In [161]:
```python
import pandas as pd
```

In [162]:
```python
data=pd.read_csv("fiat500 (1).csv")
```

In [163]:
```python
data
```

Out[163]:

	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 [164]:
```python
data=data.drop(['lat','lon','ID',],axis=1)
```

In [165]:
```
data
```

Out[165]:

	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 [166]:
```
data=pd.get_dummies(data)
```

In [167]:
```
data.shape
#data['model'] = data['model'].map({'lounge':1,'pop':2,'sport':3})
```

Out[167]:
```
(1538, 8)
```

In [168]:
```
data
```

Out[168]:

	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 [169]:
```
y=data['price']
x=data.drop('price',axis=1)
```

In [170]:
```
y
```

Out[170]:
```
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 [171]:
```
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 [172]:
```
X_test.head(5)
```

Out[172]:

	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 [173]:
```
X_train.shape
```

Out[173]:
```
(1030, 7)
```

In [174]:
```
y_train
```

Out[174]:
```
527      9990
129      9500
602      7590
331      8750
323      9100
        ...
1130    10990
1294     9800
860      5500
1459     9990
1126     8900
Name: price, Length: 1030, dtype: int64
```

In [175]:
```
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[175]:
```
▾ LinearRegression
LinearRegression()
```

In [176]:
```
ypred=reg.predict(X_test)
```

In [177]:
```
ypred
```

```
        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,
         8257.83550573, 10452.95284574,  9948.66454584,  9789.65062843,
        10582.50828537,  7568.91955482,  6804.97705225,  8065.01292384,
        10310.29143419,  8836.34894739,  8390.05091229,  9582.13932508,
         9745.34784981, 10045.45021387, 10294.09872915,  7145.15315349
```

In [178]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[178]: 0.8415526986865394

In [179]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(ypred,y_test)
```

Out[179]: 581887.727391353

In [180]:
```
#from sklearn.metrics import accuracy_score
#accuracy_score(y_test,ypred)
```

In [181]:

```
Results= pd.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(15)
```

Out[181]:

	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
10	1088	9890	10434.349636	10
11	576	7990	7732.262557	11
12	965	7380	7698.672401	12
13	1488	6800	6565.952404	13
14	1432	8900	9662.901035	14

In [182]:
```
Results= pd.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(25)
```

Out[182]:

	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
10	1088	9890	10434.349636	10
11	576	7990	7732.262557	11
12	965	7380	7698.672401	12
13	1488	6800	6565.952404	13
14	1432	8900	9662.901035	14
15	380	10500	10373.203443	15
16	754	10690	9599.948445	16
17	30	6990	7699.344004	17
18	49	4300	4941.330180	18
19	240	10500	10455.271948	19
20	344	9980	10370.515557	20

	index	Price	Predicted	Id
21	354	10500	10391.604244	21
22	124	7500	7529.066225	22
23	383	9600	9952.373401	23
24	1389	5500	7006.138457	24

In [183]:
```
data.describe()
```

Out[183]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	51.904421	1650.980494	53396.011704	1.123537	8576.003901	0.711313	0.232770	0.055917
std	3.988023	1289.522278	40046.830723	0.416423	1939.958641	0.453299	0.422734	0.229836
min	51.000000	366.000000	1232.000000	1.000000	2500.000000	0.000000	0.000000	0.000000
25%	51.000000	670.000000	20006.250000	1.000000	7122.500000	0.000000	0.000000	0.000000
50%	51.000000	1035.000000	39031.000000	1.000000	9000.000000	1.000000	0.000000	0.000000
75%	51.000000	2616.000000	79667.750000	1.000000	10000.000000	1.000000	0.000000	0.000000
max	77.000000	4658.000000	235000.000000	4.000000	11100.000000	1.000000	1.000000	1.000000

In [184]:
```
# ridge regression
```

In [185]:
```
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)
```

```
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=5.56109e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.70876e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.91585e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.08003e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.01022e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57959e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.24161e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.92759e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09091e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
```

```
Ill-conditioned matrix (rcond=7.02112e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57414e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.23284e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.9277e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09099e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.02123e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57407e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.23274e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[185]:

```
▸ GridSearchCV
▸ estimator: Ridge
    ▸ Ridge
```

In [186]:

```
ridge_regressor.best_params_
```

Out[186]:

```
{'alpha': 30}
```

In [187]:

```
ridge=Ridge(alpha=30)
ridge.fit(X_train,y_train)
y_pred_ridge=ridge.predict(X_test)
```

In [188]:
```
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[188]: 579521.7970897449

In [189]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[189]: 0.8415526986865394

In [190]:
```
Results= pd.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=y_pred_ridge
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(25)
```

Out[190]:

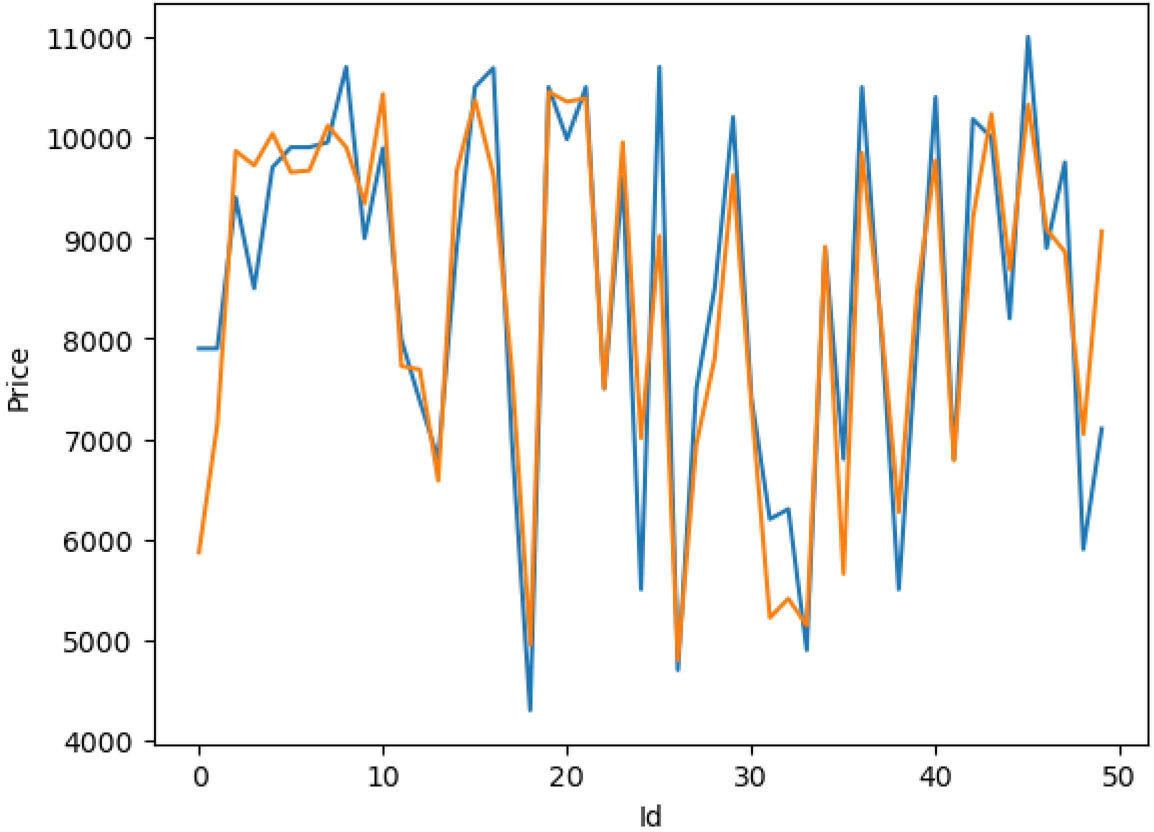
	index	Price	Predicted	Id
0	481	7900	5869.741155	0
1	76	7900	7149.563327	1
2	1502	9400	9862.785355	2
3	669	8500	9719.283532	3
4	1409	9700	10035.895686	4
5	1414	9900	9650.311090	5
6	1089	9900	9669.183317	6
7	1507	9950	10115.128380	7
8	970	10700	9900.241944	8
9	1198	8999	9347.080772	9
10	1088	9890	10431.237961	10
11	576	7990	7725.756431	11
12	965	7380	7691.089846	12
13	1488	6800	6583.674680	13
14	1432	8900	9659.240069	14
15	380	10500	10370.231518	15
16	754	10690	9620.427488	16
17	30	6990	7689.189244	17
18	49	4300	4954.595074	18
19	240	10500	10452.262871	19
20	344	9980	10353.107796	20

	index	Price	Predicted	Id
21	354	10500	10388.635632	21
22	124	7500	7503.302407	22
23	383	9600	9948.970588	23
24	1389	5500	7009.047336	24

In [192]:
```
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[192]: []



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: