Ex. No.: 1 **Write a Program for Bar, Histogram and Pie chart**

Aim:

To write a Python program to draw the Bar, Histogram and Pie chart by using the given series and csv files.

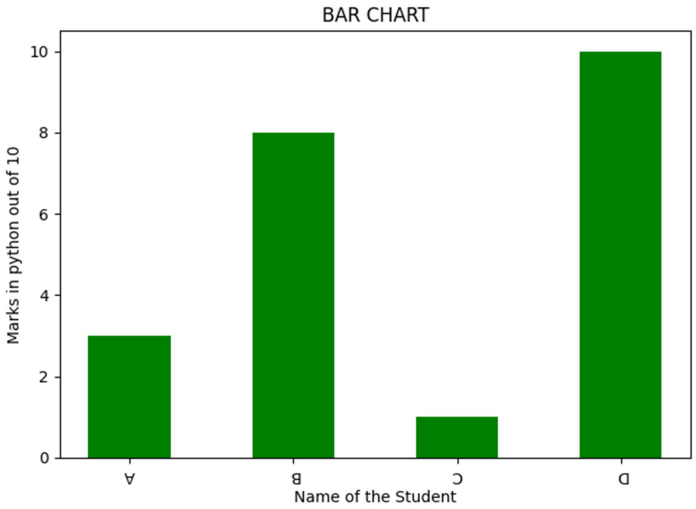
(i) Construct a Vertical Bar Chart for the following data:

Classes	A	B	C	D
No. of frequency	3	8	1	10

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y,color = "GREEN", width=0.5)
plt.xlabel('Name of the Student')
plt.ylabel('Marks in python out of 10')
plt.title('BAR CHART')
plt.xticks(rotation=180)
plt.tight_layout()
plt.show()
```

Output:



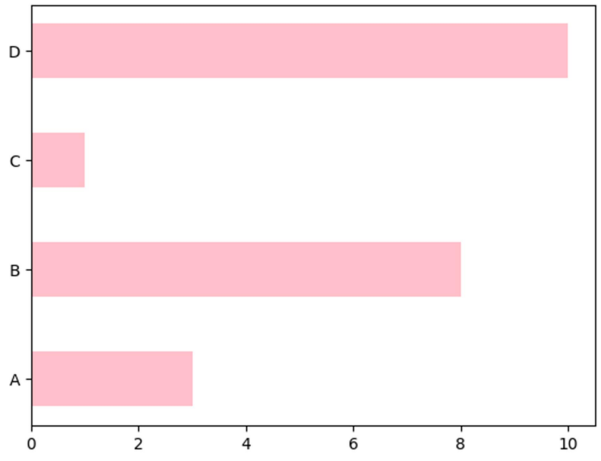
(ii) Construct a Horizontal Bar Chart for the following data:

Classes	A	B	C	D
No. of frequency	3	8	1	10

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.barh(x,y,color = "pink", height=0.5)
plt.show()
```

Output:



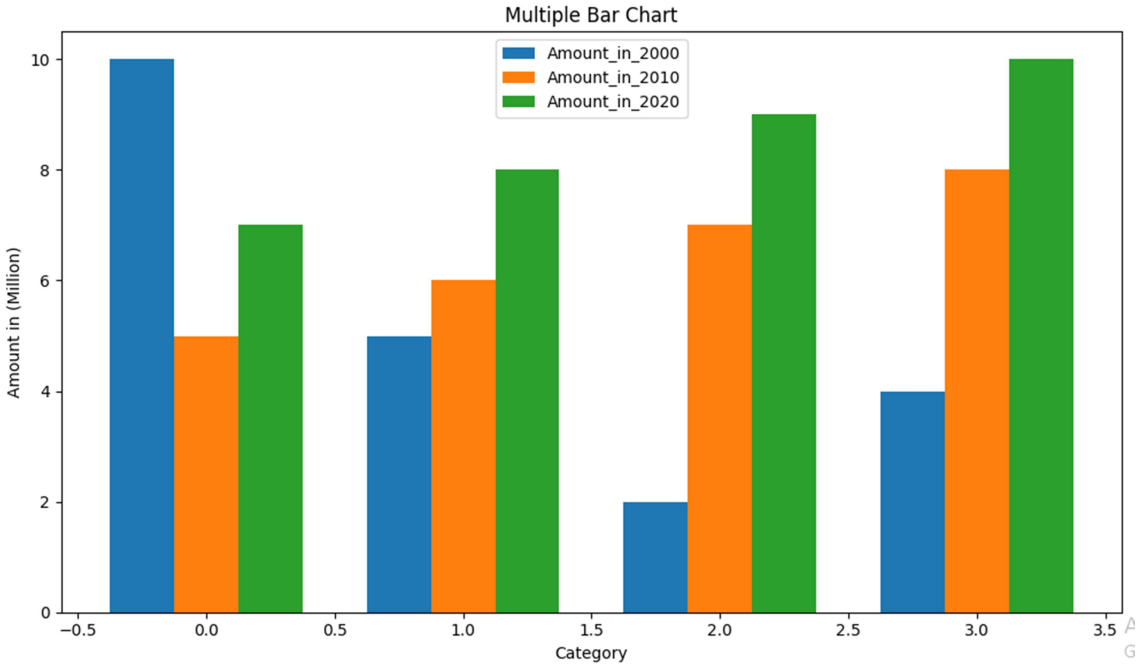
(iii) Construct a Multiple Bar Chart for the following data:

Category	Amount in (Million)		
	2000	2010	2020
A	10	5	7
B	5	6	8
C	2	7	9
D	4	8	10

Program:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
labels=['A','B','C','D']
Amount_in_2000=[10,5,2,4]
Amount_in_2010=[5,6,7,8]
Amount_in_2020=[7,8,9,10]
bar_width=0.25
x=np.arange(len(labels))
pos1=x-bar_width
pos2=x
pos3=x+bar_width
plt.figure(figsize=(10,6))
plt.bar(pos1,Amount_in_2000,width=bar_width,label='Amount_in_2000')
plt.bar(pos2,Amount_in_2010,width=bar_width,label='Amount_in_2010')
plt.bar(pos3,Amount_in_2020,width=bar_width,label='Amount_in_2020')
plt.xlabel('Category')
plt.ylabel('Amount in (Million)')
plt.title('Multiple Bar Chart')
plt.legend()
plt.tight_layout()
plt.show()
```

Output:


Multiple Bar Chart
Amount_in_2000
Amount_in_2010
Amount_in_2020
Amount in (Million)
Category
0
2
4
6
8
10
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5

(iv) Construct a Multiple Bar Chart for the following data:

S. No.	Duration	Pulse	Maxpulse	Calories
1	60	110	130	405
2	50	111	150	456
3	45	112	124	546
4	54	135	153	546
5	55	145	145	554
6	58	125	123	445
7	65	121	123	446
8	45	101	152	664
9	42	132	125	464
10	47	123	142	564
11	54	125	435	645

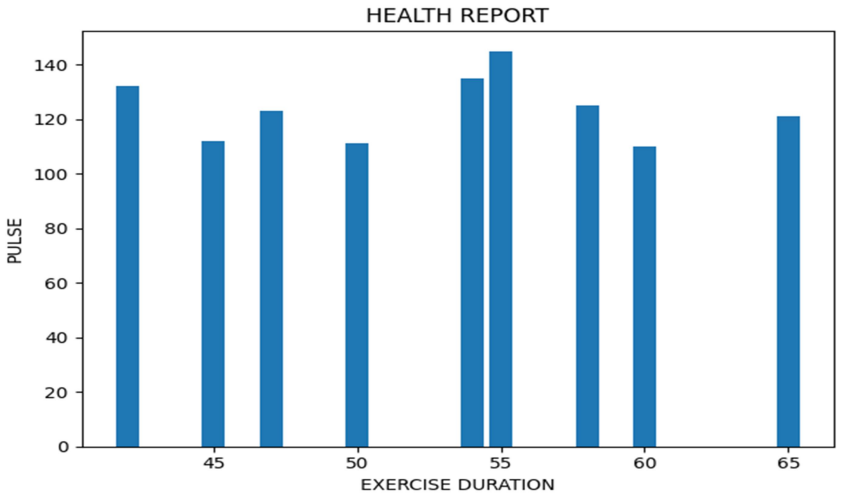
Procedure:

The following steps are converting the given data into CSV file. First ensure data in word is structured (like tables). Then select and copy table data from word and paste into Excel sheet. Next go to file menu and save as the CSV (Comma delimited) as file type.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('book.csv')
plt.bar(df['Duration'],df['Pulse'])
plt.xlabel('EXERCISE DURATION')
plt.ylabel('PULSE')
plt.title('HEALTH REPORT')
plt.tight_layout()
plt.show()
```

Output:

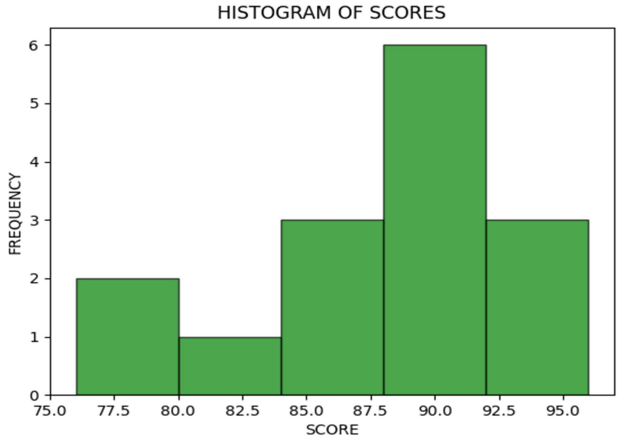


(v) Construct a Histogram Chart for the following score data: 85, 90, 78, 92, 88, 76, 95, 89, 84, 90, 87, 91, 82, 96, 88.

Program:

```
import numpy as np
import matplotlib.pyplot as plt
scores=[85,90,78,92,88,76,95,89,84,90,87,91,82,96,88]
plt.hist(scores, bins=5, alpha=0.7, color='green', edgecolor='black')
plt.xlabel('SCORE')
plt.ylabel('FREQUENCY')
plt.title('HISTOGRAM OF SCORES')
plt.show()
```

Output:

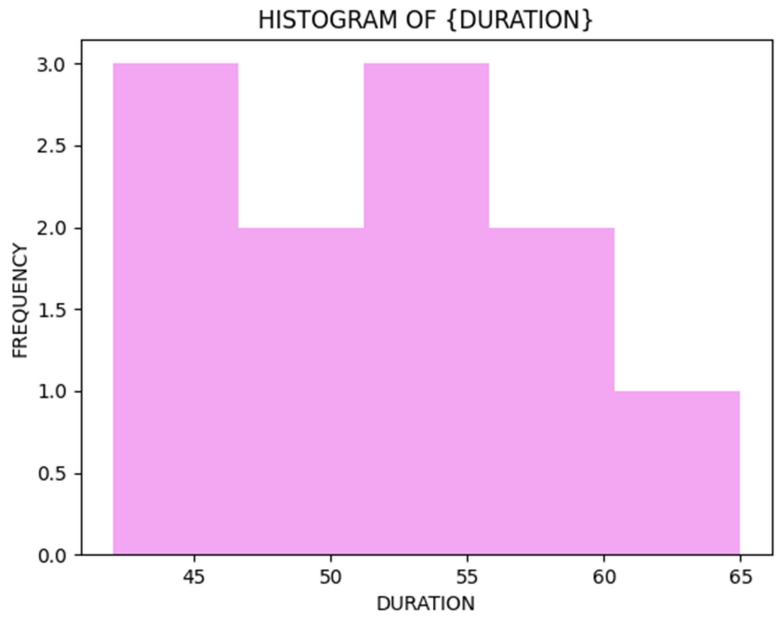


(vi) Construct a Histogram Chart for the above problem (iv):

Program:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('book.csv')
column_name='Duration'
data=df['Duration']
plt.hist(data,bins=5, alpha=0.7, color='violet')
plt.xlabel('DURATION')
plt.ylabel('FREQUENCY')
plt.title('HISTOGRAM OF {DURATION}')
plt.show()
```

Output:



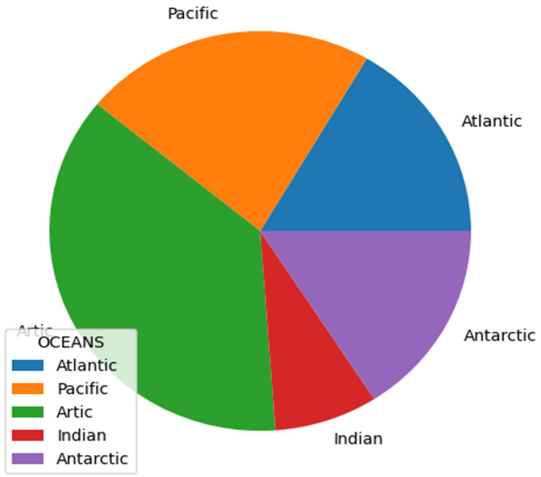
(vii) Construct a Pie Chart for the following data:

Oceans	Atlantic	Pacific	Artic	Indian	Antarctic
Thousand Kilo Meters	25	34	56	12	24

Program:

```
import numpy as np
import matplotlib.pyplot as plt
y=np.array([25,34,56,12,24])
mylabels=["Atlantic Ocean", "Pacific","Artic","Indian","Antarctic"]
plt.pie(y,labels=mylabels)
plt.legend(title="OCEANS")
plt.tight_layout()
plt.show()
```

Output:



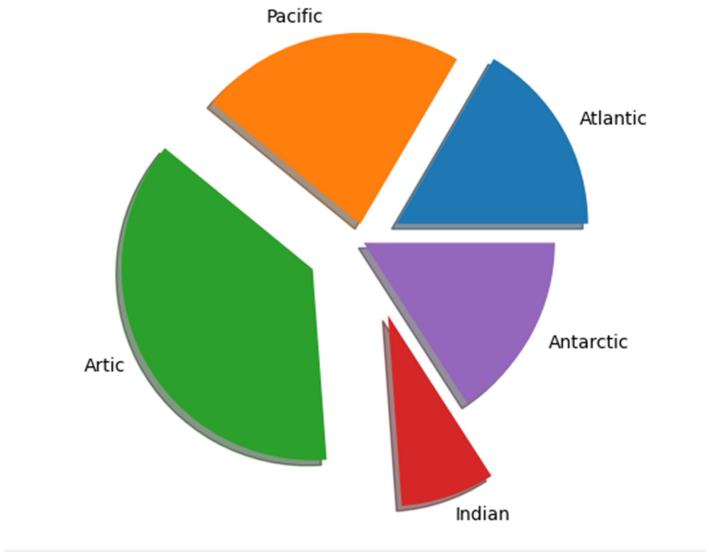
(viii) Construct a Pie Chart for the following data:

Oceans	Atlantic	Pacific	Artic	Indian	Antarctic
Thousand Kilo Meters	25	34	56	12	24

Program:

```
import numpy as np
import matplotlib.pyplot as plt
y=np.array([25,34,56,12,24])
mylabels=["Atlantic Ocean", "Pacific", "Artic", "Indian", "Antarctic"]
myexplode=[0.2,0.1,0.3,0.4,0]
plt.pie(y,labels=mylabels,explode=myexplode, shadow=True)
plt.show()
```

Output:



Ex. No.: 2 Write a Program for Mean, Median, Mode, Percentile, Standard Deviation and Variance

Aim:

To write a Python program to calculate Mean, Median, Mode, Percentile, Standard Deviation and Variance by using the given series and csv files.

(i) Calculate the Mean, Median, Mode, Percentile, Standard Deviation and Variance for the following data: 99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86.

Program:

```
import numpy as np
import statistics as stat
speed=[99,86,87,88,111,86,103,87,94,78,77,85,86]
mean=np.mean(speed)
median=np.median(speed)
mode=stat.mode(speed)
sd=np.std(speed)
variance=np.var(speed)
p1=np.percentile(speed,25)
p2=np.percentile(speed,50)
p3=np.percentile(speed,75)
p4=np.percentile(speed,100)
print("MEAN:", mean)
print("MEDIAN:", median)
print("MODE:", mode)
print("STANDARD DEVIATION:", sd)
print("VARIANCE:", variance)
print("First Quartile or 25% Percentile:", p1)
print("Second Quartile or Median or 50% Percentile:", p2)
print("Third Quartile or 75% Percentile:", p3)
print("Fourth Quartile or 100% Percentile:", p4)
```

Output:

MEAN: 89.76923076923077

MEDIAN: 87.0

MODE: 86

STANDARD DEVIATION: 9.258292301032677

VARIANCE: 85.71597633136093

First Quartile or 25% Percentile: 86.0

Second Quartile or Median or 50% Percentile: 87.0

Third Quartile or 75% Percentile: 94.0

Fourth Quartile or 100% Percentile: 111.0

(ii) Calculate the Mean, Median, Mode, Percentile, Standard Deviation and Variance for the above problem 1(iv):

Program:

```
import numpy as np
import pandas as pd
df = pd.read_csv('book.csv')
mean=df.mean(numeric_only= True)
median=df.median(numeric_only= True)
mode=df.mode(numeric_only = True)
sd=df.std(numeric_only= True)
mode_value=df['Duration'].mode()
p1=df['Duration'].quantile(0.25)
percentile=df['Pulse'].quantile([0.25,0.5,0.75])
print("MEAN:\n", mean)
print("MEDIAN:\n", median)
print("Mode:", mode)
print("STANDARD DEVIATION:\n", sd)
print("Mode (Duration):", mode_value)
print("First Quartile(Duration):", p1)
print("Percentile (Pulse):",percentile)
```

Output:

MEAN:

```
S. No.          6.000000
Duration       52.272727
Pulse         121.818182
Maxpulse      163.818182
Calories      521.363636
dtype: float64
```

MEDIAN:

```
S. No.          6.0
Duration       54.0
Pulse         123.0
Maxpulse      142.0
Calories      546.0
dtype: float64
```

Mode:

	S. No.	Duration	Pulse	Maxpulse	Calories
0	1	45.0	125.0	123.0	546.0
1	2	54.0	NaN	NaN	NaN
2	3	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN
5	6	NaN	NaN	NaN	NaN
6	7	NaN	NaN	NaN	NaN
7	8	NaN	NaN	NaN	NaN
8	9	NaN	NaN	NaN	NaN
9	10	NaN	NaN	NaN	NaN
10	11	NaN	NaN	NaN	NaN

STANDARD DEVIATION:

```
S. No.          3.316625
Duration        7.156688
Pulse          12.742199
Maxpulse       90.766534
Calories       85.054421
dtype: float64
```

Mode (Duration):

```
0    45
1    54
Name: Duration, dtype: int64
```

First Quartile(Duration): 46.0

Percentile (Pulse):

```
0.25    111.5
0.50    123.0
0.75    128.5
Name: Pulse, dtype: float64
```

Ex. No.: 3 Write a Program for Descriptive Statistics

Aim:

To write a Python program to calculate the Descriptive Statistics by using the given series and csv files.

(i) Calculate the Descriptive Statistics for the following data: 99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86.

Program:

```
import pandas as pd
speed=pd.Series([99,86,87,88,111,86,103,87,94,78,77,85,86])
DS=speed.describe()
print("Descriptive Statistics:\n", DS)
```

Output:

```
Descriptive Statistics:
count     13.000000
mean      89.769231
std        9.636336
min       77.000000
25%       86.000000
50%       87.000000
75%       94.000000
max      111.000000
dtype: float64
```

(ii) Calculate the Descriptive Statistics for the above problem 1(iv):

Program:

```
import pandas as pd
df=pd.read_csv("Book.csv", header=0, sep=",")
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
print(df.describe())
```

Output:

	S. No.	Duration	Pulse	Maxpulse	Calories
count	11.000000	11.000000	11.000000	11.000000	11.000000
mean	6.000000	52.272727	121.818182	163.818182	521.363636
std	3.316625	7.156688	12.742199	90.766534	85.054421
min	1.000000	42.000000	101.000000	123.000000	405.000000
25%	3.500000	46.000000	111.500000	124.500000	451.000000
50%	6.000000	54.000000	123.000000	142.000000	546.000000
75%	8.500000	56.500000	128.500000	151.000000	559.000000
max	11.000000	65.000000	145.000000	435.000000	664.000000

Ex. No.: 4 Write a Program for Correlation Coefficient

Aim:

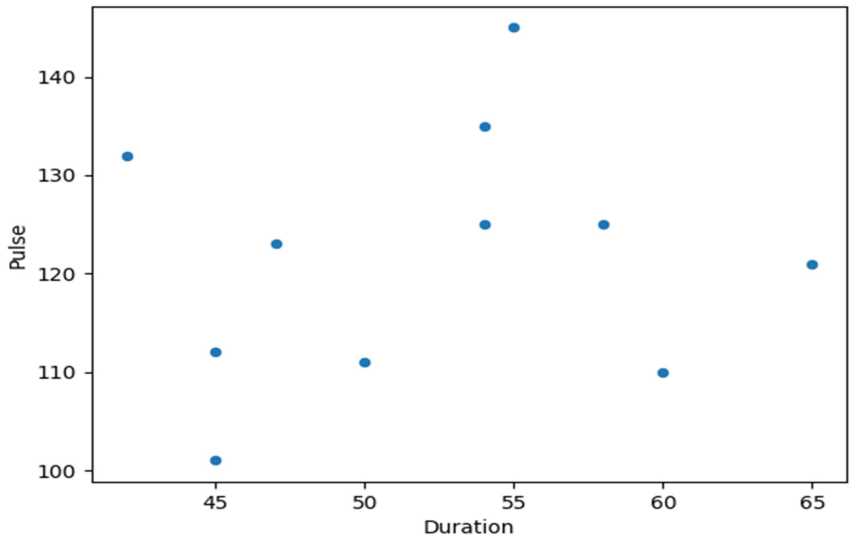
To write a Python program to calculate the Correlation coefficient by using the given series and csv files.

(i) Calculate the Correlation Coefficient between Duration and Pulse for the above problem 1(iv):

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("Book.csv", header=0, sep=",")
df.plot(x='Duration',y='Pulse',kind='scatter')
plt.show()
```

Output:



Result:

Here, the Correlation coefficient is zero. Hence there is no linear relationship between the two variables of Duration and Pulse.

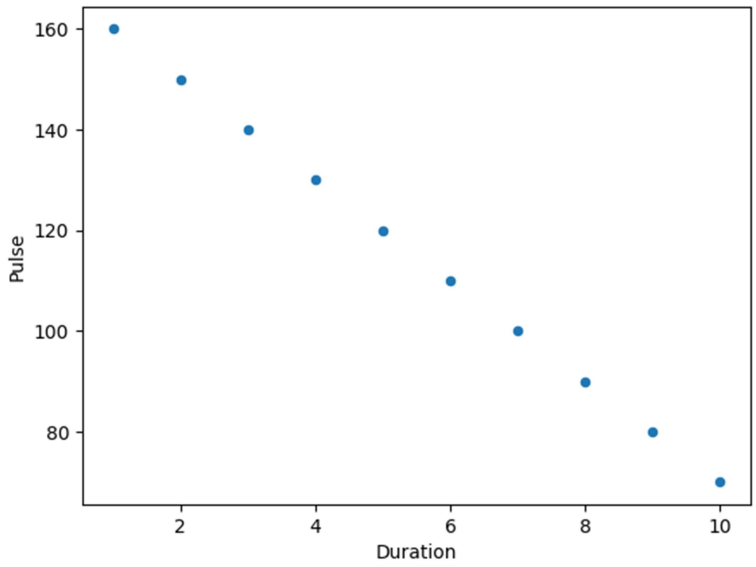
(ii) Calculate the Correlation Coefficient for the following data:

Duration	10	9	8	7	6	5	4	3	2	1
Pulse	70	80	90	100	110	120	130	140	150	160

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
Correlation={'Duration':[10,9,8,7,6,5,4,3,2,1],'Pulse':[70,80,90,100,110,120,
130,140,150,160]}
Correlation=pd.DataFrame(data=Correlation)
Correlation.plot(x='Duration',y='Pulse',kind='scatter')
plt.show()
```

Output:



Result:

Hence the value of the Correlation coefficient is -1. Therefore, we conclude that if the duration is longer hours, we tend to have lower pulse.

(iii) Calculate the Correlation Coefficient for the above problem 1(iv):

Program:

```
import pandas as pd
df=pd.read_csv("Book.csv", header=0, sep=",")
Correlation=round(df.corr(),2)
print(Correlation)
```

Output:

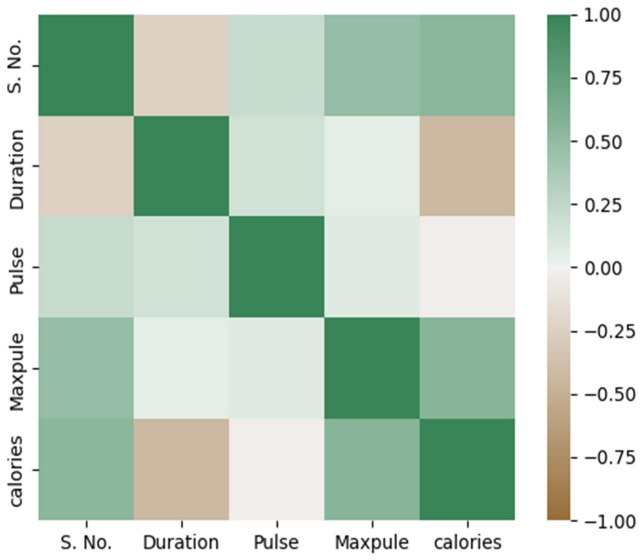
	S. No.	Duration	Pulse	Maxpulse	Calories
S. No.	1.00	-0.25	0.22	0.49	0.54
Duration	-0.25	1.00	0.16	0.05	-0.44
Pulse	0.22	0.16	1.00	0.08	-0.03
Maxpulse	0.49	0.05	0.08	1.00	0.55
Calories	0.54	-0.44	-0.03	0.55	1.00

(iii) Calculate the Correlation Coefficient and visualizes the data using heatmap for the above problem 1(iv):

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
importseaborn as sns
df=pd.read_csv("Book.csv", header=0, sep=",")
correlation=df.corr()
axis_corr=sns.heatmap(correlation, vmin=-1, vmax=1, center=0,
cmap=sns.diverging_palette(50,500,n=500), square=True)
plt.show()
```

Output:



Result:

(i) The closer the correlation coefficient is to 1, the greener the squares get.

(ii) The closer the correlation coefficient is to -1, the browner the squares get.

Ex. No.: 5 Write a Program for Simple Linear Regression

Aim:

To write a Python program to calculate the simple linear regression by using the given series and csv files.

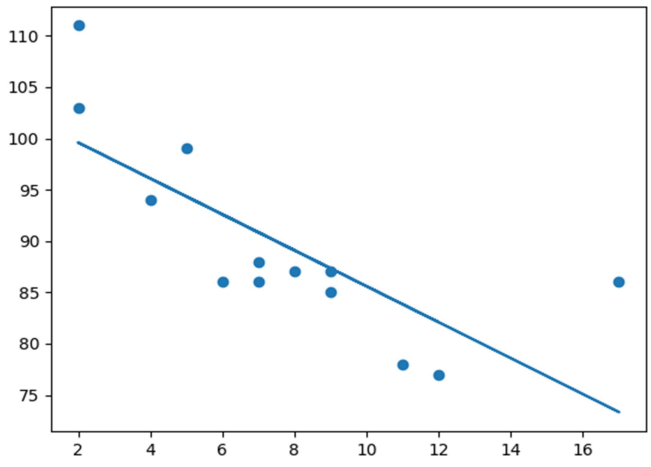
(i) Fit the linear regression for the following data:

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

Program:

```
import matplotlib.pyplot as plt
from scipy import stats
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)
defmyfunc(x):
return slope * x + intercept
mymodel = list(map(myfunc, x))
plt.plot(x, mymodel)
plt.scatter(x, y)
plt.show()
```

Output:



.

(ii) Calculate the linear regression for the above problem (1(iv)) and also find the value of Duration if Calorie is: 56, 45, 65:

Program:

```
import pandas as pd
import statsmodels.formula.api as smf
df=pd.read_csv("Book.csv", header=0, sep=",")
model=smf.ols('Duration ~ Calories', data=df)
results = model.fit()
print(results.summary())
```

Output:

```
Warning (from warnings module):
  File "C:\Users\admin\AppData\Roaming\Python\Python38\site-packages\scipy\stats\_stats_py.py", line 1736
    warnings.warn("kurtosistest only valid for n>=20 ... continuing "
UserWarning: kurtosistest only valid for n>=20 ... continuing anyway, n=11
                            OLS Regression Results
==============================================================================
Dep. Variable:               Duration   R-squared:                       0.190
Model:                            OLS   Adj. R-squared:                  0.100
Method:                 Least Squares   F-statistic:                     2.113
Date:                Tue, 05 Aug 2025   Prob (F-statistic):              0.180
Time:                        21:23:01   Log-Likelihood:                -35.573
No. Observations:                  11   AIC:                             75.15
Df Residuals:                       9   BIC:                             75.94
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     71.4004     13.318      5.361      0.000      41.273     101.528
calories      -0.0367      0.025     -1.454      0.180      -0.094       0.020
==============================================================================
Omnibus:                        0.458   Durbin-Watson:                   1.513
Prob(Omnibus):                  0.795   Jarque-Bera (JB):                0.408
Skew:                          -0.368   Prob(JB):                        0.816
Kurtosis:                       2.409   Cond. No.                     3.43e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.43e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
>>>
```


Result:

The linear regression function can be rewritten mathematically as: Duration =-0.0367 * Calories + 71.4004 [Y=aX+b]. Here P-value is 0.180. Which is greater than 0.05 and also R-Squared value is 0.190. Hence we conclude that there is a no significant relationship between the Duration and calories. Hence the regression line does not fit for the data.

Program:

```
def Duration(Calories):
    return (-0.0367 * Calories + 71.4004)
print(Duration(56))
print(Duration(45))
print(Duration(65))
```

Output:

```
69.3452
69.7489
69.01490000000001
```

Result:

(i) If Calorie is 56 then Duration becomes 69.3452.
(ii) If Calorie is 45 then Duration becomes 69.7489.
(iii) If Calorie is 65 then Duration becomes 69.01490000000001.

Ex. No.: 6 Write a program for Polynomial Regression Algorithm

Aim:

To write a Python program to calculate the Polynomial regression by using the given series and csv files.

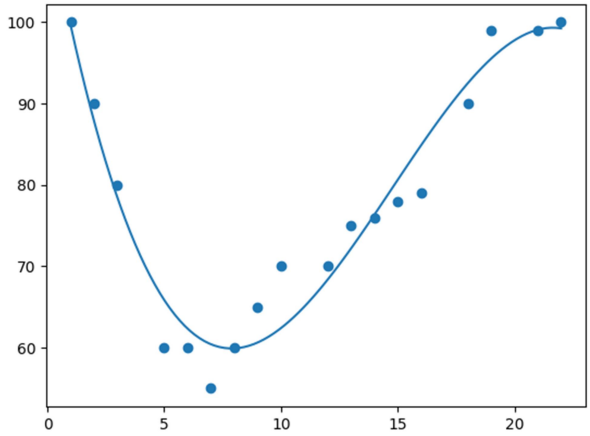
(i) Fit the polynomial regression for the following data and also find the value of R-Squared and speed if a car is passing at 17th hour:

Car Passing hour (X)	1	2	3	5	6	7	8	9	10	12	13	14	15	16	18	19	21	22
Speed (Y)	100	90	80	60	60	55	60	65	70	70	75	76	78	79	90	99	99	100

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]
mymodel = np.poly1d(np.polyfit(x, y, 3))
myline = np.linspace(1, 22, 100)
plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
print(r2_score(y, mymodel(x)))
speed = mymodel(17)
print(speed)
```

Output:



0.9432150416451027
88.87331269697984

Result:

Here R-Squared value is 0.9432150416451027. The result 0.94 shows that there is a very good relationship and we can use polynomial regression in future predictions. The predicted speed of a car passing at 17th hour is 88.87331269697984.

(ii) Calculate the value of R-Squared value for the following data:

X	1	2	3	5	6	7	8	9	10	12	13	14	15	16	18	19	21	22
Y	100	90	80	60	60	55	60	65	70	70	75	76	78	79	90	99	99	100

Program:

```
import numpy as np
from sklearn.metrics import r2_score
x = [89, 43, 36, 36, 95, 10, 66, 34, 38, 20, 26, 29, 48, 64, 6, 5, 36, 66, 72, 40]
y = [21, 46, 3, 35, 67, 95, 53, 72, 58, 10, 26, 34, 90, 33, 38, 20, 56, 2, 47, 15]
mymodel = np.poly1d(np.polyfit(x, y, 3))
print(r2_score(y, mymodel(x)))
```

Output:

0.009952707566680652

Result:

Here R-Squared value = 0.009952707566680652. The result 0.00995 indicates a very bad relationship and this data set is not suitable for polynomial regression.

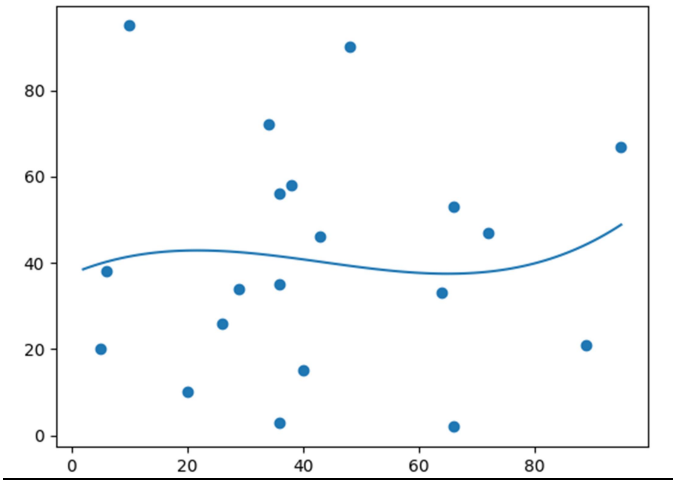
(iii) Fit the polynomial regression for the following data:

X	1	2	3	5	6	7	8	9	10	12	13	14	15	16	18	19	21	22
Y	100	90	80	60	60	55	60	65	70	70	75	76	78	79	90	99	99	100

Program:

```
import numpy as np
import matplotlib.pyplot as plt
x = [89, 43, 36, 36, 95, 10, 66, 34, 38, 20, 26, 29, 48, 64, 6, 5, 36, 66, 72,
40]
y = [21, 46, 3, 35, 67, 95, 53, 72, 58, 10, 26, 34, 90, 33, 38, 20, 56, 2, 47,
15]
mymodel = np.poly1d(np.polyfit(x, y, 3))
myline = np.linspace(2, 95, 100)
plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```

Output:



Result:

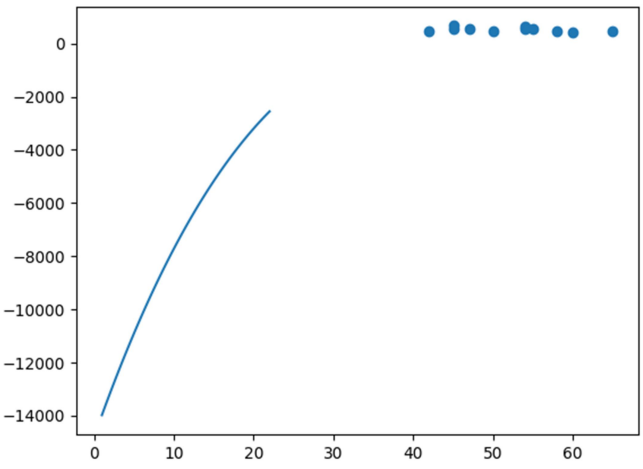
These values are very bad fit for polynomial regression and would not be the best method to predict future values.

(iv) Fit the polynomial regression for the above problem (1(iv)) and also find the value of R-Squared value:

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
df=pd.read_csv("Book.csv", header=0, sep=",")
x=df["Duration"]
y=df["Calories"]
mymodel = np.poly1d(np.polyfit(x, y, 3))
myline = np.linspace(1, 22, 100)
plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
print(r2_score(y, mymodel(x)))
```

Output:



0.40379738147486666

Result:

Here, R-Squared value is 0.40379738147486666. The values of duration and calories are not good to fit for polynomial regression and would not be the best method to predict future values.

Ex. No.: 7 Write a program for Multiple Regression

Aim:

To write a Python program to calculate the Multiple Regression by using the given csv files.

(i) Find the regression coefficient for the above problem (i(iv)) and also find the value of Duration if Pulse is 110 and Maxpulse is 130:

Program:

```
import pandas as pd
from sklearn import linear_model
df=pd.read_csv("Book.csv", header=0, sep=",")
x=df[['Pulse','Maxpulse']]
y=df['Duration']
regr=linear_model.LinearRegression()
regr.fit(x,y)
predicted_Duration=regr.predict([[110,130]])
print(predicted_Duration)
print(regr.coef_)
```

Output:

```
51.14841052
0.08745016 0.00268539
```

Result:

If Pulse is 110 and Maxpulse is 130 then the predicted Duration is 51.14841052.

The regression coefficient is 0.08745016 0.00268539.

Ex. No.: 8 Write a program for Logistics Regression

Aim:

To write a Python program to calculate the Logistics Regression by using the given data.

(i). Construct the Logistics Regression model for the following data and also predict the match outcome if run scored and overs played are 180 and 22 respectively.

Runs Scored	Overs Played	Match Outcome (1 = Win, 0 = Loss)
100	15	0
150	20	1
80	10	0
200	25	1
120	18	1

Program:

```
import numpy as np
from sklearn.linear_model import LogisticRegression
X = np.array([[100, 15], [150, 20], [80, 10], [200, 25], [120, 18]])
y = np.array([0, 1, 0, 1, 1])
model = LogisticRegression()
model.fit(X, y)
new_data = np.array([[180, 22]])
prediction = model.predict(new_data)
prediction_proba = model.predict_proba(new_data)
accuracy=model.score(X,y)
print("Accuracy",accuracy)
print(f"Runs Scored: {new_data[0, 0]}, Overs Played: {new_data[0, 1]}")
print(f"Prediction: {'Win' if prediction[0] == 1 else 'Loss'}")
print(f"Prediction Probability: {prediction_proba[0]}")
```

Output:

```
Accuracy 1.0
Runs Scored: 180, Overs Played: 22
Prediction: Win
Prediction Probability: [1.48259183e-12 1.00000000e+00]
```

(ii). Construct the Logistics Regression model for the following data and also predict if tumor is cancerous where the size is 3.46mms:

X	3.78	2.44	2.09	0.14	1.72	1.65	4.92	4.37	4.96	4.52	3.69	5.88
y	0	0	0	0	0	0	1	1	1	1	1	1

Here X represents the size of a tumor in millimeters.
Y represents whether or not the tumor is cancerous (0 for “No”, 1 for “Yes”).

Aim:

To write a Python program to calculate the Logistics Regression by using the given data.

Program:

```
import numpy
from sklearn import linear_model
X = numpy.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52,
3.69, 5.88]).reshape(-1,1)
y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
logr = linear_model.LogisticRegression()
logr.fit(X,y)
predicted = logr.predict(numpy.array([3.46]).reshape(-1,1))
accuracy=logr.score(X,y)
print("Accuracy",accuracy)
print(predicted)
```

Output:

```
Accuracy 0.9166666666666666
[0]
```

Result:

We have predicted that a tumor with a size of 3.46mm will not be cancerous.

Ex. No.: 9 **Write a program for Hierarchical Clustering**

Aim:

To write a Python program to calculate the Hierarchical Clustering by using the given series of files.

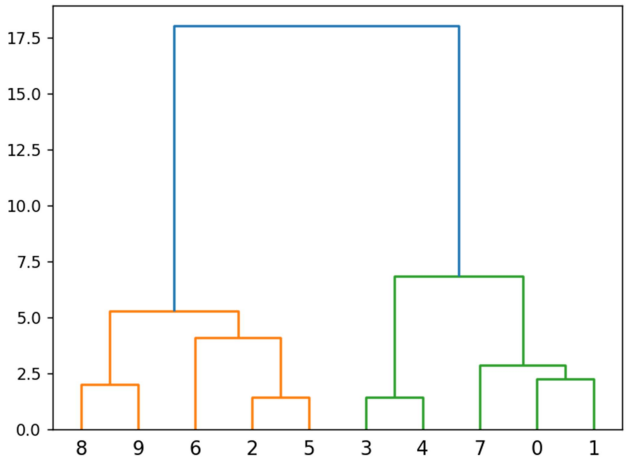
(i) Construct the Dendrogram graph in the Hierarchical Clustering for the following data:

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
data = list(zip(x, y))
linkage_data = linkage(data, method='ward', metric='euclidean')
dendrogram(linkage_data)
plt.show()
```

Output:



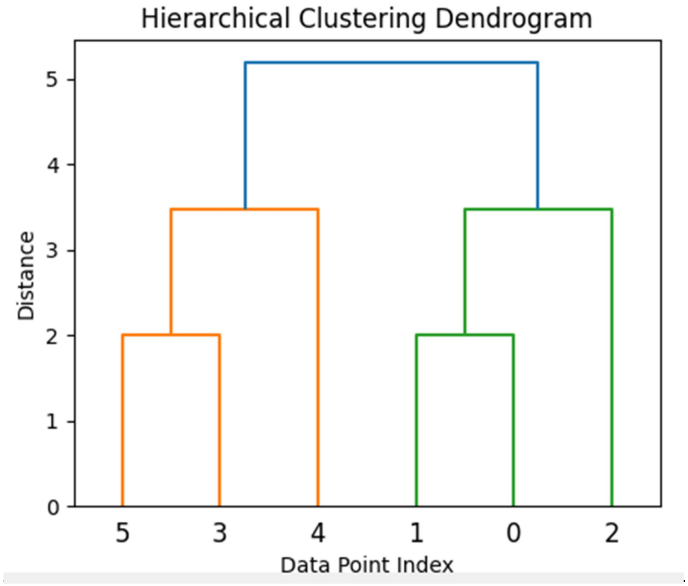
(ii) Construct the Dendrogram graph in the Hierarchical Clustering for the following data:

x	1	1	1	4	4	4
y	2	4	0	2	4	0

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
X = np.array([[1, 2], [1, 4], [1, 0], [4, 2], [4, 4], [4, 0]])
linked = linkage(X, method='ward')
plt.figure(figsize=(10, 7))
dendrogram(linked,orientation='top',labels=np.arange(len(X)),distance_sort='
descending',show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data Point Index')
plt.ylabel('Distance')
plt.show()
```

Output:



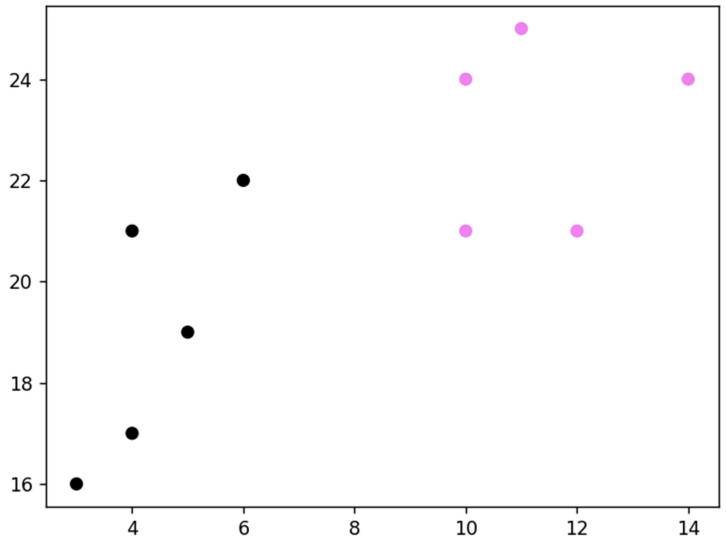
(ii) Fit the Hierarchical Clustering for the following data:

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21

Program:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
data = list(zip(x, y))
hierarchical_cluster = AgglomerativeClustering(n_clusters=2,
metric='euclidean', linkage='ward')
labels = hierarchical_cluster.fit_predict(data)
plt.scatter(x, y, c=['black' if label == 0 else 'violet' for label in labels])
plt.show()
```

Output:



Ex. No.: 10 **Write a program for K-means Clustering**

Aim:

To write a Python program to calculate the K-means clustering by using the given series of files.

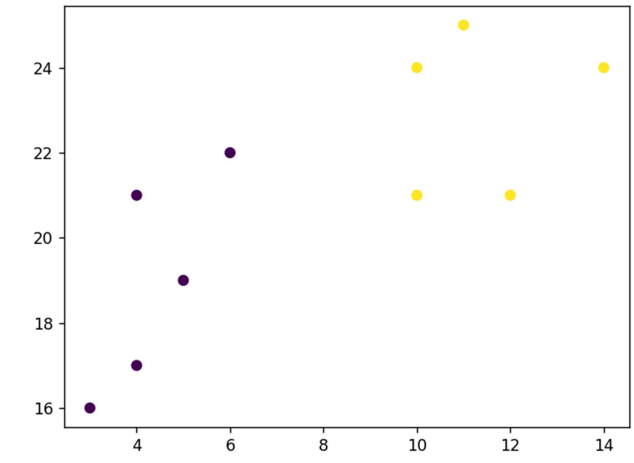
(i) Fit the K-means Clustering for the following data:

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21

Program:

```
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

x = [4, 5, 10, 4, 3, 11, 14, 6, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

data = list(zip(x, y))

kmeans = KMeans(n_clusters=2)

kmeans.fit(data)

plt.scatter(x, y, c=kmeans.labels_)

plt.show()
```

Output:



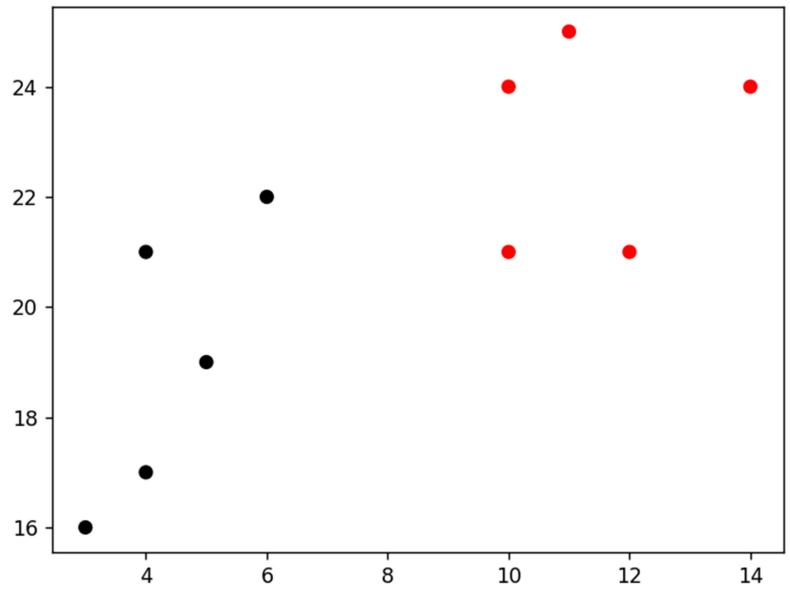
(ii) Fit the K-means Clustering for the following data:

