LinearRegression ¶

In [1]:

```
import numpy as np
import pandas as pd
```

data collection

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [3]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\12_mobile_prices_2023.csv")
df
```

Out[3]:

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor
0	POCO C50 (Royal Blue, 32 GB)	4.2	33561.0	2 GB RAM	32 GB ROM	8MP Dual Camera	5MP Front Camera	5000 mAh	Mediatek Helio A22 Processor, Upto 2.0 GHz Pro...
1	POCO M4 5G (Cool Blue, 64 GB)	4.2	77128.0	4 GB RAM	64 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor
2	POCO C51 (Royal Blue, 64 GB)	4.3	15175.0	4 GB RAM	64 GB ROM	8MP Dual Rear Camera	5MP Front Camera	5000 mAh	Helio G36 Processor
3	POCO C55 (Cool Blue, 64 GB)	4.2	22621.0	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor
4	POCO C51 (Power Black, 64 GB)	4.3	15175.0	4 GB RAM	64 GB ROM	8MP Dual Rear Camera	5MP Front Camera	5000 mAh	Helio G36 Processor
...	...	...	...	...	...	...	...	...	...
1831	Infinix Note 7 (Forest Green, 64 GB)	4.3	25582.0	4 GB RAM	64 GB ROM	48MP + 2MP + 2MP + AI Lens Camera	16MP Front Camera	5000 mAh	MediaTek Helio G70 Processor
1832	Infinix Note 7 (Bolivia Blue, 64 GB)	4.3	25582.0	4 GB RAM	64 GB ROM	48MP + 2MP + 2MP + AI Lens Camera	16MP Front Camera	5000 mAh	MediaTek Helio G70 Processor
1833	Infinix Note 7 (Aether Black, 64 GB)	4.3	25582.0	4 GB RAM	64 GB ROM	48MP + 2MP + 2MP + AI Lens Camera	16MP Front Camera	5000 mAh	MediaTek Helio G70 Processor
1834	Infinix Zero 8i (Silver Diamond, 128 GB)	4.2	7117.0	8 GB RAM	128 GB ROM	48MP + 8MP + 2MP + AI Lens Camera	16MP + 8MP Dual Front Camera	4500 mAh	MediaTek Helio G90T Processor
1835	Infinix S5 (Quetzal Cyan, 64 GB)	4.3	15701.0	4 GB RAM	64 GB ROM	16MP + 5MP + 2MP + Low Light Sensor	32MP Front Camera	4000 mAh	Helio P22 (MTK6762) Processor

1836 rows × 11 columns

first 10 rows

In [4]:

```
df.head(10)
```

Out[4]:

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	Pri
0	POCO C50 (Royal Blue, 32 GB)	4.2	33561.0	2 GB RAM	32 GB ROM	8MP Dual Camera	5MP Front Camera	5000 mAh	Mediatek Helio A22 Processor, Upto 2.0 GHz Pro...	₹5
1	POCO M4 5G (Cool Blue, 64 GB)	4.2	77128.0	4 GB RAM	64 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor	₹11
2	POCO C51 (Royal Blue, 64 GB)	4.3	15175.0	4 GB RAM	64 GB ROM	8MP Dual Rear Camera	5MP Front Camera	5000 mAh	Helio G36 Processor	₹6
3	POCO C55 (Cool Blue, 64 GB)	4.2	22621.0	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7
4	POCO C51 (Power Black, 64 GB)	4.3	15175.0	4 GB RAM	64 GB ROM	8MP Dual Rear Camera	5MP Front Camera	5000 mAh	Helio G36 Processor	₹6
5	POCO M4 5G (Power Black, 64 GB)	4.2	77128.0	4 GB RAM	64 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor	₹11
6	POCO C55 (Power Black, 64 GB)	4.2	22621.0	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7
7	POCO C55 (Forest Green, 64 GB)	4.2	22621.0	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7
8	POCO C55 (Cool Blue, 128 GB)	4.1	13647.0	6 GB RAM	128 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹9
9	POCO M4 5G (Yellow, 128 GB)	4.2	40525.0	6 GB RAM	128 GB ROM	50MP + 2MP	8MP Front Camera	5000 mAh	Mediatek Dimensity 700 Processor	₹13

data cleaning

In [5]:

```
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   float64
 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(2), object(9)
memory usage: 157.9+ KB
```

In [6]:

```
df.describe()
```

Out[6]:

	Rating ?/5	Number of Ratings
count	1836.000000	1.836000e+03
mean	4.210512	4.669473e+04
std	0.543912	9.756649e+04
min	0.000000	0.000000e+00
25%	4.200000	1.313000e+03
50%	4.300000	8.391000e+03
75%	4.400000	4.149500e+04
max	4.800000	1.342530e+06

In [7]:

```
df.columns
```

Out[7]:

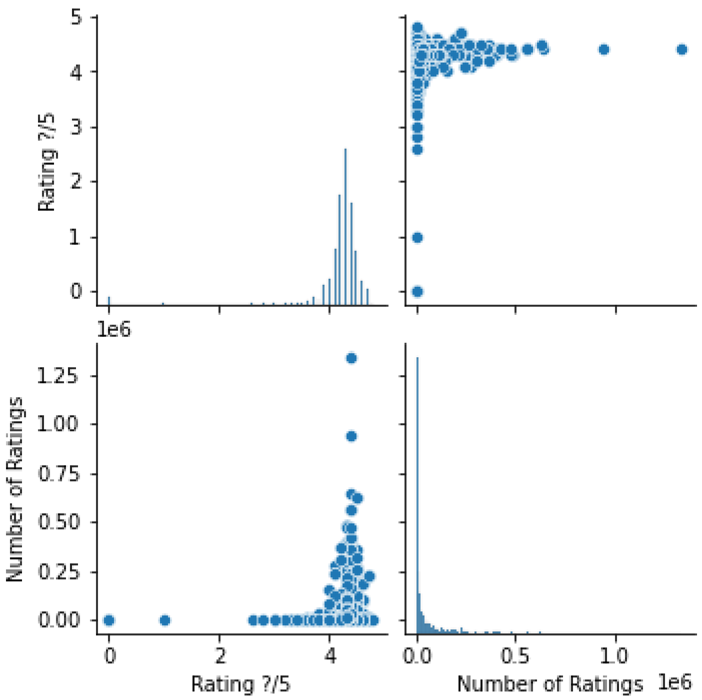
```
Index(['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM', 'ROM/Storag
e',
       'Back/Rare Camera', 'Front Camera', 'Battery', 'Processor',
       'Price in INR', 'Date of Scraping'],
      dtype='object')
```

In [8]:

```
sb.pairplot(df)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x233fb33d7f0>
```



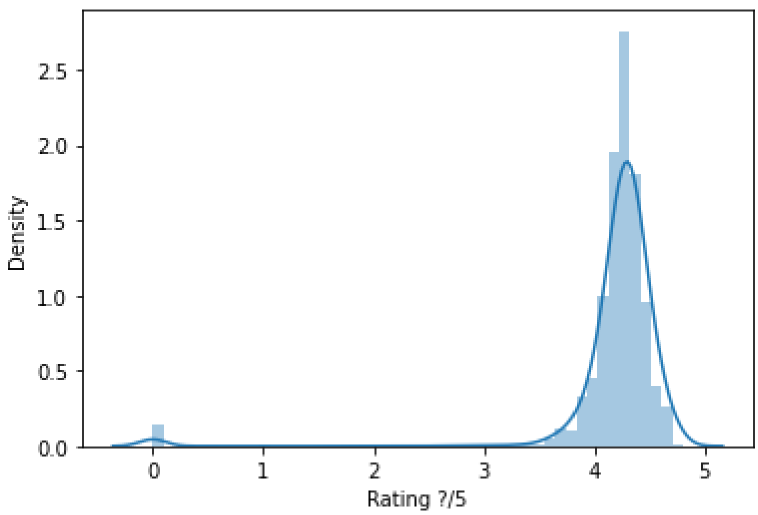
In [9]:

```
sb.distplot(df["Rating ?/5"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='Rating ?/5', ylabel='Density'>
```



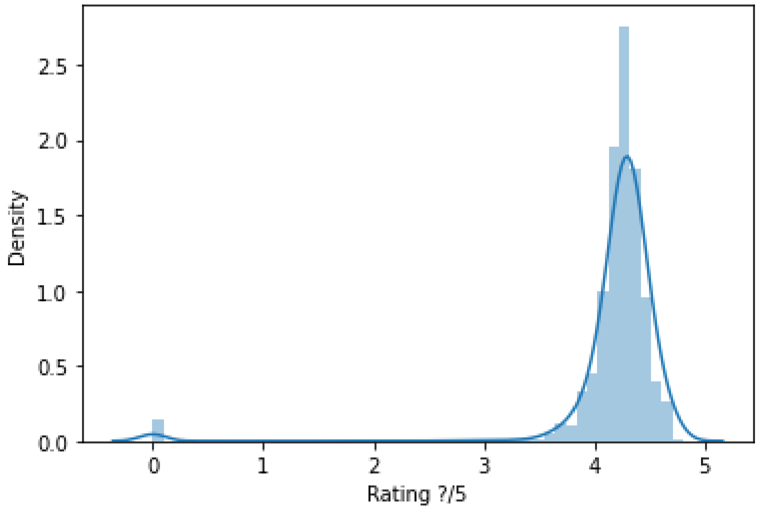
In [10]:

```
sb.distplot(df["Rating ?/5"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in
a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]:

```
<AxesSubplot:xlabel='Rating ?/5', ylabel='Density'>
```



In [11]:

```
df1=df[[ 'Rating ?/5', 'Number of Ratings']]
df1
```

Out[11]:

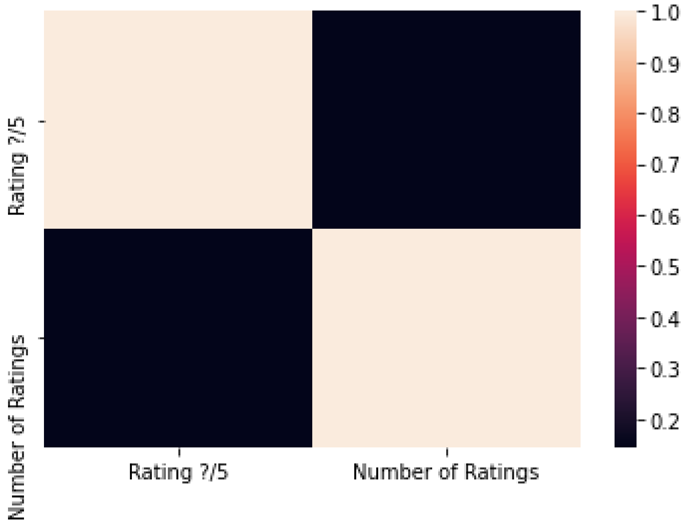
	Rating ?/5	Number of Ratings
0	4.2	33561.0
1	4.2	77128.0
2	4.3	15175.0
3	4.2	22621.0
4	4.3	15175.0
...	...	...
1831	4.3	25582.0
1832	4.3	25582.0
1833	4.3	25582.0
1834	4.2	7117.0

In [12]:

```
sb.heatmap(df1.corr())
```

Out[12]:

```
<AxesSubplot:>
```



model building

In [13]:

```
x = df1[[ 'Rating ?/5', 'Number of Ratings']]
y = df1['Rating ?/5']
```

In [14]:

```
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)
```

linear regression

In [15]:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
print(lr.intercept_)
```

```
3.552713678800501e-15
```

In [17]:

```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[17]:

	Co_efficient
Rating ?/5	1.0
Number of Ratings	0.0

In [18]:

```
print(lr.score(x_test,y_test))
```

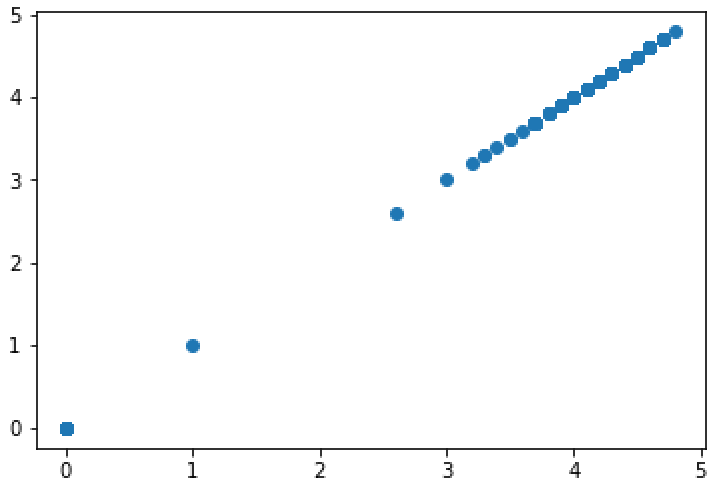
```
1.0
```

In [19]:

```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x233fdb449a0>
```



lasso and ridge regression

In [20]:

```
lr.score(x_test,y_test)
```

Out[20]:

```
1.0
```

In [21]:

```
lr.score(x_train,y_train)
```

Out[21]:

```
1.0
```

In [22]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:

```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[23]:

```
0.9989776381624211
```

In [24]:

```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[24]:

```
0.021594734348732825
```

elasticnet

In [25]:

```
from sklearn.linear_model import ElasticNet
e = ElasticNet()
e.fit(x_train,y_train)
```

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(e.coef_)
```

```
[0.00000000e+00 7.58286151e-07]
```

In [27]:

```
print(e.intercept_)
```

```
4.19216616679754
```

In [28]:

```
predictions = e.predict(x_test)
predictions
```

Out[28]:

```
array([4.19315346, 4.21136976, 4.19448576, 4.27364401, 4.19333241,
       4.19327781, 4.21619095, 4.29548265, 4.19319213, 4.19760763,
       4.30271974, 4.21144938, 4.19270834, 4.19234057, 4.19264389,
       4.1953995 , 4.30672349, 4.34051272, 4.19216617, 4.19344615,
       4.19470946, 4.20057708, 4.2138668 , 4.30672349, 4.1950287 ,
       4.20242881, 4.19218588, 4.1929017 , 4.21928627, 4.22820599,
       4.2716065 , 4.19992798, 4.19259687, 4.19470946, 4.20960978,
       4.19234361, 4.26541661, 4.55961799, 4.21014589, 4.21972911,
       4.19327857, 4.24444545, 4.19444178, 4.23389465, 4.29117938,
       4.21780382, 4.20150749, 4.1924892 , 4.19421506, 4.37768087,
       4.22626478, 4.20047774, 4.22414385, 4.194645  , 4.1953995 ,
       4.19814525, 4.19563988, 4.1935508 , 4.42542636, 4.2037634 ,
       4.19387989, 4.25355019, 4.19401639, 4.21071536, 4.19237545,
       4.28956196, 4.19557694, 4.19323156, 4.20057708, 4.19659228,
       4.19528651, 4.34051272, 4.21972911, 4.19270834, 4.20264568,
       4.19270834, 4.19261811, 4.1924892 , 4.20924884, 4.22129649,
       4.19327857, 4.19310189, 4.19371686, 4.35345818, 4.19219195,
       4.55961799, 4.19238986, 4.19421202, 4.3358068 , 4.34051272,
```

In [29]:

```
print(e.score(x_test,y_test))
```

```
0.012885651390037056
```

In [30]:

```
from sklearn import metrics
```

mean absolute error

In [31]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predictions))
```

```
Mean Absolute Error: 0.2531499665858157
```

mean squared error

In [32]:

```
print("Mean Squared Error:", metrics.mean_squared_error(y_test,predictions))
```

```
Mean Squared Error: 0.42228846838100303
```

root mean squared error

In [33]:

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

```
Root Mean Squared Error 0.6498372629982087
```

In [ ]: