In [1]:

```
import pandas as pd
data=pd.read_csv("/home/placement/Desktop/BhanuSiva4K8/fiat500.csv")
```

In [2]:

```
import warnings
warnings.filterwarnings("ignore")
```

In [39]:

```
data.describe()
```

Out[39]:

	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 [3]:

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

```
In [4]: data
```

```
Out[4]:
        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 [5]: data=pd.get_dummies(data)
```

In [6]:
```
data
```

Out[6]:

	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 [7]:
```
#predicted value we removed from dataframe
y=data['price']
x=data.drop('price',axis=1)
```

In [8]:
```
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)
```

linear regression

In [9]:
```
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[9]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [10]:
```
ypred=reg.predict(x_test)
```

In [11]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[11]:
```
0.8415526986865394
```

In [12]:
```
from sklearn.metrics import mean_squared_error #calculating MSE
mean_squared_error(ypred,y_test)
```

Out[12]:
```
581887.727391353
```

In [13]:

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

Out[13]:

	index	Price	Predicted
0	481	7900	5867.650338
1	76	7900	7133.701423
2	1502	9400	9866.357762
3	669	8500	9723.288745
4	1409	9700	10039.591012
5	1414	9900	9654.075826
6	1089	9900	9673.145630
7	1507	9950	10118.707281
8	970	10700	9903.859527
9	1198	8999	9351.558284
10	1088	9890	10434.349636
11	576	7990	7732.262557
12	965	7380	7698.672401
13	1488	6800	6565.952404
14	1432	8900	9662.901035

In [14]:

```
Results['DIFF']=Results.apply(lambda row:row.Price-row.Predicted,axis=1)
```

In [15]:

```
Results
```

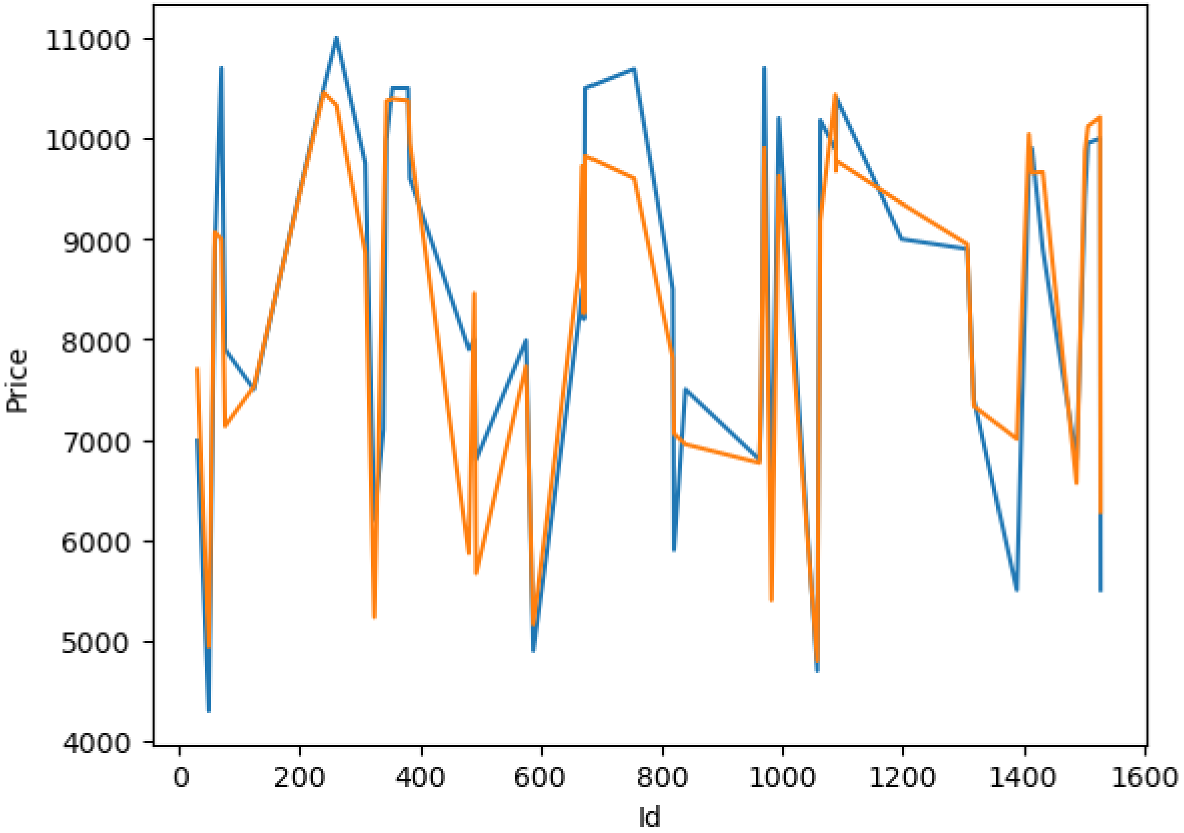
Out[15]:

	Price	Predicted	Id	DIFF
481	7900	5867.650338	481	2032.349662
76	7900	7133.701423	76	766.298577
1502	9400	9866.357762	1502	-466.357762
669	8500	9723.288745	669	-1223.288745
1409	9700	10039.591012	1409	-339.591012
...	...	...	...	...
291	10900	10032.665135	291	867.334865
596	5699	6281.536277	596	-582.536277
1489	9500	9986.327508	1489	-486.327508
1436	6990	8381.517020	1436	-1391.517020
575	10900	10371.142553	575	528.857447

508 rows × 4 columns

In [16]:
```
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[16]:
```
[]
```



Ridge regression

In [17]:

```
#ridge regression
```

In [18]:

```
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[18]:

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [19]:

```
ridge_regressor.best_params_
```

Out[19]:

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

In [20]:

```
#x_train=[2]
```

In [21]:

```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

```
In [22]: y_pred_ridge
```

```
Out[22]: array([ 5869.74115507,  7149.56332694,  9862.78535486,  9719.28353248,
               10035.89568574,  9650.31109035,  9669.18331738, 10115.12838027,
                9900.24194354,  9347.08077182, 10431.23796139,  7725.75643127,
                7691.08984564,  6583.67468036,  9659.24006885, 10370.23151754,
                9620.42748841,  7689.18924428,  4954.59507446, 10452.26287068,
               10353.10779648, 10388.63563168,  7503.30240667,  9948.97058812,
                7009.04733578,  9020.73569412,  4798.12691579,  6944.67171049,
                7803.34446535,  9619.98788702,  7326.43443918,  5218.4077102 ,
                5408.53918256,  5141.35782797,  8914.90902841,  5656.63497772,
                9843.54231891,  8236.55007384,  6271.31566471,  8476.67006596,
                9770.02244191,  6784.29000107,  9203.55210535, 10231.79726073,
                8688.72507822, 10325.35487633,  9089.06645878,  8862.41881997,
                7048.7619628 ,  9068.9099975 ,  9409.53675932, 10290.6563444 ,
               10068.75380626,  6766.38650916,  9782.42178795,  9375.38267977,
                9528.4069177 , 10440.0567266 ,  9791.53263494,  7216.09577125,
               10104.686048  ,  7001.39195702,  9850.13133436,  7139.90750908,
                6408.14610807,  9993.32275333,  9777.34727934,  8535.02652876,
                8450.89417219,  6490.79570767,  7761.36847462,  6833.92199079,
                8342.12534099, 10436.01203789,  7349.55597282,  8557.12693543,
                9817.00392891, 10032.09281089,  7363.58036249,  9418.31857132,
```

```
In [23]: from sklearn.metrics import mean_squared_error
         Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
         Ridge_Error
```

```
Out[23]: 579521.7970897449
```

```
In [24]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred_ridge)
```

```
Out[24]: 0.8421969385523054
```

```
In [25]: Results=pd.DataFrame(columns=['Actual','predicted'])
         Results['Actual']=y_test
         Results['predicted']=y_pred_ridge
```

In [26]:

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

Out[26]:

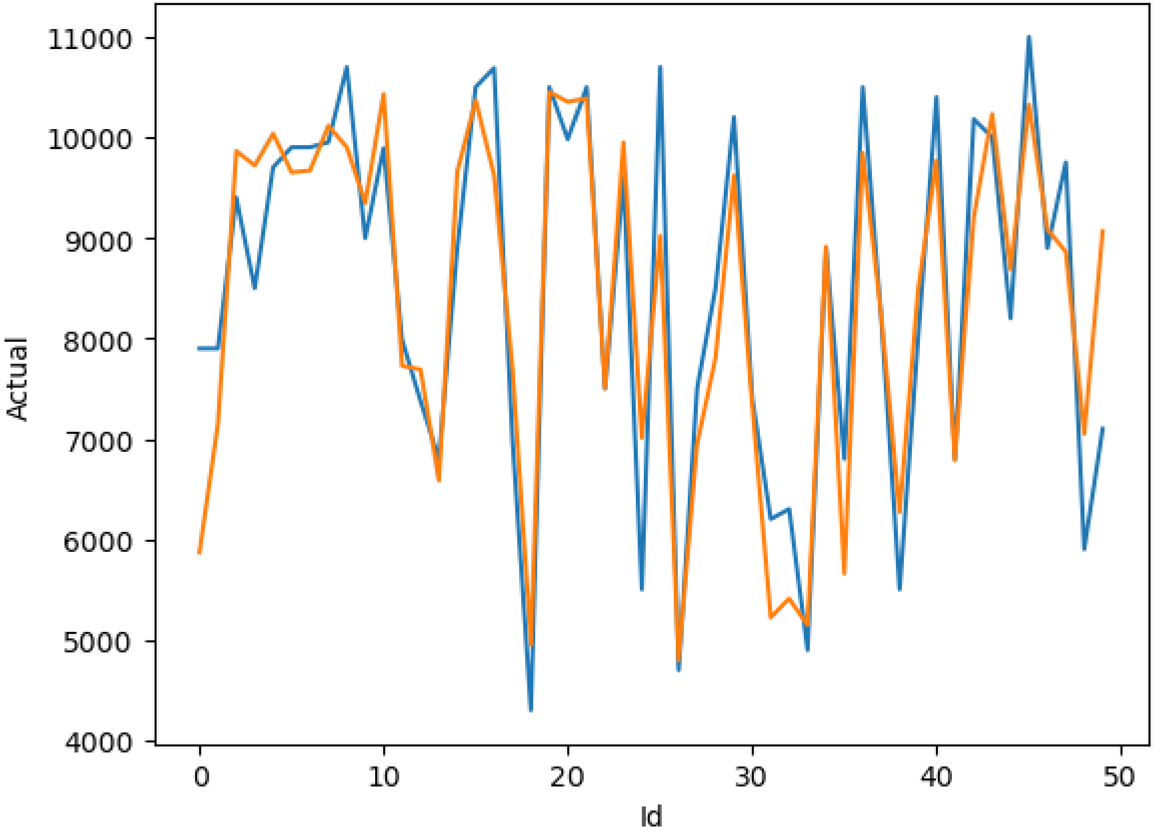
```
   index  Actual     predicted  Id
0    481    7900   5869.741155   0
1     76    7900   7149.563327   1
2   1502    9400   9862.785355   2
3    669    8500   9719.283532   3
4   1409    9700  10035.895686   4
5   1414    9900   9650.311090   5
6   1089    9900   9669.183317   6
7   1507    9950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

In [27]:

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

```
In [28]: sns.lineplot(x='Id',y='Actual',data=Results.head(50))
         sns.lineplot(x='Id',y='predicted',data=Results.head(50))
         plt.plot()

Out[28]: []
```



Elastic Net Model

In [29]:

```
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[29]:

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:

```
elastic_regressor.best_params_
```

Out[30]:

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

In [31]:

```
#elastic regression
```

In [32]:

```
#x_train=[2]
```

In [33]:

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

In [34]:

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

Out[34]:

```
0.841688021120299
```

In [35]:
```
from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[35]:
```
581390.7642825295
```

In [36]:
```
Results=pd.DataFrame(columns=['Actual','predicted'])
Results['Actual']=y_test
Results['predicted']=y_pred_elastic
#Result['km']=x_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

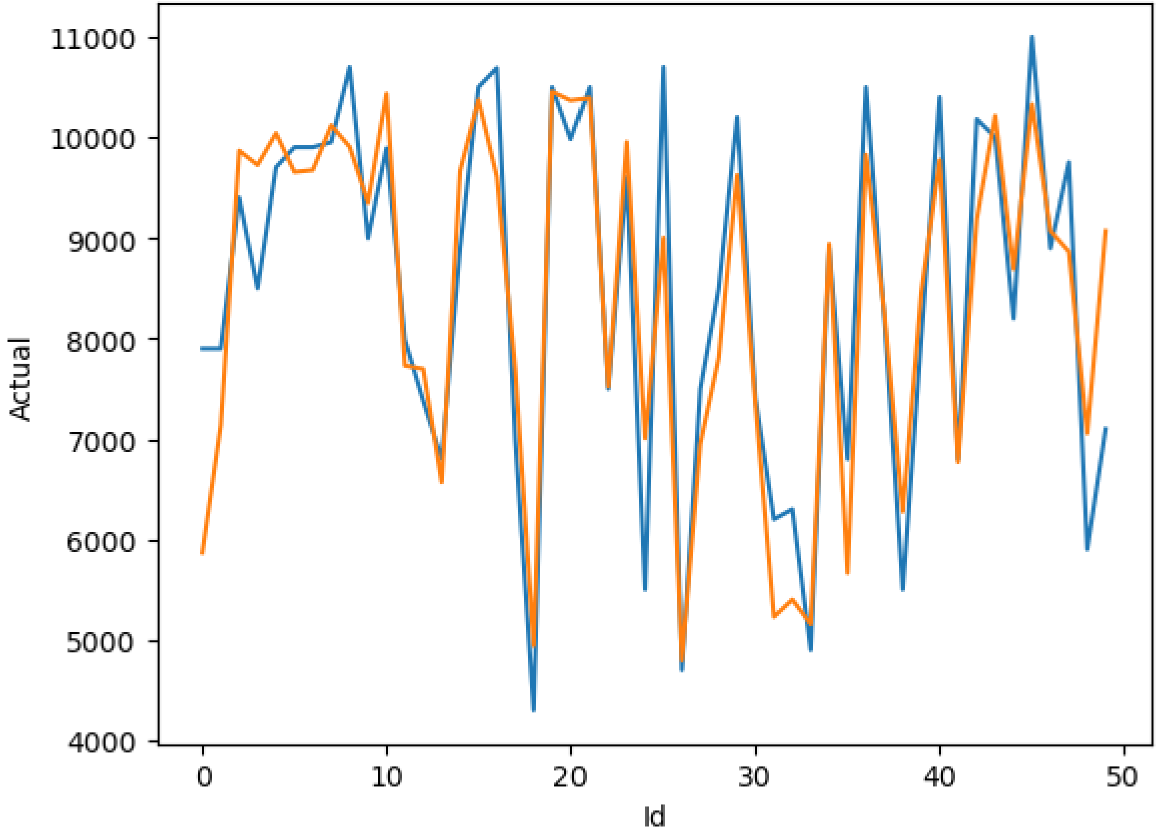
Out[36]:

	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 [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [38]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='predicted',data=Results.head(50))
plt.plot()
```

Out[38]:
```
[]
```



In [ ]: