# 1. DataSet Breast Cancer

# DATA COLLECTION:

In [1]:

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

In [2]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\8_BreastCancerPrediction.csv")
a
```

Out[2]:

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

569 rows × 33 columns

In [3]:

```
b=a.head(10)
b
```

Out[3]:

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

10 rows × 33 columns

# DATA CLEANING AND PRE-PROCESSING

In [4]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       10 non-null     int64  
 1   diagnosis                10 non-null     object 
 2   radius_mean              10 non-null     float64
 3   texture_mean             10 non-null     float64
 4   perimeter_mean           10 non-null     float64
 5   area_mean                10 non-null     float64
 6   smoothness_mean          10 non-null     float64
 7   compactness_mean         10 non-null     float64
 8   concavity_mean           10 non-null     float64
 9   concave points_mean      10 non-null     float64
 10  symmetry_mean            10 non-null     float64
 11  fractal_dimension_mean   10 non-null     float64
 12  radius_se                10 non-null     float64
 13  texture_se               10 non-null     float64
 14  perimeter_se             10 non-null     float64
 15  area_se                  10 non-null     float64
 16  smoothness_se            10 non-null     float64
 17  compactness_se           10 non-null     float64
 18  concavity_se             10 non-null     float64
 19  concave points_se        10 non-null     float64
 20  symmetry_se              10 non-null     float64
 21  fractal_dimension_se     10 non-null     float64
 22  radius_worst             10 non-null     float64
 23  texture_worst            10 non-null     float64
 24  perimeter_worst          10 non-null     float64
 25  area_worst               10 non-null     float64
 26  smoothness_worst         10 non-null     float64
 27  compactness_worst        10 non-null     float64
 28  concavity_worst          10 non-null     float64
 29  concave points_worst     10 non-null     float64
 30  symmetry_worst           10 non-null     float64
 31  fractal_dimension_worst  10 non-null     float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 2.7+ KB
```

In [5]:

```
b.describe()
```

Out[5]:

| | id | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_ |
|---|---|---|---|---|---|---|
| **count** | 1.000000e+01 | 10.000000 | 10.00000 | 10.000000 | 10.000000 | 10.0( |
| **mean** | 4.261848e+07 | 15.983000 | 18.64900 | 106.222000 | 830.380000 | 0.1' |
| **std** | 4.403463e+07 | 3.686001 | 4.10719 | 23.680745 | 377.613035 | 0.0' |
| **min** | 8.423020e+05 | 11.420000 | 10.38000 | 77.580000 | 386.100000 | 0.0{ |
| **25%** | 8.439292e+05 | 12.595000 | 16.21750 | 84.852500 | 487.775000 | 0.1( |
| **50%** | 4.257294e+07 | 15.850000 | 20.18000 | 104.900000 | 789.450000 | 0.1' |
| **75%** | 8.435588e+07 | 19.330000 | 21.14500 | 128.200000 | 1162.250000 | 0.1: |
| **max** | 8.450100e+07 | 20.570000 | 24.04000 | 135.100000 | 1326.000000 | 0.1 |

8 rows × 32 columns

In [6]:

```
a.isna()
```

Out[6]:

| | id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_ |
|---|---|---|---|---|---|---|---|
| **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 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **564** | False | False | False | False | False | False | |
| **565** | False | False | False | False | False | False | |
| **566** | False | False | False | False | False | False | |
| **567** | False | False | False | False | False | False | |
| **568** | False | False | False | False | False | False | |

569 rows × 33 columns

In [7]:

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

Out[7]:

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

569 rows × 32 columns

In [8]:

```
d=b.head(100)
d
```

Out[8]:

| | id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothnes |
|---|---|---|---|---|---|---|---|
| **0** | 842302 | M | 17.990 | 10.38 | 122.80 | 1001.0 | |
| **1** | 842517 | M | 20.570 | 17.77 | 132.90 | 1326.0 | |
| **2** | 84300903 | M | 19.690 | 21.25 | 130.00 | 1203.0 | |
| **3** | 84348301 | M | 11.420 | 20.38 | 77.58 | 386.1 | |
| **4** | 84358402 | M | 20.290 | 14.34 | 135.10 | 1297.0 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **95** | 86208 | M | 20.260 | 23.03 | 132.40 | 1264.0 | |
| **96** | 86211 | B | 12.180 | 17.84 | 77.79 | 451.1 | |
| **97** | 862261 | B | 9.787 | 19.94 | 62.11 | 294.5 | |
| **98** | 862485 | B | 11.600 | 12.84 | 74.34 | 412.6 | |
| **99** | 862548 | M | 14.420 | 19.77 | 94.48 | 642.5 | |

100 rows × 32 columns

In [9]:

```
c=d.columns[2:10]
c
```

Out[9]:

```
Index(['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean'],
      dtype='object')
```

In [10]:

```
c1=d[['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean']]
c1
```

Out[10]:

| | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness |
|---|---|---|---|---|---|---|
| **0** | 17.990 | 10.38 | 122.80 | 1001.0 | 0.11840 | ( |
| **1** | 20.570 | 17.77 | 132.90 | 1326.0 | 0.08474 | ( |
| **2** | 19.690 | 21.25 | 130.00 | 1203.0 | 0.10960 | ( |
| **3** | 11.420 | 20.38 | 77.58 | 386.1 | 0.14250 | ( |
| **4** | 20.290 | 14.34 | 135.10 | 1297.0 | 0.10030 | ( |
| **...** | ... | ... | ... | ... | ... | |
| **95** | 20.260 | 23.03 | 132.40 | 1264.0 | 0.09078 | ( |
| **96** | 12.180 | 17.84 | 77.79 | 451.1 | 0.10450 | ( |
| **97** | 9.787 | 19.94 | 62.11 | 294.5 | 0.10240 | ( |
| **98** | 11.600 | 12.84 | 74.34 | 412.6 | 0.08983 | ( |
| **99** | 14.420 | 19.77 | 94.48 | 642.5 | 0.09752 | ( |

100 rows × 8 columns

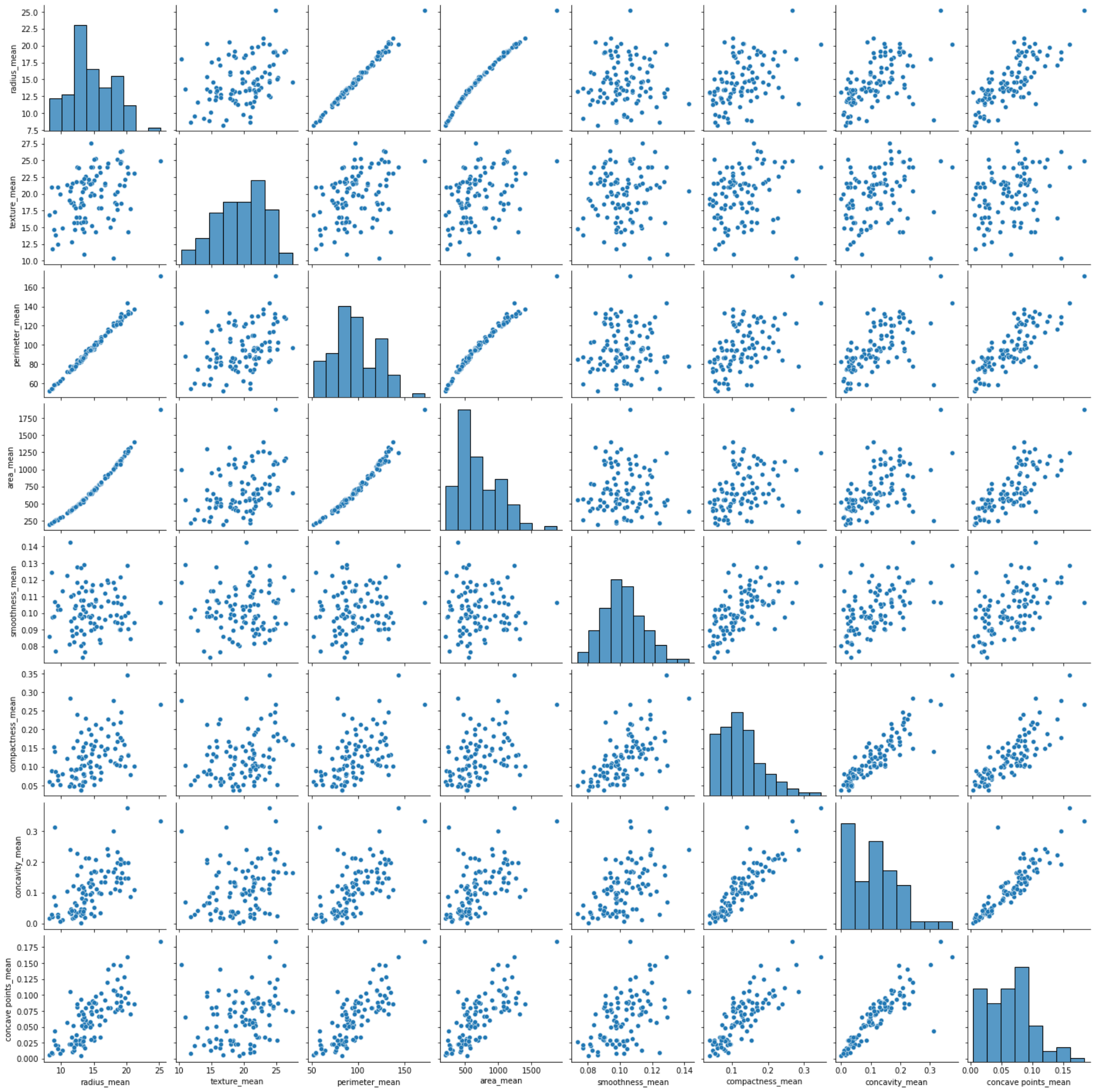
# EDA and VISUALIZATION

In [11]:

```
sns.pairplot(c1)
```

Out[11]:

<seaborn.axisgrid.PairGrid at 0x231c8695c40>



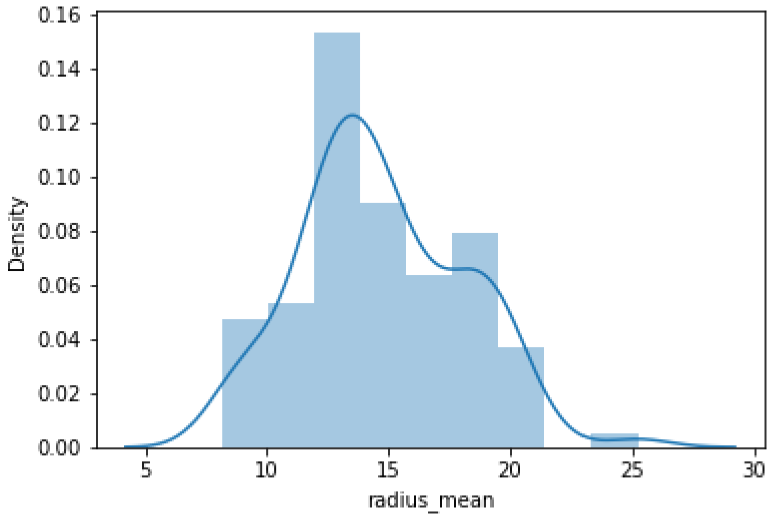
In [12]:

```
sns.distplot(c1['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[12]:

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

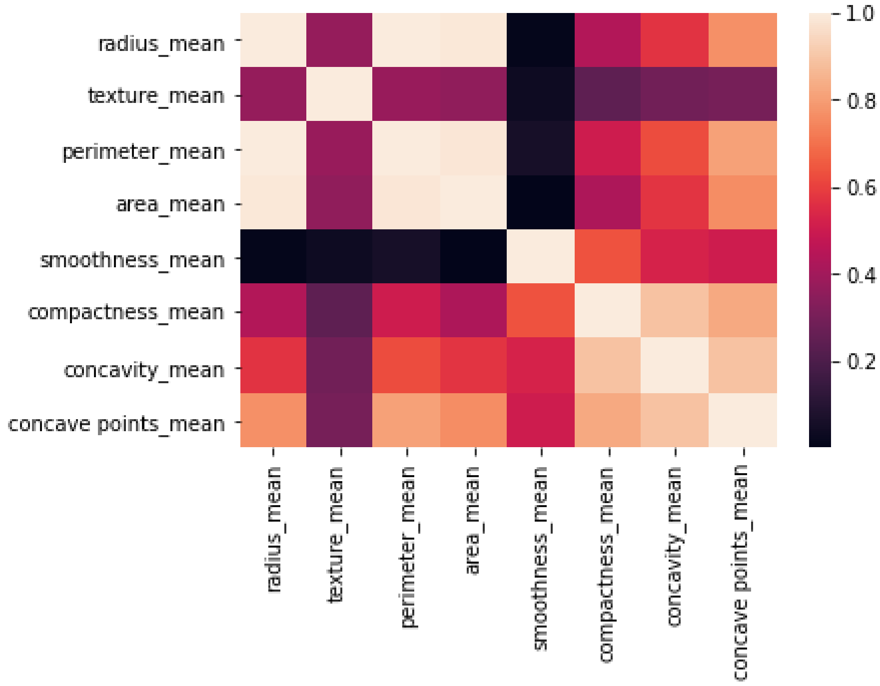



In [13]:

```
sns.heatmap(c1.corr())
```

Out[13]:

```
<AxesSubplot:>
```



In [14]:

```
x=d[['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean',
       'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean']]
y=d['perimeter_worst']
```

In [15]:

```
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 [16]:

```
from sklearn.linear_model import LinearRegression

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

Out[16]:

```
LinearRegression()
```

In [17]:

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

```
-3.8758647007806246
```

In [18]:

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

Out[18]:

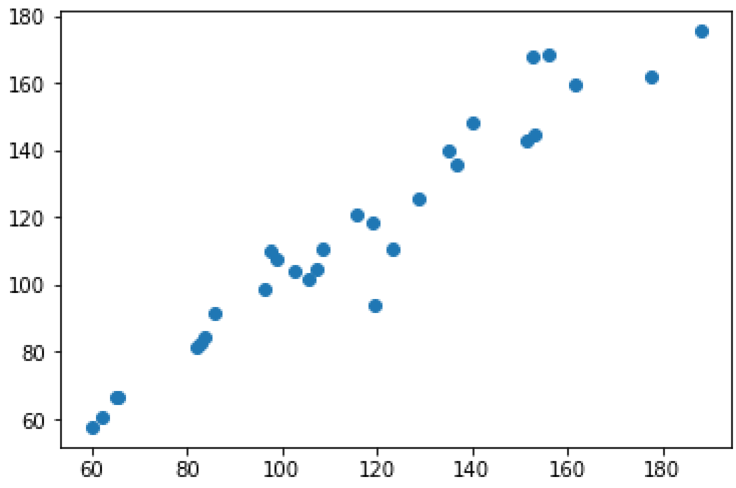
| | Co-efficient |
|---|---|
| **radius_mean** | 22.256808 |
| **texture_mean** | 0.072167 |
| **perimeter_mean** | -2.561876 |
| **area_mean** | 0.027773 |
| **smoothness_mean** | -68.500175 |
| **compactness_mean** | 140.786306 |
| **concavity_mean** | -0.063608 |
| **concave points_mean** | 130.048004 |

In [19]:

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

Out[19]:

<matplotlib.collections.PathCollection at 0x231cccc7250>




In [20]:

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

0.9378445550033725

In [21]:

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

In [22]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[22]:

Ridge(alpha=10)

In [23]:

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

Out[23]:

0.9318282923626972

In [24]:

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

Out[24]:

Lasso(alpha=10)

In [25]:

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

Out[25]:

```
0.9242125082436077
```

# 2. DataSet Uber

In [26]:

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

In [27]:

```
a1=pd.read_csv(r"C:\Users\user\Downloads\uber - uber.csv")
a1
```

Out[27]:

| | 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 [28]:

```
c1=a1.head(10)
c1
```

Out[28]:

| | Unnamed: 0 | key | fare_amount | pickup_datetime | pickup_longitude | pickup_latitude | drop |
|---|---|---|---|---|---|---|---|
| **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 | |
| **5** | 44470845 | 2011-02-12 2:27:09 | 4.9 | 2011-02-12 02:27:09 UTC | -73.969019 | 40.755910 | |
| **6** | 48725865 | 2014-10-12 7:04:00 | 24.5 | 2014-10-12 07:04:00 UTC | -73.961447 | 40.693965 | |
| **7** | 44195482 | 2012-12-11 13:52:00 | 2.5 | 2012-12-11 13:52:00 UTC | 0.000000 | 0.000000 | |
| **8** | 15822268 | 2012-02-17 9:32:00 | 9.7 | 2012-02-17 09:32:00 UTC | -73.975187 | 40.745767 | |
| **9** | 50611056 | 2012-03-29 19:06:00 | 12.5 | 2012-03-29 19:06:00 UTC | -74.001065 | 40.741787 | |

In [29]:

```
a1.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 [30]:

```
a1.describe()
```

Out[30]:

| | Unnamed: 0 | fare_amount | pickup_longitude | pickup_latitude | dropoff_longitude | dro |
|---|---|---|---|---|---|---|
| **count** | 2.000000e+05 | 200000.000000 | 200000.000000 | 200000.000000 | 199999.000000 | 19 |
| **mean** | 2.771250e+07 | 11.359955 | -72.527638 | 39.935885 | -72.525292 | |
| **std** | 1.601382e+07 | 9.901776 | 11.437787 | 7.720539 | 13.117408 | |
| **min** | 1.000000e+00 | -52.000000 | -1340.648410 | -74.015515 | -3356.666300 | |
| **25%** | 1.382535e+07 | 6.000000 | -73.992065 | 40.734796 | -73.991407 | |
| **50%** | 2.774550e+07 | 8.500000 | -73.981823 | 40.752592 | -73.980093 | |
| **75%** | 4.155530e+07 | 12.500000 | -73.967153 | 40.767158 | -73.963659 | |
| **max** | 5.542357e+07 | 499.000000 | 57.418457 | 1644.421482 | 1153.572603 | |

In [31]:

```
a1.columns
```

Out[31]:

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

In [32]:

```
b1=c1[['pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
b1
```

Out[32]:

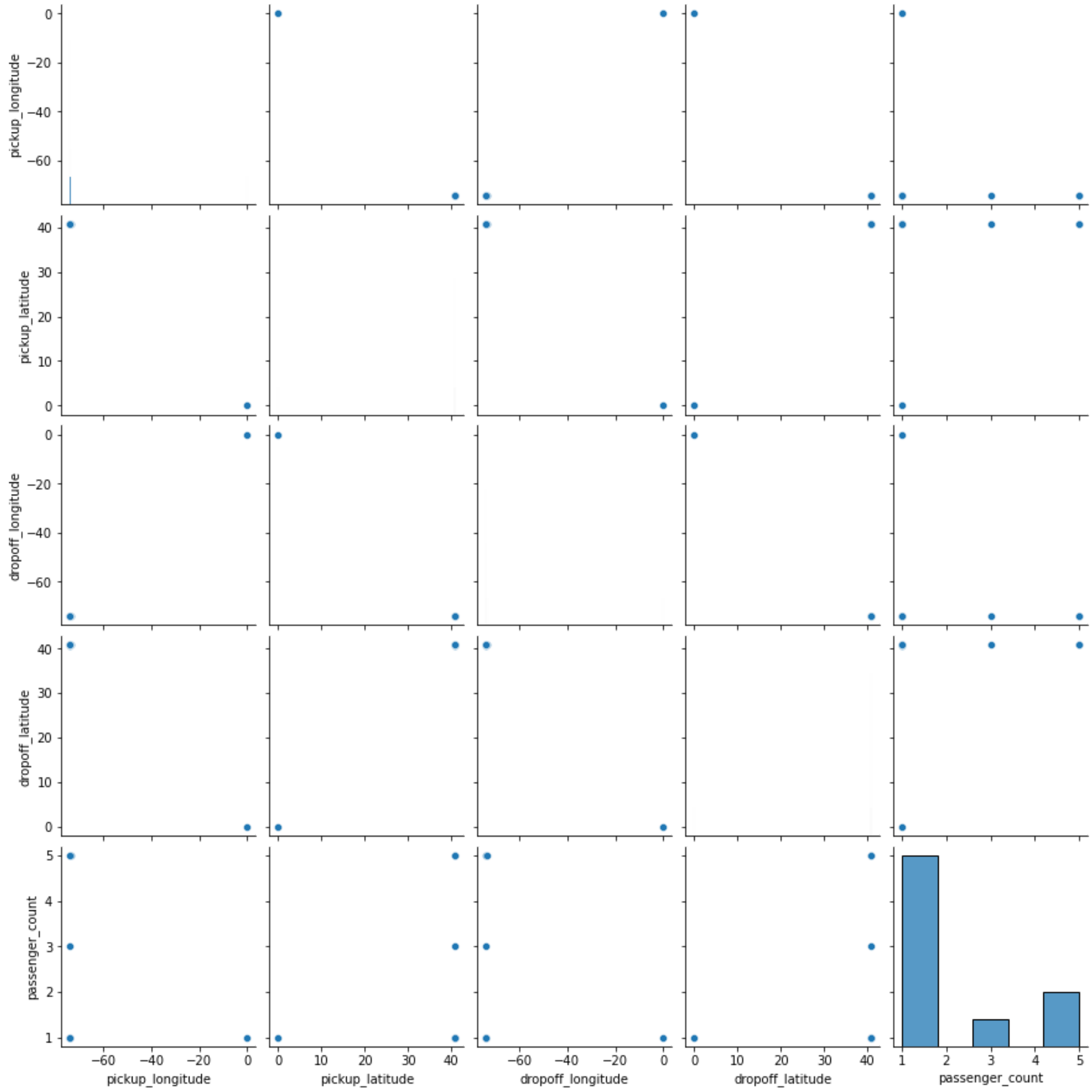
| | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count |
|---|---|---|---|---|---|
| **0** | -73.999817 | 40.738354 | -73.999512 | 40.723217 | 1 |
| **1** | -73.994355 | 40.728225 | -73.994710 | 40.750325 | 1 |
| **2** | -74.005043 | 40.740770 | -73.962565 | 40.772647 | 1 |
| **3** | -73.976124 | 40.790844 | -73.965316 | 40.803349 | 3 |
| **4** | -73.925023 | 40.744085 | -73.973082 | 40.761247 | 5 |
| **5** | -73.969019 | 40.755910 | -73.969019 | 40.755910 | 1 |
| **6** | -73.961447 | 40.693965 | -73.871195 | 40.774297 | 5 |
| **7** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 1 |
| **8** | -73.975187 | 40.745767 | -74.002720 | 40.743537 | 1 |
| **9** | -74.001065 | 40.741787 | -73.963040 | 40.775012 | 1 |

In [33]:

```
sns.pairplot(b1)
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0x231d25ff490>
```



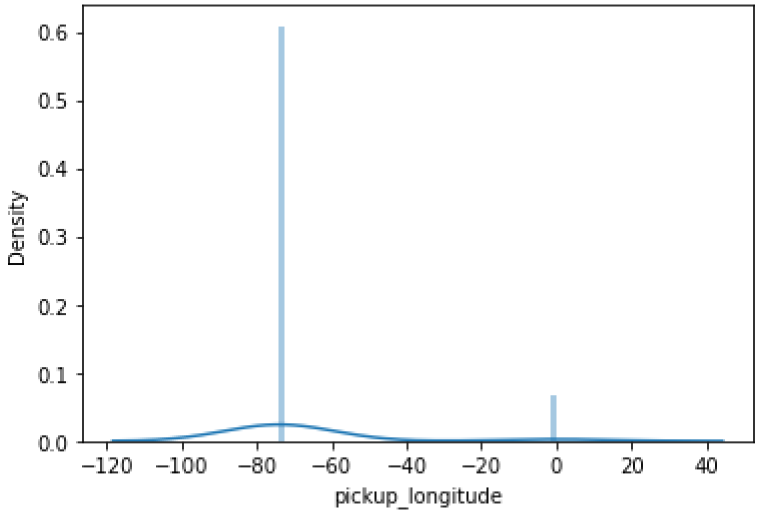
In [34]:

```
sns.distplot(b1['pickup_longitude'])
```

```
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[34]:

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




In [35]:

```
x1=c1[['pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count']]
y1=c1['fare_amount']
```

In [36]:

```
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.3)
```

In [37]:

```
lr=LinearRegression()
lr.fit(x1_train,y1_train)
```

Out[37]:

```
LinearRegression()
```

In [38]:

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

```
17646.223954229048
```

In [39]:

```
coe1=pd.DataFrame(lr.coef_,x1.columns,columns=['Co-efficient'])
coe1
```

Out[39]:

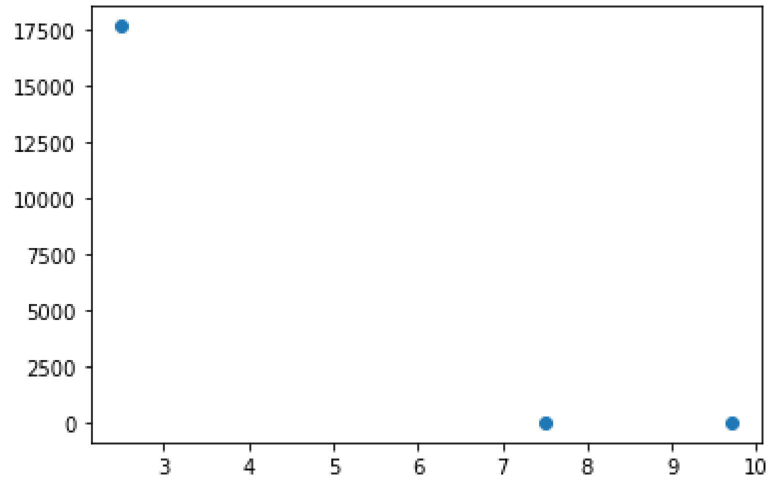
| | Co-efficient |
|---|---|
| **pickup_longitude** | 488.144582 |
| **pickup_latitude** | -630.670006 |
| **dropoff_longitude** | -143.720128 |
| **dropoff_latitude** | 823.074354 |
| **passenger_count** | -5.987244 |

In [40]:

```
prediction=lr.predict(x1_test)
plt.scatter(y1_test,prediction)
```

Out[40]:

<matplotlib.collections.PathCollection at 0x231d7994ee0>




In [41]:

```
print(lr.score(x1_test,y1_test))
```

-11425952.705272354

In [42]:

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

In [43]:

```
rr=Ridge(alpha=10)
rr.fit(x1_train,y1_train)
```

Out[43]:

Ridge(alpha=10)

In [44]:

```
rr.score(x1_test,y1_test)
```

Out[44]:

```
-9.435149906804877
```

In [45]:

```
la=Lasso(alpha=10)
la.fit(x1_train,y1_train)
```

Out[45]:

```
Lasso(alpha=10)
```

In [46]:

```
la.score(x1_test,y1_test)
```

Out[46]:

```
-3.2186956565192224
```

# 3. DataSet SalesWorkload

In [47]:

```
a2=pd.read_csv(r"C:\Users\user\Downloads\6_Salesworkload1.csv")
a2
```

Out[47]:

| | 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 [48]:

```
b2=a2.fillna(value=17)
b2
```

Out[48]:

| | 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 [49]:

```
c2=b2.head(10)
c2
```

Out[49]:

| | 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 |
| **1** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 2.0 | Frozen | 1582.941 | 0.0 |
| **2** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 3.0 | other | 47.205 | 0.0 |
| **3** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 4.0 | Fish | 1623.852 | 0.0 |
| **4** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 5.0 | Fruits & Vegetables | 1759.173 | 0.0 |
| **5** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 6.0 | Meat | 8270.316 | 0.0 |
| **6** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 13.0 | Food | 16468.251 | 0.0 |
| **7** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 7.0 | Clothing | 4698.471 | 0.0 |
| **8** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 8.0 | Household | 1183.272 | 0.0 |
| **9** | 10.2016 | 1.0 | United Kingdom | 88253.0 | London (I) | 9.0 | Hardware | 2029.815 | 0.0 |

In [50]:

```
c2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MonthYear      10 non-null     object
 1   Time index     10 non-null     float64
 2   Country        10 non-null     object
 3   StoreID        10 non-null     float64
 4   City           10 non-null     object
 5   Dept_ID        10 non-null     float64
 6   Dept. Name     10 non-null     object
 7   HoursOwn       10 non-null     object
 8   HoursLease     10 non-null     float64
 9   Sales units    10 non-null     float64
 10  Turnover       10 non-null     float64
 11  Customer       10 non-null     float64
 12  Area (m2)      10 non-null     object
 13  Opening hours  10 non-null     object
dtypes: float64(7), object(7)
memory usage: 1.2+ KB
```

In [51]:

```
c2.describe()
```

Out[51]:

| | Time index | StoreID | Dept_ID | HoursLease | Sales units | Turnover | Customer |
|---|---|---|---|---|---|---|---|
| **count** | 10.0 | 10.0 | 10.000000 | 10.0 | 1.000000e+01 | 1.000000e+01 | 10.0 |
| **mean** | 1.0 | 88253.0 | 5.800000 | 0.0 | 6.543725e+05 | 1.978511e+06 | 17.0 |
| **std** | 0.0 | 0.0 | 3.614784 | 0.0 | 9.914003e+05 | 2.861420e+06 | 0.0 |
| **min** | 1.0 | 88253.0 | 1.000000 | 0.0 | 5.491500e+04 | 2.904000e+05 | 17.0 |
| **25%** | 1.0 | 88253.0 | 3.250000 | 0.0 | 1.034225e+05 | 4.033612e+05 | 17.0 |
| **50%** | 1.0 | 88253.0 | 5.500000 | 0.0 | 2.615525e+05 | 5.770455e+05 | 17.0 |
| **75%** | 1.0 | 88253.0 | 7.750000 | 0.0 | 4.284400e+05 | 1.518067e+06 | 17.0 |
| **max** | 1.0 | 88253.0 | 13.000000 | 0.0 | 3.107935e+06 | 8.714679e+06 | 17.0 |

In [52]:

```
c2.columns
```

Out[52]:

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

In [53]:

```
d2=c2[['Time index','StoreID','HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)']]
d2
```

Out[53]:

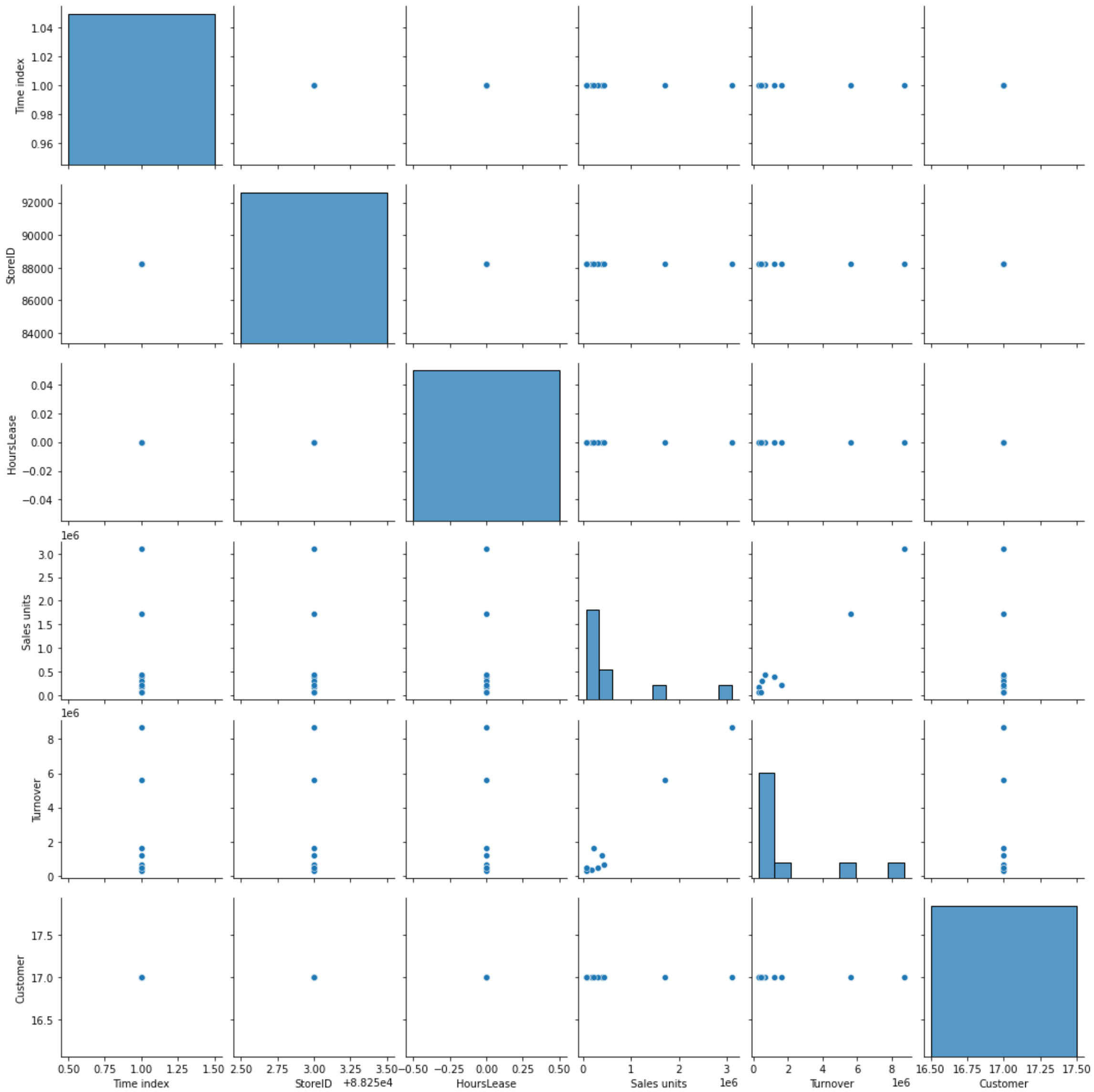
| | Time index | StoreID | HoursLease | Sales units | Turnover | Customer | Area (m2) |
|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 88253.0 | 0.0 | 398560.0 | 1226244.0 | 17.0 | 953.04 |
| **1** | 1.0 | 88253.0 | 0.0 | 82725.0 | 387810.0 | 17.0 | 720.48 |
| **2** | 1.0 | 88253.0 | 0.0 | 438400.0 | 654657.0 | 17.0 | 966.72 |
| **3** | 1.0 | 88253.0 | 0.0 | 309425.0 | 499434.0 | 17.0 | 1053.36 |
| **4** | 1.0 | 88253.0 | 0.0 | 165515.0 | 329397.0 | 17.0 | 1053.36 |
| **5** | 1.0 | 88253.0 | 0.0 | 1713310.0 | 5617137.0 | 17.0 | 11735.16 |
| **6** | 1.0 | 88253.0 | 0.0 | 3107935.0 | 8714679.0 | 17.0 | 19865.64 |
| **7** | 1.0 | 88253.0 | 0.0 | 213680.0 | 1615341.0 | 17.0 | 8513.52 |
| **8** | 1.0 | 88253.0 | 0.0 | 54915.0 | 290400.0 | 17.0 | 4842.72 |
| **9** | 1.0 | 88253.0 | 0.0 | 59260.0 | 450015.0 | 17.0 | 5608.8 |

In [54]:

```
sns.pairplot(d2)
```

Out[54]:

<seaborn.axisgrid.PairGrid at 0x231d7ac91f0>



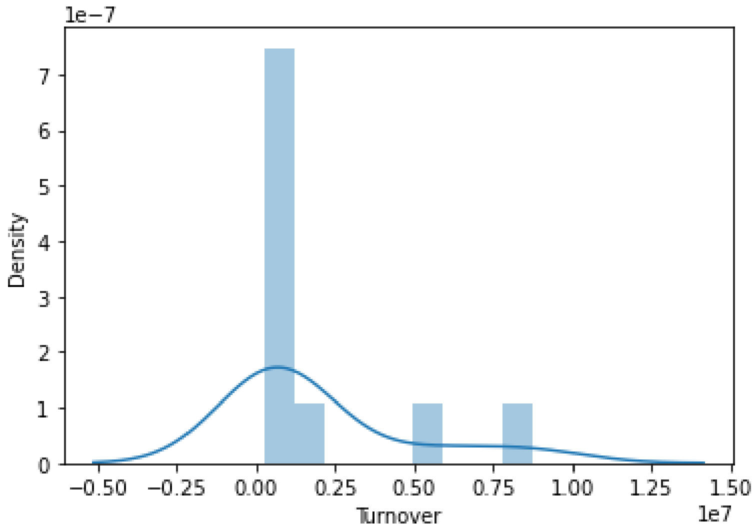
In [55]:

```
sns.distplot(d2['Turnover'])
```

```
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[55]:

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

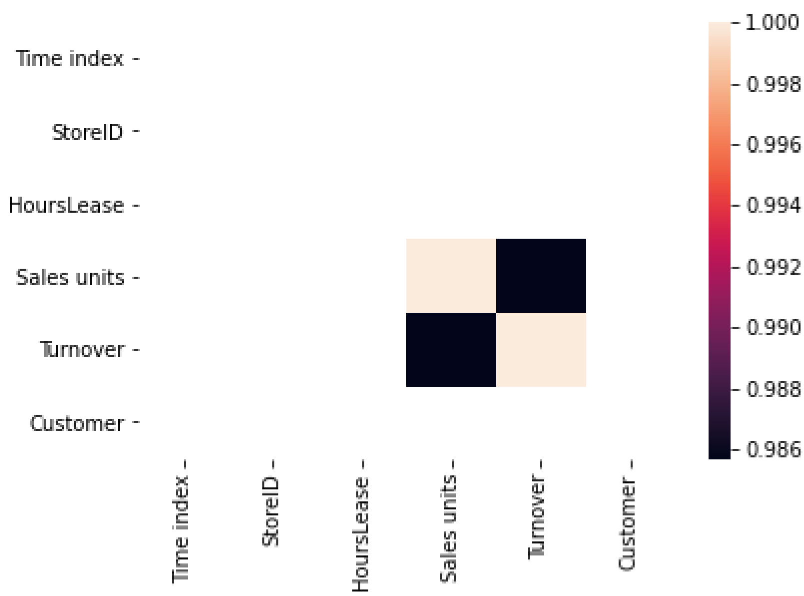



In [56]:

```
sns.heatmap(d2.corr())
```

Out[56]:

```
<AxesSubplot:>
```



In [57]:

```
x2=d2[['Time index','StoreID','HoursLease', 'Sales units', 'Turnover',
       'Customer', 'Area (m2)']]
y2=c2['Dept_ID']
```

In [58]:

```
x2_train,x2_test,y2_train,y2_test=train_test_split(x2,y2,test_size=0.3)
```

In [59]:

```
lr=LinearRegression()
lr.fit(x2_train,y2_train)
```

Out[59]:

```
LinearRegression()
```

In [60]:

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

```
2.2472590733374114
```

In [61]:

```
coeff=pd.DataFrame(lr.coef_,x2.columns,columns=['Co-efficient'])
coeff
```

Out[61]:

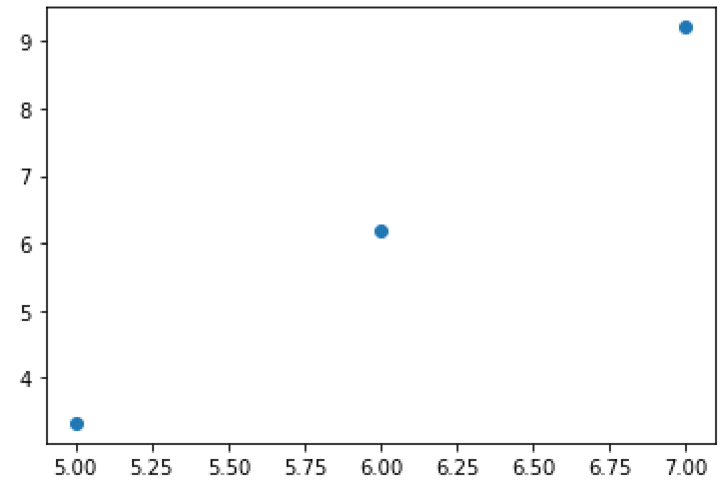
| | Co-efficient |
|---|---|
| **Time index** | 0.000000e+00 |
| **StoreID** | 1.792322e-17 |
| **HoursLease** | 1.409463e-18 |
| **Sales units** | 5.163565e-06 |
| **Turnover** | -3.834808e-06 |
| **Customer** | 0.000000e+00 |
| **Area (m2)** | 1.415987e-03 |

In [62]:

```
prediction=lr.predict(x2_test)
plt.scatter(y2_test,prediction)
```

Out[62]:

```
<matplotlib.collections.PathCollection at 0x231dc228640>
```




In [63]:

```
print(lr.score(x2_test,y2_test))
```

```
-2.85300720094631
```

In [64]:

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

In [65]:

```
rr=Ridge(alpha=10)
rr.fit(x2_train,y2_train)
```

Out[65]:

```
Ridge(alpha=10)
```

In [66]:

```
rr.score(x2_test,y2_test)
```

Out[66]:

```
-2.853007279806269
```

In [67]:

```
la=Lasso(alpha=10)
la.fit(x2_train,y2_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_coordinat
e_descent.py:530: ConvergenceWarning: Objective did not converge. You migh
t want to increase the number of iterations. Duality gap: 0.35076343700064
25, tolerance: 0.0115428571428571 43
  model = cd_fast.enet_coordinate_descent(
```

Out[67]:

```
Lasso(alpha=10)
```

In [68]:

```
la.score(x2_test,y2_test)
```

Out[68]:

```
-2.8637964269888623
```

# 4. DataSet Instagram

In [69]:

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

Out[69]:

| | 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 | 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 |
| **118** | 36919 | 13473 | 4176 | 16444 | 2547 | 653 | 5 | 26 | 443 | 611 |

119 rows × 13 columns

In [70]:

```
b=a.head(10)
b
```

Out[70]:

| | 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 | 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 |
| **5** | 3884 | 2046 | 1214 | 329 | 43 | 74 | 7 | 10 | 144 | 9 |
| **6** | 2621 | 1543 | 599 | 333 | 25 | 22 | 5 | 1 | 76 | 26 |
| **7** | 3541 | 2071 | 628 | 500 | 60 | 135 | 4 | 9 | 124 | 12 |
| **8** | 3749 | 2384 | 857 | 248 | 49 | 155 | 6 | 8 | 159 | 36 |
| **9** | 4115 | 2609 | 1104 | 178 | 46 | 122 | 6 | 3 | 191 | 31 |

In [71]:

```
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 [72]:

```
a.describe()
```

Out[72]:

| | Impressions | From Home | From Hashtags | From Explore | From Other | Saves | Co |
|---|---|---|---|---|---|---|---|
| **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 [73]:

```
a.columns
```

Out[73]:

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

In [74]:

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

Out[74]:

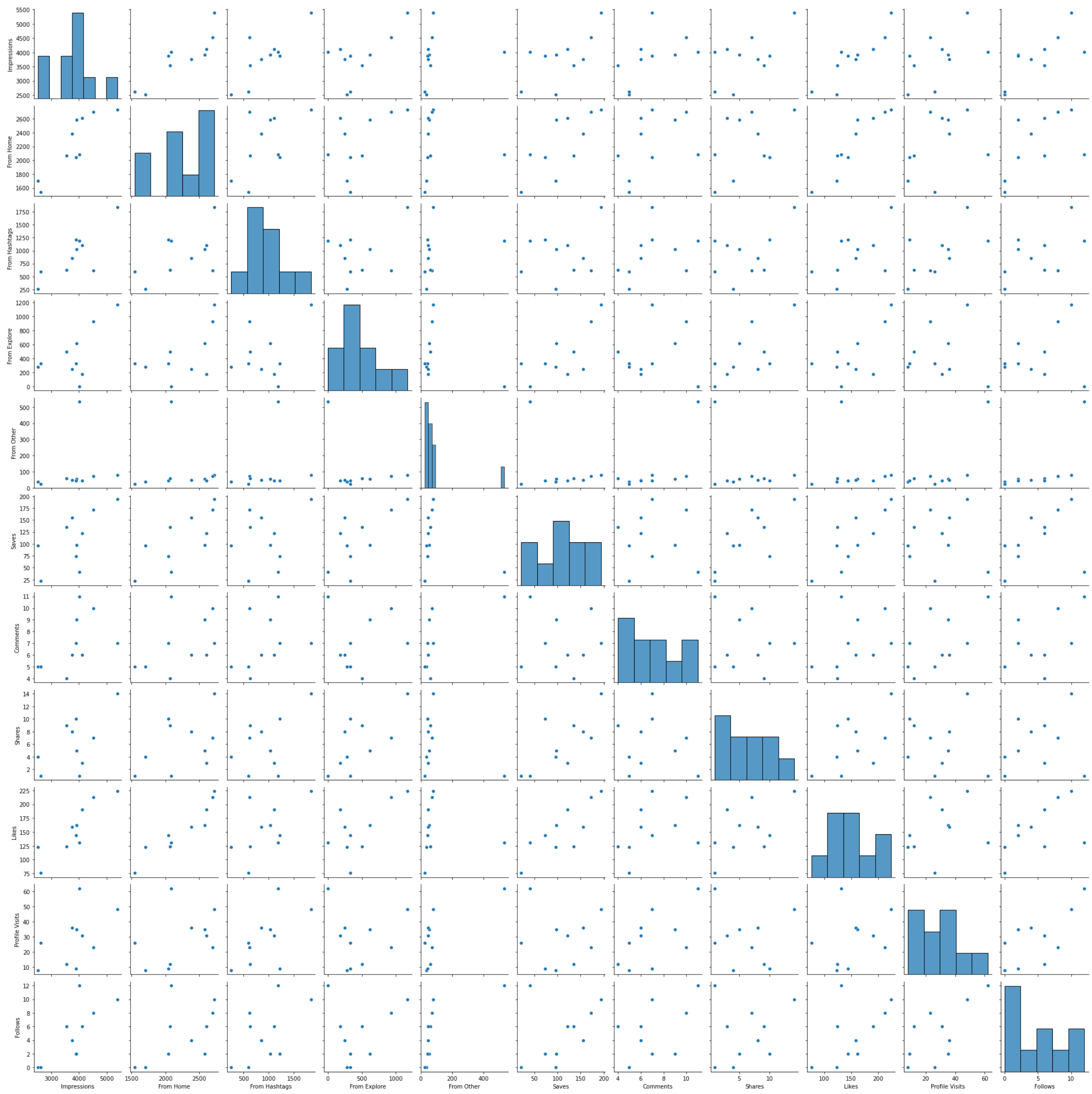
| | Impressions | From Home | From Hashtags | From Explore | From Other | Saves | Comments | Shares | Likes | Profile Visits |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 3920 | 2586 | 1028 | 619 | 56 | 98 | 9 | 5 | 162 | 35 |
| **1** | 5394 | 2727 | 1838 | 1174 | 78 | 194 | 7 | 14 | 224 | 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 |
| **5** | 3884 | 2046 | 1214 | 329 | 43 | 74 | 7 | 10 | 144 | 9 |
| **6** | 2621 | 1543 | 599 | 333 | 25 | 22 | 5 | 1 | 76 | 26 |
| **7** | 3541 | 2071 | 628 | 500 | 60 | 135 | 4 | 9 | 124 | 12 |
| **8** | 3749 | 2384 | 857 | 248 | 49 | 155 | 6 | 8 | 159 | 36 |
| **9** | 4115 | 2609 | 1104 | 178 | 46 | 122 | 6 | 3 | 191 | 31 |

In [75]:

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

Out[75]:

<seaborn.axisgrid.PairGrid at 0x231d79dfd60>



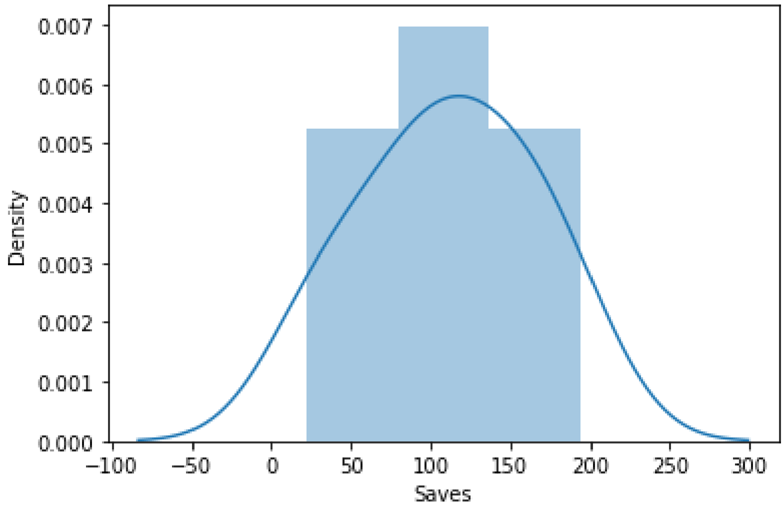
In [76]:

```
sns.distplot(c['Saves'])
```

```
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 function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[76]:

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

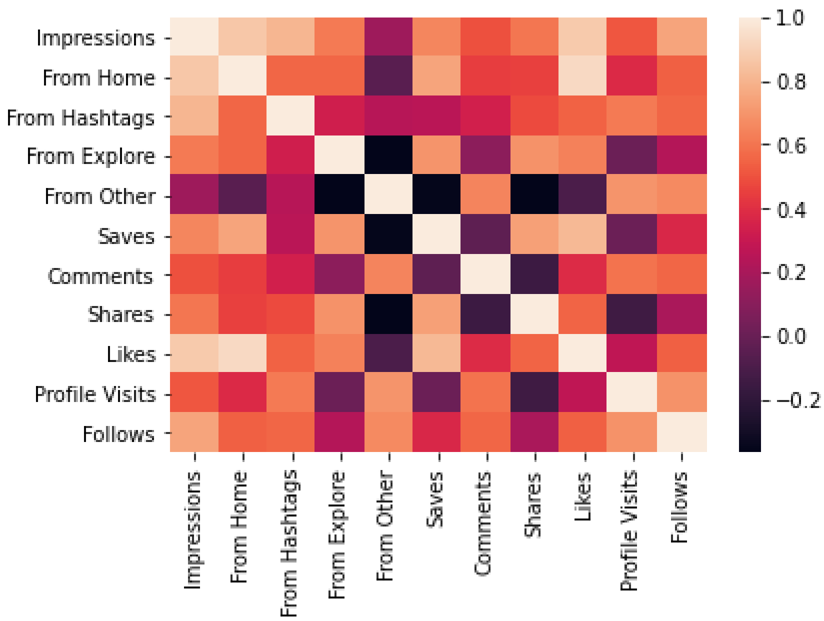



In [77]:

```
sns.heatmap(c.corr())
```

Out[77]:

```
<AxesSubplot:>
```



In [78]:

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

In [79]:

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

In [80]:

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

Out[80]:

```
LinearRegression()
```

In [81]:

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

```
14.728575896975386
```

In [82]:

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

Out[82]:

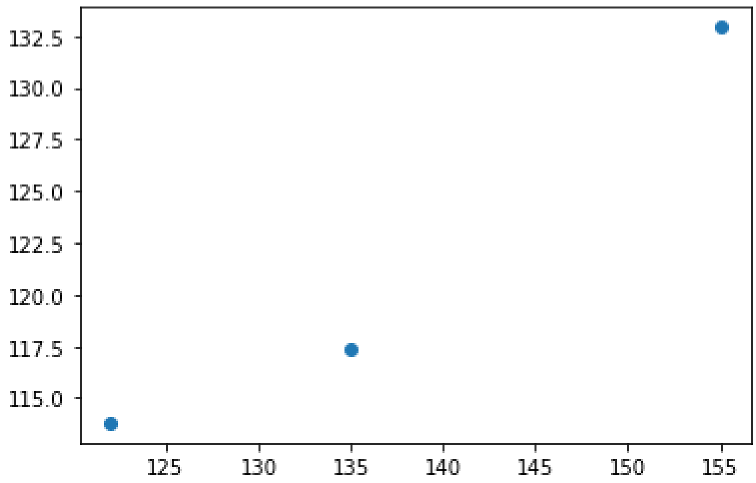
| | Co-efficient |
|---|---|
| **Impressions** | -0.015035 |
| **From Home** | -0.007878 |
| **From Hashtags** | 0.008178 |
| **From Explore** | 0.026761 |
| **From Other** | 0.031747 |
| **Saves** | 0.761812 |
| **Comments** | -0.018727 |
| **Shares** | 0.051975 |
| **Likes** | 0.402693 |
| **Profile Visits** | -0.118908 |
| **Follows** | 0.004743 |

In [83]:

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

Out[83]:

```
<matplotlib.collections.PathCollection at 0x231e1833520>
```




In [84]:

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

```
-0.5655413671813168
```

In [85]:

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

In [86]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[86]:

```
Ridge(alpha=10)
```

In [87]:

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

Out[87]:

```
-0.5847572731952695
```

In [88]:

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

Out[88]:

```
Lasso(alpha=10)
```

In [89]:

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

Out[89]:

```
0.9913405617952354
```

# 5. DataSet Drug

In [90]:

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

Out[90]:

| | 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 [91]:

```
b=a.head(10)
b
```

Out[91]:

| | 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 [92]:

```
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 [93]:

```
a.describe()
```

Out[93]:

| | 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 [94]:

```
a.columns
```

Out[94]:

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

In [95]:

```
c=b[['Age','Na_to_K']]
c
```

Out[95]:

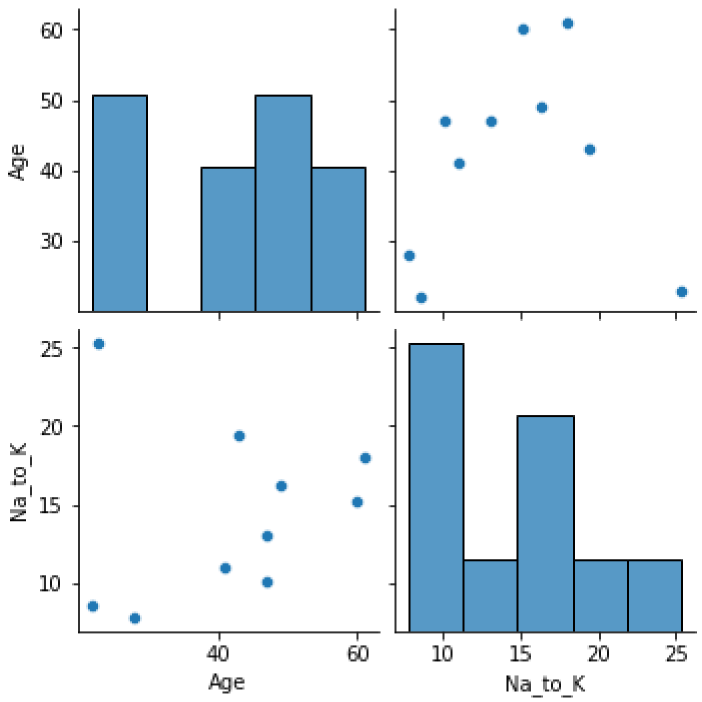
| | Age | Na_to_K |
|---|---|---|
| **0** | 23 | 25.355 |
| **1** | 47 | 13.093 |
| **2** | 47 | 10.114 |
| **3** | 28 | 7.798 |
| **4** | 61 | 18.043 |
| **5** | 22 | 8.607 |
| **6** | 49 | 16.275 |
| **7** | 41 | 11.037 |
| **8** | 60 | 15.171 |
| **9** | 43 | 19.368 |

In [96]:

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

Out[96]:

```
<seaborn.axisgrid.PairGrid at 0x231e187c220>
```



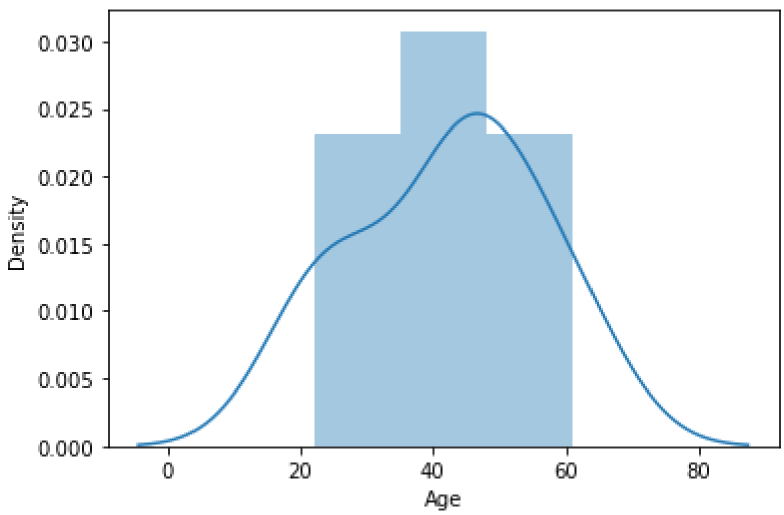

In [97]:

```
sns.distplot(c['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[97]:

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

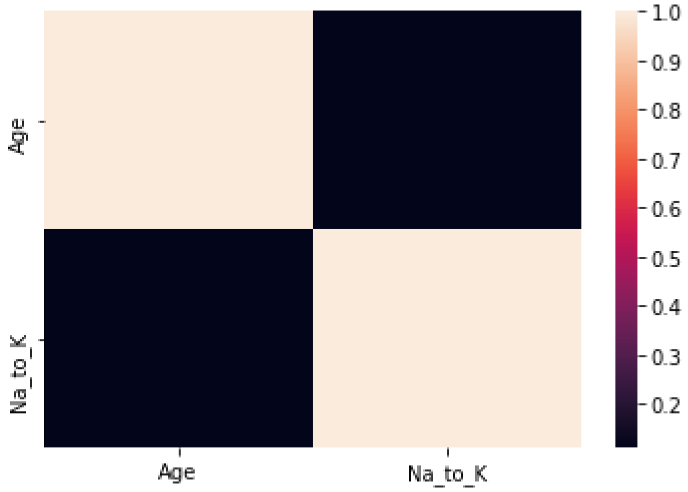


In [98]:

```
sns.heatmap(c.corr())
```

Out[98]:

<AxesSubplot:>




In [99]:

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

In [100]:

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

In [101]:

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

Out[101]:

LinearRegression()

In [102]:

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

7.105427357601002e-15

In [103]:

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

Out[103]:

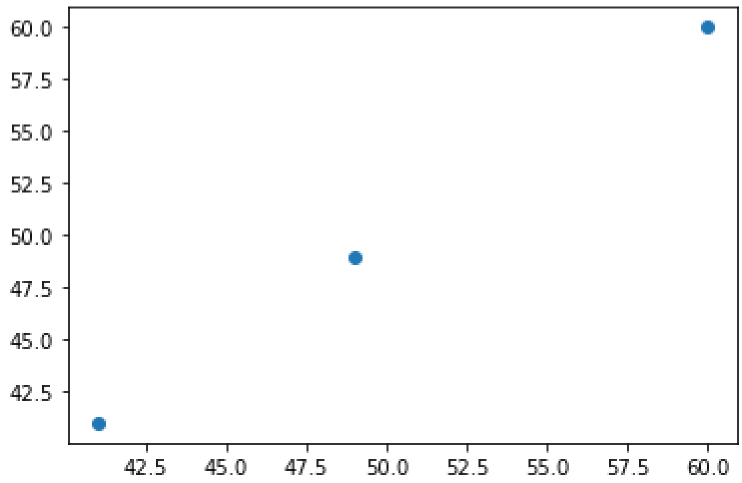
| | Co-efficient |
|---|---|
| **Age** | 1.000000e+00 |
| **Na_to_K** | 3.704420e-17 |

In [104]:

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

Out[104]:

<matplotlib.collections.PathCollection at 0x231e2b0ff70>




In [105]:

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

1.0

In [106]:

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

In [107]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[107]:

Ridge(alpha=10)

In [108]:

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

Out[108]:

0.9998171297144919

In [109]:

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

Out[109]:

Lasso(alpha=10)

In [110]:

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

Out[110]:

```
0.99092184773655922
```

# 6. DataSet Vehicle

In [111]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\Vehicle.csv")
a
```

Out[111]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | lounge | 51.0 | 882.0 | 25000.0 | 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** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **1545** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **1546** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | Null |
| **1547** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| **1548** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

1549 rows × 11 columns

In [112]:

```
b=a.head(10)
b
```

Out[112]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | l |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | lounge | 51.0 | 882.0 | 25000.0 | 1.0 | 44.907242 | 8.611559ε |
| **1** | 2.0 | pop | 51.0 | 1186.0 | 32500.0 | 1.0 | 45.666359 | 12.24188ς |
| **2** | 3.0 | sport | 74.0 | 4658.0 | 142228.0 | 1.0 | 45.503300 | 11.417 |
| **3** | 4.0 | lounge | 51.0 | 2739.0 | 160000.0 | 1.0 | 40.633171 | 17.63460ς |
| **4** | 5.0 | pop | 73.0 | 3074.0 | 106880.0 | 1.0 | 41.903221 | 12.49565C |
| **5** | 6.0 | pop | 74.0 | 3623.0 | 70225.0 | 1.0 | 45.000702 | 7.68227C |
| **6** | 7.0 | lounge | 51.0 | 731.0 | 11600.0 | 1.0 | 44.907242 | 8.611559ε |
| **7** | 8.0 | lounge | 51.0 | 1521.0 | 49076.0 | 1.0 | 41.903221 | 12.49565C |
| **8** | 9.0 | sport | 73.0 | 4049.0 | 76000.0 | 1.0 | 45.548000 | 11.54946ς |
| **9** | 10.0 | sport | 51.0 | 3653.0 | 89000.0 | 1.0 | 45.438301 | 10.99170C |

In [113]:

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

Out[113]:

| | ID | model | engine_power | age_in_days | km | previous_owners | lat | l |
|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | lounge | 51.0 | 882.0 | 25000.0 | 1.0 | 44.907242 | 8.611559ε |
| **1** | 2.0 | pop | 51.0 | 1186.0 | 32500.0 | 1.0 | 45.666359 | 12.24188ς |
| **2** | 3.0 | sport | 74.0 | 4658.0 | 142228.0 | 1.0 | 45.503300 | 11.417 |
| **3** | 4.0 | lounge | 51.0 | 2739.0 | 160000.0 | 1.0 | 40.633171 | 17.63460ς |
| **4** | 5.0 | pop | 73.0 | 3074.0 | 106880.0 | 1.0 | 41.903221 | 12.49565C |
| **5** | 6.0 | pop | 74.0 | 3623.0 | 70225.0 | 1.0 | 45.000702 | 7.68227C |
| **6** | 7.0 | lounge | 51.0 | 731.0 | 11600.0 | 1.0 | 44.907242 | 8.611559ε |
| **7** | 8.0 | lounge | 51.0 | 1521.0 | 49076.0 | 1.0 | 41.903221 | 12.49565C |
| **8** | 9.0 | sport | 73.0 | 4049.0 | 76000.0 | 1.0 | 45.548000 | 11.54946ς |
| **9** | 10.0 | sport | 51.0 | 3653.0 | 89000.0 | 1.0 | 45.438301 | 10.99170C |

In [114]:

```
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 [115]:

```
a.describe()
```

Out[115]:

| | ID | engine_power | age_in_days | km | previous_owners | la |
|---|---|---|---|---|---|---|
| **count** | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.000000 | 1538.00000 |
| **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 [116]:

```
a.columns
```

Out[116]:

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

In [117]:

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

Out[117]:

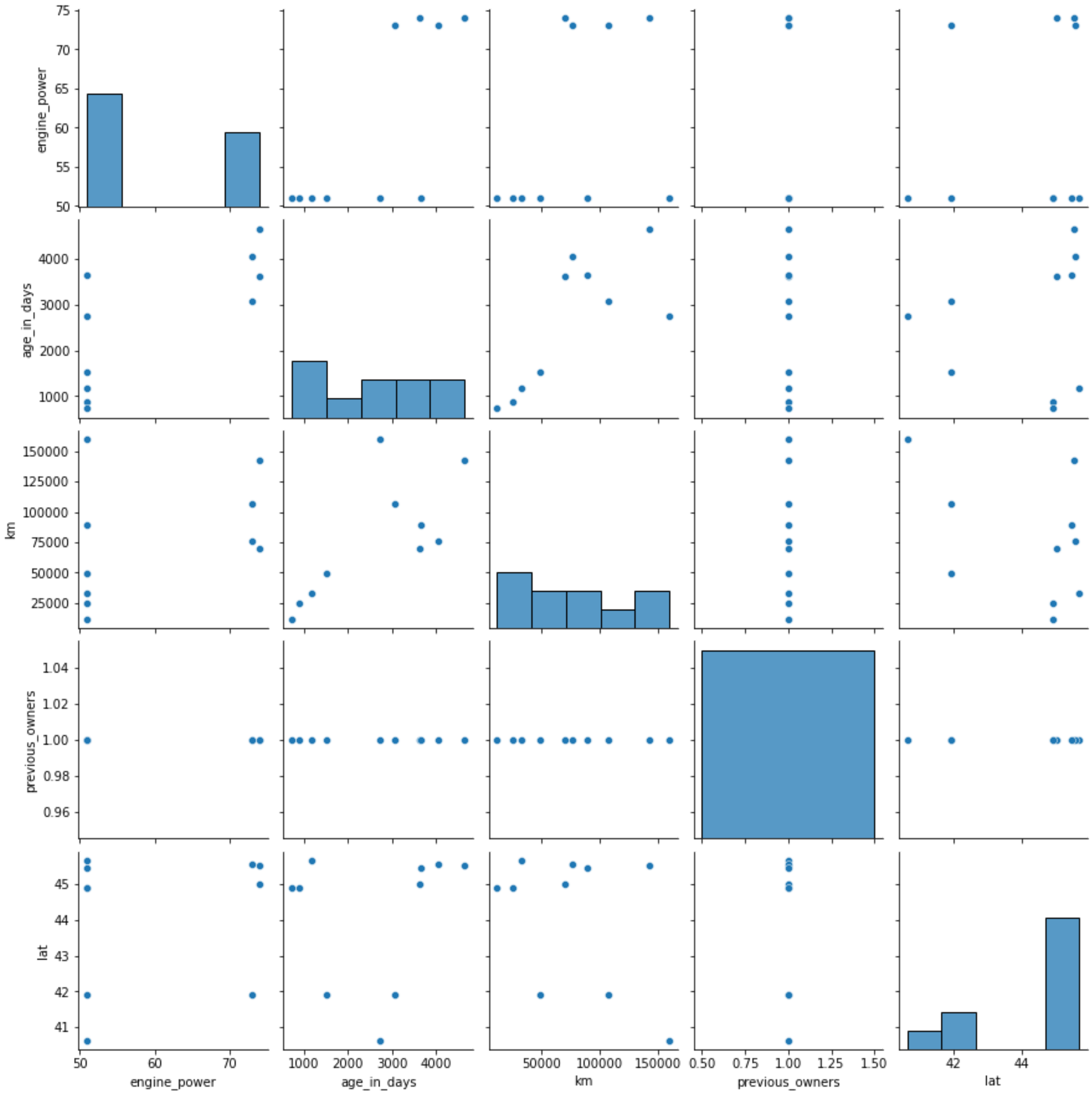
| | engine_power | age_in_days | km | previous_owners | lat | lon | price |
|---|---|---|---|---|---|---|---|
| **0** | 51.0 | 882.0 | 25000.0 | 1.0 | 44.907242 | 8.611559868 | 8900 |
| **1** | 51.0 | 1186.0 | 32500.0 | 1.0 | 45.666359 | 12.24188995 | 8800 |
| **2** | 74.0 | 4658.0 | 142228.0 | 1.0 | 45.503300 | 11.41784 | 4200 |
| **3** | 51.0 | 2739.0 | 160000.0 | 1.0 | 40.633171 | 17.63460922 | 6000 |
| **4** | 73.0 | 3074.0 | 106880.0 | 1.0 | 41.903221 | 12.49565029 | 5700 |
| **5** | 74.0 | 3623.0 | 70225.0 | 1.0 | 45.000702 | 7.68227005 | 7900 |
| **6** | 51.0 | 731.0 | 11600.0 | 1.0 | 44.907242 | 8.611559868 | 10750 |
| **7** | 51.0 | 1521.0 | 49076.0 | 1.0 | 41.903221 | 12.49565029 | 9190 |
| **8** | 73.0 | 4049.0 | 76000.0 | 1.0 | 45.548000 | 11.54946995 | 5600 |
| **9** | 51.0 | 3653.0 | 89000.0 | 1.0 | 45.438301 | 10.99170017 | 6000 |

In [118]:

```
sns.pairplot(d)
```

Out[118]:

<seaborn.axisgrid.PairGrid at 0x231e2b4b460>



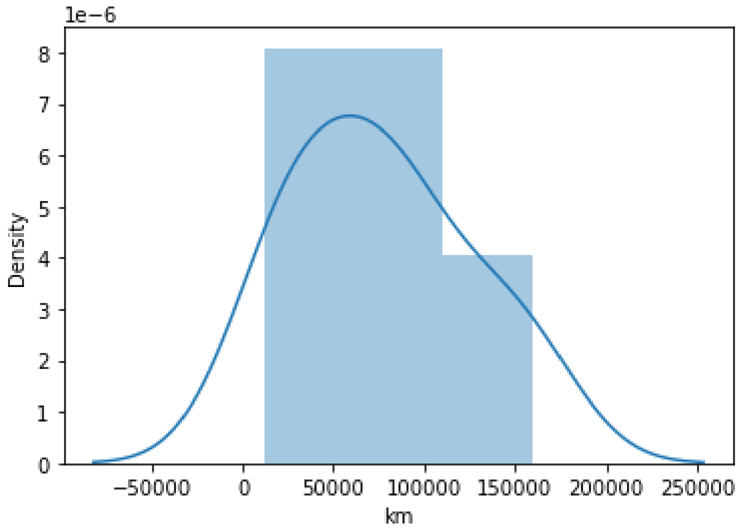
In [119]:

```
sns.distplot(d['km'])
```

```
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[119]:

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

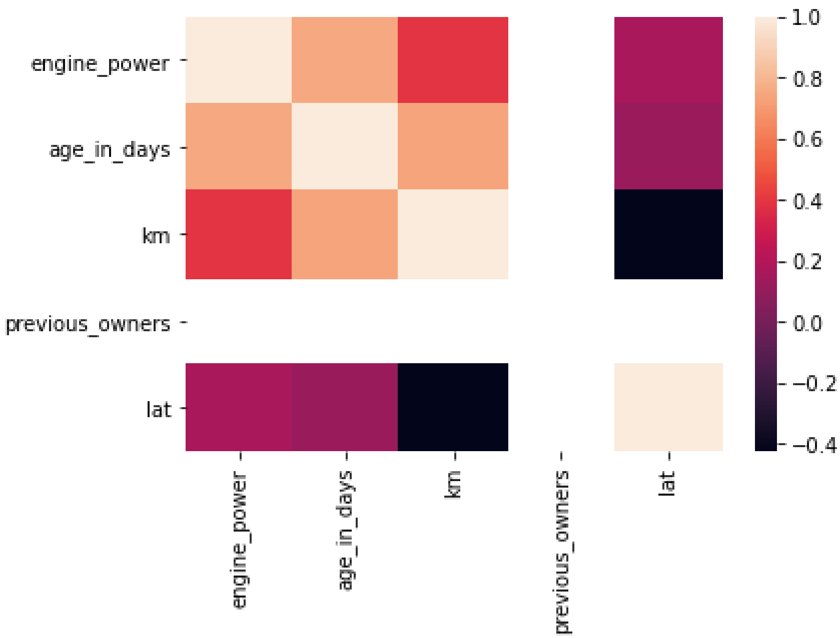



In [120]:

```
sns.heatmap(d.corr())
```

Out[120]:

```
<AxesSubplot:>
```



In [121]:

```
x=d[['engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
y=d['price']
```

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_)
```

```
-5.929905455559492e-10
```

In [125]:

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

Out[125]:

| | Co-efficient |
|---|---|
| **engine_power** | 1.782942e-13 |
| **age_in_days** | 9.071686e-16 |
| **km** | 5.556020e-15 |
| **previous_owners** | 1.110223e-16 |
| **lat** | 1.139652e-12 |
| **lon** | 1.484235e-12 |
| **price** | 1.000000e+00 |

In [126]:

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

```
1.0
```

In [127]:

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

In [128]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[128]:

```
Ridge(alpha=10)
```

In [129]:

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

Out[129]:

```
0.9999999999979275
```

In [130]:

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

Out[130]:

```
Lasso(alpha=10)
```

In [131]:

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

Out[131]:

```
0.9999999931211635
```

# 7. DataSet 2015

In [132]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\2015 - 2015.csv")
a
```

Out[132]:

| | 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 [133]:

```
b=a.head(10)
b
```

Out[133]:

| | Country | Region | Happiness Rank | Happiness Score | Standard Error | Economy (GDP per Capita) | Family | Health (Life Expectancy) | F |
|---|---|---|---|---|---|---|---|---|---|
| **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 | |
| **5** | Finland | Western Europe | 6 | 7.406 | 0.03140 | 1.29025 | 1.31826 | 0.88911 | |
| **6** | Netherlands | Western Europe | 7 | 7.378 | 0.02799 | 1.32944 | 1.28017 | 0.89284 | |
| **7** | Sweden | Western Europe | 8 | 7.364 | 0.03157 | 1.33171 | 1.28907 | 0.91087 | |
| **8** | New Zealand | Australia and New Zealand | 9 | 7.286 | 0.03371 | 1.25018 | 1.31967 | 0.90837 | |
| **9** | Australia | Australia and New Zealand | 10 | 7.284 | 0.04083 | 1.33358 | 1.30923 | 0.93156 | |

In [134]:

```
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 [135]:

```
a.describe()
```

Out[135]:

| | 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 [136]:

```
a.columns
```

Out[136]:

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

In [137]:

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

Out[137]:

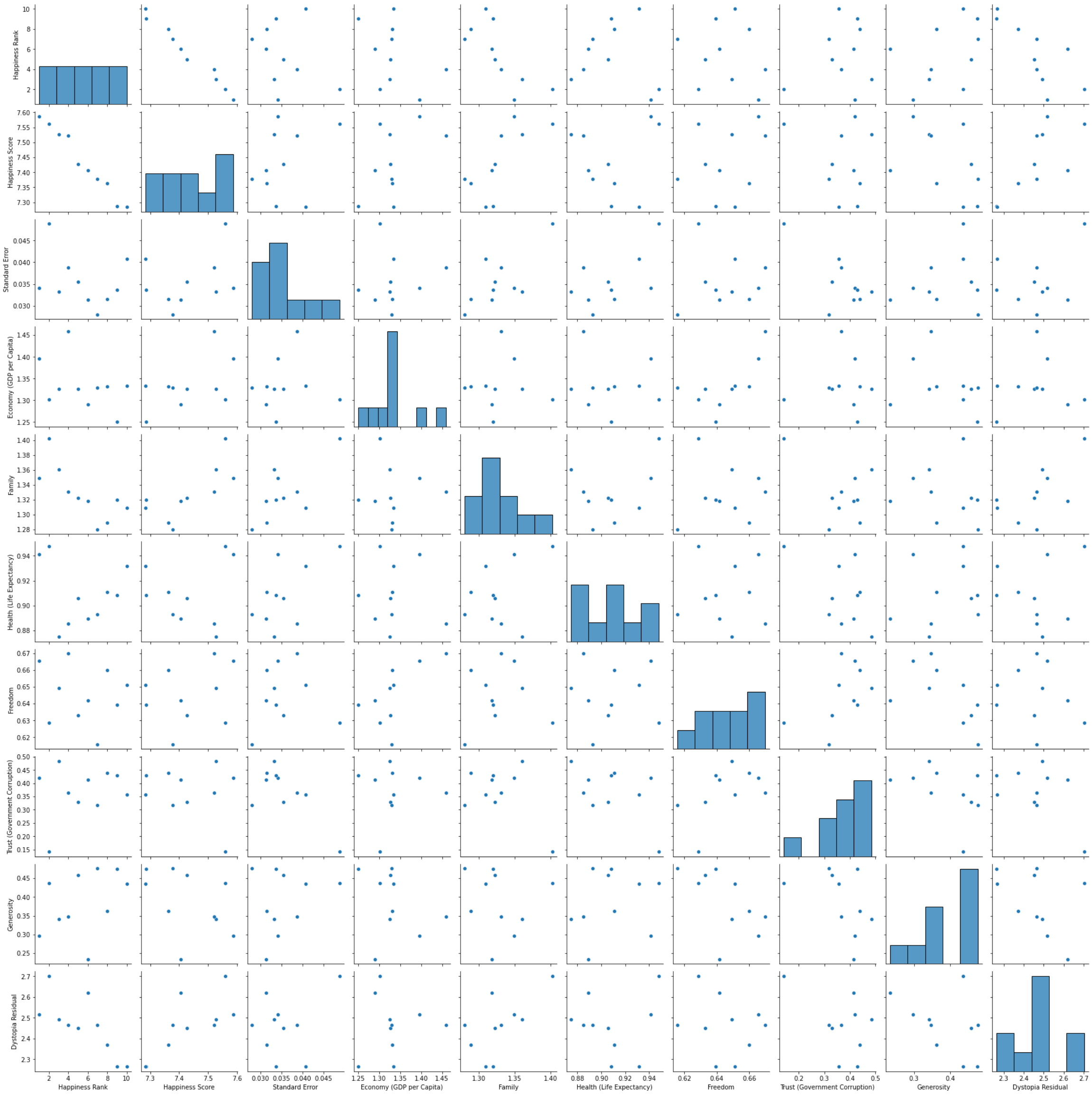
| | Happiness Rank | Happiness Score | Standard Error | Economy (GDP per Capita) | Family | Health (Life Expectancy) | Freedom | Trust (Government Corruption) |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 7.587 | 0.03411 | 1.39651 | 1.34951 | 0.94143 | 0.66557 | 0.41978 |
| **1** | 2 | 7.561 | 0.04884 | 1.30232 | 1.40223 | 0.94784 | 0.62877 | 0.14145 |
| **2** | 3 | 7.527 | 0.03328 | 1.32548 | 1.36058 | 0.87464 | 0.64938 | 0.48357 |
| **3** | 4 | 7.522 | 0.03880 | 1.45900 | 1.33095 | 0.88521 | 0.66973 | 0.36503 |
| **4** | 5 | 7.427 | 0.03553 | 1.32629 | 1.32261 | 0.90563 | 0.63297 | 0.32957 |
| **5** | 6 | 7.406 | 0.03140 | 1.29025 | 1.31826 | 0.88911 | 0.64169 | 0.41372 |
| **6** | 7 | 7.378 | 0.02799 | 1.32944 | 1.28017 | 0.89284 | 0.61576 | 0.31814 |
| **7** | 8 | 7.364 | 0.03157 | 1.33171 | 1.28907 | 0.91087 | 0.65980 | 0.43844 |
| **8** | 9 | 7.286 | 0.03371 | 1.25018 | 1.31967 | 0.90837 | 0.63938 | 0.42922 |
| **9** | 10 | 7.284 | 0.04083 | 1.33358 | 1.30923 | 0.93156 | 0.65124 | 0.35637 |

In [138]:

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

Out[138]:

<seaborn.axisgrid.PairGrid at 0x231e3b41fd0>



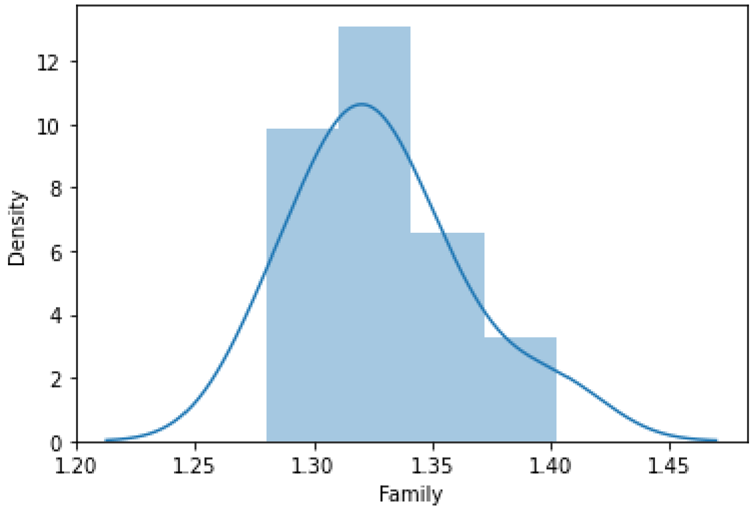
In [139]:

```
sns.distplot(c['Family'])
```

```
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[139]:

<AxesSubplot:xlabel='Family', ylabel='Density'>

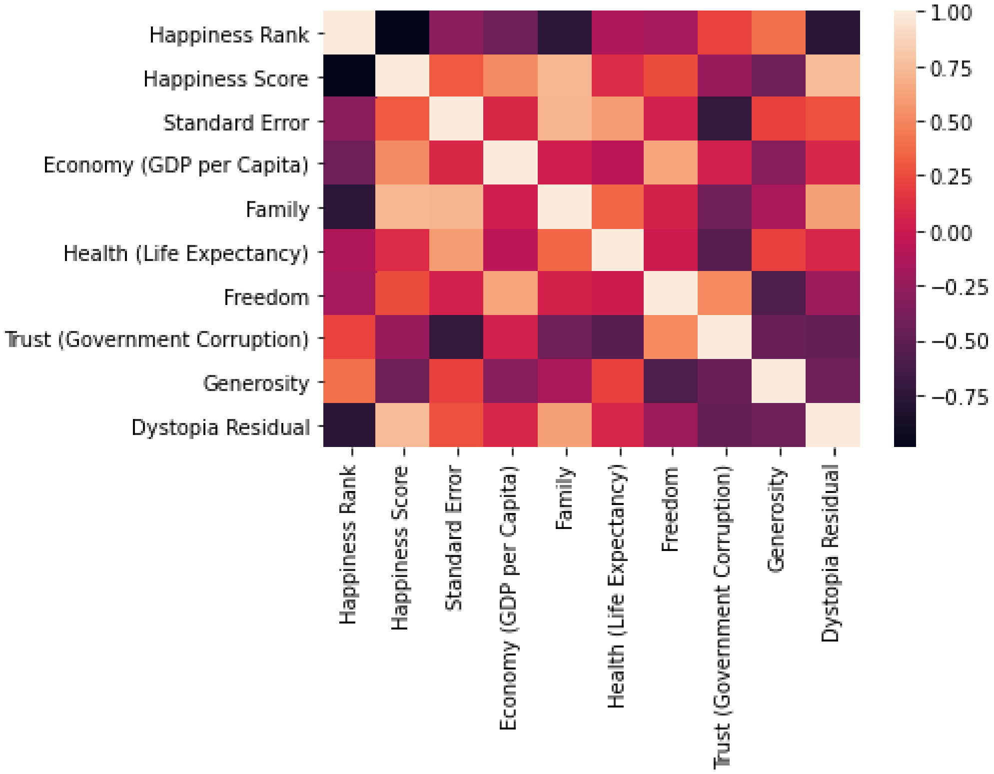


In [140]:

```
sns.heatmap(c.corr())
```

Out[140]:

```
&lt;AxesSubplot:&gt;
```




In [141]:

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

In [142]:

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

In [143]:

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

Out[143]:

```
LinearRegression()
```

In [144]:

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

```
1.8632387318551316
```

In [145]:

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

Out[145]:

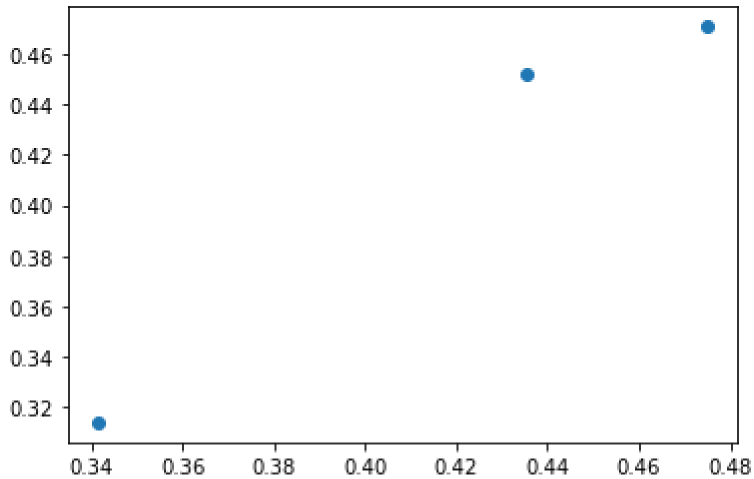
| | Co-efficient |
|---|---|
| **Happiness Rank** | -0.006093 |
| **Happiness Score** | -0.104144 |
| **Standard Error** | -0.043451 |
| **Economy (GDP per Capita)** | -0.028701 |
| **Family** | -0.186589 |
| **Health (Life Expectancy)** | 0.012070 |
| **Freedom** | -0.188674 |
| **Trust (Government Corruption)** | -0.267593 |
| **Generosity** | 0.743399 |
| **Dystopia Residual** | -0.187075 |

In [146]:

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

Out[146]:

```
<matplotlib.collections.PathCollection at 0x231e94791f0>
```




In [147]:

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

```
0.8890913969924003
```

In [148]:

```
la=Ridge(alpha=10)
la.fit(x_train,y_train)
```

Out[148]:

```
Ridge(alpha=10)
```

In [149]:

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

Out[149]:

```
-0.05469422106242772
```

In [150]:

```
rr=Lasso(alpha=10)
rr.fit(x_train,y_train)
```

Out[150]:

```
Lasso(alpha=10)
```

In [151]:

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

Out[151]:

```
-0.6279493751598471
```

# 8. DataSet Win_Equality

In [152]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\11_winequality-red.csv")
a
```

Out[152]:

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

1599 rows × 12 columns

In [153]:

```
b=a.head(10)
b
```

Out[153]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alco |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | |
| **1** | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.9968 | 3.20 | 0.68 | |
| **2** | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.9970 | 3.26 | 0.65 | |
| **3** | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.9980 | 3.16 | 0.58 | |
| **4** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | |
| **5** | 7.4 | 0.66 | 0.00 | 1.8 | 0.075 | 13.0 | 40.0 | 0.9978 | 3.51 | 0.56 | |
| **6** | 7.9 | 0.60 | 0.06 | 1.6 | 0.069 | 15.0 | 59.0 | 0.9964 | 3.30 | 0.46 | |
| **7** | 7.3 | 0.65 | 0.00 | 1.2 | 0.065 | 15.0 | 21.0 | 0.9946 | 3.39 | 0.47 | 1 |
| **8** | 7.8 | 0.58 | 0.02 | 2.0 | 0.073 | 9.0 | 18.0 | 0.9968 | 3.36 | 0.57 | |
| **9** | 7.5 | 0.50 | 0.36 | 6.1 | 0.071 | 17.0 | 102.0 | 0.9978 | 3.35 | 0.80 | 1 |

In [154]:

```
a.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 [155]:

```
a.describe()
```

Out[155]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total di |
|---|---|---|---|---|---|---|---|
| **count** | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.000000 | 1599.0 |
| **mean** | 8.319637 | 0.527821 | 0.270976 | 2.538806 | 0.087467 | 15.874922 | 46.4 |
| **std** | 1.741096 | 0.179060 | 0.194801 | 1.409928 | 0.047065 | 10.460157 | 32.8 |
| **min** | 4.600000 | 0.120000 | 0.000000 | 0.900000 | 0.012000 | 1.000000 | 6.0 |
| **25%** | 7.100000 | 0.390000 | 0.090000 | 1.900000 | 0.070000 | 7.000000 | 22.0 |
| **50%** | 7.900000 | 0.520000 | 0.260000 | 2.200000 | 0.079000 | 14.000000 | 38.0 |
| **75%** | 9.200000 | 0.640000 | 0.420000 | 2.600000 | 0.090000 | 21.000000 | 62.0 |
| **max** | 15.900000 | 1.580000 | 1.000000 | 15.500000 | 0.611000 | 72.000000 | 289.0 |

In [156]:

```
a.columns
```

Out[156]:

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

In [157]:

```
c=b[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide']]
c
```

Out[157]:

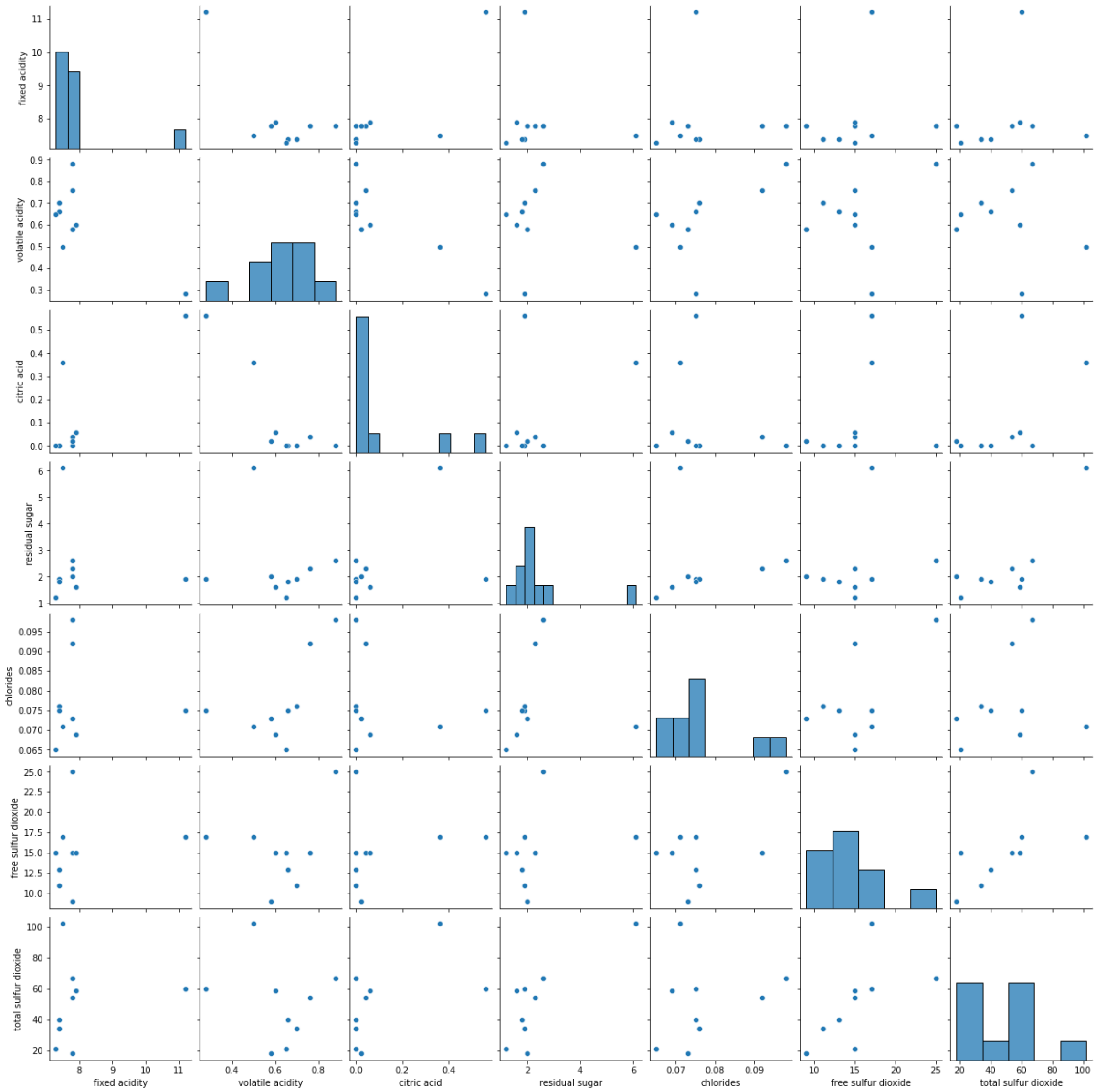
| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide |
|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 |
| **1** | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 |
| **2** | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 |
| **3** | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 |
| **4** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 |
| **5** | 7.4 | 0.66 | 0.00 | 1.8 | 0.075 | 13.0 | 40.0 |
| **6** | 7.9 | 0.60 | 0.06 | 1.6 | 0.069 | 15.0 | 59.0 |
| **7** | 7.3 | 0.65 | 0.00 | 1.2 | 0.065 | 15.0 | 21.0 |
| **8** | 7.8 | 0.58 | 0.02 | 2.0 | 0.073 | 9.0 | 18.0 |
| **9** | 7.5 | 0.50 | 0.36 | 6.1 | 0.071 | 17.0 | 102.0 |

In [158]:

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

Out[158]:

```
<seaborn.axisgrid.PairGrid at 0x231e94c17c0>
```



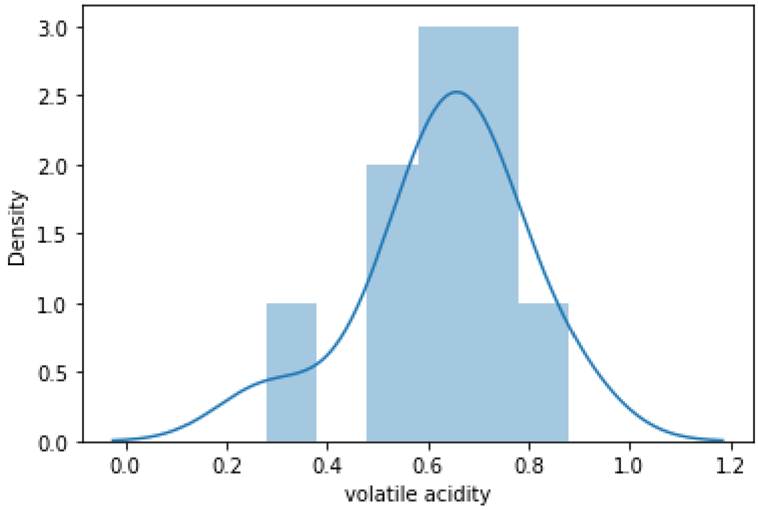
In [159]:

```
sns.distplot(c['volatile acidity'])
```

```
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[159]:

```
<AxesSubplot:xlabel='volatile acidity', ylabel='Density'>
```

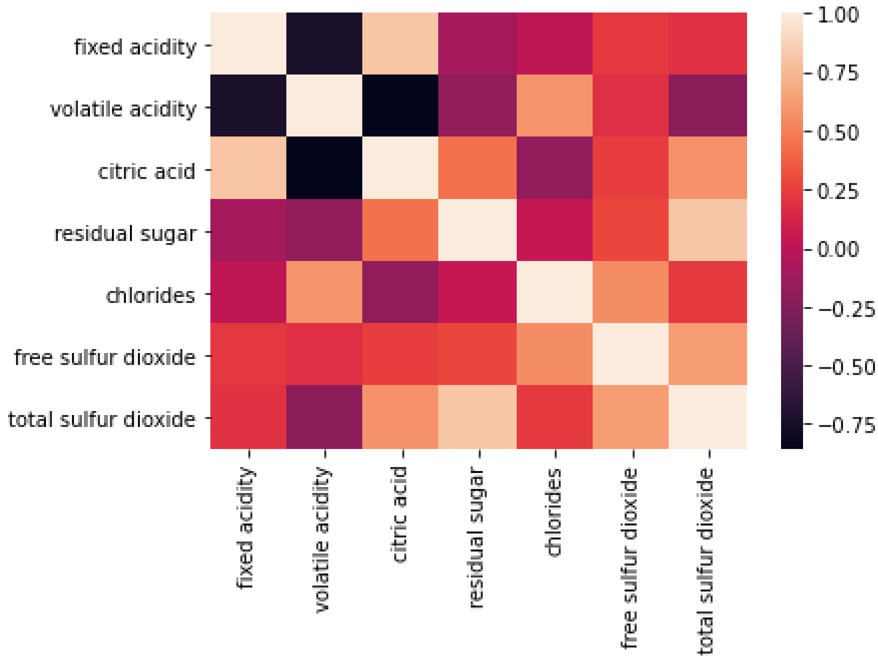



In [160]:

```
sns.heatmap(c.corr())
```

Out[160]:

```
<AxesSubplot:>
```



In [161]:

```
x=b[['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide']]
y=b['free sulfur dioxide']
```

In [162]:

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

In [163]:

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

Out[163]:

```
LinearRegression()
```

In [164]:

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

```
-0.00276335903562952
```

In [165]:

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

Out[165]:

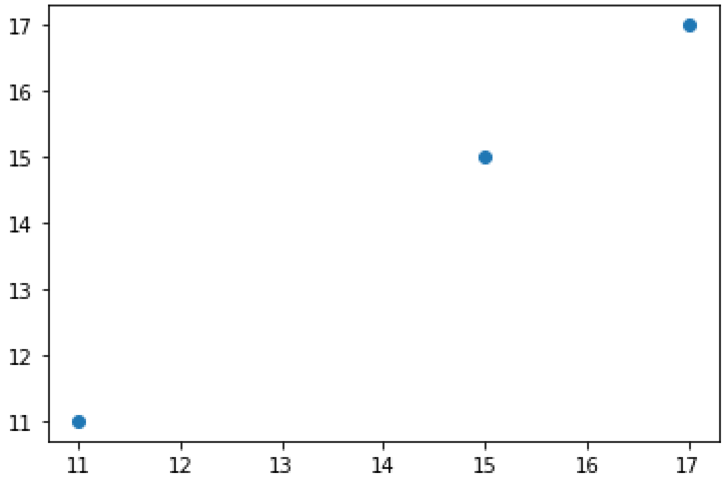
| | Co-efficient |
|---|---|
| **fixed acidity** | 0.000406 |
| **volatile acidity** | 0.000524 |
| **citric acid** | -0.002410 |
| **residual sugar** | -0.000206 |
| **chlorides** | -0.002677 |
| **free sulfur dioxide** | 0.999987 |
| **total sulfur dioxide** | 0.000004 |

In [166]:

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

Out[166]:

```
&lt;matplotlib.collections.PathCollection at 0x231eb2b06a0&gt;
```




In [167]:

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

```
0.9999998641877811
```

In [168]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[168]:

```
Ridge(alpha=10)
```

In [169]:

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

Out[169]:

```
0.8246480821325929
```

In [170]:

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

Out[170]:

```
Lasso(alpha=10)
```

In [171]:

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

Out[171]:

```
-3.1018628501862358
```

# 9. DataSet Mobile_prices

In [172]:

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

Out[172]:

| | Phone Name | Rating ?/5 | Number of Ratings | RAM | ROM/Storage | Back/Rare Camera | Front Camera | Battery | Processor |
|---|---|---|---|---|---|---|---|---|---|
| **0** | POCO C50 (Royal Blue, 32 GB) | 4.2 | 33,561 | 2 GB RAM | 32 GB ROM | 8MP Dual Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio A22 Processor, Upto 2.0 GHz Pro... |
| **1** | POCO M4 5G (Cool Blue, 64 GB) | 4.2 | 77,128 | 4 GB RAM | 64 GB ROM | 50MP + 2MP | 8MP Front Camera | 5000 mAh | Mediatek Dimensity 700 Processor |
| **2** | POCO C51 (Royal Blue, 64 GB) | 4.3 | 15,175 | 4 GB RAM | 64 GB ROM | 8MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Helio G36 Processor |
| **3** | POCO C55 (Cool Blue, 64 GB) | 4.2 | 22,621 | 4 GB RAM | 64 GB ROM | 50MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio G85 Processor |
| **4** | POCO C51 (Power Black, 64 GB) | 4.3 | 15,175 | 4 GB RAM | 64 GB ROM | 8MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Helio G36 Processor |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1831** | Infinix Note 7 (Forest Green, 64 GB) | 4.3 | 25,582 | 4 GB RAM | 64 GB ROM | 48MP + 2MP + 2MP + AI Lens Camera | 16MP Front Camera | 5000 mAh | MediaTek Helio G70 Processor |
| **1832** | Infinix Note 7 (Bolivia Blue, 64 GB) | 4.3 | 25,582 | 4 GB RAM | 64 GB ROM | 48MP + 2MP + 2MP + AI Lens Camera | 16MP Front Camera | 5000 mAh | MediaTek Helio G70 Processor |
| **1833** | Infinix Note 7 (Aether Black, 64 GB) | 4.3 | 25,582 | 4 GB RAM | 64 GB ROM | 48MP + 2MP + 2MP + AI Lens Camera | 16MP Front Camera | 5000 mAh | MediaTek Helio G70 Processor |
| **1834** | Infinix Zero 8i (Silver Diamond, 128 GB) | 4.2 | 7,117 | 8 GB RAM | 128 GB ROM | 48MP + 8MP + 2MP + AI Lens Camera | 16MP + 8MP Dual Front Camera | 4500 mAh | MediaTek Helio G90T Processor |
| **1835** | Infinix S5 (Quetzal Cyan, 64 GB) | 4.3 | 15,701 | 4 GB RAM | 64 GB ROM | 16MP + 5MP + 2MP + Low Light Sensor | 32MP Front Camera | 4000 mAh | Helio P22 (MTK6762) Processor |

1836 rows × 11 columns

In [173]:

```
b=a.dropna()
b
```

Out[173]:

| | Phone Name | Rating ?/5 | Number of Ratings | RAM | ROM/Storage | Back/Rare Camera | Front Camera | Battery | Processor |
|---|---|---|---|---|---|---|---|---|---|
| **0** | POCO C50 (Royal Blue, 32 GB) | 4.2 | 33,561 | 2 GB RAM | 32 GB ROM | 8MP Dual Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio A22 Processor, Upto 2.0 GHz Pro... |
| **1** | POCO M4 5G (Cool Blue, 64 GB) | 4.2 | 77,128 | 4 GB RAM | 64 GB ROM | 50MP + 2MP | 8MP Front Camera | 5000 mAh | Mediatek Dimensity 700 Processor |
| **2** | POCO C51 (Royal Blue, 64 GB) | 4.3 | 15,175 | 4 GB RAM | 64 GB ROM | 8MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Helio G36 Processor |
| **3** | POCO C55 (Cool Blue, 64 GB) | 4.2 | 22,621 | 4 GB RAM | 64 GB ROM | 50MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio G85 Processor |
| **4** | POCO C51 (Power Black, 64 GB) | 4.3 | 15,175 | 4 GB RAM | 64 GB ROM | 8MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Helio G36 Processor |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1831** | Infinix Note 7 (Forest Green, 64 GB) | 4.3 | 25,582 | 4 GB RAM | 64 GB ROM | 48MP + 2MP + 2MP + AI Lens Camera | 16MP Front Camera | 5000 mAh | MediaTek Helio G70 Processor |
| **1832** | Infinix Note 7 (Bolivia Blue, 64 GB) | 4.3 | 25,582 | 4 GB RAM | 64 GB ROM | 48MP + 2MP + 2MP + AI Lens Camera | 16MP Front Camera | 5000 mAh | MediaTek Helio G70 Processor |
| **1833** | Infinix Note 7 (Aether Black, 64 GB) | 4.3 | 25,582 | 4 GB RAM | 64 GB ROM | 48MP + 2MP + 2MP + AI Lens Camera | 16MP Front Camera | 5000 mAh | MediaTek Helio G70 Processor |
| **1834** | Infinix Zero 8i (Silver Diamond, 128 GB) | 4.2 | 7,117 | 8 GB RAM | 128 GB ROM | 48MP + 8MP + 2MP + AI Lens Camera | 16MP + 8MP Dual Front Camera | 4500 mAh | MediaTek Helio G90T Processor |
| **1835** | Infinix S5 (Quetzal Cyan, 64 GB) | 4.3 | 15,701 | 4 GB RAM | 64 GB ROM | 16MP + 5MP + 2MP + Low Light Sensor | 32MP Front Camera | 4000 mAh | Helio P22 (MTK6762) Processor |

1291 rows × 11 columns

In [174]:

```
c=b.head(10)
c
```

Out[174]:

| | Phone Name | Rating ?/5 | Number of Ratings | RAM | ROM/Storage | Back/Rare Camera | Front Camera | Battery | Processor | Pri |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | POCO C50 (Royal Blue, 32 GB) | 4.2 | 33,561 | 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 | 77,128 | 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 | 15,175 | 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 | 22,621 | 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 | 15,175 | 4 GB RAM | 64 GB ROM | 8MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Helio G36 Processor | ₹6 |
| 5 | POCO M4 5G (Power Black, 64 GB) | 4.2 | 77,128 | 4 GB RAM | 64 GB ROM | 50MP + 2MP | 8MP Front Camera | 5000 mAh | Mediatek Dimensity 700 Processor | ₹11 |
| 6 | POCO C55 (Power Black, 64 GB) | 4.2 | 22,621 | 4 GB RAM | 64 GB ROM | 50MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio G85 Processor | ₹7 |
| 7 | POCO C55 (Forest Green, 64 GB) | 4.2 | 22,621 | 4 GB RAM | 64 GB ROM | 50MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio G85 Processor | ₹7 |
| 8 | POCO C55 (Cool Blue, 128 GB) | 4.1 | 13,647 | 6 GB RAM | 128 GB ROM | 50MP Dual Rear Camera | 5MP Front Camera | 5000 mAh | Mediatek Helio G85 Processor | ₹9 |
| 9 | POCO M4 5G (Yellow, 128 GB) | 4.2 | 40,525 | 6 GB RAM | 128 GB ROM | 50MP + 2MP | 8MP Front Camera | 5000 mAh | Mediatek Dimensity 700 Processor | ₹13 |

In [175]:

```
a.info()
```

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

In [176]:

```
a.describe()
```

Out[176]:

| | Rating ?/5 |
|---|---|
| **count** | 1836.000000 |
| **mean** | 4.210512 |
| **std** | 0.543912 |
| **min** | 0.000000 |
| **25%** | 4.200000 |
| **50%** | 4.300000 |
| **75%** | 4.400000 |
| **max** | 4.800000 |

In [177]:

```
a.columns
```

Out[177]:

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

In [178]:

```
d=c[['Rating ?/5', 'Number of Ratings']]
d
```

Out[178]:

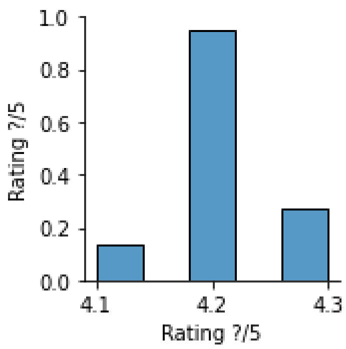
| | Rating ?/5 | Number of Ratings |
|---|---|---|
| **0** | 4.2 | 33,561 |
| **1** | 4.2 | 77,128 |
| **2** | 4.3 | 15,175 |
| **3** | 4.2 | 22,621 |
| **4** | 4.3 | 15,175 |
| **5** | 4.2 | 77,128 |
| **6** | 4.2 | 22,621 |
| **7** | 4.2 | 22,621 |
| **8** | 4.1 | 13,647 |
| **9** | 4.2 | 40,525 |

In [179]:

```
sns.pairplot(d)
```

Out[179]:

<seaborn.axisgrid.PairGrid at 0x231ec2d0a00>



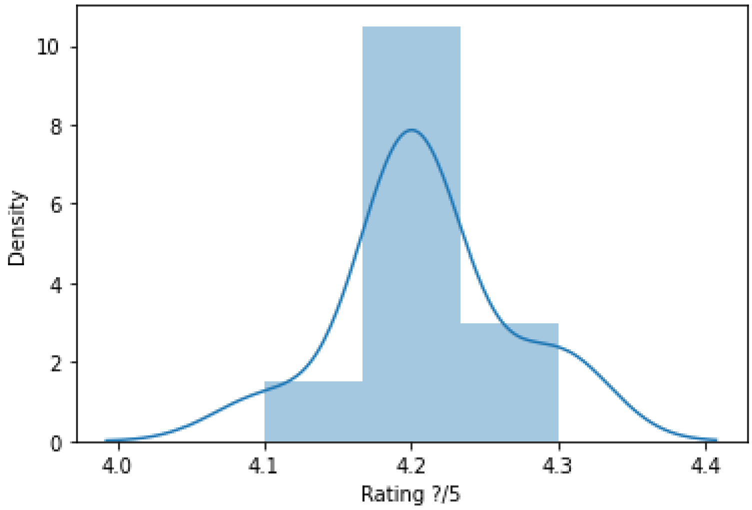
In [180]:

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

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

Out[180]:

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




In [181]:

```
sns.heatmap(d.corr())
```

Out[181]:

```
<AxesSubplot:>
```


In [182]:

```
x=c[['Rating ?/5']]
y=c['Rating ?/5']
```

In [183]:

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

In [184]:

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

Out[184]:

```
LinearRegression()
```

In [185]:

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

```
0.0
```

In [186]:

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

Out[186]:

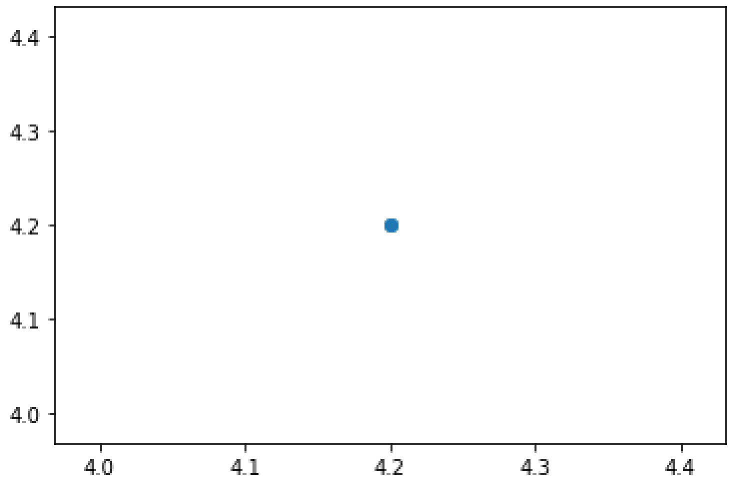
| | Co-efficient |
|---|---|
| **Rating ?/5** | 1.0 |

In [187]:

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

Out[187]:

```
<matplotlib.collections.PathCollection at 0x231ec4befa0>
```




In [188]:

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

```
1.0
```

In [189]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[189]:

```
Ridge(alpha=10)
```

In [190]:

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

Out[190]:

```
0.0
```

In [191]:

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

Out[191]:

```
Lasso(alpha=10)
```

In [192]:

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

Out[192]:

```
0.0
```

# 10. DataSet Placement

In [193]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\13_placement.csv")
a
```

Out[193]:

| | cgpa | placement_exam_marks | placed |
|---|---|---|---|
| **0** | 7.19 | 26.0 | 1 |
| **1** | 7.46 | 38.0 | 1 |
| **2** | 7.54 | 40.0 | 1 |
| **3** | 6.42 | 8.0 | 1 |
| **4** | 7.23 | 17.0 | 0 |
| **...** | ... | ... | ... |
| **995** | 8.87 | 44.0 | 1 |
| **996** | 9.12 | 65.0 | 1 |
| **997** | 4.89 | 34.0 | 0 |
| **998** | 8.62 | 46.0 | 1 |
| **999** | 4.90 | 10.0 | 1 |

1000 rows × 3 columns

In [194]:

```
b=a.head(10)
b
```

Out[194]:

| | cgpa | placement_exam_marks | placed |
|---|---|---|---|
| **0** | 7.19 | 26.0 | 1 |
| **1** | 7.46 | 38.0 | 1 |
| **2** | 7.54 | 40.0 | 1 |
| **3** | 6.42 | 8.0 | 1 |
| **4** | 7.23 | 17.0 | 0 |
| **5** | 7.30 | 23.0 | 1 |
| **6** | 6.69 | 11.0 | 0 |
| **7** | 7.12 | 39.0 | 1 |
| **8** | 6.45 | 38.0 | 0 |
| **9** | 7.75 | 94.0 | 1 |

In [195]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cgpa                  1000 non-null   float64
 1   placement_exam_marks  1000 non-null   float64
 2   placed                1000 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 23.6 KB
```

In [196]:

```
a.describe()
```

Out[196]:

| | cgpa | placement_exam_marks | placed |
|---|---|---|---|
| **count** | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 6.961240 | 32.225000 | 0.489000 |
| **std** | 0.615898 | 19.130822 | 0.500129 |
| **min** | 4.890000 | 0.000000 | 0.000000 |
| **25%** | 6.550000 | 17.000000 | 0.000000 |
| **50%** | 6.960000 | 28.000000 | 0.000000 |
| **75%** | 7.370000 | 44.000000 | 1.000000 |
| **max** | 9.120000 | 100.000000 | 1.000000 |

In [197]:

```
a.columns
```

Out[197]:

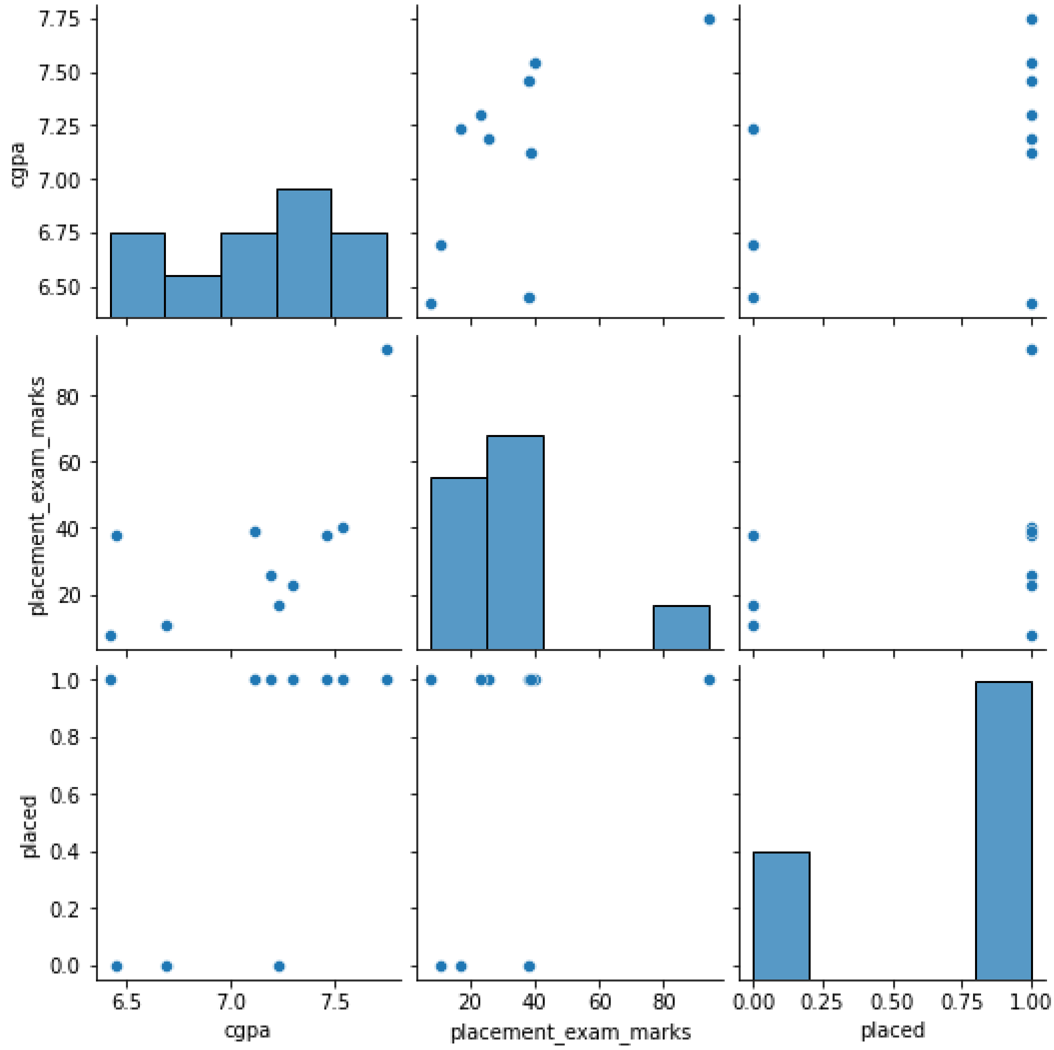
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [198]:

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

Out[198]:

```
<seaborn.axisgrid.PairGrid at 0x231ec4dc5e0>
```



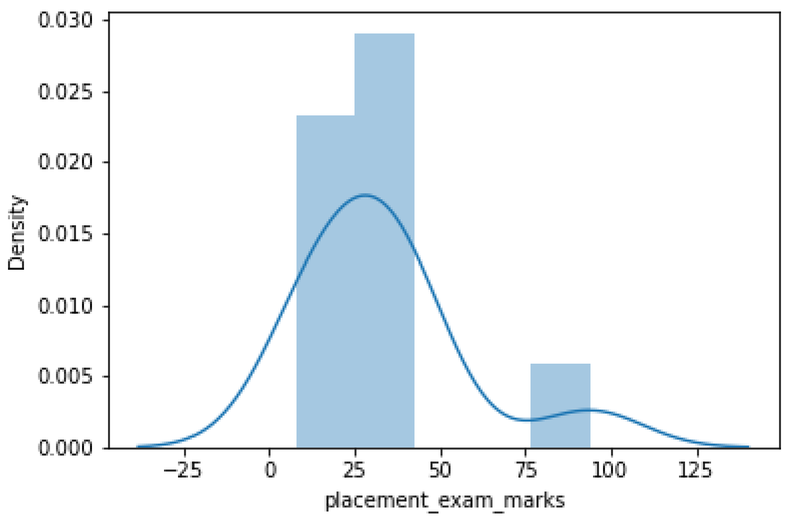
In [199]:

```
sns.distplot(b['placement_exam_marks'])
```

```
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[199]:

<AxesSubplot:xlabel='placement_exam_marks', ylabel='Density'>

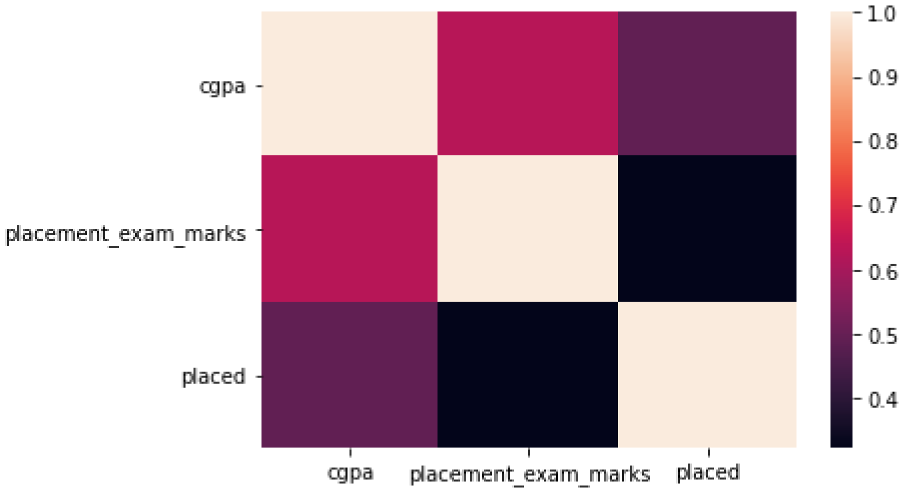



In [200]:

```
sns.heatmap(b.corr())
```

Out[200]:

<AxesSubplot:>




In [201]:

```
x=b[['cgpa', 'placement_exam_marks', 'placed']]
y=b['cgpa']
```

In [202]:

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

In [203]:

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

Out[203]:

```
LinearRegression()
```

In [204]:

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

```
4.440892098500626e-15
```

In [205]:

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

Out[205]:

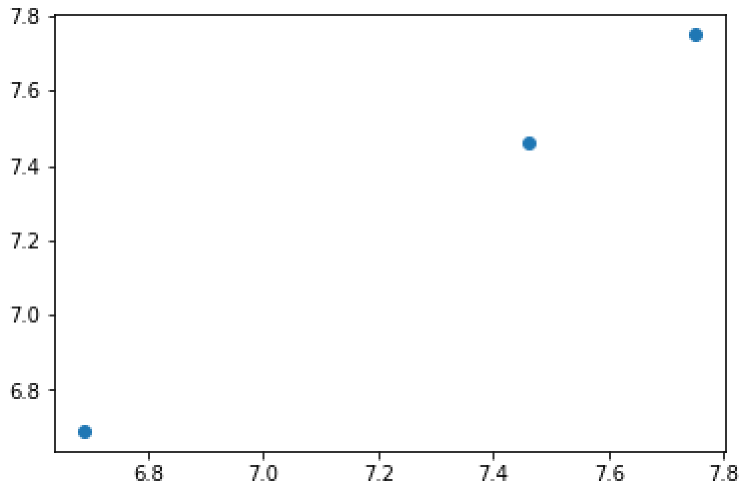
| | Co-efficient |
|---|---|
| **cgpa** | 1.000000e+00 |
| **placement_exam_marks** | 5.671633e-18 |
| **placed** | 2.896583e-16 |

In [206]:

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

Out[206]:

```
<matplotlib.collections.PathCollection at 0x231ec499190>
```



In [207]:

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

```
1.0
```

In [208]:

```
rr=Ridge(alpha=15)
rr.fit(x_train,y_train)
```

Out[208]:

```
Ridge(alpha=15)
```

In [209]:

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

Out[209]:

```
0.8265838494445259
```

In [210]:

```
la=Lasso(alpha=20)
la.fit(x_train,y_train)
```

Out[210]:

```
Lasso(alpha=20)
```

In [211]:

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

Out[211]:

```
-0.3491183211038509
```

In [ ]:

```

```