In [73]:

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [74]:

```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [75]:

```
data.describe()
```

Out[75]:

	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 [76]:

```
data.head()
```

Out[76]:

	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

```
In [77]: data1=data.drop(['lat','lon','ID'],axis=1)#unwanted columns removed
         data1
```

```
Out[77]:
         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 [78]:
```
data2=pd.get_dummies(data1)
data2
```

Out[78]:

	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 [79]:
```
data2.shape
```

Out[79]:
```
(1538, 8)
```

In [80]:
```
y=data2['price']
x=data2.drop('price',axis=1)
#predicted value we removed from data frame
```

In [81]:
```
y
```

Out[81]:
```
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 [82]:
```
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)
#divide the columns into testing and training
```

In [83]:
```
x_test.head(5)
```

Out[83]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
481	51	3197	120000	2	0	1	0
76	62	2101	103000	1	0	1	0
1502	51	670	32473	1	1	0	0
669	51	913	29000	1	1	0	0
1409	51	762	18800	1	1	0	0

```
In [84]: y_test.head(5)
```

```
Out[84]: 481     7900
         76      7900
         1502    9400
         669     8500
         1409    9700
         Name: price, dtype: int64
```

```
In [85]: x_train.head(5)
```

Out[85]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
527	51	425	13111	1	1	0	0
129	51	1127	21400	1	1	0	0
602	51	2039	57039	1	0	1	0
331	51	1155	40700	1	1	0	0
323	51	425	16783	1	1	0	0

```
In [86]: y_train.head(5)
```

```
Out[86]: 527    9990
         129    9500
         602    7590
         331    8750
         323    9100
         Name: price, dtype: int64
```

```
In [87]: # for linear regression
         from sklearn.linear_model import LinearRegression
         reg=LinearRegression() #creating object of LinearRegression
         reg.fit(x_train,y_train) #training and fitting LR object using training data
```

```
Out[87]: LinearRegression()
```

```
In [88]: y_pred=reg.predict(x_test)
```

In [89]:
```
y_pred
```

```
          8456.30006203,  6499.76668237,  7768.57829985,  6832.86406122,
          8347.96113362, 10439.02404036,  7356.43463051,  8562.56562053,
          9820.78555199, 10035.83571539,  7370.77198022,  9411.45894006,
         10352.85155564,  8045.21588007, 10446.80664758,  3736.20118868,
         10348.63930496, 10435.96627494,  6167.80169017, 10390.11317804,
          6527.69471073,  9116.4755691 , 10484.52829   ,  9335.69889855,
          6709.57413543,  3390.72353093, 10106.33753331,  9792.46732008,
          6239.49568346,  4996.26346266,  9044.38667681,  9868.09959448,
          5484.13199252,  5698.5954821 , 10086.86206874,  8115.81693479,
         10392.37800936,  6835.6573351 ,  6657.61744836,  5738.50576764,
          8896.80120764,  9952.37340054, 10390.28377419,  9419.10788866,
          9082.56591129, 10122.82465116, 10410.00504522, 10151.77663915,
          9714.85367238,  9291.92963633, 10346.99073888,  5384.22311343,
          9772.85146492,  6069.77107828,  9023.26394782, 10220.56195956,
          9238.89392583,  9931.47195375,  8321.42715662,  8377.80491069,
          7528.53327408, 10552.64805598, 10465.02437243, 10110.68940664,
         10238.17869436,  6841.77264488,  9625.64505547, 10412.59988875,
          9653.06224923,  7948.63618724,  9704.82523573,  7971.05970955,
         10399.51752022,  9176.43567301,  5803.03205787,  6698.19524313,
```

In [90]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[90]:
```
0.8415526986865394
```

In [91]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_pred,y_test)
```

Out[91]:
```
581887.727391353
```

In [92]:

```
Results=pd.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=y_pred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[92]:

	index	price	predicted	Id
0	481	7900	5867.650338	0
1	76	7900	7133.701423	1
2	1502	9400	9866.357762	2
3	669	8500	9723.288745	3
4	1409	9700	10039.591012	4
5	1414	9900	9654.075826	5
6	1089	9900	9673.145630	6
7	1507	9950	10118.707281	7
8	970	10700	9903.859527	8
9	1198	8999	9351.558284	9

linear regression ends

```
In [93]: #for ridge regression
         from sklearn.model_selection import GridSearchCV
         from sklearn.linear_model import Ridge
         alpha=[1e-15,1e-10,1e-8,1e-4,1e-3,1e-2,1,5,10,20,30]
         ridge=Ridge()
         parameters={'alpha':alpha}
         ridge_regressor=GridSearchCV(ridge,parameters)
         ridge_regressor.fit(x_train,y_train)

Out[93]: GridSearchCV(estimator=Ridge(),
                      param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                            5, 10, 20, 30]})
```

```
In [94]: ridge_regressor.best_params_

Out[94]: {'alpha': 30}
```

```
In [95]: ridge=Ridge(alpha=30)
         ridge.fit(x_train,y_train)
         y_pred_ridge=ridge.predict(x_test)
```

```
In [96]: from sklearn.metrics import mean_squared_error
         Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
         Ridge_Error

Out[96]: 579521.7970897449
```

```
In [97]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred_ridge)

Out[97]: 0.8421969385523054
```

In [98]:

```
Results=pd.DataFrame(columns=['Actual','predicted'])
Results['Actual']=y_test
#Results=pd.DataFrame(columns=['price','predicted'])
#Results['price']=y_test
Results['predicted']=y_pred_ridge
#Results['km']=x_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[98]:

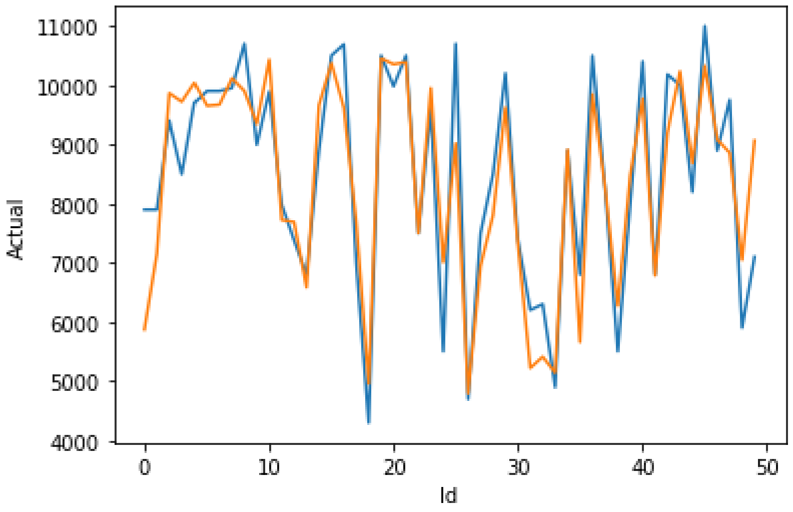
	index	Actual	predicted	Id
0	481	7900	5869.741155	0
1	76	7900	7149.563327	1
2	1502	9400	9862.785355	2
3	669	8500	9719.283532	3
4	1409	9700	10035.895686	4
5	1414	9900	9650.311090	5
6	1089	9900	9669.183317	6
7	1507	9950	10115.128380	7
8	970	10700	9900.241944	8
9	1198	8999	9347.080772	9

In [99]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [100]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))#blue
sns.lineplot(x='Id',y='predicted',data=Results.head(50))#orange
plt.plot
```

Out[100]:
```
<function matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)>
```



Ridge regression ends

In [101]:

```
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[101]:

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

In [102]:

```
elastic_regressor.best_params_
```

Out[102]:

```
{'alpha': 0.01}
```

In [103]:

```
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [104]:

```
from sklearn.metrics import mean_squared_error
Elastic_Error=mean_squared_error(y_pred_elastic,y_test)
Elastic_Error
```

Out[104]:

```
581390.7642825295
```

In [105]:

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[105]:

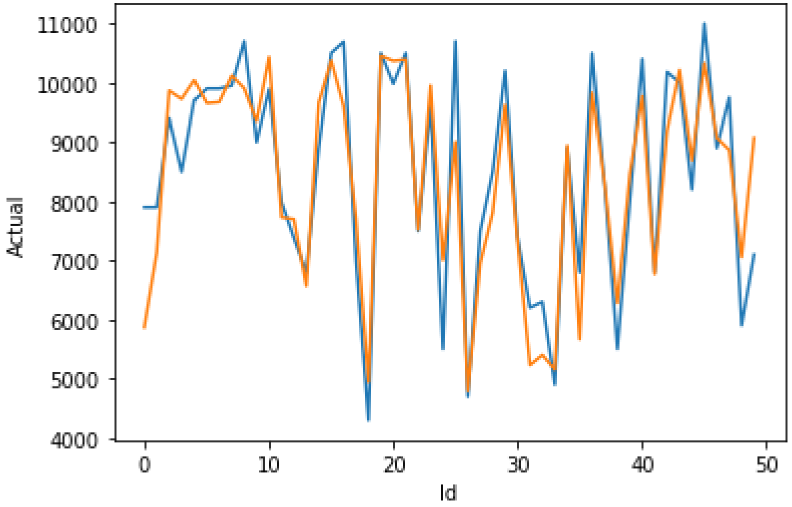
```
0.841688021120299
```

```
In [106]: Results=pd.DataFrame(columns=['Actual','predicted'])
          Results['Actual']=y_test
          #Results=pd.DataFrame(columns=['price','predicted'])
          #Results['price']=y_test
          Results['predicted']=y_pred_elastic
          #Results['km']=x_test['km']
          Results=Results.reset_index()
          Results['Id']=Results.index
          Results.head(10)
```

```
Out[106]:
     index  Actual     predicted  Id
  0    481    7900   5867.742075   0
  1     76    7900   7136.527402   1
  2   1502    9400   9865.726723   2
  3    669    8500   9722.573593   3
  4   1409    9700  10038.936496   4
  5   1414    9900   9653.407122   5
  6   1089    9900   9672.438692   6
  7   1507    9950  10118.075470   7
  8    970   10700   9903.219809   8
  9   1198    8999   9350.750929   9
```

In [107]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='Actual',data=Results.head(50))#blue
sns.lineplot(x='Id',y='predicted',data=Results.head(50))#orange
plt.plot
```

Out[107]:
```
<function matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)>
```



Elastic Net ends

In [ ]: