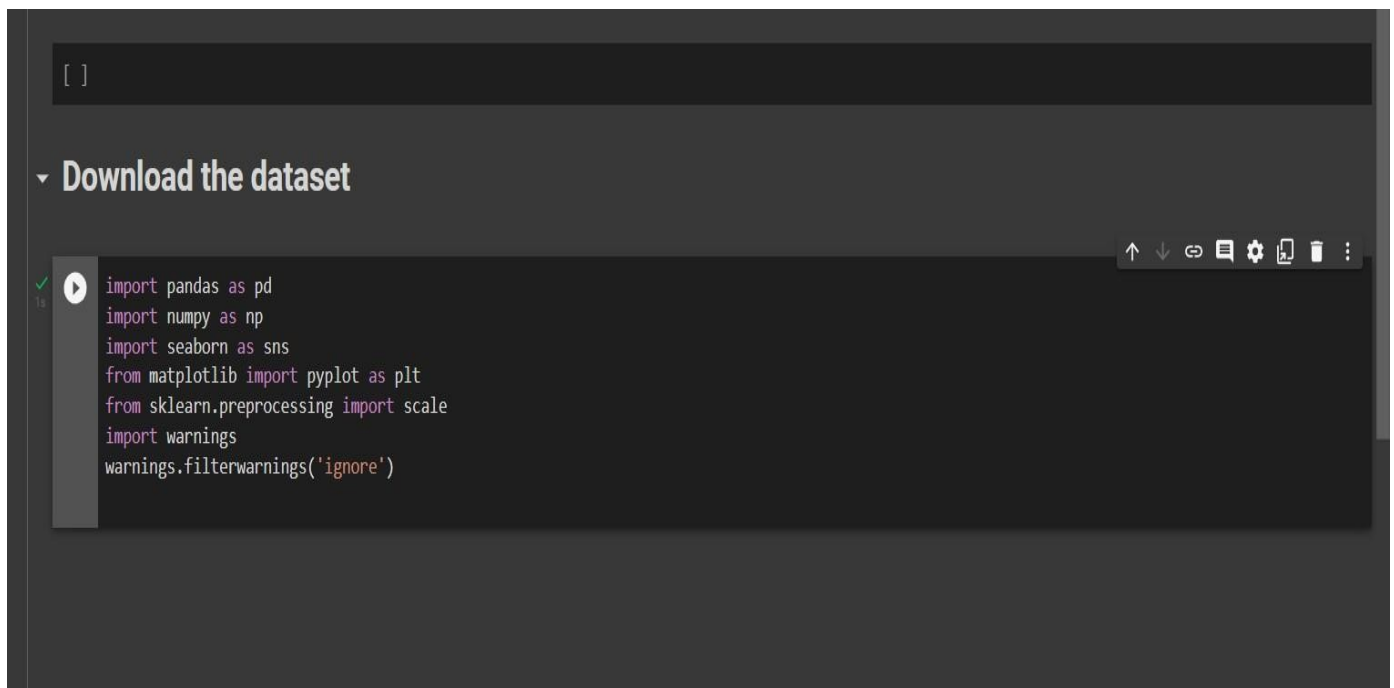


A

SSIGNMENT-04

Assignment Date	2 November 2022
Student Name	S.V.Mirudhula
Student Roll Number	113219071020
Maximum Marks	2 Marks

Problem Statement: Customer Segmentation Analysis



The screenshot shows a Jupyter Notebook interface. At the top, there is a text input field containing "[]". Below it, a section titled "Download the dataset" is visible. The main part of the image shows a code cell with the following Python code:

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.preprocessing import scale
import warnings
warnings.filterwarnings('ignore')
```

On the left side of the code cell, there is a green checkmark and a play button icon. On the right side, there is a toolbar with icons for undo, redo, copy, paste, settings, and other notebook functions.

▾ load the dataset into the tool

```
[ ] data=pd.read_csv("drive/MyDrive/CONTENT/Mall_Customers.csv")
data.head()
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
[ ] data.shape

(200, 5)
```

```
[ ] data.size
```

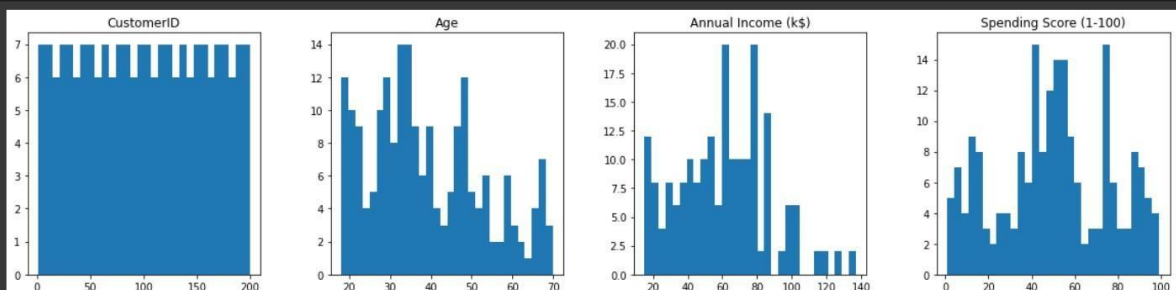
1000

▶ data.info()

```
[ ] <class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   CustomerID            200 non-null   int64
1   Gender                 200 non-null   object
2   Age                    200 non-null   int64
3   Annual Income (k$)     200 non-null   int64
4   Spending Score (1-100) 200 non-null   int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

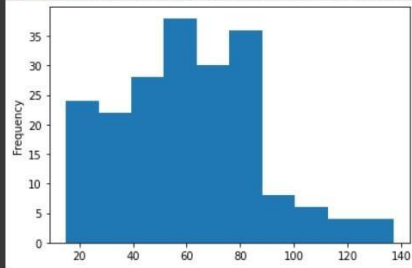
▾ Perform Below Visualizations

```
[ ] data.hist(figsize=(20,10), grid=False, layout=(2,4),bins=30)
plt.show()
```



```
[ ] data["Annual Income (k$)"].plot(kind='hist')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa78eca9290>

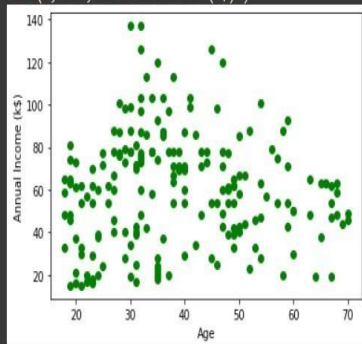


+ Code + Text

Connect Editing ^

```
[ ] from matplotlib import pyplot as plt
plt.scatter(data['Age'],data['Annual Income (k$)'],color='green')
plt.xlabel("Age")
plt.ylabel("Annual Income (k$)")
```

Text(0, 0.5, 'Annual Income (k\$)')

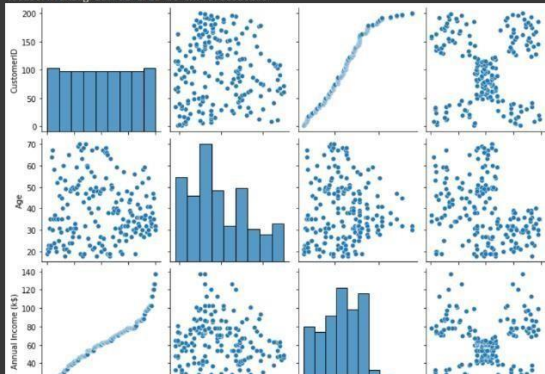


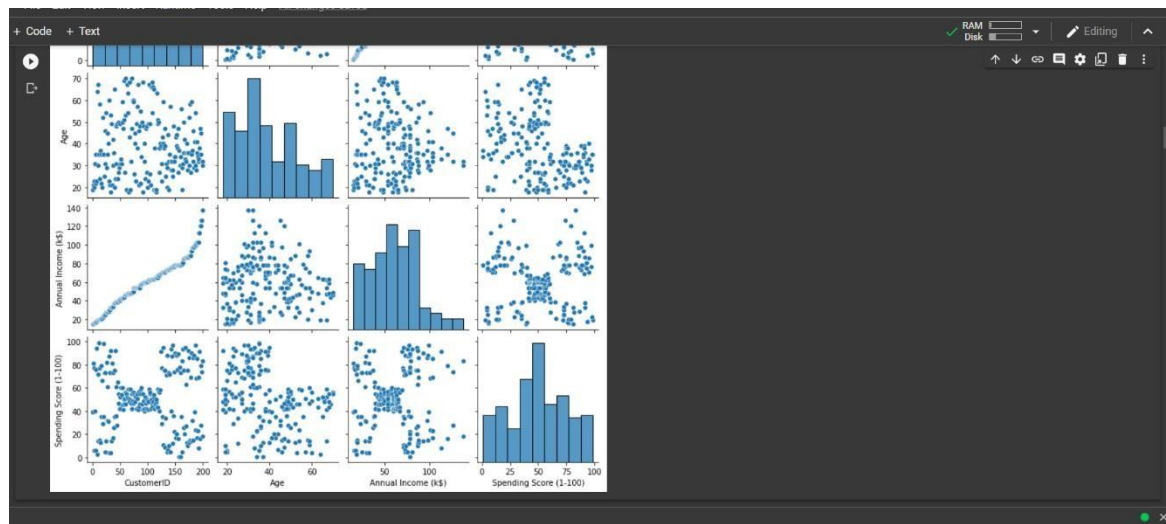
+ Code + Text

Connecting Editing ^

sns.pairplot(data)

<seaborn.axisgrid.PairGrid at 0x7fa78eb2c1d0>





Perform descriptive statistics on the dataset

```
[ ] data.describe()
```

	CustomerID	Age	Annual Income (k\$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

Check for Missing values and deal with them

```
[ ] data.isna().sum()
```

```
CustomerID      0
Gender           0
Age             0
Annual Income (k$)  0
Spending Score (1-100)  0
dtype: int64
```

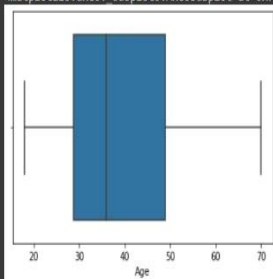
Find the outliers and replace them outliers.

```
[ ] data.skew()
```

```
CustomerID      0.000000
Age             0.485569
Annual Income (k$)  0.321843
Spending Score (1-100) -0.047220
dtype: float64
```

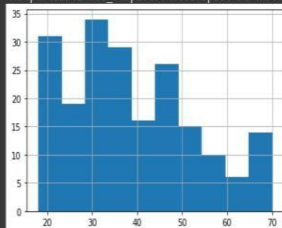
```
[ ] sns.boxplot(x=data['Age'],data=data)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa78f3fd150>



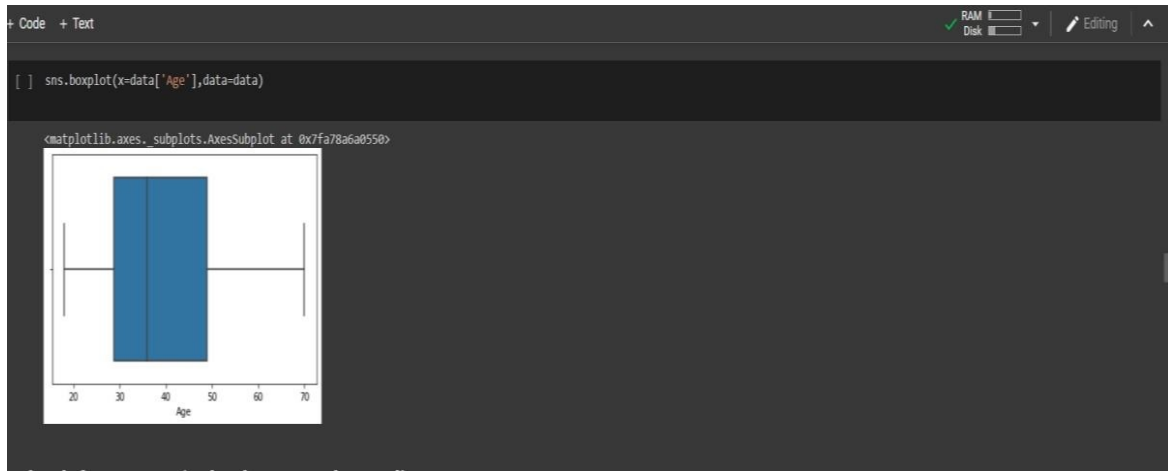
```
[ ] data['Age'].hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa78f493b50>






```
[ ] print('skewness value of Age:',data['Age'].skew())
```

skewness value of Age: 0.4855685096681657



+ Code + Text

RAM  Disk  Editing 




Check for Categorical columns and encoding

```
[ ] data.info
```

```
<bound method DataFrame.info of      CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0             1    Male   19             15           39
1             2    Male   21             15           81
2             3  Female   20             16            6
3             4  Female   23             16           77
4             5  Female   31             17           40
..          ...    ...   ...             ...           ...
195          196  Female   35            120           79
196          197  Female   45            126           28
197          198    Male   32            126           74
198          199    Male   32            137           18
199          200    Male   30            137           83

[200 rows x 5 columns]>
```

+ Code + Text

RAM  Disk  Editing 

```
[ ] from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data['Gender']=le.fit_transform(data['Gender'])
data.head()
```

	CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	1	19	15	39
1	2	1	21	15	81
2	3	0	20	16	6
3	4	0	23	16	77
4	5	0	31	17	40

```
[ ] data["Gender"].unique()

array([1, 0])
```

+ Code+ Text

RAM

Disk

Editing

Scaling the Data

x=data.drop(columns=['Gender','Age'])
print(x)

	CustomerID	Annual Income (k\$)	Spending Score (1-100)
0	1	15	39
1	2	15	81
2	3	16	6
3	4	16	77
4	5	17	40
..
195	196	120	79
196	197	126	28
197	198	126	74
198	199	137	18
199	200	137	83

[200 rows x 3 columns]

[] S=scale(x)
print(S)

[[-1.7234121 -1.73899919 -0.43480148]
[-1.70609137 -1.73899919 1.19570407]
[-1.68877065 -1.70082976 -1.71591298]
[-1.67144992 -1.70082976 1.04041783]
[-1.67144992 -1.70082976 1.04041783]
[-1.6541292 -1.66266033 -0.39597992]
[-1.6541292 -1.66266033 -0.39597992]
[-1.63680847 -1.66266033 1.00159627]
[-1.61948775 -1.62449091 -1.71591298]
[-1.60216702 -1.62449091 1.70038436]
[-1.5848463 -1.58632148 -1.83237767]
[-1.56752558 -1.58632148 0.84631002]
[-1.55020485 -1.58632148 -1.4053405]
[-1.53288413 -1.58632148 1.89449216]
[-1.5155634 -1.54815205 -1.36651894]
[-1.49824268 -1.54815205 1.04041783]
[-1.48092195 -1.54815205 -1.44416206]
[-1.46360123 -1.54815205 1.11806095]
[-1.4462805 -1.50998262 -0.59008772]
[-1.42895978 -1.50998262 0.61338066]
[-1.41163905 -1.43364376 -0.82301709]
[-1.39431833 -1.43364376 1.8556706]
[-1.3769976 -1.39547433 -0.59008772]
[-1.35967688 -1.39547433 0.88513158]
[-1.34235616 -1.3573049 -1.75473454]
[-1.32503543 -1.3573049 0.88513158]
[-1.30771471 -1.24279661 -1.4053405]
[-1.29039398 -1.24279661 1.23452563]
[-1.27307326 -1.24279661 -0.7065524]
[-1.25575253 -1.24279661 0.41927286]
[-1.23843181 -1.20462718 -0.74537397]
[-1.22111108 -1.20462718 1.42863343]

[[-1.7234121 -1.73899919 -0.43480148]
[-1.70609137 -1.73899919 1.19570407]
[-1.68877065 -1.70082976 -1.71591298]
[-1.67144992 -1.70082976 1.04041783]
[-1.6541292 -1.66266033 -0.39597992]
[-1.63680847 -1.66266033 1.00159627]
[-1.61948775 -1.62449091 -1.71591298]
[-1.60216702 -1.62449091 1.70038436]
[-1.5848463 -1.58632148 -1.83237767]
[-1.56752558 -1.58632148 0.84631002]
[-1.55020485 -1.58632148 -1.4053405]
[-1.53288413 -1.58632148 1.89449216]
[-1.5155634 -1.54815205 -1.36651894]
[-1.49824268 -1.54815205 1.04041783]
[-1.48092195 -1.54815205 -1.44416206]
[-1.46360123 -1.54815205 1.11806095]
[-1.4462805 -1.50998262 -0.59008772]
[-1.42895978 -1.50998262 0.61338066]
[-1.41163905 -1.43364376 -0.82301709]
[-1.39431833 -1.43364376 1.8556706]
[-1.3769976 -1.39547433 -0.59008772]
[-1.35967688 -1.39547433 0.88513158]
[-1.34235616 -1.3573049 -1.75473454]
[-1.32503543 -1.3573049 0.88513158]
[-1.30771471 -1.24279661 -1.4053405]
[-1.29039398 -1.24279661 1.23452563]
[-1.27307326 -1.24279661 -0.7065524]
[-1.25575253 -1.24279661 0.41927286]
[-1.23843181 -1.20462718 -0.74537397]
[-1.22111108 -1.20462718 1.42863343]

The Edit View Sheet Timeline Tools Help Extensions

+ Code+ Text

RAM

Disk

Editing

[-0.77077224 -0.67025518 -0.35715836]
[-0.75345152 -0.63208575 -0.00776431]
[-0.73613079 -0.63208575 -0.16305055]
[-0.71881007 -0.55574689 0.03105725]
[-0.70148935 -0.55574689 -0.16305055]
[-0.68416862 -0.55574689 0.22516505]
[-0.6668479 -0.55574689 0.18634349]
[-0.64952717 -0.51757746 0.06987881]
[-0.63220645 -0.51757746 0.34162973]
[-0.61488572 -0.47940803 0.03105725]
[-0.597565 -0.47940803 0.34162973]
[-0.58024427 -0.47940803 -0.00776431]
[-0.56292355 -0.47940803 -0.08540743]
[-0.54560282 -0.47940803 0.34162973]
[-0.5282821 -0.47940803 -0.12422899]
[-0.51096138 -0.4412386 0.18634349]
[-0.49364065 -0.4412386 -0.3183368]
[-0.47631993 -0.40306917 -0.04658587]
[-0.4589992 -0.40306917 0.22516505]
[-0.44167848 -0.25039146 -0.12422899]
[-0.42435775 -0.25039146 0.14752193]
[-0.40703703 -0.25039146 0.10870037]
[-0.3897163 -0.25039146 -0.08540743]
[-0.37239558 -0.25039146 0.06987881]
[-0.35507485 -0.25039146 -0.3183368]
[-0.33775413 -0.25039146 0.03105725]
[-0.3204334 -0.25039146 0.18634349]
[-0.30311268 -0.25039146 -0.35715836]
[-0.28579196 -0.25039146 -0.24069368]
[-0.26847123 -0.25039146 0.26398661]
[-0.25115051 -0.25039146 -0.16305055]
[-0.23382978 -0.43600347 -0.30300017]

The screenshot shows a code editor with a dark theme. The top bar has tabs for '+ Code' and '+ Text', and a right sidebar with icons for RAM, Disk, and Editing. The main area displays a list of coordinates in a table format, with each row enclosed in square brackets. The coordinates are organized into three columns: the first column contains values like 0.35907485, 0.37239558, etc.; the second column contains values like 0.24581112, 0.24581112, etc.; and the third column contains values like 0.22516505, -0.39597992, etc. The list is scrollable, and the bottom of the image shows the continuation of the list.

0.35907485	0.24581112	0.22516505
0.37239558	0.24581112	-0.39597992
0.3897163	0.32214998	0.30280817
0.40703703	0.32214998	1.58391968
0.42435775	0.36031941	-0.82301709
0.44167848	0.36031941	1.04041783
0.4589992	0.39848884	-0.59008772
0.47631993	0.39848884	1.73920502
0.49364065	0.39848884	-1.52189518
0.51096138	0.39848884	0.96277471
0.5282821	0.39848884	-1.5994483
0.54560282	0.39848884	0.96277471
0.56292355	0.43665827	-0.62890928
0.58024427	0.43665827	0.80748846
0.597565	0.4748277	-1.75473454
0.61488572	0.4748277	1.46745499
0.63220645	0.4748277	-1.67709142
0.64952717	0.4748277	0.88513158
0.6668479	0.51299713	-1.56062674
0.68416862	0.51299713	0.84631002
0.70148935	0.55116656	-1.75473454
0.71881007	0.55116656	1.6615628
0.73613079	0.58933599	-0.39597992
0.75345152	0.58933599	1.42863343
0.77077224	0.62750542	-1.48298362
0.78809297	0.62750542	1.81684904
0.80541369	0.62750542	-0.55126616
0.82273442	0.62750542	0.92395314
0.84005514	0.66567484	-1.09476801
0.85737587	0.66567484	1.54509812
0.87469659	0.66567484	-1.28887582
0.89201732	0.66567484	1.46745499

+ Code

+ Text

✓ RAM

Disk

Editing

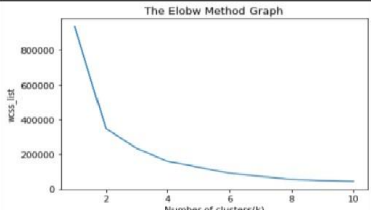
↕

Perform any of the clustering algorithms

```
from sklearn.cluster import KMeans
wcss_list= [] #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
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_)
plt.plot(range(1, 11), wcss_list)
plt.title('The Elbow Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```

The Elbow Method Graph



Number of clusters(k)	wcss_list
1	800000
2	350000
3	250000
4	180000
5	130000
6	100000
7	80000
8	60000
9	50000
10	50000

[illegible]

```
+ Code + Text
Split the data into dependent and independent variables

[ ] x=data.drop(columns=['Annual Income (k$)'],axis=1)
    print(x)

   CustomerID  Gender  Age  Spending Score (1-100)
0           1        1   19                   39
1           2        1   21                   81
2           3        0   20                    6
3           4        0   23                   77
4           5        0   31                   40
...         ...     ...   ...
195         196        0   35                   79
196         197        0   45                   28
197         198        1   32                   74
198         199        1   32                   18
199         200        1   30                   83

[200 rows x 4 columns]

[ ] y=data['Annual Income (k$)']
    y
0    15
1    15
2    16
3    16
4    17
```


+ Code

+ Text

✓ RAM

Disk

Editing

↕

Build the model

```
[ ] from sklearn.linear_model import LinearRegression
    LR = LinearRegression()
```

Train the Model

```
[ ] LR.fit(x_train,y_train)

    LinearRegression()
```

```
+ Code + Text
[ ] pred_LR.predict(x_test)
pred

array([[ 41.79651469,  35.44897396,  32.32182941,  62.15230947,
        97.15499 , 102.74527464,  57.52984542, 18.50596884,
        28.90850195,  90.05616474,  90.63951146,  25.17877999,
        21.47607213,  56.15450717,  65.58284431,  68.81365584,
        85.74449988,  31.45756756,  76.51559556,  42.98039276,
        38.70178627,  23.89238204,  36.61738406,  57.67164216,
        29.74845621,  86.65460588,  33.53032334,  29.31235764,
        100.75984295,  28.3364555 ,  37.02836966,  88.57006476,
        101.81449573,  93.23392219,  94.16104415,  58.75918464,
        93.31570423,  49.53263005,  46.78164703,  91.618992 ,
        64.85923756,  63.89021447,  98.96847593,  22.93975353,
        41.82689378,  24.95860094,  65.82297944,  33.18229176,
        96.7187877 ,  70.4308092 ,  59.76768524,  70.1173078 ,
        69.1581952 ,  40.54244593,  30.19338393,  94.32293272,
        95.33656664,  64.12923371, 102.85955135,  76.19945402]])

[ ] pred.astype(int)

array([[ 41,  35,  32,  62,  97, 102,  57,  18,  28,  90,  90,  25,  21,
        96,  65,  68,  85,  31,  76,  42,  38,  23,  36,  57,  29,  86,
        33,  29, 180,  28,  27,  88, 101,  51,  94,  58,  93,  49,  46,
        91,  64,  63,  98,  22,  41,  24,  65,  33,  96,  70,  59,  70,
        69,  40,  30,  94,  95,  64, 102,  76]])
```

```
+ Code + Text
[ ] y_test

58      46
40      38
24      33
102     62
184     99
198    137
95      60
4       17
29     29
168     87
171     87
18      23
11      19
89      58
110     63
118     67
159     78
35      33
136     73
59      46
51      42
16      21
44      39
94      60
31      30
162     81
38      37
28      29
193    113
```

```
+ Code + Text
[ ] y_test

38      37
193    113
27      28
47      40
165     85
194    120
177     88
176     88
97      60
174     88
73      50
69      48
172     87
108     63
107     63
189    103
14      20
56      44
19      23
114     65
39      37
185     99
124     70
98      61
123     69
119     67
53      43
33      33
179     93
181     97
106     63
```

Measure the performance using Evaluation Metrics.

```
from sklearn.metrics import r2_score
score=r2_score(pred,y_test)
score
```

0.9234274149757858

