Property Price Prediction System using Multiple Regression Models

#

STEP 1 : Importing the required Packages into our Python Environment

Libraries: These are frameworks in python to handle commonly required tasks. I Import any budding data scientists to familiarise themselves with these libraries:

Pandas — For handling structured data

NumPy — For linear algebra and mathematics

Seaborn — For data visualization

Matplotlib — For data visualisation graphical plotting library

Scikit Learn — For machine learning

In [1]:

```
#import required libraries
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

import statistics as sts
import math as mt
import warnings
warnings.filterwarnings('ignore')
```

STEP 2 : Importing Data and EDA

Importing the property prediction data and do some EDA on it

First, Let’s import the data and have a look to see what kind of data we are dealing with:

#

Data Description:

Train.csv - 29451 rows x 12 columns

Test.csv - 68720 rows x 11 columns

Attributes Description:

In [2]:

```
from IPython import display
display.Image("Attributes.png")
```

Out[2]:

Column	Description
POSTED_BY	Category marking who has listed the prope
UNDER_CONSTRUCTION	Under Construction or Not
RERA	Rera approved or Not
BHK_NO	Number of Rooms
BHK*OR*RK	Type of property
SQUARE_FT	Total area of the house in square feet
READY*TO*MOVE	Category marking Ready to move or Not
RESALE	Category marking Resale or not
ADDRESS	Address of the property
LONGITUDE	Longitude of the property
LATITUDE	Latitude of the property

In [3]:

```
#import Data
House_train_data = pd.read_csv('House Price train.csv')
```

In [4]:

```
House_train_data
```

Out[4]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	BHK_OR_RK	SQUARE_FT	RE
0	Owner	0	0	2	BHK	1300.236407	
1	Dealer	0	0	2	BHK	1275.000000	
2	Owner	0	0	2	BHK	933.159722	
3	Owner	0	1	2	BHK	929.921143	
4	Dealer	1	0	2	BHK	999.009247	
...	...	...	...	...	...	...	
29446	Owner	0	0	3	BHK	2500.000000	
29447	Owner	0	0	2	BHK	769.230769	
29448	Dealer	0	0	2	BHK	1022.641509	
29449	Owner	0	0	2	BHK	927.079009	
29450	Dealer	0	1	2	BHK	896.774194	

29451 rows × 12 columns

In [5]:

```
#get some information about our Data-Set
House_train_data.shape
```

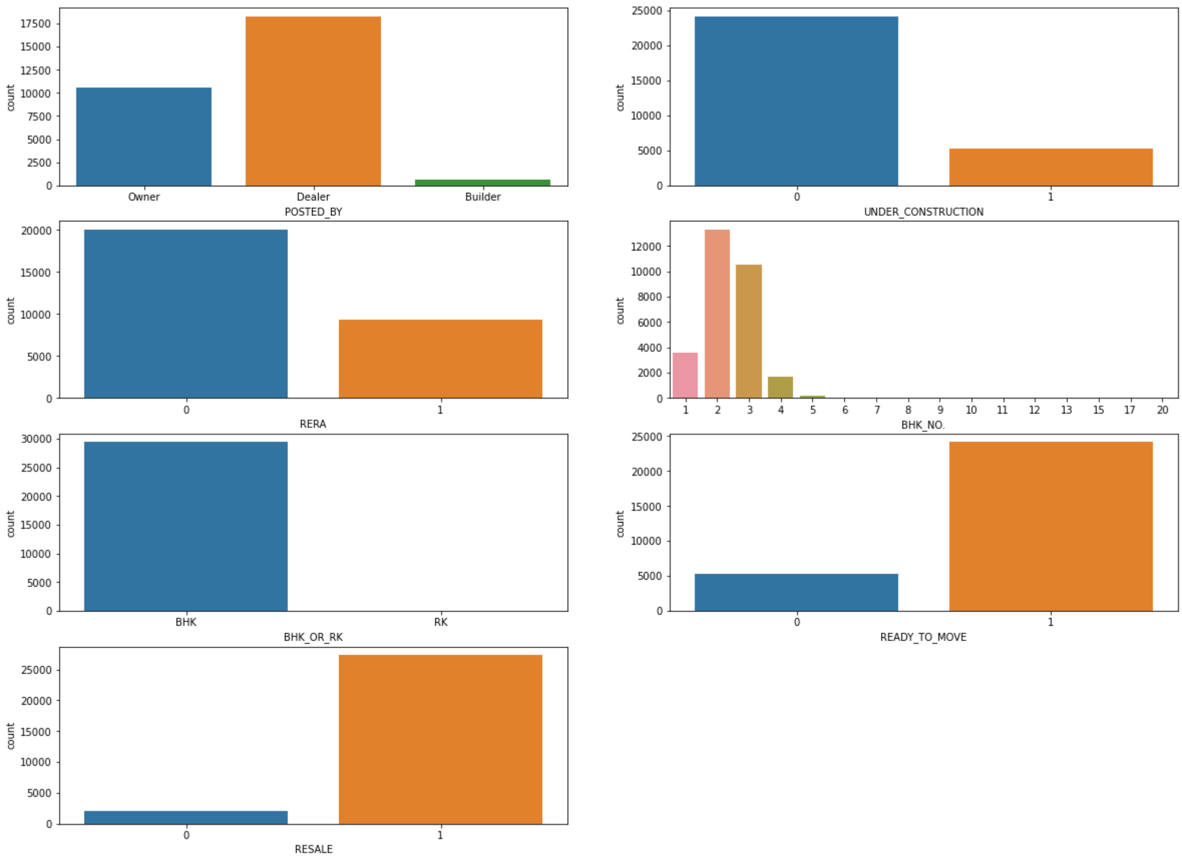
Out[5]:

```
(29451, 12)
```

In [6]:

```
# Exploratory Data Analysis

i = 1
plt.figure(figsize=(20,15))
for col in ['POSTED_BY' , 'UNDER_CONSTRUCTION' , 'RERA' , 'BHK_NO.' , 'BHK_OR_RK' ,'REA
    plt.subplot(4,2,i)
    sns.countplot(House_train_data[col])
    i+=1
```



In [7]:

```
House_test_data = pd.read_csv('House Price test.csv')
```

In [8]:

```
House_test_data
```

Out[8]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	BHK_OR_RK	SQUARE_FT	RI
0	Owner	0	0	1	BHK	545.171340	
1	Dealer	1	1	2	BHK	800.000000	
2	Dealer	0	0	2	BHK	1257.096513	
3	Dealer	0	0	3	BHK	1400.329489	
4	Owner	0	0	1	BHK	430.477830	
...	...	...	...	...	...	...	
68715	Dealer	0	1	2	BHK	856.555505	
68716	Dealer	0	1	3	BHK	2304.147465	
68717	Dealer	1	1	1	BHK	33362.792750	
68718	Dealer	0	0	2	BHK	1173.708920	
68719	Dealer	0	0	3	BHK	2439.532944	

68720 rows × 11 columns

In [9]:

```
House_test_data.shape
```

Out[9]:

```
(68720, 11)
```

In [10]:

```
House_train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29451 entries, 0 to 29450
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   POSTED_BY              29451 non-null  object 
 1   UNDER_CONSTRUCTION     29451 non-null  int64  
 2   RERA                   29451 non-null  int64  
 3   BHK_NO.                29451 non-null  int64  
 4   BHK_OR_RK              29451 non-null  object 
 5   SQUARE_FT              29451 non-null  float64
 6   READY_TO_MOVE          29451 non-null  int64  
 7   RESALE                 29451 non-null  int64  
 8   ADDRESS                29451 non-null  object 
 9   LONGITUDE              29451 non-null  float64
 10  LATITUDE               29451 non-null  float64
 11  TARGET(PRICE_IN_LACS)  29451 non-null  float64
dtypes: float64(4), int64(5), object(3)
memory usage: 2.7+ MB
```

In [11]:

```
House_test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68720 entries, 0 to 68719
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   POSTED_BY           68720 non-null  object 
 1   UNDER_CONSTRUCTION  68720 non-null  int64  
 2   RERA                68720 non-null  int64  
 3   BHK_NO.             68720 non-null  int64  
 4   BHK_OR_RK           68720 non-null  object 
 5   SQUARE_FT           68720 non-null  float64
 6   READY_TO_MOVE       68720 non-null  int64  
 7   RESALE              68720 non-null  int64  
 8   ADDRESS             68720 non-null  object 
 9   LONGITUDE           68720 non-null  float64
 10  LATITUDE            68720 non-null  float64
dtypes: float64(3), int64(5), object(3)
memory usage: 5.8+ MB
```

In [12]:

```
House_train_data.describe()
```

Out[12]:

	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	READY_TO_MOVE
count	29451.000000	29451.000000	29451.000000	2.945100e+04	29451.000000
mean	0.179756	0.317918	2.392279	1.980217e+04	0.820244
std	0.383991	0.465675	0.879091	1.901335e+06	0.383991
min	0.000000	0.000000	1.000000	3.000000e+00	0.000000
25%	0.000000	0.000000	2.000000	9.000211e+02	1.000000
50%	0.000000	0.000000	2.000000	1.175057e+03	1.000000
75%	0.000000	1.000000	3.000000	1.550688e+03	1.000000
max	1.000000	1.000000	20.000000	2.545455e+08	1.000000

In [13]:

```
House_train_data.describe(include = "object")
```

Out[13]:

	POSTED_BY	BHK_OR_RK	ADDRESS
count	29451	29451	29451
unique	3	2	6899
top	Dealer	BHK	Zirakpur,Chandigarh
freq	18291	29427	509

Determining the number of entries

In [14]:

```
print("Total no of unique values available in POSTED_BY =" ,House_train_data["POSTED_BY
print("Uniques Data  that are available in 1st feature are = ",House_train_data["POSTED
print("Number of entries of Unique value available in POSTED_BY =" ,House_train_data["P
type(House_train_data["POSTED_BY"])
```

```
Total no of unique values available in POSTED_BY = 3
Uniques Data  that are available in 1st feature are =  ['Owner' 'Dealer' 'Bu
ilder']
Number of entries of Unique value available in POSTED_BY = Dealer     18291
Owner      10538
Builder      622
Name: POSTED_BY, dtype: int64
```

Out[14]:

```
pandas.core.series.Series
```

In [15]:

```
House_train_data["POSTED_BY"] = House_train_data["POSTED_BY"].replace(to_replace=['Owne
House_test_data["POSTED_BY"]  = House_test_data["POSTED_BY"].replace(to_replace=['Owner
```

In [16]:

```
print("Total no of unique value available in UNDER_CONSTRUCTION =" ,House_train_data["U
print("Uniques Data that are available in UNDER_CONSTRUCTION are = ",House_train_data['
print("Number of entries of Unique value available in UNDER_CONSTRUCTION =" ,House_trai
type(House_train_data["UNDER_CONSTRUCTION"])
```

```
Total no of unique value available in UNDER_CONSTRUCTION = 2
Uniques Data that are available in UNDER_CONSTRUCTION are =  [0 1]
Number of entries of Unique value available in UNDER_CONSTRUCTION = 0    241
57
1     5294
Name: UNDER_CONSTRUCTION, dtype: int64
```

Out[16]:

```
pandas.core.series.Series
```

In [17]:

```
print("Total no of unique value available in RERA =" ,House_train_data["RERA"].nunique(
print("Uniques Data that are available in RERA are = ",House_train_data["RERA"].unique(
print("Number of entries of Unique value available in RERA =" ,House_train_data["RERA"]
type(House_train_data["RERA"])
```

```
Total no of unique value available in RERA = 2
Uniques Data that are available in RERA are =  [0 1]
Number of entries of Unique value available in RERA = 0    20088
1     9363
Name: RERA, dtype: int64
```

Out[17]:

```
pandas.core.series.Series
```

In [18]:

```
print("Total no of unique value available in BHK_NO.  =" ,House_train_data["BHK_NO."].r
print("Uniques Data that are available in BHK_NO. are = ",House_train_data["BHK_NO."].u
print("Number of entries of Unique value available in BHK_NO. =" ,House_train_data["BHK
type(House_train_data["BHK_NO."])
```

```
Total no of unique value available in BHK_NO.  = 16
Uniques Data that are available in BHK_NO. are =  [ 2  3  1  4  5  6 12  8 2
0 10  7  9 13 17 15 11]
Number of entries of Unique value available in BHK_NO. = 2     13324
3     10546
1      3574
4      1723
5       190
6        52
7        11
8        10
20        4
10        4
15        4
9         3
12        3
17        1
11        1
13        1
Name: BHK_NO., dtype: int64
```

Out[18]:

```
pandas.core.series.Series
```

In [19]:

```
print("Total no of unique value available in BHK_OR_RK  =" ,House_train_data["BHK_OR_RK
print("Uniques Data that are available in BHK_OR_RK are = ",House_train_data["BHK_OR_RK
print("Number of entries of Unique value available in BHK_OR_RK =" ,House_train_data["E
type(House_train_data["BHK_OR_RK"])
```

```
Total no of unique value available in BHK_OR_RK  = 2
Uniques Data that are available in BHK_OR_RK are =  ['BHK' 'RK']
Number of entries of Unique value available in BHK_OR_RK = BHK    29427
RK        24
Name: BHK_OR_RK, dtype: int64
```

Out[19]:

```
pandas.core.series.Series
```

Conversion of the Unique data into binary type for the required calculations

In [20]:

```
House_train_data["BHK_OR_RK"] = House_train_data["BHK_OR_RK"].replace(to_replace=('BHK'
```

In [21]:

```
print("Total no of unique value available in BHK_OR_RK  =" ,House_train_data["BHK_OR_RK
print("Uniques Data are that available in BHK_OR_RK are = ",House_train_data["BHK_OR_RK
print("Number of entries of Unique value available in BHK_OR_RK =" ,House_train_data["B
type(House_train_data["BHK_OR_RK"])
```

```
Total no of unique value available in BHK_OR_RK  = 2
Uniques Data are that available in BHK_OR_RK are =  [1 0]
Number of entries of Unique value available in BHK_OR_RK = 1    29427
0       24
Name: BHK_OR_RK, dtype: int64
```

Out[21]:

```
pandas.core.series.Series
```

In [22]:

```
print("Total no of unique value available in SQUARE_FT  =" ,House_train_data["SQUARE_F
print("Uniques Data that are available in SQUARE_FT are = ",House_train_data["SQUARE_F
print("Number of entries of Unique value available in SQUARE_FT =" ,House_train_data["S
type(House_train_data["SQUARE_FT"])
```

```
Total no of unique value available in SQUARE_FT  = 19561
Uniques Data that are available in SQUARE_FT are =  [1300.236407  1275.
933.1597222 ... 1022.641509   927.0790093
  896.7741935]
Number of entries of Unique value available in SQUARE_FT = 1000.000000    47
9
1250.000000    294
800.000000     202
1200.000000    179
1600.000000    125
              ...
2200.303490      1
1170.311056      1
1360.092884      1
1532.097441      1
2090.492554      1
Name: SQUARE_FT, Length: 19561, dtype: int64
```

Out[22]:

```
pandas.core.series.Series
```

In [23]:

```
print("Total no of unique value available in READY_TO_MOVE  =" ,House_train_data["READY
print("Uniques Data that are available in READY_TO_MOVE are = ",House_train_data["READY
print("Number of entries of Unique value available in READY_TO_MOVE =" ,House_train_dat
type(House_train_data["READY_TO_MOVE"])
```

```
Total no of unique value available in READY_TO_MOVE  = 2
Uniques Data that are available in READY_TO_MOVE are =  [1 0]
Number of entries of Unique value available in READY_TO_MOVE = 1    24157
0     5294
Name: READY_TO_MOVE, dtype: int64
```

Out[23]:

```
pandas.core.series.Series
```

In [24]:

```
print("Total no of unique value available in RESALE  =" ,House_train_data["RESALE"].nur
print("Uniques Data that are available in RESALE are = ",House_train_data["RESALE"].uni
print("Number of entries of Unique value available in RESALE =" ,House_train_data["RESA

```

```
Total no of unique value available in RESALE  = 2
Uniques Data that are available in RESALE are =  [1 0]
Number of entries of Unique value available in RESALE = 1    27377
0     2074
Name: RESALE, dtype: int64
```

In [25]:

```
print("Total no of unique value available in ADDRESS  =" ,House_train_data["ADDRESS"].r
print("Uniques Data that are available in ADDRESS are = ",House_train_data["ADDRESS"].u
print("Number of entries of Unique value available in ADDRESS =" ,House_train_data["ADI
type(House_train_data["ADDRESS"])
```

```
Total no of unique value available in ADDRESS  = 6899
Uniques Data that are available in ADDRESS are =  ['Ksfc Layout,Bangalore'
'Vishweshwara Nagar,Mysore' 'Jigani,Bangalore'
 ... 'west mambalam,Chennai' 'Gandhi Nagar,Gulbarga'
 'E3-108, Lake View Recidency,,Vapi']
Number of entries of Unique value available in ADDRESS = Zirakpur,Chandigarh
509
Whitefield,Bangalore          230
Raj Nagar Extension,Ghaziabad    215
Sector-137 Noida,Noida        139
New Town,Kolkata              131
                             ...
Bhangel,Noida                   1
Bakul Bagan,Kolkata             1
shanti nagar,Rajkot             1
Lawyer Pet,Ongole               1
Fatima Nagar,Pune               1
Name: ADDRESS, Length: 6899, dtype: int64
```

Out[25]:

```
pandas.core.series.Series
```

In [26]:

```
print("Total no of Unique value available in LONGITUDE  =" ,House_train_data["LONGITUDE
print("Uniques Data that are available in LONGITUDE are = ",House_train_data["LONGITUDE
print("Number of entries of Unique value available in LONGITUDE =" ,House_train_data["L
type(House_train_data["LONGITUDE"])
```

```
Total no of Unique value available in LONGITUDE  = 4087
Uniques Data that are available in LONGITUDE are =  [12.96991  12.274538 12.
778033 ... 18.9737   17.357159 39.945409]
Number of entries of Unique value available in LONGITUDE = 24.690280    1009
12.969910     671
30.662283     509
22.541110     479
19.058710     242
             ...
25.606371       1
20.802556       1
21.132949       1
21.183330       1
27.846364       1
Name: LONGITUDE, Length: 4087, dtype: int64
```

Out[26]:

```
pandas.core.series.Series
```

In [27]:

```
print("Total no of Unique value available in LATITUDE  =" ,House_train_data["LATITUDE"]
print("Uniques Data that are available in LATITUDE are = ",House_train_data["LATITUDE"]
print("Number of entries of Unique value available in LATITUDE =" ,House_train_data["LA
type(House_train_data["LATITUDE"])
```

```
Total no of Unique value available in LATITUDE  = 4078
Uniques Data that are available in LATITUDE are =  [ 77.59796   76.644605  7
7.632191 ...  73.3321    76.841908 -86.150721]
Number of entries of Unique value available in LATITUDE = 78.418890    1009
77.597960     671
76.822397     509
88.337780     479
72.899690     242
             ...
77.008060       1
77.501208       1
83.020674       1
80.189850       1
69.658066       1
Name: LATITUDE, Length: 4078, dtype: int64
```

Out[27]:

```
pandas.core.series.Series
```

In [28]:

```
print("Total no of Unique value available in TARGET(PRICE_IN_LACS)  =" ,House_train_dat
print("Uniques Data that are available in TARGET(PRICE_IN_LACS) are = ",House_train_dat
print("Number of entries of Unique value available in TARGET(PRICE_IN_LACS) =" ,House_t
type(House_train_data["TARGET(PRICE_IN_LACS)"])
```

```
Total no of Unique value available in TARGET(PRICE_IN_LACS)  = 1172
Uniques Data that are available in TARGET(PRICE_IN_LACS) are =  [  55.    5
1.    43.   ... 1170.  8660.    18.3]
Number of entries of Unique value available in TARGET(PRICE_IN_LACS) = 110.0
795
100.0     770
120.0     652
130.0     598
45.0      583
         ...
1890.0      1
248.0       1
1.5         1
7390.0      1
810.0       1
Name: TARGET(PRICE_IN_LACS), Length: 1172, dtype: int64
```

Out[28]:

```
pandas.core.series.Series
```

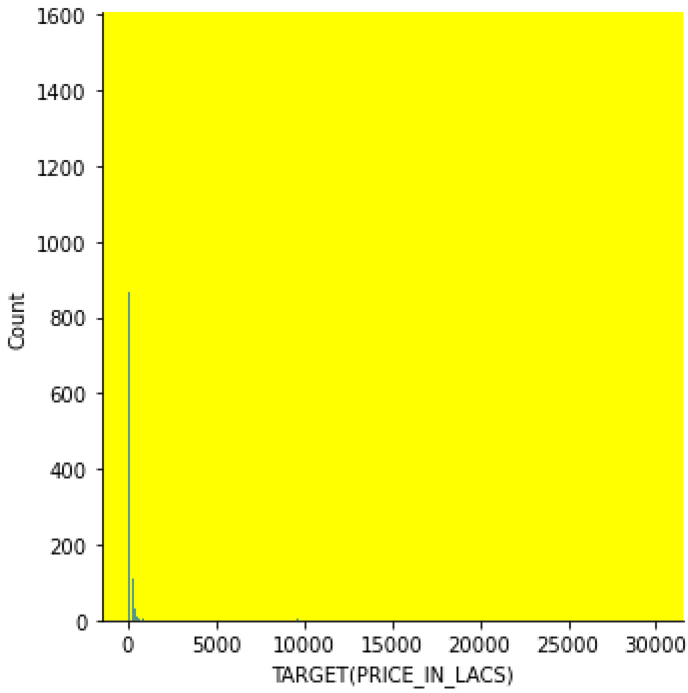
STEP 3 : DATA VISUALISATION

In [29]:

```
# Checking of outliers in TARGET(PRICE_in_lACS) of Train Data Set Column
plt.rcParams['axes.facecolor'] = "yellow"
sns.displot(House_train_data["TARGET(PRICE_IN_LACS)"])
```

Out[29]:

<seaborn.axisgrid.FacetGrid at 0x26c000dd190>

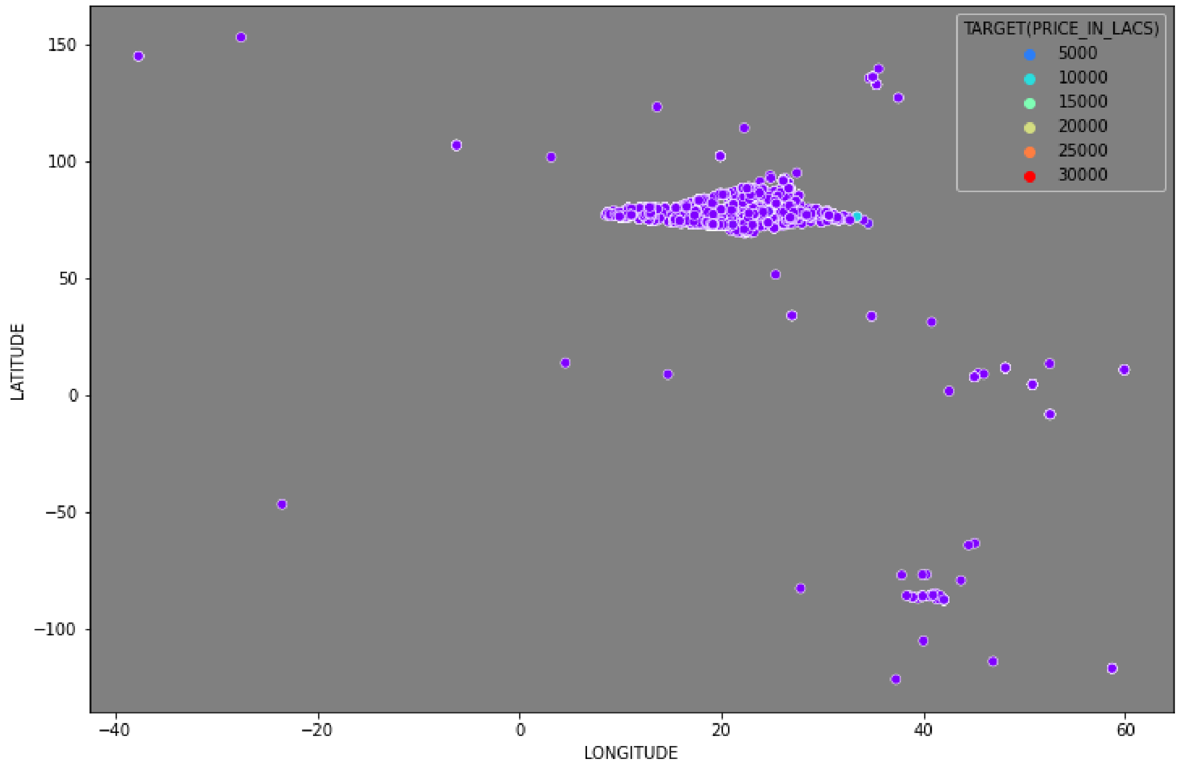


In [30]:

```
# Scatter Plot between Location (Latitude,Longitude) and Prices of the property to chec
plt.figure(figsize=(12,8))
plt.rcParams['axes.facecolor'] = "gray"
sns.scatterplot(x="LONGITUDE", y="LATITUDE", data=House_train_data, hue="TARGET(PRICE_I
```

Out[30]:

```
<AxesSubplot:xlabel='LONGITUDE', ylabel='LATITUDE'>
```

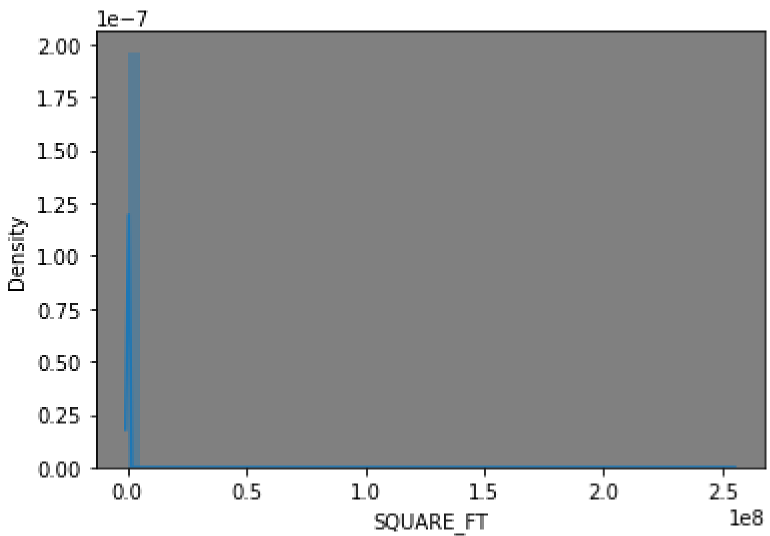


In [31]:

```
#Checking the outliers in square foot column
sns.distplot(House_train_data['SQUARE_FT'])
```

Out[31]:

```
<AxesSubplot:xlabel='SQUARE_FT', ylabel='Density'>
```



Renaming the last column for our assistance

In [32]:

```
House_train_data = House_train_data.rename(columns = {'TARGET(PRICE_IN_LACS)' : 'TARGE
```

Removing the Outliers from Train Data

Detecting outliers using quantile ranges

What Are Quantiles?

Quantiles are very easy to understand. Let’s say we have a series of 20 numbers. We can sort the numbers from lowest to highest. We can then group these points into quantiles, which are identified by cut points in the sorted data that describe the point below which X% of data falls.

Note that quantiles are generally expressed as a fraction (from 0 to 1). They correspond exactly to percentiles, which range from 0 to 100.

In [33]:

```
Quant1= House_train_data.quantile(.25)
Quant2= House_train_data.median()
Quant3= House_train_data.quantile(.80)
IQR = Quant3 - Quant1
QMIN = Quant1 - 1.5*IQR
QMAX = Quant3 + 1.5*IQR
```

In [34]:

```
print("Quantile 1 = \n",Quant1)
print("\n\nQuantile 2 = \n ",Quant2)
print("\n\nQuantile 3 = \n",Quant3)
print("\n\nInter Quantile range = \n",IQR)
print("\n\nQ-MIN = \n",QMIN)
print("\n\nQ-MAX = \n",QMAX)
```

```
Quantile 1 = 
 POSTED_BY                0.000000
UNDER_CONSTRUCTION       0.000000
RERA                     0.000000
BHK_NO.                  2.000000
BHK_OR_RK                1.000000
SQUARE_FT              900.021130
READY_TO_MOVE            1.000000
RESALE                   1.000000
LONGITUDE               18.452663
LATITUDE                73.798100
TARGET                  38.000000
Name: 0.25, dtype: float64


Quantile 2 = 
  POSTED_BY                 1.000000
UNDER_CONSTRUCTION        0.000000
RERA                      0.000000
BHK_NO.                   2.000000
BHK_OR_RK                 1.000000
SQUARE_FT              1175.056750
READY_TO_MOVE             1.000000
RESALE                    1.000000
LONGITUDE                20.750000
LATITUDE                 77.324137
TARGET                   62.000000
dtype: float64


Quantile 3 = 
 POSTED_BY                 1.000000
UNDER_CONSTRUCTION        0.000000
RERA                      1.000000
BHK_NO.                   3.000000
BHK_OR_RK                 1.000000
SQUARE_FT              1665.080875
READY_TO_MOVE             1.000000
RESALE                    1.000000
LONGITUDE                28.396092
LATITUDE                 78.436017
TARGET                  120.000000
Name: 0.8, dtype: float64


Inter Quantile range = 
 POSTED_BY                1.000000
UNDER_CONSTRUCTION       0.000000
RERA                     1.000000
BHK_NO.                  1.000000
BHK_OR_RK                0.000000
SQUARE_FT              765.059745
```

```
READY_TO_MOVE          0.000000
RESALE                 0.000000
LONGITUDE              9.943429
LATITUDE               4.637917
TARGET                82.000000
dtype: float64


Q-MIN =
 POSTED_BY             -1.500000
UNDER_CONSTRUCTION     0.000000
RERA                  -1.500000
BHK_NO.                0.500000
BHK_OR_RK              1.000000
SQUARE_FT           -247.568489
READY_TO_MOVE          1.000000
RESALE                 1.000000
LONGITUDE              3.537520
LATITUDE              66.841225
TARGET               -85.000000
dtype: float64


Q-MAX =
 POSTED_BY                2.500000
UNDER_CONSTRUCTION       0.000000
RERA                     2.500000
BHK_NO.                  4.500000
BHK_OR_RK                1.000000
SQUARE_FT             2812.670493
READY_TO_MOVE            1.000000
RESALE                   1.000000
LONGITUDE               43.311235
LATITUDE                85.392893
TARGET                 243.000000
dtype: float64
```

-Quartiles, which divide the data into 25% groups. The first quartile represents the data points that fall in the lowest 25%, the second quartile points fall between 25% and 50%, and so forth.

-Interquartile range, or IQR, which defines the range covered by 2nd and 3rd quartiles.

In [35]:

```
House_train_data.head()
```

Out[35]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	BHK_OR_RK	SQUARE_FT	READY
0	0	0	0	2	1	1300.236407	
1	1	0	0	2	1	1275.000000	
2	0	0	0	2	1	933.159722	
3	0	0	1	2	1	929.921143	
4	1	1	0	2	1	999.009247	

In [ ]:

```

```

In [36]:

```
House_train_data = House_train_data[(House_train_data.TARGET>QMIN.TARGET)&(House_train_
```

In [37]:

```
House_train_data = House_train_data[(House_train_data.SQUARE_FT>QMIN.SQUARE_FT)&(House_
```

In [38]:

```
House_train_data = House_train_data[(House_train_data.LATITUDE>QMIN.LATITUDE)&(House_tr
```

In [39]:

```
House_train_data = House_train_data[(House_train_data.LONGITUDE>QMIN.LONGITUDE)&(House_
```

In [40]:

```
House_train_data.shape
```

Out[40]:

```
(24250, 12)
```

Analysing the data through data Visualisation after the removal of outliers

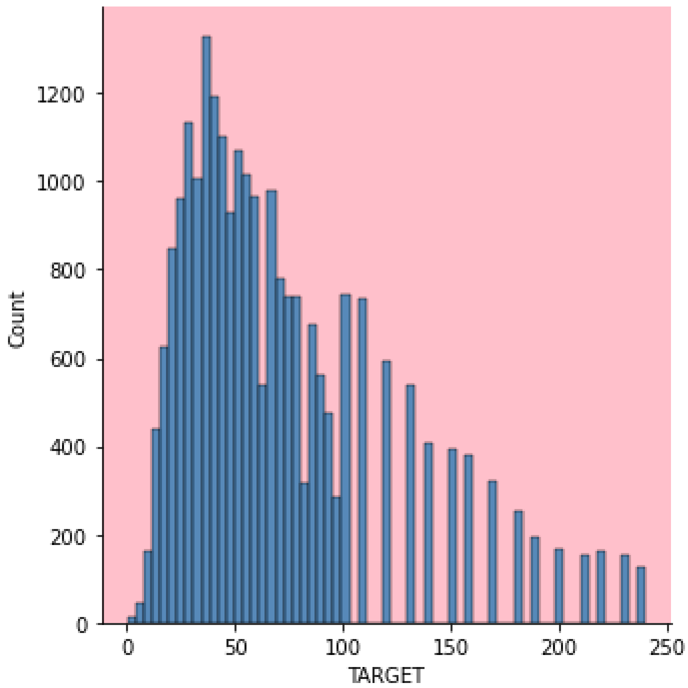
Distribution Plot

In [41]:

```
plt.rcParams['axes.facecolor'] = "pink"
sns.displot(House_train_data["TARGET"])
```

Out[41]:

```
<seaborn.axisgrid.FacetGrid at 0x26c061dd0d0>
```



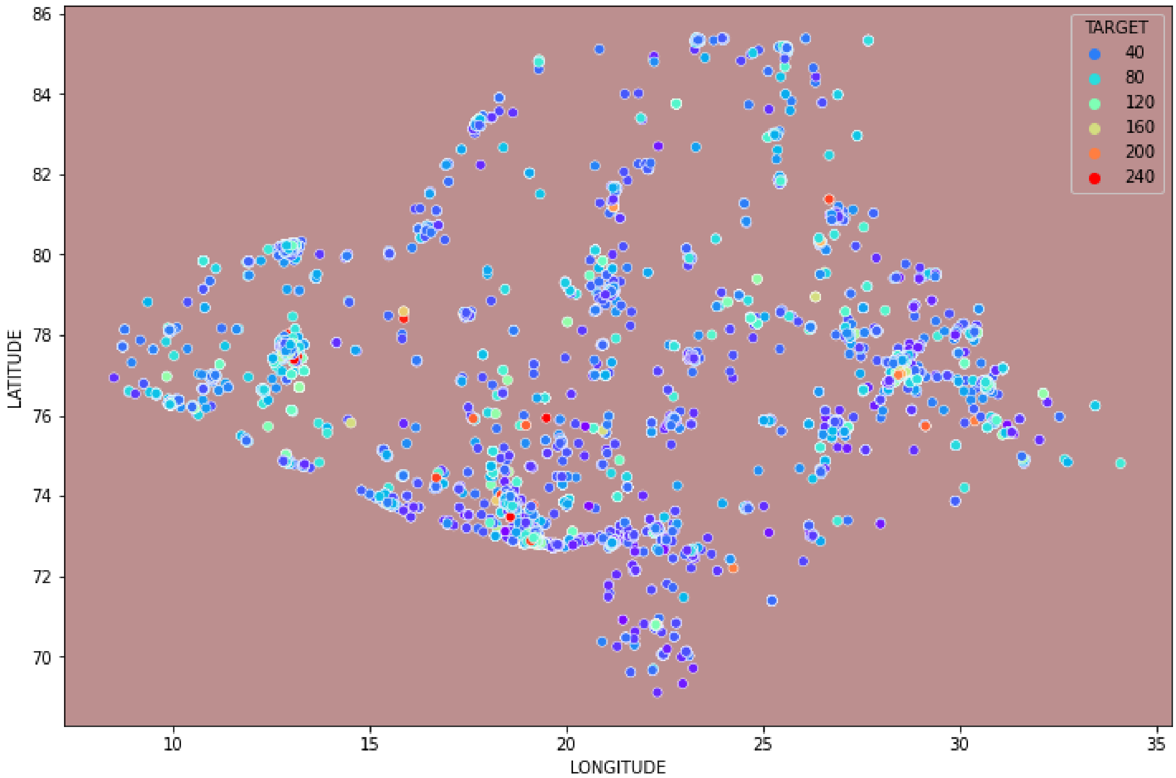
Scatter Plot

In [42]:

```
plt.figure(figsize=(12,8))
plt.rcParams['axes.facecolor'] = "RosyBrown"
sns.scatterplot(x="LONGITUDE", y="LATITUDE", data=House_train_data, hue="TARGET",palett
```

Out[42]:

<AxesSubplot:xlabel='LONGITUDE', ylabel='LATITUDE'>



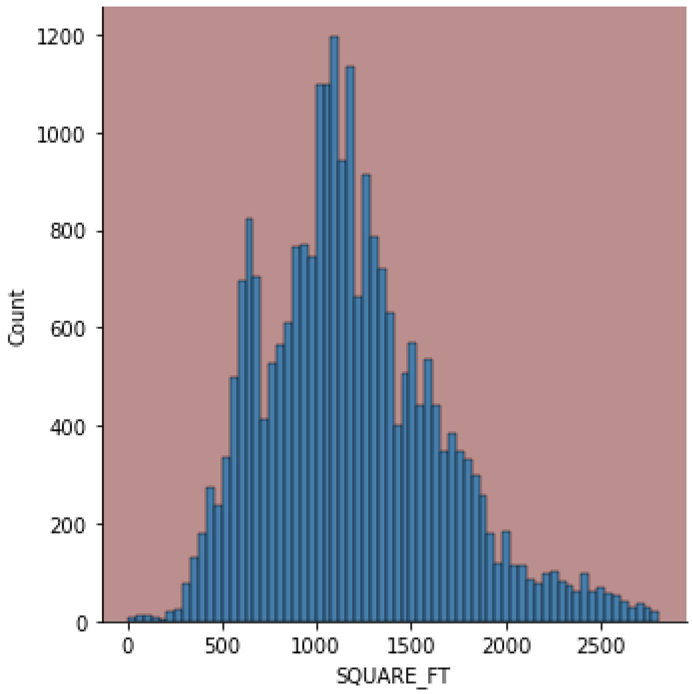
Distribution Plot

In [43]:

```
sns.displot(House_train_data['SQUARE_FT'])
```

Out[43]:

```
<seaborn.axisgrid.FacetGrid at 0x26c055abdc0>
```



Heat Map

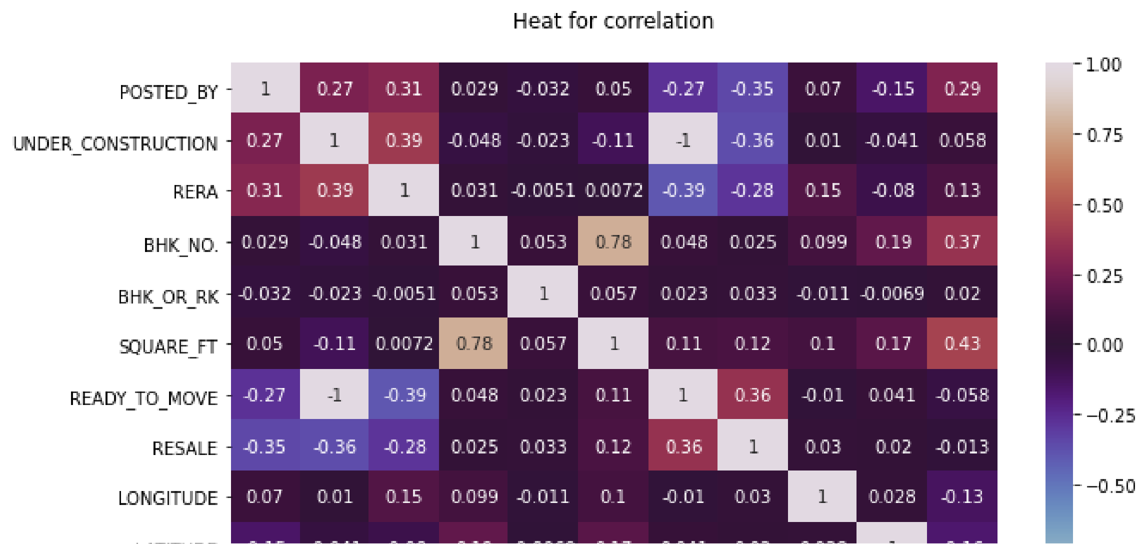
Heatmaps are very useful to find relations between two variables in a dataset. Heatmap can be easily produced using the ‘heatmap’ function provided by the seaborn package in python.

In [44]:

```
plt.figure(figsize = (10,6))
sns.heatmap(House_train_data.corr(), annot = True ,cmap='twilight')
plt.title(' Heat for correlation \n')
plt.show
```

Out[44]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



- From the Heatmap graph we can clearly see that there is a strong negative relationship between 'READY_TO_MOVE' and 'UNDER_CONSTRUCTION' which leads to multicollinearity. So we will remove one of them.

1. Since we have location based on the Longitude and Lattitude i.e why ADDRESS will be a wastaged attribute so now we can remove ADDRESS attribute too.

2. From the Heatmap we can clearly see that attribute 'BHK_OR_RK' isn't affecting our Target price so there is not significant to take it as a parameter, so we will remove this also.

Now we will remove "READY_TO_MOVE","ADDRESS" and "BHK_OR_RK" from both Train and test data.

In [45]:

```
House_train_data = House_train_data.drop(['READY_TO_MOVE'],axis = 1, inplace=False)
House_train_data = House_train_data.drop(['ADDRESS'],axis = 1, inplace=False)
House_train_data = House_train_data.drop(['BHK_OR_RK'],axis = 1, inplace=False)

House_test_data = House_test_data.drop(['READY_TO_MOVE'],axis = 1, inplace=False)
House_test_data = House_test_data.drop(['ADDRESS'],axis = 1, inplace=False)
House_test_data = House_test_data.drop(['BHK_OR_RK'],axis = 1, inplace=False)
```

In [46]:

```
House_train_data.head()
```

Out[46]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	RESALE	LONGITUDE
0	0	0	0	2	1300.236407	1	12.96991(
1	1	0	0	2	1275.000000	1	12.27453(
2	0	0	0	2	933.159722	1	12.77803(
3	0	0	1	2	929.921143	1	28.64230(
5	0	0	0	3	1250.000000	1	10.03328(

In [47]:

```
House_test_data.head()
```

Out[47]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	RESALE	LONGITUDE
0	0	0	0	1	545.171340	1	21.26200(
1	1	1	1	2	800.000000	0	18.96611(
2	1	0	0	2	1257.096513	1	22.59220(
3	1	0	0	3	1400.329489	1	26.98830(
4	0	0	0	1	430.477830	1	22.70000(

In [48]:

```
House_train_data.describe()
```

Out[48]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	I
count	24250.000000	24250.000000	24250.000000	24250.000000	24250.000000	2425(
mean	0.641361	0.178887	0.340165	2.268000	1187.273433	(
std	0.524146	0.383265	0.473774	0.752229	467.515519	(
min	0.000000	0.000000	0.000000	1.000000	3.985594	(
25%	0.000000	0.000000	0.000000	2.000000	861.090454	1
50%	1.000000	0.000000	0.000000	2.000000	1135.070181	1
75%	1.000000	0.000000	1.000000	3.000000	1459.053123	1
max	2.000000	1.000000	1.000000	20.000000	2811.142346	1

In [49]:

```
House_test_data.describe()
```

Out[49]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	I
count	68720.000000	68720.000000	68720.000000	68720.000000	6.872000e+04	6872(
mean	0.657203	0.176557	0.316531	2.388198	2.762419e+03	(
std	0.514742	0.381296	0.465126	0.864577	1.640991e+05	(
min	0.000000	0.000000	0.000000	1.000000	1.000000e+00	(
25%	0.000000	0.000000	0.000000	2.000000	9.000310e+02	1
50%	1.000000	0.000000	0.000000	2.000000	1.174982e+03	1
75%	1.000000	0.000000	1.000000	3.000000	1.550265e+03	1
max	2.000000	1.000000	1.000000	31.000000	4.016393e+07	1

MODELING

Applying Linear Regression Model on the House Price Train data File

In [50]:

```
X_train = House_train_data.drop('TARGET', axis = 1).values
Y_train = House_train_data['TARGET'].values

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train , Y_test = train_test_split(X_train, Y_train ,test_size=0.1, r
```

In [51]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [52]:

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

In [53]:

```
# Importing Linear Regression
from sklearn.linear_model import LinearRegression
LR = LinearRegression()
```

In [54]:

```
LR.fit(X_train,Y_train)
```

Out[54]:

```
LinearRegression()
```

In [55]:

```
print("The intercept value =", LR.intercept_)
```

```
The intercept value = 34.01436223143887
```

In [56]:

```
print("The coeffiecients are Theat 0 ,Theta 1, Theta 2 , Theta 3 =", LR.coef_)
```

```
The coeffiecients are Theat 0 ,Theta 1, Theta 2 , Theta 3 = [ 44.73736782
3.4711549    6.66356155 148.34717312 105.94836093
  12.6220286  -40.8688877  -59.32815402]
```

In [57]:

```
Predict = LR.predict(X_test)
```

In [58]:

```
Predict
```

Out[58]:

```
array([110.61848344,  70.49758077,  53.51686777, ..., 120.20529015,
       116.34288211, 104.43078494])
```

In [59]:

```
from sklearn.metrics import r2_score
```

In [60]:

```
r2_score(Y_test,Predict)
```

Out[60]:

```
0.035235387013067165
```

In [61]:

```
from sklearn.metrics import mean_absolute_error, max_error, mean_squared_error,r2_score
```

In [62]:

```
Y_predict = LR.predict(X_test).reshape(X_test.shape[0])


predict_df = pd.DataFrame({ 'Actual_value': Y_test, 'Predicted value': Y_predict})
print(predict_df)
```

```
      Actual_value  Predicted value
0            110.0       110.618483
1             52.0        70.497581
2             28.0        53.516868
3            150.0       127.908138
4             90.0       142.658157
...            ...              ...
2420         140.0       135.708165
2421          92.0       115.540633
2422          59.8       120.205290
2423          95.0       116.342882
2424          30.0       104.430785

[2425 rows x 2 columns]
```

MAE

In [63]:

```
mean_absolute_error(y_true=predict_df['Actual_value'], y_pred=predict_df['Predicted va
```

Out[63]:

```
39.63279287283993
```

MSE

In [64]:

```
print(mean_squared_error(y_true=predict_df['Actual_value'], y_pred=predict_df['Predicte
```

```
2303.613972229608
```

RMSE

In [65]:

```
print(np.sqrt(mean_squared_error(y_true=predict_df['Actual_value'], y_pred=predict_df[
```

```
47.995978708946105
```

VARIANCE SCORE

In [66]:

```
from sklearn.metrics import explained_variance_score
explained_variance_score(Y_test,Predict)
```

Out[66]:

```
0.27416121499459556
```

R2_SCORE

In [67]:

```
R_Squared = r2_score(Y_test,Predict)
R_Squared
```

Out[67]:

```
0.035235387013067165
```

Now Applying Linear Regression Model on the House Price Test data File

In [68]:

```
House_test_data1 = House_test_data
```

In [69]:

```
Quant11 = House_test_data1.quantile(.35)
Quant21 = House_test_data1.median()
Quant31 = House_test_data1.quantile(.60)
IQR1 = Quant31 - Quant11
QMIN1 = Quant11 - 1.5*IQR1
QMAX1 = Quant31 + 1.5*IQR1
```

In [70]:

```
House_test_data1 = House_test_data1[(House_test_data1.SQUARE_FT>QMIN1.SQUARE_FT)&(House
House_test_data1.shape
```

Out[70]:

```
(51687, 8)
```

In [71]:

```
House_test_data1 = House_test_data1[(House_test_data1.LONGITUDE>QMIN1.LONGITUDE)&(House
House_test_data1.shape
```

Out[71]:

```
(30668, 8)
```

In [72]:

```
House_test_data1 = House_test_data1[(House_test_data1.LATITUDE>QMIN1.LATITUDE)&(House_t
House_test_data1.shape
```

Out[72]:

```
(12295, 8)
```

In [73]:

```
X2_test = House_test_data1.values
X2_test
```

Out[73]:

```
array([[ 1.       ,  0.       ,  0.       , ...,  1.       , 26.9883   ,
        75.5846   ],
       [ 1.       ,  0.       ,  0.       , ...,  1.       , 19.032025,
        73.621535],
       [ 1.       ,  0.       ,  0.       , ...,  1.       , 24.69028 ,
        78.41889 ],
       ...,
       [ 1.       ,  0.       ,  0.       , ...,  1.       , 18.510861,
        73.926175],
       [ 1.       ,  0.       ,  0.       , ...,  1.       , 19.085225,
        73.661835],
       [ 1.       ,  0.       ,  0.       , ...,  1.       , 18.49667 ,
        73.94167 ]])
```

In [74]:

```
X2_test = scaler.transform(X2_test)
```

In [75]:

```
Pred1 = LR.predict(X2_test)
```

In [76]:

```
Pred1
```

Out[76]:

```
array([125.00098291,  60.97777236,  73.78911655, ...,  97.70006978,
        98.65232479, 107.52692137])
```

In [77]:

```
Testdata = pd.DataFrame({'Target':Pred1})
Testdata
```

Out[77]:

	Target
0	125.000983
1	60.977772
2	73.789117
3	85.507010
4	103.142295
...	...
12290	103.632541
12291	133.468203
12292	97.700070
12293	98.652325
12294	107.526921

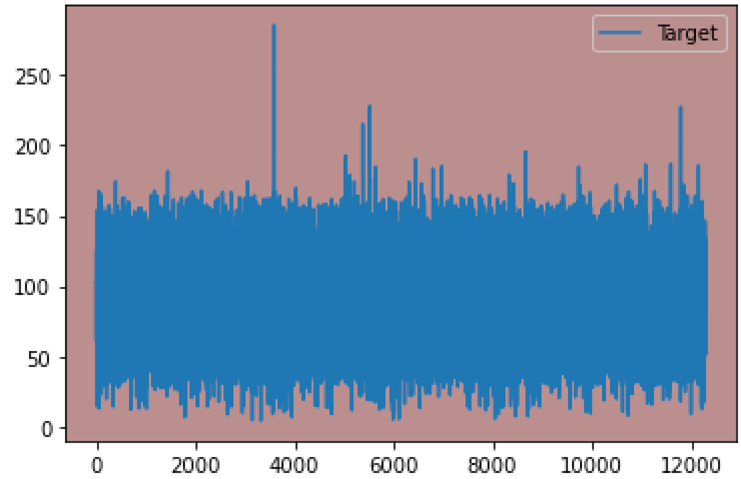
12295 rows × 1 columns

In [78]:

```
Testdata.plot()
```

Out[78]:

<AxesSubplot:>



Applying next model i.e Decision Tree on the House Price Train data File

In [79]:

```
#Importing Decision tree
from sklearn.tree import DecisionTreeRegressor
```

In [80]:

```
decision_tree_model= DecisionTreeRegressor()
```

In [81]:

```
decision_tree_model.fit(X_train,Y_train)
y_prediction_tree=decision_tree_model.predict(X_train)
```

In [82]:

```
y_prediction_tree_test=decision_tree_model.predict(X_test)
y_prediction_tree_test
```

Out[82]:

```
array([76. , 42. , 30.7, ..., 30. , 73. , 35. ])
```

RMSE and R2_square

In [83]:

```
print("RMSE:",np.sqrt(mean_squared_error(Y_train,y_prediction_tree)))
print("R_square:",r2_score(Y_train,y_prediction_tree))
```

```
RMSE: 1.942251854665174
R_square: 0.9983501066861239
```

Now Applying Decision Tree Model on the House Price Test data File

In [84]:

```
Pred2 = decision_tree_model.predict(X2_test)
```

In [85]:

```
Pred2
```

Out[85]:

```
array([150.,  45.,  86., ...,  25.,  25.,  58.])
```

In [86]:

```
Testdata1 = pd.DataFrame({'Target1':Pred2})
Testdata1
```

Out[86]:

	Target1
0	150.0
1	45.0
2	86.0
3	32.0
4	55.0
...	...
12290	25.0
12291	26.0
12292	25.0
12293	25.0
12294	58.0

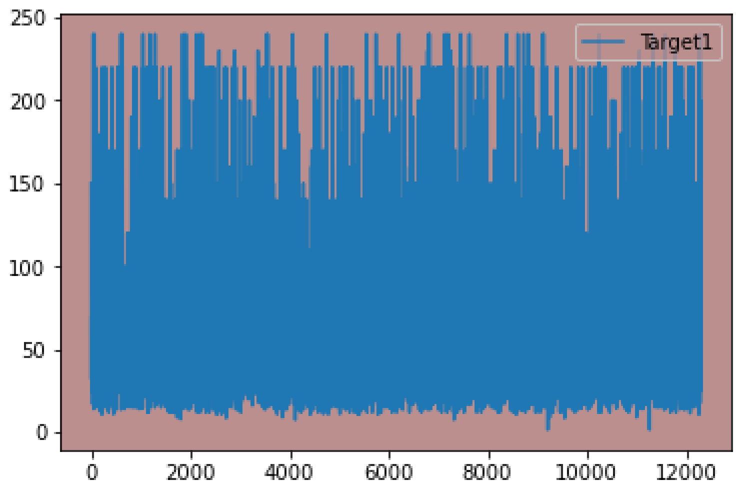
12295 rows × 1 columns

In [87]:

```
Testdata1.plot()
```

Out[87]:

<AxesSubplot:>



Model Evaluation

In [88]:

```
print("Linear Regression - R_Square: ",r2_score(Y_test,Predict))
print("Decison Tree - R_Square:",r2_score(Y_train,y_prediction_tree))
```

```
Linear Regression - R_Square:  0.035235387013067165
Decison Tree - R_Square: 0.9983501066861239
```

In [89]:

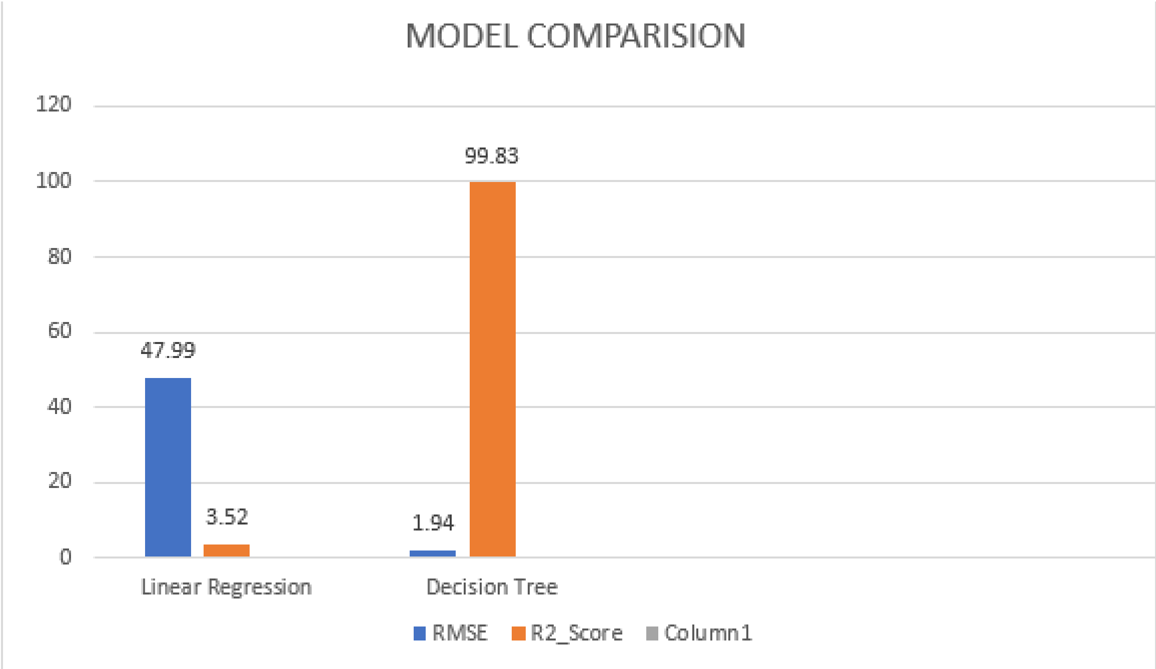
```
print("Linear Regression - RMSE",np.sqrt(mean_squared_error(y_true=predict_df['Actual_\
print("Decison Tree - RMSE:",np.sqrt(mean_squared_error(Y_train,y_prediction_tree)))
```

```
Linear Regression - RMSE 47.995978708946105
Decison Tree - RMSE: 1.942251854665174
```

In [90]:

```
display.Image("Compare.png")
```

Out[90]:



Conclusion

After going through a bunch of processes, we have successfully built and evaluated on two models that is linear regression model and Decision Tree in python also, choosing the best model for our given dataset. But, this won't stop here. Each and every model we built have their own statistical and mathematical concepts.

R-Squared is measurement of how well the dependent variable explains the variance of the independent variable. It is most popular evaluation metric for regression models. The ideal 'r2_score' of a model should be more than 0.70 (at least greater than 0.60).

We can see that, our first model i.e. Linear Regression by rounding the output values will result in a score of 0.035, similarly while rounding the outout values for Decision Tree Model will result in a score of 0.998, which means our Decison Tree model performs well on our dataset as comparison to Linear Regression.

Comparing RMSE, the model which is having the less RMSE, will be the best performing model. In our case, Linear Regression by rounding the output values will give 47.99 while the Decision Tree will give RMSE of 1.94. This implies Decision Tree performs better in this case as well.

Coming to the case of choosing the best model, Decision Tree is the best performer and the Linear Regression is the worst performer.