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

In [2]:
```
d=pd.read_csv(r"C:\Users\user\Downloads\fiat500_VehicleSelection_Dataset (2).csv")
d
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [3]:
```
d.head()
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700

In [4]:
```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              1538 non-null   int64
 1   model           1538 non-null   object
```

```
 2   engine_power     1538 non-null   int64
 3   age_in_days      1538 non-null   int64
 4   km               1538 non-null   int64
 5   previous_owners  1538 non-null   int64
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [5]:
```
d.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lo
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.0000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.5634
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.3281
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.2454
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.5050
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.8692
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.7690
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.3655

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

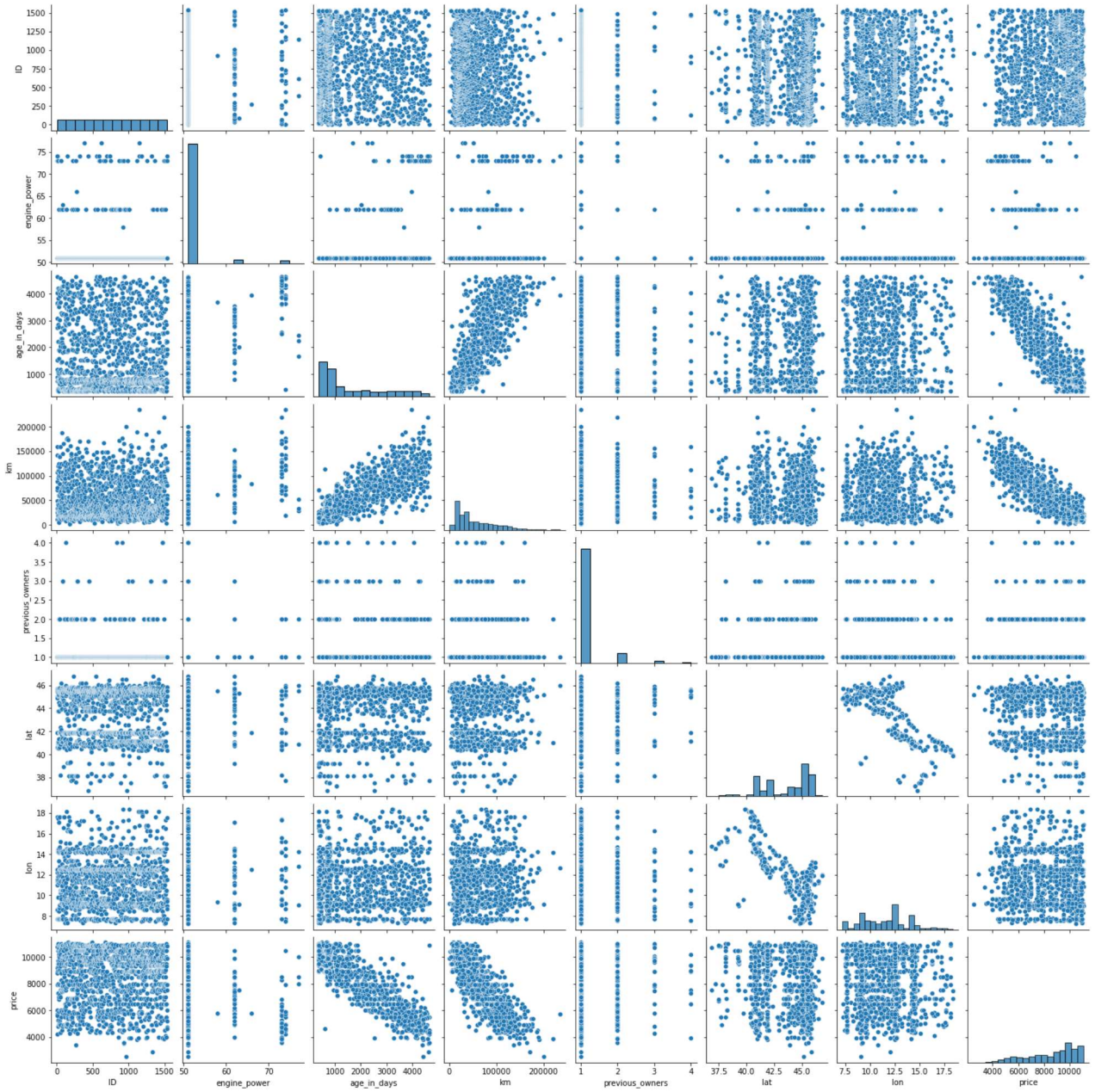
Out[6]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [7]:
```
d.index
```

Out[7]:
```
RangeIndex(start=0, stop=1538, step=1)
```

In [8]:
```
sns.pairplot(d)
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x22e35bede20>
```

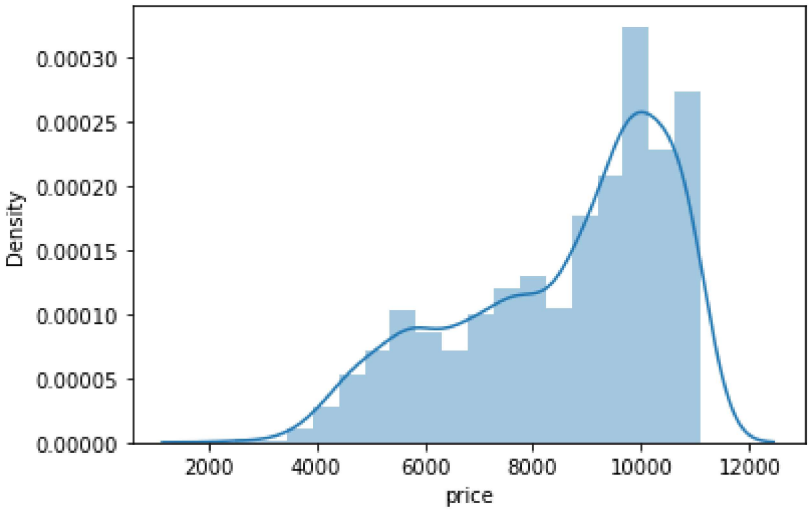


In [9]:

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

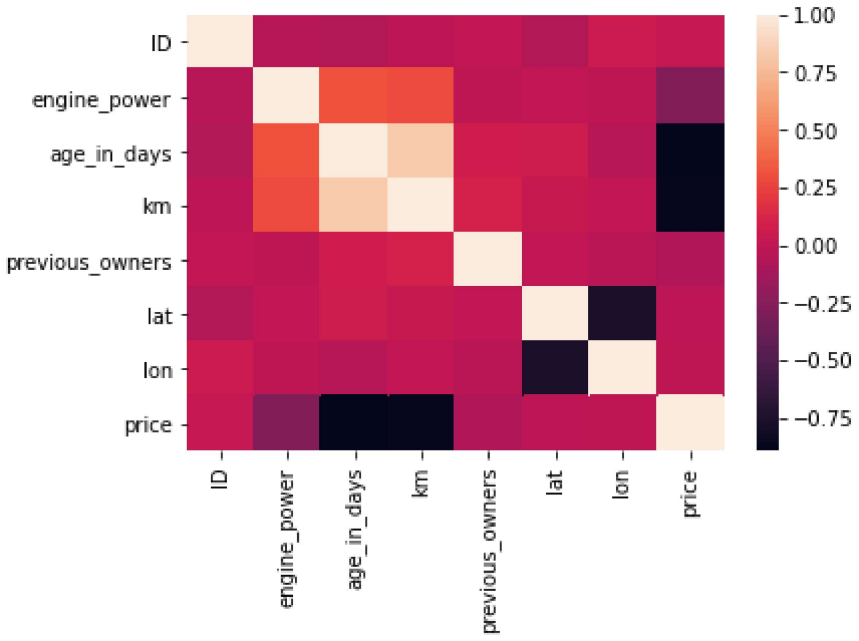
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function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

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



In [10]:
```
d1=d[['ID','engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
sns.heatmap(d1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
x=d1[['ID','engine_power', 'age_in_days', 'km', 'previous_owners','lat', 'lon']]
y =d1['price']
```

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

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

In [14]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

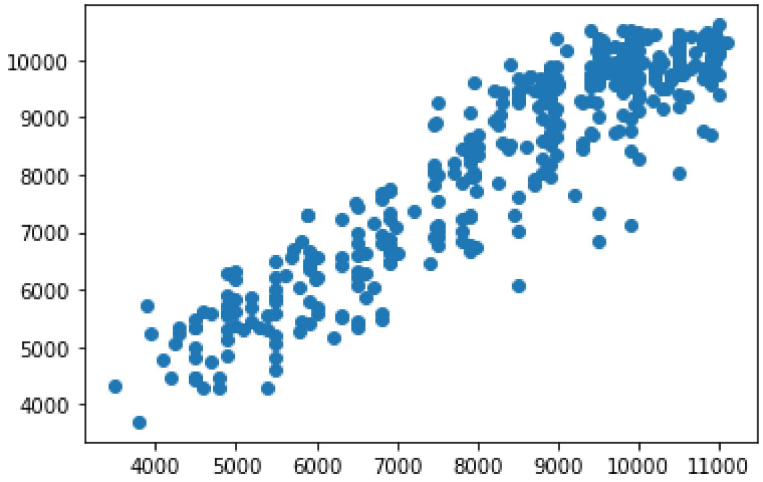
```
9362.50706129839
```

In [ ]:
```

```

In [16]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x22e39c0a880>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
0.8621731318587695
```

In [18]:
```
print(lr.score(x_train,y_train))
```

```
0.8339188064629376
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
rr.score(x_test,y_test)
```

Out[21]:
```
0.8621749319797637
```

In [22]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[22]:
```
Lasso(alpha=10)
```

In [23]:
```
la.score(x_test,y_test)
```

Out[23]:
```
0.8622470871614433
```

In [24]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[24]:
```
ElasticNet()
```

In [25]:
```
print(en.coef_)
```

```
[-6.55147411e-03  3.60921250e+00 -8.88507088e-01 -1.77651212e-02
 -1.38858129e+00  2.83604068e+01 -8.64552190e+00]
```

In [26]:
```
print(en.intercept_)
```

```
9666.547543059105
```

In [27]:
```
print(en.predict(x_test))
```

```
[10320.18912899 10444.54640207  9418.70507603 10351.1651352
  9401.60458344 10300.84298018  9513.64183018  4459.40492296
 10453.2772578   4424.38238062  5322.4173944   6199.21293604
  9650.87981257  9658.65954522  8699.31038765  5468.49601754
 10166.99987151 10019.76024545  7513.50924831 10334.23789082
 10274.82465485  9247.43790519 10452.31247383  9273.22461982
  9485.05362398  9798.47240823 10336.49220884  9537.74999785
 10343.19225973  5349.50548869  8558.08887348 10315.51874872
 10393.1802788   8484.32129568 10194.09181267  7241.77205457
 10276.23087916 10292.8591656   9580.45510328  4981.64020221
  6643.31715628 10431.66604643  6537.54770623  9422.90992289
  9675.03600434  7434.87656673  6847.94587253 10283.48364753
  8060.08460324 10288.12833703  5324.28526422 10474.06322995
 10436.91728629  9832.49722517  5701.80329578  9999.3247103
 10599.30432686 10317.59658256  9709.44919857  5626.85913942
  9926.61681174 10066.68361628  5036.76705977 10131.70644552
  8267.53810683  7565.75344527  9724.272064    8018.88475247
 10359.70480665  6929.24647904  9883.10141144  8772.61181542
 10316.50793859  9587.07883297  5873.92662387  7122.84983446
  4309.54529169  9879.77734102  6032.42512642  7300.00777083
  9151.37883679  7942.05536496  8609.377504   10375.62129409
  9566.65252599  9553.10357134  9365.69328519  9387.0042711
 10239.92134731  9291.08972571  9040.4982448   9717.64327172
  9327.33566348  9653.60718242  7127.10758091  9956.27660581
  8182.9960231   8529.95485087  5145.51170544 10203.56759762
  6491.7782055   7709.95404361  9650.65435483  6452.60223161
  9448.75458555  7998.47909698  7536.97819494  8360.18739893
  5800.83905517 10367.93293789 10026.28287726  7133.95199456
  9664.49417163  6787.75029341  5443.59462404  5569.19574409
  9184.58837192  9803.55564635  8524.18972283 10166.8883665
  5280.37157554  4764.17353719  9725.17682338  5717.80933061
  4295.08796975  9737.77687164 10465.53024777  5888.10459639
  8445.55671563  6842.21247002 10227.72931813  8014.58455768
```

10261.09554033 7335.5816821   9710.55828666 4282.36910642
 9733.54333393 6058.99197004 9066.70467914 9865.48949398
 6902.01593689 9962.34080112 9887.66105162 9869.07682399
10227.33001291 7688.31744885 7860.76322244 10089.12947556
10081.04494356 6509.08242162 5704.65874103 9666.2859947
 8204.45625483 8669.17819221 9572.61571368 10077.74322864
10329.86107938 4817.17179298 9024.10553984 10424.27823032
 9972.3829296 10300.8694592   7726.12850638 10099.65412302
 9927.74929935 7241.58530148 8518.76306172 8450.93081854
 9489.09960366 9651.92245712 6211.25358733 8013.70608675
10464.93778373 5878.46457175 10104.2940979   9635.36827546
 6969.41972426 9614.53443416 8333.32859727 8759.95999744
 9272.32909903 9460.40122992 9658.90622477 10110.00849974
 3704.43769331 10134.3471123   7817.66806742 6197.04644825
10168.1442607   9906.71161532 5322.13049055 5217.23277486
 9877.17211814 10188.0878761   9962.98860208 9652.04748849
 5487.0746646   5594.66124811 10392.18098679 9951.08802313
10386.70315874 5499.21772264 9812.58748913 10253.83107276
10516.52864305 6675.40554125 4859.04013014 9636.49829764
 9973.42383529 9900.2555188   9923.65760468 8725.39379805
 5584.15055149 9650.96395456 6176.64915909 9512.58208231
 5855.96518894 10397.9290011   8853.78499759 7606.23837082
10163.72100497 9789.45934339 5365.35209133 10143.37058419
 7832.41760962 10402.49404663 5464.51210021 9649.79274224
 9854.07583977 6615.40523437 9798.69892152 9291.6639742
10171.60906714 9165.47811326 6755.73570878 10432.24439306
 9669.85634719 7030.8970285 10317.85792943 10427.30079795
 7670.34560061 9249.48722423 8011.32003146 6327.49822904
 5059.46498812 10331.95744067 8658.32886915 7749.09215021
10181.81925337 10338.60690915 6265.59569822 10344.37412765
 6822.70356851 9807.51743712 9529.13116964 9069.92305572
 9712.52224573 9840.15264849 10412.35349168 10423.91132335
 9962.56086283 9617.7645376   6425.33038896 10055.38220852
10223.29360163 10052.58820163 6359.90489703 5550.47764937
10040.03379035 8577.04868619 8734.82355325 7009.83454171
 9230.58603971 9615.66921504 9388.45617903 7222.42023134
 9889.58481612 7914.89027788 10360.79203647 10267.23259279
 8946.89980216 10375.18238707 5603.23252085 10170.2294112
 6215.97082483 10244.60931752 5415.53828667 9389.43937319
10171.62328957 6461.16830164 7353.49379985 5449.54144261
10149.51476733 6802.70824593 6924.22804935 10131.54620871
 8967.26661164 9636.60548707 10232.32571841 9572.26872282
 5419.2341586   9411.46250857 4807.17823119 8295.88944413
 4497.64066934 6037.13604926 4284.68399158 9873.63042559
 5998.36135318 8749.91294532 8950.12236237 8474.08764763
 9877.76569624 9903.06472869 8200.59361128 10267.20631288
 8160.1710135   7989.78580979 8273.33549926 4769.23181857
 9510.98488808 9863.0258859   7309.77031707 10291.38616568
 9691.69849394 8470.33619134 9820.5671846   9755.70337177
 9739.44762951 10344.28024442 10088.58025853 10407.08698504
 9318.4173499   6434.46576567 8051.05430886 8868.74190149
 6297.25510042 6854.34468343 10180.62972082 5618.69523857
 4632.07146779 10502.81877222 9708.78959142 10229.81631239
 6546.68125314 9799.27814602 10206.18443393 8681.10324026
 5170.14160204 9754.72868028 9464.21868217 9629.10751321
 9760.3025763   6290.82584231 10389.20503657 9276.11744141
 9132.89053244 10503.20764238 8405.81824552 9033.56403915
 9672.63629037 10356.56557519 10401.73845817 10371.82486484
10288.81810111 6656.45352876 10028.51797841 10171.57255482
 9825.47959834 7084.18210867 9286.39154889 10103.29706991
 7309.84893475 5795.59605656 9317.25565845 8438.80084132
 7264.88974371 9704.43434015 6781.21802358 5341.72557131
10299.2349455   9684.42166796 9886.5995138   4458.63626339
 9790.31887548 10260.10679038 10087.65473615 9803.72055744
 6860.7338228   7012.48855116 7271.30549927 9668.26516982

```
 10416.03113102  9864.59662653  9843.88566715 10440.14986322
 10426.11832128  9734.10864491  7966.93417471  5226.5215127
  6495.12094499 10338.86253289 10440.85012031  8868.30030747
  6546.35942149  6040.35521402 10085.01106519  5853.41406122
 10256.49675158  6629.14365201  6674.1057842  10340.30349815
  4474.33916501  6299.21121138  5341.28543602  9809.36911936
  9668.97719812  9661.59681006  9777.52452911  9779.94544185
  9675.57392467  6720.74893002  7091.18262266  5214.71716536
  5336.45003544  6798.50169807  9729.35146594  9563.87448718
  5576.36991327  6636.32034393 10352.7619506   8479.98713374
  8749.12165523  5630.65546142  9598.09264838  9654.43991579
  9962.61280748  9597.02909995  9717.89652863  5573.07514807
  5814.03218124  9609.4197955  10356.59615644  9207.4139682
  9872.09578772  8845.10538118  9812.48459715  9700.08559447
  7835.68381647  9479.13202094  8913.32866427  9202.31928664
  5678.87379379  8698.89904043  8042.31688949  9746.13005186
  5325.87222343  7352.41841167 10364.74209874 10263.88738366
  9540.36872665 10416.23032652]
```

In [28]:
```
print(en.score(x_test,y_test))
```

```
0.8621575746517873
```

In [29]:
```
from sklearn import metrics
```

In [30]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 585.7380846804482
```

In [31]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 527546.1541088322
```

In [32]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 726.3237254205815
```

In [ ]: