

```
In [123]: import pandas as pd
import numpy as np
```

```
In [124]: data= pd.read_csv('/home/placement/Desktop/fiat500.csv')
```

```
In [125]: data
```

```
Out[125]:
```

	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

1538 rows × 9 columns

In [126]: data.describe()

Out[126]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
<b>count</b>	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
<b>mean</b>	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
<b>std</b>	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
<b>min</b>	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
<b>25%</b>	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
<b>50%</b>	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
<b>75%</b>	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
<b>max</b>	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [127]: data1=data.drop(columns=["ID", "lat", "lon"])

In [128]: data1

Out[128]:

	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

## getting dummies to the data

In [129]: data2=pd.get\_dummies(data1)

In [130]: data2

Out[130]:

	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 [131]: data2.shape

Out[131]: (1538, 8)

**removed the unwanted data from the data frame**

In [132]: `y=data2['price']`  
`x=data2.drop('price',axis=1)      #unwanted columns removed`

In [133]:

x

Out[133]:

	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 [134]:

y

Out[134]:

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 [135]: ## !pip instal scikit-learn
```

## splitting the data into testing set and training set

```
In [136]: 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) #66 & 33
```

```
In [137]: x_test
```

```
Out[137]:
```

	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 [138]: x\_train

Out[138]:

	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 [139]: y\_test

Out[139]:

481	7900
76	7900
1502	9400
669	8500
1409	9700
...	...
291	10900
596	5699
1489	9500
1436	6990
575	10900

Name: price, Length: 508, dtype: int64

```
In [140]: y_train
```

```
Out[140]: 527      9990
          129      9500
          602      7590
          331      8750
          323      9100
          ...
          1130     10990
          1294      9800
          860      5500
          1459      9990
          1126      8900
          Name: price, Length: 1030, dtype: int64
```

```
In [141]: from sklearn.linear_model import LinearRegression
          reg=LinearRegression()  ## creating object of linear regression
          reg.fit(x_train,y_train)  ## training and fitting LR data using training data
```

```
Out[141]: 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 [142]: ypred=reg.predict(x_test)
ypred
```

```
Out[142]: 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,
  8820.78555100, 10025.02571520,  7270.77100000,  8411.45004000]
```

```
In [143]: from sklearn.metrics import r2_score      # to know the effency b/w the predicted price and actual price
r2_score(y_test,ypred)                             # y_test is the actual price and ypred is the predicted price
```

```
Out[143]: 0.8415526986865394
```

```
In [147]: from sklearn.metrics import mean_squared_error #calculating Mean Square Error (MSR)
mean_squared_error(y_test,ypred)
```

```
Out[147]: 581887.727391353
```

```
In [161]: import math
a=581887.727391353
print(math.sqrt(a))
```

```
762.8156575420782
```

In [167]: `y_test.head(10)`

```
Out[167]: 481      7900
          76      7900
          1502    9400
          669     8500
          1409    9700
          1414    9900
          1089    9900
          1507    9950
          970    10700
          1198    8999
          Name: price, dtype: int64
```

In [168]: `ypred`

```
Out[168]: 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,
        8820.78555100, 10025.02571520,  7270.77108022,  8411.45804006])
```

```
In [172]: Results=pd.DataFrame(columns=['Price', 'Predicted'])  
Results['Price']=y_test  
Results['Predicted']=ypred
```

```
In [173]: Results
```

```
Out[173]:
```

	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 [174]: Results['Difference']=Results.apply(lambda row:row.Price-row.Predicted ,axis=1)
```

In [175]: Results

Out[175]:

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

508 rows × 3 columns