show() – This function is used to display all diagrams.

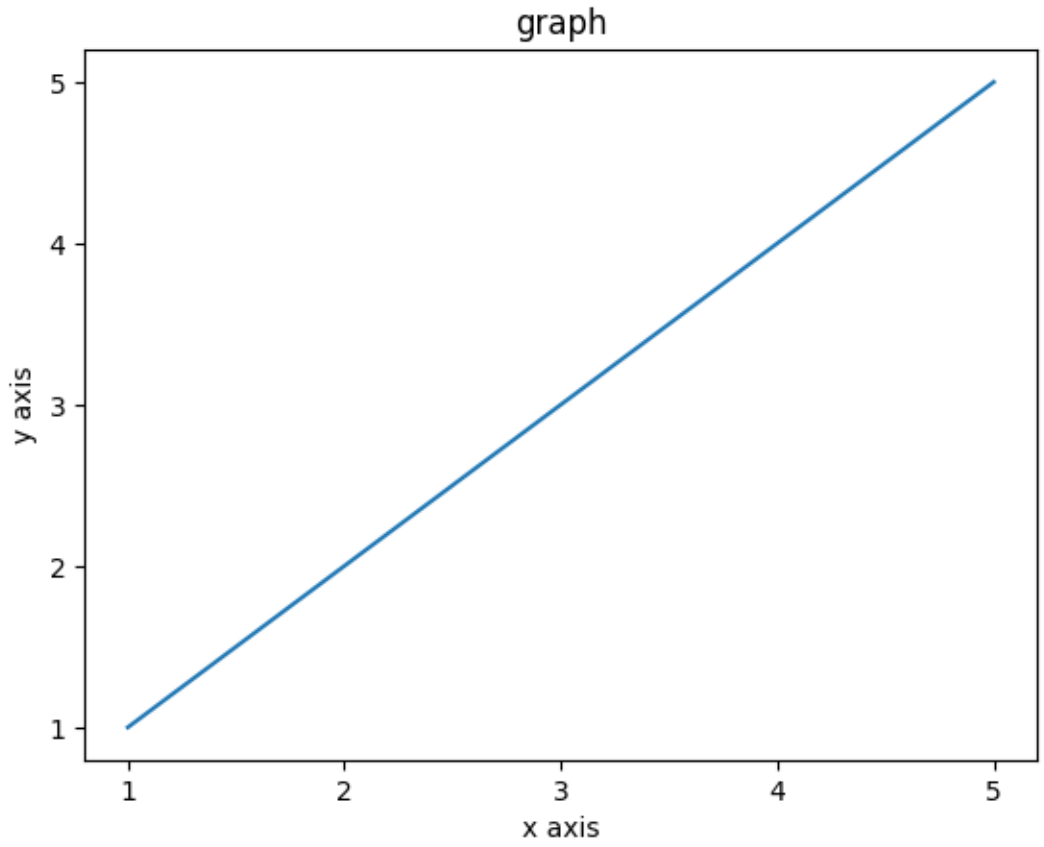
Code

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([12,14,16,18,20,22,24])
ypoints = np.array([100,200,250,400,300,450,500])
plt.plot(xpoints, ypoints,marker = 'o',color="blue",mec = 'g', mfc = 'g',linestyle = 'dotted')
plt.show()
```

Inference

We can create diagrams using the matplotlib function plot() and the diagram can be displayed using show().

Result and Observations(Inc UI)



LINE DIAGRAM USING TEXT FILE

Program No.: 17 **Date:** 16-09-2022

AIM

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

Theoretical Support

xlabel()- This function is used to set the label for x-axis.

ylabel()- This function is used to set the label for y-axis.

plot() -This function is used to draw points(markers) in a diagram.

show() – This function is used to display all diagrams.

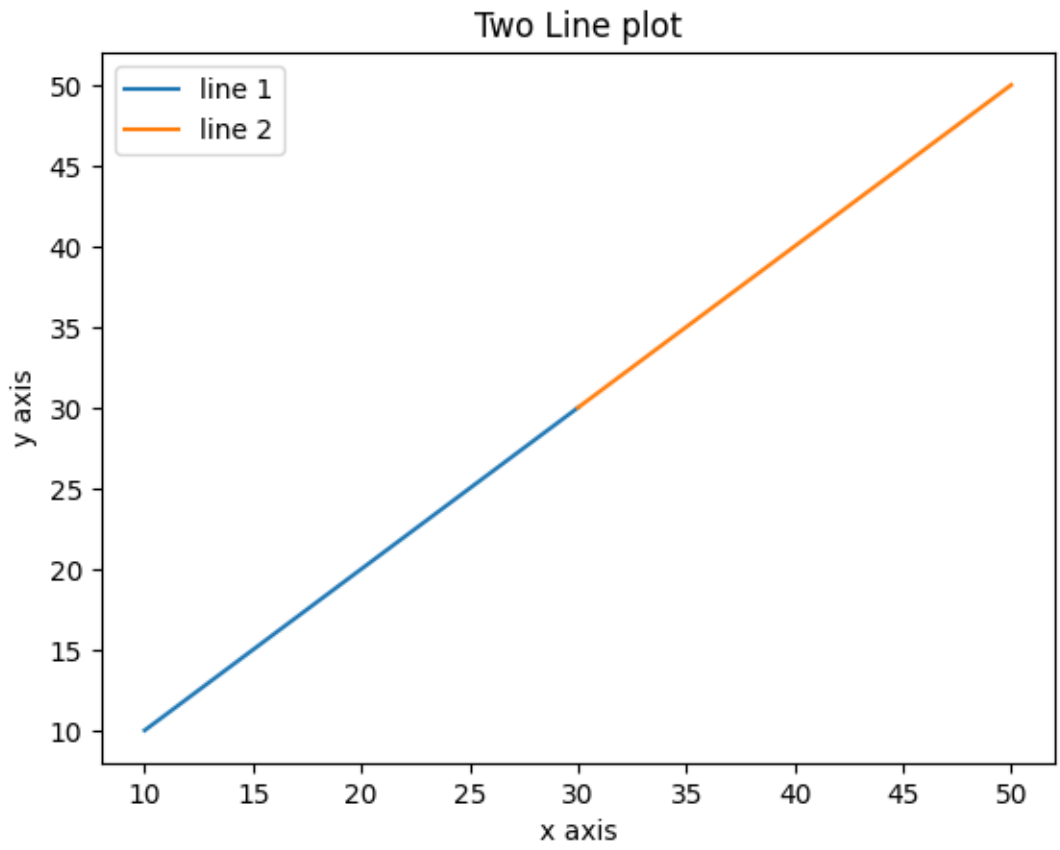
Code

