In [1]:

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

In [2]:

```
df=pd.read_csv(r'C:\Users\user\Downloads\fiat500_VehicleSelection_Dataset (2).csv')
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	l
0	1	lounge	51	882	25000	1	44.907242	8.6115
1	2	pop	51	1186	32500	1	45.666359	12.2418
2	3	sport	74	4658	142228	1	45.503300	11.4178
3	4	lounge	51	2739	160000	1	40.633171	17.6346
4	5	pop	73	3074	106880	1	41.903221	12.4956
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.7049
1534	1535	lounge	74	3835	112000	1	45.845692	8.6668
1535	1536	pop	51	2223	60457	1	45.481541	9.4134
1536	1537	lounge	51	2557	80750	1	45.000702	7.6822
1537	1538	pop	51	1766	54276	1	40.323410	17.5682

1538 rows × 9 columns

In [3]:

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

Out[3]:

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

In [4]:

```
df.info()
```

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

In [5]:

```
df.describe()
```

Out[5]:

	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 [6]:

```
df.columns
```

Out[6]:

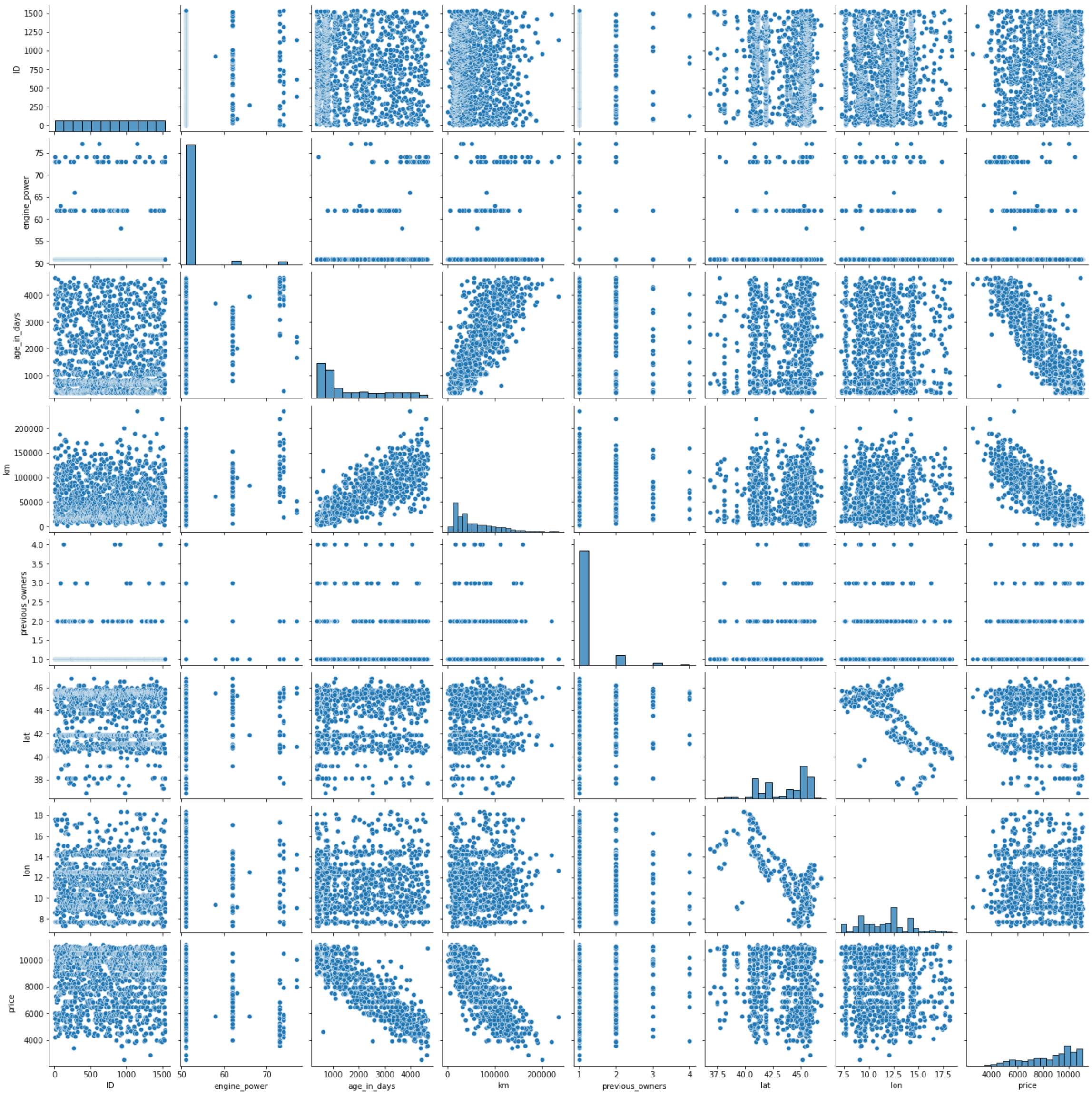
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owner
s',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x25f60022d60>
```



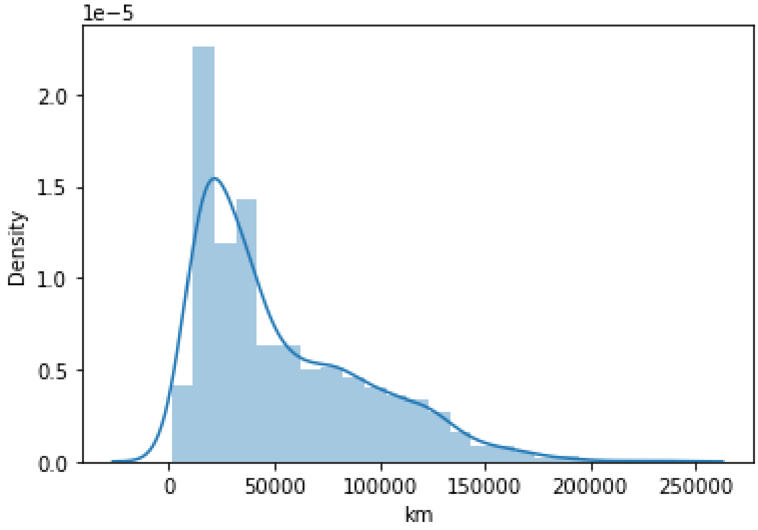
In [8]:

```
sns.distplot(df['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[8]:

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

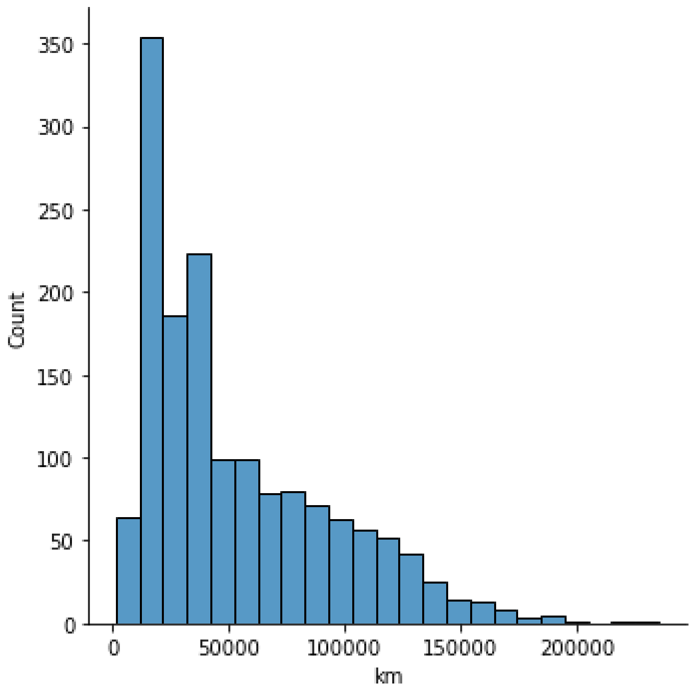


In [9]:

```
sns.displot(df["km"])
```

Out[9]:

<seaborn.axisgrid.FacetGrid at 0x25f5ef28ee0>



In [10]:

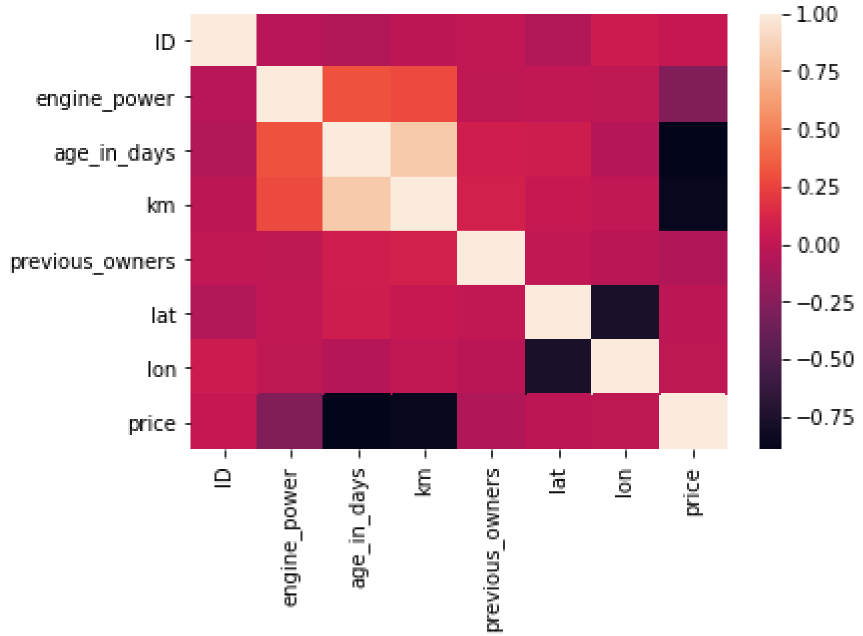
```
df1=df[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
```

In [11]:

```
sns.heatmap(df1.corr())
```

Out[11]:

<AxesSubplot:>



In [12]:

```
x=df1[['engine_power', 'age_in_days', 'km']]
y=df1[['price']]
```

In [13]:

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

In [14]:

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

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[15]:

```
LinearRegression()
```

In [16]:

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

[10351.85850095]

In [17]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[17]:

	0	1	2
0	12.564848	-0.876082	-0.018245

In [18]:

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

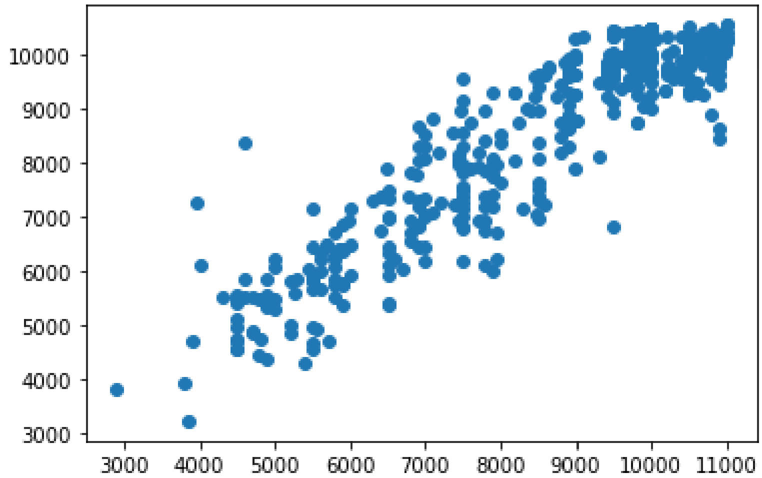
0.8450735960679158

In [19]:

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

Out[19]:

<matplotlib.collections.PathCollection at 0x25f647843d0>



In [20]:

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

Out[20]:

0.8450735960679158

In [21]:

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

Out[21]:

```
0.8394465521653983
```

In [22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

```
0.8450745616135951
```

In [25]:

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

Out[25]:

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

In [26]:

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

Out[26]:

```
0.8451538804515608
```

Elastic Net

In [37]:

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

Out[37]:

```
ElasticNet()
```

In [38]:

```
print(en.coef_)
```

```
[12.07110421 -0.87572413 -0.01824132]
```

In [39]:

```
print(en.intercept_)
```

```
[10376.67285731]
```

In [60]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[ 7248.5802076   9001.68330157  9901.36873507 10300.18196274
  7824.5433544  10295.96821859 10177.19900569 10020.36739509
 10329.97397894  5848.5554829   9626.35495483 10409.42483946
  7327.23683204  9353.56382082 10022.49635347  4684.54229102
  6036.57579158  5573.27727714  5307.71298458  6991.29831677
  9763.87757086  7902.85979122  8912.8578378   7379.32565062
 10042.04244816  9330.6310977  10036.19254303  7334.13862508
 10310.38280483  6743.51469833  5345.50039594  7397.36522808
 10426.18305197  9922.38363868  5680.12994228  9987.31849813
  8058.18148946  9806.61037059  9813.21372722  9164.04950619
  9907.79058534  9863.13056645 10099.95425975  6003.65112178
 10008.83056801 10343.44043951  9238.41621109  6946.91971073
  8890.81918406  8532.17242891 10133.24903155  7803.85011522
 10503.08844308  8783.41735072  4744.84172356  7965.86312956
  9416.48466038 10329.84852612  9120.43968974  9920.99729861
  5474.4943907  10006.06620281  3219.9456864  10339.56496545
  9956.81070171 10045.37096459  9932.59454809  9606.33895477
 10313.07420477  9607.10533493  6492.49726911  9923.68841669
  9879.69472972  5373.94392131  8977.56538259  8074.53434301
  9641.91479796 10319.89645721  7290.75419868  4458.1361403
  5847.44143132  6130.70954543 10068.31417211  7383.28467828
  7033.38966814 10329.84852612  9267.282289    6845.76472344
  9480.72089619  8758.8968971   9712.39396994  6325.71638808
 10146.05670926  6776.03145391  7795.61796873  7397.7931303
  9891.50545841  9342.69283596  9689.93031232  9451.83961623
  6238.2218744   9716.78708799  9562.89065589  9876.56582005
  7484.41254641  9633.42731004  9575.09685497 10238.562795
  7292.46245513  8498.72840545 10329.97397894  6421.66931272
  9936.07982824  9559.26894961 10398.16438955 10003.73568314
  5391.5199538  10348.79901776  7989.33540927 10031.83153875
  8316.32802638  8695.8119597   9659.86425357  8953.02749052
 10303.17353867 10177.02819158  9792.50983279  8783.41735072
 10256.41667517  9896.11191473  8755.40833283 10065.01686265
  7868.02799907  9909.82501601  9874.43686166  8305.90259003
 10286.73811235  9867.14033499 10058.92953854  6318.41738097
  9842.13044209  7951.08542666 10095.46162019 10379.92307287
 10371.12638937 10091.21666907 10141.5209448   9660.13787332
  9755.17317868  6710.85194529  5737.8688813   4956.13617031
  9650.19635573  6699.83419002 10342.77106832 10380.81689739
  6343.30877019  6936.34051374  7886.89094914  6103.97836351
 10310.38280483 10343.44043951  9733.30274679  9267.69020323
  6423.30662535 10193.35644341 10503.08844308  6471.19972616
  5949.60115666  9342.69283596 10404.71857976  7171.06981376
  6934.14970432  9629.89177873 10411.87354476  9383.21773741
  8991.77609714  9768.22095877  6242.76700152  5490.41329887
  9885.90100532  6386.08797893 10174.11380884 10253.72223596
  7903.85974251  9739.41068954  9985.7088934   7649.88917452
  7207.55244712  7067.11674094  9800.47824891  9960.17105005
  3818.3222655  10243.95785587 10339.0034307  10245.40723482
 10014.89500008 10564.70761082  8558.90000784  8965.00965826
  9219.6766804   6449.06231677  9810.59225328 10031.83153875
 10432.73168465  4707.6851053   5544.8501014  10238.87289738
  9504.02365438  9876.56582005 10332.19547351  9858.32450337
  7154.84342377  9734.11633884 10119.99951225 10256.534438
  5122.07970154  8036.15765198  9467.23923357  9624.31629623
  9663.76485607  5809.76545939  9780.42006777  9746.57952699
 10333.05676146  9868.36687207 10381.39625066 10355.83185106
  8379.47307441 10467.48139293  7283.55632373 10346.49666579
  9577.40822652  4873.83571832 10140.54555823  9815.88118739
  7528.8141476   6244.47525797  6719.6675665   6194.33962687
  9849.87440489  9803.60055333  7813.40518119  9986.72646106
```

```
  4384.03103659  7963.89519983  6088.46796867 10540.44665964
  9884.59622732  9939.10565041 10181.41033552  4728.28192915
  9803.98784891  9316.20644414 10206.2530543  10037.88342796
  9862.68550765  9847.00087364  8989.83609026  9014.04801594
 10415.25213441  9703.14562239  9967.46757672  8653.83103704
  6458.85919642  9660.13787332  9905.96645367  9045.9679058
  4715.28645869  9874.22233472 10196.69897322 10347.30322979
 10372.99137253  9652.92727757  8404.49690668  6064.24259332
  6933.73095772  9788.44334877  8763.240285    5576.07043091
  6229.52803354 10004.78403498  8183.87915557  8847.8414533
  9305.60212361  7661.25774274  7477.40043567 10041.54028809
  9982.06063006  9304.50645595 10075.9155255  10409.01360411
  7072.23832299  6501.61792745 10292.65224502  9758.88577349
  9662.03102419  6875.74600293  9652.57104803  9818.38024777
  9265.06544516  6844.056467    5541.24995676  8215.51288282
  7981.9198941   4570.23217449 10269.86489442  9664.05011188
  9795.99919994  9305.1084194   9757.64630776  8313.50394343
  9530.09780012 10287.10688475  8170.54028029  5495.26995781
  9483.9612453   5850.08289326  8396.0386342   9418.18106582
  9287.66776612  5780.68244021  9244.53373018 10503.08844308
  9634.68596089  9508.18770539  5543.94880183  8535.00929959
  9632.4071259   5821.07940759  8627.72240021 10337.59053438
  7765.23017154  9664.05011188  9766.91618076 10317.8982273
  9580.65651048 10466.82470553  8311.49208593 10503.63568258
  6501.61792745  8051.90312548  9586.43900787  9472.66791575
  8684.15257815 10442.21716933  4728.23381212  6233.23007702
 10144.4083485   7169.96677497  9791.94131271 10050.07285543
  5993.68514341  4564.36412788 10313.07420477  9694.28167207
 10335.461576   10407.18947244  6757.4126271   4299.65050911
  6199.08649321  5399.68147534  7027.54709713  9562.01783091
 10425.79561546  8754.9722558   9753.96156179 10398.16438955
  9808.00635518  6462.12215571  9345.74479729  4950.4880044
  8783.41735072  5868.315858    7553.69097417 10196.00338886
  9163.21889848  9876.94584843  3926.56697871  9737.58218901
 10055.95316466  7379.74191678  6832.78571061  7186.39143504
  5572.64022349  5517.27359944  9610.54833006  7016.66129601
 10318.80634707  8559.95758631  8038.84992172  6977.82773219
  9738.18282286 10344.88706106  6079.77374692  9840.18826573
 10283.7687238   8784.53782795  5924.40742115  9995.3489054
  9896.11191473  8108.89636991  9993.78979669  5854.48023924
 10008.61604108  9230.51838291 10351.49041769 10503.08844308
  7143.7485123   7225.7937638   8763.240285   10564.70761082
  6136.8043816   7264.35629223  9751.21149183 10328.51904247
  9732.65147598  7286.13282217  9878.70428203  8094.09406529
  9988.13498852 10503.08844308  4903.59272242 10278.25153123
  7197.49711277  8197.75946797  6925.92786511  9940.62495536
 10040.19570632  6562.60804877 10034.74592149  8761.69194256
  5868.20200424  8649.99504787  4999.54617215  8098.11480795
  7007.35930924  8473.20334982 10462.00899792  9449.71900977
  8386.34439379  4969.36011418 10243.68860499  4573.06852007
 10033.13631676  5693.39019084 10346.2678436   7336.54644188
  9277.04030869  4877.52273769  5732.64092762  9090.16333199
  5412.62183607 10305.80423434 10503.08844308  9775.45842963
  9699.61669633  9941.92973337  6844.73045191  9797.10797419
  7409.94076247  5868.50505602]
```

In [61]:

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

0.8451266309579982

Evaluation Metrics

In [62]:

```
from sklearn import metrics
```

In [63]:

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

Mean Absolute Error: 607.672143905509

In [64]:

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

Mean Squared Error: 592922.1522159289

In [65]:

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

Root Mean Squared Error: 770.0143844214398