Sri Sathya Sai Institute of Higher Learning

(Deemed to be University)

MDSC 106 Project

House Price Prediction

S Sai Siddhanth

21238

Contents

Introduction

Introduction

Thousands of houses are sold every day. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price? In this paper, a machine learning model is proposed to predict a house price based on data related to the house (its size, the year it was built in, etc.). In this project, Python programming language with a number of Python packages will be used. In this project I have attempted to predict the price of a house. I hope I achieve this by the end of this project. So let’s start.

Data description

Data description

We see that the given data consists of 29451 observations × 12 features.

The features are defines as follows:

Feature	Description
POSTED_BY	Category marking who has listed the property
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

Possible values of each feature:

Posted by: Owner, Dealer and Builder

Under Construction: 0 –No, 1 – Yes

RERA: 0 – Not Approved, 1 – Approved

BHK No: Any positive whole number

BHK or RK: BHK – Bathroom Hall Kitchen and RK – Room Kitchen

Square feet – Area of the property

Ready to move – 0 No, 1 – Yes

Address – Any string

Longitude – Longitude coordinates

Latitude – Latitude coordinates

A small glance of the data

After loading the dataset to our python environment, let's see some entries in our dataset. For achieving this we use the head method, which would show us the first 5 observations of our dataset.

train.head()

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	BHK_OR_RK	SQUARE_FT	READY_TO_MOVE	RESALE	ADDRESS	LONGITUDE	LATITUDE	TARGET(PRICE_IN_LACS)
0	Owner	0	0	2	BHK	1300.236407	1	1	Ksfc Layout,Bangalore	12.969910	77.597960	55.0
1	Dealer	0	0	2	BHK	1275.000000	1	1	Vishweshwara Nagar,Mysore	12.274538	76.644605	51.0
2	Owner	0	0	2	BHK	933.159722	1	1	Jigani,Bangalore	12.778033	77.632191	43.0
3	Owner	0	1	2	BHK	929.921143	1	1	Sector-1 Vaishali,Ghaziabad	28.642300	77.344500	62.5
4	Dealer	1	0	2	BHK	999.009247	0	1	New Town,Kolkata	22.592200	88.484911	60.5

Data Information

With the use of info method we can see the information related to the features of our dataset, such as the name of the column, non null count and data type of the column.

train.info()

```
 #   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
```

Dataset description

Now, with the help of describe method, we get the statistical view of the data, such as count, mean, standard deviation, min, max etc.

train.describe()

	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	READY_TO_MOVE	RESALE	LONGITUDE	LATITUDE	TARGET(PRICE_IN_LAC
count	29451.000000	29451.000000	29451.000000	2.945100e+04	29451.000000	29451.000000	29451.000000	29451.000000	29451.0000
mean	0.179756	0.317918	2.392279	1.980217e+04	0.820244	0.929578	21.300255	76.837695	142.8987
std	0.383991	0.465675	0.879091	1.901335e+06	0.383991	0.255861	6.205306	10.557747	656.8807
min	0.000000	0.000000	1.000000	3.000000e+00	0.000000	0.000000	-37.713008	-121.761248	0.2500
25%	0.000000	0.000000	2.000000	9.000211e+02	1.000000	1.000000	18.452663	73.798100	38.0000
50%	0.000000	0.000000	2.000000	1.175057e+03	1.000000	1.000000	20.750000	77.324137	62.0000
75%	0.000000	1.000000	3.000000	1.550688e+03	1.000000	1.000000	26.900926	77.828740	100.0000
max	1.000000	1.000000	20.000000	2.545455e+08	1.000000	1.000000	59.912884	152.962676	30000.0000

Exploratory Data Analysis

Data Cleaning

Before proceeding to the exploratory data analysis (EDA), we check whether the dataset has any null value, if they are there we either drop the column using *dataframe.dropna* or the best way to deal with it is to take the mean of the column and add the value if the cell is null.

Train.isnull().sum()

```
POSTED_BY                0
UNDER_CONSTRUCTION       0
RERA                     0
BHK_NO.                  0
BHK_OR_RK                0
SQUARE_FT                0
READY_TO_MOVE            0
RESALE                   0
ADDRESS                  0
LONGITUDE                0
LATITUDE                 0
TARGET(PRICE_IN_LACS)    0
dtype: int64
```

Pre-processing Data

In order to understand the data and make the data crisp we add or modify features which may help us in the process of data modelling too.

In this project, I have added the following features which helps me understand the data and helps in prediction.

Feature	Description
CITY	City where the house is located. (From ADDRESS feature)
TOWN	Town where the house is located. ((From ADDRESS feature)
CITY_TIER	Depicts the City Tier

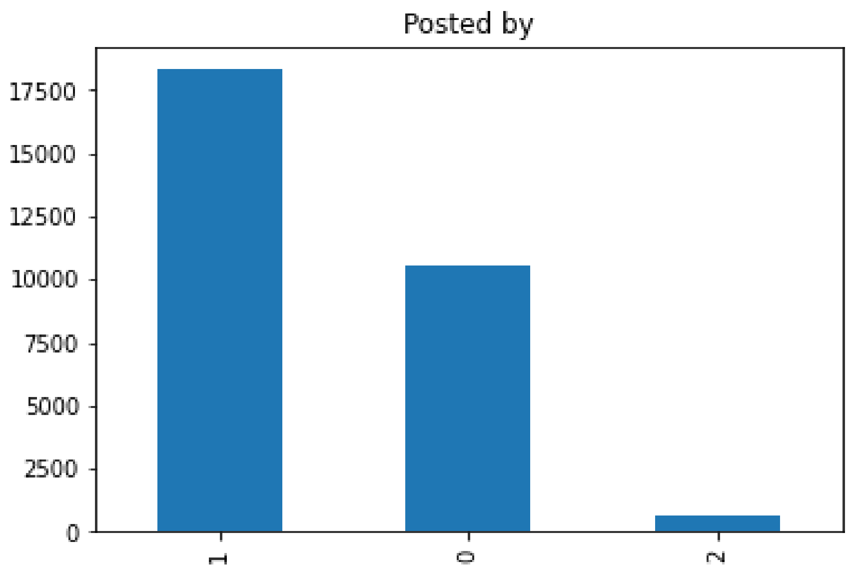
Data Visualization

Data visualization

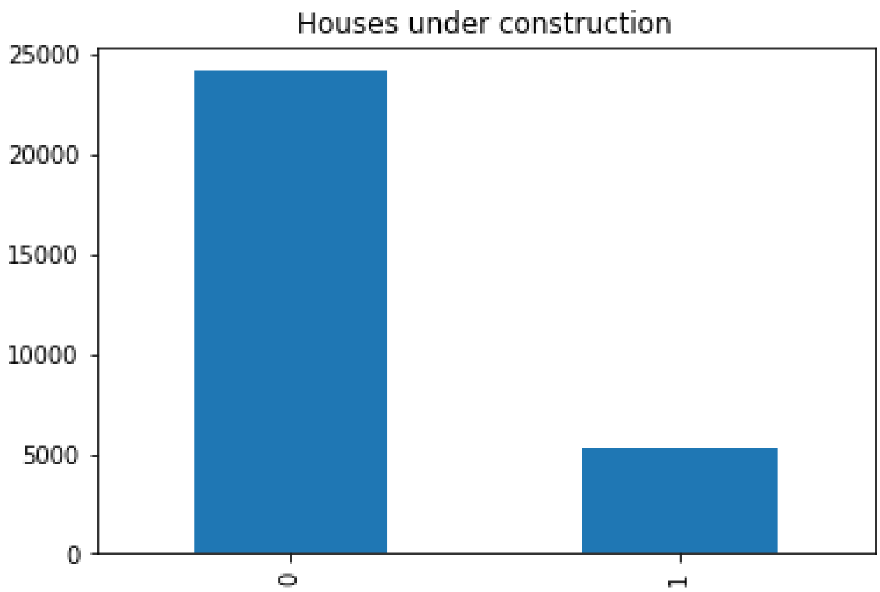
Let’s see some plots here.

Bar charts

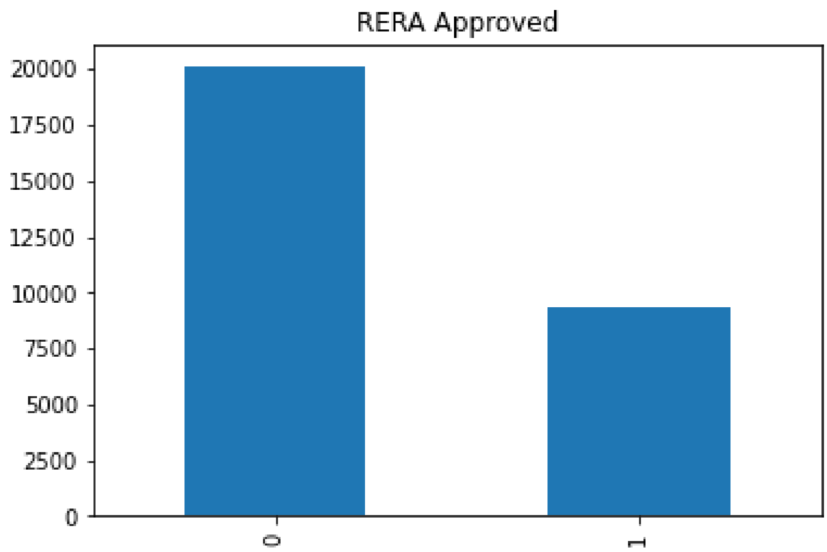
I have plotted bar charts for all the features with its value counts.



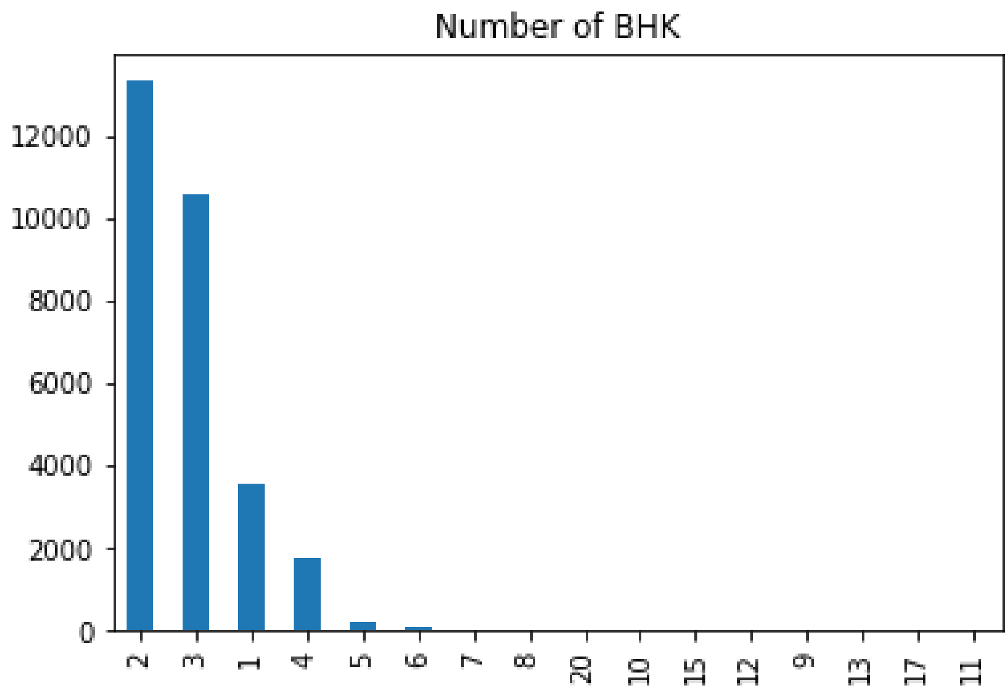
This bar plot shows us the value counts of houses posted by Owners, Dealers and Builder



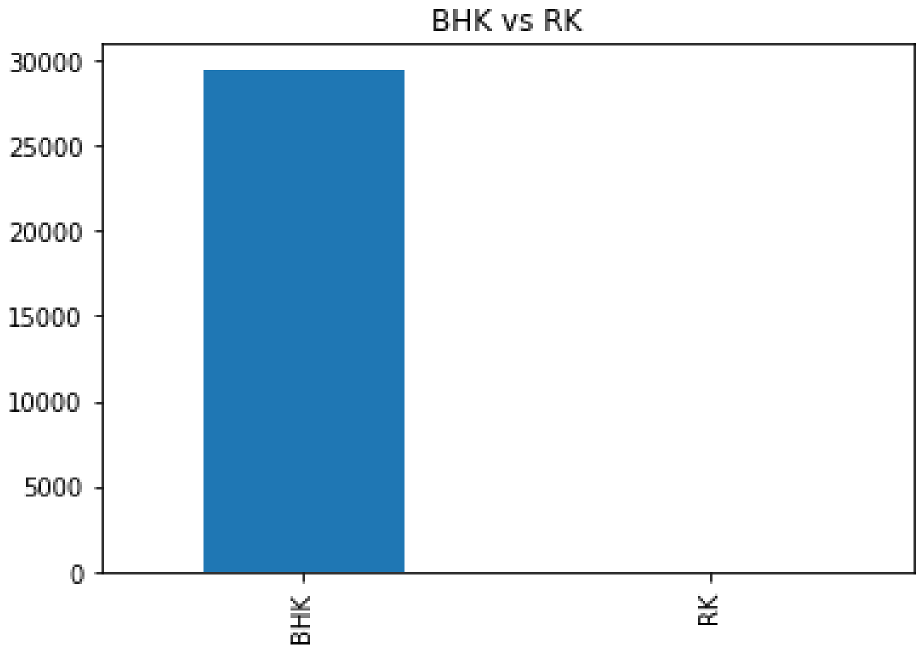
This bar plot shows us the value counts of houses under construction



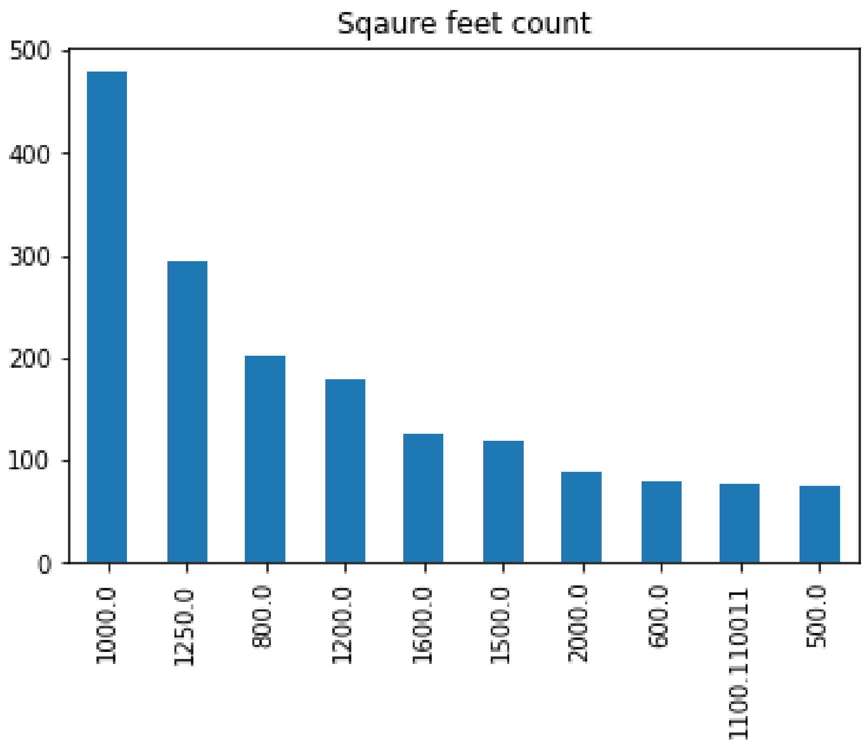
This bar plot shows us the value count of houses which have been approved by RERA



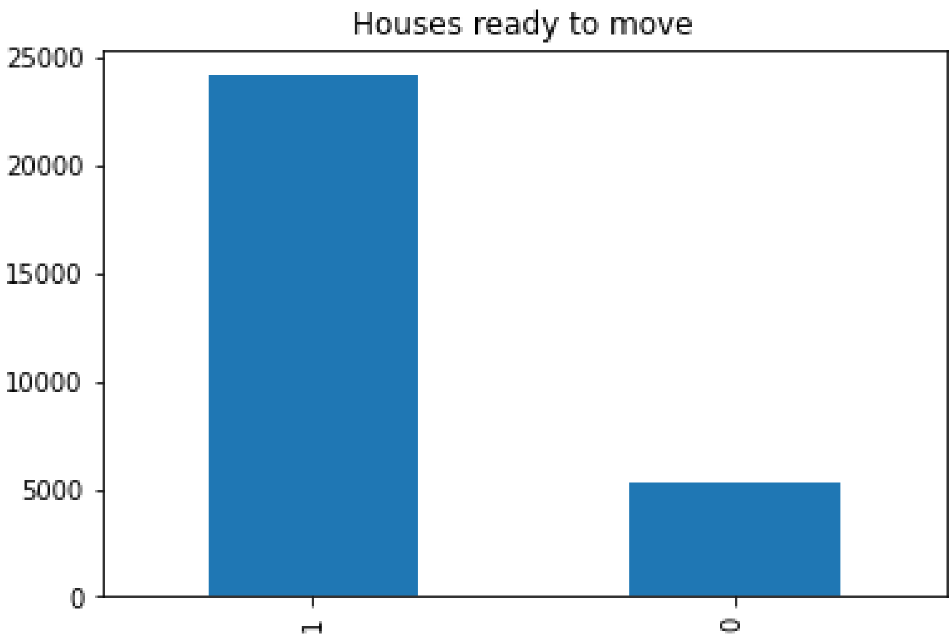
This bar plot shows us the value count of houses with respect to the number of BHK



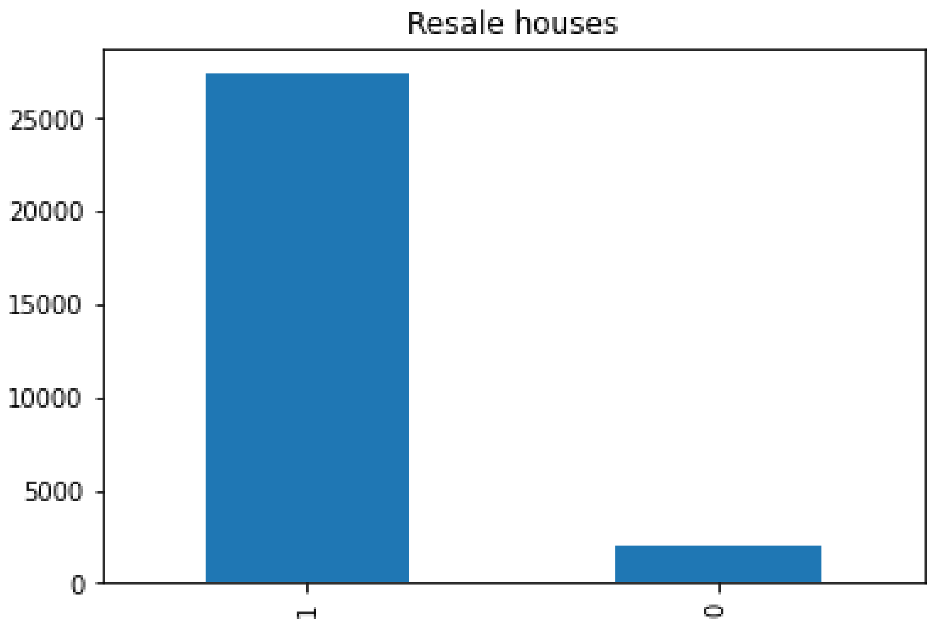
This bar plot shows us the value count of houses with BHK or RK(Room Kitchen)



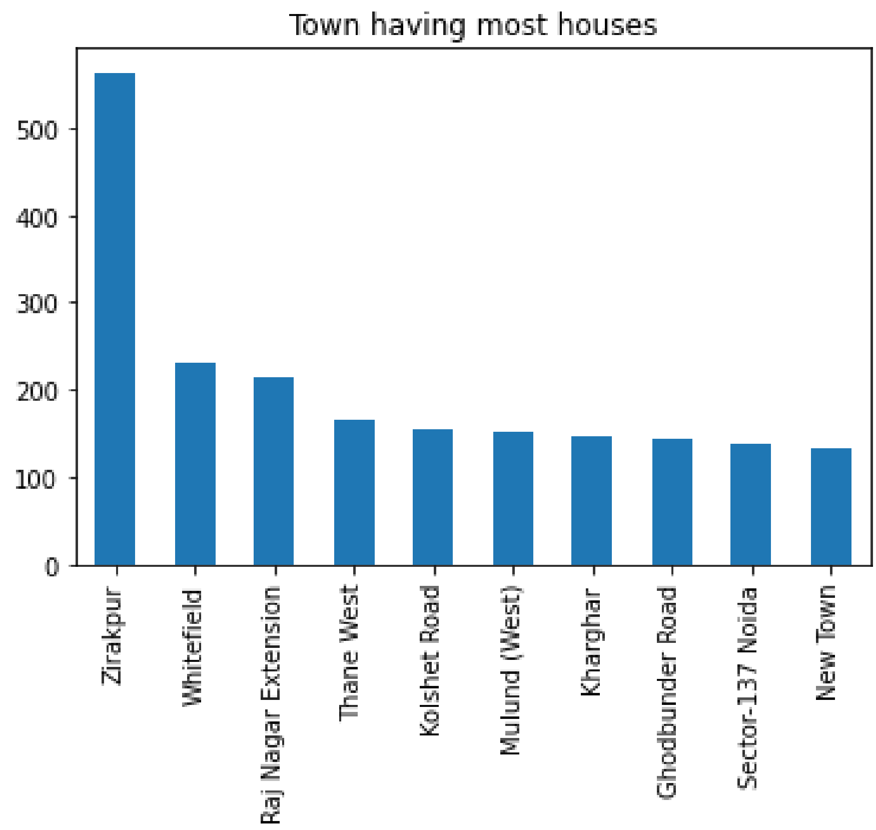
This bar plot shows us the value count of houses with respect to house's square feet



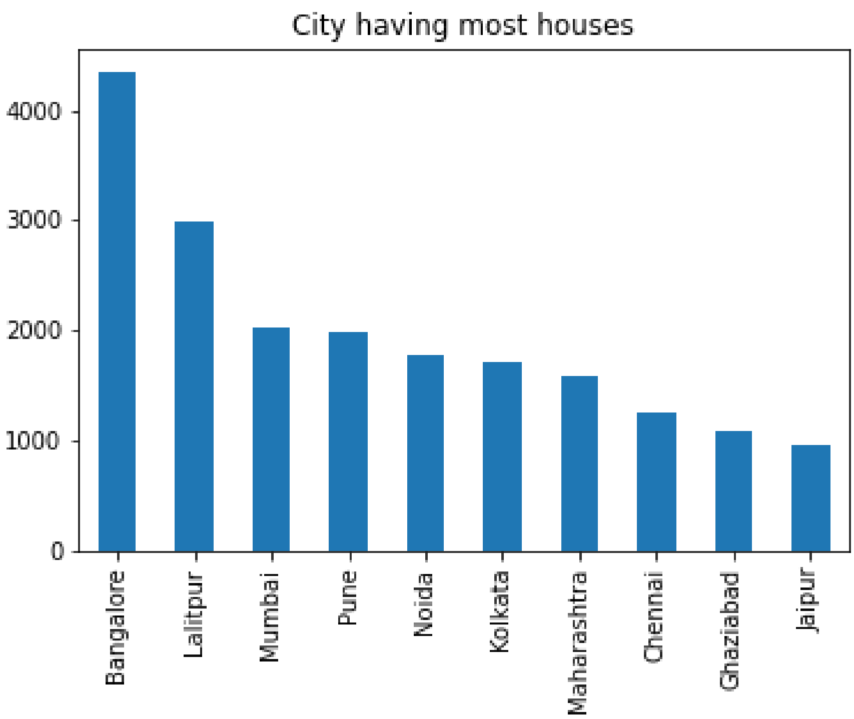
This bar plot shows us the value count of houses which are available to move in



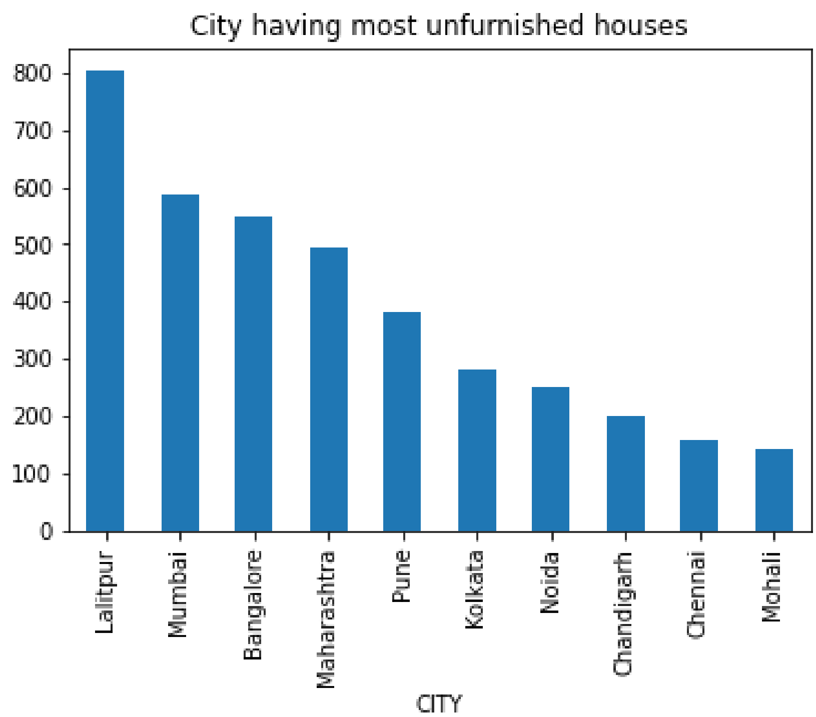
This bar plot shows us the value count of houses which are posted for resale



This bar plot shows us the town with most houses

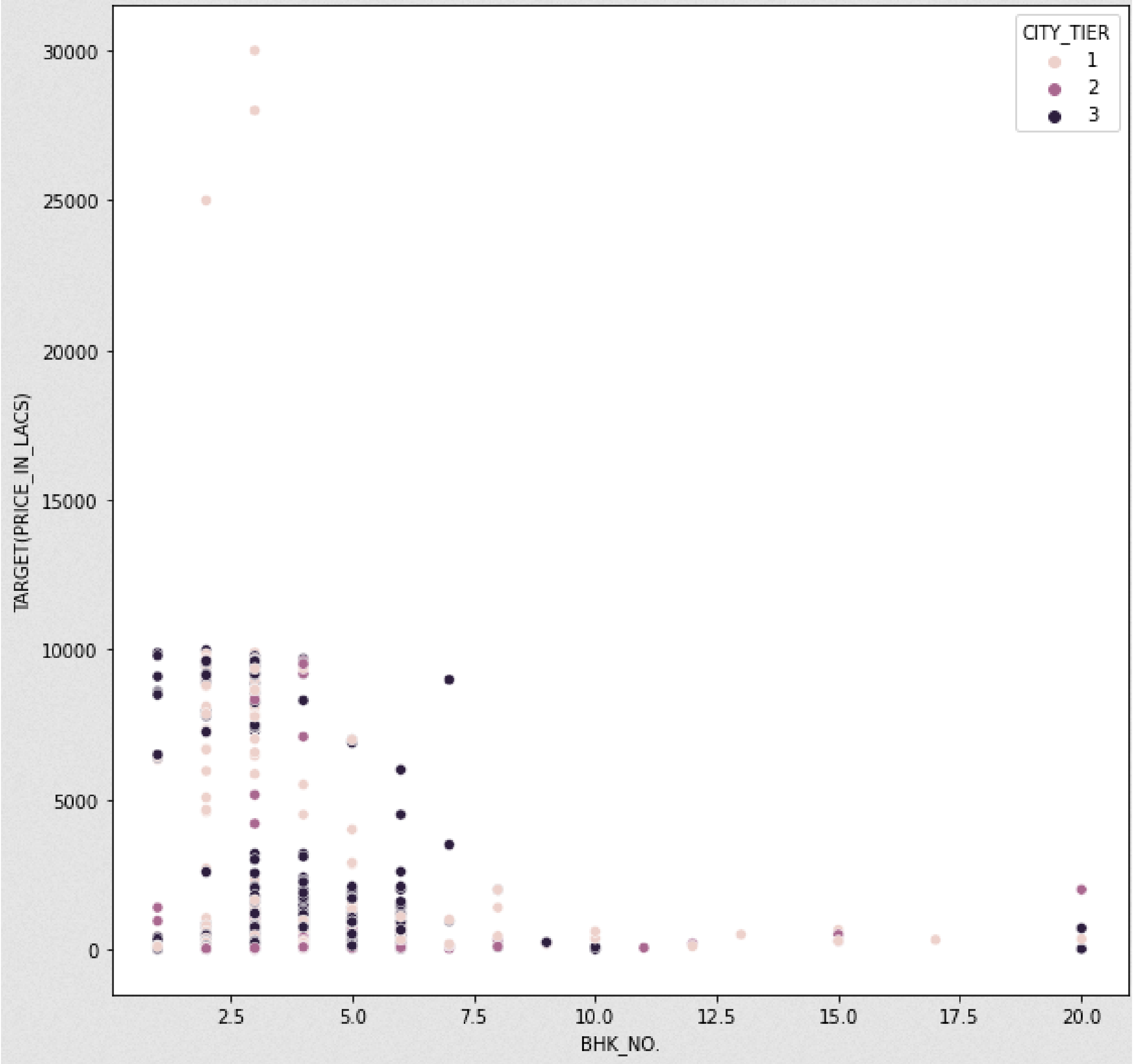


This bar plot shows us the city with most houses

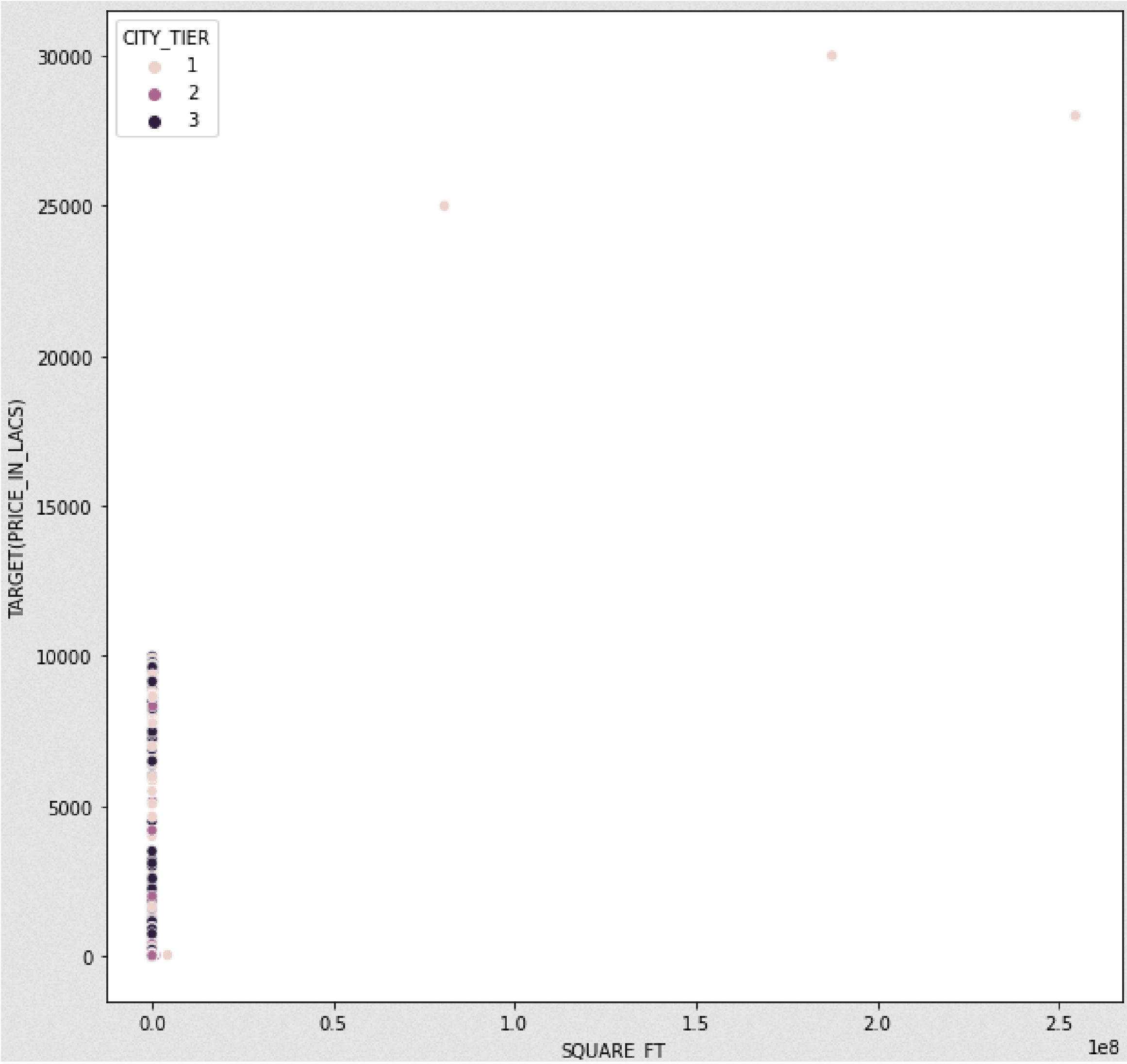


This bar plot shows us the city which has most unfurnished house

Scatter plots

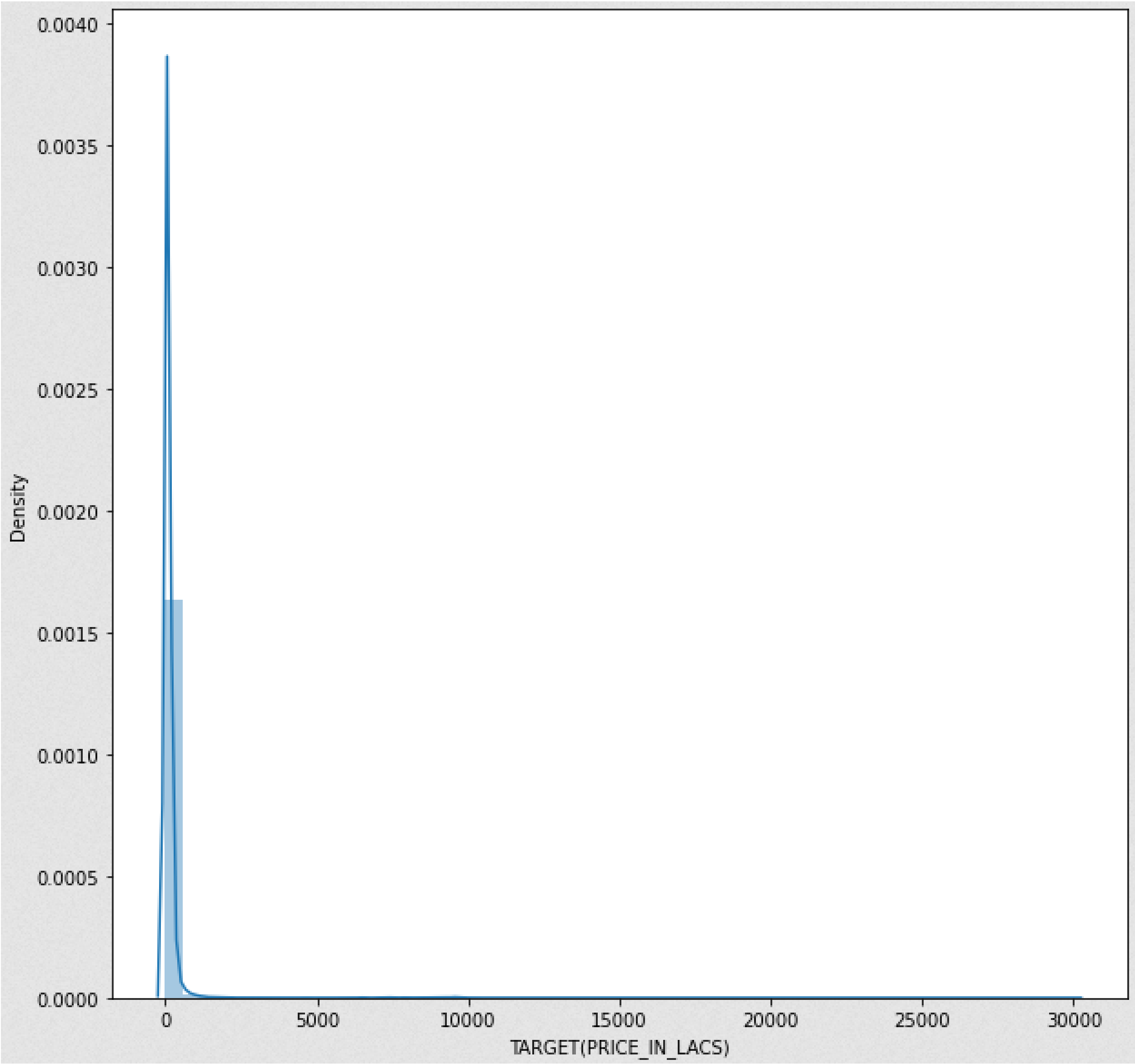


This scatter plot shows us the BHK number vs price with respect to their city tiers.



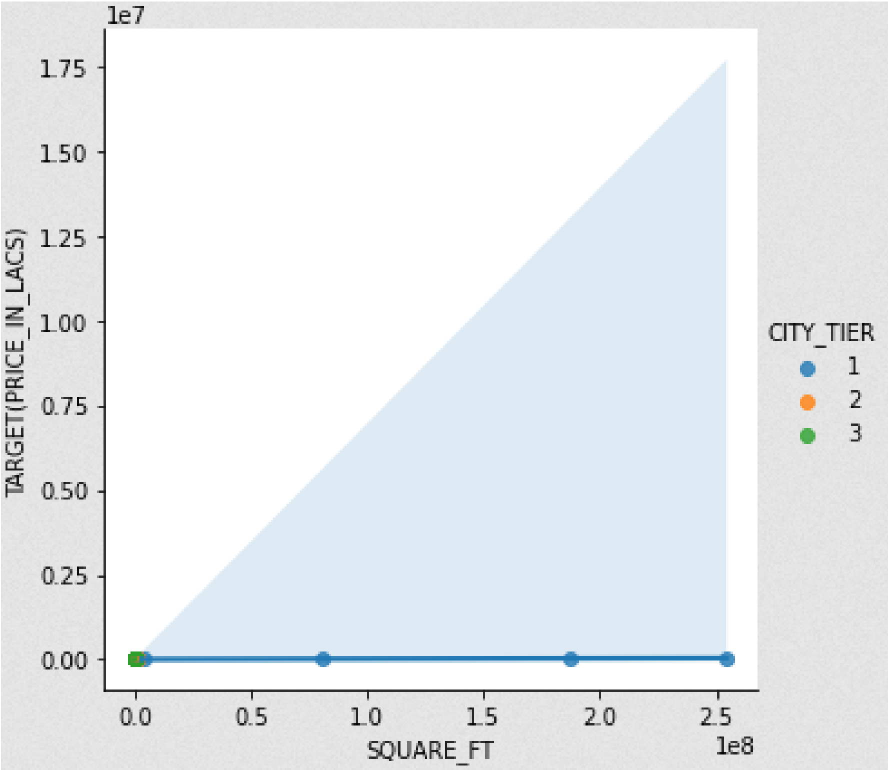
This scatter plot shows us the price vs square feet with respect to their city tiers

Distribution Plot



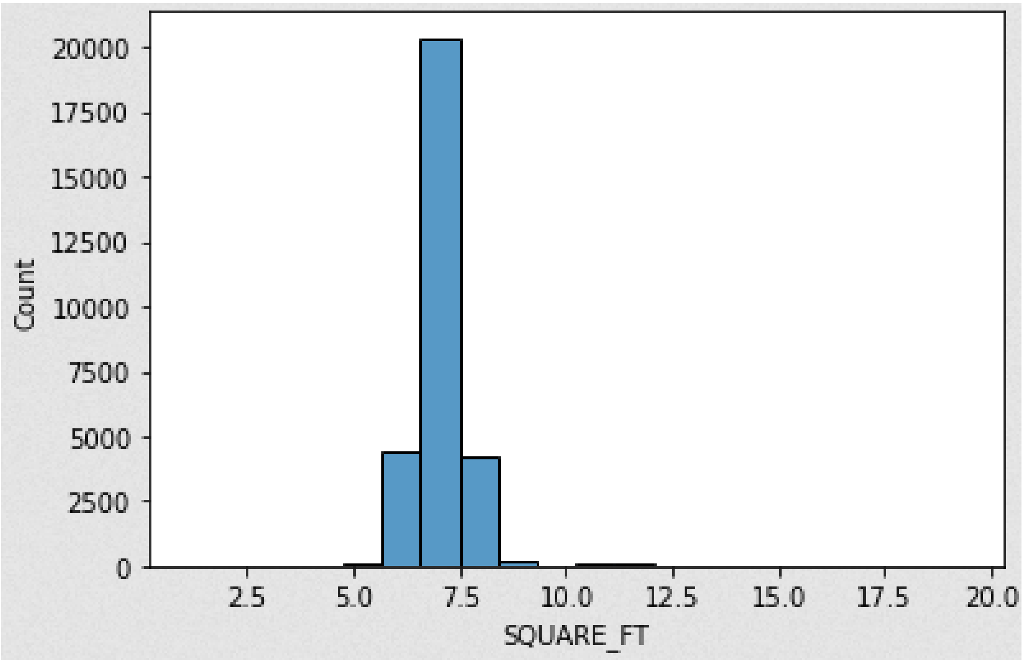
This distribution plot shows the distribution of the prices.

Line Plot

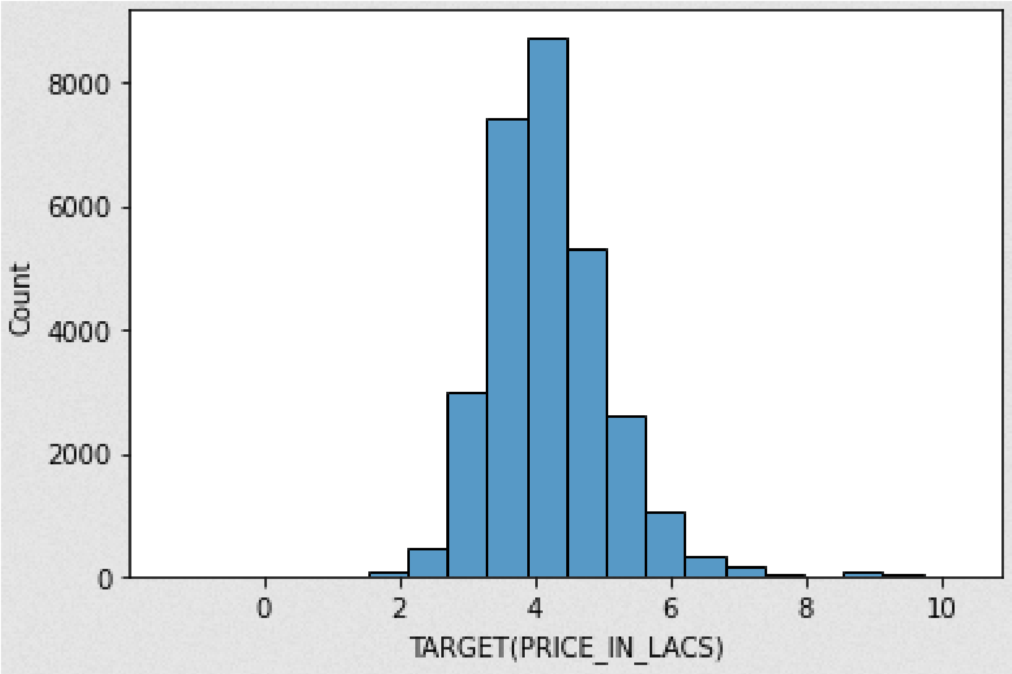


This line plot shows us the price vs square feet with respect to their city tiers

Histogram plots

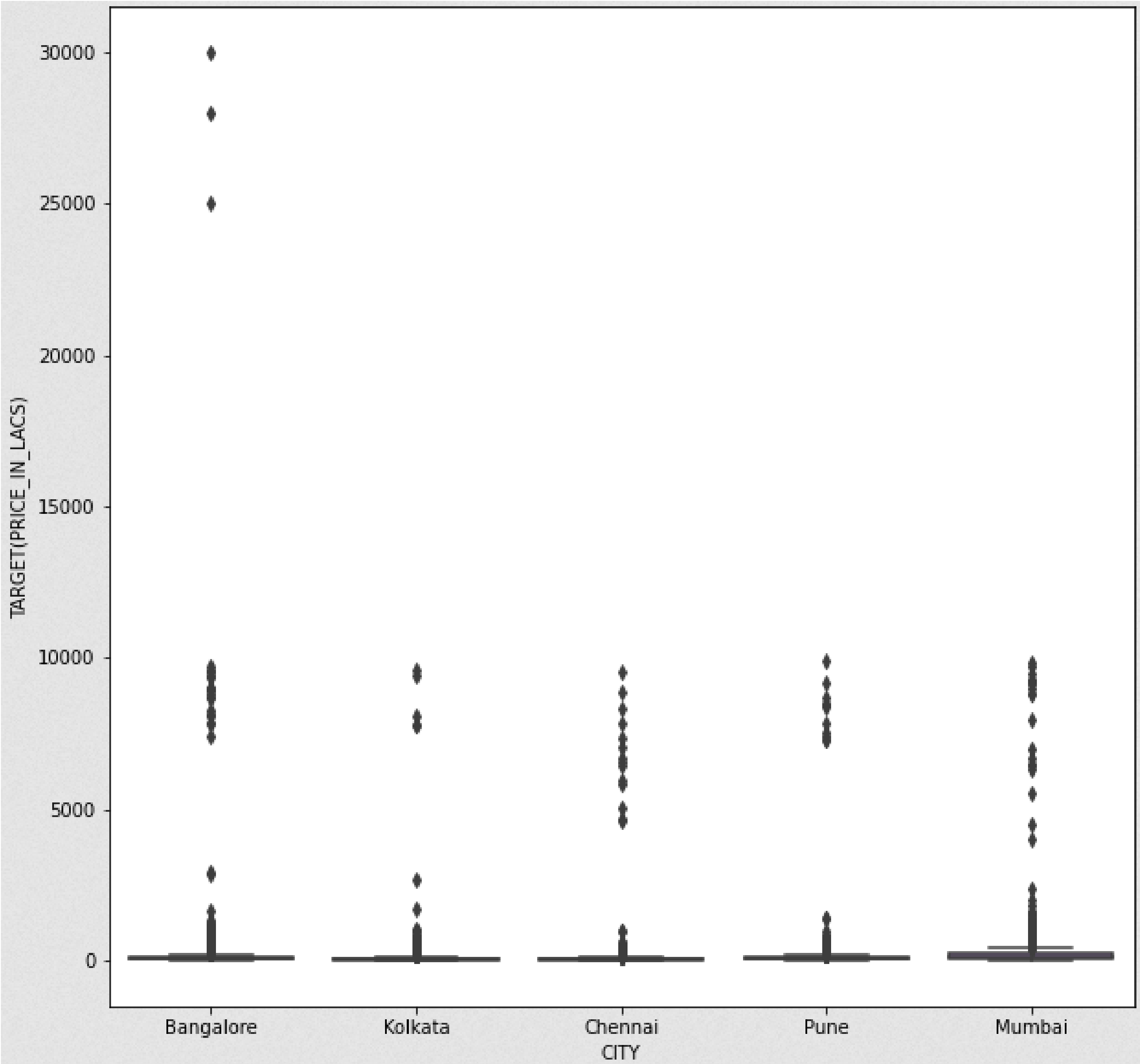


This histogram plot shows the log of square feet



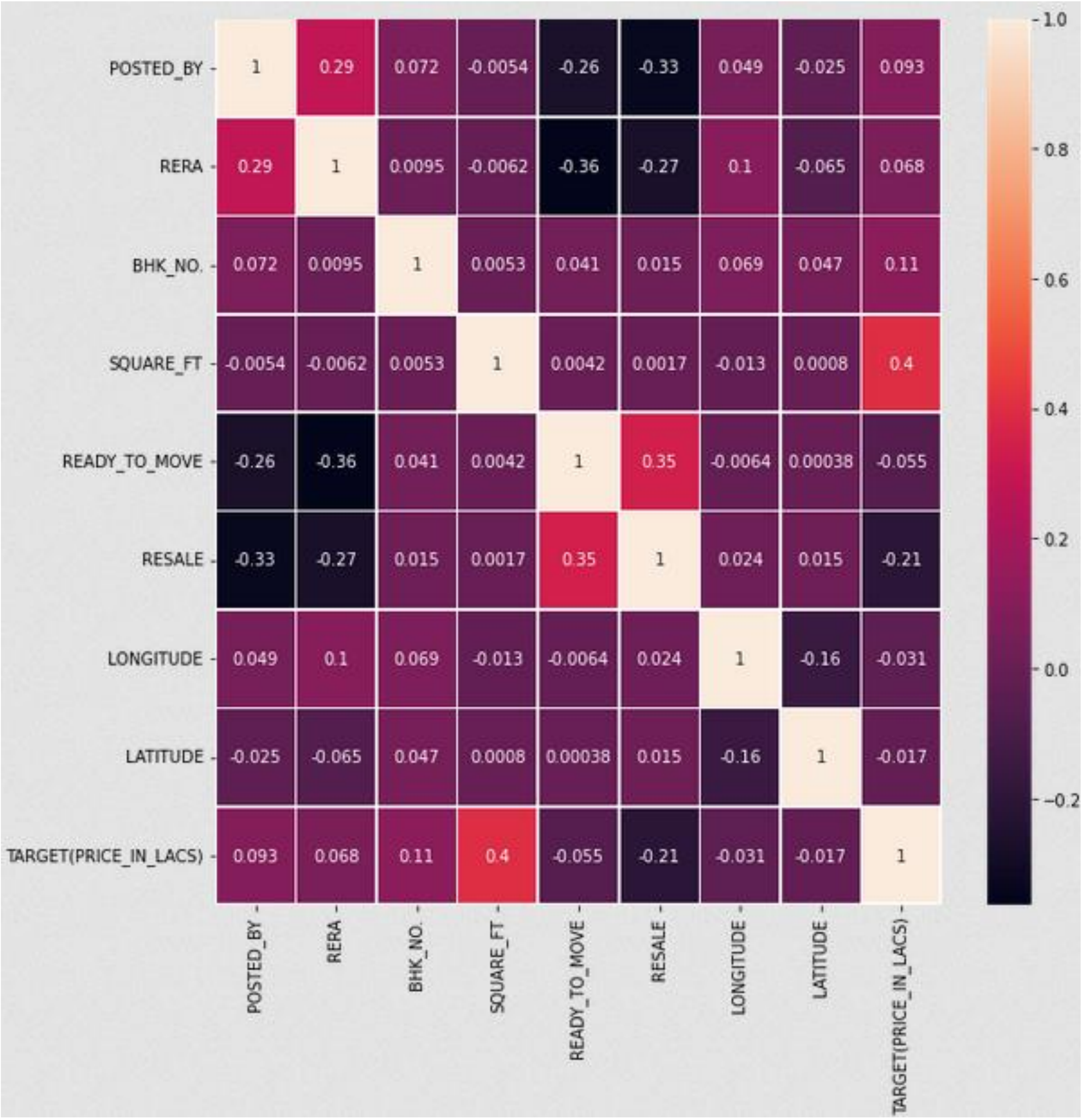
This histogram plot shows the log of price

Box plot



This box plot shows us the price of the houses in Tier 1 cities

Correlation plot



This correlation plot shows us that Target (price in lacs) is highly correlated to Square feet

Modelling

Modelling

Here for modelling I have built and trained two different types of linear regression models which are Ordinary Least Squares model and Ridge regression mode and I have used a regressor which is Gradient Boosting Regressor. Here we will be using the pre-built algorithms provided by the scikit learn package. We define a variable, we fit the train data and then we predict the data.

Models used:

Linear Regression:

Linear Regression fits a linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Ridge Regression:

Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients.

Gradient Boosting Regressor:

Gradient Boosting builds an additive model in a forward stage-wise fashion. It allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. Gradient Boosting Regressor supports a number of different loss functions for regression which can be specified via the argument loss; the default loss function for regression is squared error

Metrics of each models

```
Linear Regression

For original:
R2 score:  -3.699076323221991e+16
Mean Squared Error:  1.5724281299327596e+22
Mean Absolute Error:  1026437053.4144949
Mean Absolute Percentage Error: 1795323.453415927
----------
For standard:
R2 score:  -27938.731329297945
Mean Squared Error:  11876808058.581697
Mean Absolute Error:  77052.98290969369
Mean Absolute Percentage Error: 1780.2887586197214
----------
For normalized:
R2 score:  0.010663519075077121
Mean Squared Error:  420553.7752246251
Mean Absolute Error:  146.9495907322291
Mean Absolute Percentage Error: 2.1891762621804047
```

```
Ridge Regression

For original:
R2 score:  -1428263403898.3777
Mean Squared Error:  6.071357703938127e+17
Mean Absolute Error:  6382156.099814719
Mean Absolute Percentage Error: 11233.174533983745
----------
For standard:
R2 score:  -6.914121953865496
Mean Squared Error:  3364187.9476380027
Mean Absolute Error:  1274.6137401965977
Mean Absolute Percentage Error: 28.319336284473412
----------
For normalized:
R2 score:  0.0009504533541915272
Mean Squared Error:  424682.6702332316
Mean Absolute Error:  144.56707474756035
Mean Absolute Percentage Error: 2.191965479724098
```

```
eXtra Gradient Boosting Regressor

R2 score: 0.9474624242526161
Mean Squared Error: 22333.024453981197
Mean Absolute Error: 57.00104749270489
Mean Absolute Percentage Error: 0.7192098440395773
```

Conclusion

We first understood the data, cleaned the data, processed the data which was helpful in understanding the data as well as predict. Then we saw all the possible plots in the exploratory data analysis. For modelling we compared two models, Linear Regression and Ridge. We calculated the R squared error, Mean square error, Mean absolute error and the Mean absolute percentage error with help of the train data and compared with both the models.

So we can see that the metrics of normalized Ridge model is lesser compared to that of Linear Regression model. After that I tried XGB Regressor from XGBoost. Since the gradient boosting regressor had more accuracy, we chose that.

Even if the model is accurate, sometimes the model with lower accuracy rate can outperform the model with higher accuracy.

References

https://kaggle.com

https://pandas.pydata.org

https://numpy.org

https://scikit-learn.org

https://seaborn.pydata.org

https://matplotlib.org