```
import matplotlib.pyplot as plt
with open("programs/cycle-1/exersise3/data1.txt") as f: data = f.read()
    data = data.split('\n')
x = [row.split(' ')[0] for row in data]
y = [row.split(' ')[1] for row in data]
plt.plot(x, y)
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.title('graph')
plt.show()
```

Inference

We can draw a line using given axis values taken from a text file.

Result and Observations(Inc UI)



MULTIPLE LINES ON SAME PLOT

Program No: 18 **Date:** 16-09-2022

AIM

Write a Python program to plot two or more lines on same plot with suitable legends of each line.

Theoretical Support

xlabel()- This function is used to set the label for x-axis.

ylabel()- This function is used to set the label for y-axis.

plot() -This function is used to draw points(markers) in a diagram.

show() – This function is used to display all diagrams.

legend()- It is used to place legend in the axis.

Code

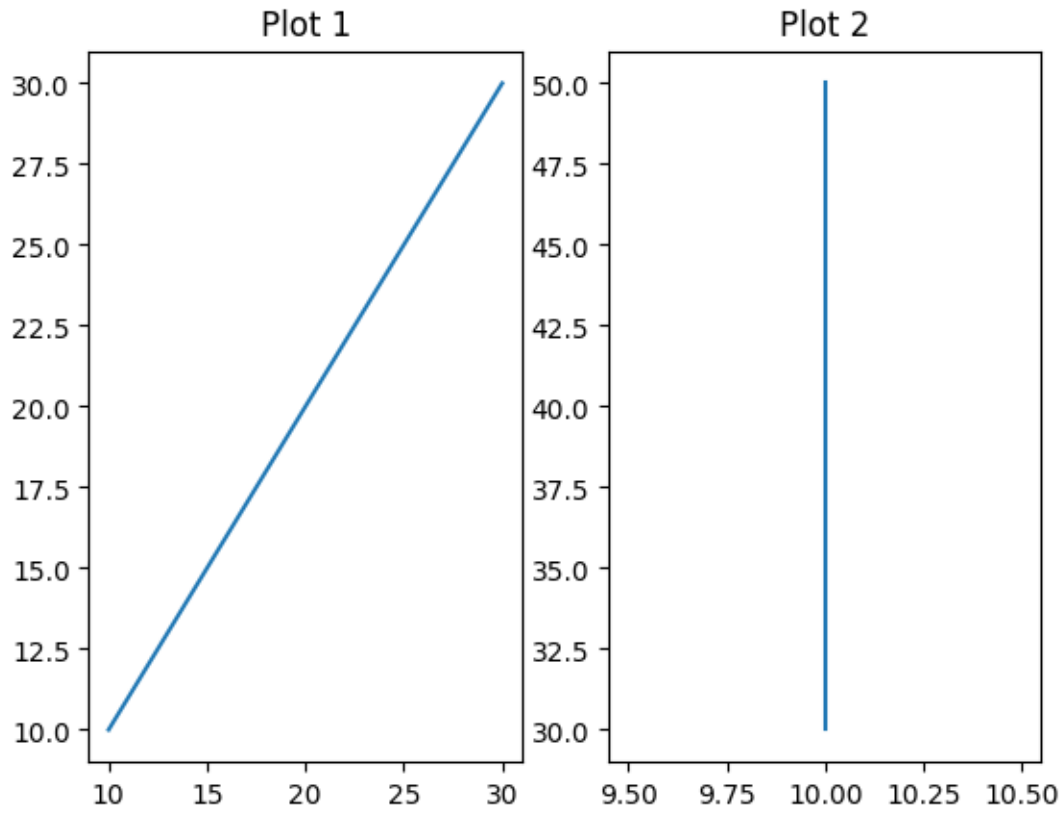
```
import matplotlib.pyplot as plt

x1 = [10,20,30]

y1 = [10,20,30]

plt.plot(x1, y1, label = "line 1")

x2 = [30,40,50]

y2 = [30,40,50]

plt.plot(x2, y2, label = "line 2")

plt.xlabel('x axis')

plt.ylabel('y axis')
```

```
plt.title('Two Line plot')

plt.legend()

plt.show()
```

Inference

We can plot two or more lines in a same plot using xlabel() and ylabel() and also plot suitable legends on it.

Result and Observations(Inc UI)



MULTIPLE PLOTS

Program No: 19 **Date:** 19-09-2022

AIM

Write a Python program to create multiple plots.

Theoretical Support

plot() -This function is used to draw points(markers) in a diagram.

show() – This function is used to display all diagrams.

subplot() -This function is used to draw multiple plots on same figure.

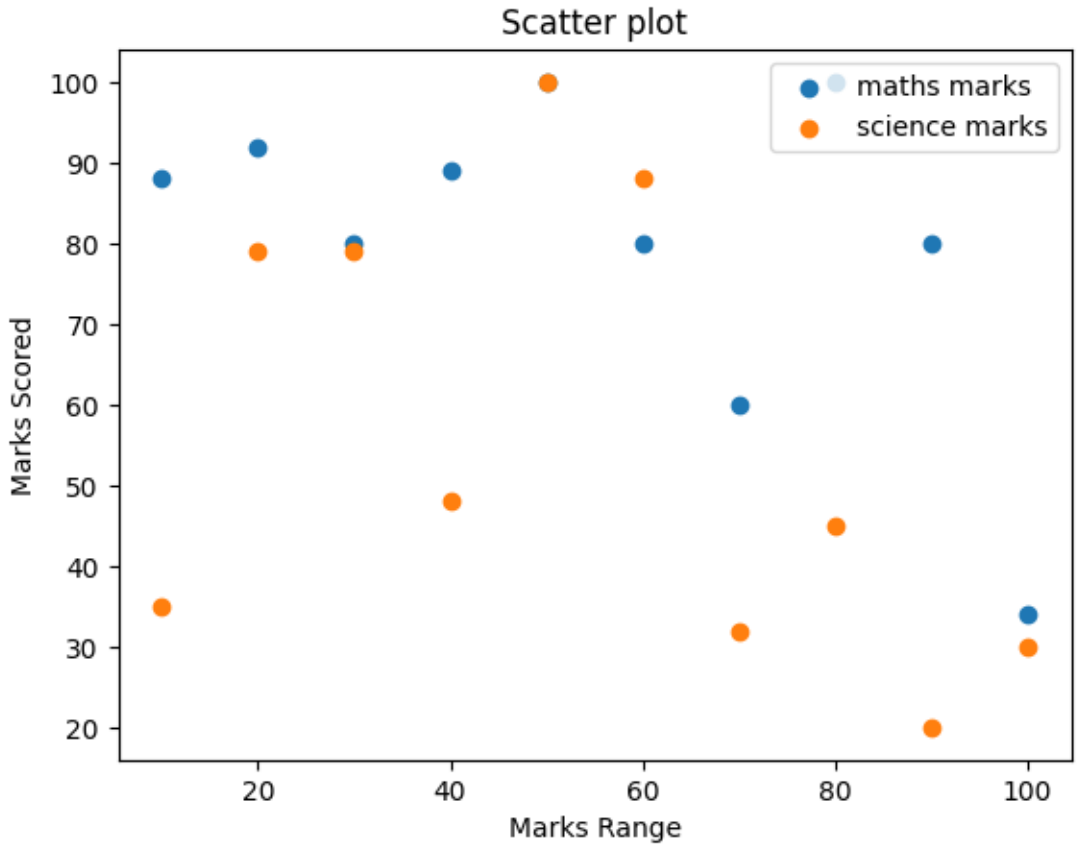
Code

```
import matplotlib.pyplot as plt
figure, axis = plt.subplots(1,2)
dt=0.01
x1 = [10,20,30]
y1 = [10,20,30]
axis[0].plot(x1,y1)
#axis[0,0].plot(x1,y1)
axis[0].set_title("Plot 1")
x2 = [10,10,10]
y2 = [30,40,50]
axis[1].plot(x2, y2)
#axis[0,1].plot(x2, y2)
axis[1].set_title("Plot 2")
plt.show()
```

Inference

We can draw multiple plots on same figure using subplot().

Result and Observations(Inc UI)



SCATTER PLOT

Program No: 20 **Date:** 19-09-2022

AIM

Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students.

Sample data:

Test Data:

math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]

science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]

marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

Theoretical Support

plot() -This function is used to draw points(markers) in a diagram.

show() – This function is used to display all diagrams.

scatter()- This function is used to draw scatter plot.

Code

```
import matplotlib.pyplot as plt
x = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
m = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
s = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
plt.scatter(x, m,label="maths marks")
plt.scatter(x, s,label="science marks")
plt.legend(loc='upper right')
```

```
plt.xlabel("Marks Range")
plt.ylabel("Marks Scored")
plt.title("Scatter plot")
plt.show()
```

Inference

We can draw scatter plot using matplotlib method scatter().

Result and Observations(Inc UI)

```
List ['a', 'b', 'c']
Series
0    a
1    b
2    c
dtype: object
```

EXERCISE 4: INTRODUCTION TO PANDAS

LIST TO SERIES CONVERSION

Program No: 21 **Date:** 21-09-2022

AIM

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

Theoretical Support

pandas.series()- It is a one dimensional labeled array capable of holding data of any type.

Code

```
import pandas as pd

lis = ['a','b','c']

print("List",lis)

ser= pd.Series(lis)

print("Series")

print(ser)
```

Inference

In this we can convert list to series using the series().

Result and Observations(Inc UI)

2021-05-01
2021-05-02
2021-05-03
2021-05-04
2021-05-05
2021-05-06
2021-05-07
2021-05-08
2021-05-09
2021-05-10
2021-05-11
2021-05-12

GENERATING SERIES OF DATES

Program No: 22 **Date:** 23-09-2022

AIM

Write a python program to Generate the series of dates from 1st May, 2021 to 12th May, 2021 (both inclusive).

Theoretical Support

pandas.series()- It is a one dimensional labeled array capable of holding data of any type.

Code

```
import pandas as pd

sr = pd.Series(pd.date_range('2021-05-01','2021-05-12',freq = 'D'))

print(sr.to_string(index=False))
```

Inference

In this we can create a program to generate a series of dates from a given range.

Result and Observations(Inc UI)

	Name	Age
0	anand	24
1	basil	25
2	rajiv	26
3	dasan	27

DICTIONARY TO DATA FRAME

Program No: 23 **Date:** 23-09-2022

AIM

Given a dictionary, convert it into corresponding data frame and display it.

Theoretical Support

pandas.DataFrame() – It is a 2-dimensional data structure , like 2- dimensional array or a table with rows and columns.

Code

```
import pandas as pd

data = {


   'Name' : ['anand','basil','rajiv','dasan'],

   'Age' : [24,25,26,27],

}

print(pd.DataFrame(data))
```

Inference

We can convert a given dictionary into a corresponding dataframe using DataFrame().

Result and Observations(Inc UI)

	0	1
0	arun	22
1	aju	24

LIST TO DATA FRAME

Program No: 24 **Date:** 30-09-2022

AIM

Given a 2D List, convert it into corresponding data frame and display it.

Theoretical Support

pandas.DataFrame() – It is a 2-dimensional data structure , like 2- dimensional array or a table with rows and columns.

Code

```
import pandas as pd

details = [["arun",22],["aju",24]]

df = pd.DataFrame(details)

print(df)
```

Inference

We can convert a given list into corresponding dataframe using DataFrame().

Result and Observation(Inc UI)

	0	1	2
0	html	css	javascript
1	c	cpp	java
2	100	200	300

CSV FILE TO DATA FRAME

Program No: 25 **Date:** 30-09-2022

AIM

Given a CSV file, read it into a dataframe and display it.

Theoretical Support

pandas.DataFrame() – It is a 2-dimensional data structure , like 2- dimensional array or a table with rows and columns.

Code

```
import pandas as pd
data = pd.read_csv ('sample.csv',header=None)
df=pd.DataFrame(data)
print(df)
```

Inference

We can convert a given csv file into corresponding dataframe using DataFrame().

Result and Observations(Inc UI)

	Name	Age
0	anand	24
1	basil	25

DISPLAY FIRST TWO ROWS OF DATA FRAME

Program No: 26 **Date:** 10-10-2022

AIM

Given a data frame, select first 2 rows and output them.

Theoretical Support

pandas.DataFrame() – It is a 2-dimensional data structure , like 2- dimensional array or a table with rows and columns.

Code

```
import pandas as pd


da = {

   'Name' : ['anand','basil','arun','dasan'],

   'Age' : [24,25,26,27],

}


df = pd.DataFrame(da)


print(df[0:2])
```

Inference

In this we create an dataframe and can select and display data from a given rows.

Result and Observations(Inc UI)

```
    Name   Occupation  Salary
0  anand       driver   20000
1  basil     engineer   30000
2   arun       driver   15000
3  dasan      plumber   27000
4  anoop  electrician   23000
Average salary per occupation :
Occupation
driver         17500.0
electrician    23000.0
engineer       30000.0
plumber        27000.0
Name: Salary, dtype: float64
```

AVERAGE SALARY PER OCCUPATION

Program No: 27 **Date:** 10-10-2022

AIM

Given is a dataframe showing name, occupation, salary of people. Find the average salary per occupation.

Theoretical Support

pandas.DataFrame() – It is a 2-dimensional data structure , like 2- dimensional array or a table with rows and columns.

Pandas.groupby() - is used for grouping the data according to the categories and apply a function to the categories.

Code

```
import pandas as pd

data = {

    'Name' : ['anand','basil','arun','dasan','anoop'],

    'Occupation' : ['driver','engineer','driver','plumber','electrician'],

    'Salary' : [20000,30000,15000,27000,23000],}

  df = pd.DataFrame(data)

print(df)

occ_average_age = df.groupby('Occupation')['Salary'].mean()

print("Average salary per occupation : ")

print(occ_average_age)
```

Inference

In this we create an dataframe and find average salary per occupation from the data.

Result and Observations(Inc UI)

	eid	ename	stipend	designation
0	1	alex	1000	asst.manager
1	2	basil	2000	manager
2	3	syam	900	sales_executive
3	4	deepu	700	hr

DISPLAYING EMPLOYEE DETAILS FROM DATA FRAME

Program No:28 **Date:** 17-10-2022

AIM

Given are 2 dataframes, with one dataframe containing Employee ID (eid),Employee Name (ename) and Stipend (stipend) and the other dataframe containing Employee ID(eid) and designation of the employee (designation). Output the Dataframe containing Employee ID (eid), Employee Name (ename), Stipend (stipend) and Position (position).

Theoretical Support

pandas.DataFrame() – It is a 2-dimensional data structure , like 2- dimensional array or a table with rows and columns.

Pandas.merge() - This method updates the content of two DataFrame by merging them together.

Code

```
import pandas as pd


details_1 = {


    'eid' : [1,2,3,4],

    'ename' : ['alex','basil','syam','deepu'],

    'stipend':[1000,2000,900,700],


}

details_2 = {
```

```
  'eid' : [1,2,3,4],

 'designation' : ['asst.manager','manager','sales_executive','hr'],  }

df_1 = pd.DataFrame(details_1)

df_2 = pd.DataFrame(details_2)

dataframe = pd.merge(df_1, df_2, how = 'inner', on = 'eid')

print(dataframe)
```

Inference

We create two dataframe and merge two dataframe by specified data from the dataframe.

Result and Observations(Inc UI)

```
Accuracy
0.9666666666666667
```

EXERCISE 5

K-NN CLASSIFICATION

Program No: 29 **Date:** 21-10-2022

AIM

Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.

Theoretical Support

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

Dataset used : Iris dataset

Code

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
irisData = load_iris()
X = irisData.data
y = irisData.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
print("Accuracy")
print(knn.score(X_test,y_test))
```

Inference

The dataset which is loaded is split into 80% of training data and 20% of testing data. Then train the data and an accuracy of 96% while testing.

Result and Observations(Inc UI)

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 1 1 1 2 0 2 0 0]
CM [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]


predicted output for [[5,5,4,4]]: [2]
Naive bayes score               : 1.0
```

EXERCISE 6

NAIVE BAYES ALGORITHM

Program No: 30 **Date:** 28-10-2022

AIM

Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.

Theoretical Support

Naïve Bayes classifiers are a collection of classification algorithms based on Bayes Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of feature being classified is independent of each other.

Dataset Used : Iris Dataset

Code

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

X,y=load_iris(return_X_y=True)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=0)

gnb=GaussianNB()

y_pred=gnb.fit(X_train,y_train).predict(X_test)

print(y_pred)

x_new=[[5,5,4,4]]

y_new=gnb.fit(X_train,y_train).predict(x_new)

from sklearn.metrics import confusion_matrix
```

```
cm = confusion_matrix(y_test, y_pred)

print ("CM", cm)

print("\n")

print("predicted output for [[5,5,4,4]]:",y_new)

print("Naive bayes score             :",gnb.score(X_test,y_test))
```

Inference

The dataset which is loaded is split into 70% of training data and 30% of testing data. Then train the data and an accuracy of 100% while testing.

Result and Observations(Inc UI)

```
Accuracy:  0.9555555555555556
```

EXERCISE 7

DECISION TREE

Program No: 31 **Date:** 08-11-2022

AIM

Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of the algorithm.

Theoretical Support

A decision tree is a non-parametric supervised learning algorithm,which is utilized for both classification and regression tasks.It has a hierarchical,tree structure,which consists of a root node, branches ,internal nodes and leaf nodes.

Dataset Used :Iris Dataset

Code

```
from sklearn.datasets import load_iris

from sklearn import metrics

from sklearn import tree

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

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)

clf=DecisionTreeClassifier()

clf=clf.fit(x_train,y_train)

y_pred=clf.predict(x_test)

print("Accuracy: ",metrics.accuracy_score(y_test,y_pred))
```

Inference

The dataset which is loaded is split into 70% of training data and 30% of testing data. Then train the data and an accuracy of 95% while testing.

Result and Observations(Inc UI)

```
Coefficients:
 [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

EXERCISE 8

REGRESSION

Program No: 32 **Date:** 18-11-2022

AIM

Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance.

Theoretical Support

Linear regression is one of the most common techniques of regression analysis. Multiple regression is a broader class of regressions that encompasses linear and nonlinear regressions with multiple explanatory variables.

Dataset used : Diabetes dataset

Code

```
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.shape

diabetes_y.shape

diabetes_X = diabetes_X[:, np.newaxis, 2]
```

```
diabetes_X.shape

diabetes_X_train = diabetes_X[:-20]

diabetes_X_test = diabetes_X[-20:]

diabetes_y_train = diabetes_y[:-20]

diabetes_y_test = diabetes_y[-20:]

regr = linear_model.LinearRegression()

regr.fit(diabetes_X_train, diabetes_y_train)

diabetes_y_pred = regr.predict(diabetes_X_test)

print("Coefficients: \n", regr.coef_)

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))
```

Inference

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. Multiple regression find a relation between multiple variable or features. It also defines the correlation between independent variables dependent variables.

Result and Observations(Inc UI)

```
Accuracy :  1.0
['setosa']
```

EXERCISE 9

SUPPORT VECTOR MACHINE

Program No: 33 **Date:** 24-11-2022

AIM

Program to implement text classification using a Support vector machine.

Theoretical Support

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple. The algorithm creates a line or a hyperplane which separates the data into classes.

Dataset Used : Iris dataset

Code

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn import metrics

from sklearn.svm import SVC

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)

classifier = SVC(kernel='linear', random_state=0)

classifier.fit(x_train, y_train)
```

```
y_pred= classifier.predict(x_test)

print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))

sample = [[1,1,1,2]]

pred = classifier.predict(sample)

pred_v = [iris.target_names[p] for p in pred]

print(pred_v)
```

Inference

The dataset which is loaded is split into 70% of training data and 30% of testing data. Then train the data and an accuracy of 100% while testing.

Result and Observations(Inc UI)

```
[[5.77358491 2.69245283]
 [5.006      3.428     ]
 [6.81276596 3.07446809]]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 0 2 0 2 0 2 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0
 2 2 2 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 0 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

EXERCISE 10

K-MEANS CLUSTERING

Program No: 34 **Date:** 02-12-2022

AIM

Program to implement k-means clustering technique using any standard dataset available in the public domain.

Theoretical Support

K-means clustering is a method of vector quatization, orginally from signal processing , that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean , serving as a prototype of the cluster.

Dataset Used : Iris Dataset

Code

```
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
km = KMeans(n_clusters = 3, init='k-means++', n_init=10, max_iter=300, tol=0.0001,
```

```
verbose=0, random_state=21, copy_x=True, algorithm="auto")
km.fit(X)
centers = km.cluster_centers_
print(centers)
new_labels = km.labels_
print(new_labels)
print(y)
```

Inference

K-means algorithm identifies k number of centroids and then allocates every data point to the nearest cluster ,while keeping the centroid as small as possible.