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

In [2]:
```python
sai=pd.read_csv('/home/placement/Downloads/fiat500.csv')
```

In [3]:
```python
sai.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]:
```python
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
```

In [5]:
```
sai['model']=sai['model'].map({'lounge':1,'pop':2,'sport':3})
sai
```

Out[5]:

	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 [6]:
```
s=sai.drop(['lat','lon','ID'],axis=1)
s
```

Out[6]:

	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

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

In [8]:
```
#s=pd.get_dummies(s)
#s
```

In [9]:
```
#cor=s.corr()
```

In [10]:
```
#sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidth=5,cmap='bwr')
```

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

0	1	51	882	25000	1
1	2	51	1186	32500	1
2	3	74	4658	142228	1
3	1	51	2739	160000	1
4	2	73	3074	106880	1
...	...	...	...	...	...
1533	3	51	3712	115280	1
1534	1	74	3835	112000	1
1535	2	51	2223	60457	1
1536	1	51	2557	80750	1
1537	2	51	1766	54276	1

1538 rows × 5 columns

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: 1538, dtype: int64
```

In [13]:
```
# !pip3 scikit-learn
```

train& test data process

In [14]:
```
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 [15]:
```
x_test.head(10)
```

Out[15]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1
1414	1	51	762	39751	1
1089	1	51	882	33160	1
1507	1	51	701	17324	1
970	1	51	701	29000	1
1198	1	51	1155	38000	1

In [16]:
```
x_train.shape
```

Out[16]:
```
(1030, 5)
```

In [17]:
```
x_test.head(5)
```

Out[17]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1

In [18]:
```
y_test.head(5)
```

Out[18]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [19]:
```
y_train.head(5)
```

Out[19]:
```
527    9990
129    9500
602    7590
331    8750
323    9100
Name: price, dtype: int64
```

In [20]:
```
x_train.head(5)
```

Out[20]:

	model	engine_power	age_in_days	km	previous_owners
527	1	51	425	13111	1
129	1	51	1127	21400	1
602	2	51	2039	57039	1
331	1	51	1155	40700	1
323	1	51	425	16783	1

linear regression

In [21]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[21]:
```
▾ LinearRegression
LinearRegression()
```

In [22]:
```
ypred=reg.predict(x_test)
ypred
```

Out[22]:
```
array([ 5994.51703157,  7263.58726658,  9841.90754881,  9699.31627673,
        10014.19892635,  9630.58715835,  9649.4499026 , 10092.9819664 ,
         9879.19498711,  9329.19347948, 10407.2964056 ,  7716.91706011,
         7682.89152522,  6673.95810983,  9639.42618839, 10346.53679153,
         9366.53363673,  7707.90063494,  4727.33552438, 10428.17092937,
        10359.87663878, 10364.84674179,  7680.16157493,  9927.58506055,
         7127.7284177 ,  9097.51161986,  4929.31229715,  6940.60225317,
         7794.35120591,  9600.43942019,  7319.85877519,  5224.05298205,
         5559.52039134,  5201.35403287,  8960.11762682,  5659.72968338,
         9915.79926869,  8255.93615893,  6270.40332834,  8556.73835062,
         9749.72882426,  6873.76758364,  8951.72659758, 10301.95669828,
         8674.89268564, 10301.93257222,  9165.73586068,  8846.92420399,
         7044.68964545,  9052.4031418 ,  9390.75738772, 10267.3912561 ,
        10046.90924744,  6855.71260655,  9761.93338967,  9450.05744337,
         9274.98388541, 10416.00474283,  9771.10646661,  7302.96566423,
        10082.61483093,  6996.96553454,  9829.40534825,  7134.21944391,
         6407.26222178,  9971.82132188,  9757.01618446,  8614.84049875,
         8437.92452169,  6489.24658616,  7752.65456507,  6626.60510856,
         8329.88998217, 10412.00324329,  7342.77348105,  8543.63624413,
```

efficiency

In [23]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[23]:
```
0.8383895235218546
```

mean_squared_error

In [24]:
```
from sklearn.metrics import mean_squared_error as ns
o=ns(y_test,ypred)
o
```

Out[24]:
```
593504.2888137395
```

In [25]:
```
import math
math.sqrt(o)
```

Out[25]: 770.3922954013361

In [26]:
```
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[26]:

	index	price	predicted	ID
0	481	7900	5994.517032	0
1	76	7900	7263.587267	1
2	1502	9400	9841.907549	2
3	669	8500	9699.316277	3
4	1409	9700	10014.198926	4
...	...	...	...	...
145	435	10900	10425.807765	145
146	615	10200	9923.807956	146
147	1274	9990	9781.536212	147
148	78	10900	10586.810806	148
149	1167	6900	7672.288472	149

150 rows × 4 columns

In [27]:
```
results['actual price']=results.apply(lambda column:column.price-column.predicted,axis=1)
results
```

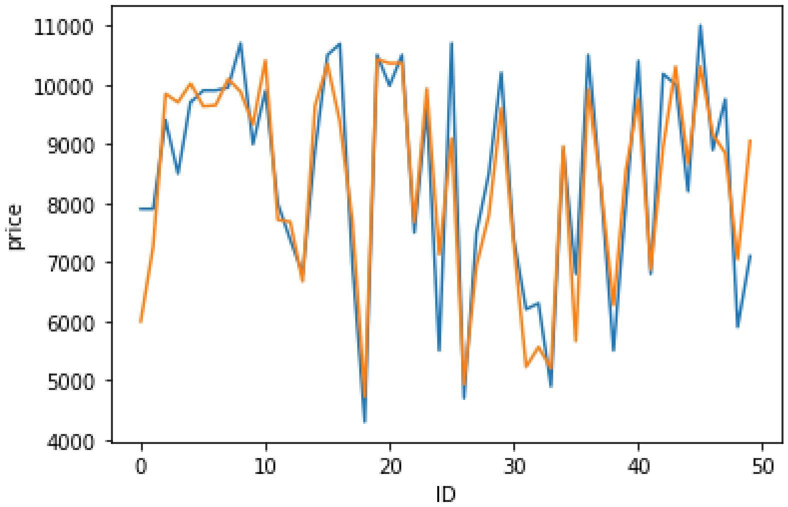
Out[27]:

	index	price	predicted	ID	actual price
0	481	7900	5994.517032	0	1905.482968
1	76	7900	7263.587267	1	636.412733
2	1502	9400	9841.907549	2	-441.907549
3	669	8500	9699.316277	3	-1199.316277
4	1409	9700	10014.198926	4	-314.198926
...	...	...	...	...	...
503	291	10900	10007.364639	503	892.635361
504	596	5699	6390.174715	504	-691.174715
505	1489	9500	10079.478928	505	-579.478928
506	1436	6990	8363.337585	506	-1373.337585
507	575	10900	10344.486077	507	555.513923

508 rows × 5 columns

In [28]:
```
sns.lineplot(x='ID',y='price',data=results.head(50))
sns.lineplot(x='ID',y='predicted',data=results.head(50))
plt.plot()
```

Out[28]: []



In [ ]: