

In [1]:

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

In [94]:

```
from sklearn.model_selection import train_test_split
```

In [95]:

```
from sklearn.linear_model import LinearRegression
```

DataSet Vehicle

In [4]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\1.fiat500_VehicleSelection_Dataset - fiat500s.csv")
a
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.61151
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.2411
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.1111
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.6344
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.4944
...
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1549 rows × 11 columns



In [5]:

a.head()

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.61155986
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.2418895
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.4178
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.6346092
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.4956502

In [6]:

a.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1549 entries, 0 to 1548
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   ID               1538 non-null    float64
 1   model            1538 non-null    object  
 2   engine_power     1538 non-null    float64
 3   age_in_days      1538 non-null    float64
 4   km               1538 non-null    float64
 5   previous_owners  1538 non-null    float64
 6   lat              1538 non-null    float64
 7   lon              1549 non-null    object  
 8   price             1549 non-null    object  
 9   Unnamed: 9        0 non-null      float64
 10  Unnamed: 10       1 non-null      object  
dtypes: float64(7), object(4)
memory usage: 133.2+ KB
```

In [7]:

```
a.describe()
```

Out[7]:

	ID	engine_power	age_in_days	km	previous_owners	lat
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.54136
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.13351
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.85583
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.80299
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.39409
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.46796
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.79561

In [8]:

```
a.columns
```

Out[8]:

```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',  
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10'],  
      dtype='object')
```

In [9]:

a.isna()

Out[9]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...
1544	True	True	True	True	True	True	True	False	False
1545	True	True	True	True	True	True	True	False	False
1546	True	True	True	True	True	True	True	False	False
1547	True	True	True	True	True	True	True	False	False
1548	True	True	True	True	True	True	True	False	False

1549 rows × 11 columns



In [10]:

```
b=a.fillna(value=50)
b
```

Out[10]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1.0	lounge	51.0	882.0	250000.0	1.0	44.907242	8.6115
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495
...
1544	50.0	50	50.0	50.0	50.0	50.0	50.000000	
1545	50.0	50	50.0	50.0	50.0	50.0	50.000000	
1546	50.0	50	50.0	50.0	50.0	50.0	50.000000	Null
1547	50.0	50	50.0	50.0	50.0	50.0	50.000000	
1548	50.0	50	50.0	50.0	50.0	50.0	50.000000	

1549 rows × 11 columns



In [11]:

```
c=b.dropna(axis=1)
```

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	890
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	880
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	420
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	600
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	570
...
1544	50.0	50	50.0	50.0	50.0	50.0	50.000000	length	
1545	50.0	50	50.0	50.0	50.0	50.0	50.000000	concat	lonprice
1546	50.0	50	50.0	50.0	50.0	50.0	50.000000	Null values	N
1547	50.0	50	50.0	50.0	50.0	50.0	50.000000	find	

In [26]:

```
c.columns
```

Out[26]:

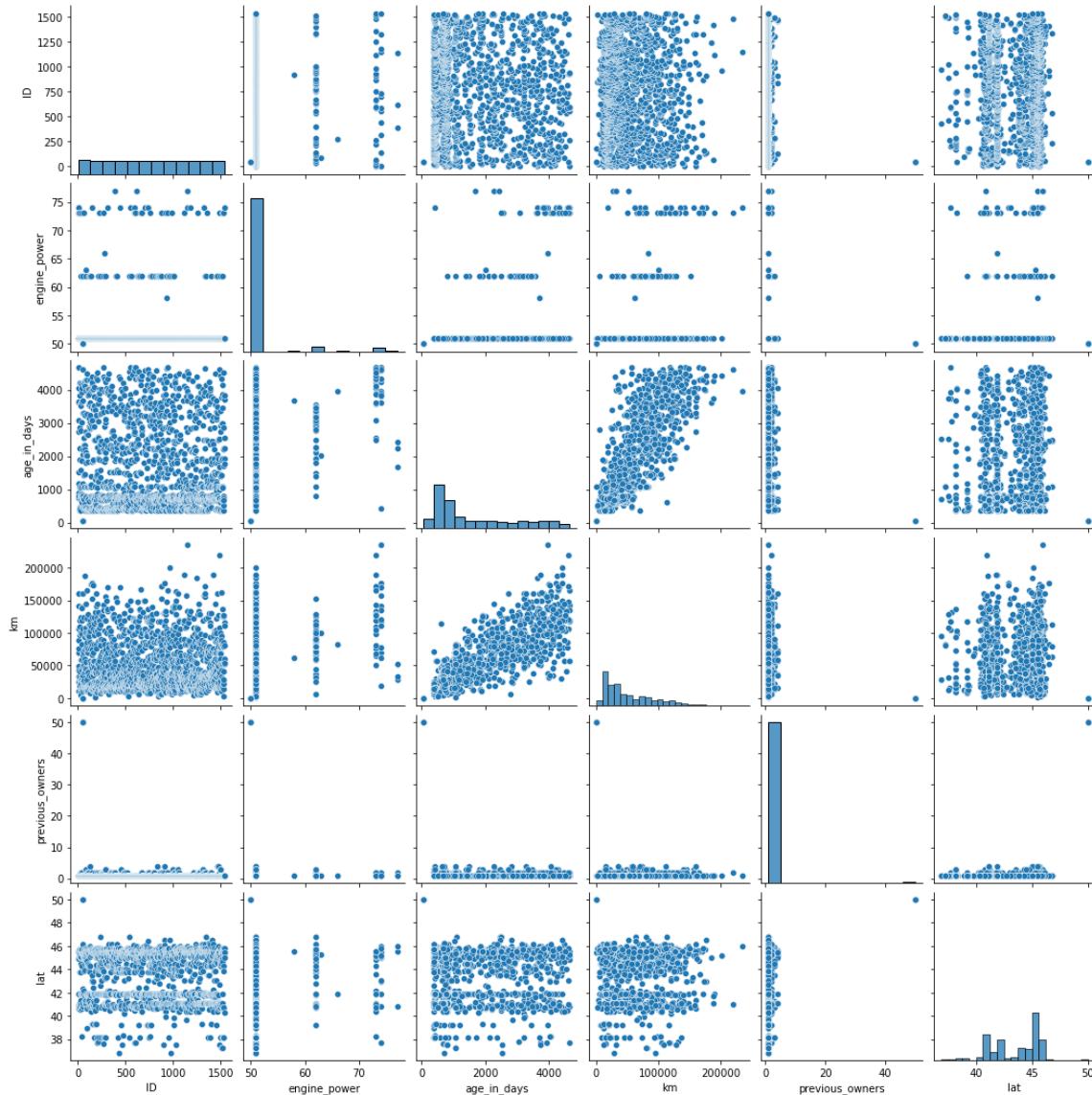
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [24]:

```
sns.pairplot(c)
```

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x23add9ebb80>
```



In [25]:

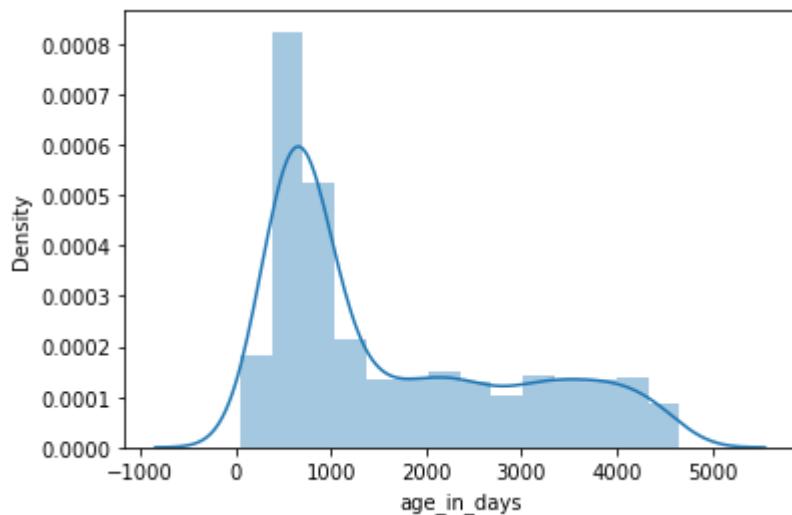
```
sns.distplot(c["age_in_days"])
```

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[25]:

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



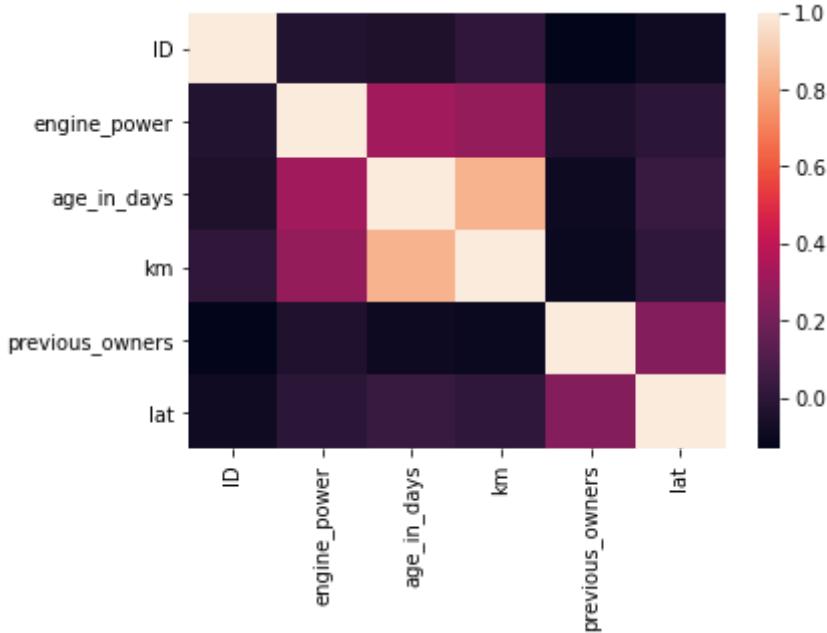
In [37]:

```
d=c[['ID', 'engine_power', 'age_in_days', 'km', 'previous_owners',
      'lat', 'lon', 'price']]

# Correlation map
sns.heatmap(c.corr())
```

Out[37]:

<AxesSubplot:>



In [67]:

```
x=c[['engine_power', 'age_in_days']]
y=c['age_in_days']
```

In [68]:

```
x_train,y_test,y_train,x_test=train_test_split(x,y)
```

In [69]:

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

Out[69]:

LinearRegression()

In [70]:

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

1.5916157281026244e-12

In [71]:

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

Out[71]:

Co-efficient	
engine_power	-3.993065e-14
age_in_days	1.000000e+00

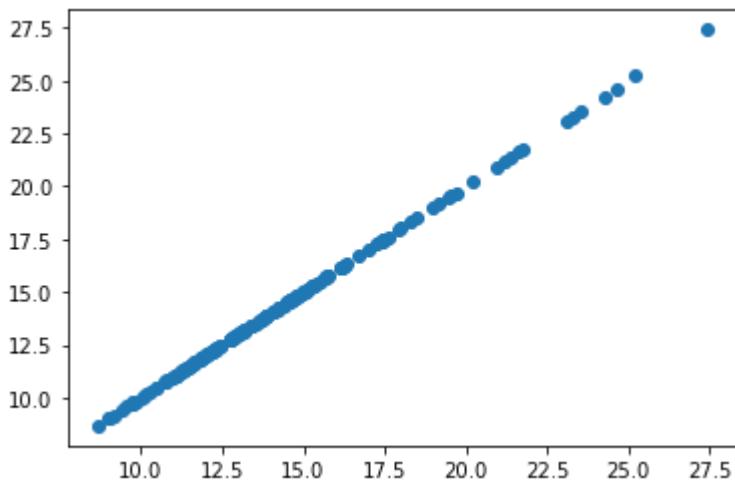
In [245]:

#Predicting

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

Out[245]:

```
<matplotlib.collections.PathCollection at 0x23ac4f1be50>
```



In [246]:

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

1.0

DataSet 2015

In [73]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\2.2015.csv")
a
```

Out[73]:

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

158 rows × 12 columns



In [74]:

a.head()

Out[74]:

	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	0.90563
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.90563
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.90563
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.90563
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.90563

◀ ▶

In [75]:

a.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 [76]:

a.describe()

Out[76]:

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

In [77]:

a.isna()

Out[77]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...
153	False	False	False	False	False	False	False	False	False
154	False	False	False	False	False	False	False	False	False
155	False	False	False	False	False	False	False	False	False
156	False	False	False	False	False	False	False	False	False
157	False	False	False	False	False	False	False	False	False

158 rows × 12 columns

In [78]:

a.columns

Out[78]:

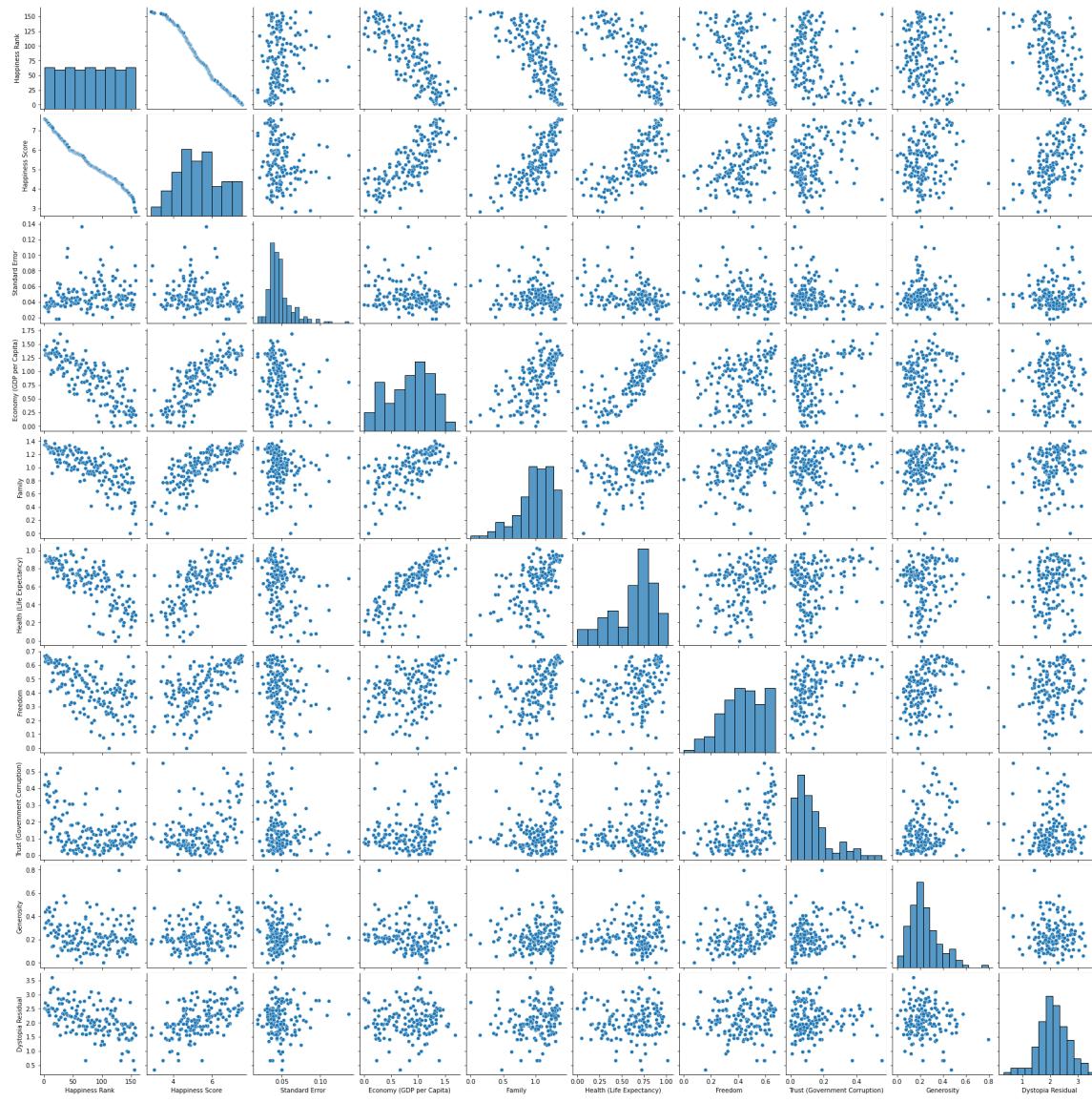
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [79]:

sns.pairplot(a)

Out[79]:

<seaborn.axisgrid.PairGrid at 0x23ae0401dc0>



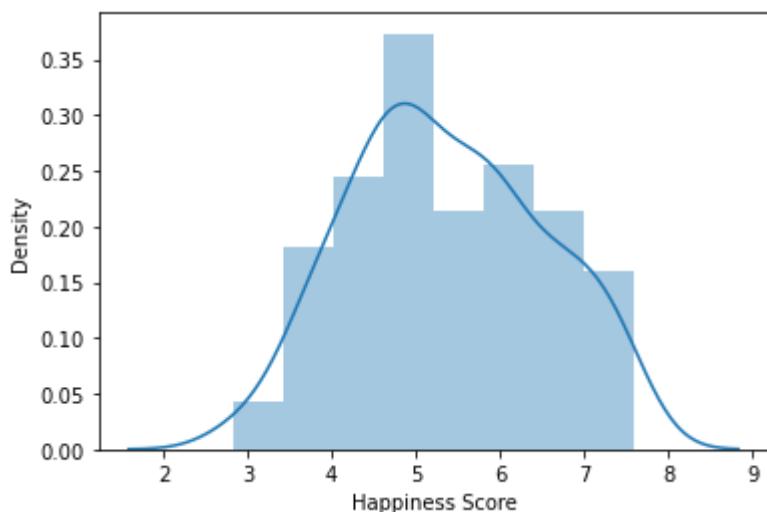
In [80]:

```
#normal distribution
sns.distplot(a['Happiness Score'])
```

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[80]:

```
<AxesSubplot:xlabel='Happiness Score', ylabel='Density'>
```



In [86]:

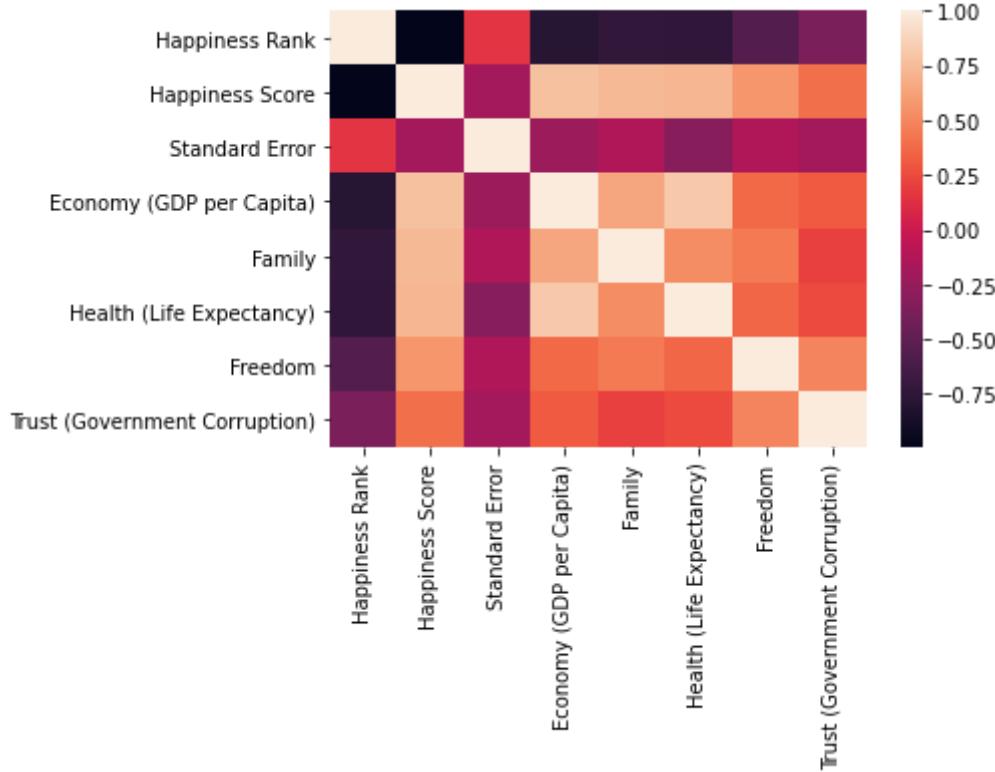
```
b=a[['Happiness Rank', 'Happiness Score',
      'Standard Error', 'Economy (GDP per Capita)', 'Family',
      'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)']]
```

In [87]:

```
# Correlation map
sns.heatmap(b.corr())
```

Out[87]:

<AxesSubplot:>



In [100]:

```
x=a[['Happiness Score',
      'Standard Error', 'Economy (GDP per Capita)', 'Family',
      'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)']]
y=a['Happiness Score']
```

In [101]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [102]:

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

Out[102]:

LinearRegression()

In [103]:

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

-2.6645352591003757e-15

In [104]:

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

Out[104]:

Co-efficient	
Happiness Score	1.000000e+00
Standard Error	-1.413934e-15
Economy (GDP per Capita)	3.869703e-16
Family	-3.033169e-16
Health (Life Expectancy)	-3.380781e-16
Freedom	2.538912e-16
Trust (Government Corruption)	3.918547e-16

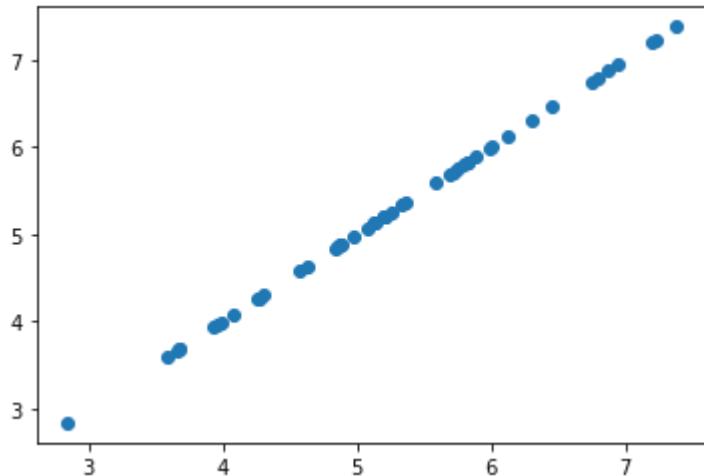
In [105]:

```
#Predicting
```

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

Out[105]:

```
<matplotlib.collections.PathCollection at 0x23ae5f31d90>
```



In [106]:

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

1.0

DataSet Fitness

In [107]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\3.Fitness-1.csv")
a
```

Out[107]:

Row Labels		Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	0.0562	0.0773	0.0616	75
1	B	0.0421	0.1727	0.1921	160
2	C	0.0983	0.1160	0.0517	101
3	D	0.0281	0.2191	0.0788	127
4	E	0.2528	0.1057	0.1182	179
5	F	0.0815	0.1624	0.1847	167
6	G	0.1854	0.0876	0.1749	171
7	H	0.2556	0.0593	0.1379	170
8	Grand Total	1.0000	1.0000	1.0000	1150

In [108]:

```
a.head()
```

Out[108]:

Row Labels		Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	0.0562	0.0773	0.0616	75
1	B	0.0421	0.1727	0.1921	160
2	C	0.0983	0.1160	0.0517	101
3	D	0.0281	0.2191	0.0788	127
4	E	0.2528	0.1057	0.1182	179

In [109]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Row Labels      9 non-null       object 
 1   Sum of Jan      9 non-null       float64
 2   Sum of Feb      9 non-null       float64
 3   Sum of Mar      9 non-null       float64
 4   Sum of Total Sales 9 non-null     int64  
dtypes: float64(3), int64(1), object(1)
memory usage: 488.0+ bytes
```

In [110]:

a.describe()

Out[110]:

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
count	9.000000	9.000000	9.000000	9.000000
mean	0.222222	0.222233	0.222211	255.555556
std	0.304383	0.296123	0.296410	337.332963
min	0.028100	0.059300	0.051700	75.000000
25%	0.056200	0.087600	0.078800	127.000000
50%	0.098300	0.116000	0.137900	167.000000
75%	0.252800	0.172700	0.184700	171.000000
max	1.000000	1.000000	1.000000	1150.000000

In [111]:

a.isna()

Out[111]:

Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
5	False	False	False	False
6	False	False	False	False
7	False	False	False	False
8	False	False	False	False

In [113]:

a.columns

Out[113]:

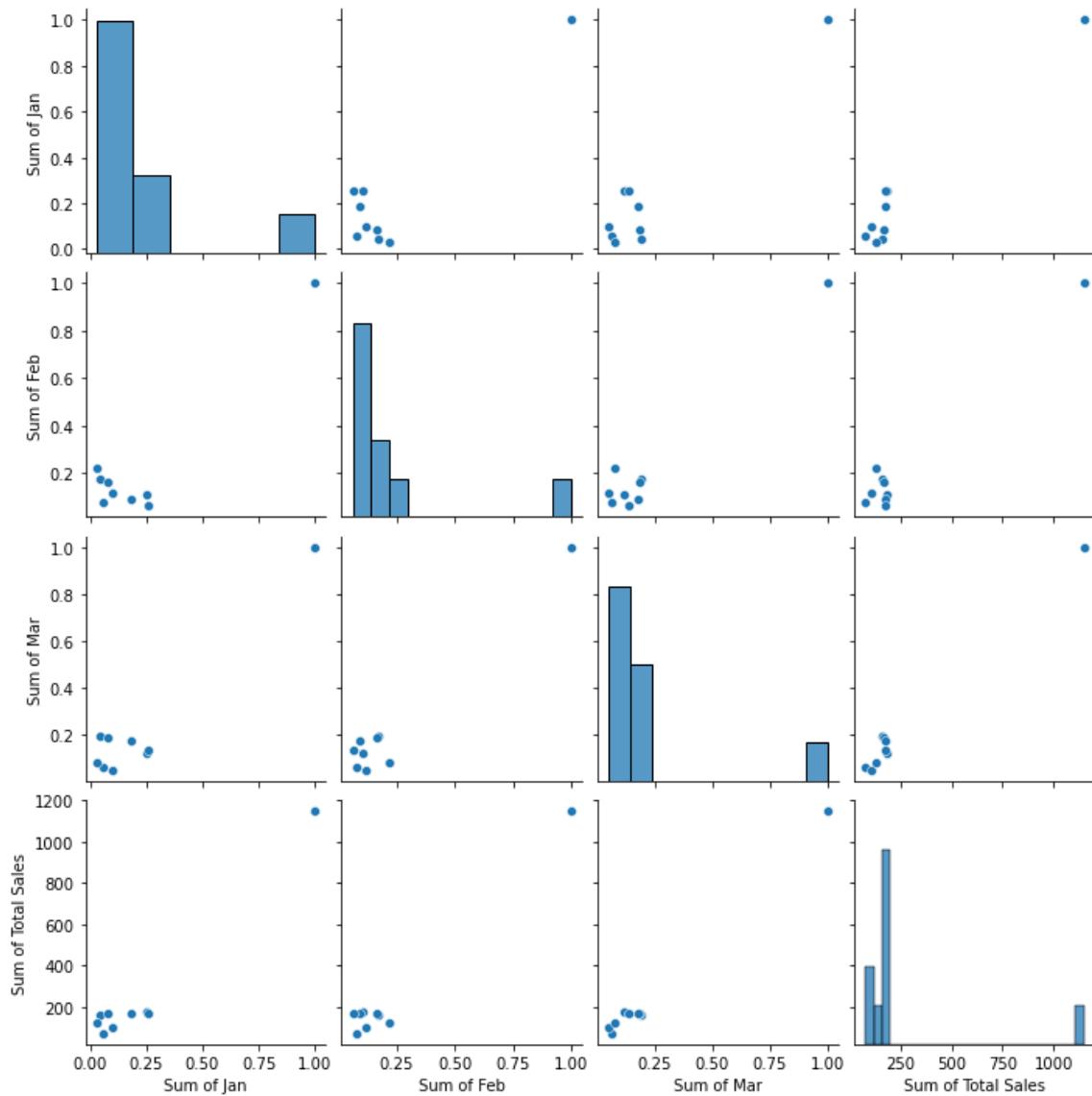
```
Index(['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales'],
      dtype='object')
```

In [114]:

```
sns.pairplot(a)
```

Out[114]:

```
<seaborn.axisgrid.PairGrid at 0x23ae5f4bd90>
```



In [115]:

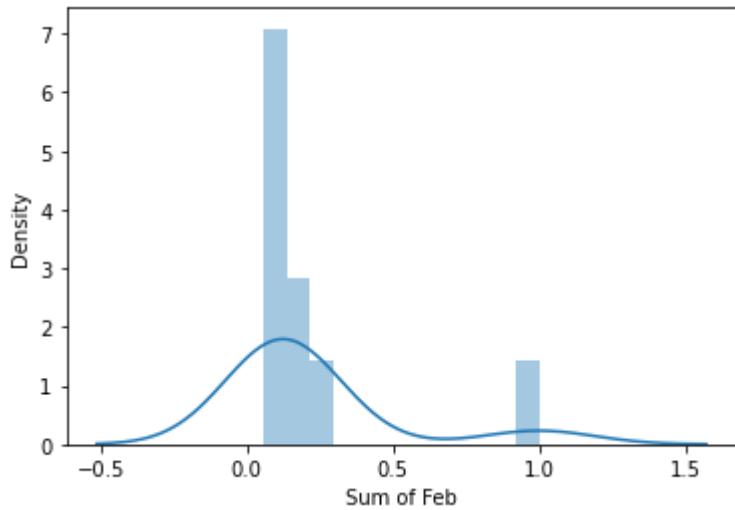
```
sns.distplot(a["Sum of Feb"])
```

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[115]:

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



In [116]:

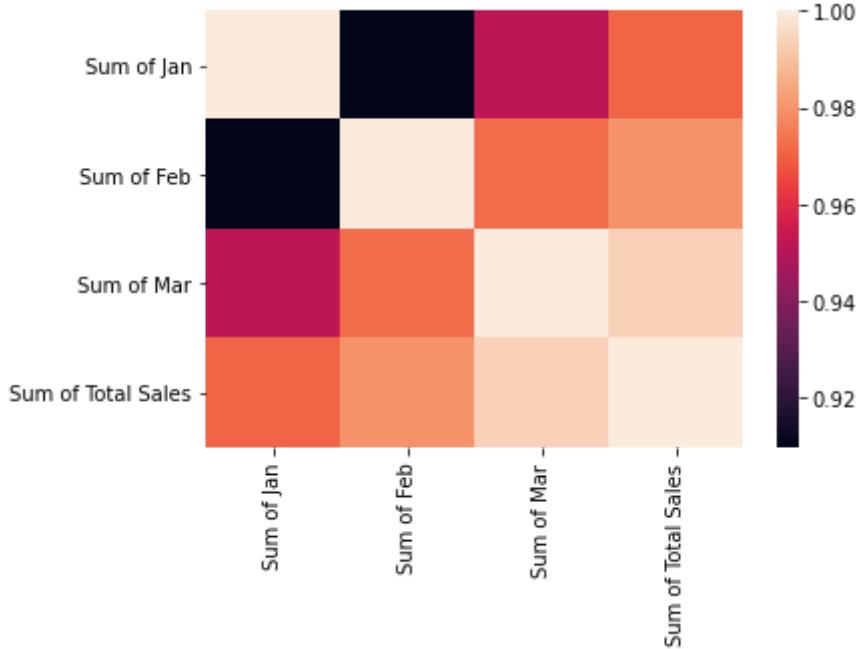
```
b=a[['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',  
      'Sum of Total Sales']]
```

In [117]:

```
# Correlation map
sns.heatmap(b.corr())
```

Out[117]:

<AxesSubplot:>



In [121]:

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

In [122]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [123]:

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

Out[123]:

LinearRegression()

In [124]:

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

-1.1102230246251565e-16

In [125]:

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

Out[125]:

Co-efficient	
Sum of Jan	1.250505e-13
Sum of Feb	1.361424e-13
Sum of Mar	1.000000e+00
Sum of Total Sales	-3.514289e-16

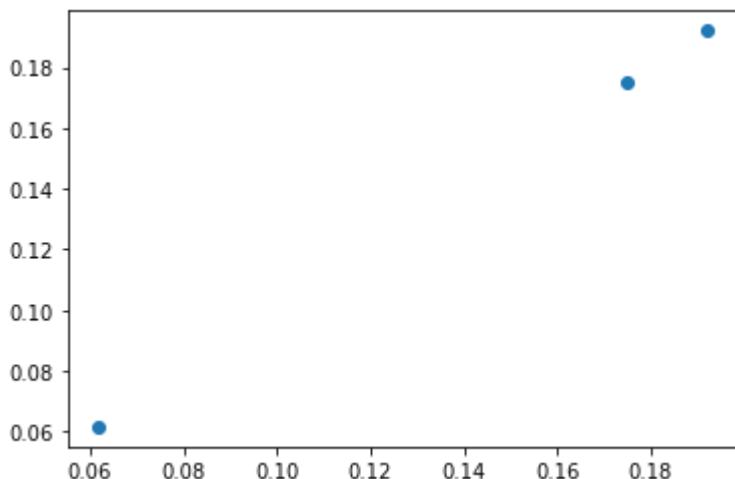
In [126]:

```
#Predicting
```

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

Out[126]:

```
<matplotlib.collections.PathCollection at 0x23ae6af2bb0>
```



In [127]:

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

```
1.0
```

DataSet Drug

In [128]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\4_drug200 - 4_drug200.csv")
a
```

Out[128]:

	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 [129]:

```
a.head(10)
```

Out[129]:

	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 [130]:

```
a.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 [131]:

```
a.describe()
```

Out[131]:

	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 [132]:

```
a.isna()
```

Out[132]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...
195	False	False	False	False	False	False
196	False	False	False	False	False	False
197	False	False	False	False	False	False
198	False	False	False	False	False	False
199	False	False	False	False	False	False

200 rows × 6 columns

In [133]:

```
a.columns
```

Out[133]:

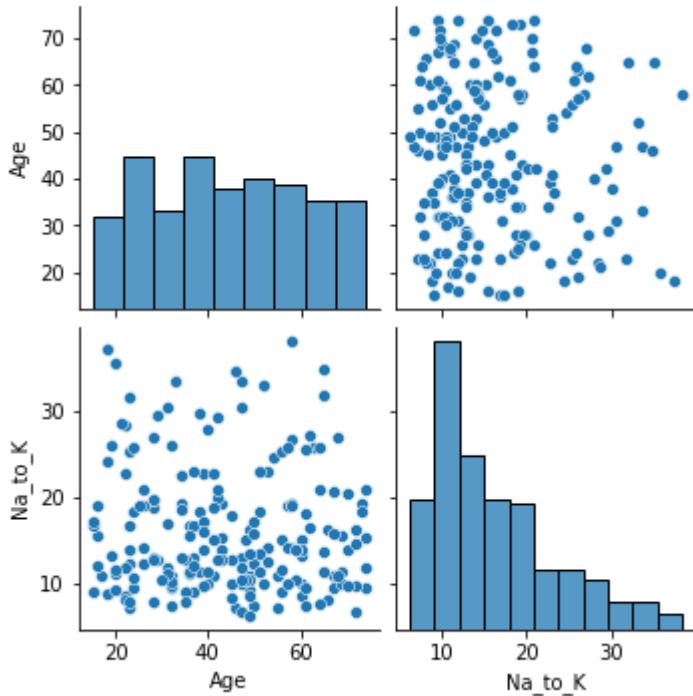
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

In [134]:

```
sns.pairplot(a)
```

Out[134]:

```
<seaborn.axisgrid.PairGrid at 0x23ae6b16d60>
```



In [135]:

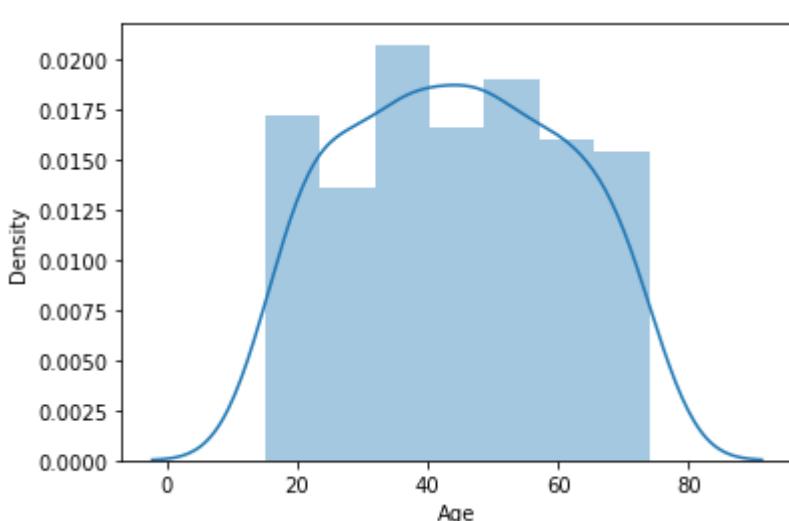
```
sns.distplot(a["Age"])
```

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[135]:

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

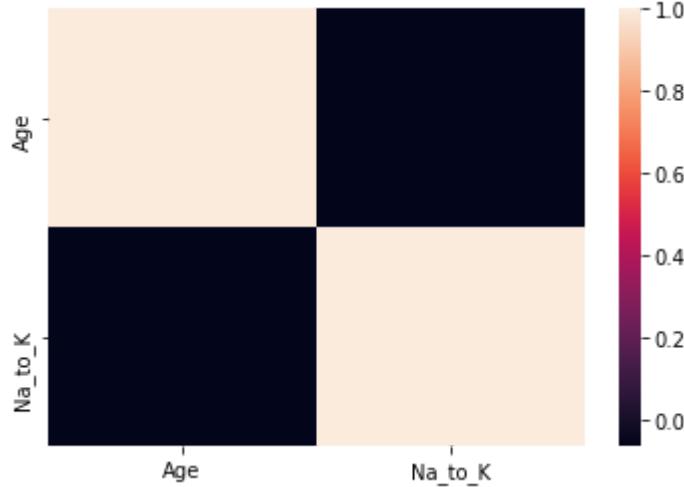


In [136]:

```
# Correlation map
sns.heatmap(a.corr())
```

Out[136]:

<AxesSubplot:>



In [137]:

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

In [140]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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

Out[140]:

LinearRegression()

In [141]:

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

0.0

In [142]:

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

Out[142]:

	Co-efficient
Age	1.000000e+00
Na_to_K	8.821081e-18

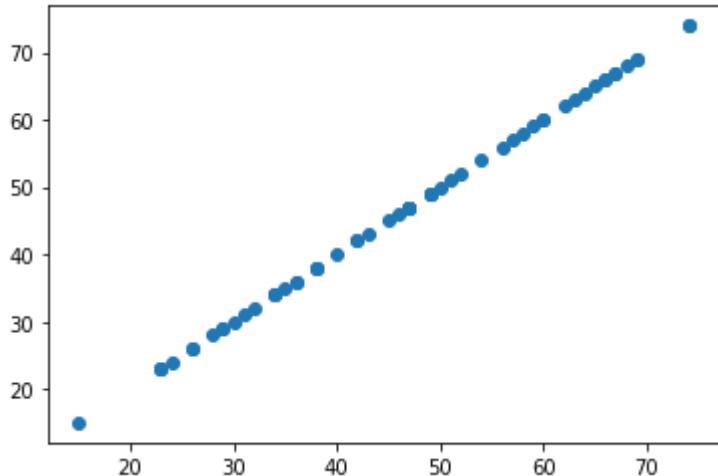
In [143]:

```
#Predicting
```

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

Out[143]:

```
<matplotlib.collections.PathCollection at 0x23ae7da1be0>
```



In [144]:

```
# Score
```

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

```
1.0
```

DataSet Instagram

In [145]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\5_Instagram data - 5_Instagram data.csv"  
a
```

Out[145]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits
0	3920	2586	1028	619	56	98	9	5	162	34
1	5394	2727	1838	1174	78	194	7	14	224	48
2	4021	2085	1188	0	533	41	11	1	131	61
3	4528	2700	621	932	73	172	10	7	213	21
4	2518	1704	255	279	37	96	5	4	123	11
...
114	13700	5185	3041	5352	77	573	2	38	373	70
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
118	36919	13473	4176	16444	2547	653	5	26	443	61

119 rows × 13 columns

In [146]:

```
a.head(20)
```

Out[146]:

Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Foll
3920	2586	1028	619	56	98	9	5	162	35	
5394	2727	1838	1174	78	194	7	14	224	48	
4021	2085	1188	0	533	41	11	1	131	62	
4528	2700	621	932	73	172	10	7	213	23	
2518	1704	255	279	37	96	5	4	123	8	
3884	2046	1214	329	43	74	7	10	144	9	
2621	1543	599	333	25	22	5	1	76	26	
3541	2071	628	500	60	135	4	9	124	12	
3749	2384	857	248	49	155	6	8	159	36	
4115	2609	1104	178	46	122	6	3	191	31	
2218	1597	411	162	15	28	6	3	81	29	
3234	2414	476	185	75	122	8	14	151	15	
4344	2168	1274	673	40	119	7	11	162	8	

Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits	Follows
3216	2524	212	201	223	121	5	5	142	20	
9453	2525	5799	208	794	100	6	10	294	181	
5055	2017	2351	298	108	101	7	11	159	17	
4002	3401	278	128	73	111	17	18	205	16	
3169	1979	707	341	32	106	8	1	121	21	
6168	2177	3450	153	296	82	6	6	151	77	
In [147]	1338	655	276	39	40	8	20	72	10	

```
a.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 [148]:

a.describe()

Out[148]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	C
count	119.000000	119.000000	119.000000	119.000000	119.000000	119.000000	11
mean	5703.991597	2475.789916	1887.512605	1078.100840	171.092437	153.310924	
std	4843.780105	1489.386348	1884.361443	2613.026132	289.431031	156.317731	
min	1941.000000	1133.000000	116.000000	0.000000	9.000000	22.000000	
25%	3467.000000	1945.000000	726.000000	157.500000	38.000000	65.000000	
50%	4289.000000	2207.000000	1278.000000	326.000000	74.000000	109.000000	
75%	6138.000000	2602.500000	2363.500000	689.500000	196.000000	169.000000	
max	36919.000000	13473.000000	11817.000000	17414.000000	2547.000000	1095.000000	1

In [149]:

a.isna()

Out[149]:

	Impressions	From Home	From Hashtags	From Explore	From Other	Saves	Comments	Shares	Likes	Profile Visits
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...
114	False	False	False	False	False	False	False	False	False	False
115	False	False	False	False	False	False	False	False	False	False
116	False	False	False	False	False	False	False	False	False	False
117	False	False	False	False	False	False	False	False	False	False
118	False	False	False	False	False	False	False	False	False	False

119 rows × 13 columns

In [150]:

```
a.columns
```

Out[150]:

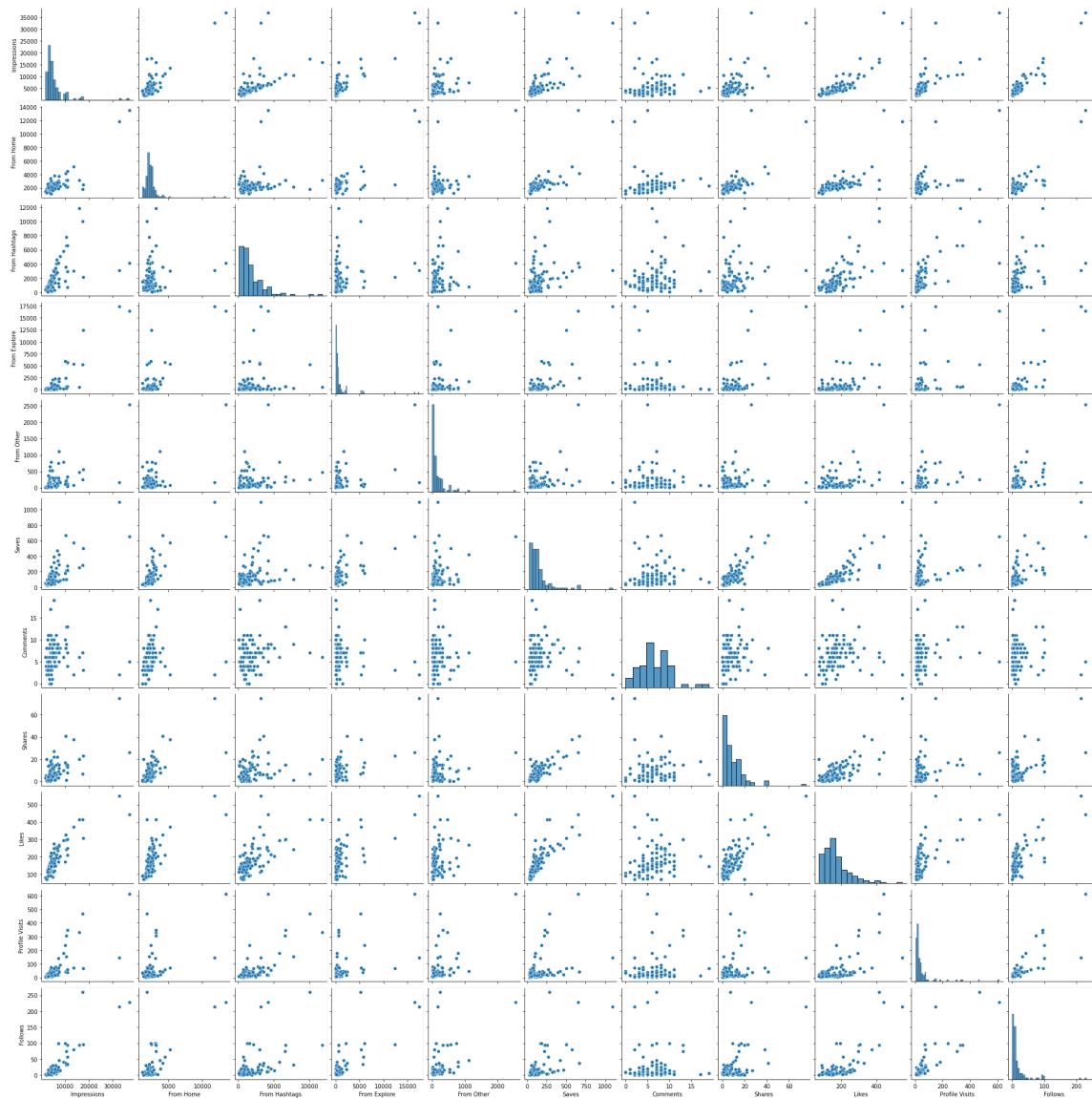
```
Index(['Impressions', 'From Home', 'From Hashtags', 'From Explore',
       'From Other', 'Saves', 'Comments', 'Shares', 'Likes', 'Profile Visits',
       'Follows', 'Caption', 'Hashtags'],
      dtype='object')
```

In [151]:

```
sns.pairplot(a)
```

Out[151]:

```
<seaborn.axisgrid.PairGrid at 0x23ae7db8640>
```



In [152]:

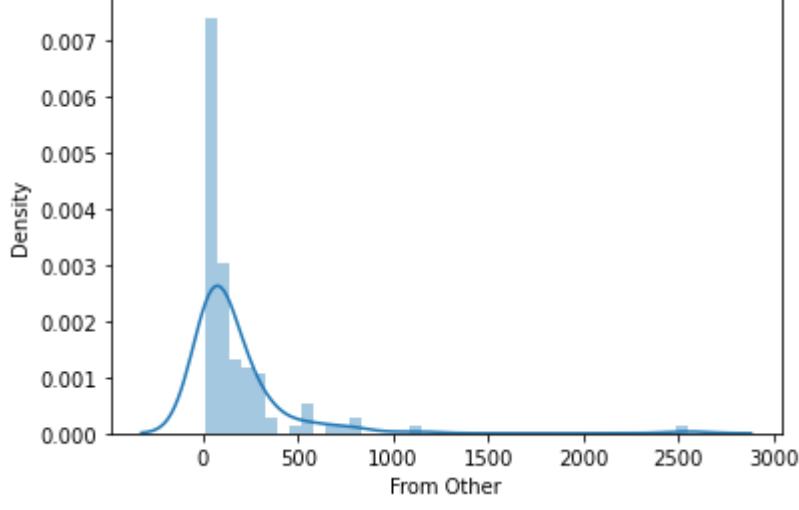
```
#normal distribution
sns.distplot(a["From Other"])
```

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[152]:

```
<AxesSubplot:xlabel='From Other', ylabel='Density'>
```

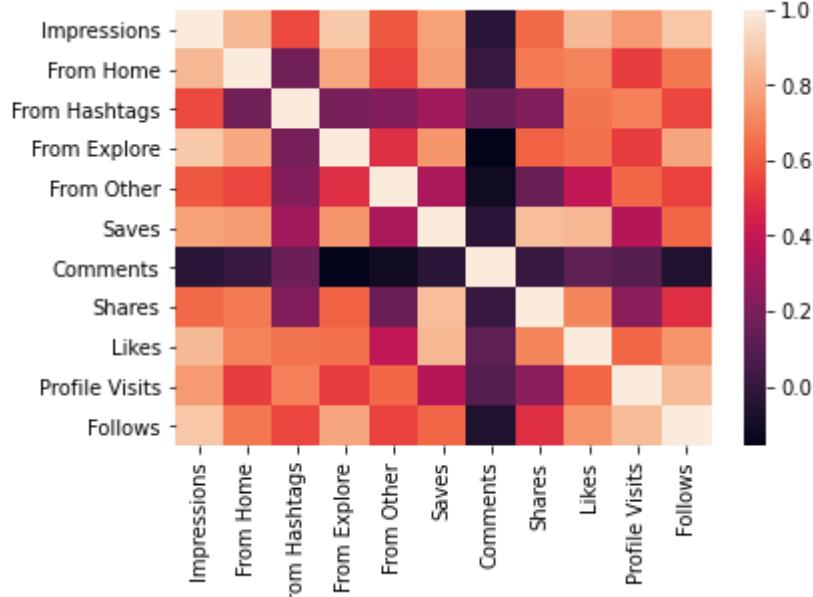


In [153]:

```
# Correlation map
sns.heatmap(a.corr())
```

Out[153]:

```
<AxesSubplot:>
```



In [154]:

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

In [155]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

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

Out[155]:

```
LinearRegression()
```

In [156]:

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

```
-7.73070496506989e-12
```

In [157]:

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

Out[157]:

	Co-efficient
Impressions	0.000000e+00
From Home	1.000000e+00
From Hashtags	5.711878e-16
From Explore	6.476466e-16
From Other	5.840199e-16
Saves	-2.641384e-16
Comments	1.473117e-14
Shares	1.520683e-14
Likes	7.700767e-17
Profile Visits	1.058926e-15
Follows	-1.574386e-15

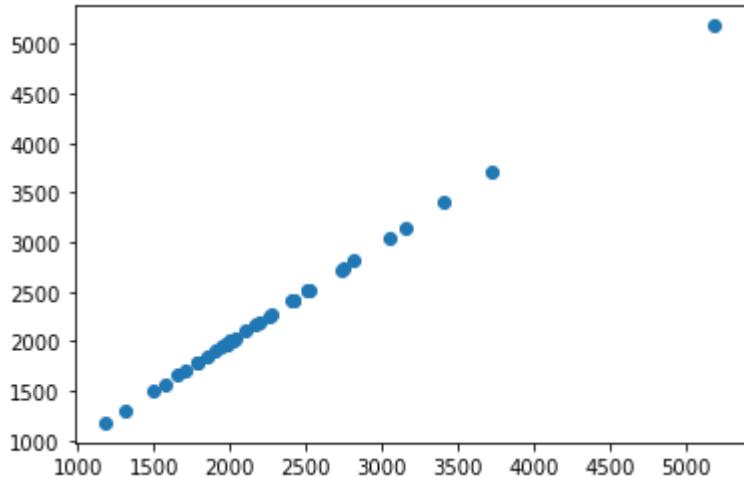
In [158]:

#Predicting

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

Out[158]:

<matplotlib.collections.PathCollection at 0x23aebc9d520>



In [159]:

Score

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

1.0

DataSet SalesWorkLoad

In [160]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\6.Salesworkload1.csv")
a
```

Out[160]:

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

7658 rows × 14 columns



In [196]:

```
b=a.head(15)
b
```

Out[196]:

	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.
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.
6	10.2016	1.0	United Kingdom	88253.0	London (I)	13.0	Food	16468.251	0.
7	10.2016	1.0	United Kingdom	88253.0	London (I)	7.0	Clothing	4698.471	0.
8	10.2016	1.0	United Kingdom	88253.0	London (I)	8.0	Household	1183.272	0.
9	10.2016	1.0	United Kingdom	88253.0	London (I)	9.0	Hardware	2029.815	0.
10	10.2016	1.0	United Kingdom	88253.0	London (I)	14.0	Non Food	7911.558	0.
11	10.2016	1.0	United Kingdom	88253.0	London (I)	15.0	Admin	4308.243	0.
12	10.2016	1.0	United Kingdom	88253.0	London (I)	12.0	Checkout	5825.097	0.
13	10.2016	1.0	United Kingdom	88253.0	London (I)	16.0	Customer Services	3320.085	0.
14	10.2016	1.0	United Kingdom	88253.0	London (I)	11.0	Delivery	0	0.



In [162]:

a.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 [163]:

a.describe()

Out[163]:

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

In [164]:

a.isna()

Out[164]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	S u
0	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	F
...
7653	False	False	False	False	False	False	False	False	False	F
7654	False	False	False	False	False	False	False	False	False	F
7655	False	False	False	False	False	False	False	False	False	F
7656	False	False	False	False	False	False	False	False	False	F
7657	False	False	False	False	False	False	False	False	False	F

7658 rows × 14 columns

In [197]:

c=b.fillna(value=1)

c

Out[197]:

	MonthYear	Time index	Country	StoreID	City	Dept_ID	Dept. Name	HoursOwn	HoursLease	Sales units
0	10.2016	1.0	United Kingdom	88253.0	London (I)	1.0	Dry	3184.764	0.0	398560.0
1	10.2016	1.0	United Kingdom	88253.0	London (I)	2.0	Frozen	1582.941	0.0	82725.0
2	10.2016	1.0	United Kingdom	88253.0	London (I)	3.0	other	47.205	0.0	438400.0
3	10.2016	1.0	United Kingdom	88253.0	London (I)	4.0	Fish	1623.852	0.0	309425.0
4	10.2016	1.0	United Kingdom	88253.0	London (I)	5.0	Fruits & Vegetables	1759.173	0.0	165515.0
5	10.2016	1.0	United Kingdom	88253.0	London (I)	6.0	Meat	8270.316	0.0	1713310.0

In [198]:

c.columns

Out[198]:

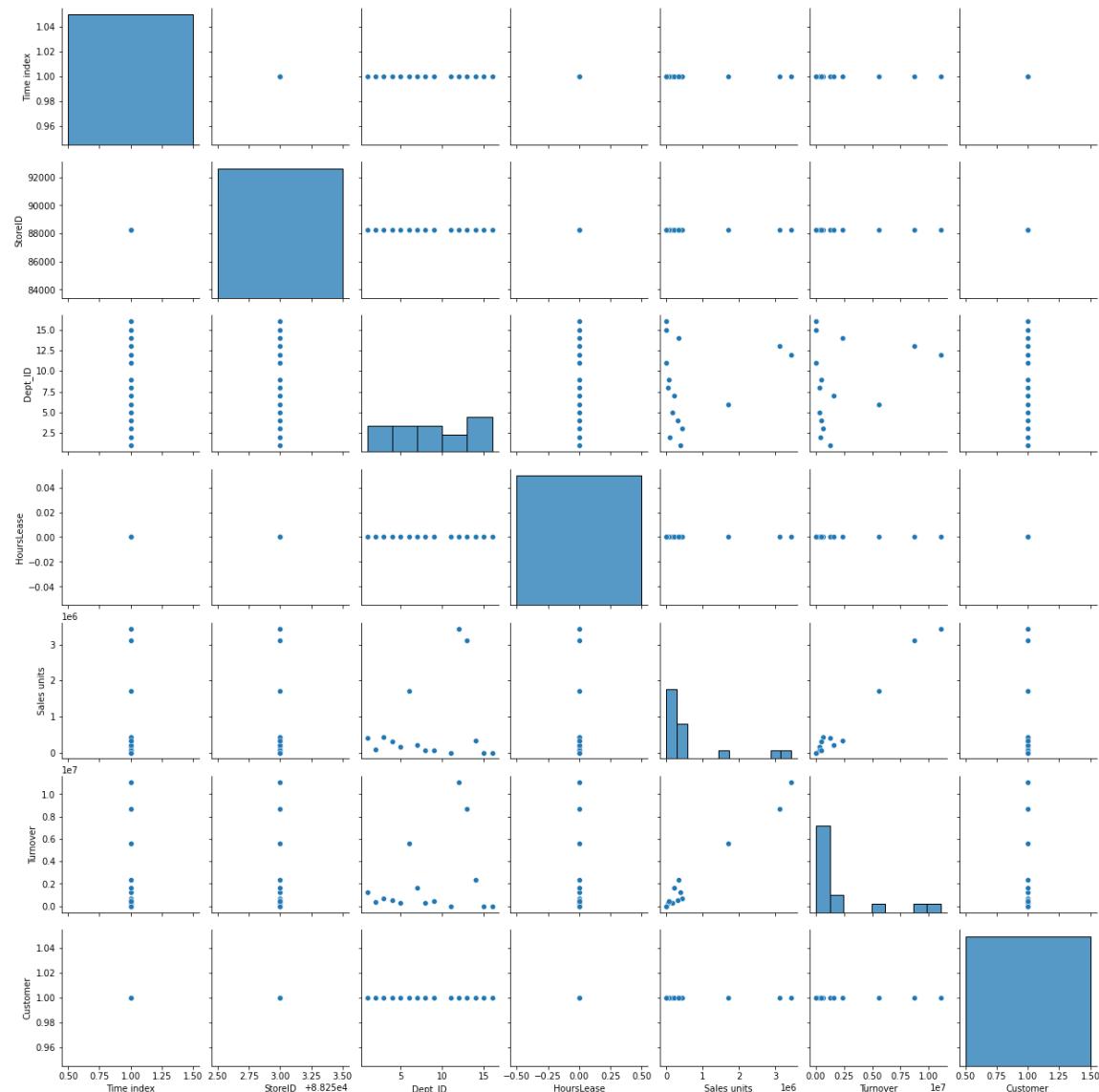
```
Index(['MonthYear', 'Time index', 'Country', 'StoreID', 'City', 'Dept_ID',
       'Dept. Name', 'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)', 'Opening hours'],
      dtype='object')
```

In [199]:

sns.pairplot(c)

Out[199]:

<seaborn.axisgrid.PairGrid at 0x23af4b80f40>



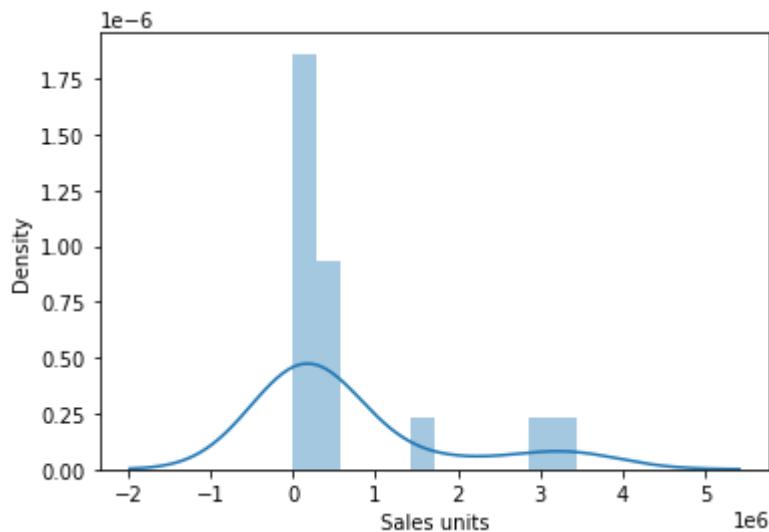
In [200]:

```
#normal distribution
sns.distplot(c["Sales units"])
```

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[200]:

```
<AxesSubplot:xlabel='Sales units', ylabel='Density'>
```

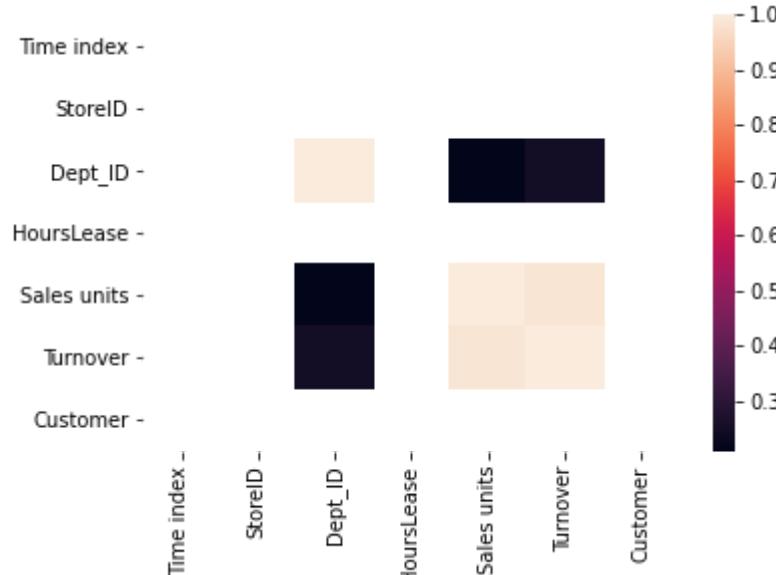


In [201]:

```
# Correlation map
sns.heatmap(c.corr())
```

Out[201]:

```
<AxesSubplot:>
```



In [202]:

```
x=c[['Time index', 'StoreID', 'Dept_ID',
      'HoursOwn', 'HoursLease', 'Sales units', 'Turnover',
      'Customer']]
y=c['Dept_ID']
```

In [203]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

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

Out[203]:

```
LinearRegression()
```

In [204]:

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

```
-2.2805433985695345e-08
```

In [205]:

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

Out[205]:

	Co-efficient
Time index	0.000000e+00
StoreID	2.591914e-13
Dept_ID	1.000000e+00
HoursOwn	-2.160772e-14
HoursLease	0.000000e+00
Sales units	-2.287525e-16
Turnover	8.469926e-17
Customer	0.000000e+00

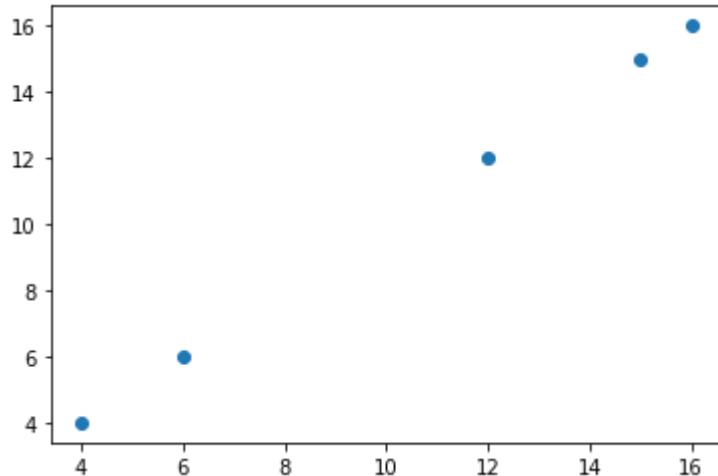
In [206]:

#Predicting

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

Out[206]:

<matplotlib.collections.PathCollection at 0x23af7a55550>



In [207]:

Score

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

1.0

DataSet Uber

In [208]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\7.uber - uber.csv")
a
```

Out[208]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770
3	25894730	2009-06-26 8:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085
...
199995	42598914	2012-10-28 10:49:00	3.0	2012-10-28 10:49:00 UTC	-73.987042	40.739367
199996	16382965	2014-03-14 1:09:00	7.5	2014-03-14 01:09:00 UTC	-73.984722	40.736837
199997	27804658	2009-06-29 0:42:00	30.9	2009-06-29 00:42:00 UTC	-73.986017	40.756487
199998	20259894	2015-05-20 14:56:25	14.5	2015-05-20 14:56:25 UTC	-73.997124	40.725452
199999	11951496	2010-05-15 4:08:00	14.1	2010-05-15 04:08:00 UTC	-73.984395	40.720077

200000 rows × 9 columns



In [209]:

```
b=a.head(1000)
b
```

Out[209]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	di
0	24238194	2015-05-07 19:52:06	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	
1	27835199	2009-07-17 20:04:56	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	
2	44984355	2009-08-24 21:45:00	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	
3	25894730	2009-06-26 8:22:21	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	
4	17610152	2014-08-28 17:47:00	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	
...
995	13439193	2011-05-04 6:39:00	5.7	2011-05-04 06:39:00 UTC	-73.969720	40.757577	
996	32405310	2011-11-23 20:43:20	8.1	2011-11-23 20:43:20 UTC	-73.993784	40.757054	
997	51612001	2010-01-11 20:58:00	8.5	2010-01-11 20:58:00 UTC	-73.972338	40.765078	
998	937243	2013-06-12 17:01:24	5.5	2013-06-12 17:01:24 UTC	-73.979054	40.784730	
999	47946613	2011-12-11 21:48:22	9.7	2011-12-11 21:48:22 UTC	-73.983675	40.729944	

1000 rows × 9 columns



In [210]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Unnamed: 0        1000 non-null   int64  
 1   key              1000 non-null   object  
 2   fare_amount      1000 non-null   float64 
 3   pickup_datetime  1000 non-null   object  
 4   pickup_longitude 1000 non-null   float64 
 5   pickup_latitude   1000 non-null   float64 
 6   dropoff_longitude 1000 non-null   float64 
 7   dropoff_latitude  1000 non-null   float64 
 8   passenger_count   1000 non-null   int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 70.4+ KB
```

In [211]:

```
b.describe()
```

Out[211]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
count	1.000000e+03	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	2.744197e+07	11.116280	-72.646044	40.017119	-72.717954	40.735620
std	1.616669e+07	9.216902	9.840422	5.420590	9.567772	5.798032
min	6.250000e+02	2.500000	-74.689831	0.000000	-74.689831	0.000000
25%	1.305167e+07	6.075000	-73.992702	40.735620	-73.991797	40.735620
50%	2.724411e+07	8.500000	-73.982114	40.752500	-73.980328	40.752500
75%	4.185744e+07	12.500000	-73.969435	40.765824	-73.963108	40.765824
max	5.525106e+07	93.160000	0.001782	40.850558	0.000875	40.850558

In [212]:

```
b.isna()
```

Out[212]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...
995	False	False	False	False	False	False	False
996	False	False	False	False	False	False	False
997	False	False	False	False	False	False	False

In [218]:

```
print(b.columns)
```

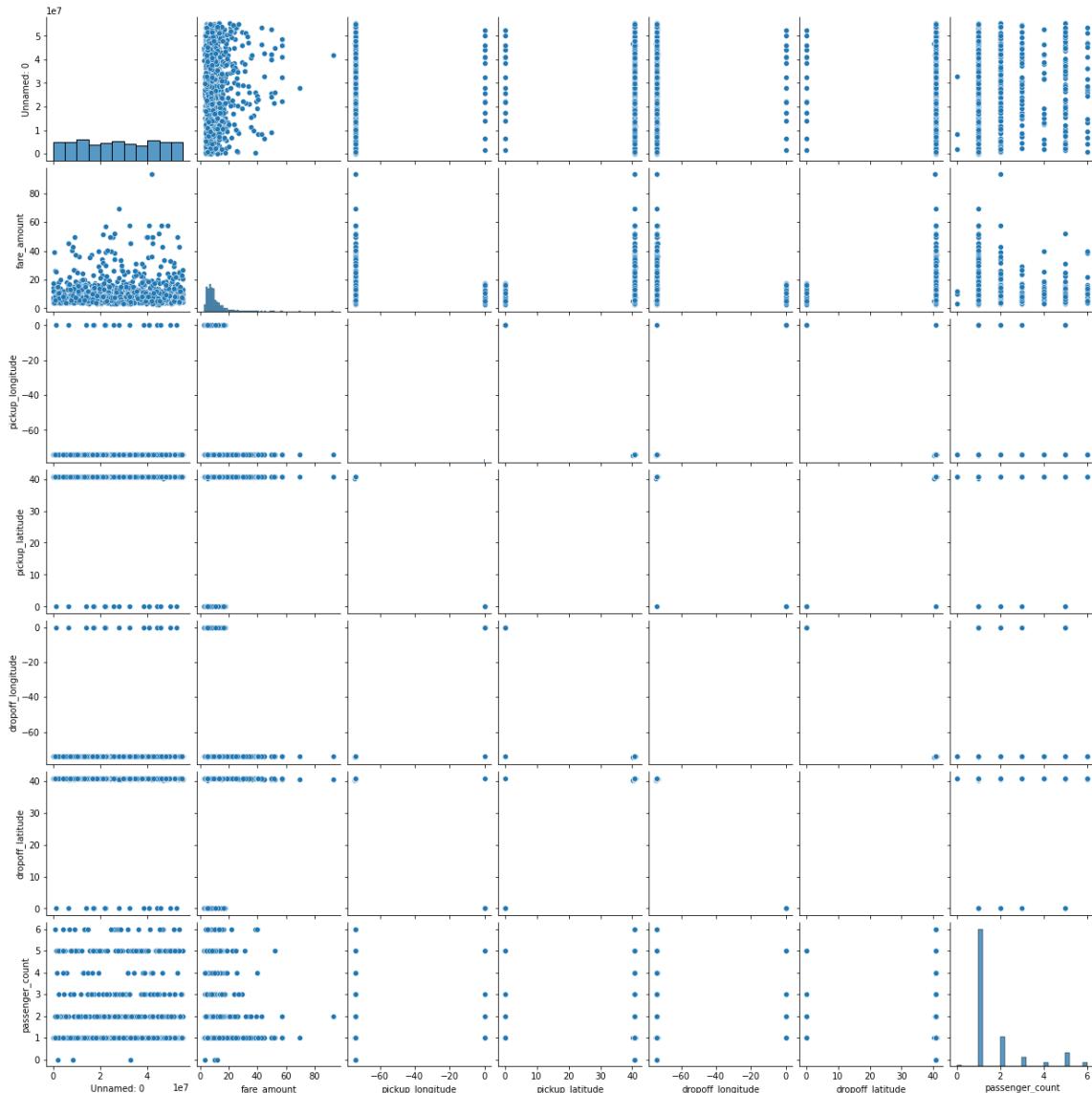
```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

In [213]:

```
sns.pairplot(b)
```

Out[213]:

```
<seaborn.axisgrid.PairGrid at 0x23af7a67790>
```



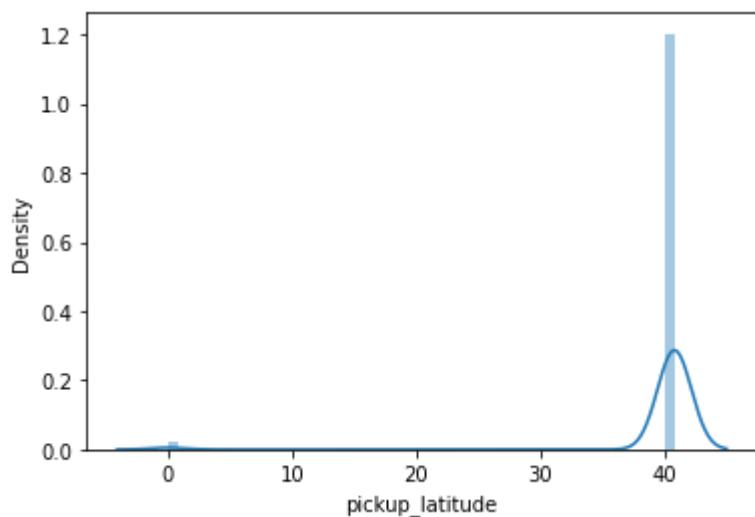
In [215]:

```
#normal distribution  
sns.distplot(b["pickup_latitude"])
```

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[215]:

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

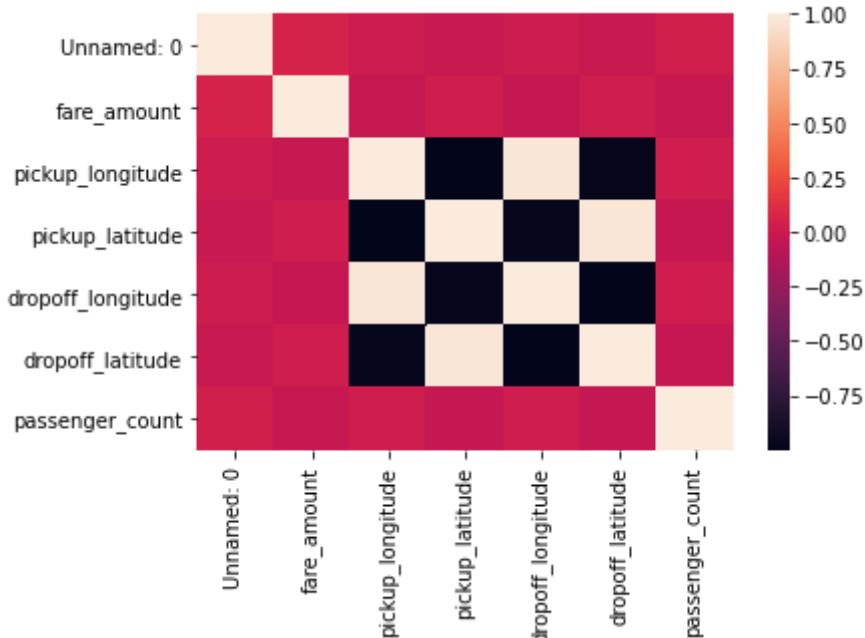


In [216]:

```
# Correlation map  
sns.heatmap(b.corr())
```

Out[216]:

```
<AxesSubplot:>
```



In [223]:

```
x=b[['Unnamed: 0', 'fare_amount',
      'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
      'dropoff_latitude', 'passenger_count']]
y=b['fare_amount']
```

In [224]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

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

Out[224]:

```
LinearRegression()
```

In [225]:

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

```
-1.3322676295501878e-13
```

In [226]:

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

Out[226]:

	Co-efficient
Unnamed: 0	6.208564e-21
fare_amount	1.000000e+00
pickup_longitude	-5.777191e-15
pickup_latitude	-1.590876e-15
dropoff_longitude	4.793152e-15
dropoff_latitude	-9.245334e-16
passenger_count	5.029177e-17

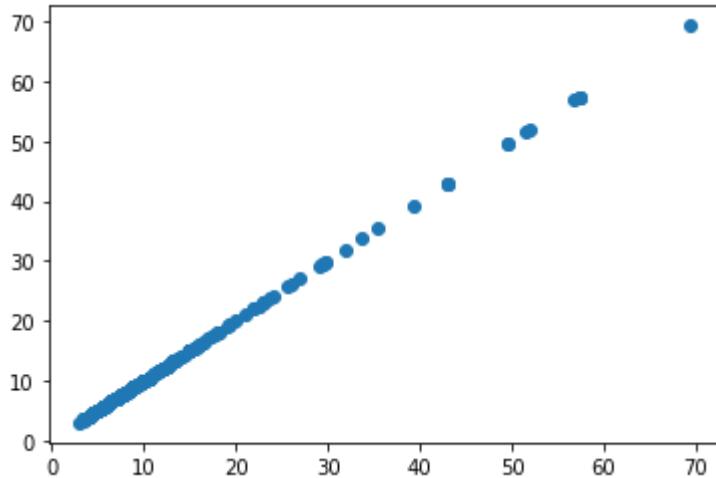
In [227]:

```
#Predicting
```

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

Out[227]:

```
<matplotlib.collections.PathCollection at 0x23a97cd88e0>
```



In [228]:

```
# Score
```

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

```
1.0
```

DataSet Cancer

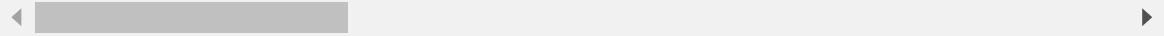
In [229]:

```
a=pd.read_csv(r"E:\Python Data Science\Documents\8_BreastCancerPrediction - 8_BreastCancer.csv")
```

Out[229]:

	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
0	842302	M	17.99	10.38	122.80	1001.0	0.1189	0.2781	0.09713	0.001913	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203		
1	842517	M	20.57	17.77	132.90	1326.0	0.1096	0.2952	0.1064	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
2	84300903	M	19.69	21.25	130.00	1203.0	0.1054	0.2991	0.1054	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
3	84348301	M	11.42	20.38	77.58	386.1	0.08498	0.2057	0.08498	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
4	84358402	M	20.29	14.34	135.10	1297.0	0.1054	0.2991	0.1054	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
...				
564	926424	M	21.56	22.39	142.00	1479.0	0.1124	0.3094	0.1124	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203		
565	926682	M	20.13	28.25	131.20	1261.0	0.1054	0.2991	0.1054	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
566	926954	M	16.60	28.08	108.30	858.1	0.09713	0.2952	0.09713	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
567	927241	M	20.60	29.33	140.10	1265.0	0.1124	0.3094	0.1124	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			
568	92751	B	7.76	24.54	47.92	181.0	0.07879	0.2351	0.07879	0.001860	0.1326	0.0005469	0.0003546	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203	0.0001203			

569 rows × 32 columns



In [230]:

a.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 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 
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [231]:

```
a.describe()
```

Out[231]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.0
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.1
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.1

8 rows × 31 columns

In [232]:

```
a.isna()
```

Out[232]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...
564	False	False	False	False	False	False	False
565	False	False	False	False	False	False	False
566	False	False	False	False	False	False	False
567	False	False	False	False	False	False	False
568	False	False	False	False	False	False	False

569 rows × 32 columns

In [233]:

```
a.columns
```

Out[233]:

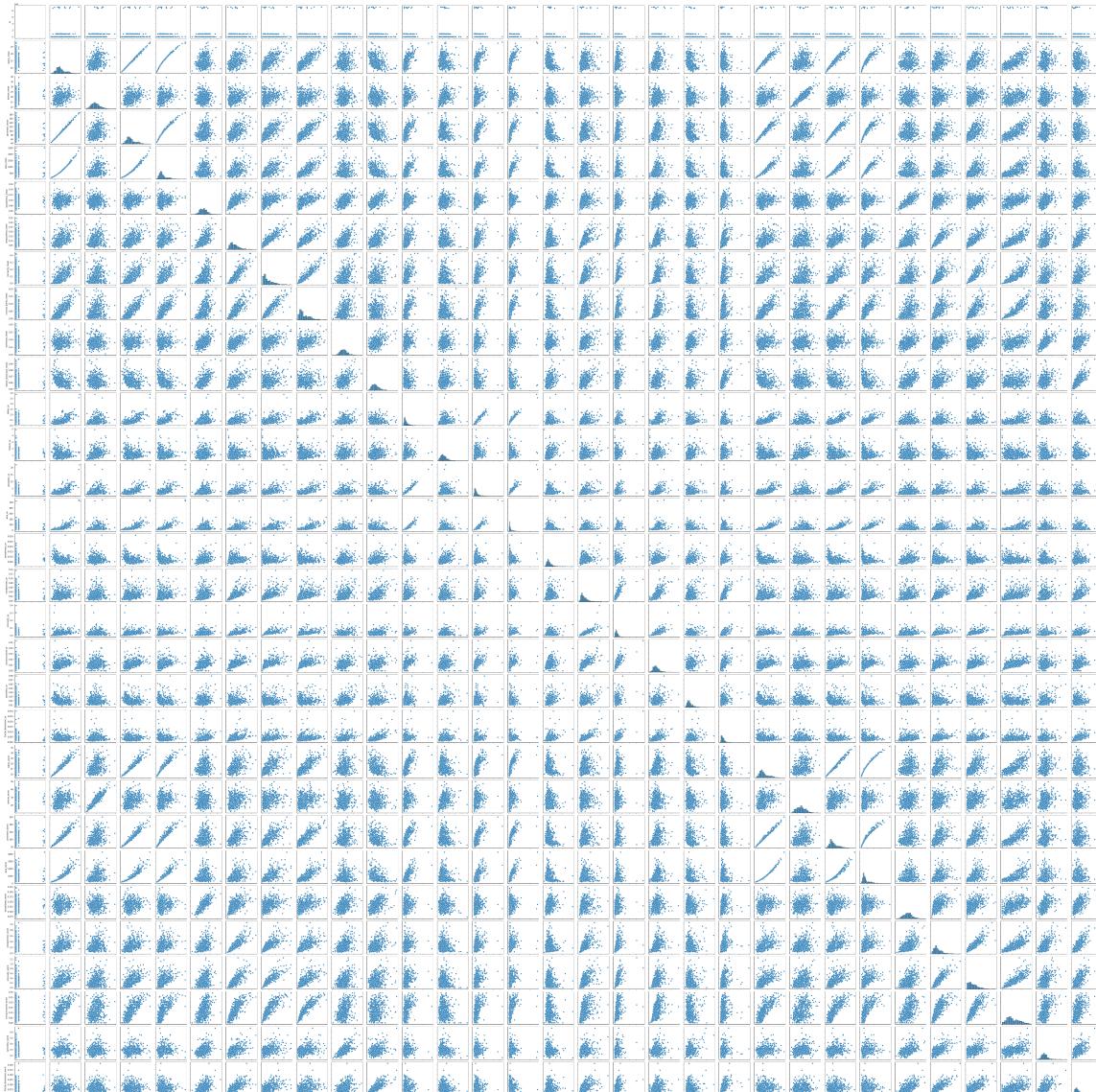
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       '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'],
      dtype='object')
```

In [234]:

```
sns.pairplot(a)
```

Out[234]:

```
<seaborn.axisgrid.PairGrid at 0x23a97d6c5e0>
```



In [235]:

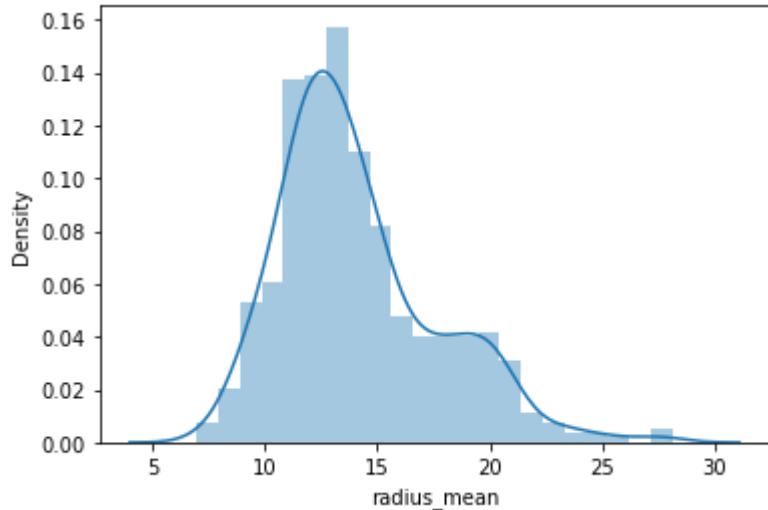
```
#normal distribution
sns.distplot(a["radius_mean"])
```

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[235]:

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

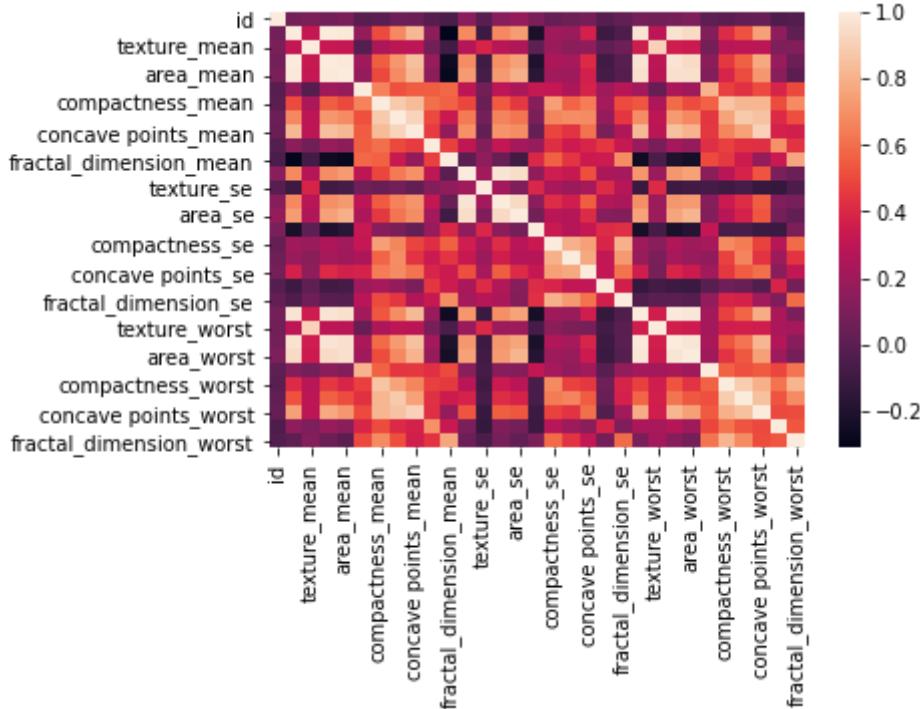


In [236]:

```
# Correlation map
sns.heatmap(a.corr())
```

Out[236]:

<AxesSubplot:>



In [239]:

```
x=a[['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']]
y=a['radius_mean']
```

In [240]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

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

Out[240]:

LinearRegression()

In [241]:

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

-2.149391775674303e-13

In [242]:

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

Out[242]:

Co-efficient	
id	1.263065e-20
radius_mean	1.000000e+00
texture_mean	6.144640e-17
perimeter_mean	9.190786e-16
area_mean	-6.337119e-16
smoothness_mean	-1.986086e-14
compactness_mean	-5.127124e-15
concavity_mean	5.627345e-15
concave points_mean	-8.631180e-15
symmetry_mean	7.953803e-17
fractal_dimension_mean	3.007222e-14
radius_se	6.762973e-15
texture_se	-5.479806e-16
perimeter_se	-1.846603e-16
area_se	-7.170486e-17
smoothness_se	2.323846e-14
compactness_se	1.012463e-14
concavity_se	-7.778514e-16
concave points_se	-4.489547e-15
symmetry_se	6.441871e-15
fractal_dimension_se	-6.623056e-14

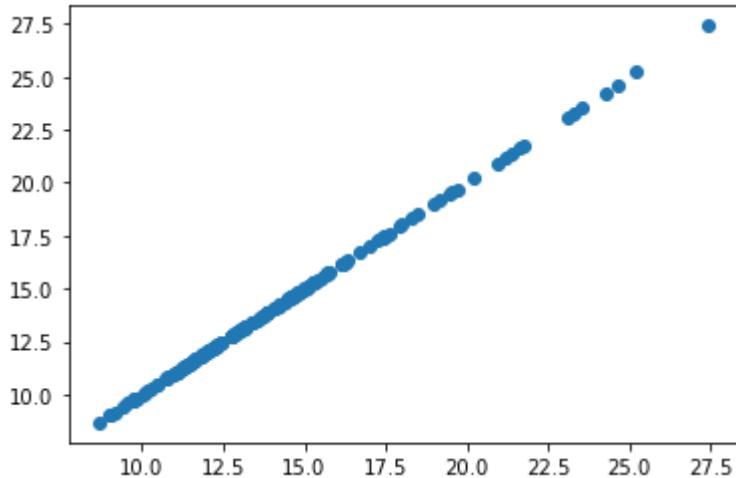
In [243]:

#Predicting

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

Out[243]:

<matplotlib.collections.PathCollection at 0x23ac4ef97f0>



In [244]:

Score

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

1.0

In []: