In [36]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [37]:
```
data= pd.read_csv('/home/placement/Desktop/fiat500.csv')
data
```

Out[37]:

	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

1538 rows × 9 columns

In [38]:
```
data.describe()
```

Out[38]:

	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 [39]:
```
data1=data.loc[(data.previous_owners==1)]    #data with only previous_owners=1 using loc()
```

In [40]:
```
data1
```

Out[40]:

	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 [41]:
```
data1=data1.drop(columns=['ID','lat','lon']) #droping unwanted columns
```

In [42]:
```
data1
```

Out[42]:

	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 [43]:
```
data1=pd.get_dummies(data1) #covert the strings into numbers of model using get_dummies()
```

In [44]:
```
data1
```

Out[44]:

	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 [45]:
```
y=data1['price']                        #copy the price column of data1 into the y
x=data1.drop(columns='price')           #drop the price column from data1
```

In [46]:
```
x
```

Out[46]:

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

1389 rows × 7 columns

In [47]:
```
y
```

Out[47]:
```
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
```

splitting the data into training set and testing set

In [48]:
```
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 [49]:
```
x_train
```

Out[49]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
915	51	397	17081	1	1	0	0
12	51	456	18450	1	1	0	0
638	51	397	21276	1	1	0	0
190	51	821	19000	1	1	0	0
701	51	701	27100	1	1	0	0
...	...	...	...	...	...	...	...
1201	51	790	50740	1	0	1	0
1239	51	4383	107600	1	0	1	0
1432	51	701	42095	1	1	0	0
951	51	3684	78000	1	1	0	0
1235	51	1613	45000	1	1	0	0

930 rows × 7 columns

In [50]:
```
x_test
```

Out[50]:

	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
...	...	...	...	...	...	...	...
115	51	397	16135	1	1	0	0
370	51	366	11203	1	0	1	0
1179	74	3804	62000	1	1	0	0
93	51	397	17250	1	1	0	0
147	51	762	15917	1	1	0	0

459 rows × 7 columns

In [51]:
```
y_train
```

Out[51]:
```
915     10900
12       9700
638     10850
190      9990
701     10300
        ...  
1201     8300
1239     3950
1432     8900
951      6500
1235     8800
Name: price, Length: 930, dtype: int64
```

In [52]:
```
y_test
```

Out[52]:
```
625      5400
187      5399
279      4900
734     10500
315      9300
        ...  
115     10650
370      9900
1179     5900
93      10050
147      9900
Name: price, Length: 459, dtype: int64
```

ElasticNet Model

In [53]:
```
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV
elastic=ElasticNet()                      #creating an object for ElasticNet
parameters={'alpha':[1e-15,1e-10,1e-8,1e-4,1e-3,1e-2,15,10,20]}
elastic_regressor=GridSearchCV(elastic,parameters)
elastic_regressor.fit(x_train,y_train)    #training and fitting
```

Out[53]:
```
▸      GridSearchCV
▸ estimator: ElasticNet
      ▸ ElasticNet
```

In [54]:
```
elastic_regressor.best_params_
```

Out[54]:
```
{'alpha': 0.01}
```

In [55]:
```
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred=elastic.predict(x_test)
```

In [56]:
```
from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred,y_test)
elastic_Error
```

Out[56]:
```
515349.9787871871
```

In [57]:
```
from sklearn.metrics import r2_score    #to know the efficiency of the predicted price
r2_score(y_test,y_pred)
```

Out[57]:
```
0.8602162350730707
```

In [58]:
```
Results=pd.DataFrame(columns=['Actual','Predicted']) #create the dataframe for actual and predicted values
Results['Actual']=y_test
Results['Predicted']=y_pred
Results=Results.reset_index()    #remove the index as ID values
Results['id']=Results.index
```

In [59]:
```
Results
```

Out[59]:

	index	Actual	Predicted	id
0	625	5400	5482.171479	0
1	187	5399	5127.531740	1
2	279	4900	4803.203231	2
3	734	10500	9662.825235	3
4	315	9300	9408.645424	4
...	...	...	...	...
454	115	10650	10396.366249	454
455	370	9900	10235.109546	455
456	1179	5900	6766.292878	456
457	93	10050	10377.386719	457
458	147	9900	10069.771989	458

459 rows × 4 columns

In [60]:
```
Results["Difference"]=Results['Actual']-Results['Predicted']  #add the column for difference b/w the actua
```

In [61]:
```
Results
```

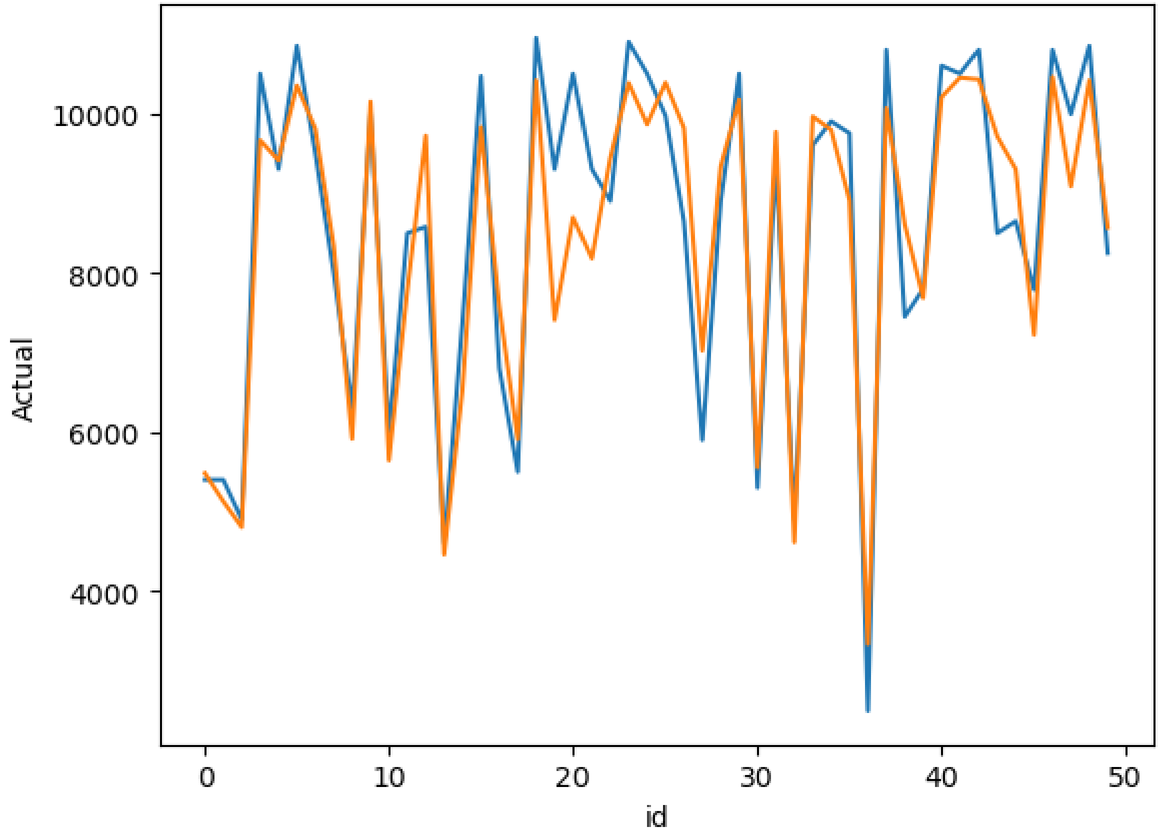
Out[61]:

	index	Actual	Predicted	id	Difference
0	625	5400	5482.171479	0	-82.171479
1	187	5399	5127.531740	1	271.468260
2	279	4900	4803.203231	2	96.796769
3	734	10500	9662.825235	3	837.174765
4	315	9300	9408.645424	4	-108.645424
...	...	...	...	...	...
454	115	10650	10396.366249	454	253.633751
455	370	9900	10235.109546	455	-335.109546
456	1179	5900	6766.292878	456	-866.292878
457	93	10050	10377.386719	457	-327.386719
458	147	9900	10069.771989	458	-169.771989

459 rows × 5 columns

Plot the data using seaborn and matplotlib libraries

```
In [62]: 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.show()
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```