In [21]:
```
import pandas as pd
```

In [22]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [23]:
```
data.describe()
```

Out[23]:

	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 [24]:
```
data.head(100)
```

Out[24]:

	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
...	...	...	...	...	...	...	...	...	...
95	96	sport	51	4292	165600	1	44.715408	11.308300	5950
96	97	pop	51	1066	28000	1	41.769051	12.662810	8500
97	98	sport	51	2009	86000	2	40.633171	17.634609	7800
98	99	lounge	51	456	18592	2	45.393600	10.482240	10900
99	100	pop	51	731	41558	2	45.571220	9.159140	8790

100 rows × 9 columns

In [25]:
```
data=data.loc[(data.previous_owners==1)]
```

In [26]:
```
data
```

Out[26]:

	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 [27]:
```
data=data.drop(['lat','lon','ID'],axis=1)
```

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

```
Out[28]:
        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 [29]: data=pd.get_dummies(data)
In [30]: data
```

```
Out[30]:
      engine_power  age_in_days      km  previous_owners  price  model_lounge  model_pop  model_sport
   0            51          882   25000                1   8900             1          0            0
   1            51         1186   32500                1   8800             0          1            0
   2            74         4658  142228                1   4200             0          0            1
   3            51         2739  160000                1   6000             1          0            0
   4            73         3074  106880                1   5700             0          1            0
 ...           ...          ...     ...              ...    ...           ...        ...          ...
1533            51         3712  115280                1   5200             0          0            1
1534            74         3835  112000                1   4600             1          0            0
1535            51         2223   60457                1   7500             0          1            0
1536            51         2557   80750                1   5990             1          0            0
1537            51         1766   54276                1   7900             0          1            0

1389 rows × 8 columns
```

```
In [31]: y=data['price']
```

```
In [32]: x=data.drop('price',axis=1)
```

In [33]:

```
y
```

Out[33]:

```
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
```

In [34]:

```
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 [35]:

```
x_test.head(5)
```

Out[35]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
625	51	3347	148000	1	1	0	0
187	51	4322	117000	1	1	0	0
279	51	4322	120000	1	0	1	0
734	51	974	12500	1	0	1	0
315	51	1096	37000	1	1	0	0

In [36]:
```
y_test.head(5)
```

Out[36]:
```
625     5400
187     5399
279     4900
734    10500
315     9300
Name: price, dtype: int64
```

In [38]:
```
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)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_coordinate_descent.py:631: C
onvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check
the scale of the features or consider increasing regularisation. Duality gap: 2.860e+08, tolerance: 3.519
e+05
  model = cd_fast.enet_coordinate_descent(
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_coordinate_descent.py:631: C
onvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check
the scale of the features or consider increasing regularisation. Duality gap: 2.750e+08, tolerance: 3.611
e+05
  model = cd_fast.enet_coordinate_descent(
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_coordinate_descent.py:631: C
onvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check
the scale of the features or consider increasing regularisation. Duality gap: 2.703e+08, tolerance: 3.517
e+05
  model = cd_fast.enet_coordinate_descent(
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_coordinate_descent.py:631: C
onvergenceWarning: Objective did not converge. You might want to increase the number of iterations, check
the scale of the features or consider increasing regularisation. Duality gap: 2.854e+08, tolerance: 3.711
e+05
  model = cd_fast.enet_coordinate_descent(
```

In [39]:
```
elastic_regressor.best_params_
```

Out[39]:
```
{'alpha': 0.01}
```

```
In [40]: elastic=ElasticNet(alpha=0.1)
         elastic.fit(x_train,y_train)
         y_pred_elastic=elastic=elastic.predict(x_test)
```

```
In [41]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred_elastic)
```

```
Out[41]: 0.8488205369102257
```

```
In [43]: from sklearn.metrics import mean_squared_error
         elastic_Error=mean_squared_error(y_pred_elastic,y_test)
         elastic_Error
```

```
Out[43]: 604156.8414511626
```

In [45]:

```
Results=pd.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=y_pred_elastic
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(15)
```

Out[45]:

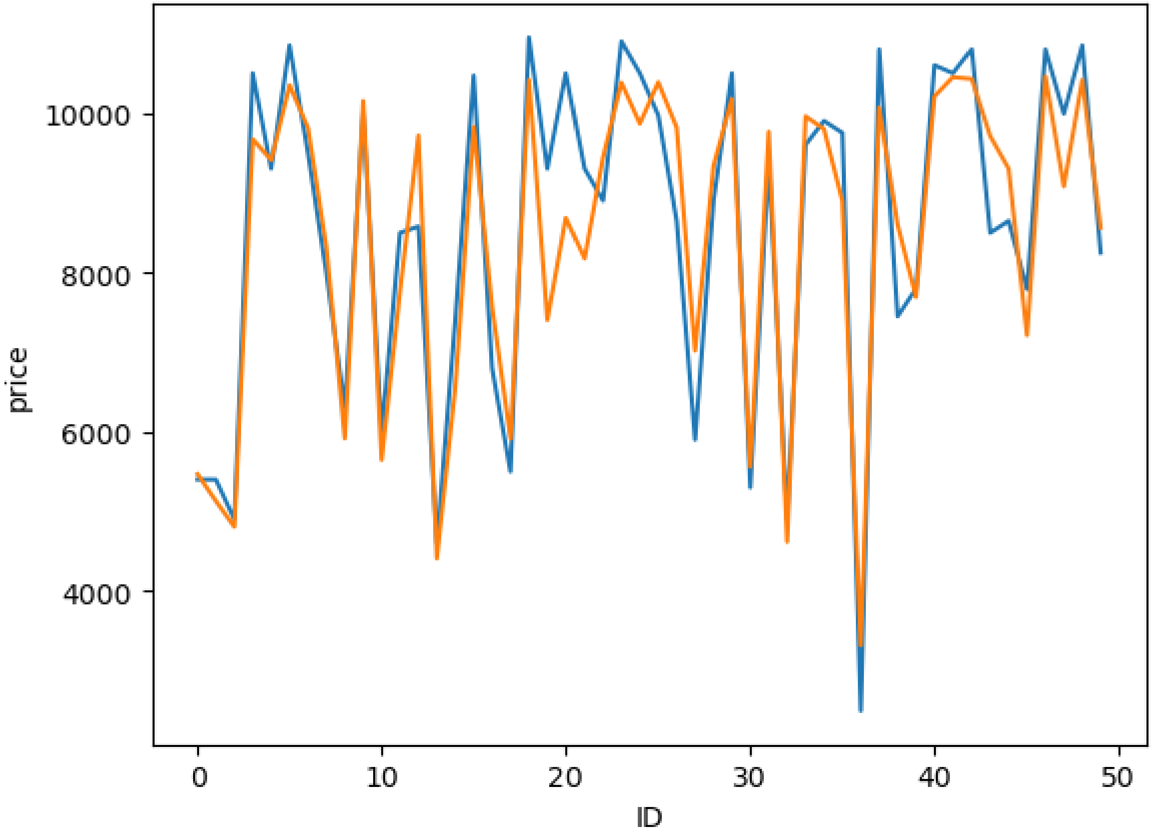
	index	price	predicted	ID
0	625	5400	5467.419026	0
1	187	5399	5128.262319	1
2	279	4900	4805.437183	2
3	734	10500	9669.896114	3
4	315	9300	9407.643214	4
5	652	10850	10349.137010	5
6	1472	9500	9810.214625	6
7	619	7999	8312.374646	7
8	992	6300	5916.485067	8
9	1154	10000	10151.323763	9
10	757	6000	5646.011712	10
11	1299	8500	7771.402126	11
12	400	8580	9717.015852	12
13	314	4600	4408.147383	13
14	72	7400	6560.282727	14

In [46]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='price',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[46]:

```
[]
```



In [ ]: