# LINEAR REGRESSION

Linear regression model for Fiat500 data vehicle data

In [103]:
```
import pandas as pd
```

In [104]:
```
data=pd.read_csv("/home/placement/Desktop/saimohan data/csv files/fiat500.csv")
```

In [105]:
```
data
```

Out[105]:

| | 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 [106]:
```
data.describe()
```

Out[106]:

| | 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 [107]:
```
data.info
```

Out[107]:
```
<bound method DataFrame.info of         ID   model  engine_power  age_in_days      km  previous_owners  \
0        1  lounge            51          882   25000                1
1        2     pop            51         1186   32500                1
2        3   sport            74         4658  142228                1
3        4  lounge            51         2739  160000                1
4        5     pop            73         3074  106880                1
...    ...     ...           ...          ...     ...              ...
1533  1534   sport            51         3712  115280                1
1534  1535  lounge            74         3835  112000                1
1535  1536     pop            51         2223   60457                1
1536  1537  lounge            51         2557   80750                1
1537  1538     pop            51         1766   54276                1

            lat        lon  price
0     44.907242   8.611560   8900
1     45.666359  12.241890   8800
2     45.503300  11.417840   4200
3     40.633171  17.634609   6000
4     41.903221  12.495650   5700
...         ...        ...    ...
1533  45.069679   7.704920   5200
1534  45.845692   8.666870   4600
1535  45.481541   9.413480   7500
1536  45.000702   7.682270   5990
1537  40.323410  17.568270   7900

[1538 rows x 9 columns]>
```

In [108]:
```
data1=data.drop(["lat","lon","ID"],axis=1)
```

In [109]:
```
data1=pd.get_dummies(data1)
```

In [110]:
```
data1
```

Out[110]:

| | 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

model_lounge=100 model_pop=010 model_sport=001 #this are the string assigned with numbers

In [111]:
```
data1.shape
```

Out[111]:
```
(1538, 8)
```

In [112]:
```
#which is the parameter is predected values can be removed from the data file
#1) we copied the data into another data("y")
#2) later we can removed those file from main data set
y=data1['price']
x=data1.drop(['price'],axis=1)
```

In [113]:
```
y
```

Out[113]:
```
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
```

# sklearn package ----------

In [114]:
```
#!pip install scikit-learn
#this is install line for installing the package
```

In [115]:
```
#i am calling function to split
#split enter data into ->67% traning  ,  ->33% testing

from sklearn.model_selection import train_test_split
#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 [116]:
```
x_train.head(5)
```

Out[116]:

| | 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 [117]:
```
#testing and training
```

In [118]:
```
y_train.head(5)
```

Out[118]:
```
527    9990
129    9500
602    7590
331    8750
323    9100
Name: price, dtype: int64
```

In [119]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[119]:
```
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 [120]:
```
ypred=reg.predict(x_test)
```

In [121]:
```
ypred
```

Out[121]:
```
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,
       10352.85155564,  8045.21588007, 10446.80664758,  3736.20118868,
       10348.63930496, 10435.96627494,  6167.80169017, 10390.11317804,
        6527.69471073,  9116.4755691 , 10484.52829   ,  9335.69889855,
        6709.57413543,  3390.72353093, 10106.33753331,  9792.46732008,
        6239.49568346,  4996.26346266,  9044.38667681,  9868.09959448,
        5484.13199252,  5698.5954821 , 10086.86206874,  8115.81693479,
       10392.37800936,  6835.6573351 ,  6657.61744836,  5738.50576764,
        8896.80120764,  9952.37340054, 10390.28377419,  9419.10788866,
        9082.56591129, 10122.82465116, 10410.00504522, 10151.77663915,
        9714.85367238,  9291.92963633, 10346.99073888,  5384.22311343,
        9772.85146492,  6069.77107828,  9023.26394782, 10220.56195956,
        9238.89392583,  9931.47195375,  8321.42715662,  8377.80491069,
        7528.53327408, 10552.64805598, 10465.02437243, 10110.68940664,
       10238.17869436,  6841.77264488,  9625.64505547, 10412.59988875,
        9653.06224923,  7948.63618724,  9704.82523573,  7971.05970955,
       10399.51752022,  9176.43567301,  5803.03205787,  6698.19524313,
        8257.83550573, 10452.95284574,  9948.66454584,  9789.65062843,
       10582.50828537,  7568.91955482,  6804.97705225,  8065.01292384,
       10310.29143419,  8836.34894739,  8390.05091229,  9582.13932508,
```

```
 9745.34784981, 10045.45021387, 10294.09872915,  7145.15315349,
 9727.85493167,  6281.78952194,  7901.36245623,  9387.9203723 ,
 5039.55649797,  9351.49777725,  9980.70844784, 10094.79341516,
 6359.24321991,  9856.10227211,  9099.07023804,  5234.05388382,
 5534.45288323,  4495.02309231, 10199.78432943, 10024.87037067,
 5465.58034188,  8520.72057674,  7034.71038647, 10054.65061446,
10191.12067767,  6008.34860428,  9748.18097947,  9669.4333196 ,
 9145.3756075 ,  9175.66562699, 10087.86753845,  9825.02990067,
 7340.29803785,  5083.8487301 ,  9441.50914802, 10243.05490667,
 5556.42300245, 10676.01945733,  6126.99295838,  9845.16661356,
 9850.77978959,  7840.83596305,  6552.05146566,  9938.82104889,
 8327.79232274,  9119.62204137,  6111.83787367, 10410.00504522,
 6360.97695249,  8601.59209793,  8377.80258216,  9803.81343895,
 8285.09831762, 10091.75635129, 10003.86694939, 10028.60283146,
10354.61956534,  8552.21002673,  6726.65446676,  9381.22662706,
 6520.9999373 , 10352.85155564,  9063.7534579 , 10456.89121831,
 9127.72470241,  9952.37340054,  8376.6975881 ,  9220.36267675,
10036.24981328,  8418.65456209,  4717.7579531 , 10076.86950203,
10017.8490121 , 10590.33289679, 10161.75393066,  4927.49556508,
 7276.18410037,  9678.26477249,  9764.65653403,  5643.53722047,
10062.84554534,  5163.04602382,  8307.60791348,  7441.80993846,
 7868.82460983,  9725.36143983,  8669.20982667, 10447.15719448,
 7124.58453563,  9718.32989102,  8059.66615638,  7430.65975056,
10425.57075395, 10364.18738085,  5433.2724385 ,  9102.40298437,
 9629.06913727, 10532.3506032 , 10129.42684118,  9149.48843328,
 6158.13422239,  9721.03634157, 10419.02236947,  8838.50241314,
 8182.78836676, 10012.21373766,  9468.92324529,  9904.31954667,
10475.66003551, 10475.0702782 ,  9609.27020577,  8115.22501265,
10439.02404036, 10363.81936482,  8720.0683498 ,  8274.3579289 ,
 6889.7195761 , 10191.45963957,  4819.0674709 ,  8814.11814085,
 5737.62378403, 10051.06593609,  8840.87520652, 10054.31165256,
 9686.269121  , 10463.56977746, 10133.15815395,  9762.80613855,
 9793.03056946,  6796.69068198,  9599.3262671 ,  8488.31539047,
 6705.66818403, 10307.58651641, 10045.18332239, 10120.36242166,
 5836.93199112,  8772.49782933,  9680.77538859,  5719.87463854,
 8398.59735084,  9680.77538859,  4334.81943405, 10015.00600846,
 9850.72458719,  7864.73798641, 10072.71245374, 10552.64805598,
10253.47474908,  6861.80736606,  6484.22649656, 10374.62123623,
 8426.37409382,  5447.47569851,  9914.20077691,  4687.39013431,
 7885.32100747,  5431.00822998,  9911.86294348, 10390.16991322,
 9680.84745901,  8844.57815539,  7764.08471024,  4257.54640953,
 9882.76503303, 10341.35258769,  5736.4484335 , 10179.87154436,
```

```
 9501.423448  ,  7997.3181334 ,  5532.33458288,  9894.57834738,
10437.97459358,  6381.35845844,  9591.23555726,  9574.27908517,
10322.30715736,  9501.22785499,  9789.955758  ,  9593.26549752,
 6775.82788536,  7915.34831306, 10389.98590521, 10351.58343315,
 7381.32686464,  9966.53983093, 10430.87188433, 10554.43156462,
10285.85574963, 10035.88086558,  9526.63034431,  7742.78157141,
 9297.64938364, 10051.42272678, 10004.81256571,  9985.84167026,
 9374.6573594 ,  9561.57499854,  9754.94184269,  9819.85893758,
 8780.31447831,  6255.99008069,  6281.53627686,  8190.88781577,
 8588.91394592,  6566.97963218,  6850.70237466,  5511.29438169,
 8119.97866315,  9847.74830838,  7775.93862032,  9875.05509733,
10121.29366536,  5791.92464084,  9835.42728501, 10043.91426822,
 8027.28015259,  4527.22080416, 10609.02444098,  3808.29240951,
 9952.37340054, 10511.20945172,  5746.34019592,  5486.40214756,
10395.91036208,  6788.47519216,  8953.20120295, 10442.24187982,
 9455.6934072 ,  9976.26574762,  8528.35753837,  7960.77147517,
10400.05054235,  5359.97362399,  9899.4913613 , 10203.35814213,
10303.33499967,  9507.16596227,  9151.43928526,  9805.06469343,
 5661.99787503,  4904.40690461,  4742.8827765 ,  9663.32864144,
 6102.95247322,  9870.62050425, 10066.06916341,  5001.24291171,
 8029.35471733,  9773.79143856,  5962.75261232, 10401.02638592,
 5511.44251977,  9627.19072277, 10106.26833963, 10199.67798189,
 9458.07047019,  4890.1778697 ,  5833.90060934,  7022.25799652,
10011.26407146, 10402.02002918,  9945.08219601,  7770.52280413,
 8840.08397206,  9916.27565791, 10287.45603992,  9964.3213269 ,
 8403.51255128,  9345.81907605,  8521.46225147,  9743.68712672,
 9791.34520178,  9779.16293972,  6753.27416058,  7354.16762745,
 8760.24542762,  9923.66596418,  9812.92276721, 10466.90125415,
 8163.46726237,  6659.46839415,  9987.65677522,  8866.7826029 ,
 9952.37340054, 10187.72427693, 10231.39378767, 10091.11325493,
 9365.98570732, 10009.10088406,  9141.00566394, 10099.11667176,
 7803.77049829,  6009.84398185,  8800.33824151, 10237.60733785,
 5609.98366311, 10097.61555355,  9684.99946572,  7644.67379732,
 9276.37891542,  7371.5492091 , 10287.98873148, 10067.26428381,
10552.64805598,  9966.72383894, 10068.46126756,  6232.53552963,
10584.55044373,  9965.98687522, 10529.44404458,  9602.67646085,
 9665.77720284,  6186.06948587,  8073.87436253, 10345.58323918,
 6344.74803956,  7361.62678204, 10058.57116223,  6792.219309  ,
 7897.72464823,  5261.45936067,  4540.24137423,  8709.36468047,
 6882.0117409 ,  7406.73353952,  6795.61189392,  7047.27998963,
 9945.33400083,  8856.93910595,  9378.02074127, 10389.561154  ,
10092.46332921, 10381.52000388,  9723.92466625,  5996.3331428 ,
```

```
9786.14866981,  7708.49649098,  5583.48163469,  4932.92788329,
9856.66053994,  9236.22981005, 10092.64052142,  6256.43516278,
8592.63841379, 10341.5365957 ,  5177.96595576, 10032.66513491,
6281.53627686,  9986.327508  ,  8381.51701951, 10371.14255313])
```

In [122]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
#we finded model effecency by importing r2_score
```

Out[122]: 0.8415526986865394

In [123]:
```
#root mean square error
```

In [124]:
```
from sklearn.metrics import mean_squared_error #calculating MSE
mean_squared_error(ypred,y_test)
```

Out[124]: 581887.727391353

In [125]:
```
#write the python code to find the root mean square value
```

In [128]:
```
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[128]:

| | 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 [132]:
```
#data2["SWM"]=data2.apply(lambda row:row.JUN+row.JUL+row.AUG+row.SEP,axis=1)
# ---> i want difference b/w price and predicted ,so i use
Results["predicted value"]=Results.apply(lambda row:row.price-row.predicted,axis=1)
Results
```

Out[132]:

| | index | price | predicted | ID | predicted value |
|---|---|---|---|---|---|
| **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 |
| **...** | ... | ... | ... | ... | ... |
| **503** | 291 | 10900 | 10032.665135 | 503 | 867.334865 |
| **504** | 596 | 5699 | 6281.536277 | 504 | -582.536277 |
| **505** | 1489 | 9500 | 9986.327508 | 505 | -486.327508 |
| **506** | 1436 | 6990 | 8381.517020 | 506 | -1391.517020 |
| **507** | 575 | 10900 | 10371.142553 | 507 | 528.857447 |

508 rows × 5 columns

In [ ]:

In [ ]: