CAPSTONE PROJECT

HOUSE PRICES IN INDIA PREDICTION

Submitted by,

AISWARYA JEYABRINDA S

Abstract

In India, house ownership is critical for very large population to meet general aspirations of economic well-being. The aim is to Predict the efficient House Pricing for real estate customers with respect to their budgets and priorities. Accurately predicting house prices can be a daunting task. The buyers are just not concerned about the size (square feet) of the house and there are various other factors that play a key role to decide the price of a house/property. There have been various research works that use different methods and techniques to address the question of the changing of house prices. This work considers the issue of changing house price as a classification problem and discuss machine learning techniques to predict whether buyers would buy a house if it is less than or greater than 25 lakhs using available data. The model experimented with 3 Algorithm: Logistic Regression, Random Forest Classifier and Ada boosting Classifier. Finally, the best results were obtained by applying Logistic Regression Algorithm.

Keywords: Machine Learning, House Prices, Logistic Regression, Random Forest Classifier, Ada Boosting Classifier.

Acknowledgements

I am using this opportunity to express my gratitude to everyone who supported us throughout the course of this group project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, we were fortunate to have Mr. Anbu Joel as our mentor. He has readily shared his immense knowledge in data analytics and guide us in a manner that the outcome resulted in enhancing our data skills.

We wish to thank all the faculties, as this project utilized knowledge gained from every course that formed the DSP program.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Date: August 02, 2022 Name: Aiswarya Jeyabrinda S

Place: Trichy

Certificate of Completion

I hereby certify that the project titled "House Prices in India" was undertaken and completed (July 2022)

Mentor: Mr. Anbu Joel

Date:August 02,2022

Place – Trichy

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Real Estate Property is not only a person's primary desire, but it also reflects a person's wealth and prestige in today's society. Real estate investment typically appears to be lucrative since property values do not drop in a choppy fashion. Changes in the value of the real estate will have an impact on many home investors, bankers, policymakers, and others. Real estate investing appears to be a tempting option for investors.

Housing is one of the most critical segments in the asset price market in the emerging market economies, particularly in India where house ownership is one of the most critical factors to very large population for facilitating financial inclusion and economic growth. More importantly, housing provides much needed employment as well as identity for a significant size of population at lower stratum. Policymakers and financial institutions monitor trends in house prices to expand their understanding of real estate and credit market conditions as well as to monitor its impact on economic activity, financial stability and soundness.

Tracking housing prices becomes imperative due to increased integration with global economy, accelerated real-estate activity with high and resilient growth expectations.

1.1 Title & Objective of the Study

The problem is not demand but affordability. There is still interest in homes, especially from first-time home buyers, but the increase in interest rates has taken some of the steam out of the market. This could lead to a moderation in home prices in some micro markets, depending on the level of inventory and demand.

This study aims to analyze the accuracy of predicting house prices using Classification, Random Forest classifier and Ada boosting classifier algorithms. Thus, the purpose of this study is to deepen the knowledge in classification methods in machine learning.

1.2 Business or Enterprise under Study

There are a number of factors that impact real estate prices, availability, and investment potential. Demographics provide information on the age, income, and regional preferences of actual or potential buyers, what percentage of buyers are retirees, and what percentage might buy a vacation or second home. Interest rates impact the price and demand of real estate—lower rates bring in more buyers, reflecting the lower cost of getting a mortgage, but also expand the demand for real estate, which can then drive-up prices. Real estate prices often follow the cycles of the economy, but investors can mitigate this risk by buying REITs or other diversified holdings that are either not tied to economic cycles or that can withstand downturns. Government policies and legislation, including tax incentives, deductions, and subsidies can boost or hinder demand for real estate.

1.3 DATA SOURCES

This dataset has been collected across various property aggregators across India. Here we are provided with 12 influencing factors to predict the prices as accurately as possible.

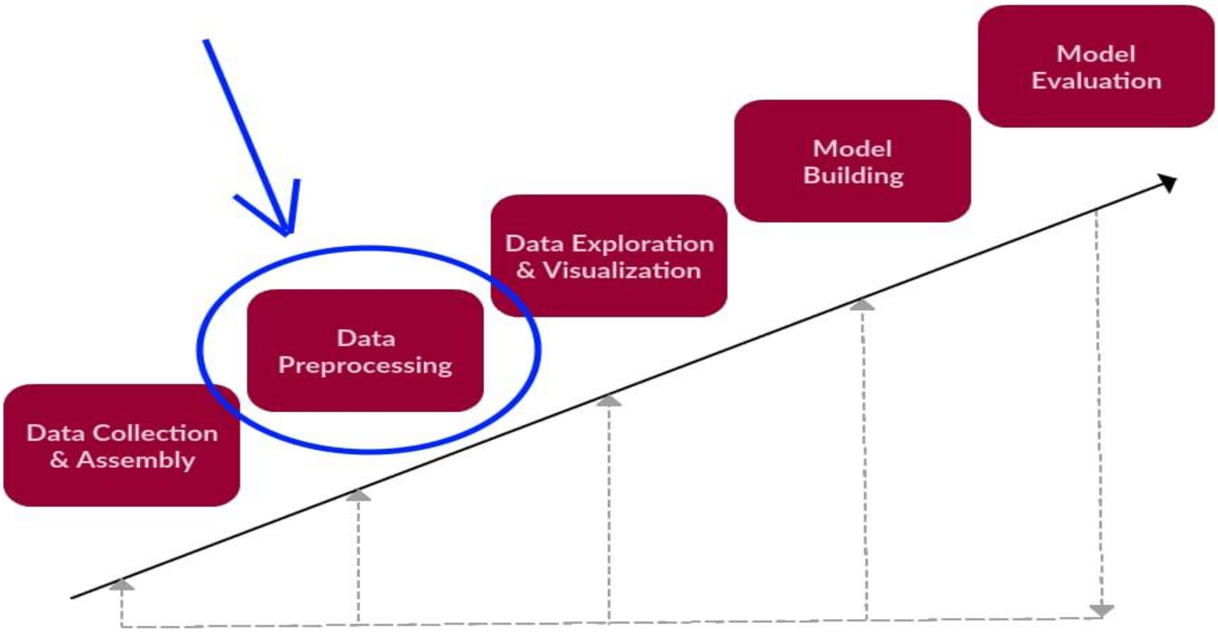
Attributes Description:

- 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*

CHAPTER 2

DATA PREPARATION AND UNDERSTANDING

Data Collection



Collecting data for training the Machine Learning model is the basic step in the machine learning pipeline. The predictions made by Machine Learning systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection:

- Inaccurate data. The collected data could be unrelated to the problem statement.
- Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.

- Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.
- Data bias. Depending on how the data, subjects and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age or region, for example. Data bias is difficult to detect and remove.

2.1 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a process of describing the data by means of statistical and visualization techniques in order to bring important aspects of that data into focus for further analysis.

The preprocessing of data involves 3 steps namely data cleaning, feature selection and data transformation. Each step is explained below:

Data transformation comprises of two explanatory variables which can be transformed from binomial form into binary form to be much applicable for the chosen models. The data cleaning step involves missing data imputation or handling. Some of the chosen algorithms cannot manage missing data that is why missing value can be transformed by median, mean or zero. However, the replacement of missing data by computed value statistically is a better choice.

The used set of data involves missing values in certain numerical variables and two categorical variables. Before training of model, feature selection is one of the most essential factors that can influence the model’s performance.

Preparation of data

The main purpose of preparation of data is to improve the quality of data and enhance the performance of data analysis. The preparation of data requires to be undertaken in a much iterative way until a conclusive result is met. The processes of preparation of data involves numerical variables discretization, missing values imputation, selection of feature of most informative variables, transformation from one discrete value set to another and derivation of new variables. The process of imputation includes changing the missing values with whole data based on an estimate from finished values. Making new variables from the information is based on transformation and discretization. Two new variables were formed to estimate the voice and transformation in usage of data. Before the data can be examined the data must be cleaned and keep it prepared so that the desired outputs can be derived from it. Data must be clean so that the errors and redundancy can be eliminated because having such information will lead to improper outcomes as well. To perform the Predictions of House Prices in India the logistic regression is used. Logistic regression is a statistical approach where the output variable is categorical rather than continuous. Logistic regression restricts the prediction to be one and zero interval.

Data Visualization

Data visualization is key, making the exploratory data analysis process streamline and easily analyzing data using wonderful plots and charts. Data Visualization represents the text or numerical data in a visual format, which makes it easy to grasp the information the data express. We, humans, remember the pictures more easily than readable text, so Python provides us various libraries for data visualization like matplotlib, seaborn, etc.

2.2 FEATURE ENGINEERING

The data was processed to convert it from its raw status into features to be used in machine learning algorithms. This process took the longest time due to the huge numbers of columns. The first idea was to aggregate values of columns per month (average, count, sum, max, min ...) for each numerical column per customer, and the count of distinct values for categorical columns.

Simply, by using Feature Engineering we improve the performance of the model.

Label Encoding

The following features are categorical therefore are transformed to binary integers

- Address
- Posted by
- BHK OR RK
- City Tier

Min-Max Scaling

Min-Max scaling is a normalization technique that enables us to scale data in a dataset to a specific range using each feature's minimum and maximum value.

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Values of numerical features are rescaled between a range of 0 and 1. Min-max scaler is the standard approach for scaling. For normally distributed features standard scaler could be used, which scales values around a mean of 0 and a standard deviation of 1. For simplicity we use min-max scaler for all numerical features.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

CHAPTER 3

FITTING MODELS TO DATA

3.1 Logistic Regression

Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets. The curve from the logistic function indicates the likelihood of something such as whether the House Prices is less than or greater than 25 lakhs.

Equation for Logistic Regression.

When predicting a qualitative outcome (class), the task is considered a classification problem. The independent variables should be independent of each other. That is, the model should have little or no multicollinearity. The independent variables are linearly related to the log odds. The logistic regression model is an analysis technique that helps predict the probability of an event happening in the future

3.2 Random Forest Classifier

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree. Random Forest algorithm was also trained, we optimized the number of trees hyper parameter. We experimented with building the model by changing the values of this parameter every time in 100, 200, 300, 400 and 500 trees. The best results show that the best number of trees was 200 trees. Increasing the number of trees after 200 will not give a significant increase in the performance.

The low correlation between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. **The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don't constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction. So, the prerequisites for random forest to perform well are:

1. There needs to be some actual signal in our features so that models built using those features do better than random guessing.
2. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

3.3 Ada Boosting Classifier

Boosting has been a prevalent technique for tackling binary classification problems. These algorithms improve the prediction power by converting a number of weak learners to strong learners. The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to rectify the errors present in the first model. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly. AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps. What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

To create the first learner, the algorithm takes the first feature, it will create 11 stumps as there are 11 features in this dataset. From these stumps, it will create eleven decision trees. This process can be called the stumps-base learner model. Out of these 11 models, the algorithm selects only one. Two properties are considered while selecting a base learner – Gini and Entropy. We must calculate Gini or Entropy the same way it is calculated for decision trees. The stump with the least value will be the first base learner.

Calculating total error and step 3 is to calculate the performance of the stumps Later Updating Weights!

CHAPTER 4

KEY FINDINGS

Below table provides a snapshot of the various models which can be choose from based on the pros and cons of each model.

S.no	***Model name***	***Accuracy***
1	Logistic Regression	0.68
2	Random Forest Classifier	0.58
3	Ada boosting Classifier	0.68

On comparing the various models, we find that Logistic Regression works the best with highest accuracy of 68.58% and Random Forest performs least with an accuracy of 58.47%.

The measures used in this study to verify the performance of classification are accuracy, recall, F-measure and precision. The significance of accuracy, recall, precision and F- measure is used to compare various classifiers effectiveness for prediction of House Prices. These metrics are applicable for examining any model performance which is constructed using both unbalanced and balanced set of data. Accuracy is defined as the accuracy prediction ratio to total set of predictions in a model. Precision is referred as an exactness measure. It can also be referred as from the samples mentioned as positive and how many actually belongs to the positive set of attributes. Recall is regarded as completeness measure and it mentions about how many positive sample classes are classified properly.

CHAPTER 5

RECOMMENDATION AND CONCLUSION

In the future, the GUI can be made more attractive and interactive. It can also be turned into any real-estate sale website where sellers can give the details and house for sale and buyers can contact according to the details given on the website. To simplify it for the user, there can also be a recommending system to recommend real-estate properties to the user based on the predicted price. To make the system even more informative and user-friendly, Google maps can also be included. This will show the neighborhood amenities such as hospitals, schools surrounding a region of 1 km from the given location. This can also be included in making predictions since the presence of such factors increases the price of real estate property.

CONCLUSION

We have used machine learning algorithms to predict the house prices. We have performed step by step procedure to analyze the dataset and found the correlation between the parameters. The manually collected Real-time Dataset has been collected which contains 68720 entries and independent variables. We analyze and pre- process this dataset before performing Exploratory Data Analysis. This analyzed feature set was given as an input to machine learning algorithms and calculated the performance of each model to compare based on Accuracy score. We found that Logistic Regression fits our dataset and gives the highest accuracy of 68.58%. Random Forest Classifier gives the least accuracy of 58.47%. Ada Boosting Classifier gives an accuracy of 68.53%. Thus, we conclude that we implemented Classification techniques to check how well an algorithm fits to given problem statement of House price prediction.

House Prices in India

The aim is to Predict the efficient House Pricing for real estate customers with respect to their budgets and priorities.

Importing Libraries

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

Understanding the Data

```
train = pd.read_csv("C:/Users/senth/Downloads/train.csv")
test = pd.read_csv("C:/Users/senth/Downloads/test.csv")
```

```
train.shape
```

```
(29451, 12)
```

```
train.columns
```

```
Index(['POSTED_BY', 'UNDER_CONSTRUCTION', 'RERA', 'BHK_NO.', 'BHK_OR_RK',
       'SQUARE_FT', 'READY_TO_MOVE', 'RESALE', 'ADDRESS', 'LONGITUDE',
       'LATITUDE', 'TARGET(PRICE_IN_LACS)'],
      dtype='object')
```

```
test.shape
```

```
(68720, 11)
```

```
test.columns
```

```
Index(['POSTED_BY', 'UNDER_CONSTRUCTION', 'RERA', 'BHK_NO.', 'BHK_OR_RK',
       'SQUARE_FT', 'READY_TO_MOVE', 'RESALE', 'ADDRESS', 'LONGITUDE',
       'LATITUDE'],
      dtype='object')
```

Adding a column to identify whether a row comes from train or not

```
test['is_train'] = 0
train['is_train'] = 1
```

is_train -> if price of house is less than 25 lakhs assign 0 else is_train = 1

Combining train and test

```
data = pd.concat([train,test],ignore_index=True)

data.head()

   POSTED_BY  UNDER_CONSTRUCTION  RERA  BHK_NO. BHK_OR_RK    SQUARE_FT  \
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   
..       ...                 ...   ...      ...       ...          ...   
95    Dealer                   0     0        3       BHK  1200.082764   
96    Dealer                   0     0        2       BHK  1584.283904   
97     Owner                   0     0        2       BHK   900.000000   
98    Dealer                   0     1        2       BHK  1193.440032   
99     Owner                   0     0        2       BHK  1487.357462   

    READY_TO_MOVE  RESALE                      ADDRESS  LONGITUDE  
LATITUDE  \
0               1       1        Ksfc Layout,Bangalore  12.969910  
77.597960   
1               1       1    Vishweshwara Nagar,Mysore  12.274538  
76.644605   
2               1       1             Jigani,Bangalore  12.778033  
77.632191   
3               1       1  Sector-1 Vaishali,Ghaziabad  28.642300  
77.344500   
4               0       1             New Town,Kolkata  22.592200  
88.484911   
..            ...     ...                          ...        ...  
...   
95              1       1             Satgachi,Kolkata  23.207157  
88.404471   
96              1       1           Tilak Nagar,Kanpur  26.459137  
79.506922   
97              1       1            Kalol,Gandhinagar  23.225700  
72.516790   
98              1       1                Talegaon,Pune  18.441256  
74.647361   
99              1       1  Mansarovar Extension,Jaipur  26.862600  
75.763300   

    TARGET(PRICE_IN_LACS)  is_train  
0                    55.0         1  
1                    51.0         1  
2                    43.0         1  
3                    62.5         1  
4                    60.5         1  
..                    ...       ...  
```

```
95                58.0         1
96               100.0         1
97                18.0         1
98                60.4         1
99                30.0         1

[100 rows x 13 columns]
```

```
data.shape
```

```
(98171, 13)
```

```
data.columns
```

```
Index(['POSTED_BY', 'UNDER_CONSTRUCTION', 'RERA', 'BHK_NO.', 'BHK_OR_RK',
       'SQUARE_FT', 'READY_TO_MOVE', 'RESALE', 'ADDRESS', 'LONGITUDE',
       'LATITUDE', 'TARGET(PRICE_IN_LACS)', 'is_train'],
      dtype='object')
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98171 entries, 0 to 98170
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   POSTED_BY              98171 non-null  object 
 1   UNDER_CONSTRUCTION     98171 non-null  int64  
 2   RERA                   98171 non-null  int64  
 3   BHK_NO.                98171 non-null  int64  
 4   BHK_OR_RK              98171 non-null  object 
 5   SQUARE_FT              98171 non-null  float64
 6   READY_TO_MOVE          98171 non-null  int64  
 7   RESALE                 98171 non-null  int64  
 8   ADDRESS                98171 non-null  object 
 9   LONGITUDE              98171 non-null  float64
 10  LATITUDE               98171 non-null  float64
 11  TARGET(PRICE_IN_LACS)  29451 non-null  float64
 12  is_train               98171 non-null  int64  
dtypes: float64(4), int64(6), object(3)
memory usage: 9.7+ MB
```

```
data.describe()
```

```
       UNDER_CONSTRUCTION          RERA        BHK_NO.     SQUARE_FT  \
count        98171.000000  98171.000000  98171.000000  9.817100e+04   
mean             0.177517      0.316947      2.389423  7.874292e+03   
std              0.382107      0.465289      0.868954  1.050427e+06   
min              0.000000      0.000000      1.000000  1.000000e+00   
25%              0.000000      0.000000      2.000000  9.000277e+02   
50%              0.000000      0.000000      2.000000  1.175007e+03   
75%              0.000000      1.000000      3.000000  1.550388e+03   
```

```
max                1.000000      1.000000     31.000000  2.545455e+08

       READY_TO_MOVE        RESALE     LONGITUDE      LATITUDE  \
count   98171.000000  98171.000000  98171.000000  98171.000000
mean        0.822483      0.932322     21.291708     76.894881
std         0.382107      0.251194      6.186898     10.240142
min         0.000000      0.000000    -38.391261   -121.761248
25%         1.000000      1.000000     18.452663     73.798100
50%         1.000000      1.000000     20.904426     77.324966
75%         1.000000      1.000000     26.893640     77.968485
max         1.000000      1.000000     65.183330    175.278040

       TARGET(PRICE_IN_LACS)      is_train
count           29451.000000  98171.000000
mean              142.898746      0.299997
std               656.880713      0.458259
min                 0.250000      0.000000
25%                38.000000      0.000000
50%                62.000000      0.000000
75%               100.000000      1.000000
max             30000.000000      1.000000
```

```
data.nunique()
```

```
POSTED_BY                    3
UNDER_CONSTRUCTION           2
RERA                         2
BHK_NO.                     19
BHK_OR_RK                    2
SQUARE_FT                48733
READY_TO_MOVE                2
RESALE                       2
ADDRESS                  13322
LONGITUDE                 6801
LATITUDE                  6803
TARGET(PRICE_IN_LACS)     1172
is_train                     2
dtype: int64
```

```
data.POSTED_BY.unique()
```

```
array(['Owner', 'Dealer', 'Builder'], dtype=object)
```

Exploratory Data Analysis

Let's begin some exploratory data analysis! We'll start by checking out missing data!

```
data.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)    68720
is_train                     0
dtype: int64
```

```
train['TARGET(PRICE_IN_LACS)'].isnull().mean()
```

```
0.0
```

We can drop LONGITUDE and LATITUDE which does not affect TARGET(PRICE_IN_LACS)

```
data.drop(['LONGITUDE','LATITUDE'],axis = 1,inplace = True)
```

Extract city from address

```
data['ADDRESS'] = data['ADDRESS'].str.split(',').apply(lambda x: x[-1])
data['ADDRESS'].value_counts().head(15)
```

```
Bangalore      14341
Lalitpur       10063
Pune            6591
Mumbai          6539
Kolkata         5850
Noida           5826
Maharashtra     5258
Chennai         4136
Ghaziabad       3604
Jaipur          3229
Chandigarh      2186
Faridabad       2127
Mohali          1882
Vadodara        1853
Surat           1449
Name: ADDRESS, dtype: int64
```

Map all cities into tier-1, tier-2 and tier-3

Reference https://en.wikipedia.org/wiki/Classification_of_Indian_cities

```
def map_city(city):
    if city in ['Ahmedabad', 'Bangalore', 'Chennai', 'Delhi', 'Hyderabad',
'Kolkata', 'Mumbai', 'Pune', 'Maharashtra']:
```

```
        return 'tier-1'
    elif city in ['Agra', 'Ajmer', 'Aligarh', 'Amravati', 'Amritsar',
'Asansol', 'Aurangabad', 'Bareilly',
                  'Belgaum', 'Bhavnagar', 'Bhiwandi', 'Bhopal',
'Bhubaneswar', 'Bikaner', 'Bilaspur', 'Bokaro Steel City',
                  'Chandigarh', 'Coimbatore', 'Cuttack', 'Dehradun',
'Dhanbad', 'Bhilai', 'Durgapur', 'Dindigul', 'Erode',
                  'Faridabad', 'Firozabad', 'Ghaziabad', 'Gorakhpur',
'Gulbarga', 'Guntur', 'Gwalior', 'Gurgaon', 'Guwahati',
                  'Hamirpur', 'Hubli–Dharwad', 'Indore', 'Jabalpur',
'Jaipur', 'Jalandhar', 'Jammu', 'Jamnagar', 'Jamshedpur',
                  'Jhansi', 'Jodhpur', 'Kakinada', 'Kannur', 'Kanpur',
'Karnal', 'Kochi', 'Kolhapur', 'Kollam', 'Kozhikode',
                  'Kurnool', 'Ludhiana', 'Lucknow', 'Madurai',
'Malappuram', 'Mathura', 'Mangalore', 'Meerut', 'Moradabad',
                  'Mysore', 'Nagpur', 'Nanded', 'Nashik', 'Nellore',
'Noida', 'Patna', 'Pondicherry', 'Purulia', 'Prayagraj',
                  'Raipur', 'Rajkot', 'Rajahmundry', 'Ranchi', 'Rourkela',
'Ratlam', 'Salem', 'Sangli', 'Shimla', 'Siliguri',
                  'Solapur', 'Srinagar', 'Surat', 'Thanjavur',
'Thiruvananthapuram', 'Thrissur', 'Tiruchirappalli', 'Tirunelveli',
                  'Tiruvannamalai', 'Ujjain', 'Bijapur', 'Vadodara',
'Varanasi', 'Vasai-Virar City', 'Vijayawada', 'Visakhapatnam',
                  'Vellore', 'Warangal']:
        return 'tier-2'
    else:
        return 'tier-3'
data['CITY_TIER'] = data['ADDRESS'].apply(map_city)
```

```
data.head()
```

```
  POSTED_BY  UNDER_CONSTRUCTION  RERA  BHK_NO. BHK_OR_RK    SQUARE_FT  \
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

   READY_TO_MOVE  RESALE    ADDRESS  TARGET(PRICE_IN_LACS)  is_train
CITY_TIER
0              1       1  Bangalore                   55.0         1
tier-1
1              1       1     Mysore                   51.0         1
tier-2
2              1       1  Bangalore                   43.0         1
tier-1
3              1       1  Ghaziabad                   62.5         1
tier-2
4              0       1    Kolkata                   60.5         1
tier-1
```

```
categorical_features = ['POSTED_BY','BHK_OR_RK','ADDRESS','CITY_TIER']
numerical_features = ['UNDER_CONSTRUCTION','RERA','BHK_NO.','SQUARE_FT',
'READY_TO_MOVE', 'RESALE']
```

Checking for Outliers in Numerical Features

```
numerical_features = ['UNDER_CONSTRUCTION','RERA','BHK_NO.','SQUARE_FT',
'READY_TO_MOVE', 'RESALE']
data_num = data[numerical_features]
data_num.describe()

Q1 = data_num.quantile(0.25)
Q3 = data_num.quantile(0.75)
IQR = Q3 - Q1
((data_num < (Q1 - 1.5 * IQR)) | (data_num > (Q3 + 1.5 * IQR))).any()
```

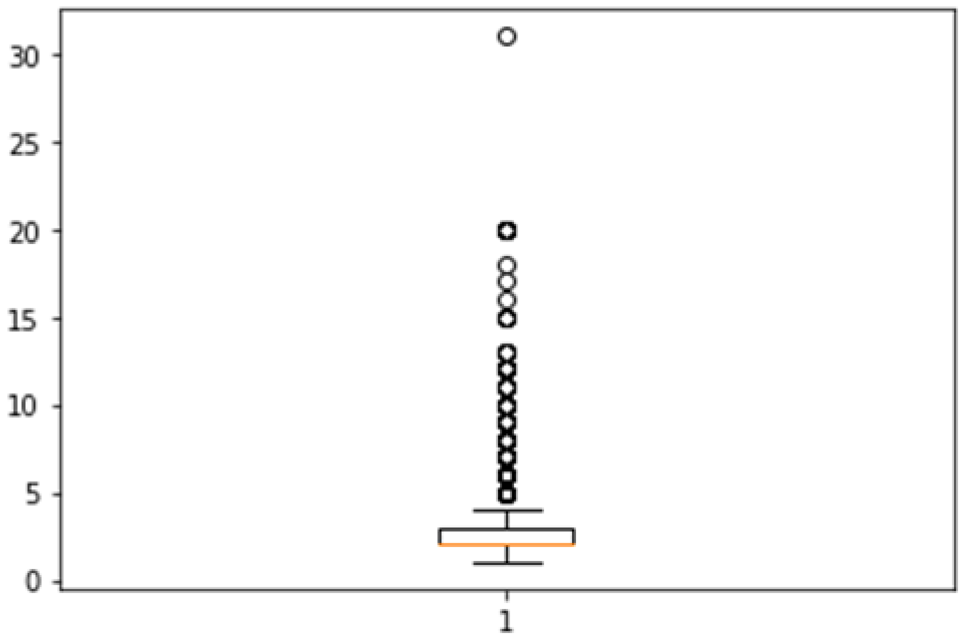
```
UNDER_CONSTRUCTION     True
RERA                  False
BHK_NO.                True
SQUARE_FT              True
READY_TO_MOVE          True
RESALE                 True
dtype: bool
```

There are outliers in numerical features detected with the IQR method.

Continuous Data BHK_NO.,SQUARE_FT and TARGET(PRICE_IN_LACS)'])

```
plt.boxplot(data['BHK_NO.'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x198116532b0>,
  <matplotlib.lines.Line2D at 0x19811653640>],
 'caps': [<matplotlib.lines.Line2D at 0x198116539d0>,
  <matplotlib.lines.Line2D at 0x19811653d60>],
 'boxes': [<matplotlib.lines.Line2D at 0x19811764ee0>],
 'medians': [<matplotlib.lines.Line2D at 0x1982adc0130>],
 'fliers': [<matplotlib.lines.Line2D at 0x1982adc04c0>],
 'means': []}
```




```
Q1 = data['BHK_NO.'].quantile(0.25)
Q3 = data['BHK_NO.'].quantile(0.75)
IQR = Q3 - Q1

UE = Q3 + 1.5 * (IQR)
LE = Q1 - 1.5 * (IQR)

data['BHK_NO.'][data['BHK_NO.']>UE]=UE
data['BHK_NO.'][data['BHK_NO.']<LE]=LE
```

```
C:\Users\senth\AppData\Local\Temp/ipykernel_15244/2566831775.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['BHK_NO.'][data['BHK_NO.']>UE]=UE
C:\Users\senth\AppData\Local\Temp/ipykernel_15244/2566831775.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['BHK_NO.'][data['BHK_NO.']<LE]=LE
```

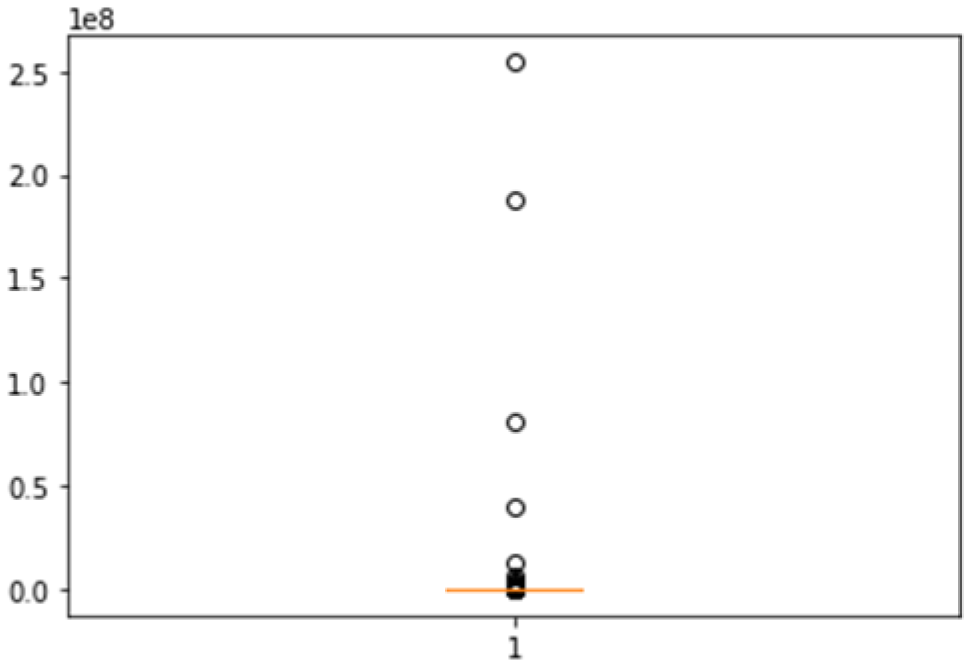
```
plt.boxplot(data['BHK_NO.'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x19857753e20>,
  <matplotlib.lines.Line2D at 0x1985773f1f0>],
 'caps': [<matplotlib.lines.Line2D at 0x1985773f580>,
  <matplotlib.lines.Line2D at 0x1985773f910>],
 'boxes': [<matplotlib.lines.Line2D at 0x19857753a90>],
 'medians': [<matplotlib.lines.Line2D at 0x1985773fca0>],
```

```
 'fliers': [<matplotlib.lines.Line2D at 0x1981f420070>],
 'means': []}
```

```
plt.boxplot(data['SQUARE_FT'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1985a2ba550>,
  <matplotlib.lines.Line2D at 0x1985a2ba8e0>],
 'caps': [<matplotlib.lines.Line2D at 0x1985a2bac70>,
  <matplotlib.lines.Line2D at 0x1985a2a3040>],
 'boxes': [<matplotlib.lines.Line2D at 0x1985a2ba1c0>],
 'medians': [<matplotlib.lines.Line2D at 0x1985a2a33d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1985a2a3760>],
 'means': []}
```



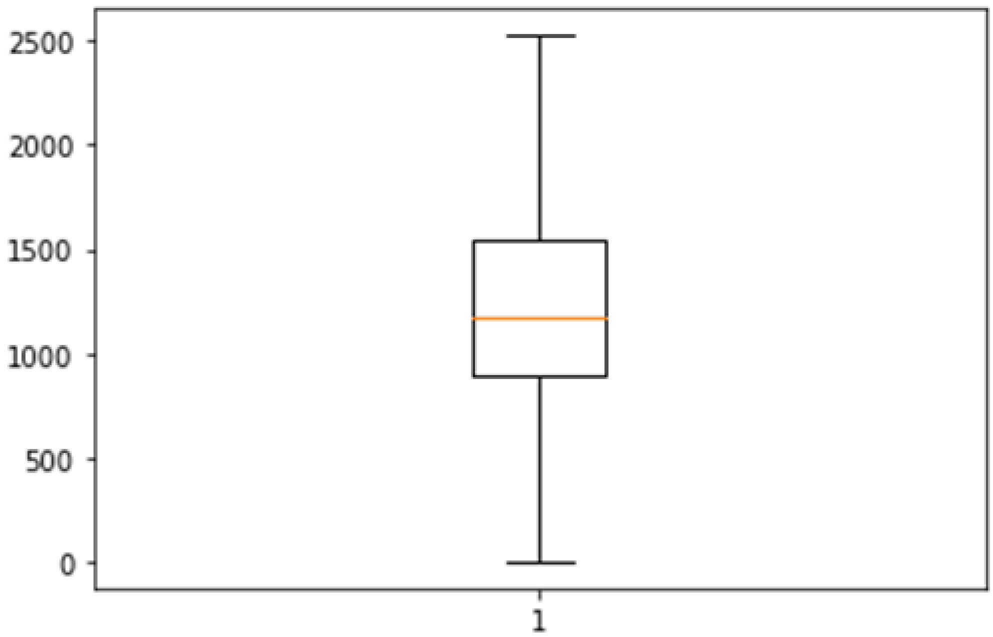

```
Q1 = data['SQUARE_FT'].quantile(0.25)
Q3 = data['SQUARE_FT'].quantile(0.75)
IQR = Q3 - Q1

UE = Q3 + 1.5 * (IQR)
LE = Q1 - 1.5 * (IQR)

data['SQUARE_FT'][data['SQUARE_FT']>UE]=UE
data['SQUARE_FT'][data['SQUARE_FT']<LE]=LE
```

```
C:\Users\senth\AppData\Local\Temp/ipykernel_15244/1657066256.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  data['SQUARE_FT'][data['SQUARE_FT']>UE]=UE
C:\Users\senth\AppData\Local\Temp/ipykernel_15244/1657066256.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['SQUARE_FT'][data['SQUARE_FT']<LE]=LE
```

```
plt.boxplot(data['SQUARE_FT'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x19857902e80>,
  <matplotlib.lines.Line2D at 0x1985ed81250>],
 'caps': [<matplotlib.lines.Line2D at 0x1985ed815e0>,
  <matplotlib.lines.Line2D at 0x1985ed81970>],
 'boxes': [<matplotlib.lines.Line2D at 0x19857902af0>],
 'medians': [<matplotlib.lines.Line2D at 0x1985ed81d00>],
 'fliers': [<matplotlib.lines.Line2D at 0x1985ed6b0d0>],
 'means': []}
```

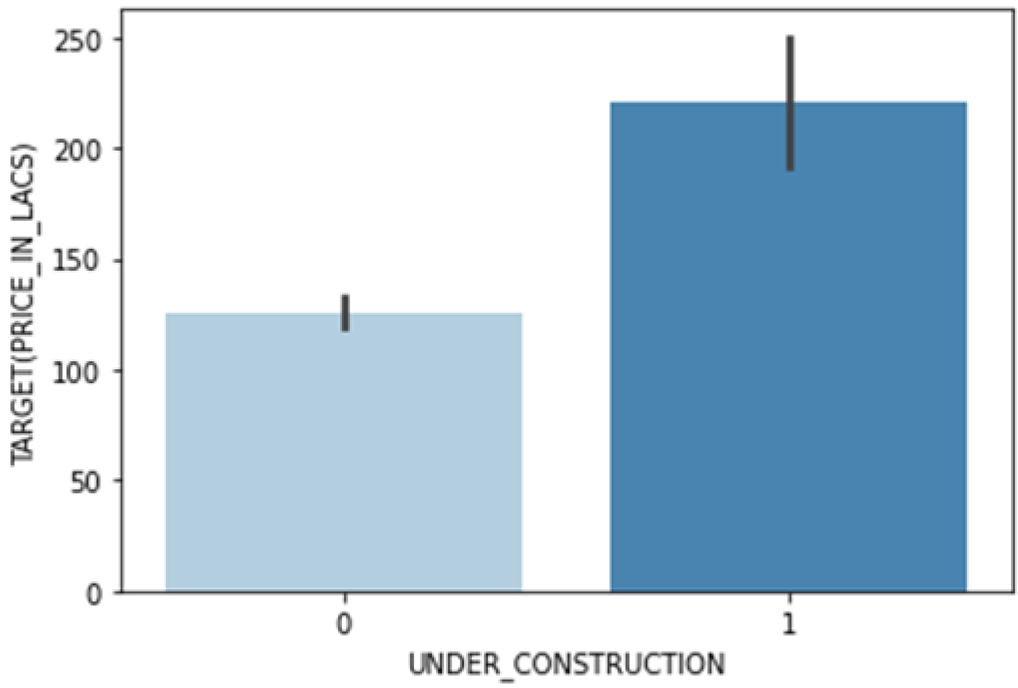


DATA VISUALIZATION

UNDER_CONSTRUCTION Vs TARGET PRICE

```
sns.barplot(x = data['UNDER_CONSTRUCTION'], y =
data['TARGET(PRICE_IN_LACS)'], palette='Blues')
```

```
<AxesSubplot:xlabel='UNDER_CONSTRUCTION', ylabel='TARGET(PRICE_IN_LACS)'>
```

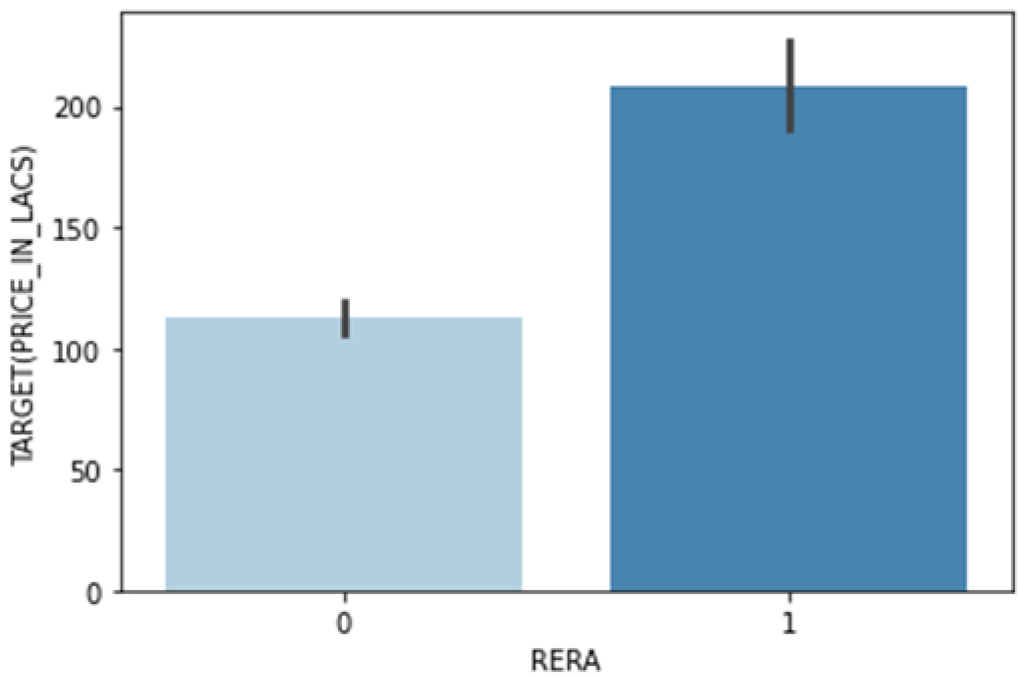


The House UNDER_CONSTRUCTION contribute Higher Prices

RERA Vs TARGET PRICE

```
sns.barplot(x = data['RERA'], y =
data['TARGET(PRICE_IN_LACS)'],palette='Blues')
```

```
<AxesSubplot:xlabel='RERA', ylabel='TARGET(PRICE_IN_LACS)'>
```

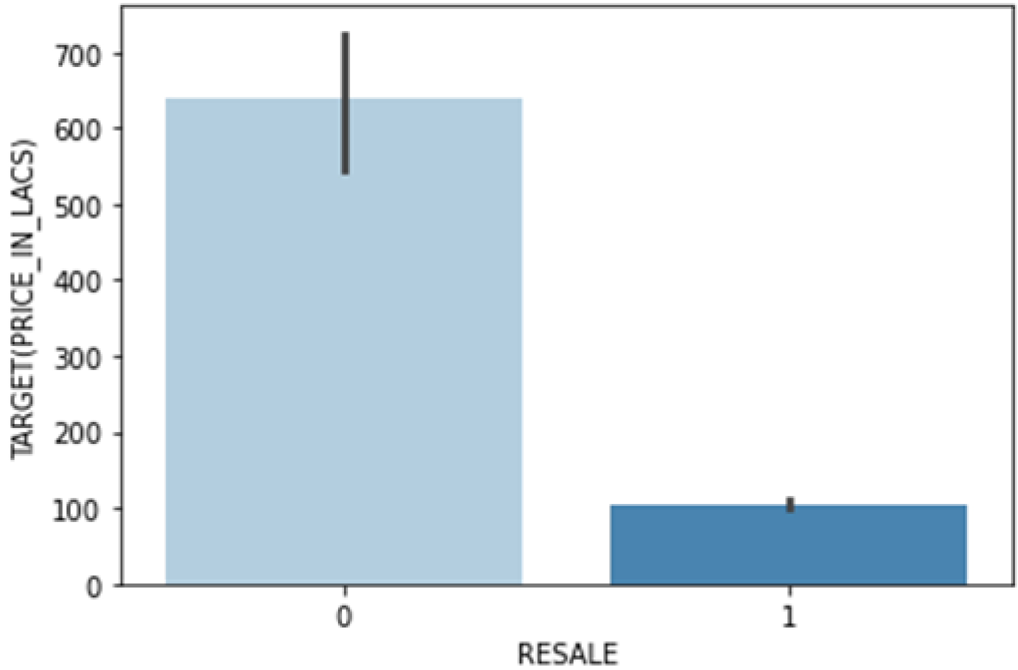


It is clear that from the above barplot the price of house with RERA approval is valued higher price than the house without RERA approval

RESALE Vs TARGET PRICE

```
sns.barplot(x = data['RESALE'], y =
data['TARGET(PRICE_IN_LACS)'],palette='Blues')
```

```
<AxesSubplot:xlabel='RESALE', ylabel='TARGET(PRICE_IN_LACS)'>
```

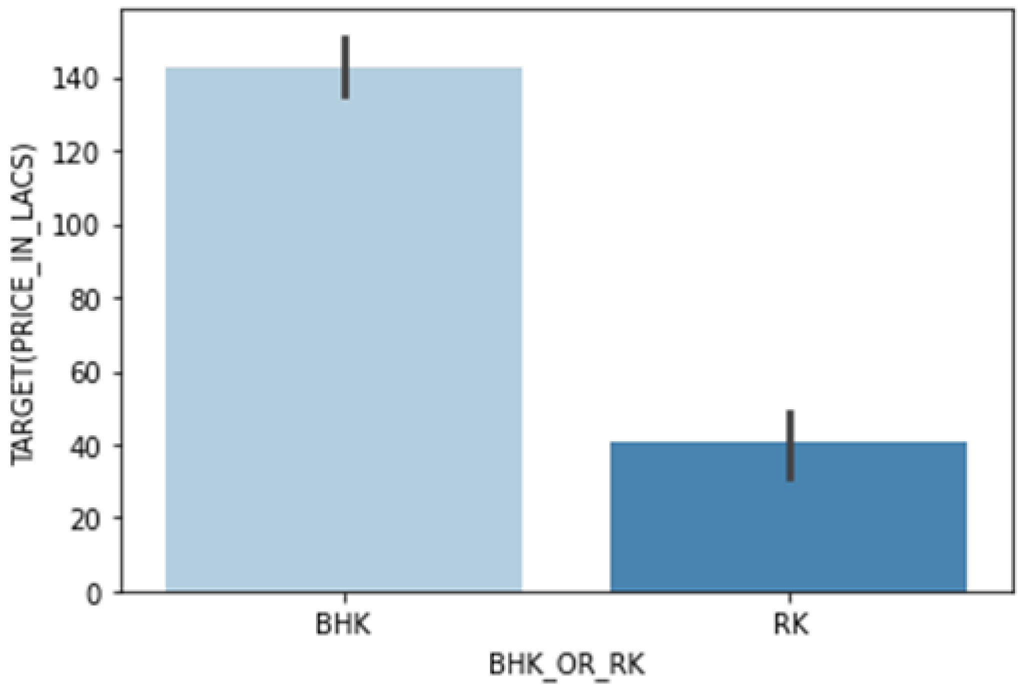


It's clear from the above Barplot that the price of a brand new house is higher than the price of the house which is resold

BHK_OR_RK Vs TARGET PRICE

```
sns.barplot(x = data['BHK_OR_RK'], y =
data['TARGET(PRICE_IN_LACS)'],palette='Blues')
```

```
<AxesSubplot:xlabel='BHK_OR_RK', ylabel='TARGET(PRICE_IN_LACS)'>
```

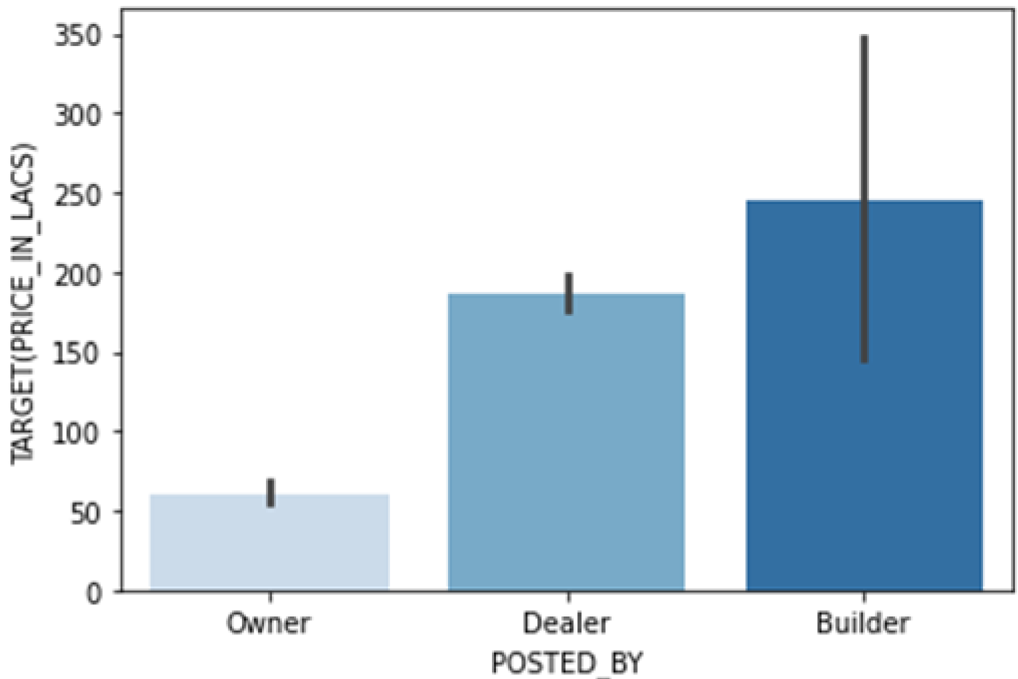


The price of house with BHK(Bedroom, Hall, Kitchen) is costlier than a house with only RK(Room, Kichen).

POSTED_BY Vs TARGET PRICE

```
sns.barplot(x = data['POSTED_BY'], y =
data['TARGET(PRICE_IN_LACS)'],palette='Blues')
```

```
<AxesSubplot:xlabel='POSTED_BY', ylabel='TARGET(PRICE_IN_LACS)'>
```



The houses posted by Builder gets sold for higher price, followed by dealer and Owner

Dropping TARGET(PRICE_IN_LACS) column as it is not present in the test

```
data = data.drop(['TARGET(PRICE_IN_LACS)'],axis = 1)
```

Data Preparation

Converting Categorical Features

Converting the categorical data to numerical values

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
```

```
data.ADDRESS = label_encoder.fit_transform(data.ADDRESS)
```

```
data.ADDRESS.unique()
```

```
array([ 24, 196,  97, 166, 163, 126, 190,  60, 273, 209, 239,  49, 311,
       234, 194, 201,  73,  46,  89, 174, 180, 299, 307, 301, 185,  19,
       213, 305,  33,  48, 176, 151,  94, 231,   1, 249, 106, 297, 123,
       141,  64, 300, 217, 283, 302, 107,  10,  14, 294, 266, 241, 309,
       291, 265, 179,  58, 267, 108, 243, 280,   7,  35, 257,  77, 175,
        84, 221, 279,  83,  98, 135, 117, 124, 120, 192, 121, 156, 228,
        53, 251, 188, 170, 129,  12, 227, 177,   8, 169, 220, 165,  86,
        42,  41,  18, 137, 186,  15,  85,  43,  72, 104, 130, 296, 105,
       206, 115, 274, 118, 171,  75, 205,  22, 207,  80, 293,  67, 114,
       202, 134, 148, 145, 152, 125, 223, 195, 263, 139,  93, 143, 191,
         2, 131,  47, 214, 150, 155,   4, 140,  96,   9,   3, 109,  78,
       233, 167, 211,  39,  27, 245, 219,   6,  87, 259, 285, 210, 208,
       235,  69, 270, 278,  70, 144, 147,   0, 310, 111, 212,  38,  28,
       255,  59,  55, 138,  45, 172,  13, 112, 261, 113, 298, 290, 153,
       187,  92, 254,  99, 162, 281, 282, 189, 252,  26, 269, 256, 218,
       116, 314,  23,  61,  29, 304, 287, 149,  52, 253, 232, 200, 204,
       258, 197, 103, 216,  37, 173,  81, 262, 178, 225,  25, 133, 122,
       168, 240, 272,  17, 198, 158,  68, 199, 271,  16,  21,  71, 242,
       119,  32,  91, 128, 110,  62,  88, 181,  36, 284, 229, 308, 244,
        66, 238, 264, 100, 303,  40,  50, 277, 312, 295,  44, 161, 222,
        56, 159,  34, 193, 184, 230,  11, 250, 182, 101, 142, 132,  63,
        31,  54,  51, 276, 164, 183, 289, 288,  30, 154, 248,  20,  74,
       160,  90,  57, 226, 286, 313, 102,  95,   5, 275, 146, 246, 127,
       247, 215, 157, 237, 236,  76, 260,  65,  79, 224, 203, 306, 292,
        82, 136, 268])
```

```
data['POSTED_BY'] = label_encoder.fit_transform(data['POSTED_BY'])
```

```
data['BHK_OR_RK'] = label_encoder.fit_transform(data['BHK_OR_RK'])
```

```
data.CITY_TIER = label_encoder.fit_transform(data.CITY_TIER)
```

```
data.head()
```

```
   POSTED_BY  UNDER_CONSTRUCTION  RERA  BHK_NO.  BHK_OR_RK    SQUARE_FT  \
0          2                   0     0      2.0          0  1300.236407
1          1                   0     0      2.0          0  1275.000000
2          2                   0     0      2.0          0   933.159722
3          2                   0     1      2.0          0   929.921143
4          1                   1     0      2.0          0   999.009247

   READY_TO_MOVE  RESALE  ADDRESS  is_train  CITY_TIER
0              1       1       24         1          0
1              1       1      196         1          1
2              1       1       24         1          0
3              1       1       97         1          1
4              0       1      166         1          0
```

We'll need to convert categorical features to dummy variables using pandas! Otherwise our machine learning algorithm won't able to take those features as inputs directly

```
dummy = pd.get_dummies(data)
dummy
```

```
       POSTED_BY  UNDER_CONSTRUCTION  RERA  BHK_NO.  BHK_OR_RK
SQUARE_FT  \
0              2                   0     0      2.0          0
1300.236407
1              1                   0     0      2.0          0
1275.000000
2              2                   0     0      2.0          0
933.159722
3              2                   0     1      2.0          0
929.921143
4              1                   1     0      2.0          0
999.009247
...          ...                 ...   ...      ...        ...
...
98166          1                   0     1      2.0          0
856.555505
98167          1                   0     1      3.0          0
2304.147465
98168          1                   1     1      1.0          0
2525.927453
98169          1                   0     0      2.0          0
1173.708920
98170          1                   0     0      3.0          0
2439.532944

       READY_TO_MOVE  RESALE  ADDRESS  is_train  CITY_TIER
0                  1       1       24         1          0
1                  1       1      196         1          1
2                  1       1       24         1          0
3                  1       1       97         1          1
```

```
4                  0       1      166         1           0
...              ...     ...      ...       ...         ...
98166              1       1      180         0           0
98167              1       1      190         0           2
98168              0       0      180         0           0
98169              1       1      234         0           0
98170              1       1      194         0           0

[98171 rows x 11 columns]
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
model = scaler.fit(data)

scaled_data = model.transform(data)

scaled_data1=pd.DataFrame(scaled_data)

scaled_data1.describe()
```

```
                  0             1             2             3             4  \
count  98171.000000  98171.000000  98171.000000  98171.000000  98171.000000
mean       0.670483      0.177517      0.316947      0.393276      0.000835
std        0.257464      0.382107      0.465289      0.226425      0.028889
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.500000      0.000000      0.000000      0.285714      0.000000
50%        0.500000      0.000000      0.000000      0.285714      0.000000
75%        1.000000      0.000000      1.000000      0.571429      0.000000
max        1.000000      1.000000      1.000000      1.000000      1.000000

                  5             6             7             8             9  \
count  98171.000000  98171.000000  98171.000000  98171.000000  98171.000000
mean       0.500660      0.822483      0.932322      0.461281      0.299997
std        0.211655      0.382107      0.251194      0.255043      0.458259
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.356061      1.000000      1.000000      0.191083
```

```
0.000000
50%        0.464967      1.000000      1.000000      0.554140
0.000000
75%        0.613636      1.000000      1.000000      0.617834
1.000000
max        1.000000      1.000000      1.000000      1.000000
1.000000

                 10
count   98171.000000
mean        0.382588
std         0.380872
min         0.000000
25%         0.000000
50%         0.500000
75%         0.500000
max         1.000000
```

Train-Test Split

```
x = data.drop('is_train',axis = 1).values
y = data['is_train'].values

from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test =
train_test_split(x,y,test_size=0.2,random_state=5)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

model=LogisticRegression()

model.fit(x_train,y_train)

LogisticRegression()

predicted_y=model.predict(x_test)

abc = model.score(x_test,y_test)

from sklearn.metrics import confusion_matrix

confusion_matrix(y_test,predicted_y)

array([[13755,     0],
       [ 5880,     0]], dtype=int64)

from sklearn.metrics import
precision_score,recall_score,f1_score,classification_report

precision_score(y_test, predicted_y, average=None)
```

```
C:\Users\senth\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

array([0.70053476, 0.        ])

recall_score(y_test, predicted_y, average=None)

array([1., 0.])

f1_score(y_test, predicted_y, average=None)

array([0.82389937, 0.        ])

pd.DataFrame(classification_report(y_test,predicted_y,output_dict =
True,zero_division = 1)).T  #zeo_division for removing warning

              precision    recall  f1-score       support
0              0.700535  1.000000  0.823899  13755.000000
1              1.000000  0.000000  0.000000  5880.000000
accuracy       0.700535  0.700535  0.700535  0.700535
macro avg      0.850267  0.500000  0.411950  19635.000000
weighted avg   0.790214  0.700535  0.577170  19635.000000
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators = 1000,random_state = 42)

model.fit(x_train,y_train)

RandomForestClassifier(n_estimators=1000, random_state=42)

predict = model.predict(x_test)

mno = model.score(x_test,y_test)

confusion_matrix(y_test,predict)

array([[10376,  3379],
       [ 4367,  1513]], dtype=int64)

precision_score(y_test, predict, average=None)

array([0.70379163, 0.30928046])

recall_score(y_test, predict, average=None)

array([0.75434387, 0.25731293])

f1_score(y_test, predict, average=None)
```

```
array([0.72819145, 0.28091348])
```

```
pd.DataFrame(classification_report(y_test,predict,output_dict = True)).T
```

	precision	recall	f1-score	support
0	0.703792	0.754344	0.728191	13755.0000
1	0.309280	0.257313	0.280913	5880.0000
accuracy	0.605500	0.605500	0.605500	0.6055
macro avg	0.506536	0.505828	0.504552	19635.0000
weighted avg	0.585649	0.605500	0.594247	19635.0000

ADABOOSTING CLASSIFIER

```
from sklearn.ensemble import AdaBoostClassifier

model = AdaBoostClassifier(n_estimators = 1000,random_state = 42)

model.fit(x_train,y_train)

AdaBoostClassifier(n_estimators=1000, random_state=42)

predict_y = model.predict(x_test)

xyz = model.score(x_test,y_test)

confusion_matrix(y_test,predict_y)

array([[13746,     9],
       [ 5873,     7]], dtype=int64)

precision_score(y_test, predict_y, average=None)

array([0.70064733, 0.4375    ])

recall_score(y_test, predict_y, average=None)

array([0.99934569, 0.00119048])

f1_score(y_test, predict_y, average=None)

array([0.82375502, 0.00237449])

pd.DataFrame(classification_report(y_test,predict_y,output_dict = True)).T
```

	precision	recall	f1-score	support
0	0.700647	0.999346	0.823755	13755.000000
1	0.437500	0.001190	0.002374	5880.000000
accuracy	0.700433	0.700433	0.700433	0.700433
macro avg	0.569074	0.500268	0.413065	19635.000000
weighted avg	0.621844	0.700433	0.577780	19635.000000

```
final = [abc,mno,xyz]
table = [['LOGISTIC REGRESSION',abc],['RANDOM FOREST
CLASSIFIER',mno],['ADA BOOSTING CLASSIFIER',xyz]]
print(table)
```

```
[['LOGISTIC REGRESSION', 0.7005347593582888], ['RANDOM FOREST CLASSIFIER', 0.7005347593582888], ['ADA BOOSTING CLASSIFIER', 0.7005347593582888]]
```

```
print(table[final.index(max(final))]," IS THE BEST FIT!")
```

```
['LOGISTIC REGRESSION', 0.7005347593582888]  IS THE BEST FIT!
```

CHAPTER 6

REFERENCES

1.Ada Boosting Classifier

https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/

2.Random Forest Classifier

https://www.javatpoint.com/machine-learning-random-forest-algorithm

3.Classification of Indian Cities

Classification of Indian cities - Wikipedia