In [103]: `import pandas as pd`

In [104]: `data=pd.read_csv("/home/placement/Downloads/fiat500.csv")#read data from file as csv`

In [105]: `data.describe()#describe data like count,mean,max value`

Out[105]:

	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 [106]: `data.tail(10)#shows last 10 rows`

Out[106]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1528	1529	lounge	51	2861	126000	1	43.841980	10.51531	5500
1529	1530	lounge	51	731	22551	1	38.122070	13.36112	9900
1530	1531	lounge	51	670	29000	1	45.764648	8.99450	10800
1531	1532	sport	73	4505	127000	1	45.528511	9.59323	4750
1532	1533	pop	51	1917	52008	1	45.548000	11.54947	9900
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [107]: `data1=data.drop(['lat','ID','lon'],axis=1)#remove column of lat,id,lon by using drop function`

In [108]: `data1`

Out[108]:

	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 [109]: `data2=pd.get_dummies(data1)`*#where the lounge model it shows"1" other models it shows "0"*
`data2`

Out[109]:

	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

1538 rows × 8 columns

In [110]: `data2.shape`*#how many rows and columns in the data frame*

Out[110]: `(1538, 8)`

In [111]: `z=data2.loc[(data.previous_owners==1)]`*#determine only for lounge cars*

In [112]: z

Out[112]:

	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 [113]:
```python
y=z['price']#removing price from data2 and put in new data frame y
x=z.drop(['price'],axis=1)#remaining data can be put in another data fram
```

In [114]:
```python
x# to get data in xmport 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[114]:

	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 [115]: `y# to get data in y`

Out[115]:
```
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 [116]:
```python
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)#dividing training data ar
```

In [117]: `x_test.head(5)#display top 5 data in testing data`

Out[117]:

	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 [118]: `y_test.head(5)#display top 5 data in testing data price dataframe`

Out[118]:
```
625      5400
187      5399
279      4900
734     10500
315      9300
Name: price, dtype: int64
```

In [119]: `x_train.head(5)#display top 5 data in training data`

Out[119]:

	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

In [120]: `y_train.head(5)#display top 5 data in training data price dataframe`

Out[120]:
```
915     10900
12       9700
638     10850
190      9990
701     10300
Name: price, dtype: int64
```

In [121]:
```python
from sklearn.linear_model import ElasticNet#elastic net

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[121]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

In [128]:
```python
elastic_regressor.best_params_#finding alpha value
```

Out[128]: {'alpha': 0.01}

In [123]:
```python
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [124]:
```python
from sklearn.metrics import mean_squared_error
Elasticnet_Error=mean_squared_error(y_pred_elastic,y_test)
Elasticnet_Error
```

Out[124]: 515349.978787187

In [125]:
```python
from sklearn.metrics import r2_score#accuracy
r2_score(y_test,y_pred_elastic)
```

Out[125]: 0.8602162350730707

In [126]:
```python
Results=pd.DataFrame(columns=['Actual','predicted'])#different data frames for actual predicted
Results['Actual']=y_test
Results['predicted']=y_pred_elastic
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(10)
```

Out[126]:

	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
5	652	10850	10350.952605	5
6	1472	9500	9806.127960	6
7	619	7999	8341.142824	7
8	992	6300	5913.786719	8
9	1154	10000	10149.093829	9

In [127]:
```python
import seaborn as sns#plotting data
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()
```
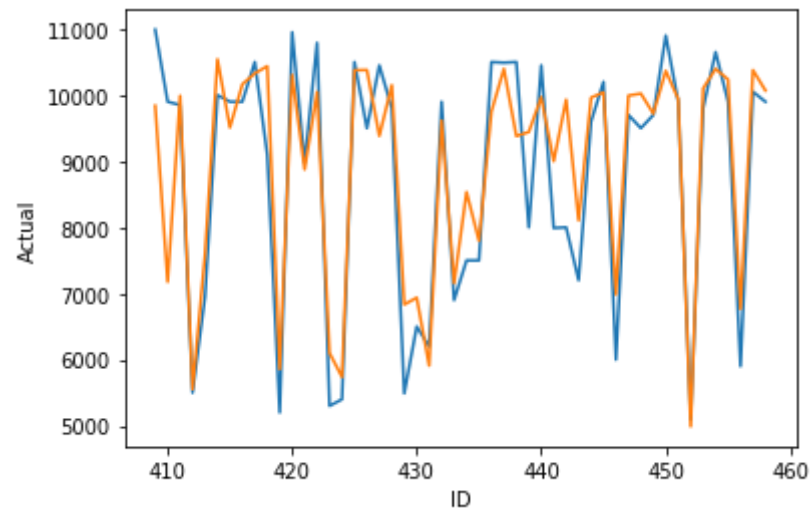
Out[127]: []
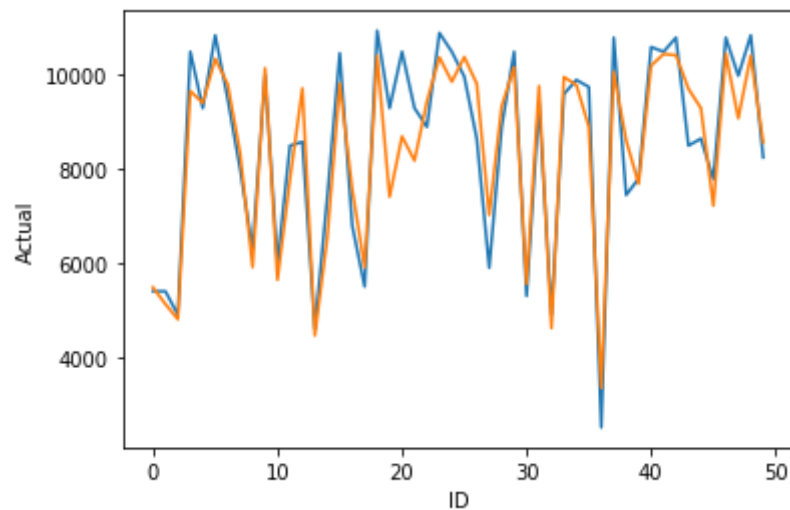
In [130]:
```python
import seaborn as sns#ploting data
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='Actual',data=Results.tail(50))
sns.lineplot(x='ID',y='predicted',data=Results.tail(50))
plt.plot()
```

Out[130]: []

In [ ]: