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\fiat500_VehicleSelection_Dataset.csv")
df
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1	lounge	51	882	25000	1	44.907242	8.611560	890
1	2	pop	51	1186	32500	1	45.666359	12.241890	880
2	3	sport	74	4658	142228	1	45.503300	11.417840	420
3	4	lounge	51	2739	160000	1	40.633171	17.634609	600
4	5	pop	73	3074	106880	1	41.903221	12.495650	570
...	...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	520
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	460
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	750
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	599
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	790

1538 rows × 9 columns

In [4]:
```
df.head(10)
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
5	6	pop	74	3623	70225	1	45.000702	7.682270	7900
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
8	9	sport	73	4049	76000	1	45.548000	11.549470	5600
9	10	sport	51	3653	89000	1	45.438301	10.991700	6000

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [6]:
```
df.describe()
```

Out[6]:

	ID	engine_power	age_in_days	km	previous_owners	lat	
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.0
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.5
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.3
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.2
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.5
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.8
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.7
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.3

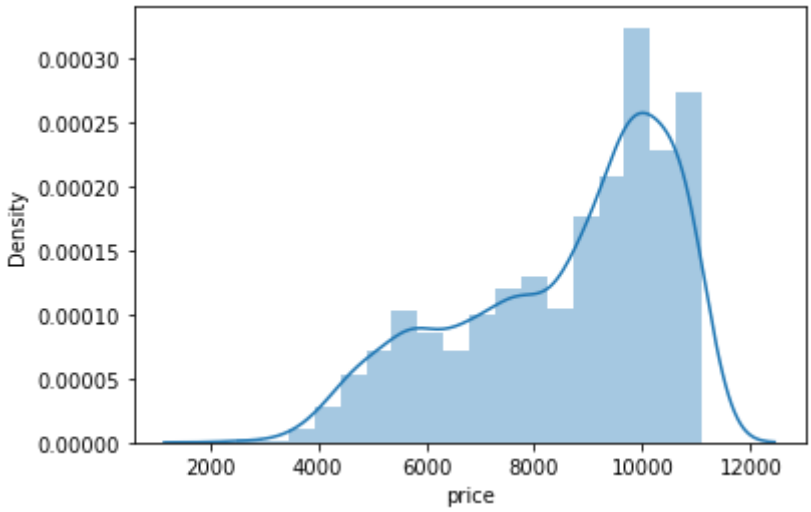
In [7]:
```
df.columns
```

Out[7]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [8]:
```
sb.distplot(df["price"])
```

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

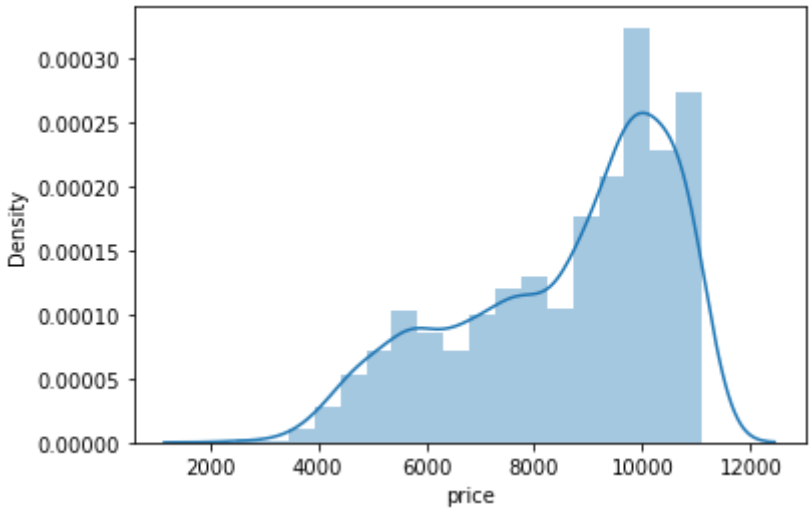
Out[8]:
```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



In [9]:
```
sb.distplot(df["price"])
```

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

Out[9]: <AxesSubplot:xlabel='price', ylabel='Density'>



In [10]:
```
df1=df[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
df1
```

Out[10]:

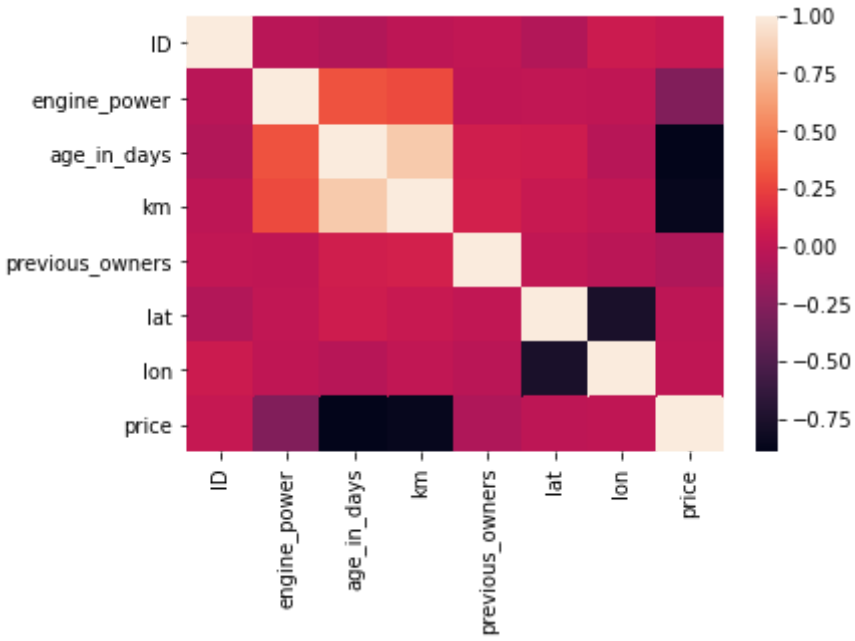
	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1	lounge	51	882	25000	1	44.907242	8.611560	89(
1	2	pop	51	1186	32500	1	45.666359	12.241890	88(
2	3	sport	74	4658	142228	1	45.503300	11.417840	42(
3	4	lounge	51	2739	160000	1	40.633171	17.634609	60(
4	5	pop	73	3074	106880	1	41.903221	12.495650	57(
...	...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	52(
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	46(

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	750
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	599
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	790

1538 rows × 9 columns

In [11]:
```
sb.heatmap(df1.corr())
```

Out[11]: <AxesSubplot:>



In [15]:
```
x = df1[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
y = df1['price']
```

In [16]:
```
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)
```

In [17]:
```
from sklearn.linear_model import LinearRegression

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

Out[17]: LinearRegression()

In [18]:
```
print(lr.intercept_)
```

```
-1.8189894035458565e-12
```

In [19]:
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[19]:

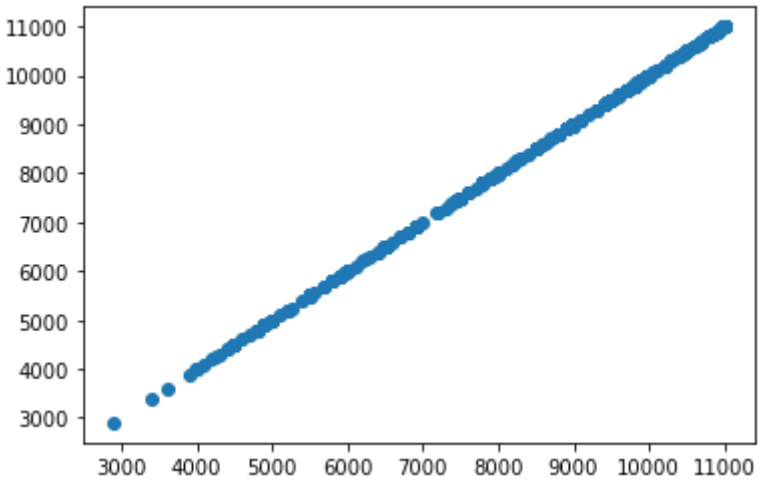
	Co_efficient
ID	1.244282e-16
engine_power	-9.009622e-14
age_in_days	5.655280e-16
km	-2.517120e-17
previous_owners	9.150978e-14
lat	1.146320e-15
lon	-1.482767e-14
price	1.000000e+00

In [20]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [21]:
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x19dbcdb18b0>
```



In [22]:
```
lr.score(x_test,y_test)
```

Out[22]:
```
1.0
```

In [23]:
```
lr.score(x_train,y_train)
```

Out[23]:
```
1.0
```

In [24]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[25]:
```
1.0
```

In [26]:
```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[26]:
```
0.9999999886151987
```

In [ ]:

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [27]:
```
df = pd.read_csv(r"C:\Users\user\Desktop\2015.csv")
df
```

Out[27]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedo
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.665
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.628
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.649
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.669
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.632
...	...	...	...	...	...	...	...	...	
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	0.592
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	0.484
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	0.156
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	0.118
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	0.364

158 rows × 12 columns

In [28]:
```
df.head(10)
```

Out[28]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297
5	Finland	Western Europe	6	7.406	0.03140	1.29025	1.31826	0.88911	0.64169
6	Netherlands	Western Europe	7	7.378	0.02799	1.32944	1.28017	0.89284	0.61576
7	Sweden	Western Europe	8	7.364	0.03157	1.33171	1.28907	0.91087	0.65980
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63938
9	Australia	Australia and New Zealand	10	7.284	0.04083	1.33358	1.30923	0.93156	0.65124

In [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        158 non-null    object 
 1   Region                         158 non-null    object 
 2   Happiness Rank                 158 non-null    int64  
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

In [30]:
```
df.describe()
```

Out[30]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Govern Corrup
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.0(

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	(Govern Corrup
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.1
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.1
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.0
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.0
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.1
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.1
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.5

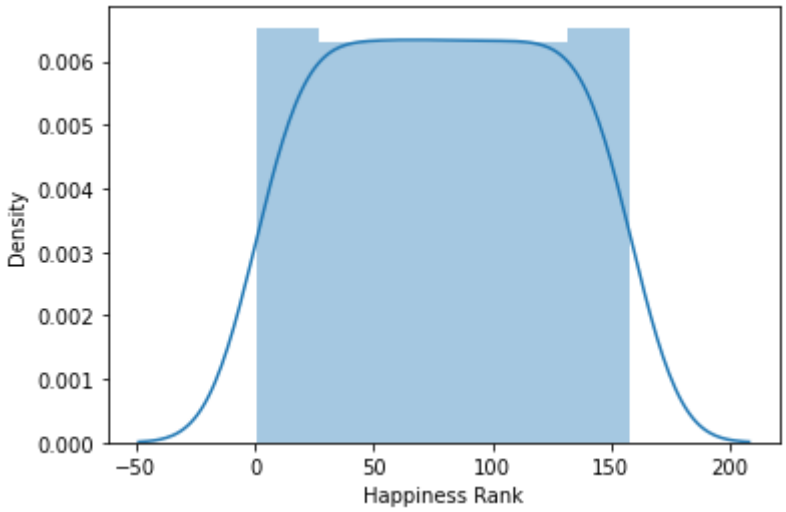
In [31]:
```
df.columns
```

Out[31]:
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [32]:
```
sb.distplot(df["Happiness Rank"])
```

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

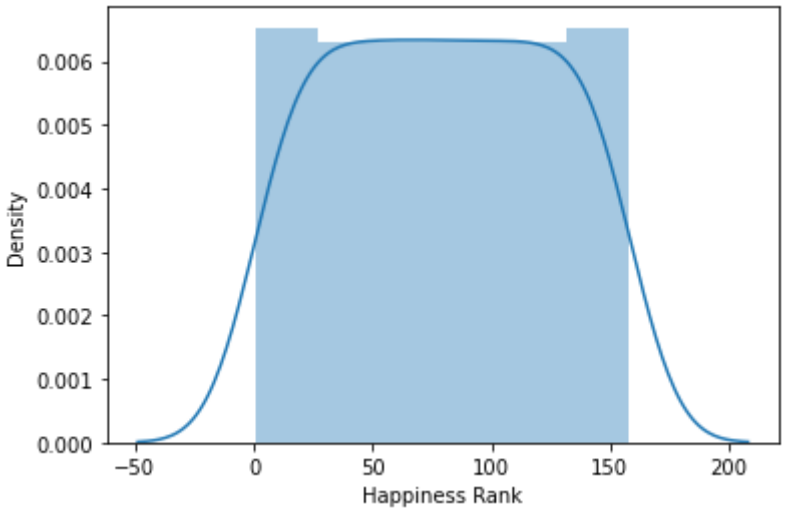
Out[32]:
```
<AxesSubplot:xlabel='Happiness Rank', ylabel='Density'>
```



In [33]:
```
sb.distplot(df["Happiness Rank"])
```

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

Out[33]:
```
<AxesSubplot:xlabel='Happiness Rank', ylabel='Density'>
```



In [34]:
```
df1=df[['Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
df1
```

Out[34]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Gen
0	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0
1	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0
2	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0
3	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0
4	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0
...	...	...	...	...	...	...	...	...	
153	154	3.465	0.03464	0.22208	0.77370	0.42864	0.59201	0.55191	0
154	155	3.340	0.03656	0.28665	0.35386	0.31910	0.48450	0.08010	0
155	156	3.006	0.05015	0.66320	0.47489	0.72193	0.15684	0.18906	0
156	157	2.905	0.08658	0.01530	0.41587	0.22396	0.11850	0.10062	0
157	158	2.839	0.06727	0.20868	0.13995	0.28443	0.36453	0.10731	0

158 rows × 10 columns

In [35]:
```
sb.heatmap(df1.corr())
```

Out[35]: <AxesSubplot:>

In [37]:
```
x = df1[['Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual']]
y = df1['Happiness Rank']
```

In [38]:
```
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)
```

In [39]:
```
from sklearn.linear_model import LinearRegression

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

Out[39]:
```
LinearRegression()
```

In [40]:
```
print(lr.intercept_)
```

```
1.7053025658242404e-13
```

In [41]:
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[41]:

	Co_efficient
Happiness Rank	1.000000e+00
Happiness Score	4.627458e-12
Standard Error	-9.626202e-14
Economy (GDP per Capita)	-4.644885e-12
Family	-4.640041e-12

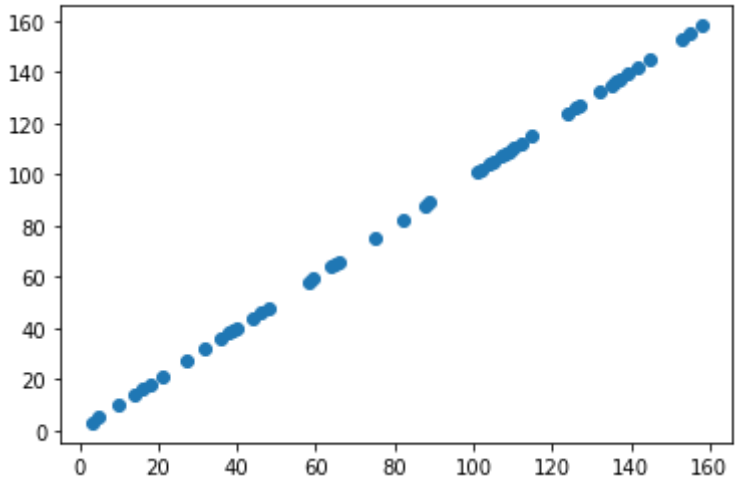
	Co_efficient
Health (Life Expectancy)	-4.665510e-12
Freedom	-4.657212e-12
Trust (Government Corruption)	-4.647434e-12
Generosity	-4.635445e-12
Dystopia Residual	-4.651994e-12

In [42]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [43]:
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[43]:
```
<matplotlib.collections.PathCollection at 0x19dbe00c790>
```



In [44]:
```
lr.score(x_test,y_test)
```

Out[44]:
```
1.0
```

In [45]:
```
lr.score(x_train,y_train)
```

Out[45]:
```
1.0
```

In [46]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [47]:
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[47]:
```
0.9999999928039329
```

In [48]:
```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
```

```
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[48]: 0.9999759199332721

In [ ]:

In [2]:

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

In [65]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\fitness1.csv")
df
```

Out[65]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62	0.08	0.06	75
1	B	4.21	0.17	0.19	160
2	C	9.83	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170

In [66]:

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

Out[66]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62	0.08	0.06	75
1	B	4.21	0.17	0.19	160
2	C	9.83	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170

In [67]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8 entries, 0 to 7
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Row Labels          8 non-null      object 
 1   Sum of Jan          8 non-null      float64
 2   Sum of Feb          8 non-null      float64
 3   Sum of Mar          8 non-null      float64
 4   Sum of Total Sales  8 non-null      int64  
dtypes: float64(3), int64(1), object(1)
memory usage: 448.0+ bytes
```

In [68]:

```
df.describe()
```

Out[68]:

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
count	8.000000	8.000000	8.000000	8.000000
mean	2.558750	0.126250	0.123750	143.750000
std	3.659299	0.053436	0.055275	38.384335
min	0.030000	0.060000	0.050000	75.000000
25%	0.162500	0.087500	0.075000	120.500000
50%	0.255000	0.115000	0.130000	163.500000
75%	4.562500	0.162500	0.172500	170.250000
max	9.830000	0.220000	0.190000	179.000000

In [69]:

```
df.columns
```

Out[69]:

```
Index(['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales'],
      dtype='object')
```

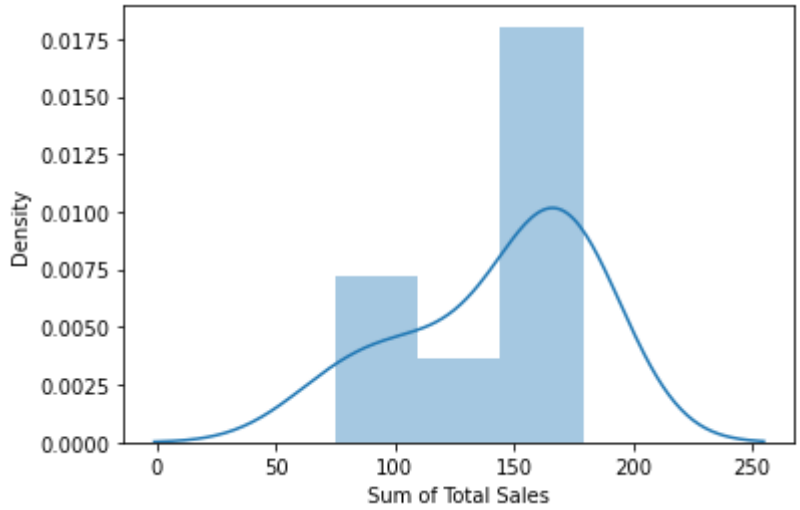
In [70]:

```
sb.distplot(df["Sum of Total Sales"])
```

```
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[70]:

```
<AxesSubplot:xlabel='Sum of Total Sales', ylabel='Density'>
```



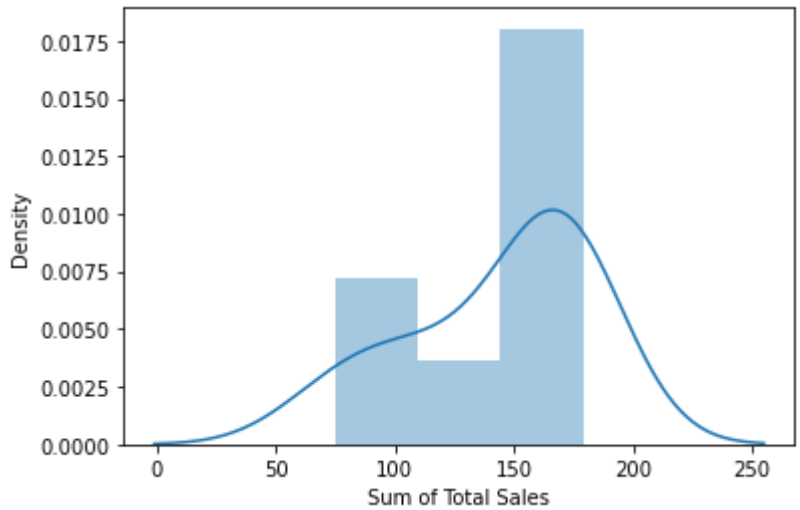
In [71]:

```
sb.distplot(df["Sum of Total Sales"])
```

```
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[71]:

```
<AxesSubplot:xlabel='Sum of Total Sales', ylabel='Density'>
```



In [72]:

```
df1=df[['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales']]
df1
```

Out[72]:

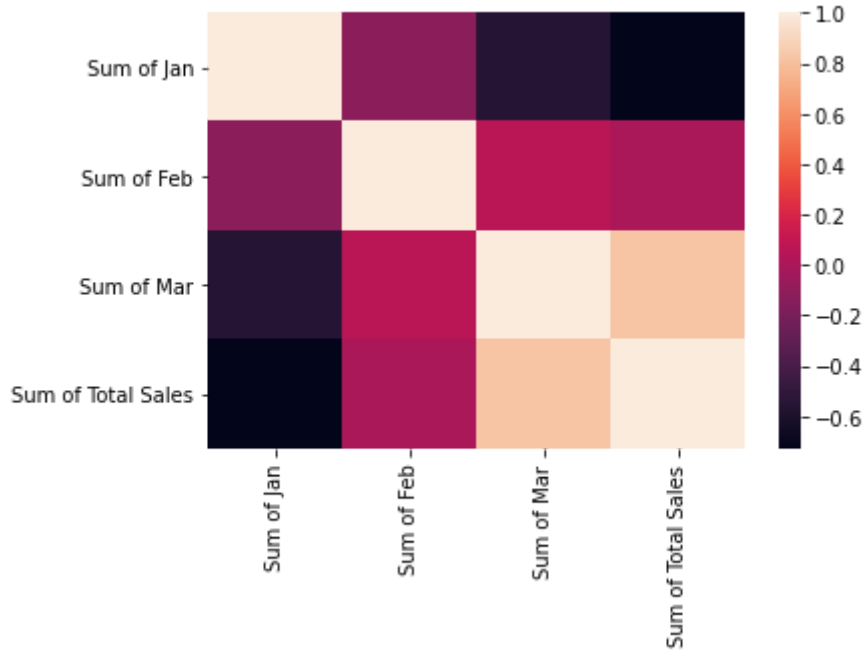
	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62	0.08	0.06	75
1	B	4.21	0.17	0.19	160
2	C	9.83	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170

In [73]:

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

Out[73]:

```
<AxesSubplot:>
```



In [74]:

```
x = df1[['Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales']]
y = df1['Sum of Total Sales']
```

In [75]:

```
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)
```

In [76]:

```
from sklearn.linear_model import LinearRegression

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

Out[76]:

```
LinearRegression()
```

In [77]:

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

```
-5.684341886080802e-14
```

In [78]:

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

Out[78]:

	Co_efficient
Sum of Jan	3.303895e-15
Sum of Feb	4.915735e-14
Sum of Mar	-8.827062e-14
Sum of Total Sales	1.000000e+00

In [79]:

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

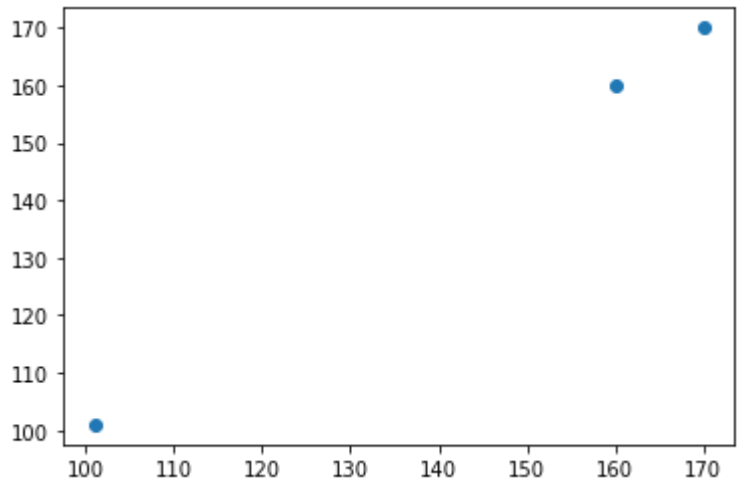
```
1.0
```

In [80]:

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

Out[80]:

```
<matplotlib.collections.PathCollection at 0x19dbf765400>
```



In [81]:

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

Out[81]:

```
1.0
```

In [82]:

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

Out[82]:

```
1.0
```

In [83]:

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

In [84]:

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

Out[84]:

```
0.9999975046658962
```

In [85]:

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

Out[85]:

```
0.9999559417609463
```

In [ ]:

```

```

In [87]:

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

In [88]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\4_drug200.csv")
df
```

Out[88]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

200 rows × 6 columns

In [89]:

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

Out[89]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	drugY
7	41	M	LOW	HIGH	11.037	drugC
8	60	M	NORMAL	HIGH	15.171	drugY
9	43	M	LOW	NORMAL	19.368	drugY

In [90]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Age          200 non-null    int64  
 1   Sex          200 non-null    object 
 2   BP           200 non-null    object 
 3   Cholesterol  200 non-null    object 
 4   Na_to_K      200 non-null    float64
 5   Drug         200 non-null    object 
dtypes: float64(1), int64(1), object(4)
memory usage: 9.5+ KB
```

In [91]:

```
df.describe()
```

Out[91]:

	Age	Na_to_K
count	200.000000	200.000000
mean	44.315000	16.084485
std	16.544315	7.223956
min	15.000000	6.269000
25%	31.000000	10.445500
50%	45.000000	13.936500
75%	58.000000	19.380000
max	74.000000	38.247000

In [92]:

```
df.columns
```

Out[92]:

```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='objec
t')
```

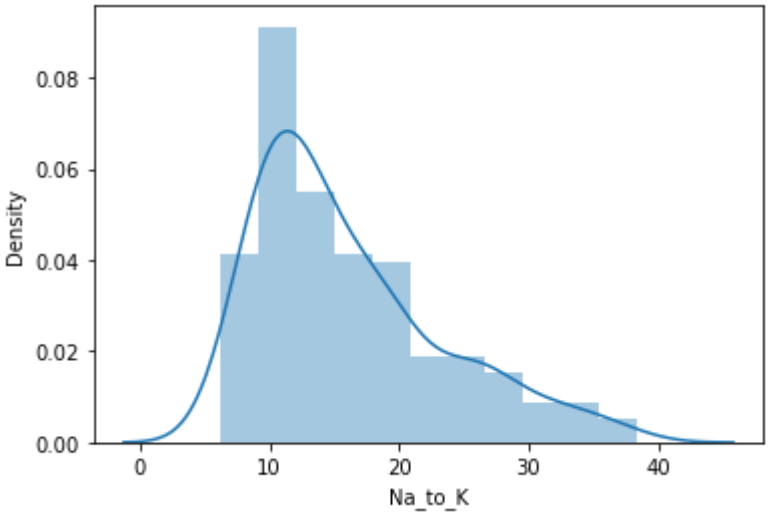
In [93]:

```
sb.distplot(df["Na_to_K"])
```

```
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[93]:

```
<AxesSubplot:xlabel='Na_to_K', ylabel='Density'>
```



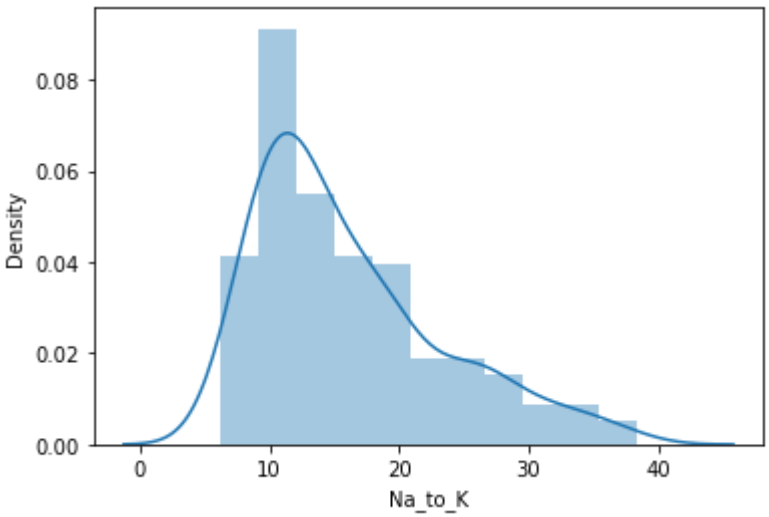
In [94]:

```
sb.distplot(df["Na_to_K"])
```

```
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[94]:

```
<AxesSubplot:xlabel='Na_to_K', ylabel='Density'>
```



In [95]:

```
df1=df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug']]
df1
```

Out[95]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
...	...	...	...	...	...	...
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

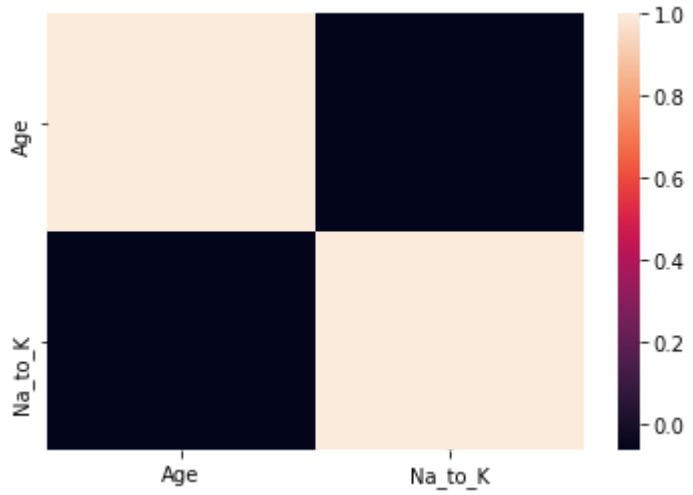
200 rows × 6 columns

In [96]:

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

Out[96]:

```
<AxesSubplot:>
```



In [97]:

```
x = df1[['Age', 'Na_to_K']]
y = df1['Na_to_K']
```

In [98]:

```
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)
```

In [99]:

```
from sklearn.linear_model import LinearRegression

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

Out[99]:

```
LinearRegression()
```

In [100]:

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

```
7.105427357601002e-15
```

In [101]:

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

Out[101]:

	Co_efficient
Age	0.0
Na_to_K	1.0

In [102]:

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

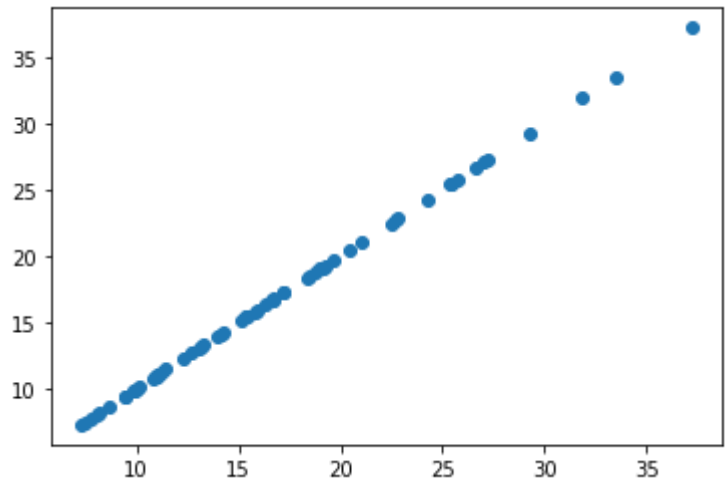
```
1.0
```

In [103]:

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

Out[103]:

```
<matplotlib.collections.PathCollection at 0x19dbe5592e0>
```



In [104]:

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

Out[104]:

```
1.0
```

In [105]:

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

Out[105]:

```
1.0
```

In [106]:

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

In [107]:

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

Out[107]:

```
0.9999981803498956
```

In [108]:

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

Out[108]:

```
0.9643476602209567
```

In [ ]:

```

```

In [109…

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

In [110…

```
df = pd.read_csv(r"C:\Users\user\Desktop\5_Instagram data.csv")
df
```

Out[110…

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follo
0	3920	2586	1028	619	56	98	9	5	162	35	
1	5394	2727	1838	1174	78	194	7	14	224	48	
2	4021	2085	1188	0	533	41	11	1	131	62	
3	4528	2700	621	932	73	172	10	7	213	23	
4	2518	1704	255	279	37	96	5	4	123	8	
...	...	...	...	...	...	...	...	...	...	...	
114	13700	5185	3041	5352	77	573	2	38	373	73	
115	5731	1923	1368	2266	65	135	4	1	148	20	
116	4139	1133	1538	1367	33	36	0	1	92	34	

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follo
117	32695	11815	3147	17414	170	1095	2	75	549	148	2
118	36919	13473	4176	16444	2547	653	5	26	443	611	2

119 rows × 13 columns

In [111…

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

Out[111…

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follows
0	3920	2586	1028	619	56	98	9	5	162	35	2
1	5394	2727	1838	1174	78	194	7	14	224	48	10
2	4021	2085	1188	0	533	41	11	1	131	62	12
3	4528	2700	621	932	73	172	10	7	213	23	8
4	2518	1704	255	279	37	96	5	4	123	8	0
5	3884	2046	1214	329	43	74	7	10	144	9	2

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follows
6	2621	1543	599	333	25	22	5	1	76	26	0
7	3541	2071	628	500	60	135	4	9	124	12	6
8	3749	2384	857	248	49	155	6	8	159	36	4
9	4115	2609	1104	178	46	122	6	3	191	31	6

In [112…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119 entries, 0 to 118
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Impressions     119 non-null    int64 
 1   From Home       119 non-null    int64 
 2   From Hashtags   119 non-null    int64 
 3   From Explore    119 non-null    int64 
 4   From Other      119 non-null    int64 
 5   Saves           119 non-null    int64 
 6   Comments        119 non-null    int64 
 7   Shares          119 non-null    int64 
 8   Likes           119 non-null    int64 
 9   Profile Visits  119 non-null    int64 
 10  Follows         119 non-null    int64 
 11  Caption         119 non-null    object
 12  Hashtags        119 non-null    object
dtypes: int64(11), object(2)
memory usage: 12.2+ KB
```

In [113…

```
df.describe()
```

Out[113…

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	6.663866
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	3.544576
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	0.000000

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	4.000000
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	6.000000
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	8.000000
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	19.000000

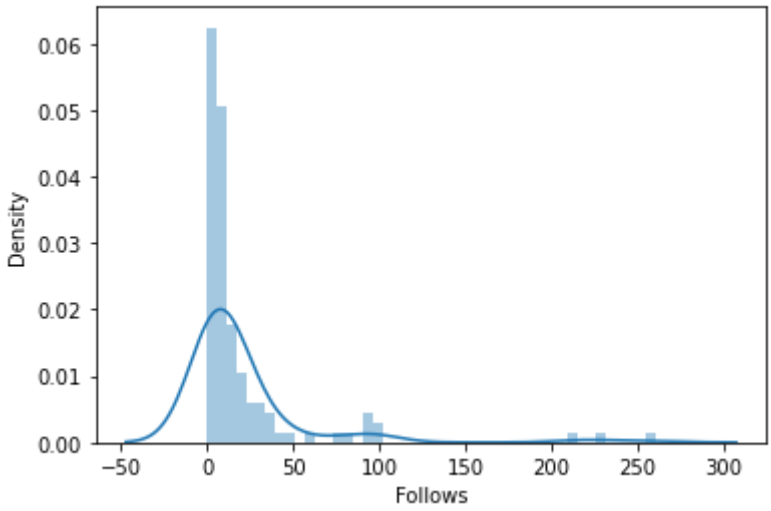
In [114...
```
df.columns
```

Out[114...
```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

In [115...
```
sb.distplot(df["Follows"])
```

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

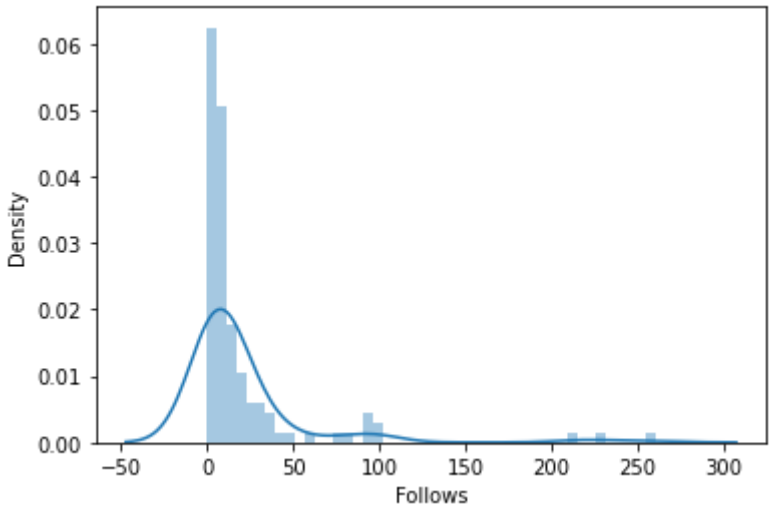
Out[115...
```
<AxesSubplot:xlabel='Follows', ylabel='Density'>
```



In [116...
```
sb.distplot(df["Follows"])
```

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

Out[116...
```
<AxesSubplot:xlabel='Follows', ylabel='Density'>
```



In [117…

```
df1=df[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows']]
df1
```

Out[117…

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follo
0	3920	2586	1028	619	56	98	9	5	162	35	
1	5394	2727	1838	1174	78	194	7	14	224	48	
2	4021	2085	1188	0	533	41	11	1	131	62	
3	4528	2700	621	932	73	172	10	7	213	23	
4	2518	1704	255	279	37	96	5	4	123	8	
…	…	…	…	…	…	…	…	…	…	…	
114	13700	5185	3041	5352	77	573	2	38	373	73	
115	5731	1923	1368	2266	65	135	4	1	148	20	
116	4139	1133	1538	1367	33	36	0	1	92	34	
117	32695	11815	3147	17414	170	1095	2	75	549	148	2
118	36919	13473	4176	16444	2547	653	5	26	443	611	2

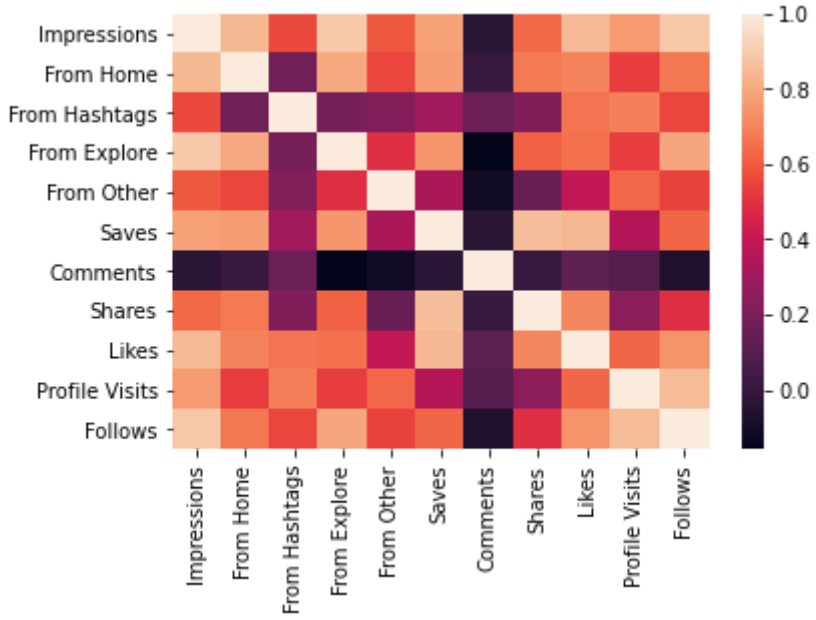
119 rows × 11 columns

In [118…

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

Out[118…

```
<AxesSubplot:>
```



In [119…
```
x = df1[['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows']]
y = df1['Follows']
```

In [120…
```
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)
```

In [121…
```
from sklearn.linear_model import LinearRegression

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

Out[121… LinearRegression()

In [122…
```
print(lr.intercept_)
```

-1.0658141036401503e-14

In [123…
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[123…

	Co_efficient
Impressions	-1.186783e-17
From Home	4.372055e-18
From Hashtags	1.495256e-17
From Explore	8.822022e-18
From Other	-8.054544e-17
Saves	6.530166e-17
Comments	2.958506e-16
Shares	2.599474e-16

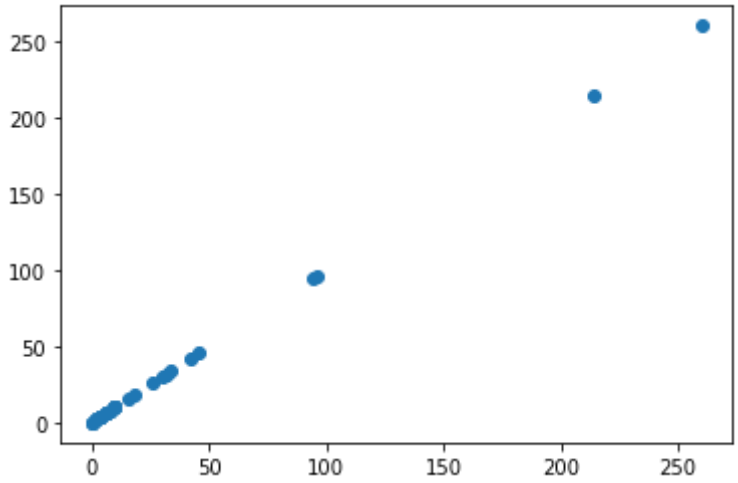
	Co_efficient
Likes	1.021599e-16
Profile Visits	2.396130e-16
Follows	1.000000e+00

In [124…
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [125…
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[125…
```
<matplotlib.collections.PathCollection at 0x19dc0000460>
```



In [126…
```
lr.score(x_test,y_test)
```

Out[126…
```
1.0
```

In [127…
```
lr.score(x_train,y_train)
```

Out[127…
```
1.0
```

In [128…
```
from sklearn.linear_model import Ridge,Lasso
```

In [129…
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[129…
```
0.9999997863886183
```

In [130…
```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[130… 0.9991340064271627

In [ ]:

In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
```

In [27]:
```
import seaborn as sb
```

In [37]:
```
df = pd.read_csv(r"C:\Users\user\Desktop\6_Salesworkload1.csv")
df
```

Out[37]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLeas
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.
...	...	...	...	...	...	...	...	...	.
7653	06.2017	9.0	Sweden	29650.0	Gothenburg	12.0	Checkout	6322.323	0.
7654	06.2017	9.0	Sweden	29650.0	Gothenburg	16.0	Customer Services	4270.479	0.
7655	06.2017	9.0	Sweden	29650.0	Gothenburg	11.0	Delivery	0	0.
7656	06.2017	9.0	Sweden	29650.0	Gothenburg	17.0	others	2224.929	0.
7657	06.2017	9.0	Sweden	29650.0	Gothenburg	18.0	all	39652.2	0.

7658 rows × 14 columns

In [38]:
```
df.head(10)
```

Out[38]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	1713
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.0	3107
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.0	213
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.0	54
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.0	59

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7658 entries, 0 to 7657
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   MonthYear      7658 non-null   object 
 1   Time index     7650 non-null   float64
 2   Country        7650 non-null   object 
 3   StoreID        7650 non-null   float64
 4   City           7650 non-null   object 
 5   Dept_ID        7650 non-null   float64
 6   Dept. Name     7650 non-null   object 
 7   HoursOwn       7650 non-null   object 
 8   HoursLease     7650 non-null   float64
 9   Sales units    7650 non-null   float64
 10  Turnover       7650 non-null   float64
 11  Customer       0 non-null      float64
 12  Area (m2)      7650 non-null   object 
 13  Opening hours  7650 non-null   object 
dtypes: float64(7), object(7)
memory usage: 837.7+ KB
```

In [40]:
```
df.describe()
```

Out[40]:

	Time index	StoreID	Dept_ID	HoursLease	Sales units	Turnover	Customer
count	7650.000000	7650.000000	7650.000000	7650.000000	7.650000e+03	7.650000e+03	0.0
mean	5.000000	61995.220000	9.470588	22.036078	1.076471e+06	3.721393e+06	NaN
std	2.582158	29924.581631	5.337429	133.299513	1.728113e+06	6.003380e+06	NaN
min	1.000000	12227.000000	1.000000	0.000000	0.000000e+00	0.000000e+00	NaN
25%	3.000000	29650.000000	5.000000	0.000000	5.457125e+04	2.726798e+05	NaN
50%	5.000000	75400.500000	9.000000	0.000000	2.932300e+05	9.319575e+05	NaN
75%	7.000000	87703.000000	14.000000	0.000000	9.175075e+05	3.264432e+06	NaN
max	9.000000	98422.000000	18.000000	3984.000000	1.124296e+07	4.271739e+07	NaN

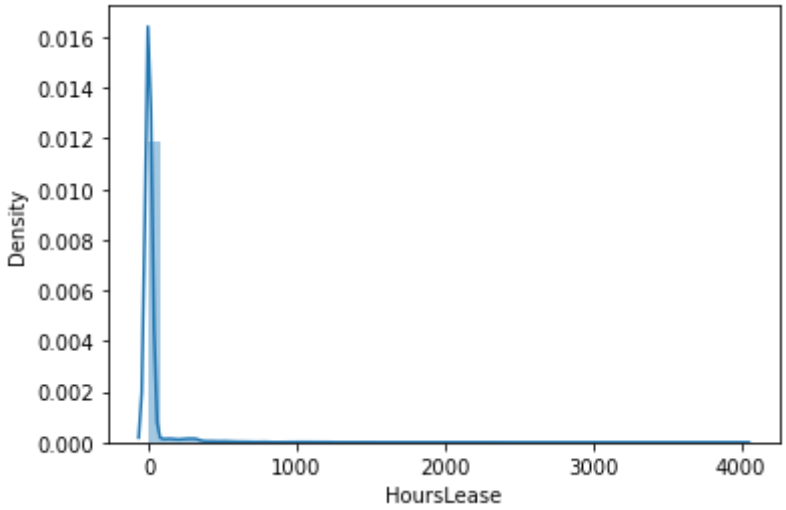
In [41]:
```
df.columns
```

Out[41]:
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [42]:
```
sb.distplot(df["HoursLease"])
```

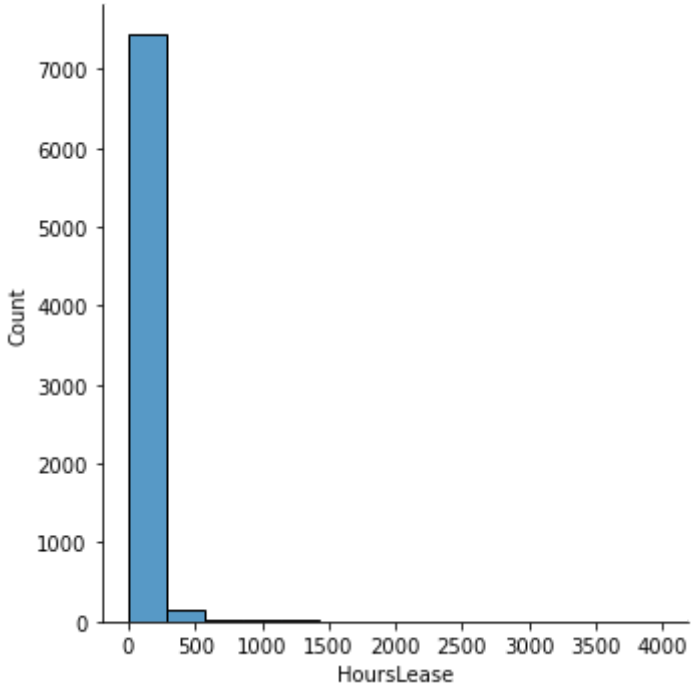
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function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

Out[42]:
```
<AxesSubplot:xlabel='HoursLease', ylabel='Density'>
```



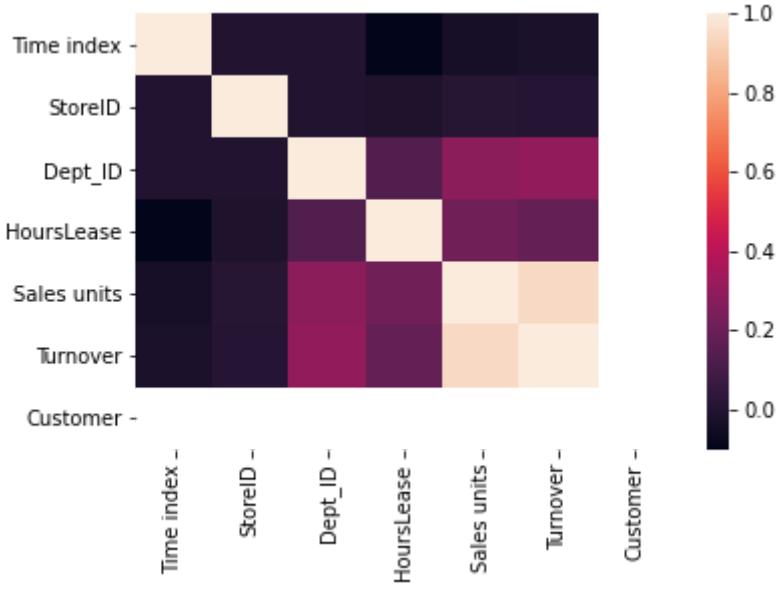
In [43]:
```
sb.displot(df["HoursLease"])
```

Out[43]:
```
<seaborn.axisgrid.FacetGrid at 0x21226ed6a90>
```



In [44]:
```
sb.heatmap(df.corr())
```

Out[44]: <AxesSubplot:>



In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
```

In [2]:
```
import seaborn as sb
```

In [92]:
```
df = pd.read_csv(r"C:\Users\user\Desktop\7_uber.csv")
df
```

Out[92]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latit
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744
...	...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720

200000 rows × 9 columns

In [93]:
```
df.head(10)
```

Out[93]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	d
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
5	44470845	2011-02-12 02:27:09.0000006	4.9	2011-02-12 02:27:09 UTC	-73.969019	40.755910	
6	48725865	2014-10-12 07:04:00.0000002	24.5	2014-10-12 07:04:00 UTC	-73.961447	40.693965	
7	44195482	2012-12-11 13:52:00.00000029	2.5	2012-12-11 13:52:00 UTC	0.000000	0.000000	
8	15822268	2012-02-17 09:32:00.00000043	9.7	2012-02-17 09:32:00 UTC	-73.975187	40.745767	
9	50611056	2012-03-29 19:06:00.000000273	12.5	2012-03-29 19:06:00 UTC	-74.001065	40.741787	

In [94]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [95]:
```
df.describe()
```

Out[95]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	199999.0
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	39.9
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	6.7
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	-881.9
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	40.7
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	40.7
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.963658	40.7
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	872.6

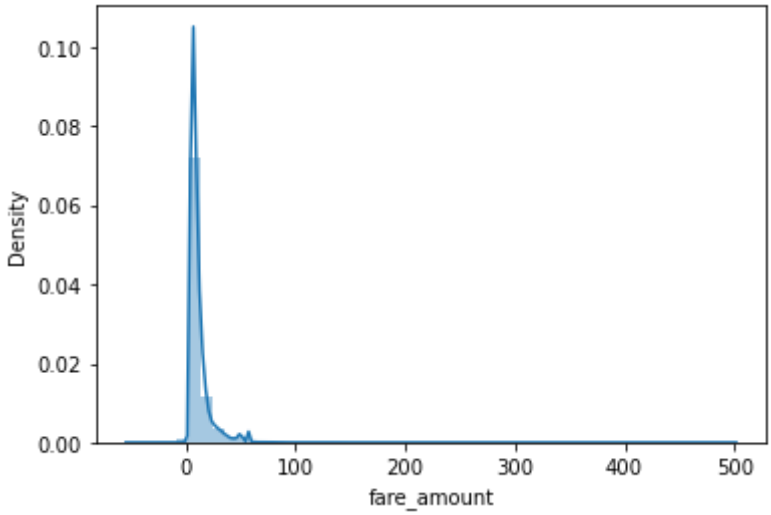
In [96]:
```
df.columns
```

Out[96]:
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [97]:
```
sb.distplot(df["fare_amount"])
```

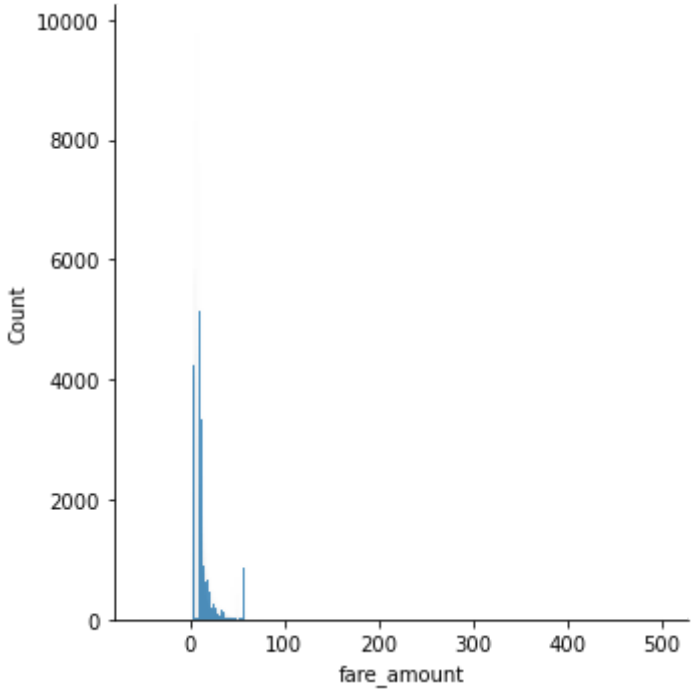
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function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

Out[97]:
```
<AxesSubplot:xlabel='fare_amount', ylabel='Density'>
```



In [99]:
```
sb.displot(df["fare_amount"])
```

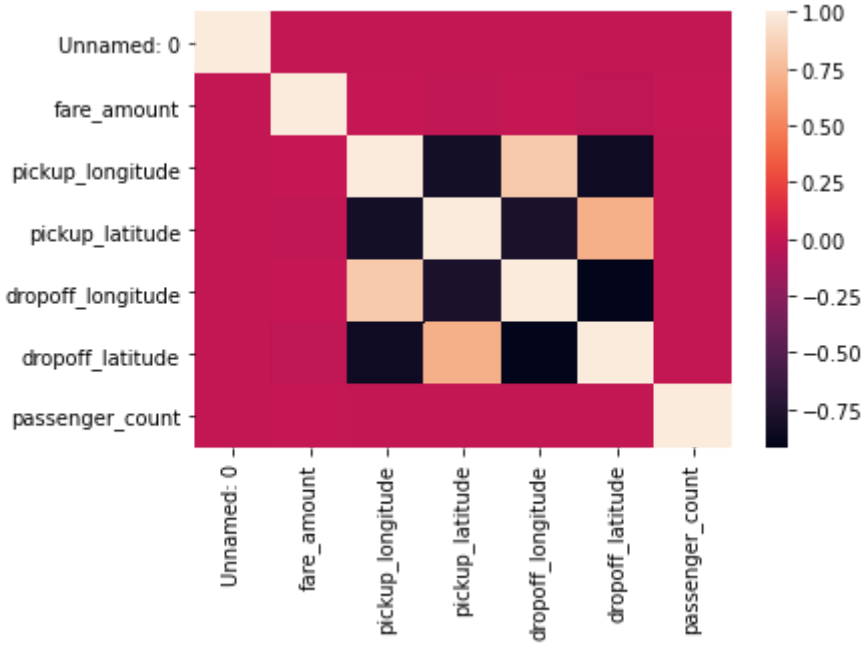
Out[99]:
```
<seaborn.axisgrid.FacetGrid at 0x23e60ee8af0>
```



In [100…
```
sb.heatmap(df.corr())
```

Out[100…

```
<AxesSubplot:>
```



In [104…

```
df = df.dropna()
x = df[['fare_amount', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
y = df[['passenger_count']]
```

In [105…

```
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)
```

In [106…

```
from sklearn.linear_model import LinearRegression

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

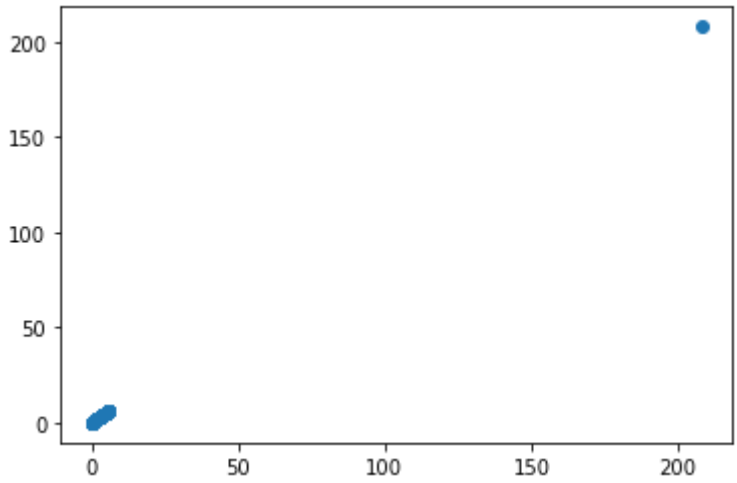
Out[106…

```
LinearRegression()
```

In [107…

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

Out[107… <matplotlib.collections.PathCollection at 0x23e61cafaf0>



In [ ]:

In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [20]:
```
df = pd.read_csv(r"C:\Users\user\Desktop\8_BreastCancerPrediction.csv")
df
```

Out[20]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.1184
1	842517	M	20.57	17.77	132.90	1326.0	0.0847
2	84300903	M	19.69	21.25	130.00	1203.0	0.1096
3	84348301	M	11.42	20.38	77.58	386.1	0.1425
4	84358402	M	20.29	14.34	135.10	1297.0	0.1003
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.1110
565	926682	M	20.13	28.25	131.20	1261.0	0.0978
566	926954	M	16.60	28.08	108.30	858.1	0.0845
567	927241	M	20.60	29.33	140.10	1265.0	0.1178
568	92751	B	7.76	24.54	47.92	181.0	0.0526

569 rows × 33 columns

In [21]:
```
df.head(10)
```

Out[21]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030
5	843786	M	12.45	15.70	82.57	477.1	0.12780
6	844359	M	18.25	19.98	119.60	1040.0	0.09463
7	84458202	M	13.71	20.83	90.20	577.9	0.11890
8	844981	M	13.00	21.82	87.50	519.8	0.12730
9	84501001	M	12.46	24.04	83.97	475.9	0.11860

10 rows × 33 columns

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [23]:
```
df.describe()
```

Out[23]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 32 columns

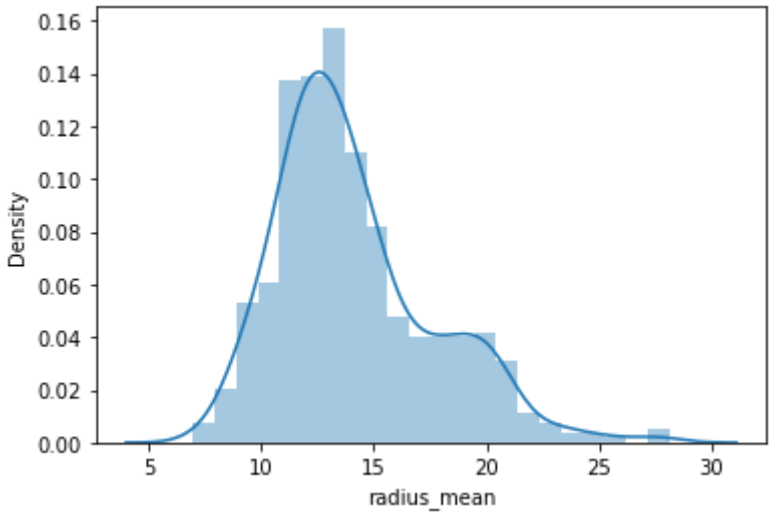
In [24]:
```
df.columns
```

Out[24]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

In [26]:
```
sb.distplot(df["radius_mean"])
```

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

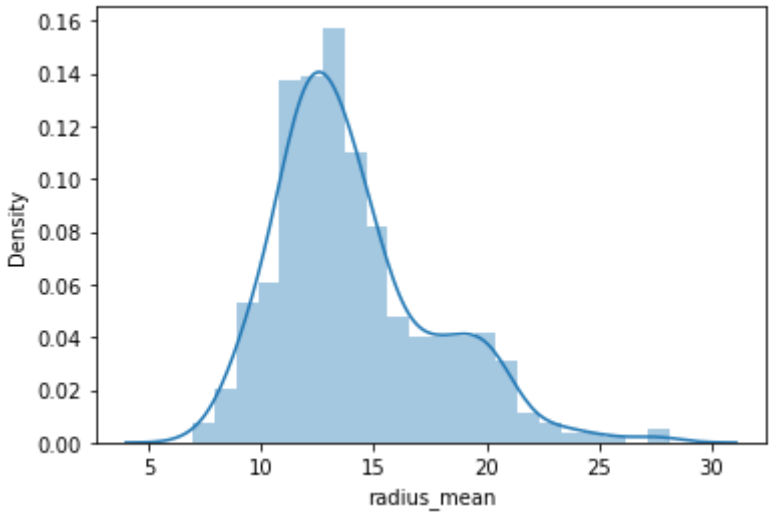
Out[26]: <AxesSubplot:xlabel='radius_mean', ylabel='Density'>



In [27]:
```
sb.distplot(df["radius_mean"])
```

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

Out[27]: <AxesSubplot:xlabel='radius_mean', ylabel='Density'>



In [28]:

```
df1=df[['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst']]
df1
```

Out[28]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.1184
1	842517	M	20.57	17.77	132.90	1326.0	0.0847
2	84300903	M	19.69	21.25	130.00	1203.0	0.1096
3	84348301	M	11.42	20.38	77.58	386.1	0.1425
4	84358402	M	20.29	14.34	135.10	1297.0	0.1003
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.1110
565	926682	M	20.13	28.25	131.20	1261.0	0.0978
566	926954	M	16.60	28.08	108.30	858.1	0.0845
567	927241	M	20.60	29.33	140.10	1265.0	0.1178
568	92751	B	7.76	24.54	47.92	181.0	0.0526

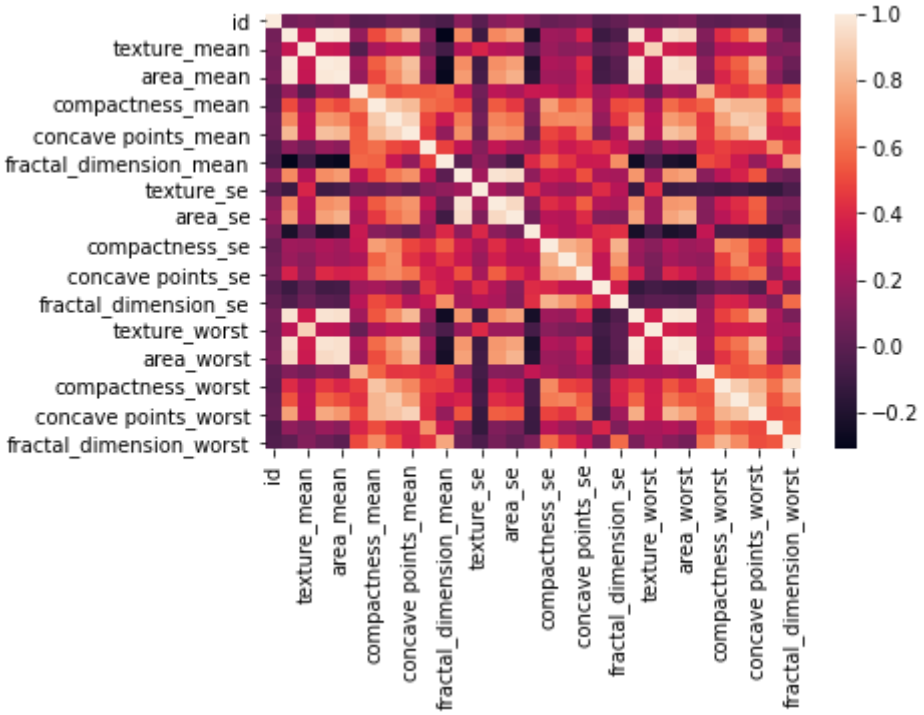
569 rows × 32 columns

In [29]:

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

Out[29]:

```
<AxesSubplot:>
```



In [37]:
```
x = df1[['id', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst']]
y = df1['radius_mean']
```

In [38]:
```
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)
```

In [39]:
```
from sklearn.linear_model import LinearRegression

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

Out[39]:
```
LinearRegression()
```

In [40]:
```
print(lr.intercept_)
```

```
4.121739607398922e-07
```

In [41]:
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[41]:

	Co_efficient
id	-1.238552e-16
radius_mean	9.999986e-01
texture_mean	1.953533e-08

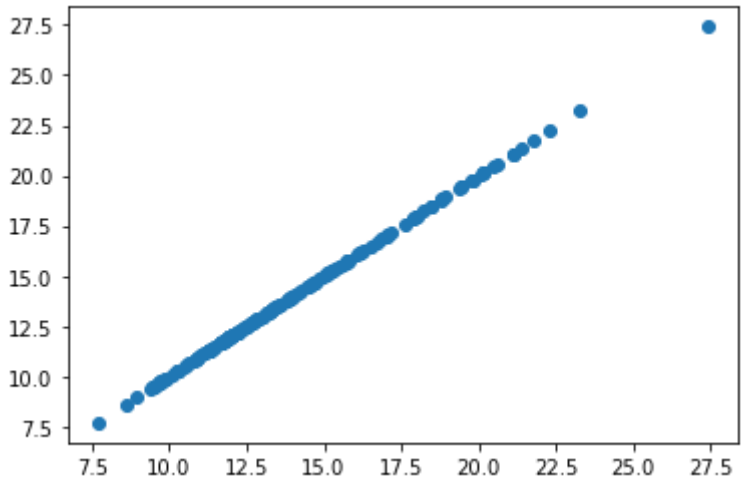
	Co_efficient
perimeter_mean	1.642178e-07
area_mean	1.614655e-09
smoothness_mean	-1.476209e-06
compactness_mean	-4.509532e-06
concavity_mean	1.591601e-07
concave points_mean	-2.171009e-06
symmetry_mean	2.533992e-07
fractal_dimension_mean	-9.843670e-07
radius_se	-3.741876e-06
texture_se	1.079828e-07
perimeter_se	3.331722e-07
area_se	6.785799e-09
smoothness_se	2.620204e-06
compactness_se	2.466441e-06
concavity_se	1.344952e-06
concave points_se	1.115306e-06
symmetry_se	-8.022986e-07
fractal_dimension_se	5.221630e-07
radius_worst	7.094790e-07
texture_worst	-2.635117e-08
perimeter_worst	-4.744591e-08
area_worst	-1.923128e-09
smoothness_worst	4.961837e-06
compactness_worst	6.238645e-08
concavity_worst	3.696179e-07
concave points_worst	-2.075276e-06
symmetry_worst	-2.832344e-06
fractal_dimension_worst	1.729066e-06

In [42]:
```
print(lr.score(x_test,y_test))
```

```
0.9999999999999958
```

In [43]:
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[43]:
```
<matplotlib.collections.PathCollection at 0x23a91fbf6a0>
```



In [44]:
```
lr.score(x_test,y_test)
```

Out[44]: 0.9999999999999958

In [45]:
```
lr.score(x_train,y_train)
```

Out[45]: 0.9999999999999958

In [46]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [47]:
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:147: LinAl
gWarning: Ill-conditioned matrix (rcond=1.18865e-18): result may not be accurate.
  return linalg.solve(A, Xy, sym_pos=True,
```

Out[47]: 0.9996637753114764

In [48]:
```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[48]: 0.9759917605421127

In [ ]:
```

```

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\10_USA_Housing.csv")
df
```

Out[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabet Stravenue\nDanieltow WI 06482
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP AE 0938
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFP AP 30153-765
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Bc 8489\nAPO AA 4299 335
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garde Suite 076\nJoshualan VA 01
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO A 7331
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridge Apt. 509\nEast Holl NV 2

5000 rows × 7 columns

first 10 rows

In [4]:
```
df.head(10)
```

Out[4]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
5	80175.754159	4.988408	6.104512	4.04	26748.428425	1.068138e+06	06039 Jennifer Islands Apt. 443\nTracyport, KS...
6	64698.463428	6.025336	8.147760	3.41	60828.249085	1.502056e+06	4759 Daniel Shoals Suite 442\nNguyenburgh, CO ...
7	78394.339278	6.989780	6.620478	2.42	36516.358972	1.573937e+06	972 Joyce Viaduct\nLake William, TN 17778-6483
8	59927.660813	5.362126	6.393121	2.30	29387.396003	7.988695e+05	USS Gilbert\nFPO AA 20957
9	81885.927184	4.423672	8.167688	6.10	40149.965749	1.545155e+06	Unit 9446 Box 0958\nDPO AE 97025

data cleaning

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Avg. Area Income             5000 non-null   float64
 1   Avg. Area House Age          5000 non-null   float64
```

```
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [7]:
```
df.describe()
```

Out[7]:

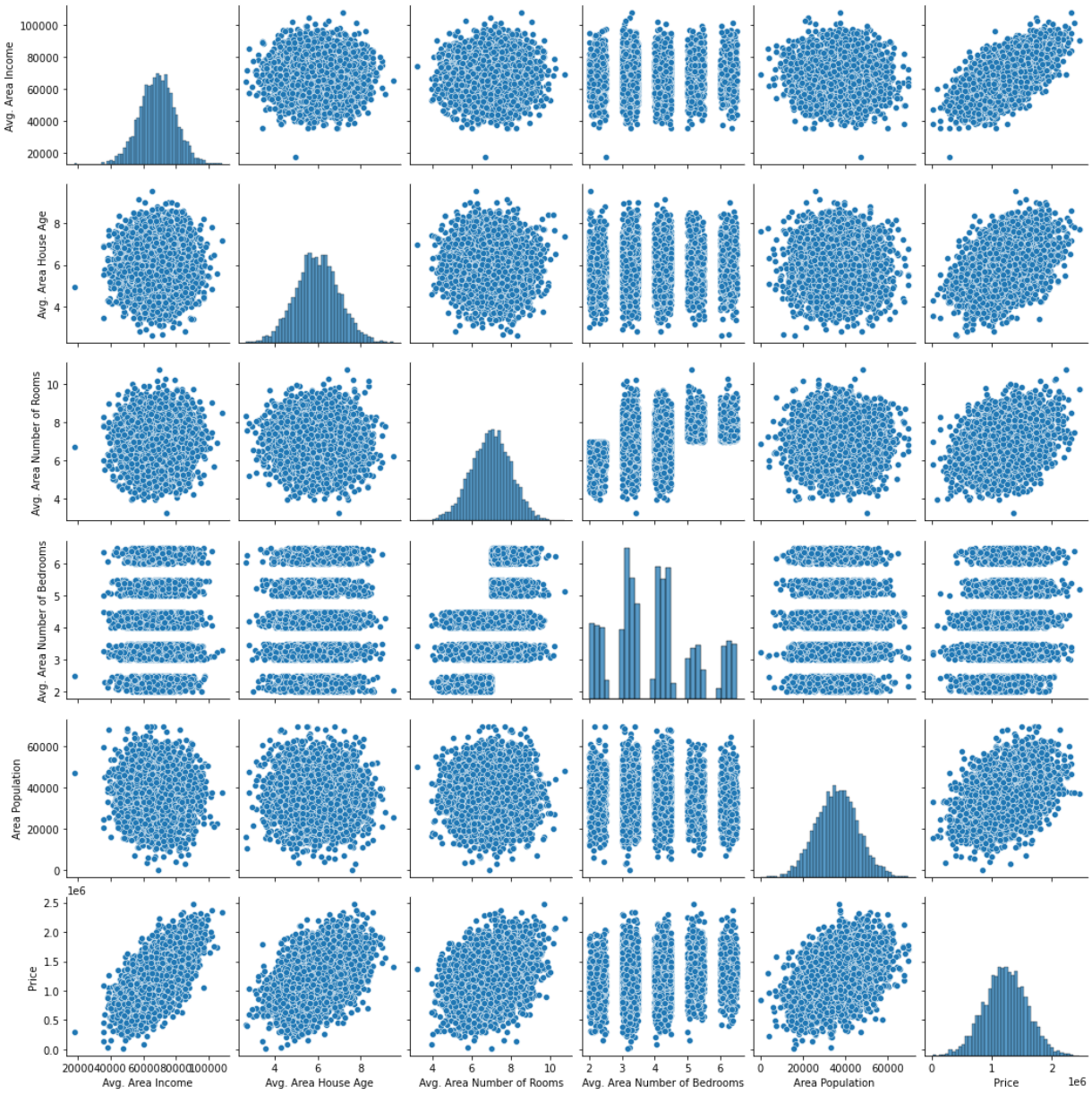
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

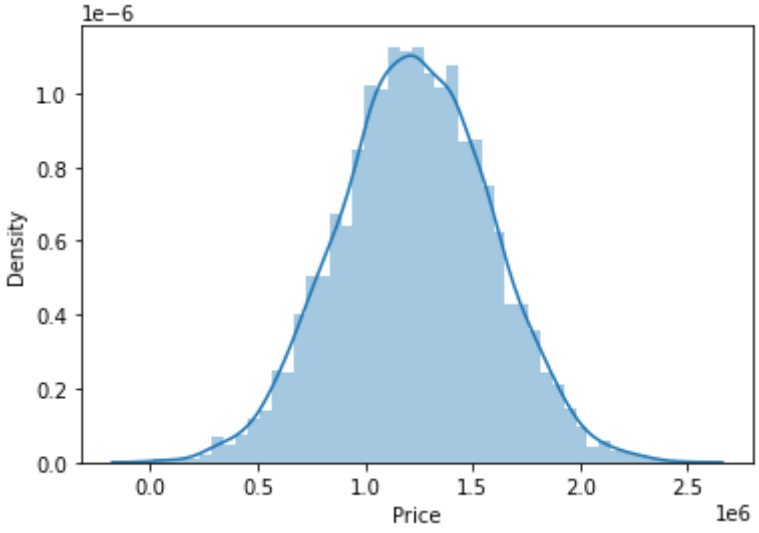
In [10]:
```
sb.pairplot(df)
```

Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x2590edcc580>
```



In [16]:
```
sb.distplot(df["Price"])
```

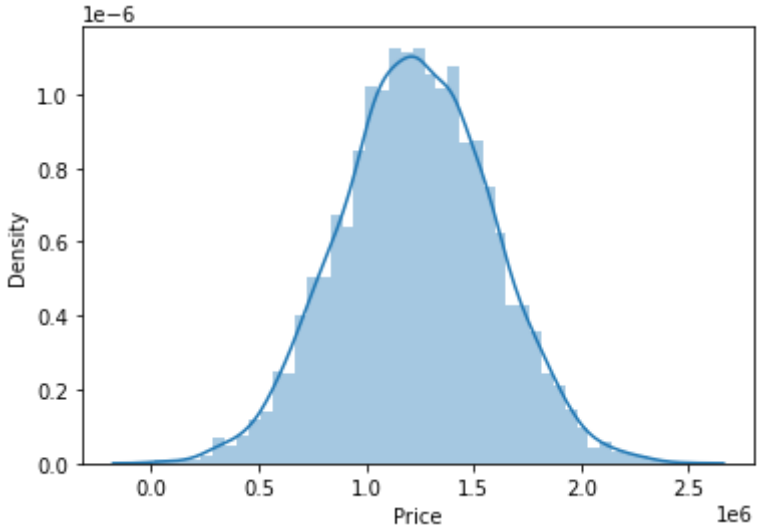
Out[16]: <AxesSubplot:xlabel='Price', ylabel='Density'>



In [17]:
```
sb.distplot(df["Price"])
```

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

Out[17]: <AxesSubplot:xlabel='Price', ylabel='Density'>



In [20]:
```
df1=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address']]
df1
```

Out[20]:

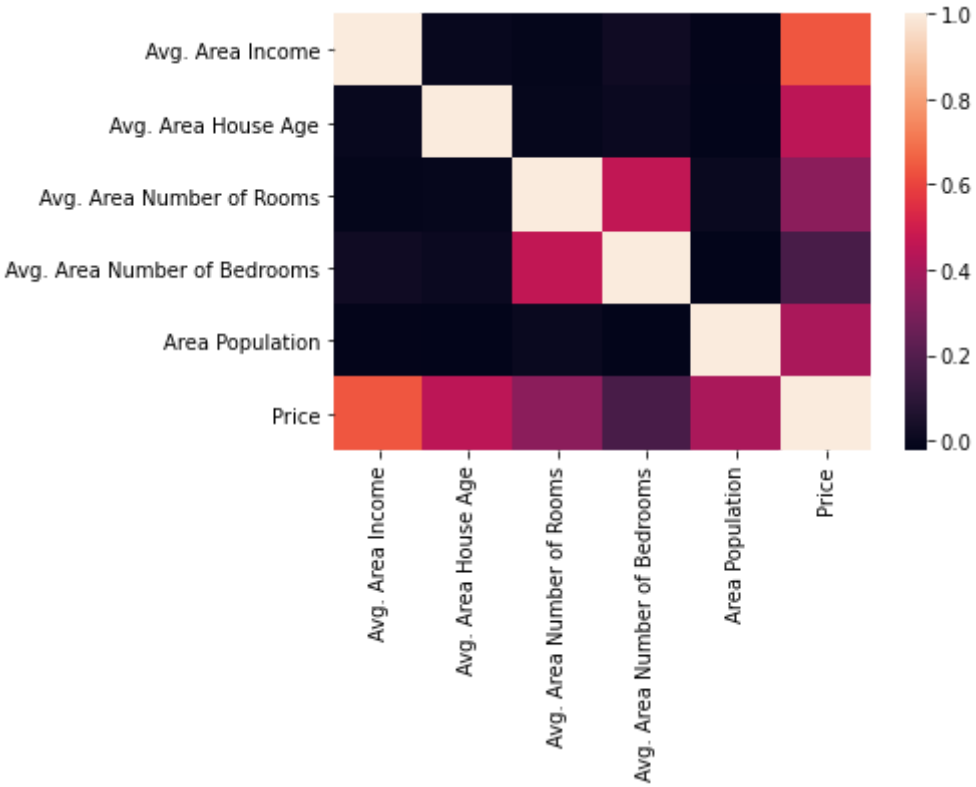
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabet Stravenue\nDanieltow WI 06482
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP AE 0938
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFP AP 30153-765
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Bc 8489\nAPO AA 4299 335
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garde Suite 076\nJoshualan VA 01

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO A 7331
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridge Apt. 509\nEast Holl NV 2

5000 rows × 7 columns

In [21]:
```
sb.heatmap(df1.corr())
```

Out[21]: <AxesSubplot:>



model building

In [34]:
```
x = df1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
y = df1['Price']
```

In [35]:
```
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)
```

In [36]:
```
from sklearn.linear_model import LinearRegression
```

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

Out[36]:
```
LinearRegression()
```

In [38]:
```
print(lr.intercept_)
```

```
-2.3283064365386963e-10
```

In [42]:
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[42]:

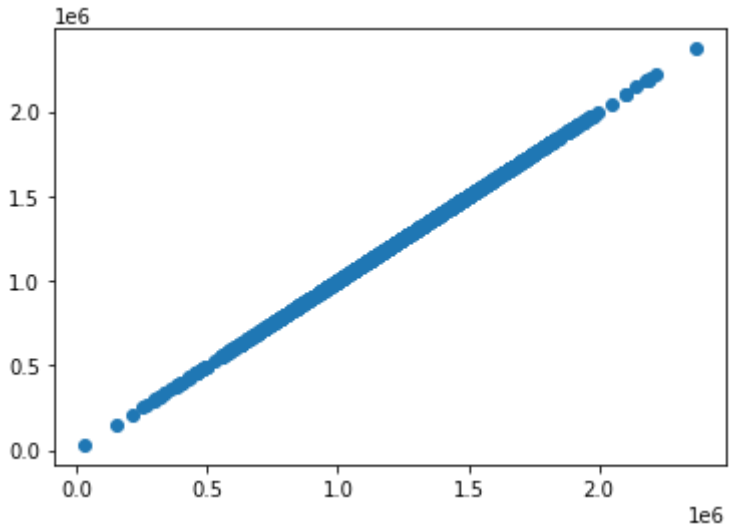
	Co_efficient
Avg. Area Income	3.184052e-15
Avg. Area House Age	-5.061788e-11
Avg. Area Number of Rooms	7.996921e-11
Avg. Area Number of Bedrooms	1.311991e-12
Area Population	9.261990e-15
Price	1.000000e+00

In [43]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [44]:
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[44]:
```
<matplotlib.collections.PathCollection at 0x25913950f40>
```



In [ ]:
```

```

In [50]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [83]:
```
df = pd.read_csv(r"C:\Users\user\Desktop\10_USA_Housing.csv")
df
```

Out[83]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabet Stravenue\nDanieltow WI 06482
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP AE 0938
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFP AP 30153-765
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Bo 8489\nAPO AA 4299 335
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garde Suite 076\nJoshualan VA 01
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO A 7331
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridge Apt. 509\nEast Holl NV 2

5000 rows × 7 columns

In [84]:
```
df.head(10)
```

Out[84]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
5	80175.754159	4.988408	6.104512	4.04	26748.428425	1.068138e+06	06039 Jennifer Islands Apt. 443\nTracyport, KS...
6	64698.463428	6.025336	8.147760	3.41	60828.249085	1.502056e+06	4759 Daniel Shoals Suite 442\nNguyenburgh, CO ...
7	78394.339278	6.989780	6.620478	2.42	36516.358972	1.573937e+06	972 Joyce Viaduct\nLake William, TN 17778-6483
8	59927.660813	5.362126	6.393121	2.30	29387.396003	7.988695e+05	USS Gilbert\nFPO AA 20957
9	81885.927184	4.423672	8.167688	6.10	40149.965749	1.545155e+06	Unit 9446 Box 0958\nDPO AE 97025

In [85]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Avg. Area Income             5000 non-null   float64
 1   Avg. Area House Age          5000 non-null   float64
 2   Avg. Area Number of Rooms    5000 non-null   float64
 3   Avg. Area Number of Bedrooms 5000 non-null   float64
 4   Area Population              5000 non-null   float64
 5   Price                        5000 non-null   float64
 6   Address                      5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [86]:

```
df.describe()
```

Out[86]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

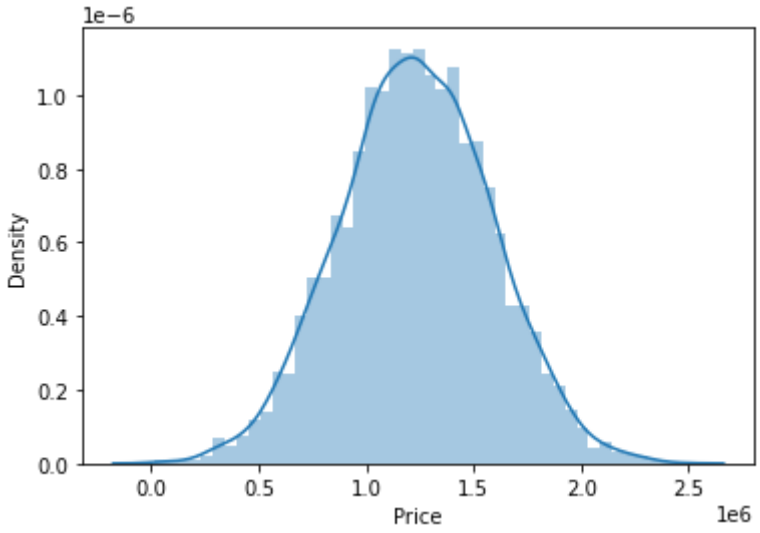
In [87]:
```
df.columns
```

Out[87]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [88]:
```
sb.distplot(df["Price"])
```

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

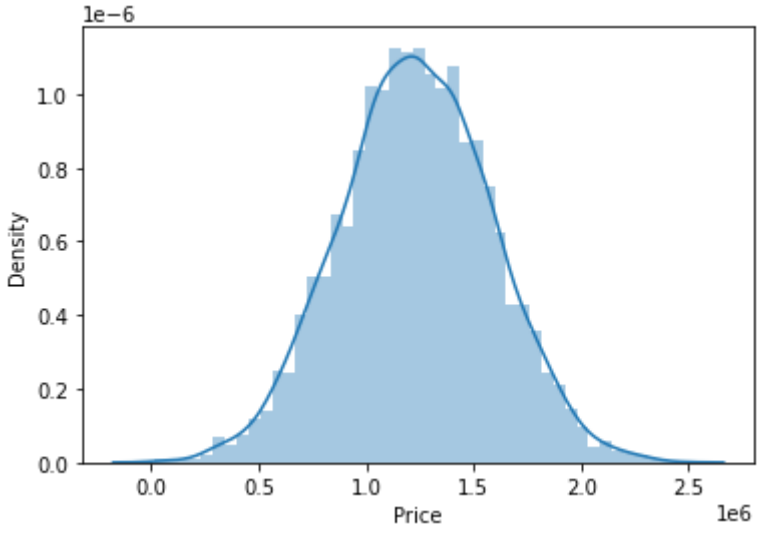
Out[88]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [89]:
```
sb.distplot(df["Price"])
```

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

Out[89]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [90]:
```
df1=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address']]
df1
```

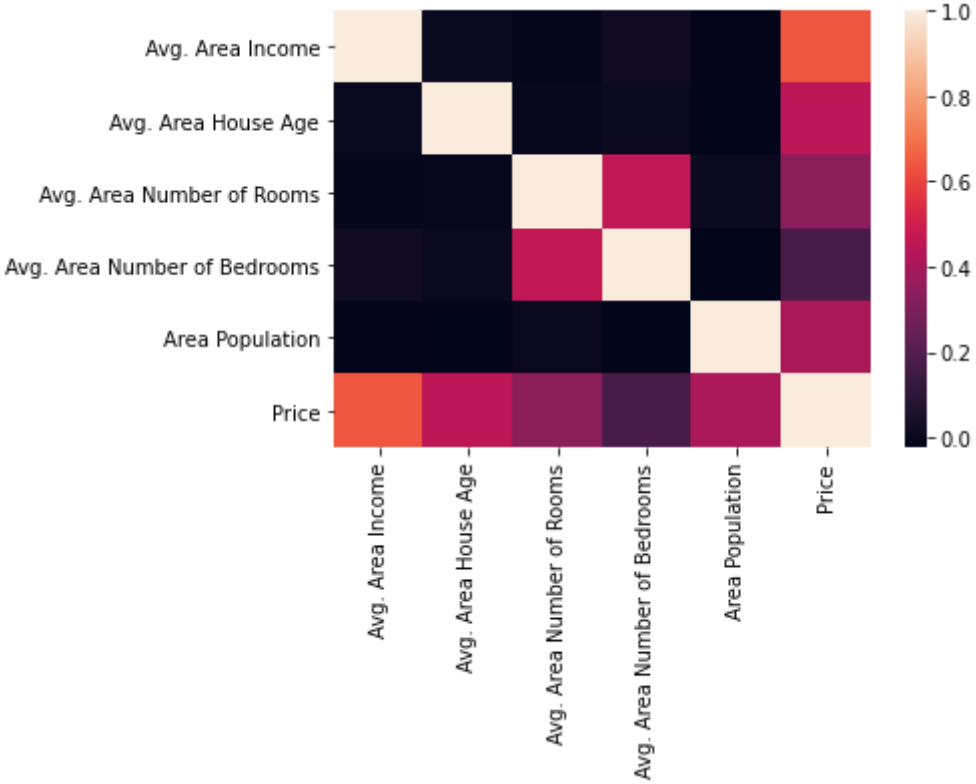
Out[90]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabet Stravenue\nDanieltow WI 06482
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFP AE 0938
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFP AP 30153-765
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Bc 8489\nAPO AA 4299 335
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garde Suite 076\nJoshualan VA 01
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO A 7331
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridge Apt. 509\nEast Holl NV 2

5000 rows × 7 columns

In [91]:
```
sb.heatmap(df1.corr())
```

Out[91]: <AxesSubplot:>



In [97]:
```
x = df1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
y = df1['Price']
```

In [98]:
```
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)
```

In [99]:
```
from sklearn.linear_model import LinearRegression

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

Out[99]: LinearRegression()

In [100…
```
print(lr.intercept_)
```

-1.6298145055770874e-09

In [101…
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[101…

	Co_efficient
Avg. Area Income	1.071616e-14

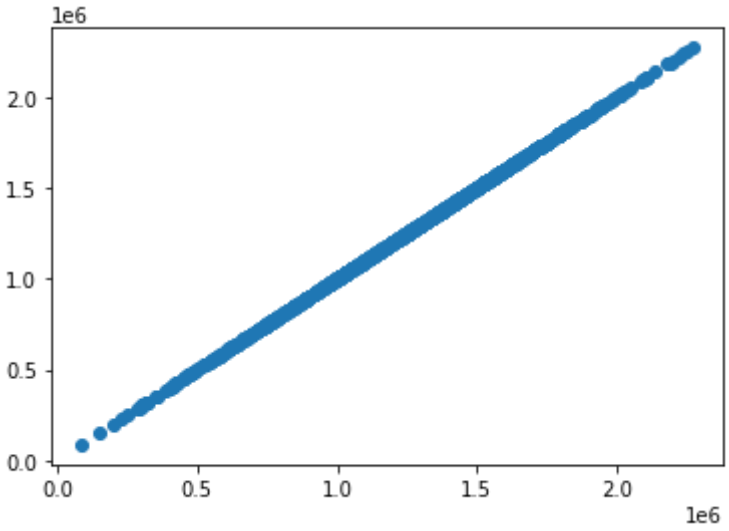
	Co_efficient
Avg. Area House Age	2.490720e-11
Avg. Area Number of Rooms	1.703335e-11
Avg. Area Number of Bedrooms	1.311203e-12
Area Population	-2.237141e-15
Price	1.000000e+00

In [102…
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [103…
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[103… `<matplotlib.collections.PathCollection at 0x23a9a493eb0>`



In [104…
```
lr.score(x_test,y_test)
```

Out[104… `1.0`

In [105…
```
lr.score(x_train,y_train)
```

Out[105… `1.0`

In [106…
```
from sklearn.linear_model import Ridge,Lasso
```

In [107…
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[107… `1.0`

In [108…

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

Out[108…

```
0.999999999991702
```

In [ ]:

In [50]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [109…
```
df = pd.read_csv(r"C:\Users\user\Desktop\11_winequality-red.csv")
df
```

Out[109…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
…	…	…	…	…	…	…	…	…	…	…	…
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 12 columns

In [110…
```
df.head(10)
```

Out[110…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qu
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
5	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.9978	3.51	0.56	9.4	
6	7.9	0.60	0.06	1.6	0.069	15.0	59.0	0.9964	3.30	0.46	9.4	
7	7.3	0.65	0.00	1.2	0.065	15.0	21.0	0.9946	3.39	0.47	10.0	
8	7.8	0.58	0.02	2.0	0.073	9.0	18.0	0.9968	3.36	0.57	9.5	

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	qu
9	7.5	0.50	0.36	6.1	0.071	17.0	102.0	0.9978	3.35	0.80	10.5	

In [111…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

In [112…

```
df.describe()
```

Out[112…

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	1
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	

In [113…

```
df.columns
```

Out[113…

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

In [114…

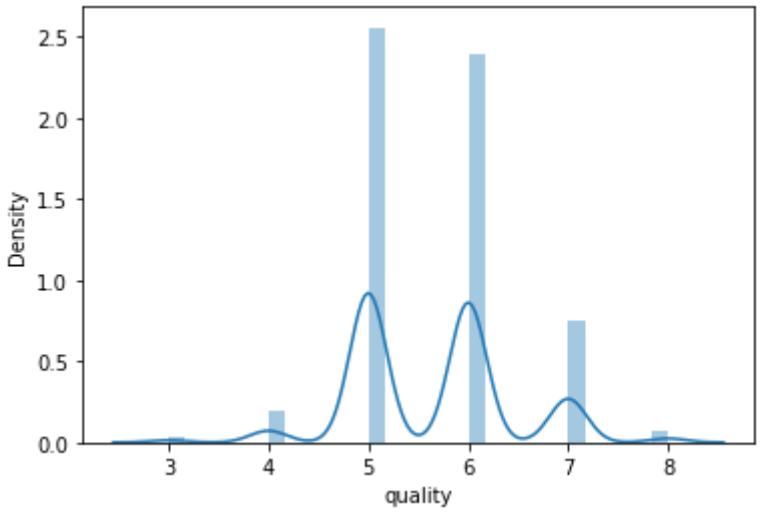
```
sb.distplot(df["quality"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarn
ing: `distplot` is a deprecated function and will be removed in a future version. Pl
```

```
ease adapt your code to use either `displot` (a figure-level function with similar f
lexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[114…

```
<AxesSubplot:xlabel='quality', ylabel='Density'>
```



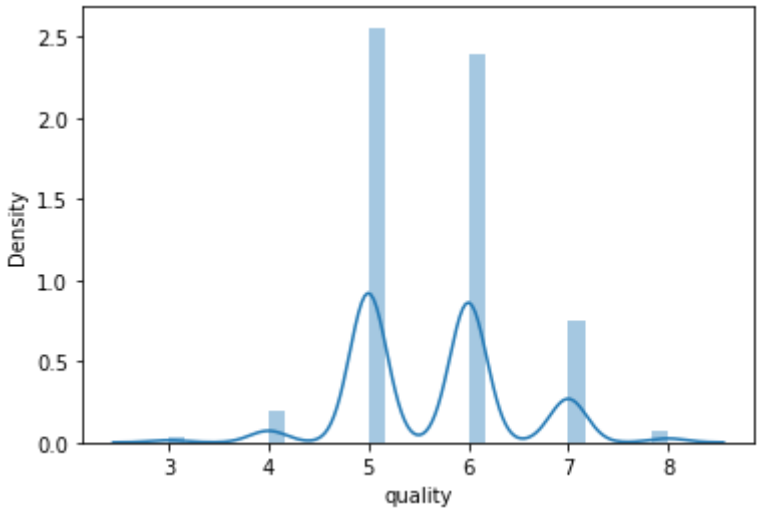
In [115…

```
sb.distplot(df["quality"])
```

