In [201]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [202]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [203]:
```
data.head(10)
```

Out[203]:

	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
5	6	pop	74	3623	70225	1	45.000702	7.682270	7900
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
8	9	sport	73	4049	76000	1	45.548000	11.549470	5600
9	10	sport	51	3653	89000	1	45.438301	10.991700	6000

In [204]:
```
data1=data.loc[(data.previous_owners==1)]
```

In [205]:
```
data1
```

Out[205]:

	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 [206]:
```
data1=data.drop(['ID','lat','lon'],axis=1)
```

In [207]:
```
data1
```

Out[207]:

	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

1538 rows × 6 columns

In [208]:
```
data1=pd.get_dummies(data)
```

In [209]:
```
data1
```

Out[209]:

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

1538 rows × 11 columns

In [210]:
```
data1.shape    #data['moddel']=data['model'].map(('longue':1,'pop':2))
```

Out[210]:
```
(1538, 11)
```

In [211]:
```
y=data1['price']
x=data1.drop('price',axis=1)
```

In [212]:
```
y
```

Out[212]:
```
0       8900
1       8800
2       4200
3       6000
4       5700
        ...
1533    5200
1534    4600
1535    7500
1536    5990
1537    7900
Name: price, Length: 1538, dtype: int64
```

In [213]:
```
x
```

Out[213]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	model_lounge	model_pop	model_sport
0	1	51	882	25000	1	44.907242	8.611560	1	0	0
1	2	51	1186	32500	1	45.666359	12.241890	0	1	0
2	3	74	4658	142228	1	45.503300	11.417840	0	0	1
3	4	51	2739	160000	1	40.633171	17.634609	1	0	0
4	5	73	3074	106880	1	41.903221	12.495650	0	1	0
...	...	...	...	...	...	...	...	...	...	...
1533	1534	51	3712	115280	1	45.069679	7.704920	0	0	1
1534	1535	74	3835	112000	1	45.845692	8.666870	1	0	0
1535	1536	51	2223	60457	1	45.481541	9.413480	0	1	0
1536	1537	51	2557	80750	1	45.000702	7.682270	1	0	0
1537	1538	51	1766	54276	1	40.323410	17.568270	0	1	0

1538 rows × 10 columns

In [214]:
```
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
```

In [215]:
```
x_test.head()
```

Out[215]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
481	482	pop	51	3197	120000	2	40.174702	18.167629
76	77	pop	62	2101	103000	1	45.797859	8.644440
1502	1503	lounge	51	670	32473	1	41.107880	14.208810
669	670	lounge	51	913	29000	1	45.778591	8.946250
1409	1410	lounge	51	762	18800	1	45.538689	9.928310

In [216]:
```
x_train.head()
```

Out[216]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
1047	1048	pop	51	1552	54000	1	43.463539	11.87765
15	16	lounge	51	1096	28200	1	45.697208	9.84597
585	586	lounge	51	640	40438	1	41.996349	12.72344
381	382	pop	51	397	17500	1	42.104679	14.70599
175	176	lounge	51	456	19133	1	45.393600	10.48224

In [217]:
```
y_test.head()
```

Out[217]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [218]:
```
y_train.head()
```

Out[218]:
```
1047    11000
15       9500
585      9800
381      9800
175     10900
Name: price, dtype: int64
```

In [233]:
```
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 [234]:
```
from sklearn.model_selection import GridSearchCV
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)

elastic_regressor.fit(x_train, y_train)
```

Out[234]:
```
▸ GridSearchCV
▸ estimator: ElasticNet
  ▸ ElasticNet
```

In [235]:
```
elastic_regressor.best_params_
```

Out[235]:
```
{'alpha': 0.01}
```

In [236]:
```
elastic=ElasticNet(alpha=30)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [237]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[237]:
```
0.8416206414238153
```

In [238]:
```
from sklearn.metrics import mean_squared_error
Elastic_Error=mean_squared_error(y_pred_elastic,y_test)
Elastic_Error
```

Out[238]:
```
581638.2119710302
```

In [239]:
```
Results=pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred_elastic
Results=Results.reset_index()
Results['Id']=Results.index
Results
```

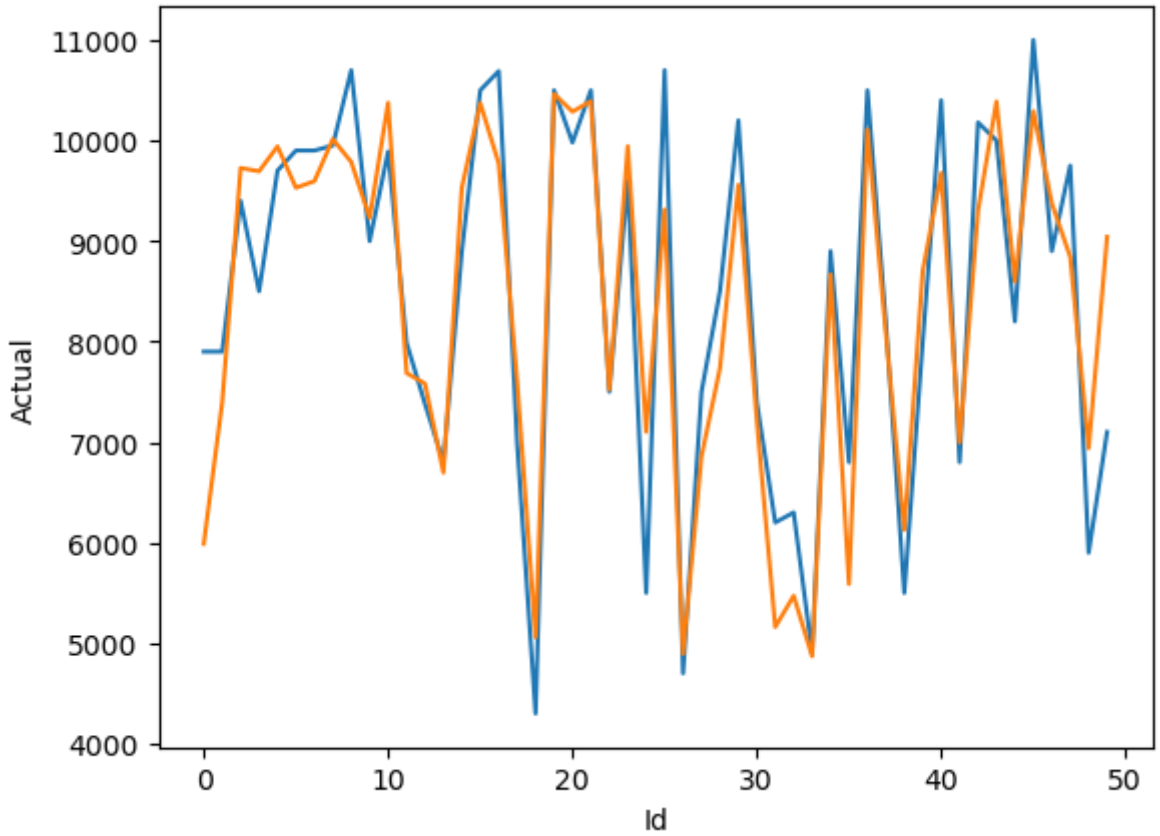
Out[239]:

	index	Actual	Predicted	Id
0	481	7900	5988.777085	0
1	76	7900	7393.904731	1
2	1502	9400	9726.595326	2
3	669	8500	9693.681751	3
4	1409	9700	9940.773084	4
...	...	...	...	...
503	291	10900	10028.732370	503
504	596	5699	6516.798511	504
505	1489	9500	10209.647976	505
506	1436	6990	8224.153844	506
507	575	10900	10329.915814	507

508 rows × 4 columns

In [240]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[240]: []



In [ ]:

```

```