In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [3]:
```
data.describe()
```

Out[3]:

	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 [4]:
```
data.head()
```

Out[4]:

	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 [5]:
```
data1=data.loc[(data.previous_owners)==1]
```

In [6]:
```
data1
```

Out[6]:

	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

1389 rows × 9 columns

In [7]:
```
data2=data1.drop(['ID','lat','lon'],axis=1)
```

In [8]:
```
data2
```

Out[8]:

	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

1389 rows × 6 columns

In [9]:
```
data2=pd.get_dummies(data2)
```

```
In [10]: data2
```

```
Out[10]:
      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

1389 rows × 8 columns
```

```
In [11]: y=data2['price']
         x=data2.drop('price',axis=1)
```

In [12]:
```
y
```

Out[12]:
```
0        8900
1        8800
2        4200
3        6000
4        5700
         ... 
1533     5200
1534     4600
1535     7500
1536     5990
1537     7900
Name: price, Length: 1389, dtype: int64
```

In [13]:
```
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 [14]:
```
x_test.head()
```

Out[14]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
625	51	3347	148000	1	1	0	0
187	51	4322	117000	1	1	0	0
279	51	4322	120000	1	0	1	0
734	51	974	12500	1	0	1	0
315	51	1096	37000	1	1	0	0

In [15]:
```
y_test.head()
```

Out[15]:
```
625     5400
187     5399
279     4900
734    10500
315     9300
Name: price, dtype: int64
```

```
In [16]:  #LinearRegression
          from sklearn.linear_model import LinearRegression
          reg=LinearRegression()
          reg.fit(x_train,y_train)

Out[16]: LinearRegression()
```

```
In [17]: ypred=reg.predict(x_test)
```

```
In [18]: ypred

Out[18]: array([ 5481.93168764,  5127.11081209,  4798.43164854,  9659.36578585,
                 9409.4127446 , 10351.98379749,  9802.72406141,  8334.75329195,
                 5913.57169572, 10150.04762334,  5643.36202062,  7780.90416594,
                 9721.15872463,  4456.3882388 ,  6541.53947176,  9829.09275112,
                 7574.52796156,  5909.39873877, 10416.87928247,  7409.77542821,
                 8693.13864599,  8182.36608361,  9441.1300824 , 10383.66774161,
                 9857.9433171 , 10388.58335816,  9818.87050889,  7023.92041959,
                 9335.62476174, 10173.88293864,  5551.06753428,  9769.38528629,
                 4609.76045054,  9962.4794893 ,  9789.3539293 ,  8904.50209071,
                 3336.10690574, 10067.44590413,  8607.43409685,  7682.12076521,
                10206.23086655, 10451.29193617, 10428.25147613,  9711.27231338,
                 9296.17132987,  7217.0720428 , 10459.74879956,  9083.60035288,
                10416.67497977,  8567.06083756, 10390.98325814,  7953.60968003,
                 5590.45997234, 10404.33169149,  5658.96046682,  8904.50209071,
                 9962.4794893 ,  5204.32975664,  8381.41911545,  6642.92293048,
                 6236.53789235,  4815.11945754, 10356.87473279,  7963.88315168,
                 5015.51747675,  9896.61284815,  8728.78349613,  5415.22108385,
                 9921.17107046,  7314.69366999, 10088.79553655,  8210.01253214,
                10343.75594017, 10399.71785545,  9720.01037852,  9579.33859859,
```

```
In [19]: from sklearn.metrics import r2_score
         r2_score(y_test,ypred)

Out[19]: 0.8601937431943694
```

In [20]:
```
from sklearn.metrics import mean_squared_error      #calculating MSE
mean_squared_error(ypred,y_test)
```

Out[20]:
```
515432.9010723155
```

In [21]:
```
n=581887.727391353
print(n**(1/2))
```

```
762.8156575420782
```

In [22]:
```
Results=pd.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=ypred
Results.head(15)
```

Out[22]:

	price	predicted
625	5400	5481.931688
187	5399	5127.110812
279	4900	4798.431649
734	10500	9659.365786
315	9300	9409.412745
652	10850	10351.983797
1472	9500	9802.724061
619	7999	8334.753292
992	6300	5913.571696
1154	10000	10150.047623
757	6000	5643.362021
1299	8500	7780.904166
400	8580	9721.158725
314	4600	4456.388239
72	7400	6541.539472

In [23]:
```
Results['diff']=Results.apply(lambda row: row.price - row.predicted,axis=1)
```

In [24]:
```
Results
```

Out[24]:

	price	predicted	diff
625	5400	5481.931688	-81.931688
187	5399	5127.110812	271.889188
279	4900	4798.431649	101.568351
734	10500	9659.365786	840.634214
315	9300	9409.412745	-109.412745
...	...	...	...
115	10650	10397.402425	252.597575
370	9900	10231.829592	-331.829592
1179	5900	6764.023619	-864.023619
93	10050	10378.419299	-328.419299
147	9900	10070.703624	-170.703624

459 rows × 3 columns

In [25]:
```
#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[25]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [26]:
```
ridge_regressor.best_params_
```

Out[26]:
```
{'alpha': 20}
```

In [27]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [28]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[28]:
```
515419.96214274364
```

In [29]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[29]:
```
0.8601972527555688
```

In [30]:

```
Results=pd.DataFrame(columns=['actual','predicted'])
Results['actual']=y_test
Results['predicted']=y_pred_ridge
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(15)
```

Out[30]:

```
    index  actual     predicted  Id
 0    625    5400   5480.612378   0
 1    187    5399   5126.772562   1
 2    279    4900   4823.164641   2
 3    734   10500   9679.384113   3
 4    315    9300   9404.679979   4
 5    652   10850  10346.266387   5
 6   1472    9500   9822.477584   6
 7    619    7999   8367.522197   7
 8    992    6300   5912.518318   8
 9   1154   10000  10144.696863   9
10    757    6000   5642.568011  10
11   1299    8500   7777.488816  11
12    400    8580   9716.019608  12
13    314    4600   4466.017542  13
14     72    7400   6540.492059  14
```

In [31]:
```
#Elastic Net
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV
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[31]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

In [32]:
```
elastic_regressor.best_params_
```

Out[32]:
```
{'alpha': 0.01}
```

In [33]:
```
elastic=ElasticNet(alpha=.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]:
```
0.8602162350730707
```

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

Out[35]:
```
515349.978787187
```

In [36]:

```
Results=pd.DataFrame(columns=['actual','predicted'])
Results['actual']=y_test
Results['predicted']=y_pred_elastic
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(15)
```

Out[36]:

	index	actual	predicted	Id
0	625	5400	5482.171479	0
1	187	5399	5127.531740	1
2	279	4900	4803.203231	2
3	734	10500	9662.825235	3
4	315	9300	9408.645424	4
5	652	10850	10350.952605	5
6	1472	9500	9806.127960	6
7	619	7999	8341.142824	7
8	992	6300	5913.786719	8
9	1154	10000	10149.093829	9
10	757	6000	5643.649619	10
11	1299	8500	7780.541311	11
12	400	8580	9720.293317	12
13	314	4600	4459.155236	13
14	72	7400	6541.667411	14

In [ ]:

```
```