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

Out[115…

```
<AxesSubplot:xlabel='quality', ylabel='Density'>
```



In [118…

```
df1=df[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
df1
```

Out[118…

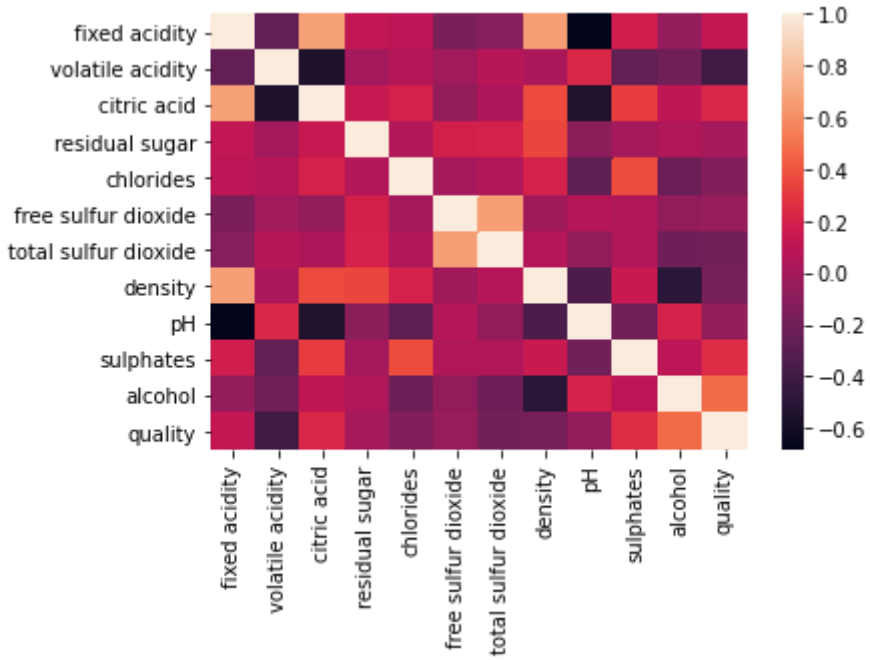
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4
...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0

1599 rows × 12 columns

In [119…
```
sb.heatmap(df1.corr())
```

Out[119… <AxesSubplot:>



In [120…
```
x = df1[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality']]
y = df1['quality']
```

In [121…
```
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)
```

In [122…
```
from sklearn.linear_model import LinearRegression

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

Out[122...
```
LinearRegression()
```

In [123...
```
print(lr.intercept_)
```

```
6.483702463810914e-14
```

In [124...
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[124...

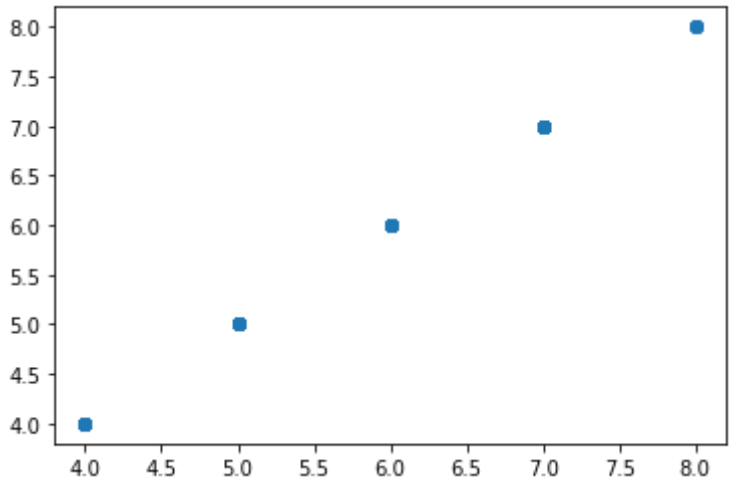
	Co_efficient
fixed acidity	-2.557601e-17
volatile acidity	1.511492e-15
citric acid	3.545944e-15
residual sugar	-9.123563e-16
chlorides	7.954054e-16
free sulfur dioxide	-9.665341e-17
total sulfur dioxide	1.862384e-17
density	-6.100842e-14
pH	-1.848207e-16
sulphates	1.996323e-16
alcohol	-5.935946e-16
quality	1.000000e+00

In [125...
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [126...
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[126...
```
<matplotlib.collections.PathCollection at 0x23a9a754520>
```



In [127...
```
lr.score(x_test,y_test)
```

Out[127…
```
1.0
```

In [128…
```
lr.score(x_train,y_train)
```

Out[128…
```
1.0
```

In [129…
```
from sklearn.linear_model import Ridge,Lasso
```

In [130…
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[130…
```
0.9997472380114248
```

In [131…
```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[131…
```
0.0
```

In [ ]:

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
```

In [2]:
```
df = pd.read_csv(r"C:\Users\user\Desktop\12_mobile_prices_2023.csv")
df
```

Out[2]:

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	P in
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
...	...	...	...	...	...	...	...	...	...	
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	₹14
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	₹14
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	₹14

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	P in
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	₹18
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	₹10

1836 rows × 11 columns

In [3]:
```
df.head(10)
```

Out[3]:

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	Price in INR
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...	₹5,649
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	₹11,999
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	₹6,999
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	₹7,749
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	₹6,999
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	₹11,999
6	POCO C55 (Power	4.2	22621.0	4 GB RAM	64 GB ROM	50MP Dual Rear Camera	5MP Front Camera	5000 mAh	Mediatek Helio G85 Processor	₹7,749

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	Price in INR
	Black, 64 GB)									
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	₹7,749
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	₹9,249
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	₹13,999

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

	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

	Rating ?/5	Number of Ratings
max	4.800000	1.342530e+06

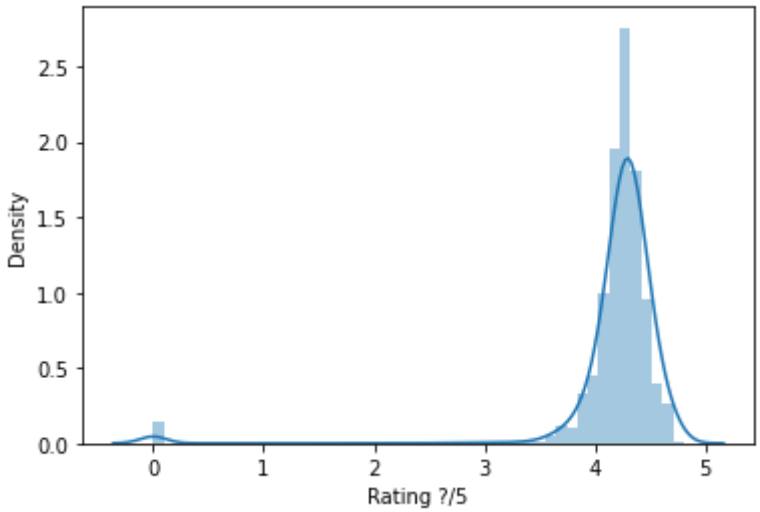
In [6]:
```
df.columns
```

Out[6]:
```
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')
```

In [7]:
```
sb.distplot(df["Rating ?/5"])
```

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

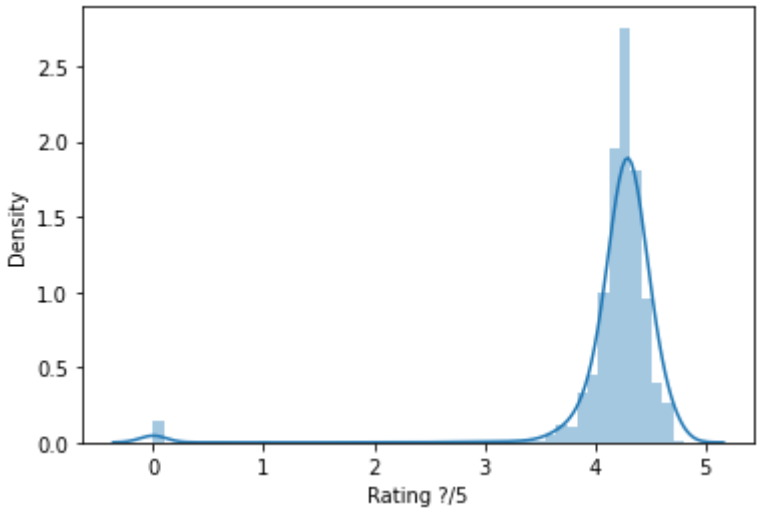
Out[7]:
```
<AxesSubplot:xlabel='Rating ?/5', ylabel='Density'>
```



In [8]:
```
sb.distplot(df["Rating ?/5"])
```

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

Out[8]:
```
<AxesSubplot:xlabel='Rating ?/5', ylabel='Density'>
```



In [9]:

```
df1=df[['Phone Name', 'Rating ?/5', 'Number of Ratings', 'RAM', 'ROM/Storage',
       'Back/Rare Camera', 'Front Camera', 'Battery', 'Processor',
       'Price in INR', 'Date of Scraping']]
df1
```

Out[9]:

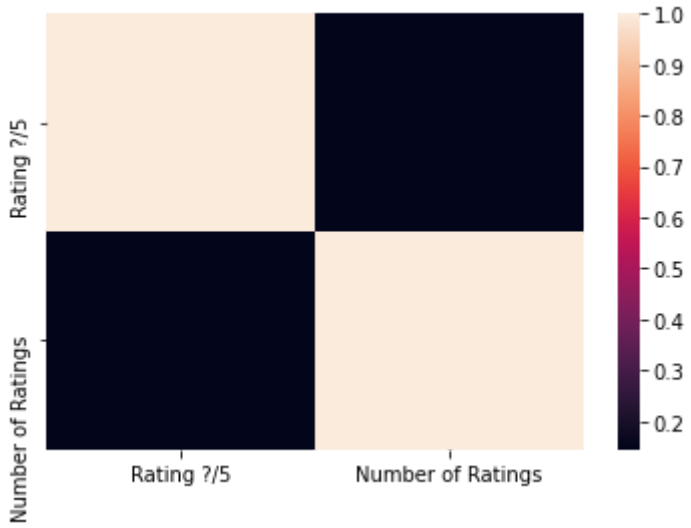
	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	P in
0	POCO C50 (Royal Blue, 32 GB)	4.2	33561.0	2 GB RAM	32 GB ROM	8MP Dual Camera	5MP Front Camera	5000 mAh	Mediatek Helio A22 Processor, Upto 2.0 GHz Pro...	₹5
1	POCO M4 5G (Cool Blue, 64 GB)	4.2	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
...	...	...	...	...	...	...	...	...	...	
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	₹14

	Phone Name	Rating ?/5	Number of Ratings	RAM	ROM/Storage	Back/Rare Camera	Front Camera	Battery	Processor	P in
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	₹14
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	₹14
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	₹18
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	₹10

1836 rows × 11 columns

In [10]:
```
sb.heatmap(df1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
x = df1[['Rating ?/5', 'Number of Ratings']]
y = df1['Rating ?/5']
```

In [12]:
```
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)
```

In [13]:
```
from sklearn.linear_model import LinearRegression

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

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
-2.6645352591003757e-15
```

In [15]:
```
coef = pd.DataFrame(lr.coef_,x.columns,columns=['Co_efficient'])
coef
```

Out[15]:

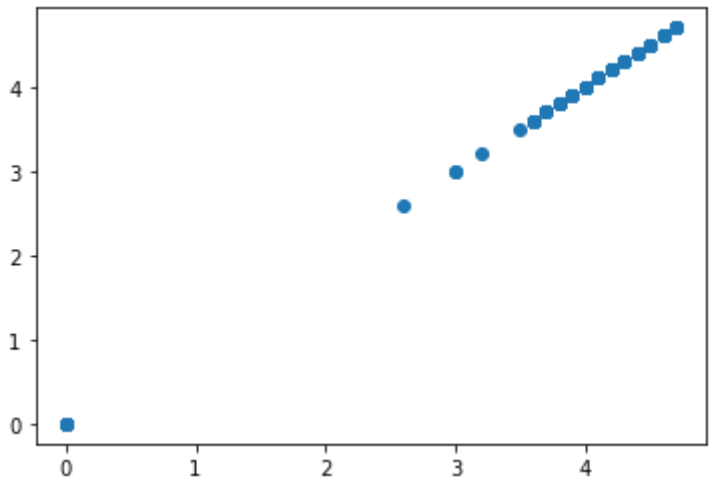
	Co_efficient
Rating ?/5	1.000000e+00
Number of Ratings	-1.301678e-22

In [16]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [17]:
```
prediction = lr.predict(x_test)
pp.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x23ca4fe7af0>
```



In [18]:
```
lr.score(x_test,y_test)
```

Out[18]:
```
1.0
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]:
```
1.0
```

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
r = Ridge(alpha=10)
r.fit(x_train,y_train)
r.score(x_test,y_test)
r.score(x_train,y_train)
```

Out[21]: 0.9993530438462878

In [22]:
```
l = Lasso(alpha=10)
l.fit(x_train,y_train)
l.score(x_test,y_test)
l.score(x_train,y_train)
```

Out[22]: 0.020542931496895456

In [ ]:
```

```