Ex. No. 1a	Basic Experiment using Python

Numpy:

Numpy is a python library used for working with arrays. It also has function for working in domains of linear algebra, fourier transform and matrices. Numpy was created in 2005 by Trawis Oliphant. It is an open source project and you can use it freely. Numpy stands for Numerical Python.

Why use Numpy:

In python we have list that serve the purpose of arrays but they are slow to process.

Numpy aims to provided an array object that is up to 50x faster than traditional python lists. The array object in Numpy is called ndarray.

Arrays are frequently used in data science where speed and resources are very important.

Which language Numpy written

Numpy is a python library which is written partially in python, but most of the parts that requires fast computation are written in C or C++.

Matplotlib

Matplotlib.pyplot is collection of functions in the Matplotlib library that provide a simple interface for creating various types oif plot and visualizations in Python.

Matplotlib is popular data visualization libray used for creating static, animated and interactive plots in Python.

Here are some common types of plots you can create with matplotlib.pyplot:

- Line plots
- Scatter plots
- Bar plots
- Bar charts
- Histograms
- Pie charts

Aim:

To implement basic numpy experiments.

Algorithm:

Step 1: Import the numpy library .

Step 2: Use np.array to create an array.

Step 3: Use np.ones and np.zeros are to create an array full of ones and zeros.

Step 4: Use np.arrange() is used to create an array within a range of particular range.

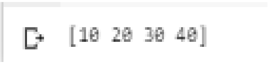
Step 5: Use astype() is used to change the type of an array.

Step 6: Concatenate is used to join two arrays.

Step 7: Use random() is used to create a array of random values.

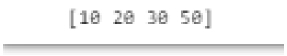
Source code :

```
Import numpy

ar=numpy.array([10,20,30,40])

print(ar)
```



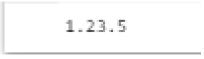
```
[10 20 30 40]
```

```
import numpy as np

arr=np.array([10,20,30,50])

print(arr)
```



```
[10 20 30 50]
```

```
print(np.__version__)
```

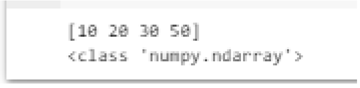


```
1.23.5
```

#Find the type of the array

```
import numpy as np

arr=np.array([10,20,30,50])
```

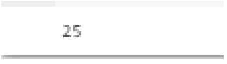
```
print(arr)

print(type(arr))
```



```
[10 20 30 50]
<class 'numpy.ndarray'>
```

#Zero Dimensional Array

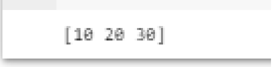
```
import numpy as np

arr=np.array(25)

print(arr)
```



```
25
```

#Create a 1-Dimensional array containing the values

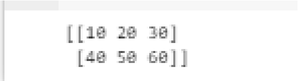
```
import numpy as np

arr=np.array([10,20,30])

print(arr)
```



```
[10 20 30]
```

#Create a 2-Dimensional array containing the values

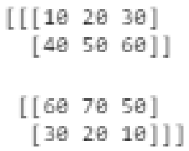
```
import numpy as np

arr=np.array([10,20,30],[40,50,60])

print(arr)
```



```
[[10 20 30]
 [40 50 60]]
```

#Create multidimensional array containing the values

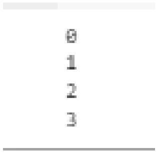
```
import numpy as np

arr = np.array([[[10, 20, 30], [40, 50, 60]], [[60, 70, 50], [30, 20, 10]]])

print(arr)
```



```
[[[10 20 30]
  [40 50 60]]

 [[60 70 50]
  [30 20 10]]]
```

#Check the dimensional of the array

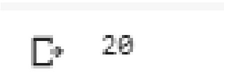
```
import numpy as np

a=np.array(10)

b=np.array([10, 20, 30,50])

c=np.array([[10,20,30], [40, 50, 60]])

d=np.array([[[10, 20, 30], [40, 50, 60]], [[10, 20, 30], [40, 50, 60]]])

print(a.ndim)

print(b.ndim)

print(c.ndim)

print(d.ndim)
```



```
0
1
2
3
```

#Indexing the first element to print

```
import numpy as np

arr = np.array([10, 20, 30, 50])

print(arr[1])
```

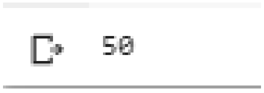


```
20
```

#Access the two elements in the array and add them

```
import numpy as np

arr = np.array([10, 20, 30, 50])
```

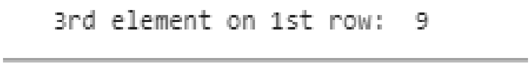
```
print(arr[1] + arr[2])
```



```
50
```

#Access the 2-Dimensional array elements

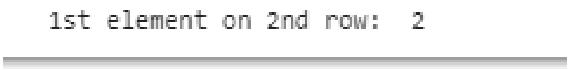
```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('3rd element on 1st row: ', arr[1, 3])
```



```
3rd element on 1st row:  9
```

#Access the 2-Dimensional Array

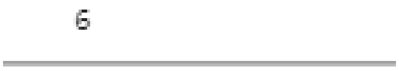
```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('1st element on 2nd row: ', arr[0, 1])
```



```
1st element on 2nd row:  2
```

#Access the Multi Dimensional array element

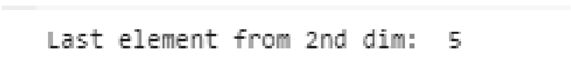
```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[0, 1, 2])
```



```
6
```

#Negative Indexing

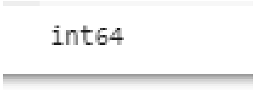
```
import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('Last element from 2nd dim: ', arr[0, -1])
```



```
Last element from 2nd dim:  5
```

#Check the datatype of the array

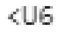
```
import numpy as np

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

print(arr.dtype)
```



```
int64
```

#Check the datatype

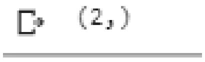
```
import numpy as np

arr = np.array(['apple', 'banana', 'cherry'])

print(arr.dtype)
```



```
<U6
```

#Check the shape of the array

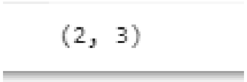
```
import numpy as np

arr = np.array(['apple', 'banana'])

print(arr.shape)
```



```
(2,)
```

#Check the shape of the 2-D array

```
import numpy as np

arr = np.array([['apple', 'banana', 'cherry'], ['a','b','c']])

print(arr.shape)
```

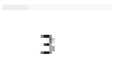


```
(2, 3)
```

#Check the array size

```
import numpy as np
```

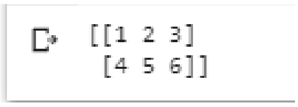
```
arr = np.array(['apple', 'banana', 'cherry'])

print(arr.size)
```



```
3
```

#Check the array size

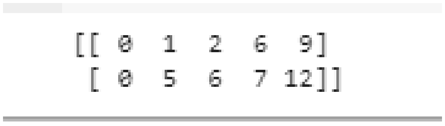
```
import numpy as np

arr=np.array([1,2,3,4,5,6])

newarr=arr.reshape(2,3)

print(newarr)
```



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

#Sort the array

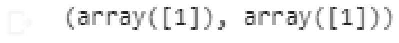
```
import numpy as np

arr = np.array([[1,0,6,2,9], [0,12,5,7,6]])

print(np.sort(arr))
```



```
[[ 0  1  2  6  9]
 [ 0  5  6  7 12]]
```

#Find the element in the array

```
import numpy as np

arr = np.array([[1,2,3,4,5], [10,12,9,7,6]])

print(np.where(arr==12))
```



```
(array([1]), array([1]))
```

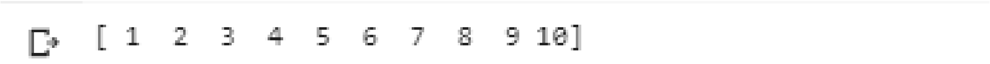
#Concatenation of array

```
import numpy as np

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

arr1=np.array([6,7,8,9,10])

c=np.concatenate((arr,arr1))

print(c)



```
[ 1  2  3  4  5  6  7  8  9 10]
```

#hstack is using to stack the array

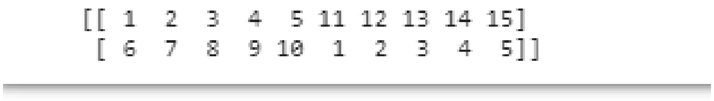
import numpy as np

arr = np.array([[1,2,3,4,5],[6,7,8,9,10]])

arr1=np.array([[11,12,13,14,15],[1,2,3,4,5]])

c=np.hstack((arr,arr1))

print(c)



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

#hstack is to used to concatenation

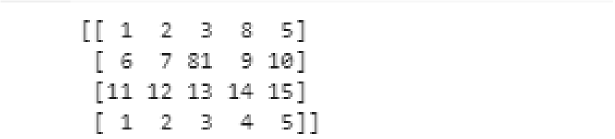
import numpy as np

arr = np.array([[1,2,3,8,5],[6,7,8,9,10]])

arr1=np.array([[11,12,13,14,15],[1,2,3,4,5]])

c=np.concatenate((arr,arr1))

print(c)



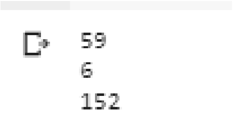
```
[[ 1  2  3  8  5]
 [ 6  7 81  9 10]
 [11 12 13 14 15]
 [ 1  2  3  4  5]]
```

#Print the maximum, minimum and sum of the array

import numpy as np

arr=np.array([11,12,13,59,51,6])

```
print(arr.max())

print(arr.min())

print(arr.sum())
```



```
59
6
152
```

Result:

Thus the basic experiments using pandas and matplotlib libraries in python was executed and ouput was verified.

Ex. No. 1b	Basic Experiment of Graphical Representation using Python

Aim:

To implement basic matplotlib experiments.

Algorithm:

Step 1: Import the numpy library .

Step 2: Import the file.

Step 3: Grphical representation of scatter plot,bar chart.

Step 4: Display the output of graphical chart.

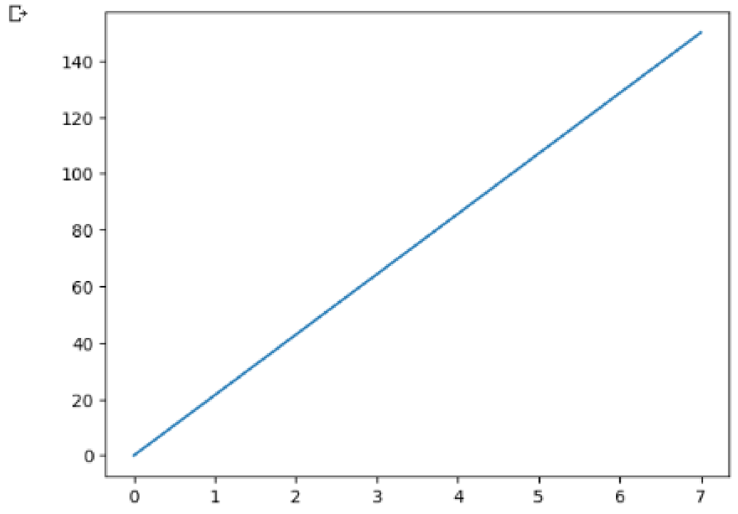
Step 5: Stop the execution.

Source code :

#Graphical Representation

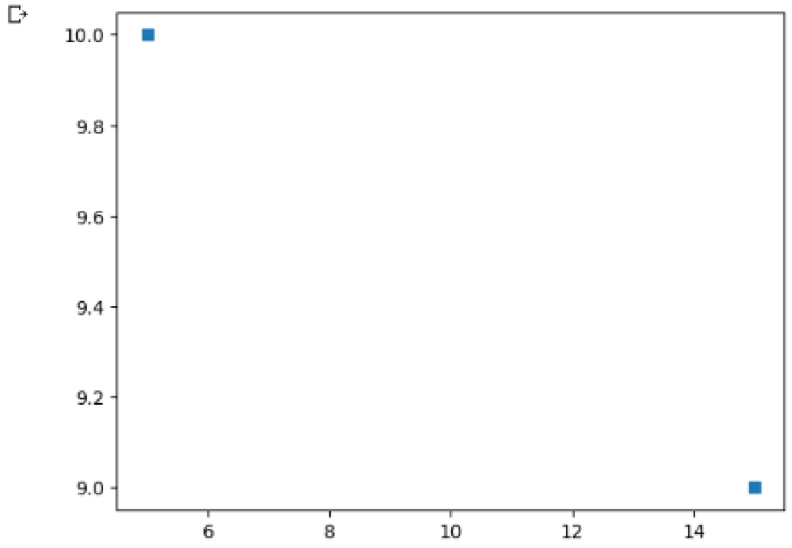
#line graph

```
import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([0, 7])

ypoints = np.array([0, 150])

plt.plot(xpoints, ypoints)

plt.show()
```



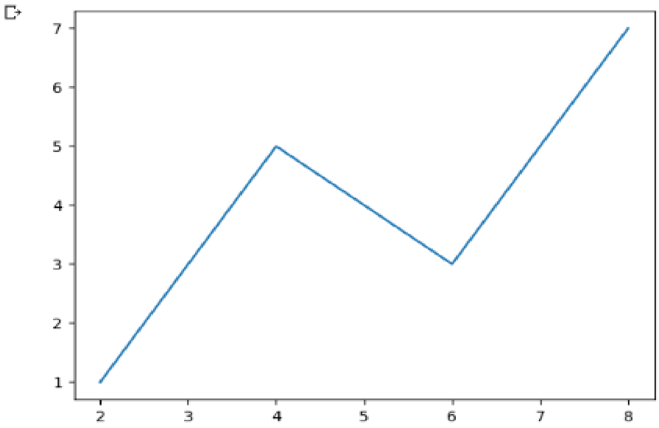
#Square Graph

```
import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([15, 5])

ypoints = np.array([9, 10])

plt.plot(xpoints, ypoints, 's')

plt.show()
```



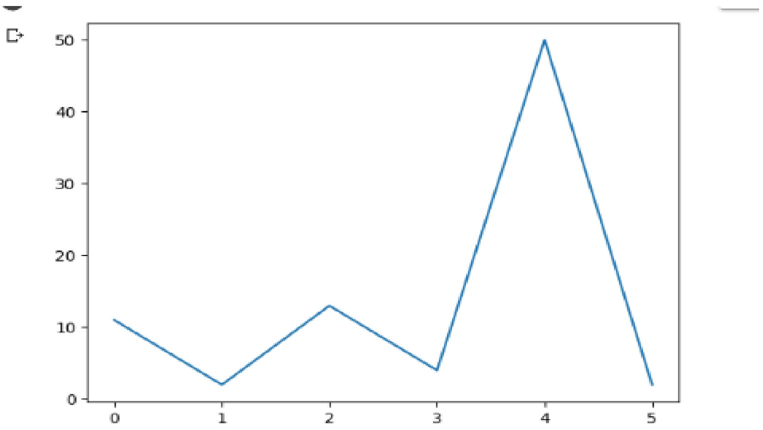
#Multi point Line graph

```
import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([2, 4, 6, 8])

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

plt.plot(xpoints, ypoints)

plt.show()
```



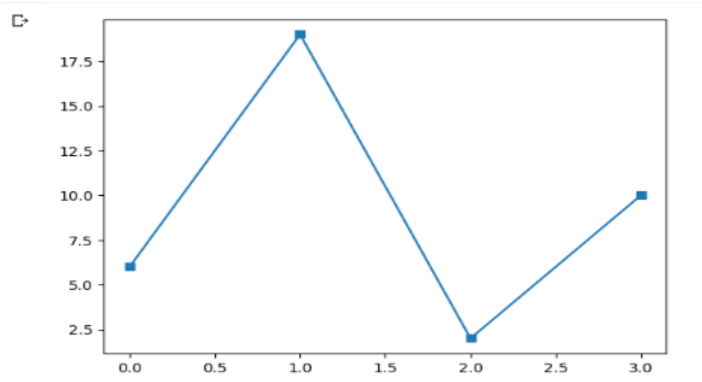
#Plotting without x-points

```
import matplotlib.pyplot as plt

import numpy as np

ypoints = np.array([11, 2, 13, 4, 50, 2])

plt.plot(ypoints)

plt.show()
```



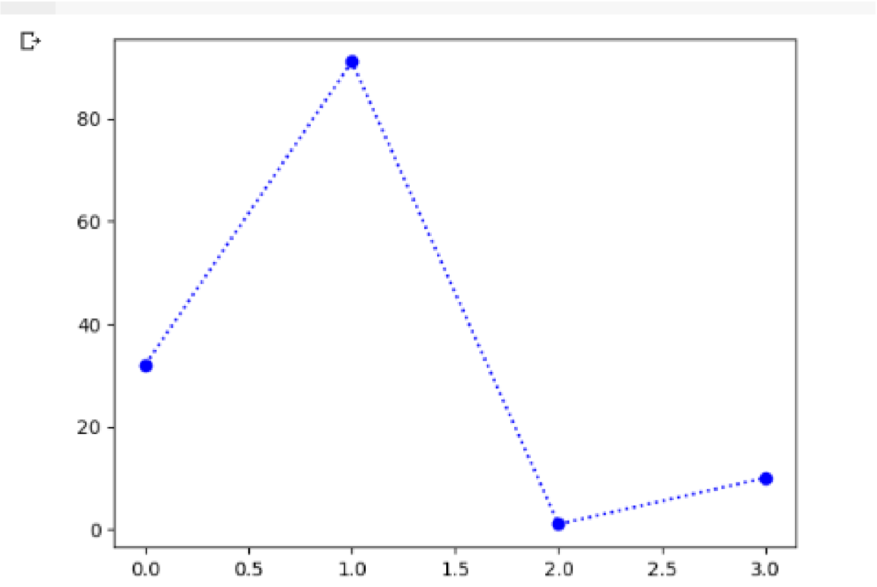
#Line graph with dotted points

```
import matplotlib.pyplot as plt

import numpy as np

ypoints = np.array([6, 19, 2, 10])

plt.plot(ypoints, marker = 's')

plt.show()
```



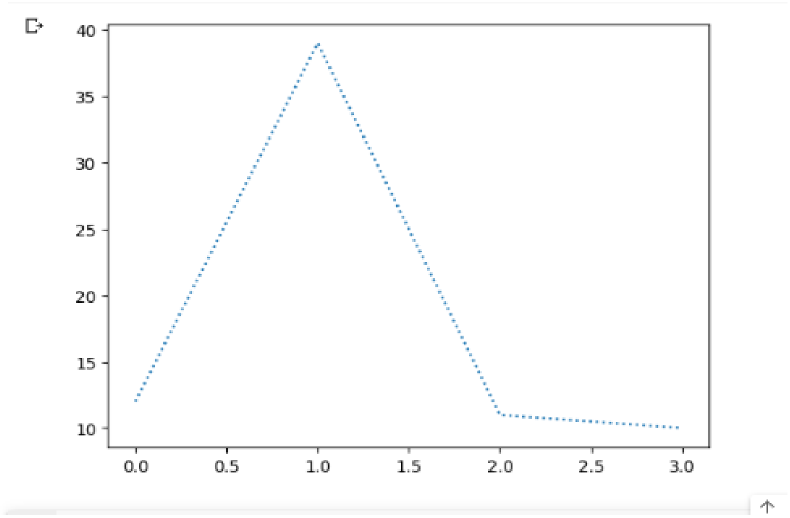
#Dotted graph with points

```
import matplotlib.pyplot as plt

import numpy as np

ypoints = np.array([32, 91, 1, 10])

plt.plot(ypoints, 'o:b')

plt.show()
```



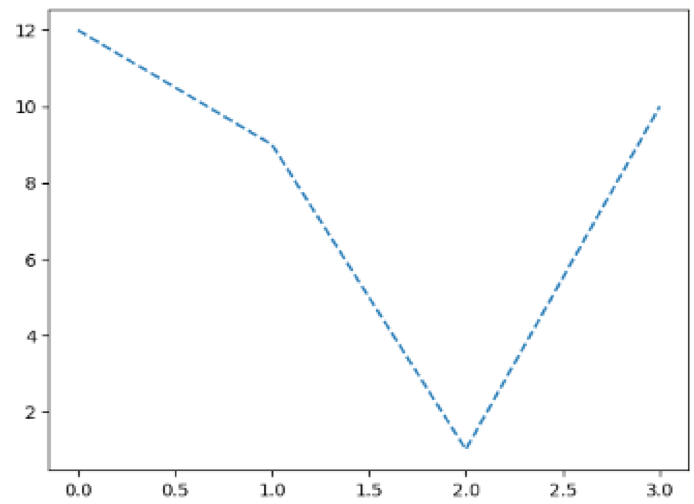
#Dotted graph with Multi points

```
plt.plot(ypoints, 'o:')

plt.show()
```



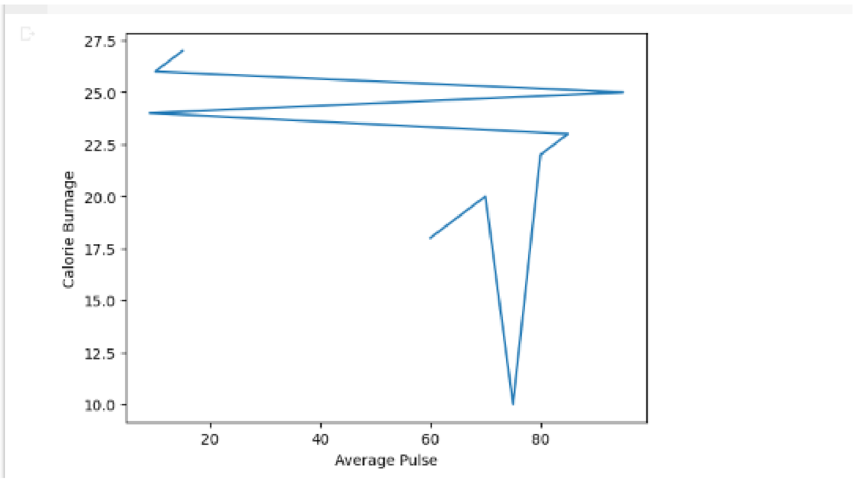
#Graphical representation of graph

```
import matplotlib.pyplot as plt

import numpy as np

ypoints = np.array([12, 9, 1, 10])

plt.plot(ypoints, ls = '--')

plt.show()
```



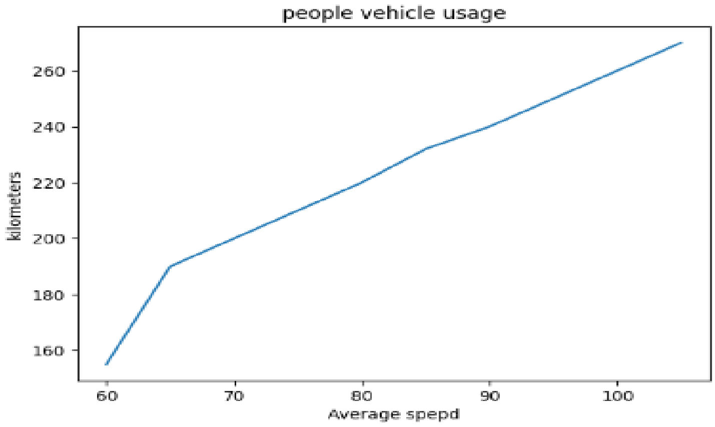
#Combination of two graphs

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([60, 65, 70, 75, 80, 85, 9, 95, 10, 15])

y = np.array([18, 19, 20, 10, 22, 23, 24, 25, 26,27])

plt.plot(x, y)

plt.xlabel("Average Pulse")

plt.ylabel("Calorie Burnage")

plt.show()
```



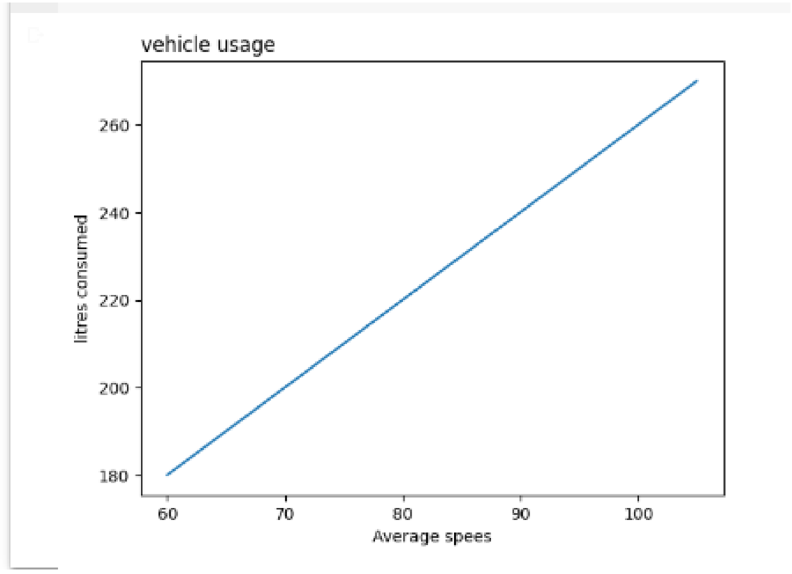
#Plot the x-axis and y-axis

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([60, 65, 70, 75, 80, 85, 90, 95, 100, 105])

y = np.array([155, 190, 200, 210, 220, 232, 240, 250, 260,270])

plt.plot(x, y)

plt.title("people vehicle usage")

plt.xlabel("Average spepd")

plt.ylabel("kilometers")

plt.show()
```



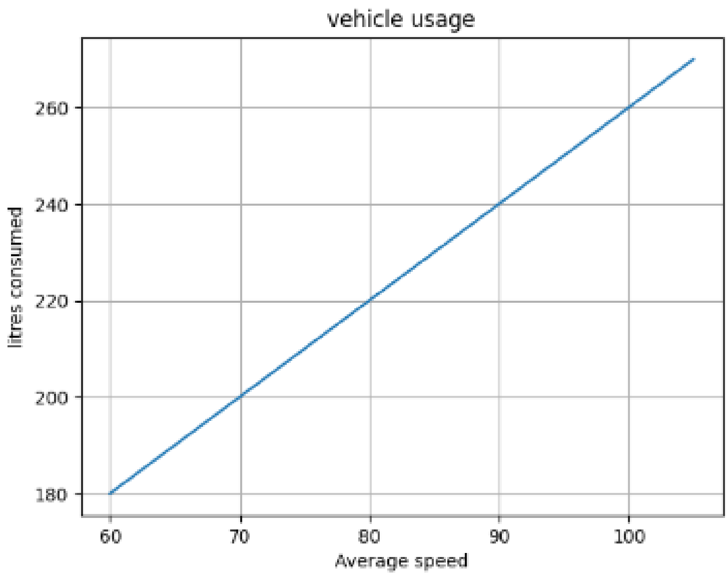
#Position of label in the alignment

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([60, 65, 70, 75, 80, 85, 90, 95, 100, 105])

y = np.array([180, 190, 200, 210, 220, 230, 240, 250, 260,270])

plt.title("vehicle usage", loc = 'left')

plt.xlabel("Average spees")

plt.ylabel("litres consumed")
plt.plot(x, y)

plt.show()
```



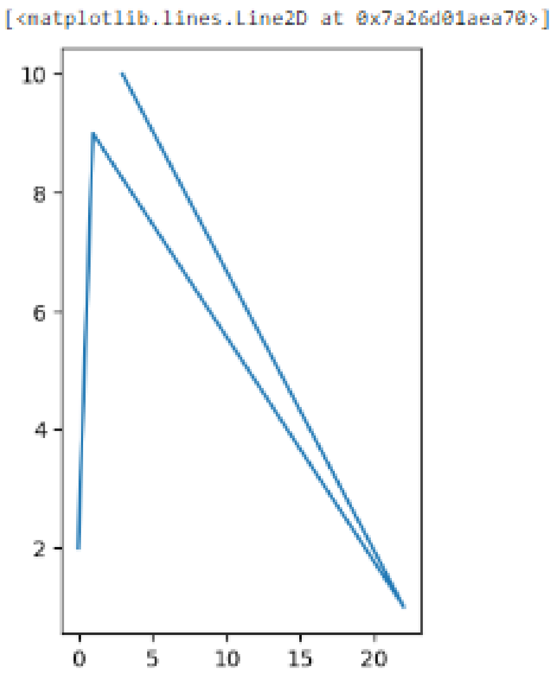
#Grid of the graph

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([60, 65, 70, 75, 80, 85, 90, 95, 100, 105])

y = np.array([180, 190, 200, 210, 220, 230, 240, 250, 260,270])

plt.title("vehicle usage")

plt.xlabel("Average speed")

plt.ylabel("litres consumed")

plt.plot(x, y)

plt.grid()

plt.show()
```



#Import the Libraries

```
import matplotlib.pyplot as plt

import numpy as np

#Subplots the graphs

x = np.array([0, 1, 22, 3])

y = np.array([2, 9, 1, 10])

plt.subplot(1, 2, 1)

plt.plot(x,y)
```



```
[<matplotlib.lines.Line2D at 0x7a26d01aea70>]
```

#Subplots of the graph

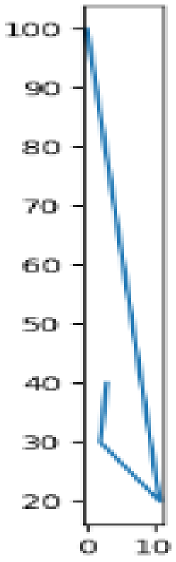
x = np.array([0, 11, 2, 3])

y = np.array([100, 20, 30, 40])

plt.subplot(1, 12, 2)

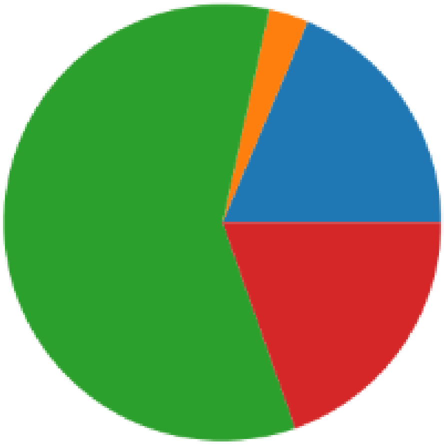
plt.plot(x,y)

plt.show()



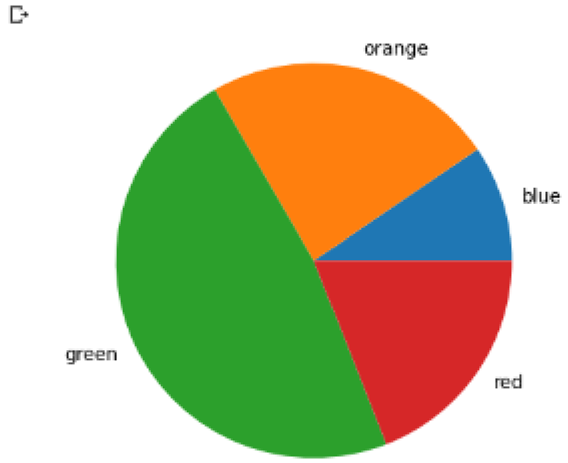
#Pie chart

```
import matplotlib.pyplot as plt

import numpy as np

y = np.array([19, 3, 60, 20])

plt.pie(y)

plt.show()
```



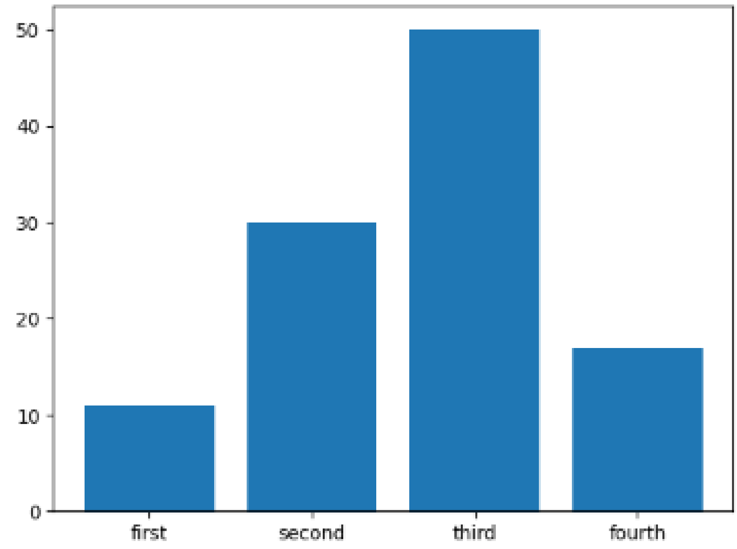
#Pie chart with labels

```
import matplotlib.pyplot as plt

import numpy as np

y = np.array([10, 25, 50, 20])

mylabels = ["blue", "orange", "green", "red"]

plt.pie(y, labels = mylabels)

plt.show()
```



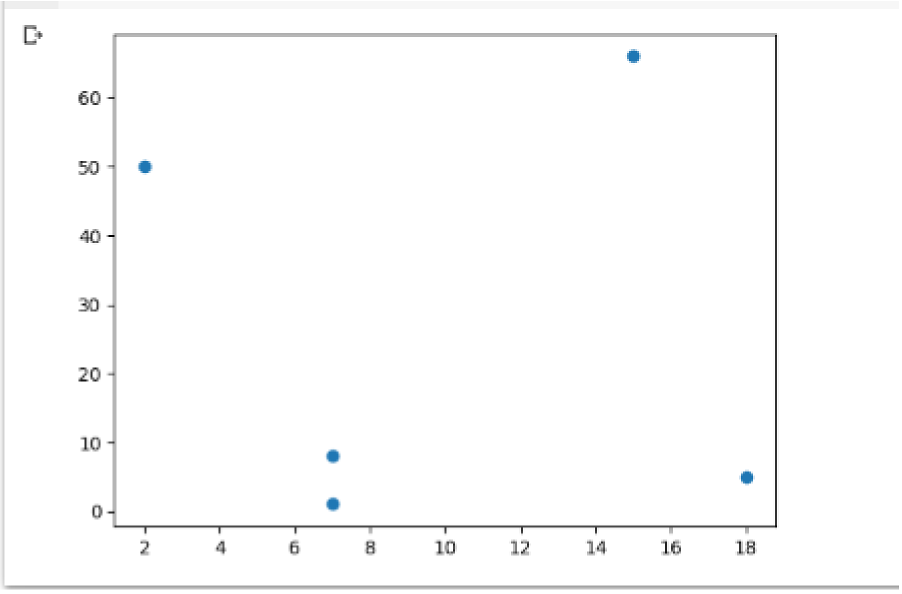
#Bar chart

```
import matplotlib.pyplot as plt

import numpy as np

x = np.array(["first", "second", "third", "fourth"])

y = np.array([11, 30, 50, 17])

plt.bar(x,y)

plt.show()
```



#Scatter Plots

```
import matplotlib.pyplot as plt

import numpy as np

x = np.array([15,7,18,7,2])

y = np.array([66,1,5,8,50])

plt.scatter(x, y)

plt.show()
```



#Version of matplotlib

```
import matplotlib

print(matplotlib.__version__)
```

```
3.7.1
```

Result:

Thuis the python program of graphical representation was executed and output was verified.

Ex. No. 2	Data Pre-processing using Python

Aim:

To implement the basic data pre-processing task for scholarship.

Algorithm:

Step 1: Import the csv file to the location.

Step 2: Execute the commands to check the table.

Step 3: Import pandas and numpy libraries to analysis the table.

Step 4: Find the shape of the table and datatypes of the table.

Step 5: Find maximum and minimum of the table and then find the normalization.

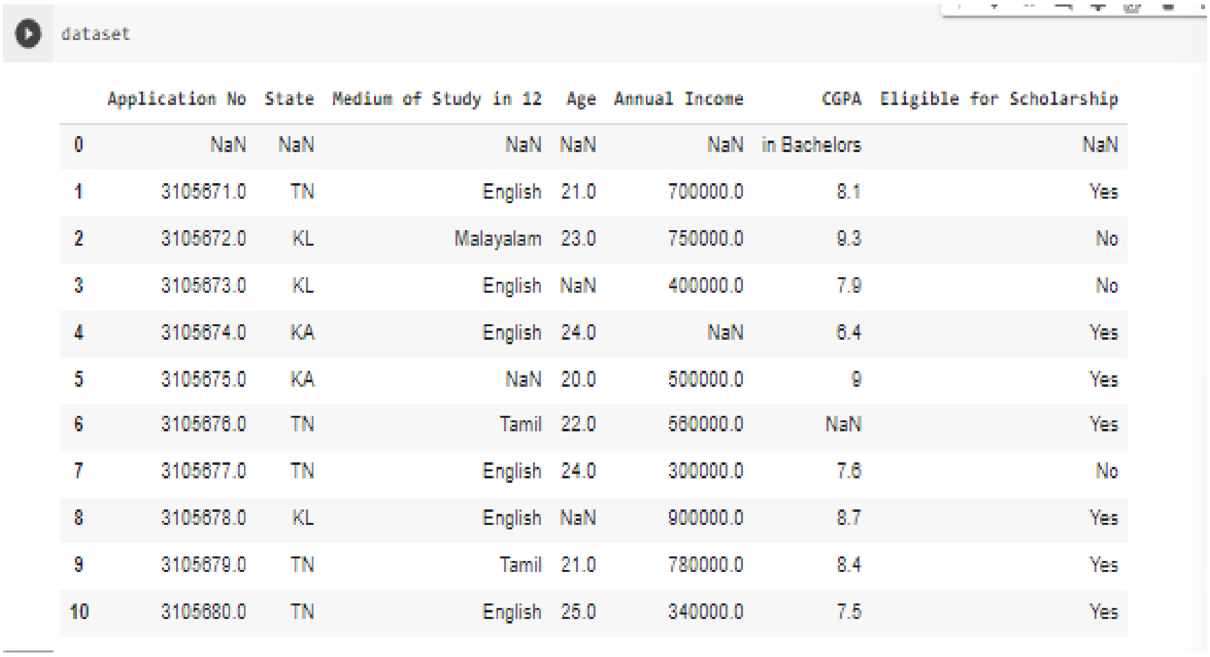
Step 6: Find standard scaler of the table.

Step 7: Display the output of the commands.

Source code :

#Print the table

dataset



```
dataset
```

	Application No	State	Medium of Study in 12	Age	Annual Income	CGPA	Eligible for Scholarship
0	NaN	NaN	NaN	NaN	NaN	in Bachelors	NaN
1	3105671.0	TN	English	21.0	700000.0	8.1	Yes
2	3105672.0	KL	Malayalam	23.0	750000.0	9.3	No
3	3105673.0	KL	English	NaN	400000.0	7.9	No
4	3105674.0	KA	English	24.0	NaN	6.4	Yes
5	3105675.0	KA	NaN	20.0	500000.0	9	Yes
6	3105676.0	TN	Tamil	22.0	560000.0	NaN	Yes
7	3105677.0	TN	English	24.0	300000.0	7.6	No
8	3105678.0	KL	English	NaN	900000.0	8.7	Yes
9	3105679.0	TN	Tamil	21.0	780000.0	8.4	Yes
10	3105680.0	TN	English	25.0	340000.0	7.5	Yes

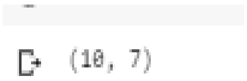
#Import the libraries

import pandas as pd

import numpy as np

#Shape of the table

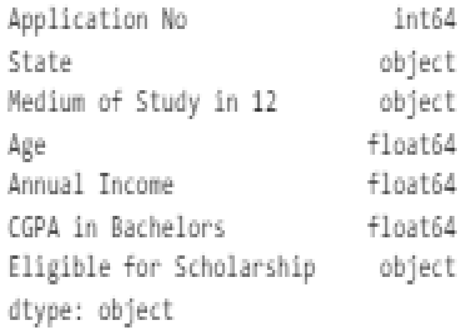
print(dataset.shape)



```
(10, 7)
```

#Datatypes of the table

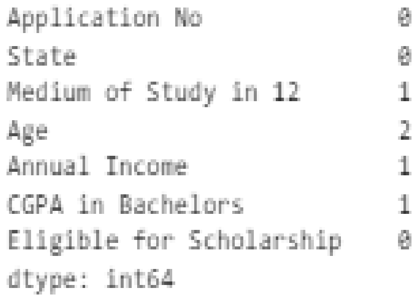
print(dataset.dtypes)



```
Application No               int64
State                       object
Medium of Study in 12       object
Age                        float64
Annual Income              float64
CGPA in Bachelors          float64
Eligible for Scholarship    object
dtype: object
```

#To find the empty detail in the table

print(dataset.isnull().sum())



```
Application No              0
State                       0
Medium of Study in 12       1
Age                         2
Annual Income               1
CGPA in Bachelors           1
Eligible for Scholarship    0
dtype: int64
```

#Label the category

dataset['binned_age'] = pd.cut(

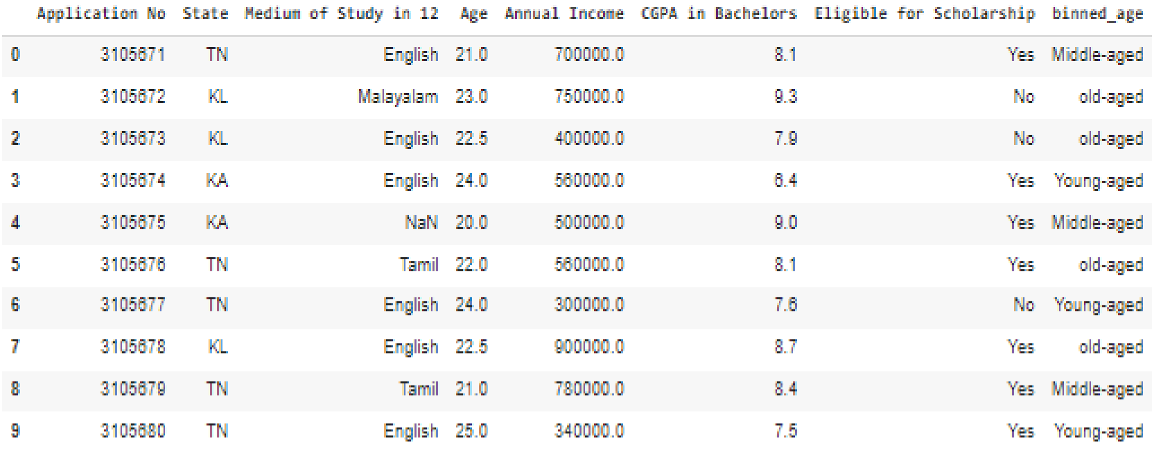
dataset['Age'],

bins=3,

```
labels=['Middle-aged', 'old-aged', 'Young-aged'])
```

#print the table

```
dataset
```

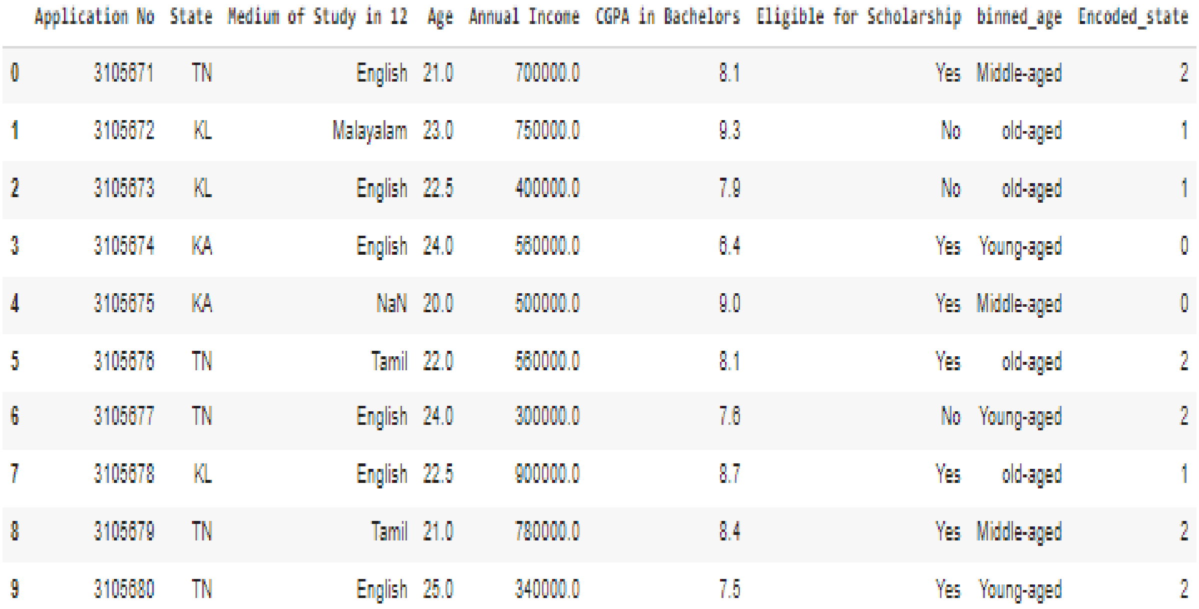


	Application No	State	Medium of Study in 12	Age	Annual Income	CGPA in Bachelors	Eligible for Scholarship	binned_age
0	3105671	TN	English	21.0	700000.0	8.1	Yes	Middle-aged
1	3105672	KL	Malayalam	23.0	750000.0	9.3	No	old-aged
2	3105673	KL	English	22.5	400000.0	7.9	No	old-aged
3	3105674	KA	English	24.0	560000.0	8.4	Yes	Young-aged
4	3105675	KA	NaN	20.0	500000.0	9.0	Yes	Middle-aged
5	3105676	TN	Tamil	22.0	560000.0	8.1	Yes	old-aged
6	3105677	TN	English	24.0	300000.0	7.6	No	Young-aged
7	3105678	KL	English	22.5	900000.0	8.7	Yes	old-aged
8	3105679	TN	Tamil	21.0	780000.0	8.4	Yes	Middle-aged
9	3105680	TN	English	25.0	340000.0	7.5	Yes	Young-aged

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

dataset['Encoded_state'] = encoder.fit_transform(dataset1['State'])
```

#print the table

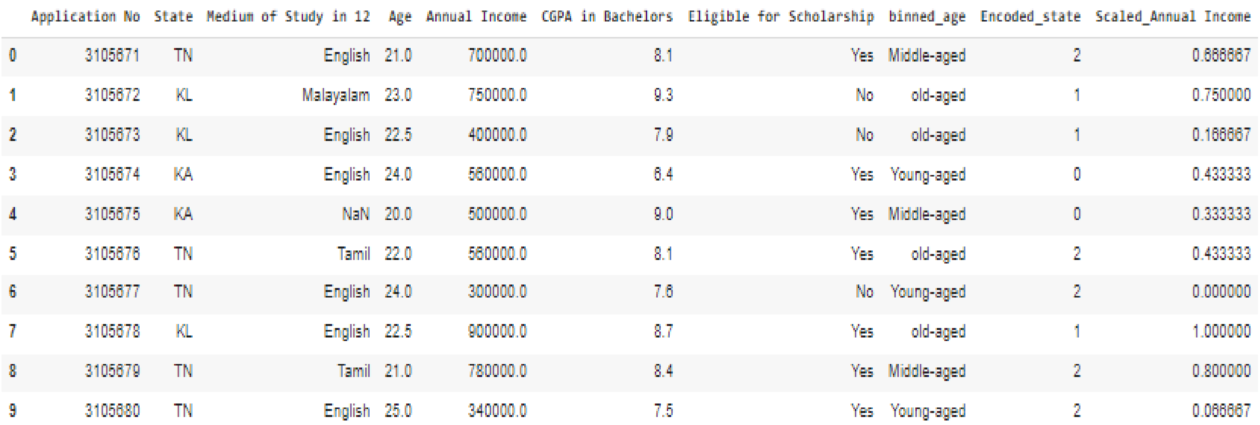
```
dataset
```



	Application No	State	Medium of Study in 12	Age	Annual Income	CGPA in Bachelors	Eligible for Scholarship	binned_age	Encoded_state
0	3105671	TN	English	21.0	700000.0	8.1	Yes	Middle-aged	2
1	3105672	KL	Malayalam	23.0	750000.0	9.3	No	old-aged	1
2	3105673	KL	English	22.5	400000.0	7.9	No	old-aged	1
3	3105674	KA	English	24.0	560000.0	8.4	Yes	Young-aged	0
4	3105675	KA	NaN	20.0	500000.0	9.0	Yes	Middle-aged	0
5	3105676	TN	Tamil	22.0	560000.0	8.1	Yes	old-aged	2
6	3105677	TN	English	24.0	300000.0	7.6	No	Young-aged	2
7	3105678	KL	English	22.5	900000.0	8.7	Yes	old-aged	1
8	3105679	TN	Tamil	21.0	780000.0	8.4	Yes	Middle-aged	2
9	3105680	TN	English	25.0	340000.0	7.5	Yes	Young-aged	2

#Min and Max Scaler

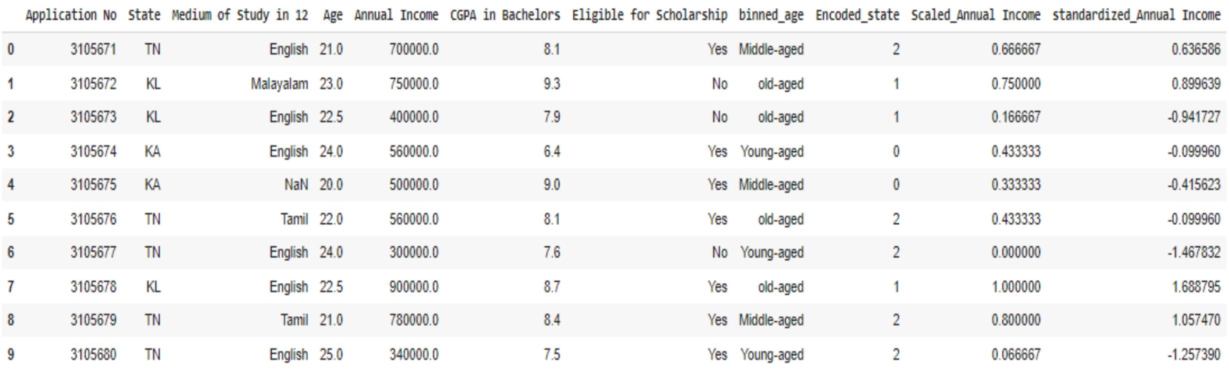
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

dataset1['Scaled_Annual Income'] = scaler.fit_transform(dataset1[['Annual Income']])
```



	Application No	State	Medium of Study in 12	Age	Annual Income	CGPA in Bachelors	Eligible for Scholarship	binned_age	Encoded_state	Scaled_Annual Income
0	3105671	TN	English	21.0	700000.0	8.1	Yes	Middle-aged	2	0.666667
1	3105672	KL	Malayalam	23.0	750000.0	9.3	No	old-aged	1	0.750000
2	3105673	KL	English	22.5	400000.0	7.9	No	old-aged	1	0.166667
3	3105674	KA	English	24.0	560000.0	6.4	Yes	Young-aged	0	0.433333
4	3105675	KA	NaN	20.0	500000.0	9.0	Yes	Middle-aged	0	0.333333
5	3105676	TN	Tamil	22.0	560000.0	8.1	Yes	old-aged	2	0.433333
6	3105677	TN	English	24.0	300000.0	7.6	No	Young-aged	2	0.000000
7	3105678	KL	English	22.5	900000.0	8.7	Yes	old-aged	1	1.000000
8	3105679	TN	Tamil	21.0	780000.0	8.4	Yes	Middle-aged	2	0.800000
9	3105680	TN	English	25.0	340000.0	7.5	Yes	Young-aged	2	0.066667

#Standard Scaler

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

dataset1['standardized_Annual Income'] = scaler.fit_transform(dataset1[['Annual Income']])
```



	Application No	State	Medium of Study in 12	Age	Annual Income	CGPA in Bachelors	Eligible for Scholarship	binned_age	Encoded_state	Scaled_Annual Income	standardized_Annual Income
0	3105671	TN	English	21.0	700000.0	8.1	Yes	Middle-aged	2	0.666667	0.636586
1	3105672	KL	Malayalam	23.0	750000.0	9.3	No	old-aged	1	0.750000	0.899639
2	3105673	KL	English	22.5	400000.0	7.9	No	old-aged	1	0.166667	-0.941727
3	3105674	KA	English	24.0	560000.0	6.4	Yes	Young-aged	0	0.433333	-0.099960
4	3105675	KA	NaN	20.0	500000.0	9.0	Yes	Middle-aged	0	0.333333	-0.415623
5	3105676	TN	Tamil	22.0	560000.0	8.1	Yes	old-aged	2	0.433333	-0.099960
6	3105677	TN	English	24.0	300000.0	7.6	No	Young-aged	2	0.000000	-1.467832
7	3105678	KL	English	22.5	900000.0	8.7	Yes	old-aged	1	1.000000	1.688795
8	3105679	TN	Tamil	21.0	780000.0	8.4	Yes	Middle-aged	2	0.800000	1.057470
9	3105680	TN	English	25.0	340000.0	7.5	Yes	Young-aged	2	0.066667	-1.257390

#Decision Tree

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import tree
```

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor

#features

features=['State','Age','Annual Income']

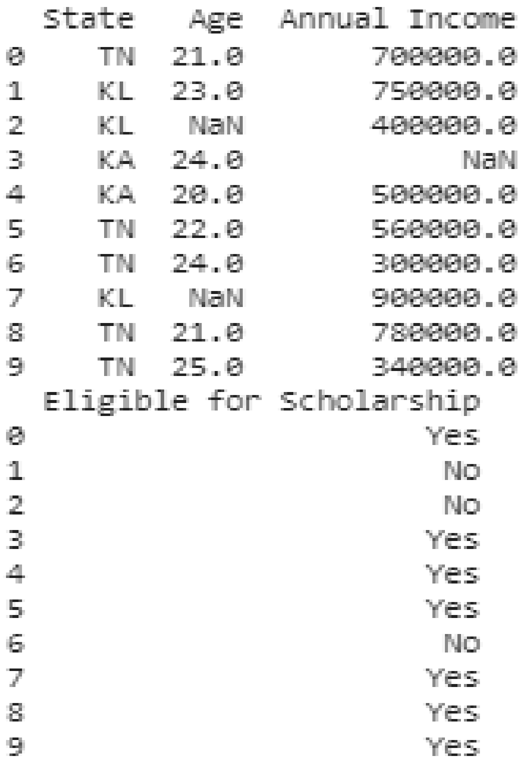
# x and y datasets

x = dataset1.loc[:, features]

y = dataset1.loc[:, ['Eligible for Scholarship']]

print(x);

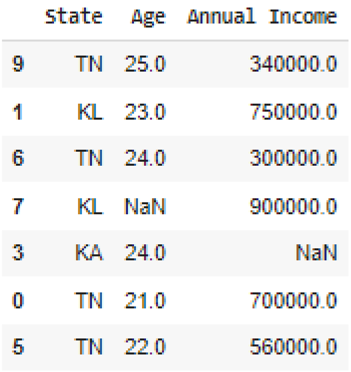
print(y);



```
  State   Age  Annual Income
0    TN  21.0       700000.0
1    KL  23.0       750000.0
2    KL   NaN       400000.0
3    KA  24.0            NaN
4    KA  20.0       500000.0
5    TN  22.0       560000.0
6    TN  24.0       300000.0
7    KL   NaN       900000.0
8    TN  21.0       780000.0
9    TN  25.0       340000.0
  Eligible for Scholarship
0                      Yes
1                       No
2                       No
3                      Yes
4                      Yes
5                      Yes
6                       No
7                      Yes
8                      Yes
9                      Yes
```

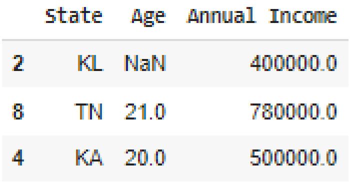
#x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, train_size = .75)

#Print x_train



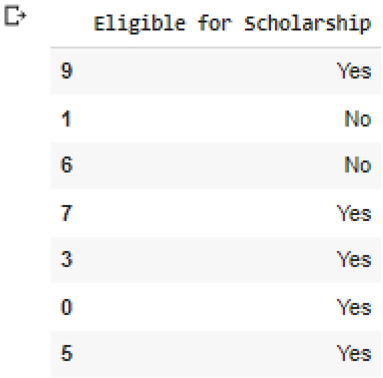
	State	Age	Annual Income
9	TN	25.0	340000.0
1	KL	23.0	750000.0
6	TN	24.0	300000.0
7	KL	NaN	900000.0
3	KA	24.0	NaN
0	TN	21.0	700000.0
5	TN	22.0	560000.0

#Print x_test



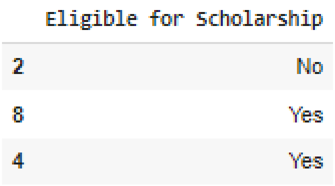
	State	Age	Annual Income
2	KL	NaN	400000.0
8	TN	21.0	780000.0
4	KA	20.0	500000.0

#Print y_train



	Eligible for Scholarship
9	Yes
1	No
6	No
7	Yes
3	Yes
0	Yes
5	Yes

#Print y_test



	Eligible for Scholarship
2	No
8	Yes
4	Yes

Result:

Thus the data pre-processing using python was executed and output was verified.

Ex. No. 3a

Analyze the Statistical and Visual Summaries of Data

Aim:

To analyze the statistical and visual summaires of the data.

Algorithm:

Step 1: Create the dataset of the table.

Step 2: Import the libraries.

Step 3: Create the dataframe of the table.

Step 4: Find the minimum and maximum then find the mean of the table.

Step 5: Find sum and median of the table..

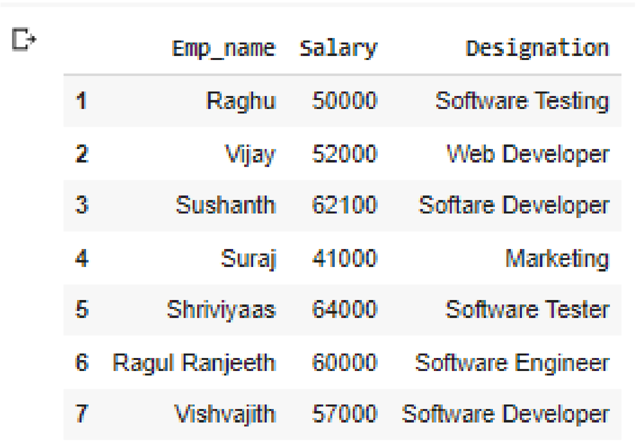
Step 6: Describe the table and summaries of the results.

Step 7: Display the output of the commands.

Source code :

```
#Create the dataset
import pandas as pd
data = [['Raghu',50000,'Software Testing'],
['Vijay',52000,'Web Developer'],
['Sushanth',62100,'Softare Developer'],
['Suraj',41000,'Marketing'],
['Shriviyaas',64000,'Software Tester'],
['Ragul Ranjeeth',60000,'Software Engineer'],
['Vishvajith',57000,'Software Developer']]
ds=pd.DataFrame(data,columns=['Emp_name','Salary','Designation'],
index=['1','2','3','4','5','6','7'])
```

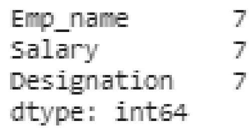
#Print ds



	Emp_name	Salary	Designation
1	Raghu	50000	Software Testing
2	Vijay	52000	Web Developer
3	Sushanth	62100	Softare Developer
4	Suraj	41000	Marketing
5	Shriviyaas	64000	Software Tester
6	Ragul Ranjeeth	60000	Software Engineer
7	Vishvajith	57000	Software Developer

#Count

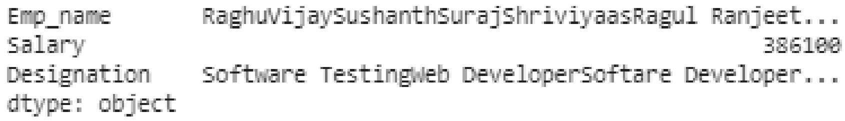
df.count()



```
Emp_name       7
Salary         7
Designation    7
dtype: int64
```

#Sum of the table

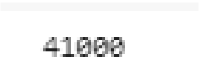
ds.sum()



```
Emp_name       RaghuVijaySushanthSurajShriviyaasRagul Ranjeet...
Salary                                                    386100
Designation    Software TestingWeb DeveloperSoftare Developer...
dtype: object
```

#Minimum

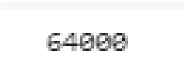
ds.Salary.min()



```
41000
```

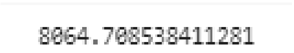
#Maximum

ds.Salary.max()



```
64000
```

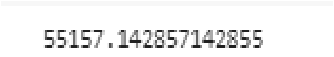
#Standard



```
8064.708538411281
```

#Mean

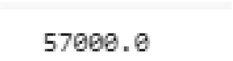
ds.Salary.mean()



```
55157.142857142855
```

#Median

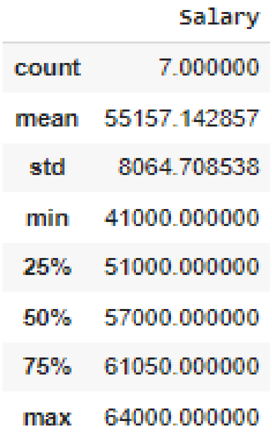
ds.Salary.median()



```
57000.0
```

#Describe the summary

ds.describe()



	Salary
count	7.000000
mean	55157.142857
std	8064.708538
min	41000.000000
25%	51000.000000
50%	57000.000000
75%	61050.000000
max	64000.000000

Result:

Thus the analysis of statistical and summaries of the data was analysed and output was verified.

Ex. No. 3b	Data Transformation

Aim:

To analyze and transform the data into forms.

Algorithm:

Step 1: Create the dataset of the table.

Step 2: Import the libraries and csv file.

Step 3: Read the data in the table and display it.

Step 4: By using the function, sort and filter the table.

Step 5: Find the duplicate and remove it.

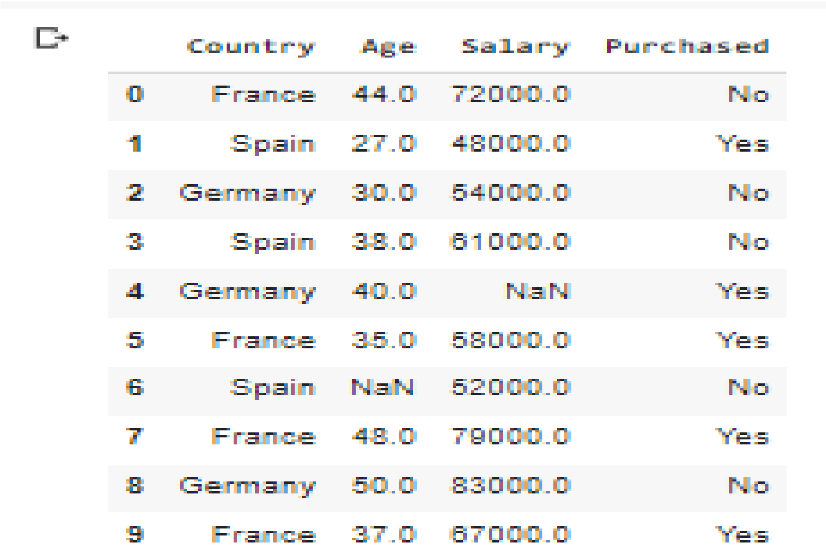
Step 6: Display the dataframe and concate it.

Step 7: Display the output of the commands.

Step 8: Stop the execution of the table.

Source code :

```
#Import the libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#Read the table

df=pd.reaad_csv(“Data.csv”)

display(df)
```

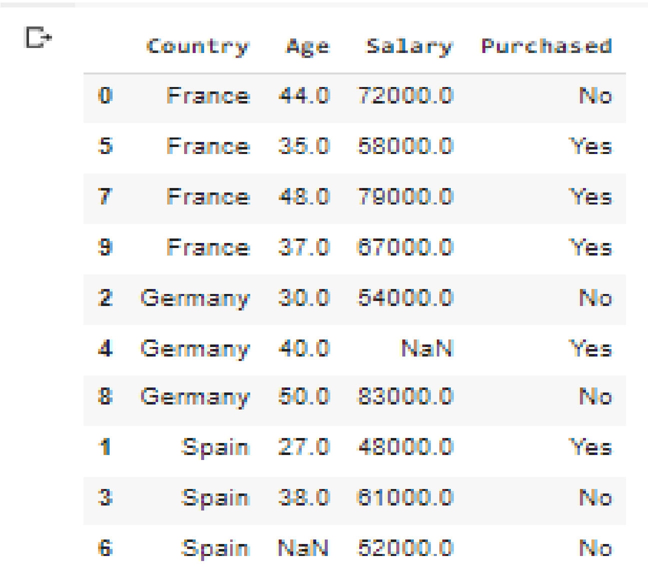


	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

#Sort the table

sorted = df.sort_values(by=['Country'])

display(sorted)

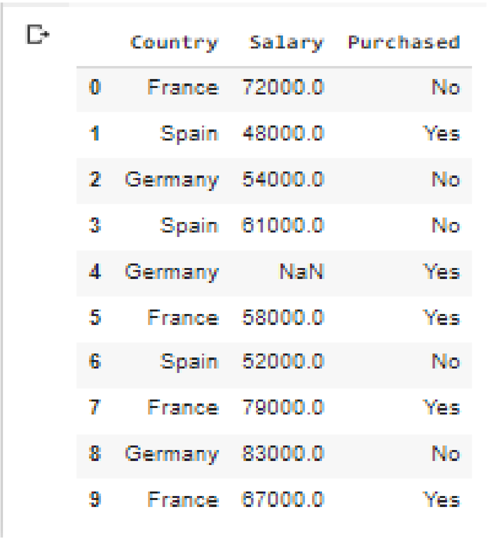


	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
5	France	35.0	58000.0	Yes
7	France	48.0	79000.0	Yes
9	France	37.0	67000.0	Yes
2	Germany	30.0	54000.0	No
4	Germany	40.0	NaN	Yes
8	Germany	50.0	83000.0	No
1	Spain	27.0	48000.0	Yes
3	Spain	38.0	61000.0	No
6	Spain	NaN	52000.0	No

#Filter columns

new = df.filter(['Country','Salary','Purchased'])

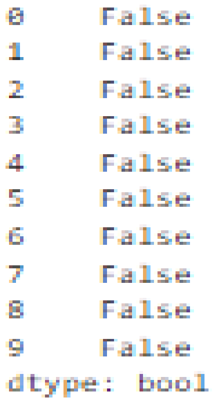
display(new)



	Country	Salary	Purchased
0	France	72000.0	No
1	Spain	48000.0	Yes
2	Germany	54000.0	No
3	Spain	61000.0	No
4	Germany	NaN	Yes
5	France	58000.0	Yes
6	Spain	52000.0	No
7	France	79000.0	Yes
8	Germany	83000.0	No
9	France	67000.0	Yes

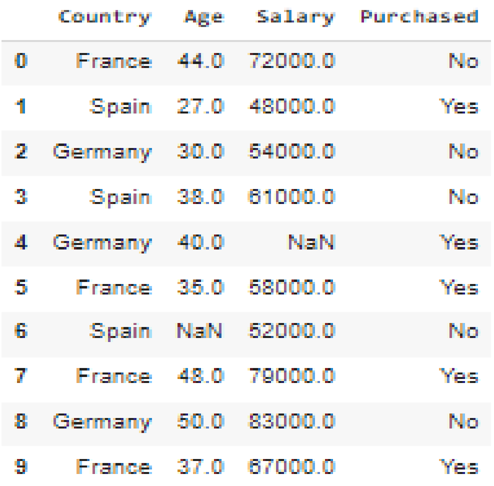
#Display the duplicates

display(df.duplicated())



```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
dtype: bool
```

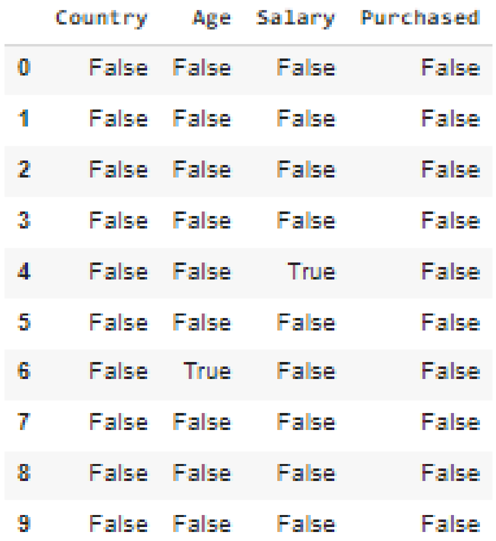
#Remove the Duplicates



	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

#Remove the null values

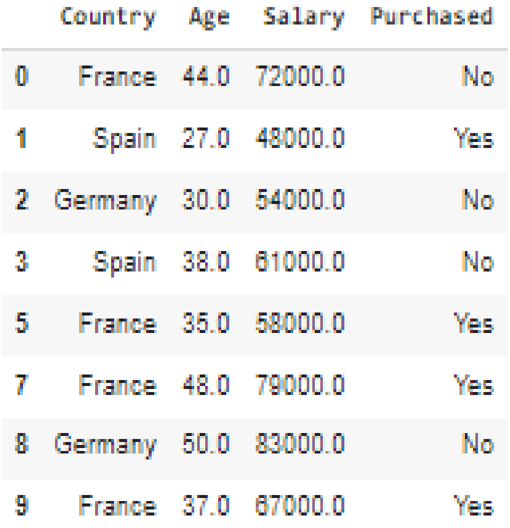
display(df.isna())



	Country	Age	Salary	Purchased
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	True	False
5	False	False	False	False
6	False	True	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False

listwise_deletion = df.dropna(how='any')

display(listwise_deletion)

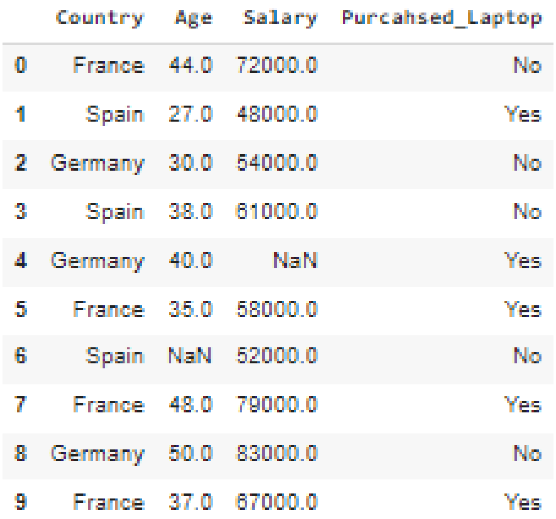


	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
5	France	35.0	58000.0	Yes
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

# Renaming columns

renamed = df.rename(acolumns={'Purchased':'Purcahsed_Laptop'})

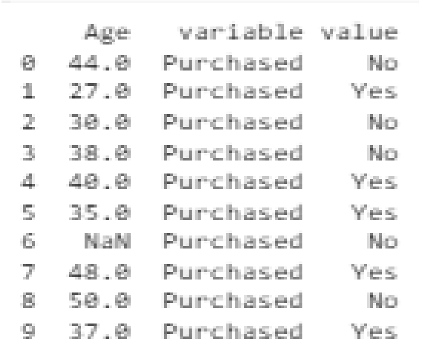
display(renamed)



	Country	Age	Salary	Purcahsed_Laptop
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

df_melted = pd.melt(df, id_vars=["Age"], value_vars=["Purchased"])

print(df_melted)



```
    Age   variable value
0  44.0  Purchased    No
1  27.0  Purchased   Yes
2  30.0  Purchased    No
3  38.0  Purchased    No
4  40.0  Purchased   Yes
5  35.0  Purchased   Yes
6   NaN  Purchased    No
7  48.0  Purchased   Yes
8  50.0  Purchased    No
9  37.0  Purchased   Yes
```

#Create the dataset

d1 = {"Name": ["Pankaj", "Lisa", "David"], "ID": [1, 2, 3], "Role": ["CEO", "Editor", "Author"]}

df = pd.DataFrame(d1)

#Print the dataset by df

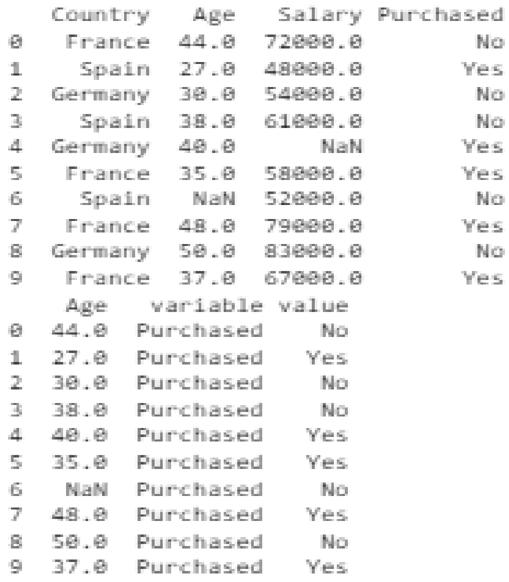
print(df)

#Melt the column

df_melted = pd.melt(df, id_vars=["Age"], value_vars=["Purchased"])

#Print the melted table

print(df_melted)



```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes
    Age   variable value
0  44.0  Purchased    No
1  27.0  Purchased   Yes
2  30.0  Purchased    No
3  38.0  Purchased    No
4  40.0  Purchased   Yes
5  35.0  Purchased   Yes
6   NaN  Purchased    No
7  48.0  Purchased   Yes
8  50.0  Purchased    No
9  37.0  Purchased   Yes
```

import pandas as pd

# First DataFrame

df1 = pd.DataFrame({'id': ['A01', 'A02', 'A03', 'A04'],

'Name': ['ABC', 'PQR', 'DEF', 'GHI']})

# Second DataFrame

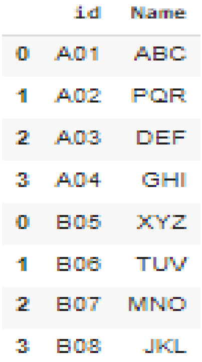
df2 = pd.DataFrame({'id': ['B05', 'B06', 'B07', 'B08'],

'Name': ['XYZ', 'TUV', 'MNO', 'JKL']})

frames = [df1, df2]

result = pd.concat(frames)

display(result)



	id	Name
0	A01	ABC
1	A02	PQR
2	A03	DEF
3	A04	GHI
0	B05	XYZ
1	B06	TUV
2	B07	MNO
3	B08	JKL

Result:

Thus the data transformation of data into the forms was executed and output was verified.

Ex. No. 4	Data Visualization Techniques using Python

Aim:

To analyze and implement the data visualization techniques using python.

Algorithm:

Step 1: Create the dataset of the table.

Step 2: Import the libraries and csv file.

Step 3: Read the data in the table and display it.

Step 4: By using the function, sort and filter the table.

Step 5: Find the duplicate and remove it.

Step 6: Display the dataframe and concate it.

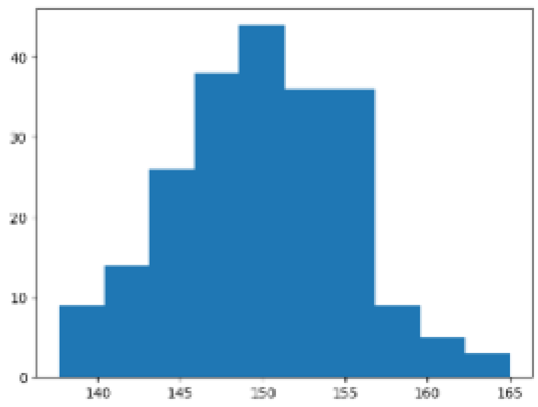
Step 7: Display the output of the commands.

Step 8: Stop the execution of the table.

Source code :

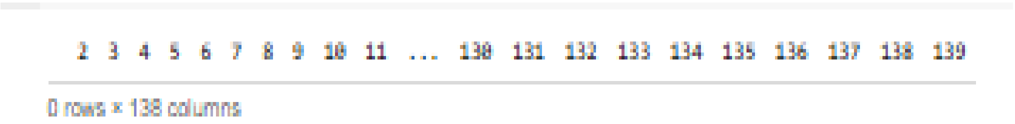
#To print the Histogram

```
import matplotlib.pyplot as plt
import numpy as np
x=np.random.normal(150,5,220)
plt.hist(x)
plt.show()
```



#Read the csv file

```
import pandas as pd

import numpy as np

data=pd.read_csv('/content/data.csv')

#Print the csv file

data
```

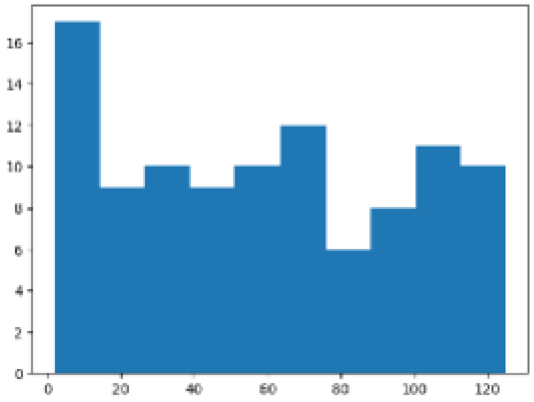


```
   2  3  4  5  6  7  8  9  10  11  ...  130  131  132  133  134  135  136  137  138  139

0 rows × 138 columns
```

#Create the data and print the histogram

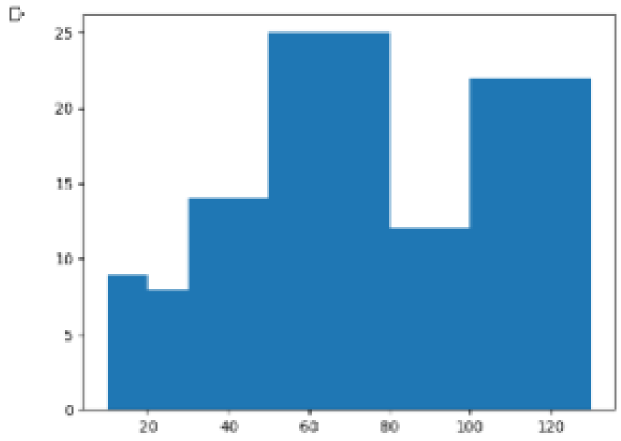
```
import matplotlib.pyplot as plt

x=[2,3,4,5,6,7,8,

   9,10,11,12,13,

   5,16,17,18,19,

   20,21,24,25,26,

   27,28,29,30,31,

   32,35,36,37,38,

   39,40,41,42,47,

   48,49,50,51,52,

   5,54,55,56,57,

   58,9,60,61,62,

   63,64,65,66,67,

   68,69,70,71,72,

   73,74,75,76,2,

   83,84,86,87,88,

   89,90,91,92,97,

   98,99,100,101,102,

   103,104,105,106,107,

   108,109,110,111,11,

   116,117,118,119,120,

   121,122,123,124,125,]
```

```
plt.hist(x,bins=10)
plt.show()
```



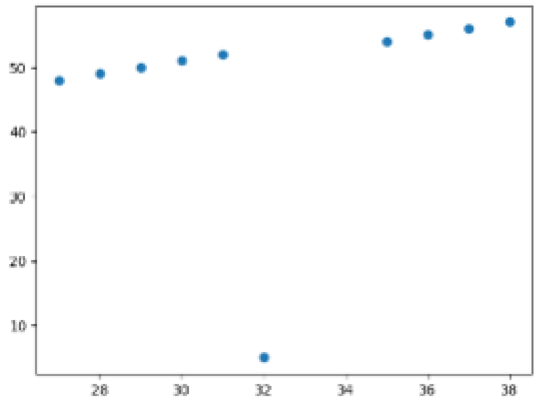
#Change the dimensions

```
import matplotlib.pyplot as plt
x=[2,3,4,5,6,7,8,
   9,10,11,12,13,
   5,16,17,18,19,
   20,21,24,25,26,
   27,28,29,30,31,
   32,35,36,37,38,
   39,40,41,42,47,
   48,49,50,51,52,
   5,54,55,56,57,
   58,9,60,61,62,
   63,64,65,66,67,
   68,69,70,71,72,
   73,74,75,76,2,
   83,84,86,87,88,
   89,90,91,92,97,
   98,99,100,101,102,
   103,104,105,106,107,
   108,109,110,111,11,
   116,117,118,119,120,
    121,122,123,124,125,]
plt.hist(x,bins=[10,20,30,50,80,100,130])
plt.show()
```



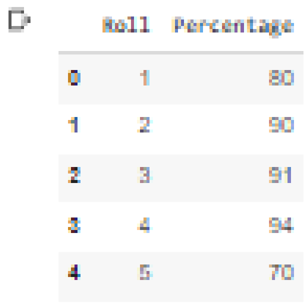
#Scatter plots

```
import matplotlib.pyplot as plt
x=[27,28,29,30,31,32,35,36,37,38,]
y=[48,49,50,51,52,5,54,55,56,57,]
plt.scatter(x,y)
plt.show()
```



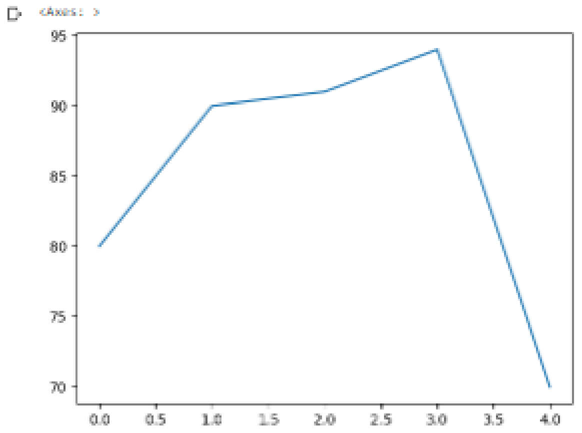
#Import library and read the csv file

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv("data1.csv")
data.head()
```



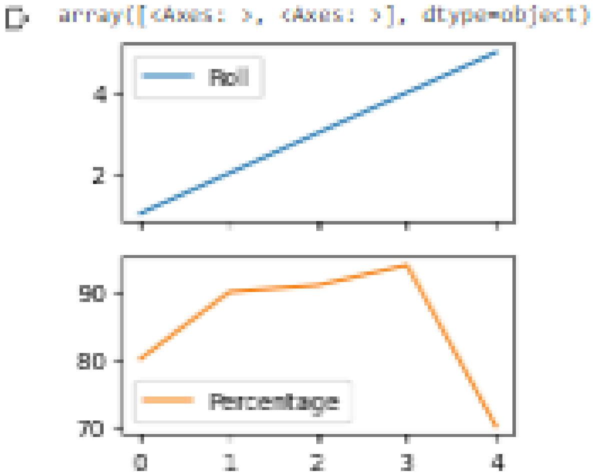
	Roll	Percentage
0	1	80
1	2	90
2	3	91
3	4	94
4	5	70

data['Percentage'].plot()



```
<Axes: >
```

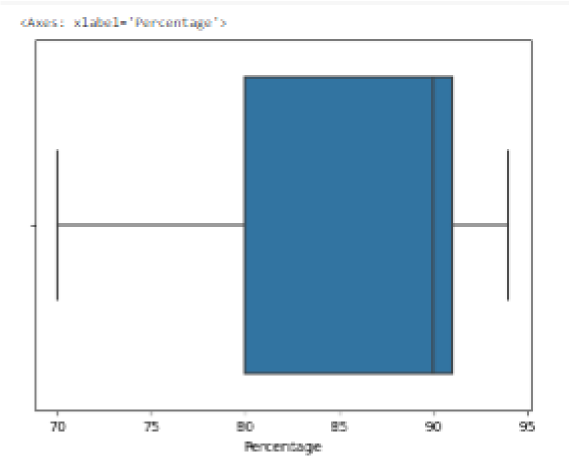
data.plot(subplots=True,figsize=(3,3))



```
array([<Axes: >, <Axes: >], dtype=object)
```

#Box Plot

sns.boxplot(data=data,x="Percentage")

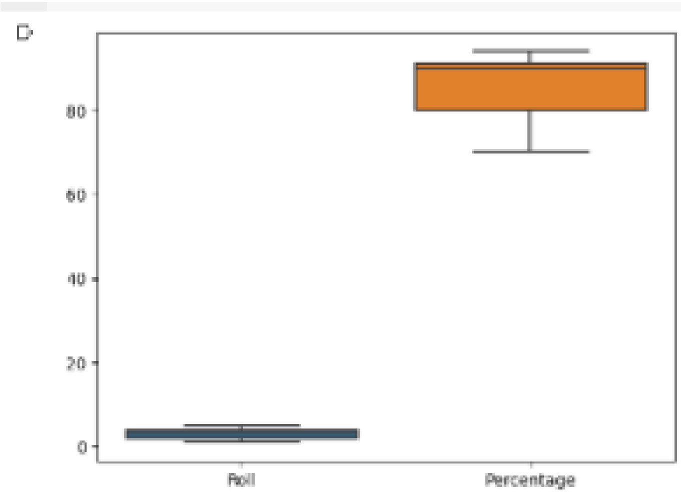


```
<Axes: xlabel='Percentage'>
```

#Combine the box plot

sns.boxplot(data=data)

plt.show()



Result:

Thus the data visualization techniques was executed and output was verified.

Ex. No. 5a

Exploratory Data Mining : Apriori Algorithm

Aim:

To implement the Exploratory data mining: Apriori Algorithm using python .

Algorithm:

Step 1: Import the necessary library.

Step 2: Computing the support for each individual item.

Step 3: Deciding on the support threshold.

Step 4: Selecting the frequent items.

Step 5: Finding the support of the frequent itemset.

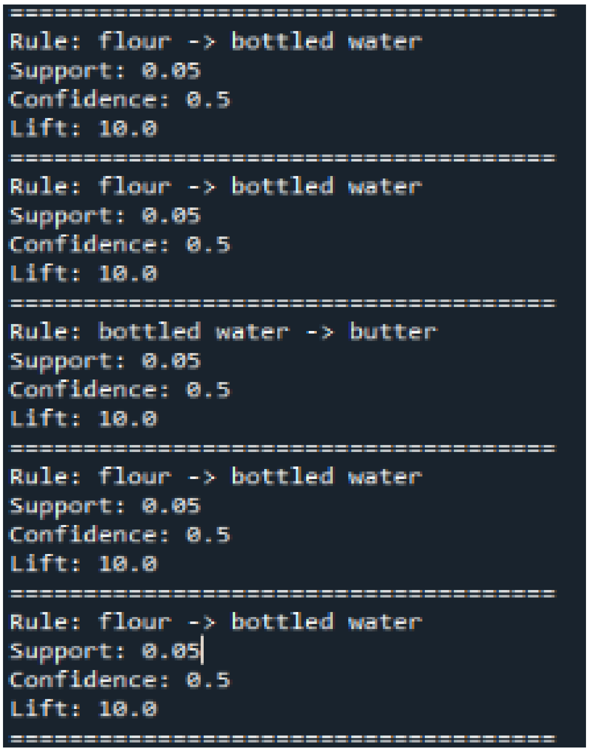
Step 6: Repeat for larger sets

Step 7: Generate Association Rules and compute Support, Confidence, Lift.

Source code :

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from apyori import apriori
dataset = pd.read_csv("C://Users//Admin//dataset.csv")
transactions = []
for i in range(0, 20):
    transactions.append([str(dataset.values[i,u]) for u in range(1, 20)])
rules = apriori(transactions, min_support=0.003, min_confidence=0.2, min_lift=2,
    min_length = 2)
results= list(rules)
for item in results:
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])
print("Support: " + str(item[1]))
print("Confidence: " + str(item[2][0][2]))
print("Lift: " + str(item[2][0][3]))
print("=====================================")
```

Output:



```
=====================================
Rule: flour -> bottled water
Support: 0.05
Confidence: 0.5
Lift: 10.0
=====================================
Rule: flour -> bottled water
Support: 0.05
Confidence: 0.5
Lift: 10.0
=====================================
Rule: bottled water -> butter
Support: 0.05
Confidence: 0.5
Lift: 10.0
=====================================
Rule: flour -> bottled water
Support: 0.05
Confidence: 0.5
Lift: 10.0
=====================================
Rule: flour -> bottled water
Support: 0.05
Confidence: 0.5
Lift: 10.0
=====================================
```

Result:

Thus, the implementation of the Apriori Algorithm had been completed successfully and the output had been verified.

Ex. No. 5b

Exploratory Data Mining: FP Growth Algorithm

Aim:

To implement the Exploratory data mining FP Growth Algorithm using Python.

Algorithm:

Step 1: Import the necessary.

Step 2: Counting the occurrences of individual items.

Step 3: Filter out non-frequent items using minimum support.

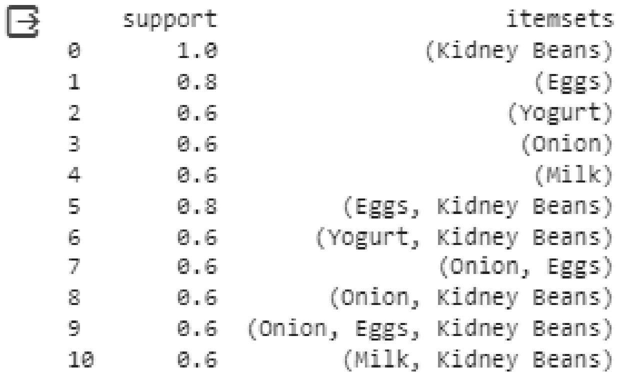
Step 4: Order the itemset based on individual occurrences.

Step 5: Create the tree and add the transactions one by one.

Source code :

```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

dataset1 = [['Milk', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

        ['Dill', 'Onion', 'Nutmeg', 'Kidney Beans', 'Eggs', 'Yogurt'],

        ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],

        ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],

        ['Corn', 'Onion', 'Onion', 'Kidney Beans', 'Ice cream', 'Eggs']]

te = TransactionEncoder()

te_ary = te.fit(dataset1).transform(dataset1)

df = pd.DataFrame(te_ary, columns=te.columns_)

from mlxtend.frequent_patterns import fpgrowth

fpgrowth(df, min_support=0.6)

print(fpgrowth(df, min_support=0.6, use_colnames=True))
```

Output:



```
    support                     itemsets
0       1.0               (Kidney Beans)
1       0.8                       (Eggs)
2       0.6                     (Yogurt)
3       0.6                      (Onion)
4       0.6                       (Milk)
5       0.8         (Eggs, Kidney Beans)
6       0.6       (Yogurt, Kidney Beans)
7       0.6                (Onion, Eggs)
8       0.6        (Onion, Kidney Beans)
9       0.6  (Onion, Eggs, Kidney Beans)
10      0.6         (Milk, Kidney Beans)
```

Result:

Thus, the implementation of the FP Growth Algorithm had been completed successfully and output had been verified.

Ex. No. 5c	Exploratory Data Mining: K Means Clustering

Aim:

To implement the Exploratory data mining K Means Clustering Algorithm using Python.

Algorithm:

Step 1: Select the Number of Clusters.

Step 2: Select ‘k’ Points at Random.

Step 3: Make ‘k’ Clusters.

Step 4: Compute the new Centroid of Each Cluster.

Step 5: Assess the Quality of Each Cluster.

Step 6: Repeat the new Centroid cluster then quality of cluster.

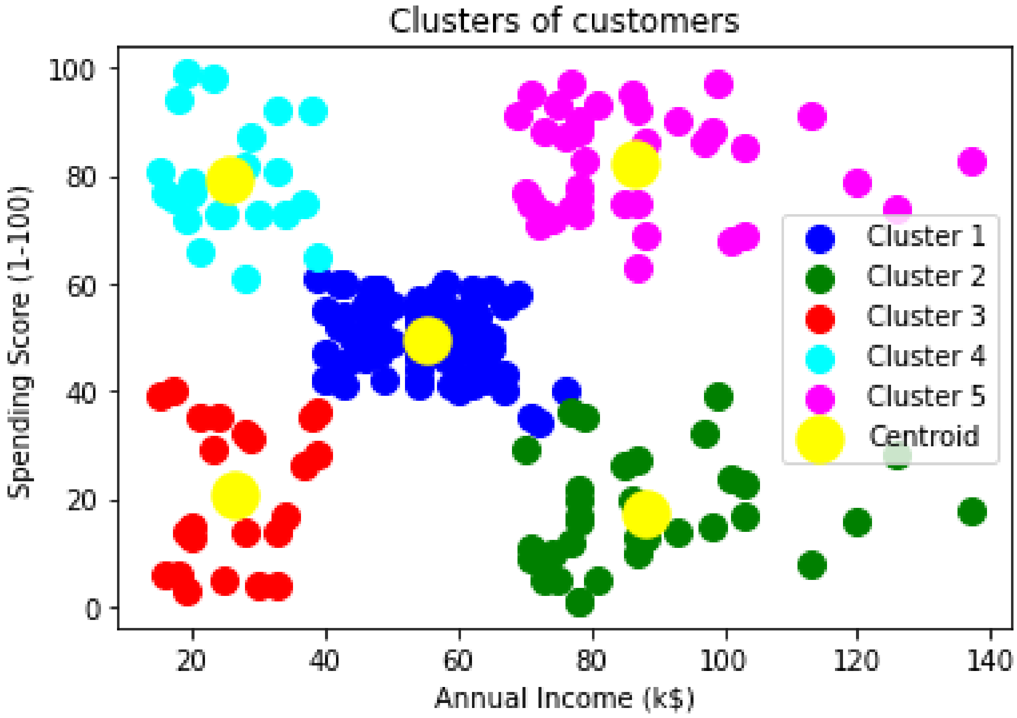
Step 7: Prune the Tree to prevent overfitting.

Source code :

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

dataset = pd.read_csv('C://Users//Admin//dats.csv')

x = dataset.iloc[:, [3, 4]].values

from sklearn.cluster import KMeans

wcss_list= []

for i in range(1, 11):

    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)

    kmeans.fit(x)

    wcss_list.append(kmeans.inertia_)
```

```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)

y_predict= kmeans.fit_predict(x)

mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #for first cluster

mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster

mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #for third cluster

mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #for fourth cluster

mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster

mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')

mtp.title('Clusters of customers')

mtp.xlabel('Annual Income (k$)')

mtp.ylabel('Spending Score (1-100)')

mtp.legend()

mtp.show()
```

Output:



Result:

Thus, the implementation of the K Means Algorithm had been completed successfully and output had been verified.

Ex. No. 5d	**Exploratory Data Mining: Classification Decision Tree**

Aim:

To implement the Exploratory data mining Decision Tree Algorithm using Python.

Algorithm:

Step 1: Choose the initial dataset with the feature and target attributes defined.

Step 2: Calculate the information gain and Entropy for each attribute.

Step 3: Pick the attribute with the highest information gain and make it the decision root node.

Step 4: Calculate the information gain for the remaining attributes.

Step 5: Create recurring child nodes by starting splitting at the decision node (i.e. for various values of the decision node, create, separate child nodes.

Step 6: Repeat this process until all the attributes are covered.

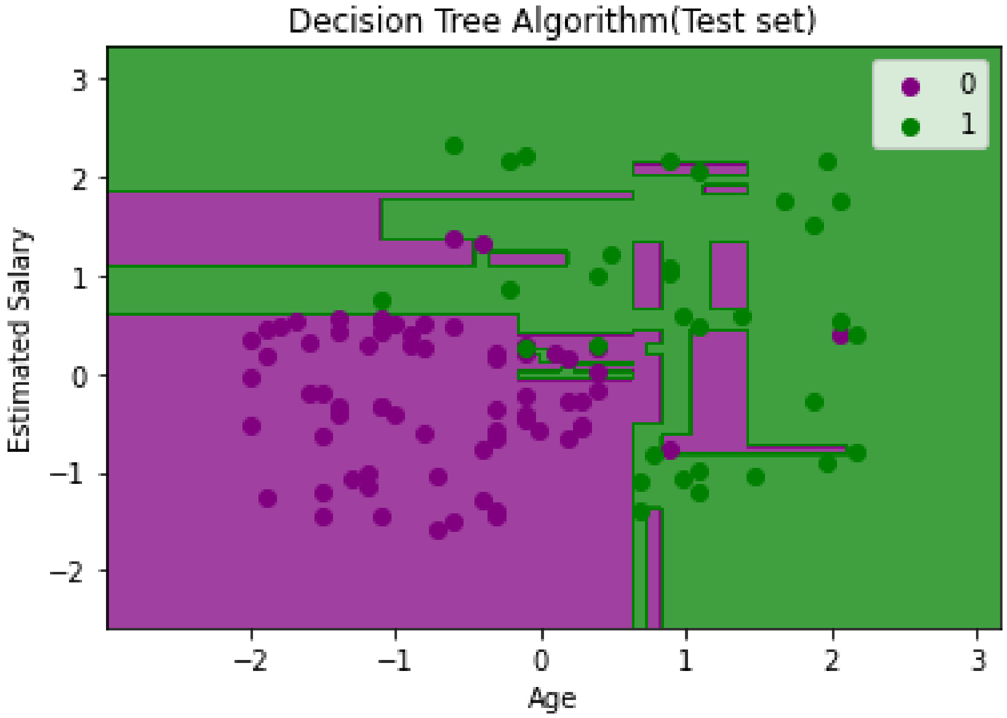
Step 7: Prune the Tree to prevent overfitting.

Source code :

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

data_set= pd.read_csv('C://Users//Admin//dec.csv')

x= data_set.iloc[:, [2,3]].values

y= data_set.iloc[:, 4].values
```

```
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

st_x= StandardScaler()

x_train= st_x.fit_transform(x_train)

x_test= st_x.transform(x_test)

classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)

classifier.fit(x_train, y_train)

y_pred= classifier.predict(x_test)

cm = confusion_matrix(y_test, y_pred)

from matplotlib.colors import ListedColormap

x_set, y_set = x_test, y_test

x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),

nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))

mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),

alpha = 0.75, cmap = ListedColormap(('purple','green' )))

mtp.xlim(x1.min(), x1.max())

mtp.ylim(x2.min(), x2.max())

for i, j in enumerate(nm.unique(y_set)):

    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

            c = ListedColormap(('purple', 'green'))(i), label = j)

    mtp.title('Decision Tree Algorithm(Test set)')

    mtp.xlabel('Age')

    mtp.ylabel('Estimated Salary')

    mtp.legend()
mtp.show()
```

Output:



Result:

Thus, the implementation of Decision Tree Algorithm had been completed successfully and output had been verified.

Ex. No. 5e

Exploratory Data Mining: Support Vector Machine

Aim:

To implement the Exploratory data mining Decision Tree Algorithm using Python.

Algorithm:

Step 1: Load the important libraries.

Step 2: Import the dataset and extract the X variables and Y separately.

Step 3: Divide the dataset into train and test.

Step 4: Initializing the SVM classifier model.

Step 5: Fitting the SVM classifier model.

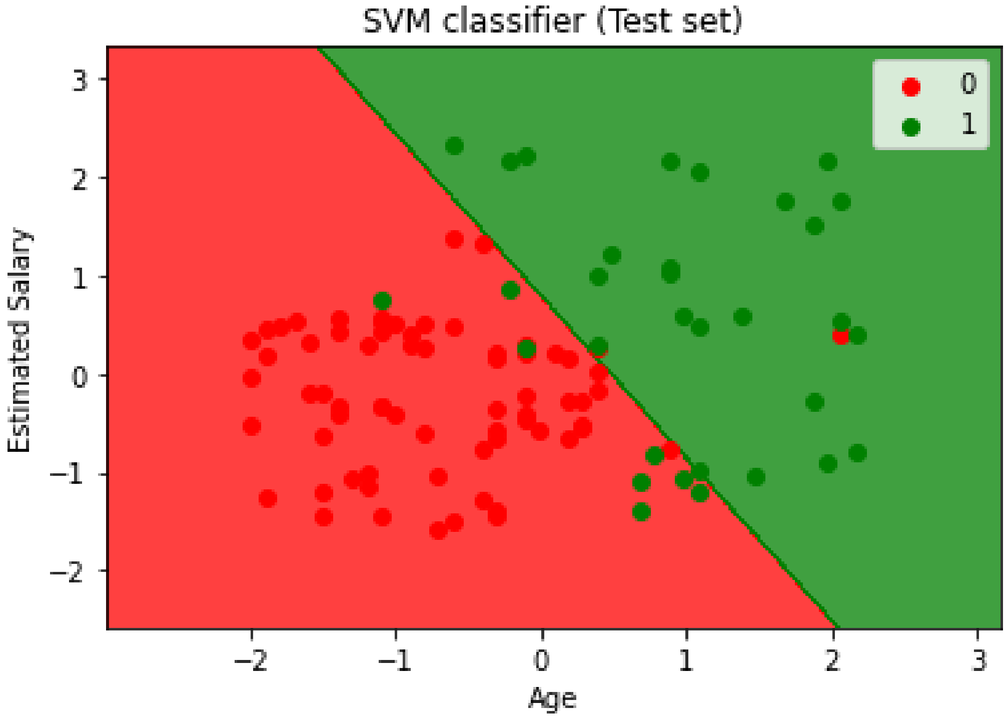
Step 6: Coming up with predictions.

Source code :

```
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion_matrix

from sklearn.svm import SVC

from matplotlib.colors import ListedColormap

data_set= pd.read_csv('C://Users//Admin//dec.csv')

x= data_set.iloc[:, [2,3]].values

y= data_set.iloc[:, 4].values

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
```

```
st_x= StandardScaler()

x_train= st_x.fit_transform(x_train)

x_test= st_x.transform(x_test)

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

classifier.fit(x_train, y_train)

y_pred= classifier.predict(x_test)

cm= confusion_matrix(y_test, y_pred)

x_set, y_set = x_test, y_test

x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),

nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))

mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),

alpha = 0.75, cmap = ListedColormap(('red','green' )))

mtp.xlim(x1.min(), x1.max())

mtp.ylim(x2.min(), x2.max())

for i, j in enumerate(nm.unique(y_set)):

    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

        c = ListedColormap(('red', 'green'))(i), label = j)

    mtp.title('SVM classifier (Test set)')

    mtp.xlabel('Age')

    mtp.ylabel('Estimated Salary')

    mtp.legend()

mtp.show()
```

Output:



Result:

Thus, the implementation of the Support Vector Machine had been completed successfully and output had been verified.

Ex. No. 6

Implementation of Hierarchical Clustering

Aim:

To implement the Hierarchical Clustering using Python.

Algorithm:

Step 1: Import the necessary library.

Step 2: Separate a X and Y variables.

Step 3: Assign affinity and metric.

Step 4: Perform the Agglomerative Clustering function used to group objects in clusters based on their similarity.

Step 5: Use dendrogram, which represent the clusters to which the data belong, with the arrows representing the distance.

Step 6: Use fit method to fit the model.

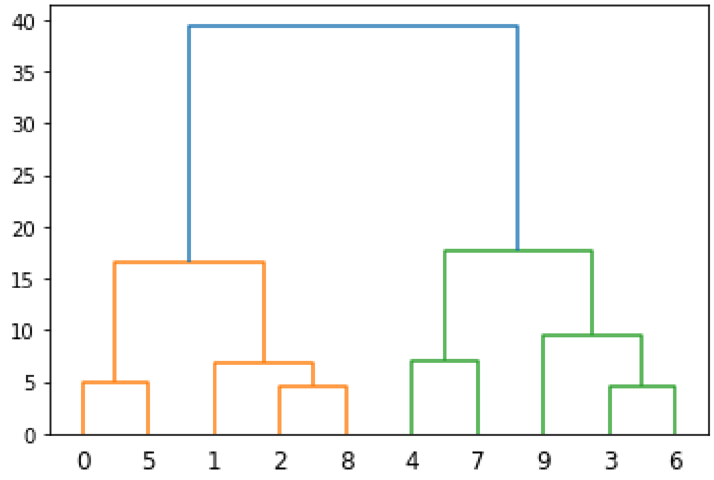
Step 7: Print the labels and show the plots.

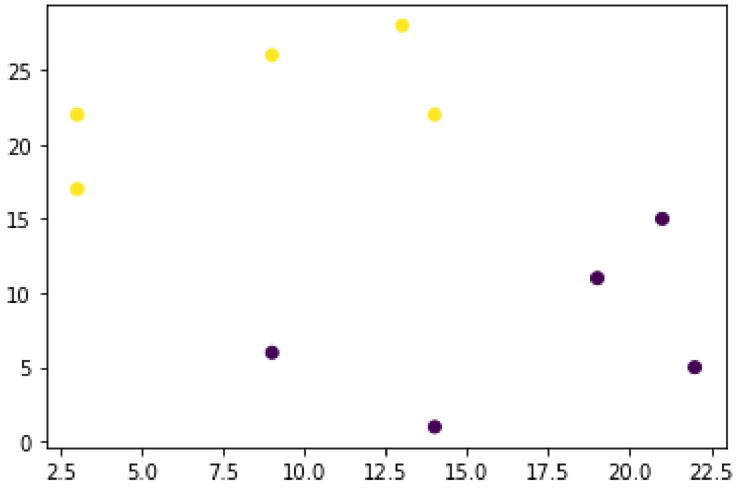
Source code :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
x = np.random.randint(30, size=10)
y = np.random.randint(30, size=10)
data = list(zip(x, y))
linkage_data = linkage(data, method='ward', metric='euclidean')
dendrogram(linkage_data)
plt.show()
```

```
hierarchical_cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
linkage='ward')

labels = hierarchical_cluster.fit_predict(data)

print(labels)

plt.scatter(x, y, c=labels)

plt.show()
```

Output:





Result:

Thus, the implementation the Hierarchical Clustering Algorithm had been completed successfully and output had been verified.

Ex. No. 7a	Implementation of Linear Regression

Aim:

To implement the Linear Regression using Python.

Algorithm:

Step 1: Import the necessary library.

Step 2: Analyzing the correlation and directionality of the data.

Step 3: Estimating the model, i.e., fitting the line.

Step 4: Evaluating the validality and usefulness of the model.

Step 5: Python SciPy scipy.stats.linregress method is used to calculate the parameters that establish a linear relationship between two sets of variables using the least-squares method.

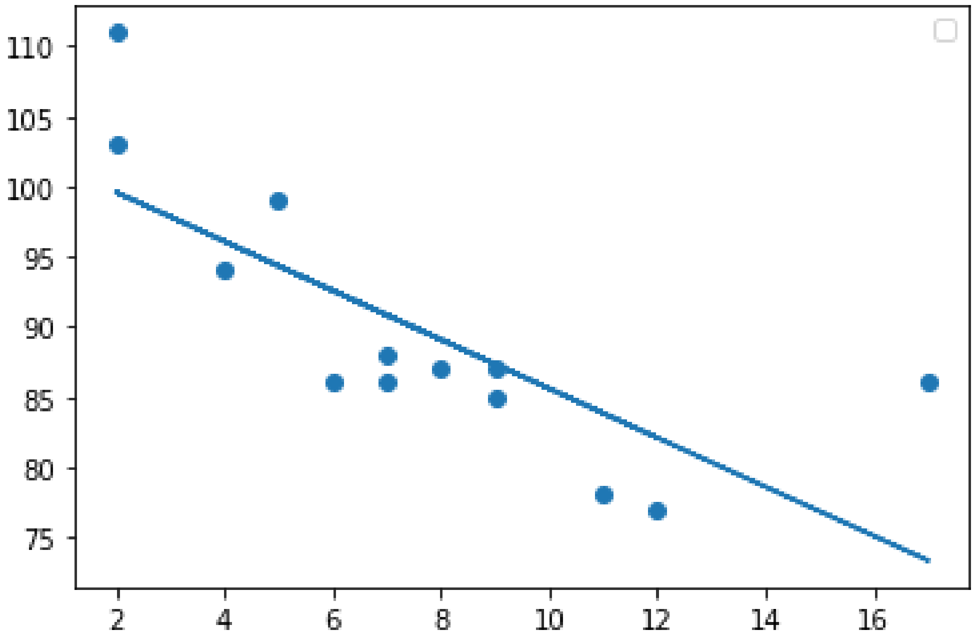
Step 6: Define a function and return slope*+intercept.

Source code :

```
from scipy import stats

import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):

   return slope * x + intercept

model = list(map(myfunc, x))

if(abs(r) < 0.5):

   print("Bad Fit")
```

```
else:
    print("Good Fit")
plt.scatter(x,y)
plt.plot(x, model)
plt.legend()
plt.show()
```

Output:



Result:

Thus, the implementation the Linear Regression had been completed successfully and output had been verified.

Ex. No. 7b

Implementation of Multiple Regression

Aim:

To implement the Multiple Regression using Python.

Algorithm:

Step 1: Import the necessary library.

Step 2: Read the CSV file.

Step 3: Check the relationship between each predictor variable and the response variable, this could be done using scatterplots and correlations.

Step 4: Try and analyze the simple linear regression between the predictor and response variable.

Step 5: Use the best-fitting model to make a prediction based on the predictor.

Step 6: Polynomial feature generates a new feature matrix consisting of all polynomials combinations of the features with a degree less than or equal to the specified degree.

Step 7: Use the fit method to fit the model.

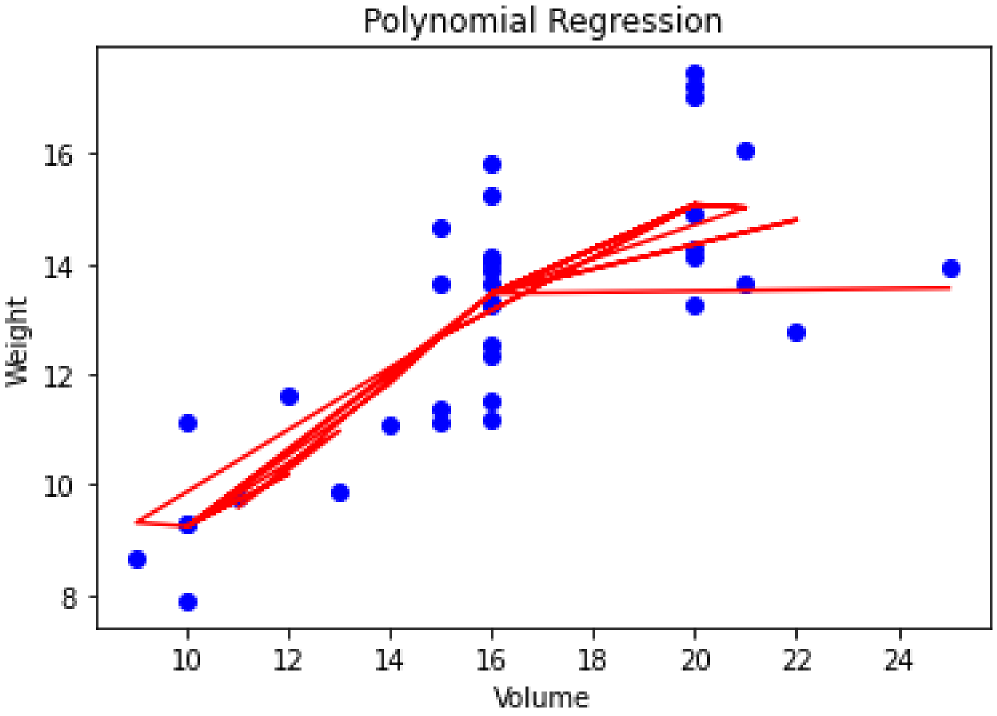
Step 8: Print the labels and show the plots.

Source code :

```
import matplotlib.pyplot as plt

import pandas as pd

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import LinearRegression

datas = pd.read_csv('data1.csv')

x = (datas.iloc[:, 2:3].values)/100

y = (datas.iloc[:, 3].values)/100

poly = PolynomialFeatures(degree = 4)
```

```
x_poly = poly.fit_transform(x)
poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(x_poly, y)
plt.scatter(x, y, color = 'blue')
plt.plot(x,lin2.predict(poly.fit_transform(X)), color = 'red')
plt.title('Polynomial Regression')
plt.xlabel('Volume')
plt.ylabel('Weight')
plt.show()
```

Output:



Result:

Thus, the implementation the Multiple Regression had been completed successfully and output had been verified.

Ex. No. 8	Case Study

Aim:

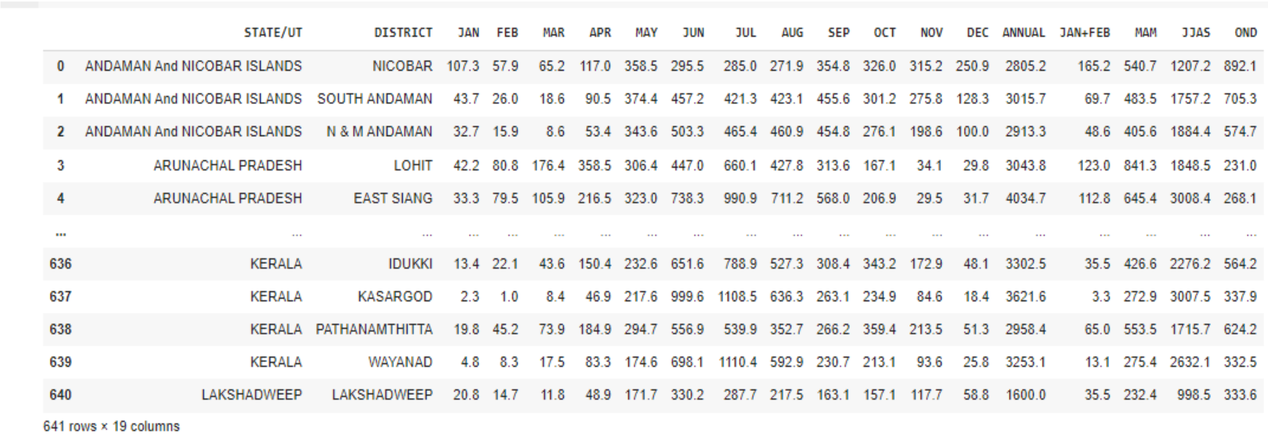
To analyse the statistical data and visualization techniques to display visual representation using Python.

Introduction:

Climate change analytics, by analyzing extensive climate-related datasets, reveals vital insights into the Earth's changing climate. It uncovers trends like rising temperatures and extreme weather events, offering data-driven forecasts and impact assessments. Visualizations effectively communicate findings to policymakers and the public, fostering awareness and informed decisions. Beyond observation, it guides action by prioritizing mitigation and adaptation strategies. Continuous monitoring, collaboration, and ethical data practices are integral. Climate change analytics empowers us to advocate for climate action, make informed decisions, and work towards a sustainable future in the face of climate change's challenges.

Commands:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

d=pd.read_excel("/content/District_Rainfall_Normal_0.xls")

display(d)
```



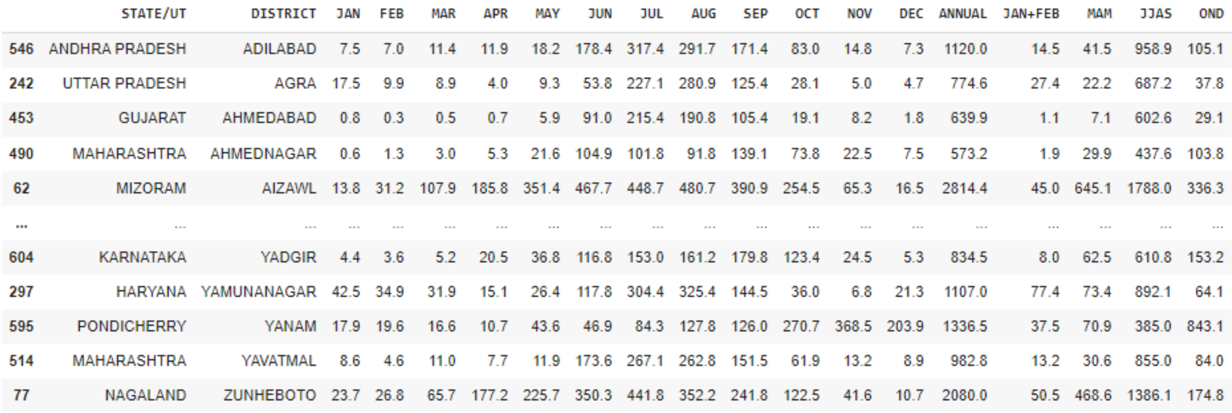
	STATE/UT	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN+FEB	MAM	JJAS	OND
0	ANDAMAN And NICOBAR ISLANDS	NICOBAR	107.3	57.9	65.2	117.0	358.5	295.5	285.0	271.9	354.8	326.0	315.2	250.9	2805.2	165.2	540.7	1207.2	892.1
1	ANDAMAN And NICOBAR ISLANDS	SOUTH ANDAMAN	43.7	26.0	18.6	90.5	374.4	457.2	421.3	423.1	455.6	301.2	275.8	128.3	3015.7	69.7	483.5	1757.2	705.3
2	ANDAMAN And NICOBAR ISLANDS	N & M ANDAMAN	32.7	15.9	8.6	53.4	343.6	503.3	465.4	460.9	454.8	276.1	198.6	100.0	2913.3	48.6	405.6	1884.4	574.7
3	ARUNACHAL PRADESH	LOHIT	42.2	80.8	176.4	358.5	306.4	447.0	660.1	427.8	313.6	167.1	34.1	29.8	3043.8	123.0	841.3	1848.5	231.0
4	ARUNACHAL PRADESH	EAST SIANG	33.3	79.5	105.9	216.5	323.0	738.3	990.9	711.2	568.0	206.9	29.5	31.7	4034.7	112.8	645.4	3008.4	268.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
636	KERALA	IDUKKI	13.4	22.1	43.6	150.4	232.6	651.6	788.9	527.3	308.4	343.2	172.9	48.1	3302.5	35.5	426.6	2276.2	564.2
637	KERALA	KASARGOD	2.3	1.0	8.4	46.9	217.6	999.6	1108.5	636.3	263.1	234.9	84.6	18.4	3621.6	3.3	272.9	3007.5	337.9
638	KERALA	PATHANAMTHITTA	19.8	45.2	73.9	184.9	294.7	556.9	539.9	352.7	266.2	359.4	213.5	51.3	2958.4	65.0	553.5	1715.7	624.2
639	KERALA	WAYANAD	4.8	8.3	17.5	83.3	174.6	698.1	1110.4	592.9	230.7	213.1	93.6	25.8	3253.1	13.1	275.4	2632.1	332.5
640	LAKSHADWEEP	LAKSHADWEEP	20.8	14.7	11.8	48.9	171.7	330.2	287.7	217.5	163.1	157.1	117.7	58.8	1600.0	35.5	232.4	998.5	333.6

641 rows × 19 columns

#Sort the values

sorted=d.sort_values(by=['DISTRICT'])

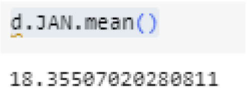
display(sorted)



	STATE/UT	DISTRICT	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN+FEB	MAM	JJAS	OND
546	ANDHRA PRADESH	ADILABAD	7.5	7.0	11.4	11.9	18.2	178.4	317.4	291.7	171.4	83.0	14.8	7.3	1120.0	14.5	41.5	958.9	105.1
242	UTTAR PRADESH	AGRA	17.5	9.9	8.9	4.0	9.3	53.8	227.1	280.9	125.4	28.1	5.0	4.7	774.6	27.4	22.2	687.2	37.8
453	GUJARAT	AHMEDABAD	0.8	0.3	0.5	0.7	5.9	91.0	215.4	190.8	105.4	19.1	8.2	1.8	639.9	1.1	7.1	602.6	29.1
490	MAHARASHTRA	AHMEDNAGAR	0.6	1.3	3.0	5.3	21.6	104.9	101.8	91.8	139.1	73.8	22.5	7.5	573.2	1.9	29.9	437.6	103.8
62	MIZORAM	AIZAWL	13.8	31.2	107.9	185.8	351.4	467.7	448.7	480.7	390.9	254.5	65.3	16.5	2814.4	45.0	645.1	1788.0	336.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
604	KARNATAKA	YADGIR	4.4	3.6	5.2	20.5	36.8	116.8	153.0	161.2	179.8	123.4	24.5	5.3	834.5	8.0	62.5	610.8	153.2
297	HARYANA	YAMUNANAGAR	42.5	34.9	31.9	15.1	26.4	117.8	304.4	325.4	144.5	36.0	6.8	21.3	1107.0	77.4	73.4	892.1	64.1
595	PONDICHERRY	YANAM	17.9	19.6	16.6	10.7	43.6	46.9	84.3	127.8	126.0	270.7	368.5	203.9	1336.5	37.5	70.9	385.0	843.1
514	MAHARASHTRA	YAVATMAL	8.6	4.6	11.0	7.7	11.9	173.6	267.1	262.8	151.5	61.9	13.2	8.9	982.8	13.2	30.6	855.0	84.0
77	NAGALAND	ZUNHEBOTO	23.7	26.8	65.7	177.2	225.7	350.3	441.8	352.2	241.8	122.5	41.6	10.7	2080.0	50.5	468.6	1386.1	174.8

#Mean

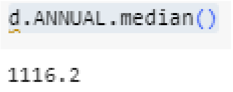
d.JAN.mean()



```
d.JAN.mean()

18.35507020280811
```

#Median

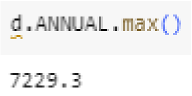
d.ANNUAL.median()



```
d.ANNUAL.median()

1116.2
```

#Maximum

d.ANNUAL.max()

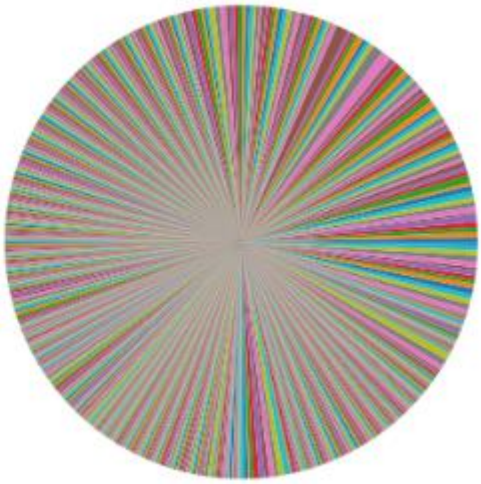


```
d.ANNUAL.max()

7229.3
```

#Pie chart

plt.pie(d.ANNUAL)

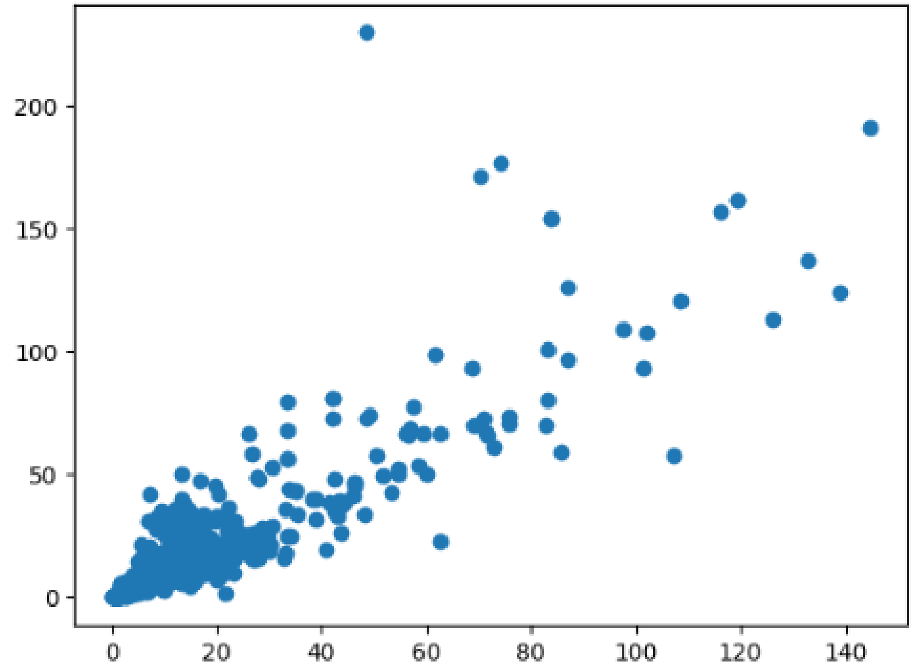
plt.show()



#Scatter Plots

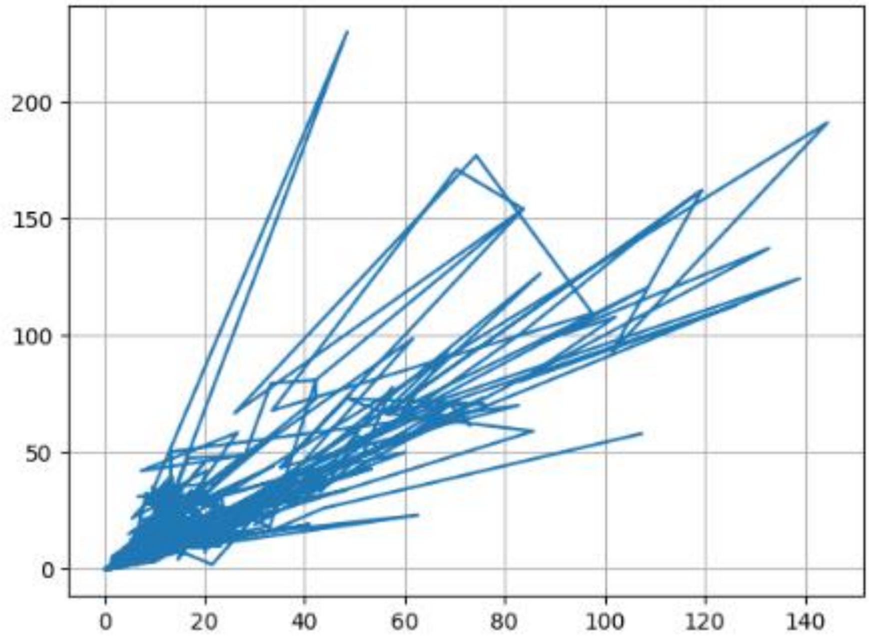
```
x=d.JAN

y=d.FEB

plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x79fca1b2c040>
```



#Line Graph

```
plt.plot(x,y)

plt.grid()
```



Result:

Thus the case study of climate change was studied and visualization was executed.

CONTENT BEYOND SYALLBUS

Ex. No. 9	**Content Beyond Syllabus**
	Random Forest

Aim:

To implement the Multiple Regression using Python.

Algorithm:

Step 1: Import the necessary library

Step 2: Select random K data points from the training set.

Step 3: Build the decision trees associated with the selected data points (Subsets).

Step 4: Choose the number N for decision trees that you want to build.

Step 5: Repeat Step 1 & 2.

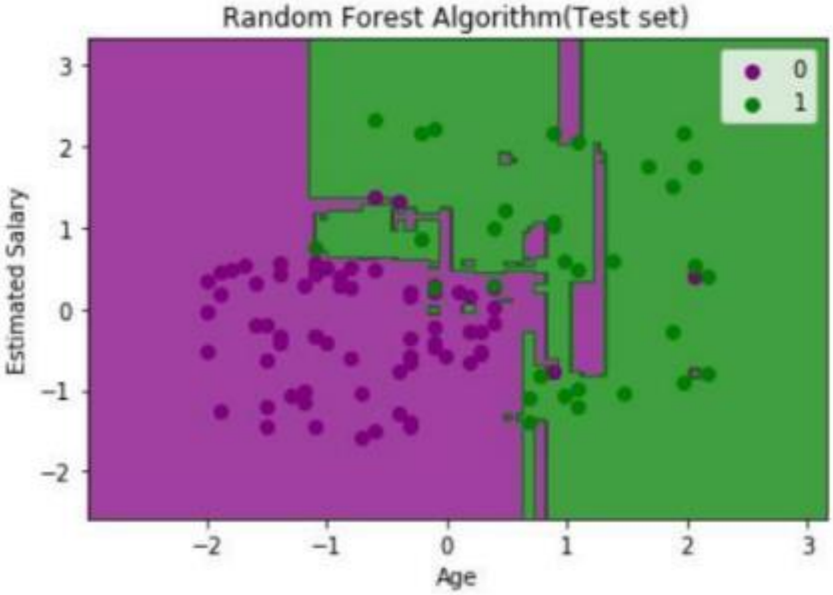
Step 6: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Source code:

```
from matplotlib.colors import ListedColormap

x_set, y_set = x_test, y_test

x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:,
0].max() + 1,step =0.01),

nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step =
0.01))

mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),

x2.ravel()]).T).reshape(x1.shape),

alpha = 0.75, cmap = ListedColormap(('purple','green' )))
```

```
mtp.xlim(x1.min(), x1.max())

mtp.ylim(x2.min(), x2.max())

for i, j in enumerate(nm.unique(y_set)):

mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],

c = ListedColormap(('purple', 'green'))(i), label = j)

mtp.title('Random Forest Algorithm(Test set)')

mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()
```

Output:



Result:

Thus, the implementation of random forest using python is verified and successfully executed