In [1]:
```
import pandas as pd
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [2]:
```
data.describe()
```

Out[2]:

	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 [3]:
```
data1=data.drop(['ID','lat','lon'],axis=1)
```

In [4]:
```
data1
```

Out[4]:

	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 [5]:
```
data1.shape
```

Out[5]:
```
(1538, 6)
```

In [6]:
```
data1=pd.get_dummies(data1)
```

In [7]:
```
data1
```

Out[7]:

	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

1538 rows × 8 columns

In [8]:
```
data1.shape
```

Out[8]:
```
(1538, 8)
```

In [9]:
```
y=data1['price']#adding to separate dataframe the value,we want to predict
x=data1.drop('price',axis=1)#removeing the values we want to predict from the original dataframe
```

In [10]:
```
y
```

Out[10]:
```
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 [11]:
```
!pip3 install scikit-learn
```

```
Requirement already satisfied: scikit-learn in ./anaconda3/lib/python3.10/site-packages (1.2.1)
Requirement already satisfied: scipy>=1.3.2 in ./anaconda3/lib/python3.10/site-packages (from scikit-learn)
(1.10.0)
Requirement already satisfied: joblib>=1.1.1 in ./anaconda3/lib/python3.10/site-packages (from scikit-lear
n) (1.1.1)
Requirement already satisfied: numpy>=1.17.3 in ./anaconda3/lib/python3.10/site-packages (from scikit-lear
n) (1.23.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in ./anaconda3/lib/python3.10/site-packages (from sciki
t-learn) (2.2.0)
```

In [12]:
```
#divide data into training and testing
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 [13]:
```
x_test
```

Out[13]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
481	51	3197	120000	2	0	1	0
76	62	2101	103000	1	0	1	0
1502	51	670	32473	1	1	0	0
669	51	913	29000	1	1	0	0
1409	51	762	18800	1	1	0	0
...	...	...	...	...	...	...	...
291	51	701	22000	1	1	0	0
596	51	3347	85500	1	0	1	0
1489	51	366	22148	1	0	1	0
1436	51	1797	61000	1	1	0	0
575	51	366	19112	1	1	0	0

508 rows × 7 columns

In [14]:
```
x_train
```

Out[14]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
527	51	425	13111	1	1	0	0
129	51	1127	21400	1	1	0	0
602	51	2039	57039	1	0	1	0
331	51	1155	40700	1	1	0	0
323	51	425	16783	1	1	0	0
...	...	...	...	...	...	...	...
1130	51	1127	24000	1	1	0	0
1294	51	852	30000	1	1	0	0
860	51	3409	118000	1	0	1	0
1459	51	762	16700	1	1	0	0
1126	51	701	39207	1	1	0	0

1030 rows × 7 columns

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

Out[15]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [16]:
```
y_train.tail()
```

Out[16]:
```
1130    10990
1294     9800
860      5500
1459     9990
1126     8900
Name: price, dtype: int64
```

In [17]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()#creating object of LinearRegrassion
reg.fit(x_train,y_train)#training and fitting LR object using training data
```

Out[17]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [18]:
```
ypred=reg.predict(x_test)
ypred
```

Out[18]:
```
array([ 5867.6503378 ,  7133.70142341,  9866.35776216,  9723.28874535,
       10039.59101162,  9654.07582608,  9673.14563045, 10118.70728123,
        9903.85952664,  9351.55828437, 10434.34963575,  7732.26255693,
        7698.67240131,  6565.95240435,  9662.90103518, 10373.20344286,
        9599.94844451,  7699.34400418,  4941.33017994, 10455.2719478 ,
       10370.51555682, 10391.60424404,  7529.06622456,  9952.37340054,
        7006.13845729,  9000.1780961 ,  4798.36770637,  6953.10376491,
        7810.39767825,  9623.80497535,  7333.52158317,  5229.18705519,
        5398.21541073,  5157.65652129,  8948.63632836,  5666.62365159,
        9822.1231461 ,  8258.46551788,  6279.2040404 ,  8457.38443276,
        9773.86444066,  6767.04074749,  9182.99904787, 10210.05195479,
        8694.90545226, 10328.43369248,  9069.05761443,  8866.7826029 ,
        7058.39787506,  9073.33877162,  9412.68162121, 10293.69451263,
       10072.49011135,  6748.5794244 ,  9785.95841801,  9354.09969973,
        9507.9444386 , 10443.01608254,  9795.31884316,  7197.84932877,
       10108.31707235,  7009.6597206 ,  9853.90699412,  7146.87414965,
        6417.69133992,  9996.97382441,  9781.18795953,  8515.83255277,
        8456.30006203,  6499.76668237,  7768.57829985,  6832.86406122,
        8347.96113362, 10439.02404036,  7356.43463051,  8562.56562053,
```

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

Out[19]:
```
0.8415526986865394
```

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

Out[20]:
```
581887.727391353
```

In [21]:
```
import math
print(math.sqrt(581887.727391353))
```

```
762.8156575420782
```

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

Out[22]:

	index	price	predicted	Id
0	481	7900	5867.650338	0
1	76	7900	7133.701423	1
2	1502	9400	9866.357762	2
3	669	8500	9723.288745	3
4	1409	9700	10039.591012	4
5	1414	9900	9654.075826	5
6	1089	9900	9673.145630	6
7	1507	9950	10118.707281	7
8	970	10700	9903.859527	8
9	1198	8999	9351.558284	9
10	1088	9890	10434.349636	10
11	576	7990	7732.262557	11
12	965	7380	7698.672401	12
13	1488	6800	6565.952404	13
14	1432	8900	9662.901035	14

In [24]:
```
Results['diff']=Results.apply(lambda row:row.price - row.predicted,axis=1)
```

In [26]:
```
Results.head(50)
```

Out[26]:

	index	price	predicted	Id	diff
0	481	7900	5867.650338	0	2032.349662
1	76	7900	7133.701423	1	766.298577
2	1502	9400	9866.357762	2	-466.357762
3	669	8500	9723.288745	3	-1223.288745
4	1409	9700	10039.591012	4	-339.591012
5	1414	9900	9654.075826	5	245.924174
6	1089	9900	9673.145630	6	226.854370
7	1507	9950	10118.707281	7	-168.707281
8	970	10700	9903.859527	8	796.140473
9	1198	8999	9351.558284	9	-352.558284
10	1088	9890	10434.349636	10	-544.349636
11	576	7990	7732.262557	11	257.737443
12	965	7380	7698.672401	12	-318.672401
13	1488	6800	6565.952404	13	234.047596
14	1432	8900	9662.901035	14	-762.901035
15	380	10500	10373.203443	15	126.796557
16	754	10690	9599.948445	16	1090.051555
17	30	6990	7699.344004	17	-709.344004
18	49	4300	4941.330180	18	-641.330180
19	240	10500	10455.271948	19	44.728052
20	344	9980	10370.515557	20	-390.515557
21	354	10500	10391.604244	21	108.395756
22	124	7500	7529.066225	22	-29.066225
23	383	9600	9952.373401	23	-352.373401

	index	price	predicted	Id	diff
24	1389	5500	7006.138457	24	-1506.138457
25	70	10700	9000.178096	25	1699.821904
26	1058	4700	4798.367706	26	-98.367706
27	839	7500	6953.103765	27	546.896235
28	818	8500	7810.397678	28	689.602322
29	994	10200	9623.804975	29	576.195025
30	1318	7400	7333.521583	30	66.478417
31	324	6200	5229.187055	31	970.812945
32	982	6300	5398.215411	32	901.784589
33	588	4900	5157.656521	33	-257.656521
34	1307	8900	8948.636328	34	-48.636328
35	493	6800	5666.623652	35	1133.376348
36	674	10499	9822.123146	36	676.876854
37	672	8200	8258.465518	37	-58.465518
38	1528	5500	6279.204040	38	-779.204040
39	490	7999	8457.384433	39	-458.384433
40	1090	10400	9773.864441	40	626.135559
41	962	6800	6767.040747	41	32.959253
42	1063	10180	9182.999048	42	997.000952
43	1527	9999	10210.051955	43	-211.051955
44	664	8200	8694.905452	44	-494.905452
45	261	11000	10328.433692	45	671.566308
46	59	8900	9069.057614	46	-169.057614
47	309	9750	8866.782603	47	883.217397
48	820	5900	7058.397875	48	-1158.397875
49	339	7100	9073.338772	49	-1973.338772

In [ ]: