ajwiyvy2p

July 31, 2023

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
df=pd.read_csv("/content/12_mobile_prices_2023.csv")
df
```

[ ]:

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery
0	POCO C50 (Royal Blue, 32 GB)	4.2	33,561	2 GB RAM	32 GB ROM	8MP Dual Camera	5MP Front Camera	5000 mAh
1	POCO M4 5G (Cool Blue, 64 GB)	4.2	77,128	4 GB RAM	64 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh
2	POCO C51 (Royal Blue, 64 GB)	4.3	15,175	4 GB RAM	64 GB ROM	8MP Dual Rear Camera		
3	POCO C55 (Cool Blue, 64 GB)	4.2	22,621	4 GB RAM	64 GB ROM	50MP Dual Rear Camera		
4	POCO C51 (Power Black, 64 GB)	4.3	15,175	4 GB RAM	64 GB ROM	8MP Dual Rear Camera		
…	…	…	…	…	…	…		
1831	Infinix Note 7 (Forest Green, 64 GB)	4.3	25,582	4 GB RAM	64 GB ROM	48MP + 2MP + 2MP + AI Lens Camera		
1832	Infinix Note 7 (Bolivia Blue, 64 GB)	4.3	25,582	4 GB RAM	64 GB ROM	48MP + 2MP + 2MP + AI Lens Camera		
1833	Infinix Note 7 (Aether Black, 64 GB)	4.3	25,582	4 GB RAM	64 GB ROM	48MP + 2MP + 2MP + AI Lens Camera		
1834	Infinix Zero 8i (Silver Diamond, 128 GB)	4.2	7,117	8 GB RAM	128 GB ROM	48MP + 8MP + 2MP + AI Lens Camera		
1835	Infinix S5 (Quetzal Cyan, 64 GB)	4.3	15,701	4 GB RAM	64 GB ROM	16MP + 5MP + 2MP + Low Light Sensor		

```
2                 5MP Front Camera  5000 mAh
3                 5MP Front Camera  5000 mAh
4                 5MP Front Camera  5000 mAh
…                              …       …
1831            16MP Front Camera  5000 mAh
1832            16MP Front Camera  5000 mAh
1833            16MP Front Camera  5000 mAh
1834  16MP + 8MP Dual Front Camera  4500 mAh
1835            32MP Front Camera  4000 mAh

                                              Processor Price in INR  \
0     Mediatek Helio A22 Processor, Upto 2.0 GHz Pro…       5,649
1                      Mediatek Dimensity 700 Processor      11,999
2                                  Helio G36 Processor       6,999
3                      Mediatek Helio G85 Processor          7,749
4                                  Helio G36 Processor       6,999
…                                                    …          …
1831                    MediaTek Helio G70 Processor      14,999
1832                    MediaTek Helio G70 Processor      14,999
1833                    MediaTek Helio G70 Processor      14,999
1834                   MediaTek Helio G90T Processor      18,999
1835                   Helio P22 (MTK6762) Processor      10,999

     Date of Scraping
0          2023-06-17
1          2023-06-17
2          2023-06-17
3          2023-06-17
4          2023-06-17
…                …
1831       2023-06-17
1832       2023-06-17
1833       2023-06-17
1834       2023-06-17
1835       2023-06-17

[1836 rows x 11 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
                      Phone Name  Rating ?/5 Number of Ratings       RAM  \
0   POCO C50 (Royal Blue, 32 GB)         4.2            33,561  2 GB RAM
1  POCO M4 5G (Cool Blue, 64 GB)         4.2            77,128  4 GB RAM
2   POCO C51 (Royal Blue, 64 GB)         4.3            15,175  4 GB RAM
3    POCO C55 (Cool Blue, 64 GB)         4.2            22,621  4 GB RAM
4  POCO C51 (Power Black, 64 GB)         4.3            15,175  4 GB RAM
```

```
  ROM/Storage       Back/Rare Camera      Front Camera   Battery  \
0   32 GB ROM        8MP Dual Camera  5MP Front Camera  5000 mAh
1   64 GB ROM             50MP + 2MP  8MP Front Camera  5000 mAh
2   64 GB ROM   8MP Dual Rear Camera  5MP Front Camera  5000 mAh
3   64 GB ROM  50MP Dual Rear Camera  5MP Front Camera  5000 mAh
4   64 GB ROM   8MP Dual Rear Camera  5MP Front Camera  5000 mAh

                                          Processor Price in INR  \
0  Mediatek Helio A22 Processor, Upto 2.0 GHz Pro…       5,649
1                  Mediatek Dimensity 700 Processor      11,999
2                              Helio G36 Processor       6,999
3                      Mediatek Helio G85 Processor       7,749
4                              Helio G36 Processor       6,999

  Date of Scraping
0       2023-06-17
1       2023-06-17
2       2023-06-17
3       2023-06-17
4       2023-06-17
```

1 DATA CLEANING AND DATA PREPROCESSING

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1836 entries, 0 to 1835
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Phone Name         1836 non-null   object
 1   Rating ?/5         1836 non-null   float64
 2   Number of Ratings  1836 non-null   object
 3   RAM                1836 non-null   object
 4   ROM/Storage        1662 non-null   object
 5   Back/Rare Camera   1827 non-null   object
 6   Front Camera       1435 non-null   object
 7   Battery            1826 non-null   object
 8   Processor          1781 non-null   object
 9   Price in INR       1836 non-null   object
 10  Date of Scraping   1836 non-null   object
dtypes: float64(1), object(10)
memory usage: 157.9+ KB
```

[ ]:
```
df.describe()
```

[ ]:
```
        Rating ?/5
count  1836.000000
mean      4.210512
std       0.543912
min       0.000000
25%       4.200000
50%       4.300000
75%       4.400000
max       4.800000
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM', 'ROM/Storage',
       'Back/Rare Camera', 'Front Camera', 'Battery', 'Processor',
       'Price in INR', 'Date of Scraping'],
      dtype='object')
```

[ ]:
```
df1=df.dropna(axis=1)
df1
```

[ ]:
```
                                  Phone Name  Rating ?/5 Number of Ratings  \
0                 POCO C50 (Royal Blue, 32 GB)         4.2            33,561
1                POCO M4 5G (Cool Blue, 64 GB)         4.2            77,128
2                 POCO C51 (Royal Blue, 64 GB)         4.3            15,175
3                  POCO C55 (Cool Blue, 64 GB)         4.2            22,621
4                POCO C51 (Power Black, 64 GB)         4.3            15,175
…                                          …         …                 …
1831      Infinix Note 7 (Forest Green, 64 GB)         4.3            25,582
1832      Infinix Note 7 (Bolivia Blue, 64 GB)         4.3            25,582
1833      Infinix Note 7 (Aether Black, 64 GB)         4.3            25,582
1834  Infinix Zero 8i (Silver Diamond, 128 GB)         4.2             7,117
1835          Infinix S5 (Quetzal Cyan, 64 GB)         4.3            15,701

           RAM Price in INR Date of Scraping
0     2 GB RAM       5,649       2023-06-17
1     4 GB RAM      11,999       2023-06-17
2     4 GB RAM       6,999       2023-06-17
3     4 GB RAM       7,749       2023-06-17
4     4 GB RAM       6,999       2023-06-17
…        …           …              …
1831  4 GB RAM      14,999       2023-06-17
1832  4 GB RAM      14,999       2023-06-17
1833  4 GB RAM      14,999       2023-06-17
1834  8 GB RAM      18,999       2023-06-17
1835  4 GB RAM      10,999       2023-06-17

[1836 rows x 6 columns]
```

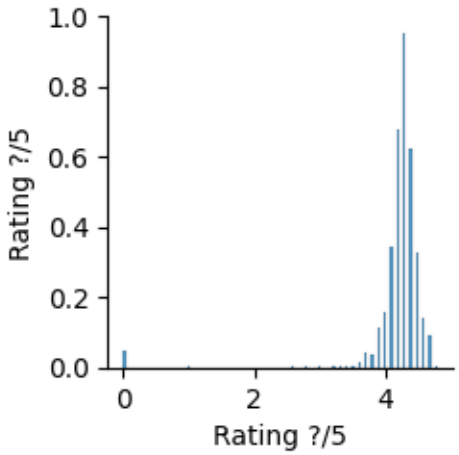
[ ]:
```
df1.columns
```

[ ]:
```
Index(['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM', 'Price in INR',
       'Date of Scraping'],
      dtype='object')
```

2 EDA AND VISUALIZATION

[ ]:
```
sns.pairplot(df1)
```

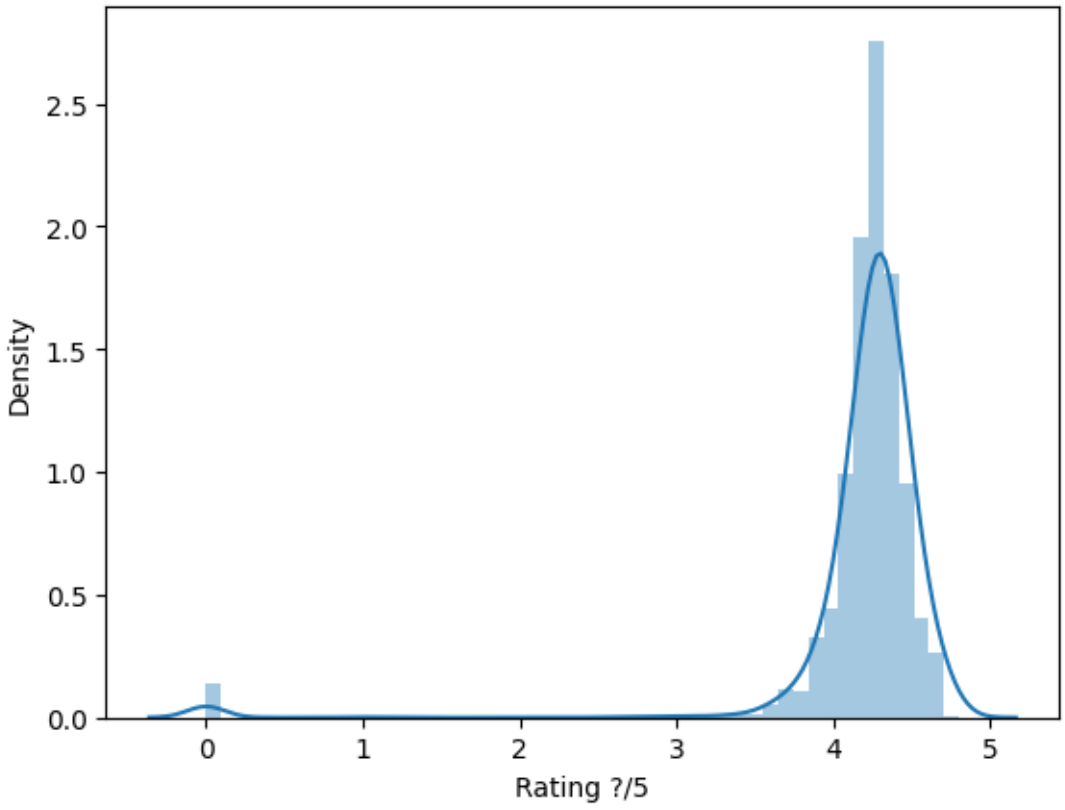
[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7cd0ea06aad0>
```



[ ]:
```
sns.distplot(df1['Rating ?/5'])
```

```
<ipython-input-10-75fcdaa8fe58>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df1['Rating ?/5'])
```

[ ]:
```
<Axes: xlabel='Rating ?/5', ylabel='Density'>
```



[ ]:
```
sns.heatmap(df1.corr())
```

```
<ipython-input-11-3ed1a1a51dc0>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df1.corr())
```

[ ]: <Axes: >

3 TO TRAIN THE MODEL AND MODEL BULDING

[ ]:
```
x=df[['Rating ?/5','Rating ?/5']]
y=df['Rating ?/5']
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

[ ]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
lr.intercept_
```

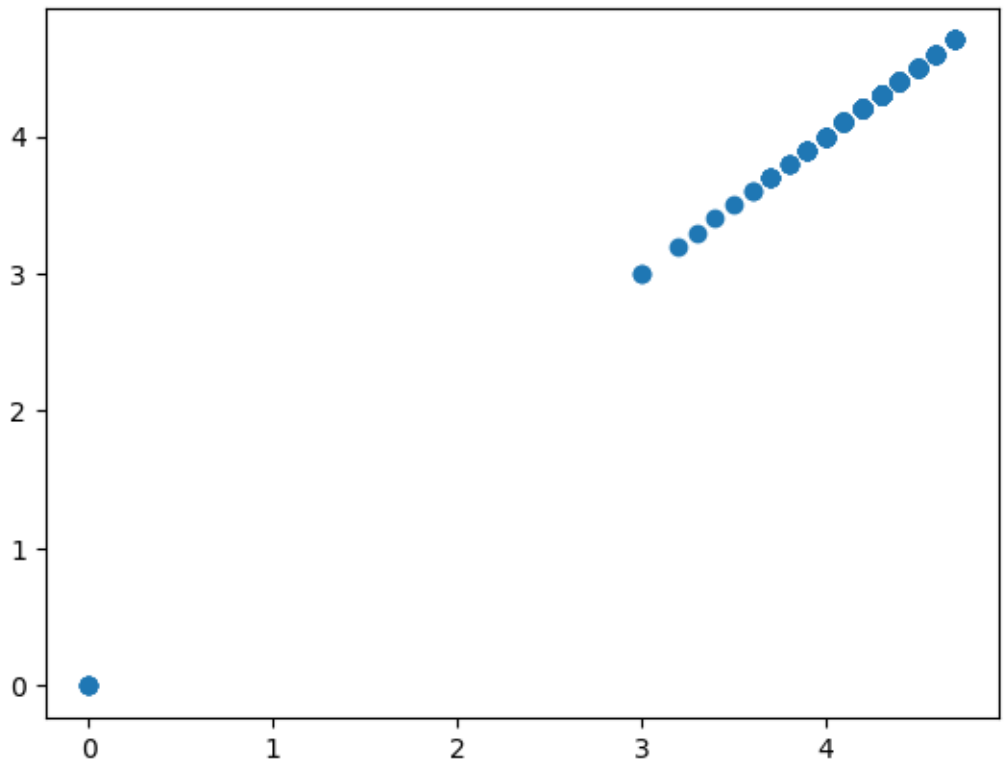
[ ]:
```
-8.881784197001252e-16
```

[ ]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[ ]:
```
             Co-efficient
Rating ?/5   3.545583e-16
Rating ?/5   1.000000e+00
```

[ ]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7cd0e4b98a30>
```



4 ACCURACY

[ ]:
```
lr.score(x_test,y_test)
```

[ ]:
```
1.0
```

[ ]:
```
lr.score(x_train,y_train)
```

[ ]:
```
1.0
```

[ ]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
```

```
rr.fit(x_train,y_train)
```

[ ]: Ridge(alpha=10)

[ ]:
```
rr.score(x_train,y_train)
```

[ ]: 0.999865212196738

[ ]:
```
rr.score(x_test,y_test)
```

[ ]: 0.9998649843620382

[ ]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

[ ]: Lasso(alpha=10)

[ ]:
```
la.score(x_train,y_train)
```

[ ]: 0.0

[ ]:
```
la.score(x_test,y_test)
```

[ ]: -0.001690321336058842

[ ]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

[ ]: ElasticNet()

[ ]:
```
print(en.coef_)
print(en.intercept_)
```

```
[0. 0.]
4.204824902723735
```

[ ]:
```
prediction = en.predict(x_test)
prediction
```

[ ]:
```
array([4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
```

4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,

```
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249,
       4.2048249, 4.2048249, 4.2048249, 4.2048249, 4.2048249])
```

[ ]:
```
en.score(x_test,y_test)
```

[ ]: -0.001690321336058842

[ ]:
```
from sklearn import metrics
print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error: ", metrics.mean_squared_error(y_test,prediction))
```

```
print("Root Mean Squared Error: ", np.sqrt(metrics.
  ↪mean_squared_error(y_test,prediction)))
```

```
Mean Absolute Error:  0.20591821025796753
Mean Squared Error:  0.21280656017384
Root Mean Squared Error:  0.46130961422220546
```