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21

Program:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
data = list(zip(x, y))
kmeans = KMeans(n_clusters=2)
kmeans.fit(data)
plt.scatter(x, y, c=['red' if label==0 else 'black' for label in kmeans.labels_])
plt.show()
```

Output:



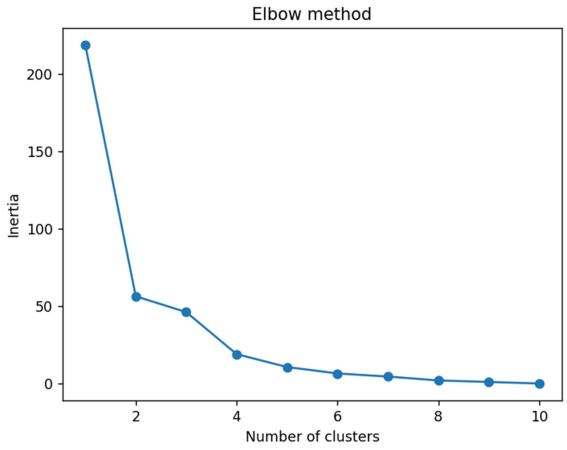
(iii) Fit the K-means Clustering by the Elbow method for the following data:

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21

Program:

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
x = [4, 5, 10, 4, 3, 11, 14 , 6, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
data = list(zip(x, y))
inertias = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)
plt.plot(range(1,11), inertias, marker='o')
plt.title('Elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()
```

Output:



Result:

The elbow method shows that 2 is a good value for K, so we retrain and visualize the result.

Ex. No.: 11 Write a program for KNN Algorithm

Aim:

To write a Python program to calculate the KNN Algorithm by using the given series of files.

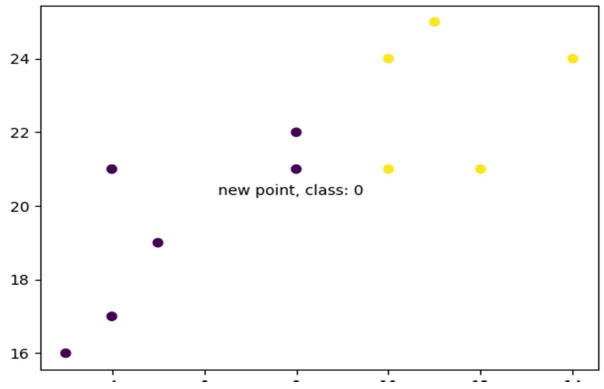
(i) Fit the KNN Algorithm for the following data if neighbors = 1 and also find the classes if x = 8 and y = 21.

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21
classes	0	0	1	0	0	1	1	0	1	1

Program:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(data, classes)
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

Output:



Result:

If x = 8 and y = 21 then classes belongs to zero.

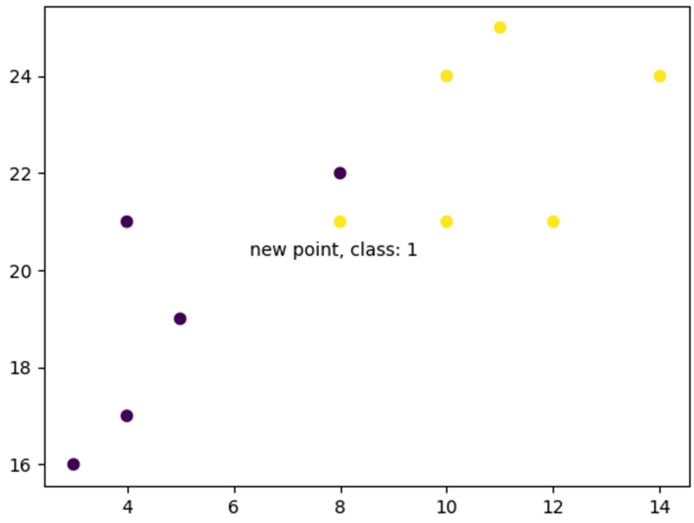
(ii) Fit the KNN Algorithm for the following data if neighbors = 5 and also find the classes if x = 8 and y = 21.

x	4	5	10	4	3	11	14	6	10	12
y	21	19	24	17	16	25	24	22	21	21
classes	0	0	1	0	0	1	1	0	1	1

Program:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]
data = list(zip(x, y))
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(data, classes)
new_x = 8
new_y = 21
new_point = [(new_x, new_y)]
prediction = knn.predict(new_point)
plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.text(x=new_x-1.7, y=new_y-0.7, s=f"new point, class: {prediction[0]}")
plt.show()
```

Output:



Result:

If x = 8 and y = 21 then classes belongs to one.

Ex. No.: 12 Write a program for Naive Baye's Algorithm

Aim:

To write a Python program to calculate the Naive Baye's Algorithm by using the given series of files.

(i) Construct the Naive Baye's Algorithm for the following data:

What is your age? (Age)	Which roads in Thoothukudi are the most dangerous for bike riders? Why? (Zone)		Are hospitals easily accessible from accident-prone highways? (Hospital)	
16	Bridges	1	YES	1
18	Thoothukudi - Madurai	2	YES	1
19	Thoothukudi - Madurai	2	YES	1
20	Bridges	1	YES	1
21	Tiruchendur - Thoothukudi	3	NO	0
22	Thoothukudi - Ramanathapuram	4	YES	1
23	Bridges	1	NO	0
24	Thoothukudi - Tirunelveli	5	YES	1
25	Bridges	1	YES	1
26	Thoothukudi - Tirunelveli	5	NO	0
27	Bridges	1	YES	1
28	Bridges	1	NO	0
29	Bridges	1	NO	0
30	Bridges	1	YES	1
31	Thoothukudi - Ramanathapuram	4	YES	1
32	Bridges	1	YES	1
33	Thoothukudi - Tirunelveli	5	YES	1
34	Bridges	1	YES	1
35	Bridges	1	YES	1
36	Bridges	1	NO	0
38	Thoothukudi - Tirunelveli	5	YES	1
39	Thoothukudi - Madurai	2	YES	1
42	Thoothukudi - Tirunelveli	5	YES	1
43	Bridges	1	YES	1
44	Tiruchendur - Thoothukudi	3	YES	1

Program:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score

data = {

    'Age':
[16,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,38,39,42,43,44],

    'Zone': [1,2,2,1,3,4,1,5,1,5,1,1,1,1,4,1,5,1,1,1,5,2,5,1,3],

    'Hospital': [1,1,1,1,0,1,0,1,1,0,1,0,0,1,1,1,1,1,1,1,0,1,1,1,1,1] }

df = pd.DataFrame(data)

X = df[['Age', 'Zone']]

y = df['Hospital']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")

new_person = pd.DataFrame({'Age': [17], 'Zone': [2]})

prediction = model.predict(new_person)

print(f"\nPrediction for new person (Age: 17, Zone: 2): {'Hospital' if prediction[0] ==
1 else 'No Hospital'}")
```

Output:

Accuracy: 0.80

Prediction for new person (Age: 17, Zone: 2): Hospital