In [25]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

read a file

In [26]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [27]:
```
data.describe()
```

Out[27]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [28]:
```
data.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
```

acess prev own==1

In [29]:
```
k=data.loc[data.previous_owners==1]
k
```

Out[29]:

	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

1389 rows × 9 columns

In [30]:
```
s=k.drop(['ID','lat','lon'],axis=1)
s
```

Out[30]:

	model	engine_power	age_in_days	km	previous_owners	price
0	lounge	51	882	25000	1	8900
1	pop	51	1186	32500	1	8800
2	sport	74	4658	142228	1	4200
3	lounge	51	2739	160000	1	6000
4	pop	73	3074	106880	1	5700
...	...	...	...	...	...	...
1533	sport	51	3712	115280	1	5200
1534	lounge	74	3835	112000	1	4600
1535	pop	51	2223	60457	1	7500
1536	lounge	51	2557	80750	1	5990
1537	pop	51	1766	54276	1	7900

1389 rows × 6 columns

In [31]:
```
s['model']=s['model'].map({'lounge':1,'pop':2,'sport':3})
s
```

Out[31]:

	model	engine_power	age_in_days	km	previous_owners	price
0	1	51	882	25000	1	8900
1	2	51	1186	32500	1	8800
2	3	74	4658	142228	1	4200
3	1	51	2739	160000	1	6000
4	2	73	3074	106880	1	5700
...	...	...	...	...	...	...
1533	3	51	3712	115280	1	5200
1534	1	74	3835	112000	1	4600
1535	2	51	2223	60457	1	7500
1536	1	51	2557	80750	1	5990
1537	2	51	1766	54276	1	7900

1389 rows × 6 columns

In [32]:
```
pd.get_dummies(s)
s
```

Out[32]:

	model	engine_power	age_in_days	km	previous_owners	price
0	1	51	882	25000	1	8900
1	2	51	1186	32500	1	8800
2	3	74	4658	142228	1	4200
3	1	51	2739	160000	1	6000
4	2	73	3074	106880	1	5700
...	...	...	...	...	...	...
1533	3	51	3712	115280	1	5200
1534	1	74	3835	112000	1	4600
1535	2	51	2223	60457	1	7500
1536	1	51	2557	80750	1	5990
1537	2	51	1766	54276	1	7900

1389 rows × 6 columns

In [33]:
```
y=s['price']
x=s.drop('price',axis=1)
x
```

Out[33]:

	model	engine_power	age_in_days	km	previous_owners
0	1	51	882	25000	1
1	2	51	1186	32500	1
2	3	74	4658	142228	1
3	1	51	2739	160000	1
4	2	73	3074	106880	1
...	...	...	...	...	...
1533	3	51	3712	115280	1
1534	1	74	3835	112000	1
1535	2	51	2223	60457	1
1536	1	51	2557	80750	1
1537	2	51	1766	54276	1

1389 rows × 5 columns

In [34]:
```
y
```

Out[34]:
```
0       8900
1       8800
2       4200
3       6000
4       5700
        ...
1533    5200
1534    4600
1535    7500
1536    5990
1537    7900
Name: price, Length: 1389, dtype: int64
```

train&split

In [35]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.1,random_state=42)
```

In [36]:
```
x_train.head()
```

Out[36]:

	model	engine_power	age_in_days	km	previous_owners
956	1	51	790	26210	1
1411	1	51	1461	46108	1
333	1	51	456	26526	1
1452	1	51	1247	75000	1
1369	1	51	701	36500	1

In [37]:
```
y_train.head()
```

Out[37]:
```
956     8750
1411    8000
333     9980
1452    8000
1369    9990
Name: price, dtype: int64
```

In [38]:
```
x_test.head()
```

Out[38]:

	model	engine_power	age_in_days	km	previous_owners
625	1	51	3347	148000	1
187	1	51	4322	117000	1
279	2	51	4322	120000	1
734	2	51	974	12500	1
315	1	51	1096	37000	1

```
In [39]: y_test.head()
```

```
Out[39]: 625     5400
187     5399
279     4900
734    10500
315     9300
Name: price, dtype: int64
```

elastic regression

```
In [40]: from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV

elastic = ElasticNet()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}

elastic_regressor = GridSearchCV(elastic, parameters)

elastic_regressor.fit(x_train, y_train)
```

```
Out[40]:
▸ GridSearchCV
▸ estimator: ElasticNet
  ▸ ElasticNet
```

```
In [41]: elastic_regressor.best_params_
```

```
Out[41]: {'alpha': 0.01}
```

```
In [42]: elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
```

```
Out[42]:
▾ ElasticNet
ElasticNet(alpha=0.01)
```

yprediction

```
In [43]: ypred=elastic.predict(x_test)
         ypred
```

```
Out[43]: array([ 5473.32539756,  5136.46111656,  4909.49119715,  9746.43736216,
                 9391.49673898, 10327.62418272,  9885.88836348,  8466.00949487,
                 5920.25378127, 10131.03008396,  5651.30859716,  7764.35604264,
                 9699.05851135,  4481.34000433,  6560.77154743,  9806.06819303,
                 7561.94514661,  5916.70308975, 10393.87844109,  7399.52559382,
                 8799.25116352,  8167.3083098 ,  9424.85077338, 10359.63581199,
                 9940.32568612, 10365.28987159,  9798.40192297,  7016.42291094,
                 9318.25428408, 10155.11191146,  5661.8150907 ,  9745.74321514,
                 4722.24018205,  9938.79382621,  9874.37622364,  8880.01111538,
                 3434.08327173, 10046.88646802,  8685.66690237,  7779.79615612,
                10187.79439164, 10427.95939687, 10406.67691559,  9692.41914975,
                 9378.79903264,  7303.19479857, 10437.19132798,  9063.62336089,
                10393.67202543,  8646.229118  , 10369.02325816,  8053.40528899,
                 5598.54702377, 10381.20107904,  5758.66881717,  8880.01111538,
                 9938.79382621,  4841.90855867,  8366.37937775,  6636.23171522,
                 6250.6192771 ,  4825.96755739, 10339.30865909,  7967.41501253,
                 4615.43382942,  9982.74444739,  8824.48548574,  5413.39977537,
                 9901.13944669,  7306.16652598, 10068.45690494,  8299.66162786,
                10319.99883458, 10376.53952529,  9703.95364197,  9555.77163773,
                10453.98497187,  9130.98427413, 10155.71876597,  9748.27792063,
                10336.54649034,  6752.19462187,  8697.87573075,  8373.84051105,
                 7135.07844023,  9818.18342413, 10362.1995098 ,  9662.46652604,
                 4538.09890343,  5743.71941274, 10142.81466275,  9454.26525766,
                 6038.98497585, 10222.26172086,  6510.85680368,  9771.32155621,
                10520.33040664,  8852.07102561, 10299.97651515, 10520.33040664,
                 9854.04985629,  6885.56394477,  9774.59461304,  9394.33014555,
                10266.90872589,  6936.66268046,  7298.18067252,  8351.06333562,
                10397.83544463,  7329.72752429,  7500.69200154, 10222.26172086,
                 6267.39649816,  4713.22041915,  5452.51588601, 10462.92965066,
                 9299.99540503, 10341.86756687,  9778.70642833,  5636.31666135,
                10445.37914933,  7167.3079715 , 10273.05681453,  7639.03893857,
                 9162.89139079,  8321.04285042, 10093.07678172,  8516.76724749,
                10492.96383312,  9699.05851135,  9612.97185122, 10310.57286151,
                10337.68764967,  8339.80778995, 10082.45359763,  9882.04118892,
                 9748.54737018,  6789.95018235,  6943.50510746])
```

mean squared error

In [44]:
```
from sklearn.metrics import mean_squared_error
e=mean_squared_error(ypred,y_test)
e
```

Out[44]: 618501.5707724169

root meansquare error

In [45]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[45]: 0.8452310245052245

In [46]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=ypred
results=results.reset_index()
results['ID']=results.index
results.head(10)
```

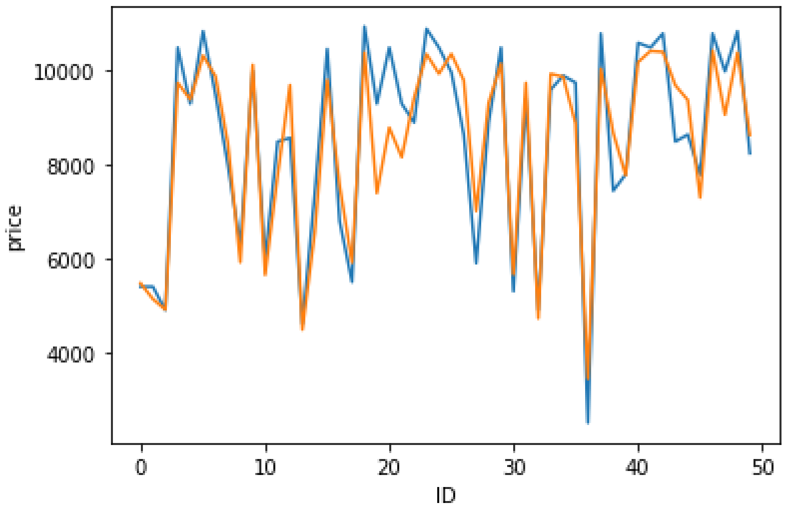
Out[46]:

	index	price	predicted	ID
0	625	5400	5473.325398	0
1	187	5399	5136.461117	1
2	279	4900	4909.491197	2
3	734	10500	9746.437362	3
4	315	9300	9391.496739	4
5	652	10850	10327.624183	5
6	1472	9500	9885.888363	6
7	619	7999	8466.009495	7
8	992	6300	5920.253781	8
9	1154	10000	10131.030084	9

graph plot

In [47]:
```
sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50))
plt.plot()
```

Out[47]: []



In [ ]:

In [ ]: