In [151]:
```
import pandas as pd
```

In [152]:
```
data=pd.read_csv("/home/palcement/Downloads/fiat500.csv")
```

In [153]:
```
data.head(10)#https://studyworld123.github.io/fiat/fiat1.pdf
```

Out[153]:

	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
5	6	pop	74	3623	70225	1	45.000702	7.682270	7900
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
8	9	sport	73	4049	76000	1	45.548000	11.549470	5600
9	10	sport	51	3653	89000	1	45.438301	10.991700	6000

In [154]:
```
data1=data.drop(['ID','lat','lon'],axis=1)  #To drop the colums in a data frame
```

In [155]:
```
data1
```

Out[155]:

	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 [156]:
```
data1=pd.get_dummies(data1)  ##This command is used to encode the string in to the numbers.
                        ## Here we observe that the lounge is converted to the 1  0  0
                        ## Here we observe that the pop is converted to the 0      1       0
                        ##Here we observe that the sport is converted to the 0  0     1
```

In [157]:
```
data1
```

Out[157]:

	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 [158]:
```
data1.shape  #It will show the no of rows and the columns
               # After removing the columns of ID,lat,lon
```

Out[158]:
```
(1538, 8)
```

In [159]:
```
data.shape
```

Out[159]:
```
(1538, 9)
```

In [160]:
```
#asdf;lkj asdf ;lkj asdf ;lkj asdf ;lkj asdf ;lkj asdf ;lkj asdf ;lkj sdf ;lkj asdf ;lkj asdf ;LKJ
```

In [161]:
```
y=data1['price']   #in the dataset named as fiat500, we simply only take the price as seperate and store the
X=data1.drop('price',axis=1) # in the data frame we removed the price column and remaining data stored in th
```

In [162]:
```
y
```

Out[162]:
```
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 [163]:
```
X
```

Out[163]:

	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 [164]:
```
#!pip3 install scikit-learn    to install skleran package run this command
```

In [165]:
```
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 [166]:
```
X_test.head(5)
```

Out[166]:

	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

In [167]:
```
X_train.head()
```

Out[167]:

	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

In [168]:
```
y_test.head(5)
```

Out[168]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [169]:
```
y_train.head(8)
```

Out[169]:
```
527      9990
129      9500
602      7590
331      8750
323      9100
1358    10900
522     10800
584      9999
Name: price, dtype: int64
```

In [170]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression() #creating of Linear Regression
reg.fit(X_train,y_train) #training and fitting LR object using training data
```

Out[170]:
```
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 [171]:
```
ypred=reg.predict(X_test)
```

In [172]:
```
ypred
```

Out[172]:
```
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 [174]:
```
from sklearn.metrics import r2_score  #model efficinecy step    #y_test is the actual price    #y_predit is
r2_score(y_test,ypred)
```

Out[174]:
```
0.8415526986865394
```

In [176]:
```
from sklearn.metrics import mean_squared_error ##Calulating mean square error
mean_squared_error(y_test,ypred)
```

Out[176]:
```
581887.727391353
```

In [181]:
```
import math
a=581887.727391353
math.sqrt(a)
```

Out[181]:
```
762.8156575420782
```

In [183]:
```
y_test.head()
```

Out[183]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [186]:
```
ypred
```

Out[186]:
```
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,
        9820.78555199, 10035.83571539,  7370.77198022,  9411.45894006
```

In [201]:
```
#Results=pd.DataFrame(columns=['Actual','Predicted'])
#Results['Actual']=y_test
Results=pd.DataFrame(columns=['Price','Predicted']) #price and predicted names are our wish
Results['Price']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()   #This line is optional
# Results['Id']=Results.index        #This line is optional
Results.head(15)
```

Out[201]:

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

In [202]:
```
Results['diff_price']=Results.apply(lambda row:row.Price-row.Predicted,axis=1)
```

In [204]:
```
Results.head(15)
```

Out[204]:

	index	Price	Predicted	diff_price
0	481	7900	5867.650338	2032.349662
1	76	7900	7133.701423	766.298577
2	1502	9400	9866.357762	-466.357762
3	669	8500	9723.288745	-1223.288745
4	1409	9700	10039.591012	-339.591012
5	1414	9900	9654.075826	245.924174
6	1089	9900	9673.145630	226.854370
7	1507	9950	10118.707281	-168.707281
8	970	10700	9903.859527	796.140473
9	1198	8999	9351.558284	-352.558284
10	1088	9890	10434.349636	-544.349636
11	576	7990	7732.262557	257.737443
12	965	7380	7698.672401	-318.672401
13	1488	6800	6565.952404	234.047596
14	1432	8900	9662.901035	-762.901035

In [ ]:

In [ ]: