LinearRegression

In [2]:

```
import numpy as np
import pandas as pd
```

data collection

In [3]:

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

In [4]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\fiat500_VehicleSelection_Dataset.csv")
df
```

Out[4]:

	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

first 10 rows

In [5]:

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

Out[5]:

	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	

data cleaning

In [6]:

```
df.info()
```

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

In [7]:

```
df.describe()
```

Out[7]:

	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 [8]:

```
df.columns
```

Out[8]:

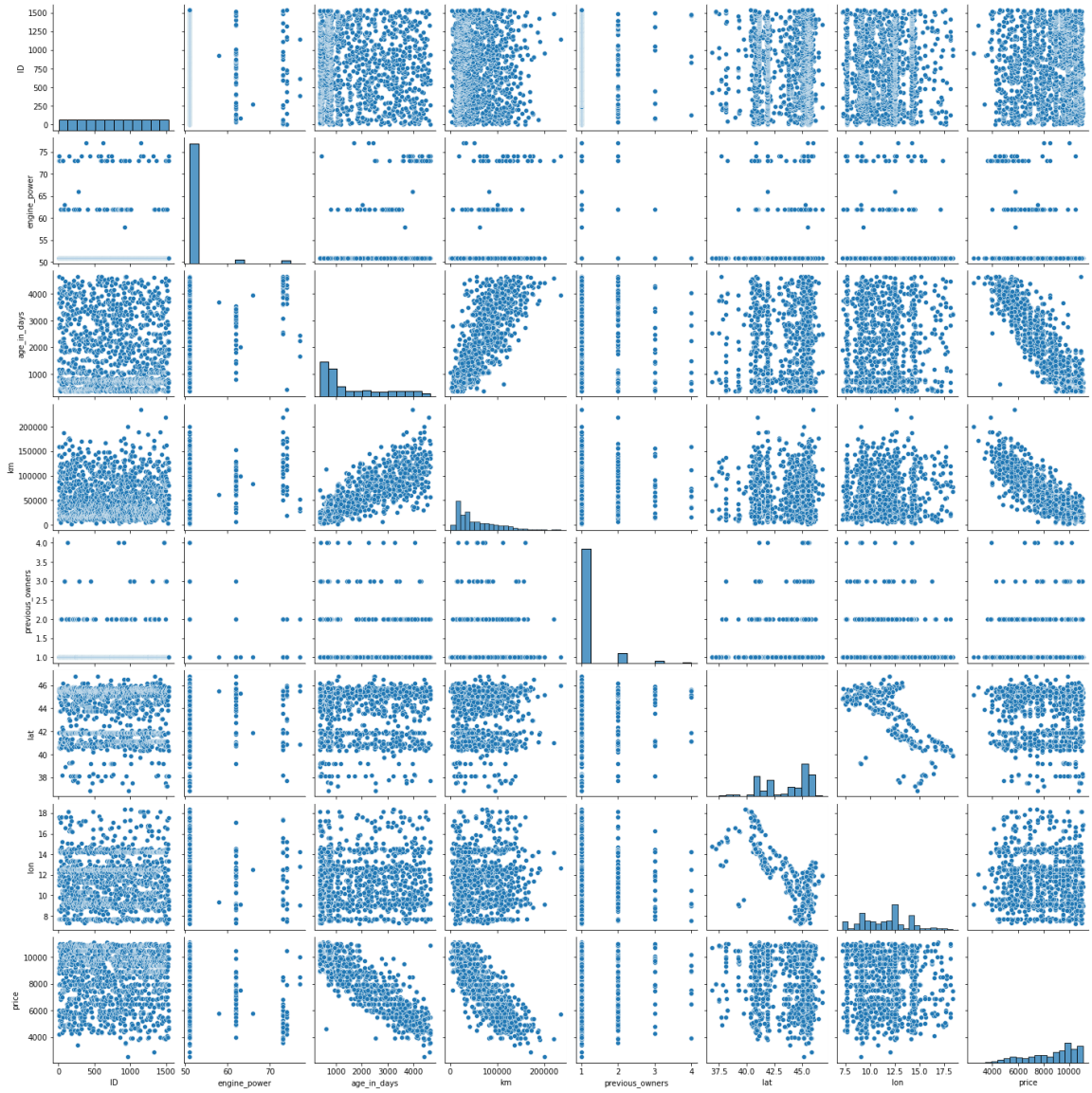
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owner
s',
       'lat', 'lon', 'price'],
      dtype='object')
```

In [9]:

```
sb.pairplot(df)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x27652350760>
```



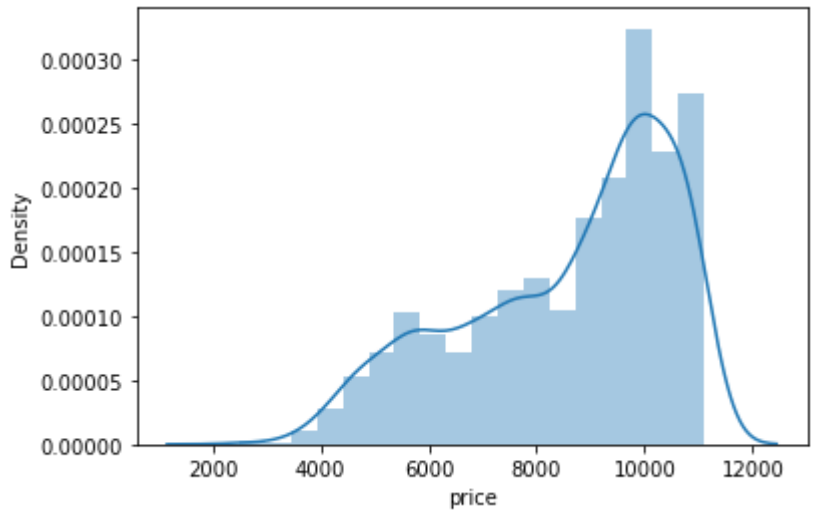
In [11]:

```
sb.distplot(df["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 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[11]:

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



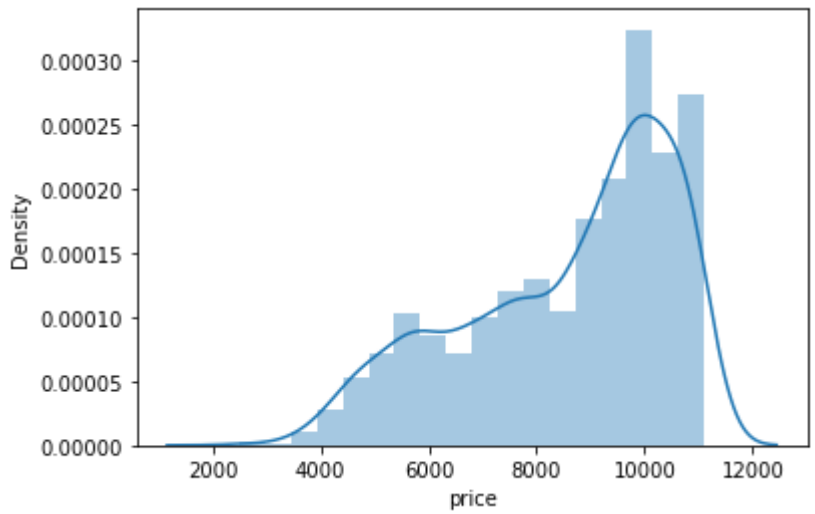
In [12]:

```
sb.distplot(df["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 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='price', ylabel='Density'>
```



In [13]:

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

Out[13]:

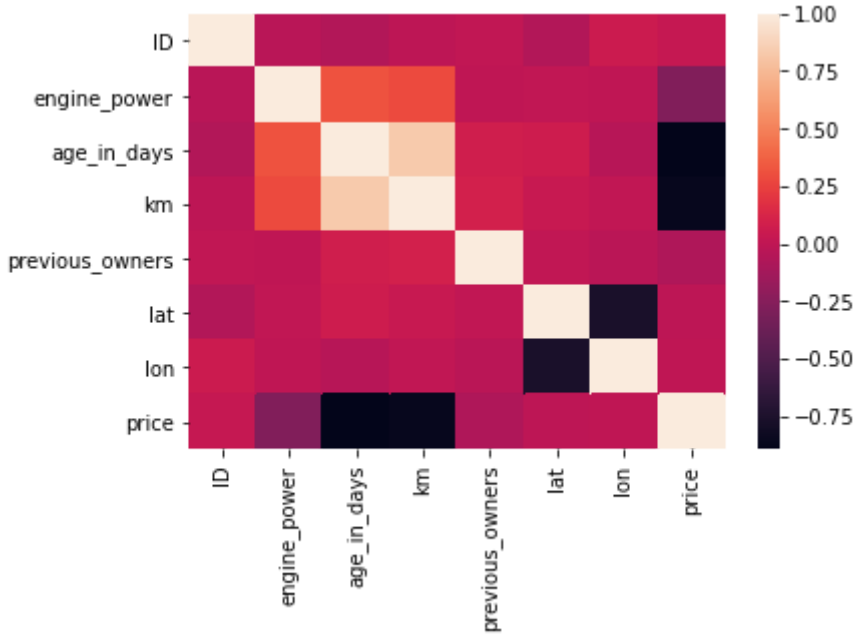
	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	51	882	25000	1	44.907242	8.611560	8900
1	2	51	1186	32500	1	45.666359	12.241890	8800
2	3	74	4658	142228	1	45.503300	11.417840	4200
3	4	51	2739	160000	1	40.633171	17.634609	6000
4	5	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...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

In [14]:

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

Out[14]:

```
<AxesSubplot:>
```



model building

In [16]:

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

In [17]:

```
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)
```

linear regression

In [18]:

```
from sklearn.linear_model import LinearRegression

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

Out[18]:

```
LinearRegression()
```

In [19]:

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

```
-1.6370904631912708e-11
```

In [20]:

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

Out[20]:

	Co_efficient
ID	-7.772387e-17
engine_power	4.243440e-13
age_in_days	-3.166077e-16
km	7.850781e-18
previous_owners	-4.675231e-14
lat	1.039060e-14
lon	-2.983376e-15
price	1.000000e+00

In [21]:

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

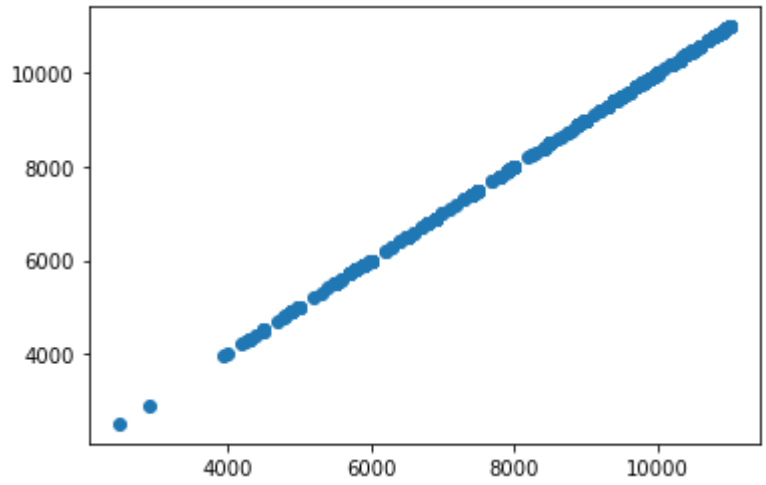
```
1.0
```

In [22]:

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

Out[22]:

```
<matplotlib.collections.PathCollection at 0x2765ba80a00>
```



lasso and ridge regression

In [23]:

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

Out[23]:

```
1.0
```

In [24]:

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

Out[24]:

```
1.0
```

In [25]:

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

In [26]:

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

Out[26]:

```
1.0
```

In [27]:

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

Out[27]:

```
0.9999999776102736
```

elasticnet

In [28]:

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

Out[28]:

```
ElasticNet()
```

In [29]:

```
print(e.coef_)
```

```
[ 1.60479771e-05 -0.00000000e+00  2.14464794e-04  1.22254470e-05
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00  1.00034957e+00]
```

In [30]:

```
print(e.intercept_)
```

```
-4.0254865927417995
```

In [31]:

```
predictions = e.predict(x_test)
predictions
```

Out[31]:

```
array([10499.94417351, 10200.21580463,  9499.64888715, 10499.86429224,
        8899.6578462 ,  5699.8241588 , 10500.0234004 ,  9699.97540706,
        4900.15210027,  8990.51349599,  9799.69034599,  8899.73347194,
        9800.21753652, 10850.05753138, 10950.10253098,  4790.61816137,
        8989.75997619,  6899.88361486,  9979.74247901, 10200.19092741,
        4799.94221991,  6499.8100457 ,  5500.37389344,  7449.96466646,
        8999.52630264,  9300.80126028, 10300.25919097,  8501.2371845 ,
       10500.43733   ,  9399.84616916, 10800.27940147,  9989.74546607,
        9969.7568727 , 10499.89051748,  9969.85301512,  9899.83028806,
        9899.92173843,  5500.34818696,  5999.77496738, 10500.04143638,
        7800.25578324,  9399.81782307,  8500.54161266,  9489.58456738,
        8499.76875645,  5500.17939637,  8899.5625524 ,  9489.89189565,
        4899.58507317,  7099.49156249,  9399.89712896,  9200.22315085,
       10900.01803121, 10199.84798454,  4489.96350253, 10850.11974807,
        6499.58566675,  9979.76235922, 10700.24805407,  4998.95418529,
        8900.19144985, 10489.95213626,  5800.47075816,  5899.29040427,
        5998.76917938,  7399.8722299 , 10490.11457947,  8899.69464269,
        9969.7235819 , 10800.00643274, 10850.03687392,  9800.04137385,
```

In [32]:

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

```
0.9999999746311603
```

In [33]:

```
from sklearn import metrics
```

mean absolute error

In [34]:

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

```
Mean Absolute Error: 0.24388453901828994
```

mean squared error

In [35]:

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

```
Mean Squared Error: 0.09651122049205854
```

root mean squared error

In [36]:

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

```
Root Mean Squared Error 0.3106625508362064
```

In [ ]:

```

```