

```
In [137]: import pandas as pd
```

```
In [138]: import warnings  
warnings.filterwarnings("ignore")
```

```
In [139]: data=pd.read_csv("/home/placement/Desktop/EEE(222)/fiat500.csv")
```

```
In [140]: data.describe()
```

Out[140]:

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

```
In [142]: data
```

```
Out[142]:
```

	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 [143]: data1=data.loc[(data.model=='lounge')]
```

```
In [144]: data1=pd.get_dummies(data1)
```

```
In [145]: data1.shape
```

```
Out[145]: (1094, 6)
```

```
In [146]: data1
```

```
Out[146]:
```

	engine_power	age_in_days	km	previous_owners	price	model_lounge
0	51	882	25000	1	8900	1
3	51	2739	160000	1	6000	1
6	51	731	11600	1	10750	1
7	51	1521	49076	1	9190	1
11	51	366	17500	1	10990	1
...
1528	51	2861	126000	1	5500	1
1529	51	731	22551	1	9900	1
1530	51	670	29000	1	10800	1
1534	74	3835	112000	1	4600	1
1536	51	2557	80750	1	5990	1

1094 rows × 6 columns

```
In [147]: y=data1['price']  
x=data1.drop('price',axis=1)
```

In [148]:

y

Out[148]:

```
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 [149]:

x

Out[149]:

	engine_power	age_in_days	km	previous_owners	model_lounge
0	51	882	25000	1	1
3	51	2739	160000	1	1
6	51	731	11600	1	1
7	51	1521	49076	1	1
11	51	366	17500	1	1
...
1528	51	2861	126000	1	1
1529	51	731	22551	1	1
1530	51	670	29000	1	1
1534	74	3835	112000	1	1
1536	51	2557	80750	1	1

1094 rows × 5 columns

```
In [150]: 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 [151]: x_test.head(5)
```

Out[151]:

	engine_power	age_in_days	km	previous_owners	model_lounge
676	51	762	18609	1	1
215	51	701	25000	1	1
146	51	4018	152900	1	1
1319	51	731	20025	1	1
1041	51	640	38231	1	1

```
In [152]: y_test.head(5)
```

Out[152]: 676 10250
215 9790
146 5500
1319 9900
1041 8900
Name: price, dtype: int64

```
In [153]: x_train.head(5)
```

Out[153]:

	engine_power	age_in_days	km	previous_owners	model_lounge
441	51	762	36448	1	1
701	51	701	27100	1	1
695	51	3197	51083	1	1
1415	51	670	33000	1	1
404	51	456	14000	1	1

```
In [154]: y_train.head(5)
```

```
Out[154]: 441      8980  
          701     10300  
          695      5880  
          1415    10490  
          404      9499  
          Name: price, dtype: int64
```

```
In [155]: x_train.shape
```

```
Out[155]: (732, 5)
```

```
In [156]: y_train.shape
```

```
Out[156]: (732,)
```

```
In [157]: x_test.shape
```

```
Out[157]: (362, 5)
```

```
In [158]: y_test.shape
```

```
Out[158]: (362,)
```

```
In [159]: x_train.tail(5)
```

```
Out[159]:
```

	engine_power	age_in_days	km	previous_owners	model_lounge
459	51	397	15628	1	1
654	51	3227	95554	1	1
189	51	1431	81900	1	1
1455	51	701	33942	1	1
1218	51	882	25000	1	1

```
In [160]: x_test.tail(5)
```

```
Out[160]:
```

	engine_power	age_in_days	km	previous_owners	model_lounge
757	51	4018	102841	1	1
167	51	397	15341	1	1
156	51	1858	35304	1	1
1145	51	456	14970	1	1
1393	51	609	32665	2	1

```
In [161]: y_test.tail(5)
```

```
Out[161]: 757      6000
167      10950
156      8000
1145     10700
1393      9400
Name: price, dtype: int64
```

```
In [162]: y_train.tail(5)
```

```
Out[162]: 459      10850
654       5900
189      10000
1455      9400
1218      8900
Name: price, dtype: int64
```

In [163]: x_test

Out[163]:

	engine_power	age_in_days	km	previous_owners	model_lounge
676	51	762	18609	1	1
215	51	701	25000	1	1
146	51	4018	152900	1	1
1319	51	731	20025	1	1
1041	51	640	38231	1	1
...
757	51	4018	102841	1	1
167	51	397	15341	1	1
156	51	1858	35304	1	1
1145	51	456	14970	1	1
1393	51	609	32665	2	1

362 rows × 5 columns


```
In [164]: x_train
```

```
Out[164]:
```

	engine_power	age_in_days	km	previous_owners	model_lounge
441	51	762	36448	1	1
701	51	701	27100	1	1
695	51	3197	51083	1	1
1415	51	670	33000	1	1
404	51	456	14000	1	1
...
459	51	397	15628	1	1
654	51	3227	95554	1	1
189	51	1431	81900	1	1
1455	51	701	33942	1	1
1218	51	882	25000	1	1

732 rows × 5 columns

```
In [165]: y_test
```

```
Out[165]: 676    10250
          215     9790
          146     5500
          1319    9900
          1041     8900
```

```
          ...
          757     6000
          167    10950
          156     8000
          1145    10700
          1393     9400
```

Name: price, Length: 362, dtype: int64

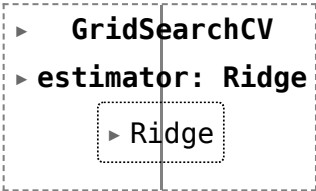
```
In [166]: y_train
```

```
Out[166]: 441      8980
          701     10300
          695      5880
          1415    10490
          404      9499
          ...
          459     10850
          654      5900
          189     10000
          1455     9400
          1218     8900
          Name: price, Length: 732, dtype: int64
```

```
In [167]: #ridge regression
```

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



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

```
In [169]: ridge_regressor.best_params_
```

```
Out[169]: {'alpha': 30}
```

```
In [170]: #x_train=[2]
```

```
In [171]: ridge=Ridge(alpha=30)
          ridge.fit(x_train,y_train)
          y_pred_ridge=ridge.predict(x_test)
```

```
In [172]: y_pred_ridge
```

```
Out[172]: array([[10045.34777889, 9989.17153543, 4769.09960336, 10048.68323752,
 9813.94479825, 8678.14356117, 10173.79792135, 10180.6270078 ,
 9107.31525896, 5625.00740732, 10565.71108835, 6776.12815534,
 9677.36019112, 10348.97135978, 8049.20104733, 9526.33575316,
 7738.85607226, 9973.09944563, 10379.76191917, 9784.95620261,
10390.79428386, 10429.52293694, 9867.32992522, 6316.76795239,
10363.01826786, 10565.71108835, 10385.15644406, 8356.2693706 ,
 6052.94959183, 4562.66804027, 10340.47145405, 5796.55307957,
 9687.69883182, 10386.93279686, 7018.31868443, 7936.55917599,
 7765.92126381, 6169.45640953, 9811.27845178, 9882.52937837,
10312.76262569, 9691.63232633, 10565.71108835, 6585.82855773,
 6916.6311432 , 10347.90965216, 10136.14357831, 8266.05175267,
10133.53282186, 10426.05302378, 10264.14549009, 9629.21583316,
 9977.36553225, 9716.74149368, 9353.11972737, 9573.46229983,
 6761.6689103 , 9804.79795157, 9932.37164515, 5625.07326046,
10146.17519266, 10332.82578954, 9734.59814219, 6678.28902489,
10293.21149128, 10312.42707921, 9427.86530055, 9815.46093328,
10394.02774477, 10436.31090369, 7098.97365343, 9677.370361 ,
 9828.47077394, 7021.16294159, 9930.12732016, 10196.92829788,
 8386.74648114, 9540.25435824, 9765.3639485 , 10368.1517171 ,
10082.88941809, 6357.42683864, 10430.78719546, 10093.60179309,
 9516.80780142, 7122.64740135, 7719.55327264, 9832.05621279,
 9757.42338865, 10436.88527011, 6004.71929229, 9921.3339132 ,
 8877.95292358, 10041.38666426, 10462.68481494, 7733.76691433,
 8850.71117831, 10421.05777646, 6942.97382254, 6912.87326813,
 6066.80915414, 6315.26734883, 9774.79076844, 6398.14508382,
 9806.68987526, 9801.49099341, 10501.27761938, 5604.99606646,
 9794.78184597, 6999.20995646, 9467.39307862, 9706.5227695 ,
10378.72288778, 10455.23018537, 6012.0099509 , 5118.30421508,
 9752.19360218, 9931.77067455, 5449.00003099, 10334.91439469,
 9977.36553225, 9855.40550683, 10381.30992956, 4889.95648612,
 5226.68791203, 9813.4154118 , 10324.79575513, 10315.57646644,
 7935.99268535, 9857.80550646, 9808.2552455 , 9853.29949663,
 9977.12724572, 9697.81111657, 10462.92321435, 9239.75790604,
 8056.55468638, 10437.56406678, 8673.59658098, 7098.99055677,
 7816.00486189, 8942.22604215, 10187.4782851 , 10424.94541065,
 8172.19745151, 10070.42740734, 5450.70371602, 9057.23399734,
 9940.60129574, 10141.83209263, 10029.2031342 , 5882.88399169,
 6988.85761289, 9792.97361779, 9740.70731227, 5189.58908619,
```

```
10448.51710998, 10350.05678205, 5930.14249834, 7973.31320996,
10408.72918181, 10398.45493529, 9622.06348934, 7981.39552103,
9831.98659262, 10339.22453573, 10450.74495548, 9894.02540083,
8829.9851558 , 9632.58136126, 10108.59863049, 7997.00644995,
10378.37478692, 10250.24717957, 9665.62091858, 9985.97917741,
10382.21603023, 10409.01979636, 10371.00614414, 8581.84728281,
10394.31253507, 9657.1968326 , 9854.59232047, 10053.40822971,
9611.32203526, 8756.17468877, 10261.99836019, 6636.54831182,
9320.38823484, 8937.41241054, 4584.0579125 , 6496.04532397,
6874.60996914, 10445.08831652, 10417.86155817, 10465.59728135,
5908.85959612, 10364.13115218, 7440.14661653, 10462.92321435,
7364.84878898, 10073.35249298, 9996.90323477, 10382.04539989,
8468.286263 , 7128.4688464 , 10157.57549578, 5690.55405714,
10473.87098636, 9394.7588997 , 7965.99904423, 8857.22351744,
6567.36918077, 10264.14549009, 9723.18859473, 9650.11302532,
7558.9528078 , 9092.04408323, 10195.32072431, 9417.92066621,
9023.56448358, 10203.89031505, 9857.70928211, 7869.82213501,
10094.23854452, 10016.87839982, 10393.56846387, 9939.33366233,
7473.59251736, 9548.45848828, 6931.35602476, 10128.41187897,
5476.07841754, 9482.94932674, 10004.71809905, 8840.35803082,
9183.58600823, 10193.44728929, 6693.42378212, 9891.64090994,
7941.18281108, 8893.89594292, 9977.36553225, 9646.02284023,
10115.36826665, 9977.36553225, 10416.41693961, 9866.37068388,
9612.95325864, 6763.86620095, 7758.56279091, 10511.3377342 ,
10110.03601389, 5985.71210505, 5379.55474479, 10086.77270664,
8590.88930427, 10392.53722069, 6098.57359345, 9709.1703 ,
10325.18497569, 9715.54054572, 7572.96558285, 9765.3639485 ,
9813.51896171, 7228.99093841, 8007.83781593, 9904.86874258,
10012.0567824 , 7912.56700171, 10375.39325339, 9673.69685851,
10398.45493529, 5460.03755965, 7280.5349007 , 9810.03252385,
5418.53480582, 10380.92701862, 8877.95292358, 8041.02830728,
9790.60757038, 10459.71922749, 6263.41225143, 9554.10680845,
10436.61206061, 9968.01613467, 9853.07133478, 9880.78402341,
10046.86686828, 9961.05411749, 9977.36553225, 10370.13693042,
10161.11057059, 9192.46590982, 9830.4375438 , 8012.10493814,
9453.45887438, 7877.55383435, 5587.00369094, 10386.10262599,
9878.41601203, 7003.92764864, 7173.72195805, 10229.36112804,
10403.26253937, 9734.78855924, 10021.13736416, 9349.69093391,
10387.84313028, 7733.76691433, 10056.64892295, 7656.3382783 ,
5702.23720624, 9811.21606677, 10415.02980736, 9872.39093245,
3901.76632267, 9789.01354159, 10420.22864401, 7511.00890116,
7204.44567391, 10074.56980756, 9103.99857109, 9688.23834295,
```

```
5656.79259391, 9811.27845178, 10076.19675014, 10339.10901004,  
10125.897234 , 5479.82528759, 5904.27386392, 7576.88220289,  
9765.75023434, 6890.55397887, 6691.60769781, 10461.98334203,  
6319.35371886, 8877.95292358, 10196.29154645, 10317.76515621,  
9935.62007856, 10042.02341569, 10431.48866837, 10403.11317719,  
5859.48252489, 5133.29631187, 10447.51975135, 10307.64700202,  
5794.71820485, 5855.33690786, 8722.08988368, 10059.34866858,  
10732.79990752, 8834.7001814 , 10565.71108835, 10324.31472354,  
6791.95158544, 5640.37864803, 10431.68116227, 8765.50686495,  
10384.88427298, 9929.72168494])
```

```
In [173]: from sklearn.metrics import mean_squared_error  
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)  
Ridge_Error
```

```
Out[173]: 519771.8129989745
```

```
In [174]: from sklearn.metrics import r2_score  
r2_score(y_test,y_pred_ridge)
```

```
Out[174]: 0.8373030813683994
```

```
In [176]: 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)
```

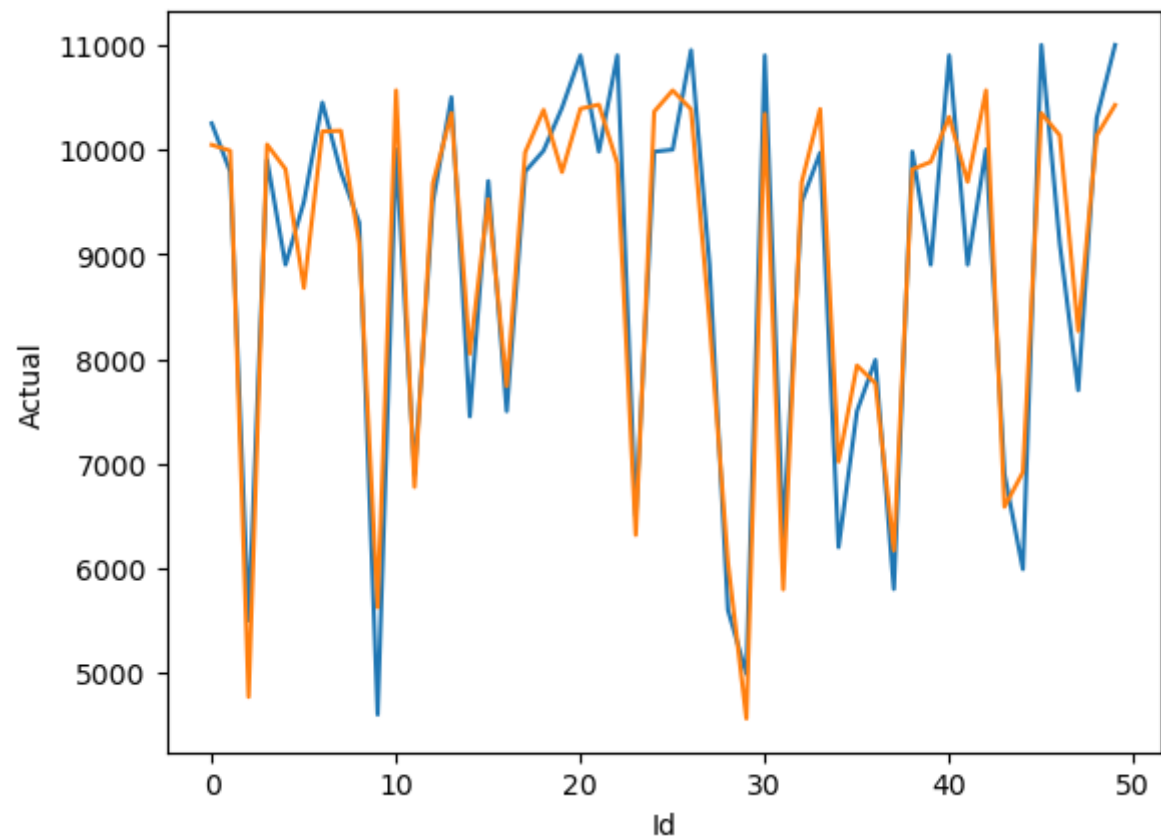
Out[176]:

	index	Actual	predicted	Id
0	676	10250	10045.347779	0
1	215	9790	9989.171535	1
2	146	5500	4769.099603	2
3	1319	9900	10048.683238	3
4	1041	8900	9813.944798	4
5	1425	9500	8678.143561	5
6	409	10450	10173.797921	6
7	617	9790	10180.627008	7
8	1526	9300	9107.315259	8
9	1010	4600	5625.007407	9

```
In [178]: import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [179]: sns.lineplot(x='Id',y='Actual',data=Results.head(50))  
sns.lineplot(x='Id',y='predicted',data=Results.head(50))  
plt.plot()
```

Out[179]: []



In []: