In [241]:
```
import pandas as pd
```

In [242]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [243]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [244]:
```
data.describe()
```

Out[244]:

	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 [245]:
```
data1=data.drop(['ID','lat','lon'],axis=1)
```

In [246]:
```
data1
```

Out[246]:
```
       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 [247]:
```
data=data.loc[(data.model=='lounge')]
```

In [248]:
```
data=pd.get_dummies(data)
```

In [249]:
```
data
```

Out[249]:
```
        ID  engine_power  age_in_days      km  previous_owners        lat        lon  price  model_lounge
   0     1            51          882   25000                1  44.907242   8.611560   8900             1
   3     4            51         2739  160000                1  40.633171  17.634609   6000             1
   6     7            51          731   11600                1  44.907242   8.611560  10750             1
   7     8            51         1521   49076                1  41.903221  12.495650   9190             1
  11    12            51          366   17500                1  45.069679   7.704920  10990             1
 ...   ...           ...          ...     ...              ...        ...        ...    ...           ...
1528  1529            51         2861  126000                1  43.841980  10.515310   5500             1
1529  1530            51          731   22551                1  38.122070  13.361120   9900             1
1530  1531            51          670   29000                1  45.764648   8.994500  10800             1
1534  1535            74         3835  112000                1  45.845692   8.666870   4600             1
1536  1537            51         2557   80750                1  45.000702   7.682270   5990             1

1094 rows × 9 columns
```

In [250]:
```
data.shape     #data['moddel']=data['model'].map(('longue':1,'pop':2))
```

Out[250]:
```
(1094, 9)
```

In [251]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1094 entries, 0 to 1536
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1094 non-null   int64  
 1   engine_power     1094 non-null   int64  
 2   age_in_days      1094 non-null   int64  
 3   km               1094 non-null   int64  
 4   previous_owners  1094 non-null   int64  
 5   lat              1094 non-null   float64
 6   lon              1094 non-null   float64
 7   price            1094 non-null   int64  
 8   model_lounge     1094 non-null   uint8  
dtypes: float64(2), int64(6), uint8(1)
memory usage: 78.0 KB
```

In [252]:

```
y=data['price']            #which paramter we want to add we can.....
x=data.drop('price',axis=1)
```

In [253]:

```
y
```

Out[253]:

```
0        8900
3        6000
6       10750
7        9190
11      10990
        ...  
1528     5500
1529     9900
1530    10800
1534     4600
1536     5990
Name: price, Length: 1094, dtype: int64
```

In [254]:

```
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 [255]:

```
x_test.head()
```

Out[255]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	model_lounge
676	677	51	762	18609	1	41.572239	13.33369	1
215	216	51	701	25000	1	44.988739	9.01050	1
146	147	51	4018	152900	1	43.067532	12.55155	1
1319	1320	51	731	20025	1	41.689281	13.25494	1
1041	1042	51	640	38231	1	41.107880	14.20881	1

In [256]:

```
x_train.head()
```

Out[256]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	model_lounge
441	442	51	762	36448	1	45.571220	9.15914	1
701	702	51	701	27100	1	41.903221	12.49565	1
695	696	51	3197	51083	1	45.571220	9.15914	1
1415	1416	51	670	33000	1	42.287029	12.40754	1
404	405	51	456	14000	1	40.840141	14.25226	1

In [257]:

```
y_test.head()
```

Out[257]:

```
676     10250
215      9790
146      5500
1319     9900
1041     8900
Name: price, dtype: int64
```

In [258]:

```
y_train.head()
```

Out[258]:

```
441      8980
701     10300
695      5880
1415    10490
404      9499
Name: price, dtype: int64
```

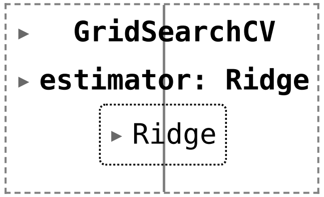
In [259]:

```
# ridge regrssion
```

In [260]:

```
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[260]:



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

In [261]:

```
ypred=ridge_regressor.predict(x_test)
```

```
In [262]: ypred
```

```
Out[262]: array([ 9912.60175361, 10141.74849333,  4775.23552146,  9870.92696571,
                  9630.41788453,  8697.09201357, 10265.82288414, 10293.85186684,
                  8614.34973762,  5749.67356711, 10671.67602325,  6488.02221144,
                  9752.99829873, 10520.17597908,  8086.90253749,  9498.92882567,
                  7801.23188858,  9783.915695  , 10522.29792692,  9641.86872663,
                 10614.24629923, 10613.19901763,  9892.38749947,  6510.06240197,
                 10549.52425763, 10625.76078907, 10568.39331427,  7946.89947635,
                  5931.34546217,  4659.2196909 , 10428.89187791,  5655.72815127,
                  9478.32068501, 10329.98145039,  7131.2852707 ,  7921.50560262,
                  7874.80635726,  5954.04367445,  9722.42751047,  9680.86485103,
                 10527.15377696,  9474.90517944, 10205.46024252,  6549.58459072,
                  6994.35871214,  9991.85800581, 10247.34928322,  8277.34560789,
                 10300.61976656, 10078.48363687, 10268.33050716,  9823.77891284,
                  9669.33394656,  9513.50322923,  9152.34918875,  9631.89820083,
                  6653.57742077,  9680.19991056,  9984.99476556,  5648.20897225,
                 10341.67956632, 10540.84441014,  9555.12631439,  6825.22781604,
                 10486.94645618, 10510.87237214,  9280.22784667,  9695.90865183,
                 10300.86096344, 10620.75242063,  7255.08871011,  9512.12507442,
                  9609.32308614,  7112.79851998, 10034.0749881 , 10330.98892175,
                  8548.73769446,  9520.16121454,  9946.6185962 , 10135.88071505,
                 10184.38248658,  6506.0325387 , 10522.28394638,  9889.0361183 ,
                  9692.79785416,  6645.09656843,  7830.50421028,  9905.63015012,
                  9577.17218464, 10582.05089567,  6097.15652897,  9714.66288548,
                  8823.94189014, 10177.17443641, 10542.43749844,  7878.55575401,
                  8982.20194888, 10550.72596946,  7089.74287761,  6771.15834746,
                  5780.82200321,  6442.12029954,  9580.92651411,  6276.86875321,
                  9929.59359002,  9679.28936525, 10535.03640665,  5771.91010315,
                  9608.4971782 ,  7176.14803032,  9525.84417673,  9786.76124829,
                 10590.77268612, 10590.43852943,  5621.28001026,  4969.18369174,
                  9837.01957868,  9839.16975778,  5070.94098034, 10540.48246758,
                 10039.03821544,  9743.55236996, 10307.24454309,  4765.01281868,
                  5409.7256093 ,  9643.2735831 , 10542.08833354, 10133.68993901,
                  8027.5823784 ,  9647.81039882,  9922.44925637,  9856.02030419,
                 10079.86899098,  9527.4017113 , 10323.2834034 ,  9269.698239  ,
                  8174.69678444, 10616.58083442,  8743.66370719,  7209.22489424,
                  7847.26975825,  8747.91121417,  9781.53808943, 10260.4486203 ,
                  7925.32703754, 10187.50685027,  4959.12317166,  8893.64244815,
                  9722.39120759, 10250.28523132, 10250.36206792,  5912.56256295,
                  6807.58831598,  9696.42747582,  9567.72838167,  5206.84300194,
                 10634.3715292 , 10556.43217805,  5999.05156088,  8131.04680241,
                 10633.13053344, 10603.33150892,  9375.79323009,  8253.42029703,
                  9621.99222439, 10146.51674371, 10357.83931499,  9967.00754951,
                  8771.07396787,  9620.54745456,  9977.38184751,  7777.47051447,
                 10520.11870767, 10240.92028123,  9721.01473511, 10188.15040931,
                 10324.27375793, 10349.61509189, 10541.09807142,  8741.26236454,
                 10243.01289328,  9887.14565488, 10065.29895276, 10132.38294069,
                  9674.31474484,  8885.27709328, 10409.16272209,  6800.49736966,
                  9117.14220826,  8864.28804571,  4840.78783722,  6300.16171102,
                  6953.75162041, 10584.08252879, 10614.11269082, 10553.96978192,
                  5804.94025697, 10221.87438241,  7326.66636302, 10325.42324143,
                  7408.64869326, 10194.44686068, 10049.03849678, 10560.98131597,
                  8561.3677542 ,  7002.24366144,  9735.12211999,  5746.03243235,
                 10133.21380035,  9154.14421372,  8101.18661858,  8973.15464568,
                  6380.90009119, 10386.97446276,  9546.7269945 ,  9704.79454985,
                  7370.37427528,  9203.56730794, 10350.60895518,  9298.59824267,
                  9132.59958648, 10216.29186327,  9704.4407033 ,  7725.46131136,
                 10287.46667159,  9609.43361413, 10214.31349489,  9879.91785657,
                  7406.28283552,  9403.64495102,  7031.26752406, 10306.11698001,
                  5029.80565798,  9548.15539101,  9534.49112983,  8955.52632748,
                  9337.90818294, 10026.51728349,  6718.22675615,  9679.48824761,
                  8046.72553537,  8767.59579597, 10096.65316184,  9775.89475575,
                 10089.23188645,  9609.76334055, 10602.57044078,  9697.14354053,
                  9745.26657969,  6596.4263745 ,  7553.46169797, 10246.65892842,
                  9855.94030922,  6156.98155366,  5277.51949478, 10104.49039084,
                  8660.57028716, 10332.35979763,  6195.48775038,  9494.48680977,
                 10410.11427034,  9528.85284008,  7712.5237104 ,  9668.73233268,
                  9992.71217651,  7077.38641746,  8069.24557391,  9703.41609333,
                 10127.18251058,  8045.84754453, 10523.18229626,  9518.60318396,
                 10343.84782629,  5348.69279347,  7461.40351053,  9612.5431617 ,
                  5438.37441051, 10162.86581681,  8982.87426257,  7854.07802564,
                  9618.76245637, 10111.99943317,  6391.21095094,  9613.57830029,
                 10189.985113  ,  9799.75936831,  9687.10794281,  9659.78629905,
                 10162.29208696, 10064.49474248, 10086.16226562, 10539.35304828,
                 10233.25044593,  9061.65656757,  9617.05943216,  8137.16294265,
                  9645.07703767,  7741.6714318 ,  5662.32693722, 10512.54814525,
                 10030.40533701,  7118.51975807,  6975.78482232, 10486.23349272,
                 10524.03417441,  9937.38057631, 10075.86556192,  9252.42552778,
                 10467.73081026,  7838.47608819, 10196.52378389,  7728.72341896,
                  5505.94851073,  9635.83851457, 10297.36829864,  9748.29752091,
                  4011.27222267,  9795.73101359, 10525.0830173 ,  7640.3285934 ,
                  7336.43417344, 10200.95543901,  9152.59811595,  9834.11005597,
                  5818.36746835,  9714.57400974, 10241.19807176, 10422.5660614 ,
                 10209.46715867,  5579.74594179,  5898.87336357,  7416.19197505,
                  9719.87271397,  7075.23773519,  6931.16474141, 10401.71299323,
                  6453.58999536,  8715.51600214, 10199.91621215, 10516.05238422,
                  9831.90876508, 10135.61019646, 10333.0173839 , 10260.98865218,
                  6011.69111458,  5220.39729696, 10384.7243347 , 10460.61757356,
                  5937.8611916 ,  5903.89776229,  8830.14162146,  9727.70650583,
                 10714.09534551,  8716.28343859, 10654.13648518, 10545.90655668,
                  6969.671378  ,  5211.67195028, 10623.12460075,  8958.70728017,
                 10522.2498154 ,  9723.90961557])
```

In [263]:

```
ridge_regressor.best_params_
```

Out[263]:

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

In [264]:

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

In [265]:

```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[265]:

```
529111.0455362241
```

In [266]:

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)  #ytest=actual price,ypred=predicted price
```

Out[266]:

```
0.8343797517106646
```

In [269]:

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

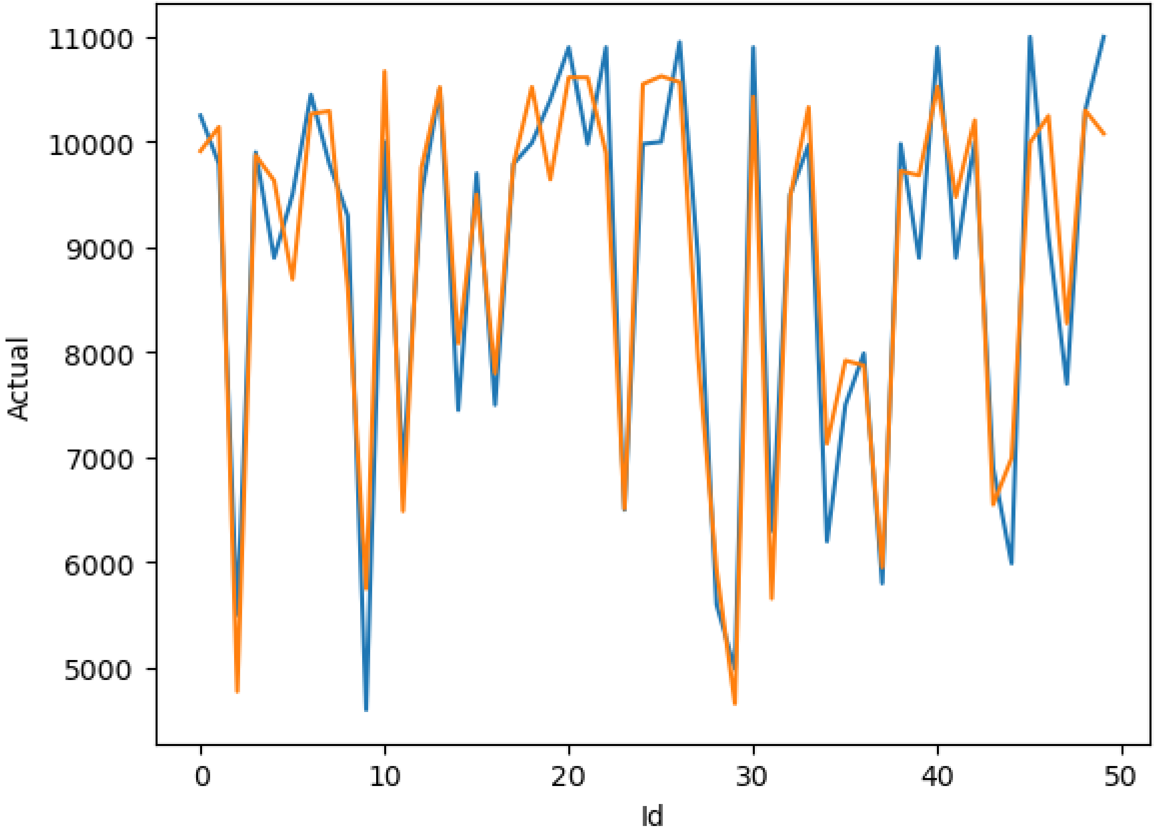
Out[269]:

	index	Actual	Predicted	Id
0	676	10250	9912.601754	0
1	215	9790	10141.748493	1
2	146	5500	4775.235521	2
3	1319	9900	9870.926966	3
4	1041	8900	9630.417885	4
...	...	...	...	...
357	757	6000	5211.671950	357
358	167	10950	10623.124601	358
359	156	8000	8958.707280	359
360	1145	10700	10522.249815	360
361	1393	9400	9723.909616	361

362 rows × 4 columns

```
In [270]: 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.plot()
```

```
Out[270]: []
```



In [ ]: