In [3]:
```
#import pandas as r
#d=r.read_csv("/home/placement/Downloads/fiat500")
import pandas
d=pandas.read_csv("/home/placement/Downloads/fiat500")
d.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 [3]:
```
#Removing the columns
d1=d.drop(['lat','ID','lon'],axis=1)
d1
```

Out[3]:

	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 [4]:
```
#converting the strings into integer
d1=r.get_dummies(d1)
d1.describe()
```

Out[4]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_sport
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
mean	51.904421	1650.980494	53396.011704	1.123537	8576.003901	0.711313	0.232770	0.055917
std	3.988023	1289.522278	40046.830723	0.416423	1939.958641	0.453299	0.422734	0.229836
min	51.000000	366.000000	1232.000000	1.000000	2500.000000	0.000000	0.000000	0.000000
25%	51.000000	670.000000	20006.250000	1.000000	7122.500000	0.000000	0.000000	0.000000
50%	51.000000	1035.000000	39031.000000	1.000000	9000.000000	1.000000	0.000000	0.000000
75%	51.000000	2616.000000	79667.750000	1.000000	10000.000000	1.000000	0.000000	0.000000
max	77.000000	4658.000000	235000.000000	4.000000	11100.000000	1.000000	1.000000	1.000000

In [5]:
```
#This command is to find the number of rows and columns
d1.shape
```

Out[5]: (1538, 8)

In [6]:
```
#splitting the data to predict
y=d1['price']
x=d1.drop('price',axis=1)
x
```

Out[6]:

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

1538 rows × 7 columns

In [7]:
```
y
```

Out[7]:
```
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 [8]:
```
x
```

Out[8]:

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

1538 rows × 7 columns

In [9]:
```
#spliting data to create the model
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 [10]:
```
x_test.describe()
```

Out[10]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
count	508.000000	508.000000	508.000000	508.000000	508.000000	508.000000	508.000000
mean	51.675197	1618.320866	52570.645669	1.143701	0.726378	0.228346	0.045276
std	3.448525	1278.194929	38793.890295	0.454020	0.446257	0.420181	0.208113
min	51.000000	366.000000	3000.000000	1.000000	0.000000	0.000000	0.000000
25%	51.000000	670.000000	19737.250000	1.000000	0.000000	0.000000	0.000000
50%	51.000000	867.000000	39162.000000	1.000000	1.000000	0.000000	0.000000
75%	51.000000	2534.500000	77152.500000	1.000000	1.000000	0.000000	0.000000
max	74.000000	4658.000000	220000.000000	4.000000	1.000000	1.000000	1.000000

In [11]:
```
x_train.describe()
```

Out[11]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	52.017476	1667.088350	53803.085437	1.113592	0.703883	0.234951	0.061165
std	4.225850	1295.387249	40662.960946	0.396424	0.456765	0.424174	0.239749
min	51.000000	366.000000	1232.000000	1.000000	0.000000	0.000000	0.000000
25%	51.000000	670.000000	20337.500000	1.000000	0.000000	0.000000	0.000000
50%	51.000000	1066.000000	38987.000000	1.000000	1.000000	0.000000	0.000000
75%	51.000000	2647.000000	80000.000000	1.000000	1.000000	0.000000	0.000000
max	77.000000	4658.000000	235000.000000	4.000000	1.000000	1.000000	1.000000

In [12]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)  #training and fitting LR object using training data
```

Out[12]:
```
LinearRegression
LinearRegression()
```

In [13]:
```
ypred=reg.predict(x_test)
ypred
```

```
 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,
 9820.78555199, 10035.83571539,  7370.77198022,  9411.45894006,
10352.85155564,  8045.21588007, 10446.80664758,  3736.20118868,
```

In [14]:
```
from sklearn.metrics import r2_score #to  check the efficiency
r2_score(y_test,ypred)
```

Out[14]:
```
0.8415526986865394
```

In [17]:
```
from sklearn.metrics import mean_squared_error #calculate MSE

mean_squared_error(ypred,y_test)
```

Out[17]: 581887.727391353

In [18]:
```
print(sqrt(mean_squared_error(ypred,y_test)))
```

762.8156575420782

In [20]:
```
results=r.DataFrame(columns=['Price','Predicted']) #To compare the actual and predicted price
results['Price']=y_test
results['Predicted']=ypred
results
```

Out[20]:

	Price	Predicted
481	7900	5867.650338
76	7900	7133.701423
1502	9400	9866.357762
669	8500	9723.288745
1409	9700	10039.591012
...	...	...
291	10900	10032.665135
596	5699	6281.536277
1489	9500	9986.327508
1436	6990	8381.517020
575	10900	10371.142553

508 rows × 2 columns

In [21]:
```
#The command is to find difference between the predicted and actual price
results['Difference']=results.apply(lambda row:row.Predicted  - row.Price,axis=1)
results
```

Out[21]:

	Price	Predicted	Difference
481	7900	5867.650338	-2032.349662
76	7900	7133.701423	-766.298577
1502	9400	9866.357762	466.357762
669	8500	9723.288745	1223.288745
1409	9700	10039.591012	339.591012
...	...	...	...
291	10900	10032.665135	-867.334865
596	5699	6281.536277	582.536277
1489	9500	9986.327508	486.327508
1436	6990	8381.517020	1391.517020
575	10900	10371.142553	-528.857447

508 rows × 3 columns