In [1]:
```
import pandas as pd
import pickle
import numpy as np
import warnings
warnings.filterwarnings('ignore') #for ignoring warnings
```

In [2]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv") #reading the data
data.describe()
```

Out[2]:

	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 [3]:
```
data1=data.drop(['lat','ID'],axis=1)
```

In [4]:
```
data2=data1.drop('lon',axis=1)
```

In [5]:
```
data2['model']=data['model'].map({'lounge':1,'pop':2,'sport':3})
```

In [6]:
```
#data2=pd.get_dummies(data2)
```

In [7]:
```
data2
```

Out[7]:

	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

1538 rows × 6 columns

In [8]:
```
y=data2['price']
x=data2.drop('price',axis=1) #droping the price column in data2 naming as x
```

In [9]:
```
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [10]:
```
x_test.head(5)
```

Out[10]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1

In [11]:
```
y_test.head(5)
```

Out[11]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [12]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
```

In [13]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [14]:
```
from sklearn.metrics import mean_squared_error#mean_squared error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[14]:
```
590569.9121697355
```

In [15]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)#finding the efficieny
```

Out[15]:
```
0.8391885506165899
```

In [16]:
```
a=data1.loc[data1.model=='lounge'] #getting the lounge model details
a
```

Out[16]:

	model	engine_power	age_in_days	km	previous_owners	lon	price
0	lounge	51	882	25000	1	8.611560	8900
3	lounge	51	2739	160000	1	17.634609	6000
6	lounge	51	731	11600	1	8.611560	10750
7	lounge	51	1521	49076	1	12.495650	9190
11	lounge	51	366	17500	1	7.704920	10990
...	...	...	...	...	...	...	...
1528	lounge	51	2861	126000	1	10.515310	5500
1529	lounge	51	731	22551	1	13.361120	9900
1530	lounge	51	670	29000	1	8.994500	10800
1534	lounge	74	3835	112000	1	8.666870	4600
1536	lounge	51	2557	80750	1	7.682270	5990

1094 rows × 7 columns

In [17]:
```
import seaborn as sns
import matplotlib.pyplot as plt
#results=a.DataFrame(column)

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

Out[17]:

	index	price	predicted	ID
0	481	7900	5987.682984	0
1	76	7900	7272.490419	1
2	1502	9400	9839.847697	2
3	669	8500	9696.775405	3
4	1409	9700	10012.040862	4
5	1414	9900	9628.286853	5
6	1089	9900	9646.945160	6
7	1507	9950	10090.960592	7
8	970	10700	9877.094341	8
9	1198	8999	9326.088982	9

In [18]:
```
results['actual price']=results.apply(lambda column:column.price-column.predicted,axis=1) #using lambda
results
```

Out[18]:

	index	price	predicted	ID	actual price
0	481	7900	5987.682984	0	1912.317016
1	76	7900	7272.490419	1	627.509581
2	1502	9400	9839.847697	2	-439.847697
3	669	8500	9696.775405	3	-1196.775405
4	1409	9700	10012.040862	4	-312.040862
...	...	...	...	...	...
503	291	10900	10005.311518	503	894.688482
504	596	5699	6400.852430	504	-701.852430
505	1489	9500	10096.776914	505	-596.776914
506	1436	6990	8358.743798	506	-1368.743798
507	575	10900	10343.148204	507	556.851796

508 rows × 5 columns

In [19]:
```
from sklearn.linear_model import ElasticNet
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) #gettng elastic_net
elastic_regressor.fit(x_train, y_train)
```

Out[19]:

▸ **GridSearchCV**

▸ **estimator: ElasticNet**

▸ ElasticNet

In [20]:
```
from sklearn.model_selection import GridSearchCV
```

In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt
#results=a.DataFrame(column)

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

Out[21]:

	index	price	predicted	ID
0	481	7900	5987.682984	0
1	76	7900	7272.490419	1
2	1502	9400	9839.847697	2
3	669	8500	9696.775405	3
4	1409	9700	10012.040862	4
5	1414	9900	9628.286853	5
6	1089	9900	9646.945160	6
7	1507	9950	10090.960592	7
8	970	10700	9877.094341	8
9	1198	8999	9326.088982	9

In [22]:
```
elastic=ElasticNet(alpha=.01)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [23]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

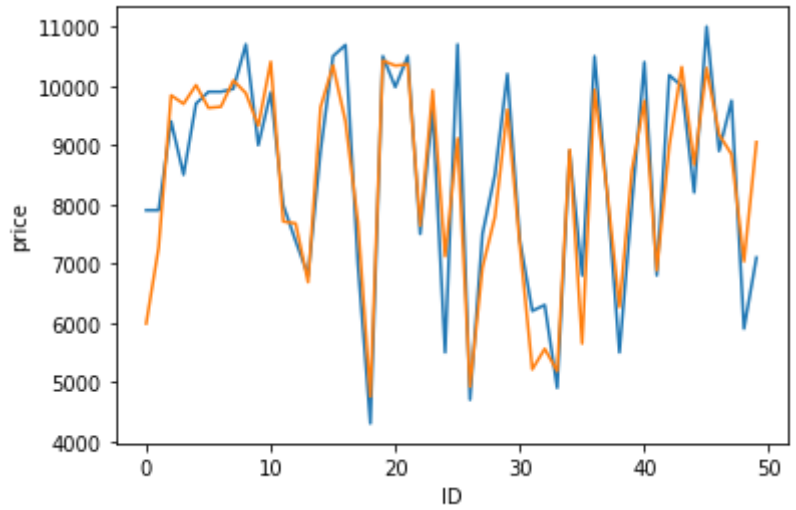
Out[23]:
```
0.8385500526604823
```

In [24]:
```
elastic_Error=mean_squared_error(y_pred_elastic,y_test) #getting elastic error
elastic_Error
```

Out[24]:
```
592914.7556700263
```

In [25]:
```
sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50)) #ploting the values in the graph
plt.plot()
```

Out[25]:
```
[]
```



In [ ]:
```

```