In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
import warnings
warnings.filterwarnings('ignore')
```

read a file

In [3]:
```python
sai=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

describe data

In [4]:
```python
sai.describe()
```

Out[4]:

	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 [5]:
```
sai.head()
```

Out[5]:

	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 [6]:
```
sai.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

maping str into int

In [7]:
```
sai['model']=sai['model'].map({'lounge':1,'pop':2,'sport':3})
sai
```

Out[7]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	1	51	882	25000	1	44.907242	8.611560	8900
1	2	2	51	1186	32500	1	45.666359	12.241890	8800
2	3	3	74	4658	142228	1	45.503300	11.417840	4200
3	4	1	51	2739	160000	1	40.633171	17.634609	6000
4	5	2	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	3	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	1	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	2	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	1	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	2	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

eliminating columns

In [8]:
```
sai.drop(['lat','lon','ID'],axis=1)
```
Out[8]:

	model	engine_power	age_in_days	km	previous_owners	price
0	1	51	882	25000	1	8900
1	2	51	1186	32500	1	8800
2	3	74	4658	142228	1	4200
3	1	51	2739	160000	1	6000
4	2	73	3074	106880	1	5700
...	...	...	...	...	...	...
1533	3	51	3712	115280	1	5200
1534	1	74	3835	112000	1	4600
1535	2	51	2223	60457	1	7500
1536	1	51	2557	80750	1	5990
1537	2	51	1766	54276	1	7900

1538 rows × 6 columns

creating dummies

In [9]:
```
pd.get_dummies(sai)
```

Out[9]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	1	51	882	25000	1	44.907242	8.611560	8900
1	2	2	51	1186	32500	1	45.666359	12.241890	8800
2	3	3	74	4658	142228	1	45.503300	11.417840	4200
3	4	1	51	2739	160000	1	40.633171	17.634609	6000
4	5	2	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	3	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	1	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	2	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	1	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	2	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [10]:
```
y=sai['price']
x=sai.drop('price',axis=1)
x
```

Out[10]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	1	51	882	25000	1	44.907242	8.611560
1	2	2	51	1186	32500	1	45.666359	12.241890
2	3	3	74	4658	142228	1	45.503300	11.417840
3	4	1	51	2739	160000	1	40.633171	17.634609
4	5	2	73	3074	106880	1	41.903221	12.495650
...	...	...	...	...	...	...	...	...
1533	1534	3	51	3712	115280	1	45.069679	7.704920
1534	1535	1	74	3835	112000	1	45.845692	8.666870
1535	1536	2	51	2223	60457	1	45.481541	9.413480
1536	1537	1	51	2557	80750	1	45.000702	7.682270
1537	1538	2	51	1766	54276	1	40.323410	17.568270

1538 rows × 8 columns

split Train&test

In [11]:
```
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 [12]:
```
x_test.head(10)
```

Out[12]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
481	482	2	51	3197	120000	2	40.174702	18.167629
76	77	2	62	2101	103000	1	45.797859	8.644440
1502	1503	1	51	670	32473	1	41.107880	14.208810
669	670	1	51	913	29000	1	45.778591	8.946250
1409	1410	1	51	762	18800	1	45.538689	9.928310
1414	1415	1	51	762	39751	1	41.903221	12.495650
1089	1090	1	51	882	33160	1	45.778999	12.997090
1507	1508	1	51	701	17324	1	45.556549	9.534470
970	971	1	51	701	29000	1	36.855839	14.760470
1198	1199	1	51	1155	38000	1	41.239281	13.933020

In [13]:
```
x_train.head()
```

Out[13]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
527	528	1	51	425	13111	1	45.022388	7.58602
129	130	1	51	1127	21400	1	44.332531	7.54592
602	603	2	51	2039	57039	1	40.748241	14.52835
331	332	1	51	1155	40700	1	42.143860	12.54016
323	324	1	51	425	16783	1	41.903221	12.49565

In [14]:
```
x_train.shape
```

Out[14]: (1030, 8)

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

Out[15]:
```
527    9990
129    9500
602    7590
331    8750
323    9100
Name: price, dtype: int64
```

In [16]:
```
y_test.head()
```

Out[16]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

linear regression

In [17]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[17]:
```
▾ LinearRegression
LinearRegression()
```

In [18]:

```python
ypred=reg.predict(x_test)
ypred
```

Out[18]:

```
array([ 5944.42227279,  7382.19406516,  9714.71708819,  9775.44311825,
        10026.35039712,  9526.60903453,  9728.62372234, 10094.04177203,
         9634.26497953,  9227.25952945, 10449.70113586,  7792.84946168,
         7686.35026828,  6404.97574199,  9519.56733619, 10398.89844376,
         9386.04052678,  7767.75450844,  4682.48932174, 10553.64836355,
        10456.95448735, 10419.26714531,  7673.16408257, 10003.9826247 ,
         7110.92931256,  9091.37732745,  4953.12222998,  6976.50459799,
         7808.74298776,  9656.97773957,  7328.87928145,  5336.18956348,
         5581.53931803,  5133.29004315,  8981.68582577,  5698.30653331,
        10010.97201748,  8329.04920479,  6210.38005208,  8489.42650549,
         9671.42346301,  6962.22983159,  8870.19517156, 10152.36356357,
         8601.72642798, 10155.20812683,  9163.50558976,  8850.10268988,
         7078.34217599,  9040.21569826,  9456.38185771, 10380.46653323,
        10089.95326992,  6929.11259302,  9675.4468094 ,  9481.41110019,
         9400.41281841, 10527.37888794,  9818.84750332,  7347.7283919 ,
         9959.54747529,  7072.02962713,  9956.12264612,  7229.742413  ,
         6479.8942209 ,  9714.46007894,  9829.86290779,  8667.69380715,
         8487.87151048,  6472.87716903,  7863.97818584,  6662.3846006 ,
         8245.10375249, 10522.64398642,  7421.87683441,  8615.1021104 ,
```

In [19]:

```python
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[19]: 0.8395593175117777

In [20]:

```python
import math
math.sqrt(0.8395593175117777)
```

Out[20]: 0.9162746954444272

In [21]:

```python
from sklearn.metrics import mean_squared_error as ns
ns(y_test,ypred)
```

Out[21]: 589208.2941160082

In [23]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=ypred
results=results.reset_index()
results['ID']=results.index
results.head(150)
```

Out[23]:

	index	price	predicted	ID
0	481	7900	5944.422273	0
1	76	7900	7382.194065	1
2	1502	9400	9714.717088	2
3	669	8500	9775.443118	3
4	1409	9700	10026.350397	4
...	...	...	...	...
145	435	10900	10488.151344	145
146	615	10200	9985.171711	146
147	1274	9990	9692.659509	147
148	78	10900	10719.228742	148
149	1167	6900	7562.688963	149

150 rows × 4 columns

In [24]:
```
results['actual price']=results.apply(lambda column:column.price-column.predicted,axis=1)
results
```

Out[24]:

	index	price	predicted	ID	actual price
0	481	7900	5944.422273	0	1955.577727
1	76	7900	7382.194065	1	517.805935
2	1502	9400	9714.717088	2	-314.717088
3	669	8500	9775.443118	3	-1275.443118
4	1409	9700	10026.350397	4	-326.350397
...	...	...	...	...	...
503	291	10900	10096.821809	503	803.178191
504	596	5699	6397.963015	504	-698.963015
505	1489	9500	10104.531307	505	-604.531307
506	1436	6990	8228.184595	506	-1238.184595
507	575	10900	10312.394358	507	587.605642

508 rows × 5 columns

In [25]:
```
''''sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50))
plt.plot()'''
```

Out[25]:
```
"'sns.lineplot(x='ID',y='price',data=results.head(50))\nsns.lineplot(x='ID',y='predicted',data=results.head(50))\npl
t.plot()"
```

ridge algorithm

In [26]:
```
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[26]:
```
▸ GridSearchCV
▸ estimator: Ridge
  ▸ Ridge
```

In [27]:
```
ridge_regressor.best_params_
```

Out[27]:
```
{'alpha': 30}
```

In [28]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
```

Out[28]:
```
▾ Ridge
Ridge(alpha=30)
```

In [29]:
```
ypred=ridge.predict(x_test)
ypred
```

Out[29]:
```
array([ 5935.946929  ,  7391.09263342,  9713.38936976,  9772.39163591,
       10023.47312697,  9525.50186705,  9722.65710008, 10091.63069299,
        9636.73096762,  9224.80526688, 10447.79862925,  7786.80716239,
        7678.19575064,  6421.69337375,  9518.10281958, 10397.95637713,
        9423.49349618,  7756.23104841,  4710.38423862, 10549.78058093,
       10437.24157043, 10418.31826122,  7632.07525135, 10001.0098366 ,
        7105.59647562,  9108.45900086,  4943.04924745,  6968.66299074,
        7803.8514556 ,  9652.86865587,  7322.65097059,  5323.78883956,
        5582.95352334,  5112.19802844,  8941.8991792 ,  5689.25952661,
       10025.39399906,  8301.88195122,  6203.90223358,  8504.37941863,
        9670.09019089,  6971.5463123 ,  8907.49429479, 10170.87993833,
        8596.94188726, 10157.76054033,  9179.69547237,  8847.52279911,
        7067.90650238,  9037.86413624,  9455.08436147, 10377.61732563,
       10088.89631974,  6940.81914325,  9673.96523836,  9499.69113131,
        9434.42007127, 10524.62886148,  9814.37143871,  7358.73006024,
        9956.75151813,  7063.75881413,  9953.33251064,  7221.0307355 ,
        6469.57255928,  9716.86980556,  9827.02367203,  8681.78718364,
        8483.20353285,  6462.42679159,  7854.31330446,  6673.32795121,
        8241.20718281, 10518.84646819,  7414.525471  ,  8608.19638711,
```

mean squared error

In [30]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(ypred,y_test)
Ridge_Error
```

Out[30]:
```
586211.7946814292
```

efficiency

In [31]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[31]:
```
0.8403752605647871
```

In [32]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=ypred
results=results.reset_index()
results['ID']=results.index
results.head(10)
```

Out[32]:

	index	price	predicted	ID
0	481	7900	5935.946929	0
1	76	7900	7391.092633	1
2	1502	9400	9713.389370	2
3	669	8500	9772.391636	3
4	1409	9700	10023.473127	4
5	1414	9900	9525.501867	5
6	1089	9900	9722.657100	6
7	1507	9950	10091.630693	7
8	970	10700	9636.730968	8
9	1198	8999	9224.805267	9

ploting overall

In [33]:
```
''''sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50))
plt.plot()'''
```

Out[33]:
```
"'sns.lineplot(x='ID',y='price',data=results.head(50))\nsns.lineplot(x='ID',y='predicted',data=results.head(50))\npl
t.plot()"
```

elastic regression

In [34]:
```
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[34]:
```
▸ GridSearchCV
▸ estimator: ElasticNet
  ▸ ElasticNet
```

In [35]:
```
elastic_regressor.best_params_
```

Out[35]:
```
{'alpha': 0.01}
```

In [36]:
```
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
```

Out[36]:
```
▾ ElasticNet
ElasticNet(alpha=0.01)
```

In [37]:

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

```
         9365.0016605 ,  9943.36741086,  8304.33332769,  7894.09163625,
        10346.13077992,  5340.9231343 ,  9766.77567364, 10221.32941809,
        10326.30612878,  9192.79866469,  9223.98290751,  9701.78967137,
         5640.27515647,  5073.15709356,  4649.52240787,  9639.46554546,
         6095.21704071,  9868.06349596, 10037.63612681,  4934.93522379,
         8008.61312299,  9677.85950355,  5896.78814422, 10120.24468547,
         5386.85309264,  9602.08238956, 10147.97161034, 10076.0201682 ,
         9578.94736471,  4916.53038674,  5799.38540503,  7183.73138185,
        10035.06741378, 10401.16117446, 10039.59883594,  7784.32777363,
         8717.02851279, 10060.02797243, 10226.30832459,  9842.00228195,
         8362.17553066,  9404.22732378,  8631.8628269 ,  9835.44577072,
         9709.73205232,  9722.22666121,  6847.99378527,  7325.98238042,
         8752.57070479,  9932.5865531 ,  9665.27227701, 10491.77865094,
         8206.01051332,  6834.46798617,  9868.21500226,  8832.18345989,
         9768.29853981, 10359.88100206, 10356.74695473,  9961.10735251,
         9314.97450128,  9875.4541559 ,  9211.95577914, 10118.72619683,
         7935.41408168,  6067.65265801,  8804.74165191, 10276.04473909,
         5789.55515251, 10162.35930573,  9570.83294736,  7797.56435077,
         9443.89967275,  7405.76504525, 10372.00554671,  9983.42468188,
```

In [38]:

```
from sklearn.metrics import mean_squared_error
e=mean_squared_error(ypred,y_test)
e
```

Out[38]: 588604.4467931123

In [39]:

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

Out[39]: 0.8397237443835166

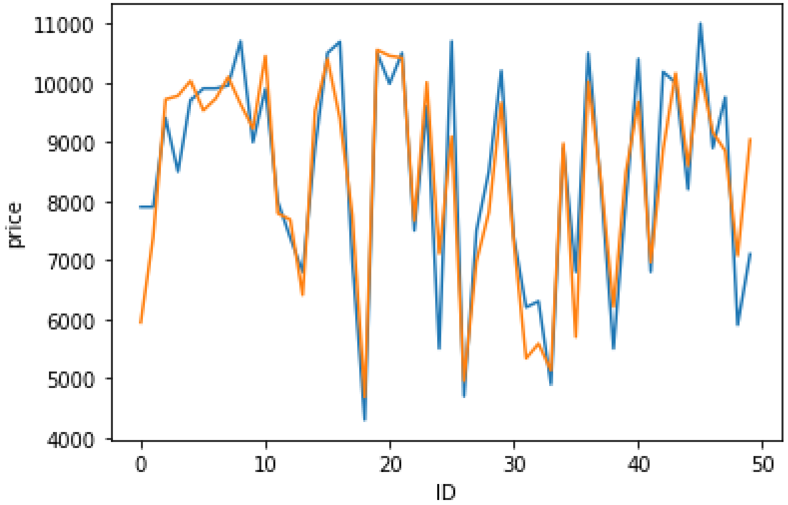
In [40]:
```
results=pd.DataFrame(columns=['price','predicted'])
results['price']=y_test
results['predicted']=ypred
results=results.reset_index()
results['ID']=results.index
results.head(10)
```

Out[40]:

	index	price	predicted	ID
0	481	7900	5942.632146	0
1	76	7900	7383.882687	1
2	1502	9400	9714.493807	2
3	669	8500	9774.901904	3
4	1409	9700	10025.845278	4
5	1414	9900	9526.430700	5
6	1089	9900	9727.540523	6
7	1507	9950	10093.624606	7
8	970	10700	9634.735077	8
9	1198	8999	9226.824793	9

In [43]:
```
sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50))
plt.plot()
```

Out[43]: []



In [ ]:

In [ ]:

In [ ]: