Real Estate Price Prediction Model

Minor project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology in

Computer Science and Engineering

By

Ananya Mishra (191213)

Prishita Singh (191223)

Under the Supervision of

Dr. Jagpreet Sidhu

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

Table of Contents

Declaration

We declare that the work supplied in the project report titled "Real Estate Price Prediction and Sales Insights Analyser" presented by Ananya Mishra (191213) and Prishita Singh(191223) in partial fulfillment of the requirements for the grant of the degree of Bachelor of Technology in Computer Science and Engineering is an authentic record of our own work carried out during a period from January 2022 to May 2022 under the supervision of **Dr. Jagpreet Sidhu** (Assistant Professor(SG) – CSE & IT).

We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by

Dr. Jagpreet Sidhu

Assistant Professor (SG)

Department of Computer Science and Engineering & Information Technology

Submitted by

Ananya Mishra	Prishita Singh
191213	191223

Certificate

This is to confirm that the work provided in the project report titled "Real Estate Price Prediction Model" is in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering filed in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an accurate record of our own work done under the supervision of Dr. Jagpreet Sidhu (Assistant Professor – CSE & IT) from January to May 2022

Ananya Mishra
191213

Prishita Singh
191223

The above statement is correct to the best of my knowledge.

Dr. Jagpreet Sidhu
Assistant Professor(SG)
Department of Computer Science and Engineering & Information Technology

Acknowledgement

Firstly, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing that made it possible for us to complete the project work successfully.

I am really grateful and wish my profound indebtedness to **Dr. Jagpreet Sidhu,** Assistant Professor(SG), Department of CSE Jaypee University of Information Technology, Wakhnaghat. His deep knowledge & keen interest in the field of Data Science helped us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, and reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would also generously thank each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. I would also like to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Ananya Mishra(191213)
Prishita Singh(191223)

Abstract

The project “Real Estate Price Prediction Model” is a prediction model that predicts the price of a flat or building in the city of Bangalore. The project uses the techniques of machine learning to build the model. The project shall be presented to the customer in the form of a web page offering the selection constraints like the locality, number of bedrooms, and sq. ft. area.

The customer is made to select the specifications of his future flat and as per the entire selection list, the model predicts the price of the flat.

The project is a prediction model. It uses the techniques of machine learning and data analysis.

Chapter 1: Introduction

1.1 Introduction

"Real Estate Price Prediction Model" is a consumer-friendly website that will help the user to predict the price of real estate in the city of Bangalore depending on the specifications of the area, location, and other constraints. The model is designed keeping in mind the business scenario and the demand of the market. The consumer would be able to choose his flat or building using the website and when the sales are conducted, the property dealer shall be able to analyze the sales himself without making consistent calls to his data analyst every third day.

The prediction model is built using machine learning techniques and the mathematical skills of regression.

1.2 Objectives

The objectives of the project are discussed below:

i. Implementing our knowledge of the basic concepts of machine learning and data science into a project

ii. Creating a consumer-friendly and convenient utility of the model in the form of a webpage

iii. Creating an easy-to-handle platform for the property dealers to analyze the sales insights, since they might not have any prior technical background

1.3 Motivation of the Minor Project

The model is designed keeping in mind the business scenario and the demand of the market. The customers demand ease of searching the desired location with their other specifications and wish to know how much it would cost if they purchase the property within an area. The model offers a convenient and efficient way of real estate price prediction to the customers. Moreover, many times the property dealers do not have a technical background and hire a data scientist to deal with the analysis of the sales. Hence, this model shall help the property dealers to analyze the sales themselves without making consistent calls to the data analyst every third day and also to keep a check on the authenticity of his work.

1.4 Language Used

The machine learning model is designed using python due to the availability of numerous machine learning libraries and methods.

1.5 Technologies Used

Following are the technologies and software used in the project:

1.Python PyCharm

2.Anaconda

3.NumPy and Pandas (Data Cleaning)

4.Matplotlib (Data Visualization)

5.Sklearn for model building

6.Python flask for http server

7.Notepad++

8.HTML/CSS/Javascript for UI

1.6 Deliverables of the Minor Project

Designing the regression model and using the techniques of web development we shall finally have a web page ready for the consumer to predict the price of his desired flat or mansion in a specific locality in the city of Bangalore. The outcomes of the regression model accuracy shall be visualized and displayed.

Chapter 2: Minor Project SDLC

2.1 Feasibility Study : To evaluate the project's potential, a feasibility analysis over different constraints is put forward.

i. Technical Feasibility: The project "Real Estate Price Prediction Model" is a technically feasible model. Using the specific technologies, the model shall work up to the user's expectations. The model has created a consumer-friendly UI and uses Anaconda which is the most feasible software of the regression models. All the technologies used are stable.

ii. Operational Feasibility: The project is an operational feasible model and would benefit the customers and the sellers at the same time. It is a user-friendly model. The model is easy to operate and functions to display the authentic and true outcomes.

iii. Scheduling Feasibility: The project has been completed as per the schedule and meets the constraints of the schedule feasibility for a model.

iv. Market Feasibility: The project has particularly been designed to meet the market demands of an easy and manageable platform for the customer to explore the real estate and the dealers to analyze their sales timely. It meets the constraints of market feasibility.

v. Economic Feasibility: The software used in the making of the project is free of cost and does not involve any expenses. The website is presently run on a localhost server, in order to publicly make it available in the cyber market we would need to buy the domain for the website, which is an economically feasible expense.

2.2 Requirements on Minor Project

2.2.1 Functional Requirements

The requirements which are fundamentally required and the end-user particularly demands from the system[1].

The functional requirements of the project are listed below:

i. The consumer has a convenient interface to operate.

ii. The property dealer has his system under authentication privacy and password protection.

iii. Once the user/customer fills in the specifications of the flat then the webpage offers a button to submit the information details and predict the price.

2.2.2 Non-Functional Requirements

The requirements of the project are not mandatory but enhance the quality of the project[1].
A few non-functional requirements of the project are shared below:

i. The credentials of the customer are not asked or shared before he starts his dealings with the property dealer.

iii. The web page loads and displays the predicted price within a time span of 4-5 seconds.

iii. The model is security feasible taking into consideration the password protection for the property dealers.

2.3 Use Case Diagram of the Minor Project

To graphically depict the interaction of the customer/user with the UI by filling the specifications of the flat and predicting the price, the case diagram is shown below. Additionally, the dealer's interaction with his sales analyser is also presented.

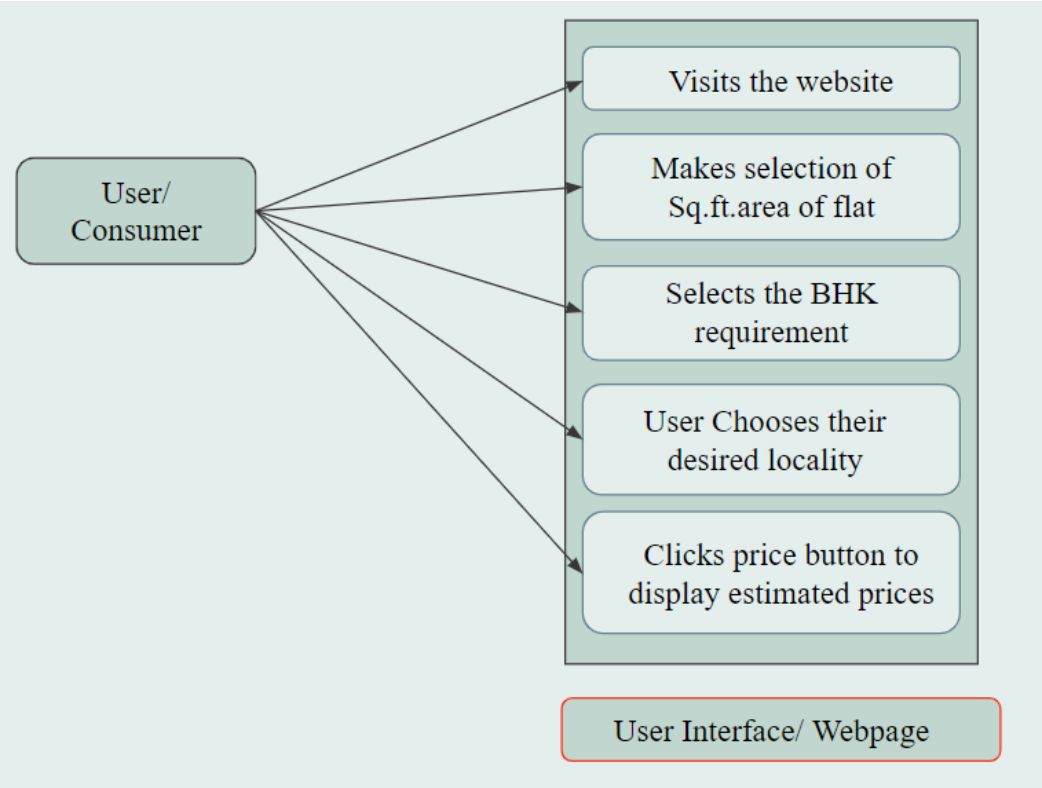

Fig. 1: Use Case Diagram of the project

2.4 DFD Diagram

The Diagram is the visual representation of the data flow in the model. The diagram is the physical and logical flow of data in our model from the user making the server request to predicting the price of the real estate.

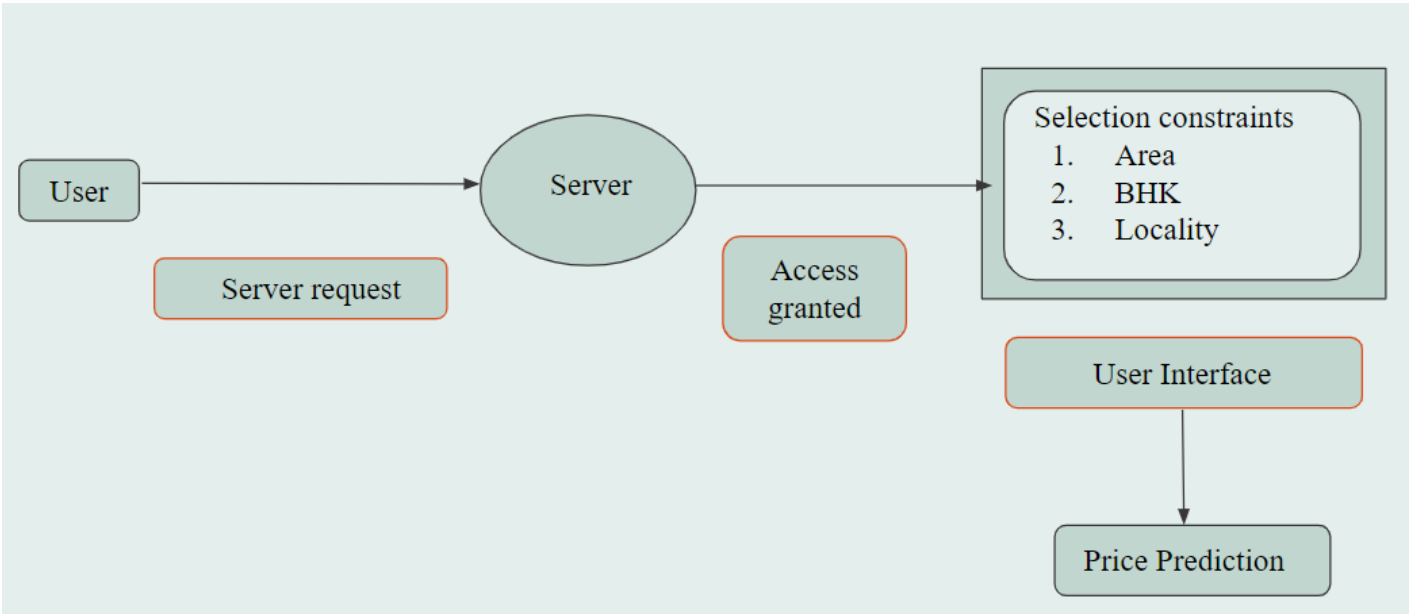

Fig. 2: Data Flow Diagram of the model

2.5 State Transition Diagram

The state diagram below depicts the various states of the model and the simple transitions between them. The diagram shows the behavioral model of how one state stimulates according to the previous states.

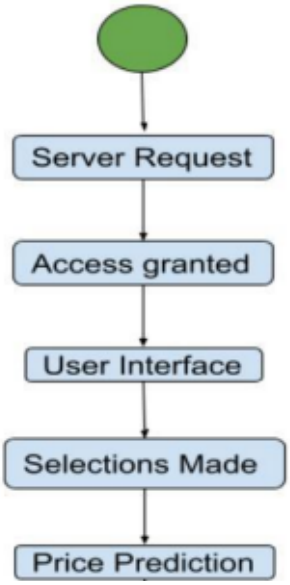

Fig. 3: State Transition Diagram of the model

Chapter 3: Implementation

3.1 Data Set used in the Minor Project

The data set used for building the regression model for the price prediction of the real estate has been imported from Kaggle. The data set is the data of Bangalore city, India. The data set is a CSV file. It consists of various fields and the data size is large which makes it easy to handle and make operations on the data.[2]

The various column fields of the data set are area type, locality, number of bedrooms, total sq. ft. area, price, etc. This data set is cleaned and thereafter used for building the regression model.[3]

	area_type	availability	location	size	society	total_sqft	bath	balcony	p
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	12
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	
5	Super built-up Area	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2.0	1.0	
6	Super built-up Area	18-May	Old Airport Road	4 BHK	Jaades	2732	4.0	NaN	2
7	Super built-up Area	Ready To Move	Rajaji Nagar	4 BHK	Brway G	3300	4.0	NaN	6
8	Super built-up Area	Ready To Move	Marathahalli	3 BHK	NaN	1310	3.0	1.0	
9	Plot Area	Ready To Move	Gandhi Bazar	6 Bedroom	NaN	1020	6.0	NaN	3
10	Super built-up Area	18-Feb	Whitefield	3 BHK	NaN	1800	2.0	2.0	
11	Plot Area	Ready To Move	Whitefield	4 Bedroom	Prrry M	2785	5.0	3.0	2
12	Super built-up Area	Ready To Move	7th Phase JP Nagar	2 BHK	Shncyes	1000	2.0	1.0	
13	Built-up Area	Ready To Move	Gottigere	2 BHK	NaN	1100	2.0	2.0	
14	Plot Area	Ready To Move	Sarjapur	3 Bedroom	Skityer	2250	3.0	2.0	1

Fig. 4: Dataset used in the model

3.2 Data Set Features

3.2.1 Types of Dataset

The dataset used in the project is a combination of various categories:

i. Numerical Data Set: The data set contains various numerical fields for the evaluation, hence the dataset used is numerical data.

ii. Multivariate Data Set: The data set consists of various independent fields which collectively determine the price of real estate.

iii. Correlation Data Set: The data set contains the two fields namely, total_sqft and size, which depend on each other. The data is unstructured so there might exist uncorrelated rows, which will be structured to display correlation after data cleaning.

3.2.2 Number of attributes, fields, description of data set

The attributes in our data set can be categorized into the following types:

i. Quantitative attributes: The data set has

1. Numeric attributes: In the data set the attributes namely, total_sqft, bath, etc. are the numeric attributes.

2. Discrete attributes: The data set consists of attributes like bath, balcony, etc. have discrete values.

3. Continuous attributes: In the data set, certain attributes like price acquire continuous values.

ii. Qualitative attributes: The data set that we have used only has a single category of qualitative attributes, i.e., nominal attributes. In the data set the attributes like area_type, and location has nominal values (related to names).

The unstructured dataset of our project contains 9 fields/attributes.

Please note in the database, the field refers to the data member and attribute is another term reference for the field to be used publicly.

The data set is the data collection of the real estate of Bangalore city, India. The dataset has been deployed from Kaggle. The dataset initially is unstructured and contains plenty of fields. The size of the dataset is nearly 13000 rows. The large size and variety of fields of the dataset help in the evaluation and better data cleaning. The dataset is cleaned and used to build the regression model for the price prediction of the real estate in Bangalore.

3.3 Design of Problem Statement

To build a model and design a platform that helps to predict the price of the real estate. The model additionally must help the property dealer to analyze the sales in a visualized manner.

3.4 Algorithm/Pseudocode of the model

The project is a regression-based model. To design the regression model is not a tedious task. The most important part is the data cleaning of the dataset. The dataset is unstructured data deployed randomly from an online platform.

1. The dataset is read in the ipynb notebook.

2. The data is cleaned. (*data cleaning shall be discussed in the stages of the project since it is not part of the algorithm of the model*)

3. Building of the regression model - The regression model is built using the sklearn train test split.

```
X = dataframe.drop(['price'],axis='columns') #x is all the independent variables  y = df12.price

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=10)
```

4. Linear Regression is used to design the price prediction model[4][5].

```
from sklearn.linear_model import LinearRegression

lr_clf = LinearRegression()
```

5. The efficiency of the model is evaluated after fitting in the model.[5]

```
lr_clf.fit(X_train,y_train)

lr_clf.score(X_test,y_test)
```

6. This model is then tested on certain specific data.

7. The tested model is exported to a pickle file that shall further help the flask server to generate the model request.

8. The python flask server imports the tested model file and creates a server request to display it on http online portal.

9. The web page is designed using the web development techniques (HTML, CSS, Java script) in order to offer the customer a better User Interface to explore.

3.5 Flow Graph of the model/project

The flow graph of the project shows the various stages of the project.

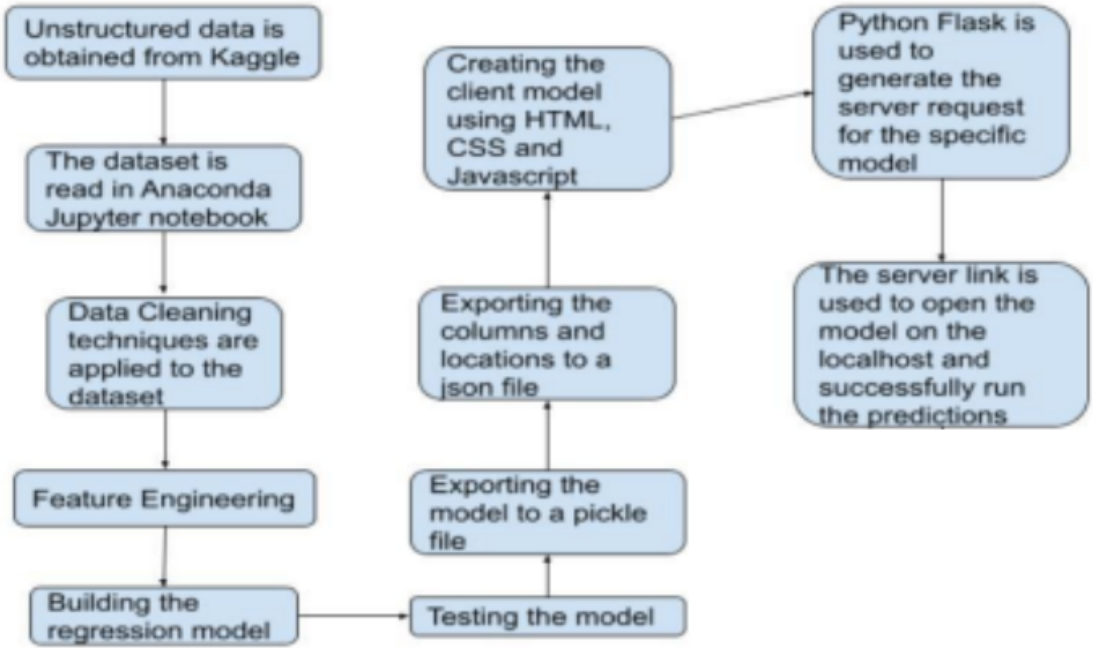

Fig. 5: Flow diagram of the model

3.6 Screenshots of the various stages of the project

1. Libraries are imported

```
1 import pandas as pd
2 import numpy as np
3 from matplotlib import pyplot as plt
4 %matplotlib inline
5 import matplotlib
6 matplotlib.rcParams["figure.figsize"] = (20,10)
```

2. The data is loaded into model

```
df1= pd.read_csv("Bengaluru_House_Data.csv")
df1.head()
```

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

```
df1.shape
```

```
(13320, 9)
```

```
df1.groupby('area_type')['area_type'].agg('count')
```

```
area_type
Built-up  Area          2418
Carpet  Area              87
Plot  Area              2025
Super built-up  Area    8790
Name: area_type, dtype: int64
```

3. Data Cleaning

This is the most important part of model designing.

3.1 The features that are not required are dropped

```
df2= df1.drop(['area_type','society','balcony','availability'],axis='columns')
df2.head()
```

	location	size	total_sqft	bath	price
0	Electronic City Phase II	2 BHK	1056	2.0	39.07
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00
2	Uttarahalli	3 BHK	1440	2.0	62.00
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00
4	Kothanur	2 BHK	1200	2.0	51.00

3.2 Check for the null values and drop them.

```
df2.isnull().sum()
```

```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

(Checking for null values)

```
df3= df2.dropna()
df3.isnull().sum()
#removed null values
```

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

(dropping null values)

3.3 We observe a unique pattern in the 'size' column of our dataset.

```
df3['size'].unique()
```

```
array(['2 BHK', '4 Bedroom', '3 BHK', '4 BHK', '6 Bedroom', '3 Bedroom',
       '1 BHK', '1 RK', '1 Bedroom', '8 Bedroom', '2 Bedroom',
       '7 Bedroom', '5 BHK', '7 BHK', '6 BHK', '5 Bedroom', '11 BHK',
       '9 BHK', '9 Bedroom', '27 BHK', '10 Bedroom', '11 Bedroom',
       '10 BHK', '19 BHK', '16 BHK', '43 Bedroom', '14 BHK', '8 BHK',
       '12 Bedroom', '13 BHK', '18 Bedroom'], dtype=object)
```

The field contains attribute values with varying units labeled as 'BHK', and 'Bedroom', and the machine learning algorithms shall consider them as different entities. So this needs to be worked on. We shall work on the technique of "**Feature Engineering**". We would add a new feature 'BHK' which will replace this field.

```
df3['bhk']= df3['size'].apply(lambda x: int(x.split(' ')[0]))
# the un-uniformity of bhk column was removed
```

```
df3.head()
```

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600	5.0	120.00	4
2	Uttarahalli	3 BHK	1440	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521	3.0	95.00	3
4	Kothanur	2 BHK	1200	2.0	51.00	2

After this feature has been added to our dataset, we need to improve it more. We made yet another observation, the feature field 'BHK' contains certain values more than 20, which is practically not a feasible flat to be purchased by a normal living family.

```
1 df3['bhk'].unique()
```

```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18])
```

```
1 df3[df3.bhk>20]
```

	location	size	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	27 BHK	8000	27.0	230.0	27
4684	Munnekollal	43 Bedroom	2400	40.0	660.0	43

Secondly, there were certain values displayed in the form of intervals and it is very clear that range values are not accepted. The dataset demands accurate numeric values in fields like "BHK".

```
1 df3.total_sqft.unique()
```

```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

```
1 #checking how many float values are there
2 def is_float(x):
3     try:
4         float(x)
5     except:
6         return False
7     return True
```

```
1 df3[-df3['total_sqft'].apply(is_float)].head(5)
```

	location	size	total_sqft	bath	price	bhk
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2

3.4 The next observation was made in the field of "total_sqft", it contains certain range values and a few values with different units. Since we have a large size of data, we ignore the rows with the different units since the number of such values is very low. However, the range values need a solution. We prefer to take the middle value of the interval in the case of range values.

```
def convert_sqft_to_num(x):
    tokens=x.split('-')
    if len(tokens) ==2:
        return (float(tokens[0])+float(tokens[1]))/2 #take avg
    try:
        return float(x)
    except:
        return None
```

Finally, this is what we obtain.

```
df4= df3.copy()
df4['total_sqft'] = df4['total_sqft'].apply(convert_sqft_to_num)
df4.head(5)
```

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3
4	Kothanur	2 BHK	1200.0	2.0	51.00	2

3.5 Feature Engineering

We will introduce another feature "price_per_sqft". The major reason behind this feature engineering technique is to make the predictions better and fast. It evaluates the influencing constraints and predicts the price using the price per sq. ft. area.

```
df5= df4.copy()
df5['price_per_sqft']= df5['price']*100000/df5['total_sqft']
df5.head()
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000

3.6 Dimensionality reduction

We check the number of columns.

```
len(df5.location.unique())
```

```
1304
```

Since the number of columns is 1304, which is very high. Such a high number of columns is known as the dimensionality curse, which is a problem. To resolve this, we shall combine the locations having less than 10 data points into a single value.

We then check the number of data points for each location.

```
df5.location= df5.location.apply(lambda x: x.strip()) #removes space at beg or end
location_stats = df5.groupby('location')['location'].agg('count').sort_values(ascending=False)
location_stats
```

```
location
Whitefield               535
Sarjapur  Road           392
Electronic City          304
Kanakpura Road           266
Thanisandra              236
                        ...
1 Giri Nagar               1
Kanakapura Road,           1
Kanakapura main  Road      1
Karnataka Shabarimala      1
whitefiled                 1
Name: location, Length: 1293, dtype: int64
```

The number of locations with less than 10 data points is found.

```
1 len(location_stats[location_stats<=10])
```

```
1052
```

```
1 #locations having less than 10 data points
2 location_stats_less_than_10 = location_stats[location_stats<=10]
3 location_stats_less_than_10
```

```
location
Basapura                 10
1st Block Koramangala    10
Gunjur Palya             10
Kalkere                  10
Sector 1 HSR Layout      10
                         ..
1 Giri Nagar              1
Kanakapura Road,          1
Kanakapura main  Road     1
Karnataka Shabarimala     1
whitefiled                1
Name: location, Length: 1052, dtype: int64
```

All the locations that have less than 10 data points are now combined in a new location value called other.

```
1 df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
2 #so all the less than 10 locations will be converted into a new category - other
3 len(df5.location.unique())
```

```
242
```

+ Code + Text

```
1 df5.head(10)
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2	3699.810606
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4	4615.384615
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3	4305.555556
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3	6245.890861
4	Kothanur	2 BHK	1200.0	2.0	51.00	2	4250.000000
5	Whitefield	2 BHK	1170.0	2.0	38.00	2	3247.863248
6	Old Airport Road	4 BHK	2732.0	4.0	204.00	4	7467.057101
7	Rajaji Nagar	4 BHK	3300.0	4.0	600.00	4	18181.818182
8	Marathahalli	3 BHK	1310.0	3.0	63.25	3	4828.244275
9	other	6 Bedroom	1020.0	6.0	370.00	6	36274.509804

3.7 Outliers Removal Using Business Logic

Check above data points. We have a 6 bhk apartment with 1020 sqft. Another one is 8 bhk and total sqft is 600. These are clear data errors that can be removed safely

```
1 df5.shape
```

```
(13246, 7)
```

```
1 df5[(df5.total_sqft/df5.bhk<300)]
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
9	other	6 Bedroom	1020.0	6.0	370.0	6	36274.509804
45	HSR Layout	8 Bedroom	600.0	9.0	200.0	8	33333.333333
58	Murugeshpalya	6 Bedroom	1407.0	4.0	150.0	6	10660.980810
68	Devarachikkanahalli	8 Bedroom	1350.0	7.0	85.0	8	6296.296296
70	other	3 Bedroom	500.0	3.0	100.0	3	20000.000000
...	...	...	...	...	...	...	...
13277	other	7 Bedroom	1400.0	7.0	218.0	7	15571.428571
13279	other	6 Bedroom	1200.0	5.0	130.0	6	10833.333333
13281	Margondanahalli	5 Bedroom	1375.0	5.0	125.0	5	9090.909091
13303	Vidyaranyapura	5 Bedroom	774.0	5.0	70.0	5	9043.927649
13311	Ramamurthy Nagar	7 Bedroom	1500.0	9.0	250.0	7	16666.666667

744 rows × 7 columns

Removal of outliers using mean and standard deviation.

```
1 df6 = df5[~(df5.total_sqft/df5.bhk<300)]
2 df6.shape
3 #checking the size of each bedroom
```

```
(12502, 7)
```

```
1 #checking price per sqft
2 df6.price_per_sqft.describe()
```

```
count     12456.000000
mean       6308.502826
std        4168.127339
min         267.829813
25%        4210.526316
50%        5294.117647
75%        6916.666667
max      176470.588235
Name: price_per_sqft, dtype: float64
```

```
# removing outliers using sd and mean
# removing outliers per location using mean and one standard deviation
def remove_pps_outliers(df):
    df_out= pd.DataFrame()
    for key, subdf in df.groupby('location'):    #grouping by loaction
        m=np.mean(subdf.price_per_sqft)   #calculating mean
        st=np.std(subdf.price_per_sqft)   #calculating sd
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
        df_out=pd.concat([df_out,reduced_df],ignore_index=True)   #apending per loc
    return df_out
```

```
df7 = remove_pps_outliers(df6)
df7.shape
```

```
(10241, 7)
```

Before Removing Outliers

We shall check if the price for a 3bhk flat is more than that of a 2bhk flat for the same square-foot area.

```
#check if price for 3bhk>2bhk for same squarefoot area
#we need a scatterplot to give us the visualization how many such cases exist
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', color='green',label='3 BHK', s=50)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh INR)")
    plt.title(location)
    plt.legend()

plot_scatter_chart(df7,"Rajaji Nagar")

```

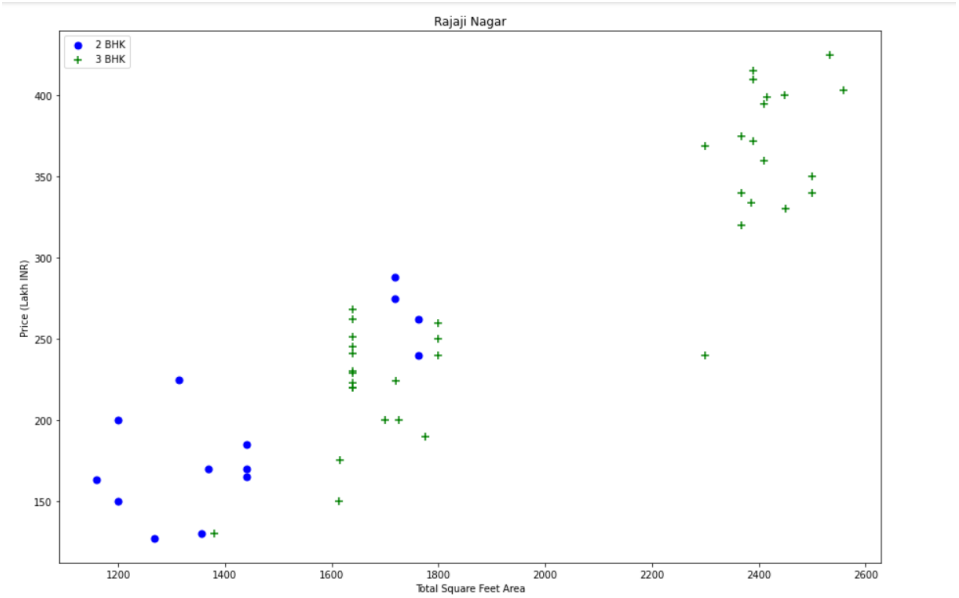

Fig. 6: Plot of variation of price per square feet with total square feet area with outliers

The major outliers have been observed that for the same sq. ft area, 2 BHK price is more than 3 BHK.

After removing outliers:

```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                'std': np.std(bhk_df.price_per_sqft),
                'count': bhk_df.shape[0]
            }
        for bhk, bhk_df in location_df.groupby('bhk'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.price_per_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices,axis='index')

df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
```

```
(7329, 7)
```

```
plot_scatter_chart(df8,"Rajaji Nagar")
```

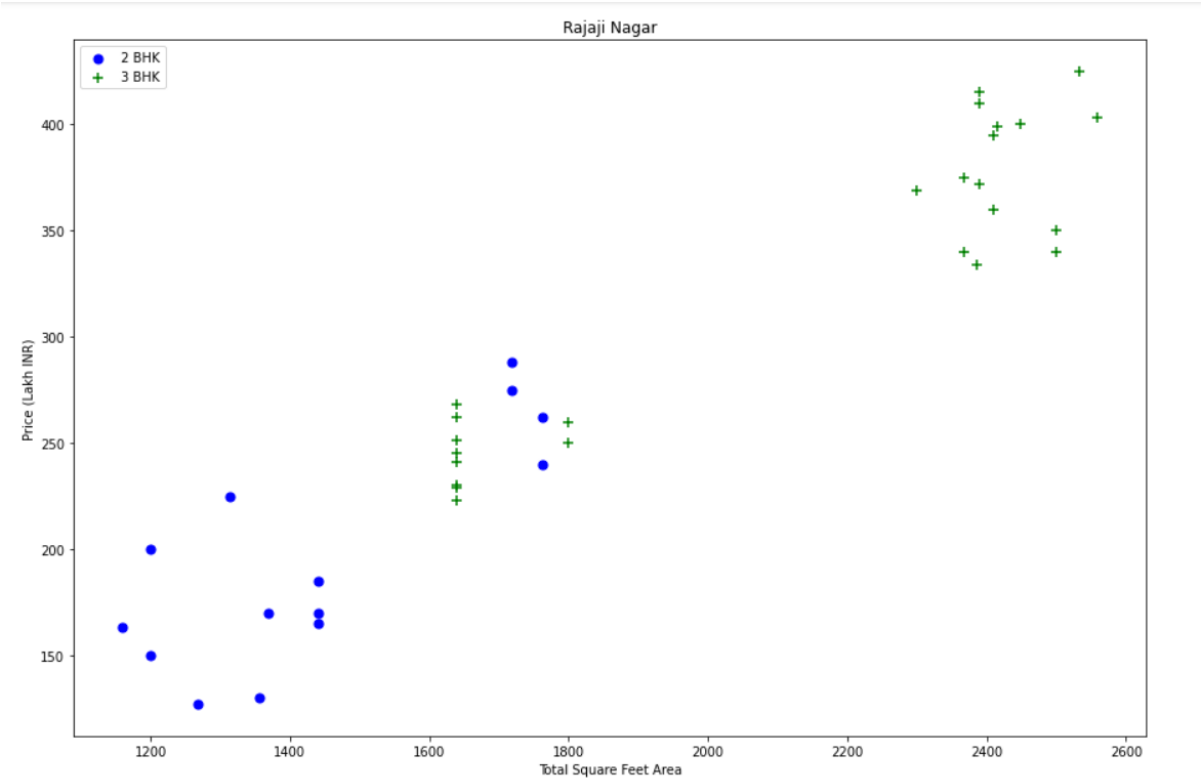

Fig. 7: Plot of the variation of price per square feet with total square feet area with outliers removal

To visualize the count of flats with the price per square feet area, we prefer to use histogram:

```
1 import matplotlib
2 matplotlib.rcParams["figure.figsize"] = (20,10)
3 plt.hist(df8.price_per_sqft,rwidth=0.8)
4 plt.xlabel("Price Per Square Feet")
5 plt.ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```

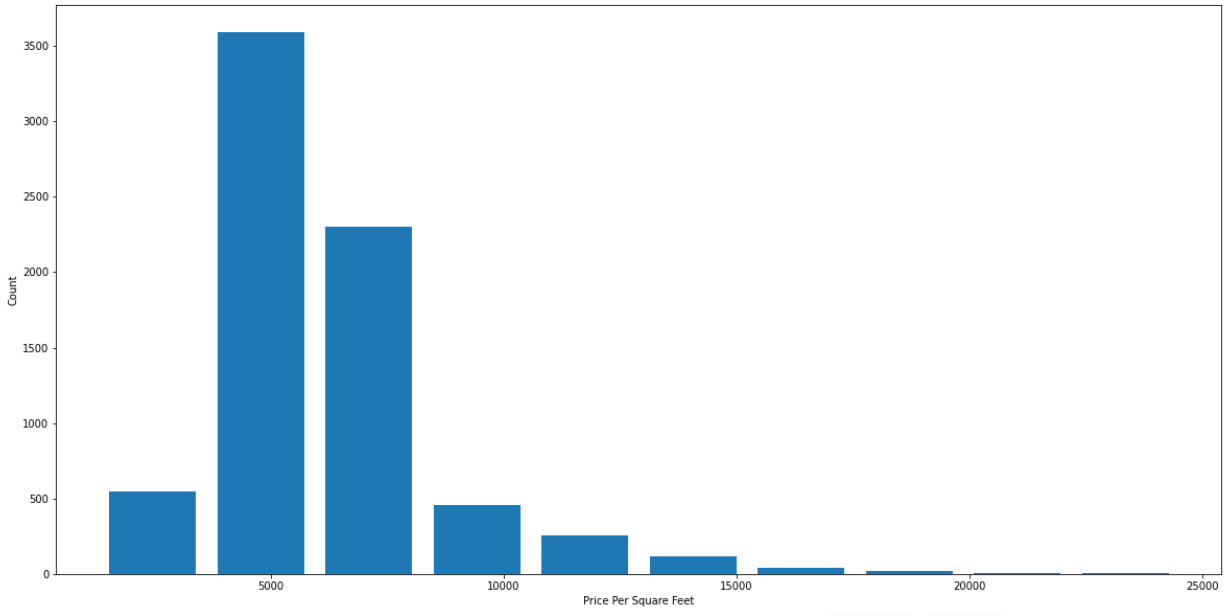

Fig. 8: Plot of the variation of count with price per square feet

From the histogram we can observe that a majority of the data points are in 0 to 10000 sq ft range. Hence, it is a normal distribution.

Next, we will remove the bathroom outliers. It is not practically common to have the number of bathrooms twice more than the number of bedrooms.

So, we shall plot a histogram to check the number of bathrooms.

```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

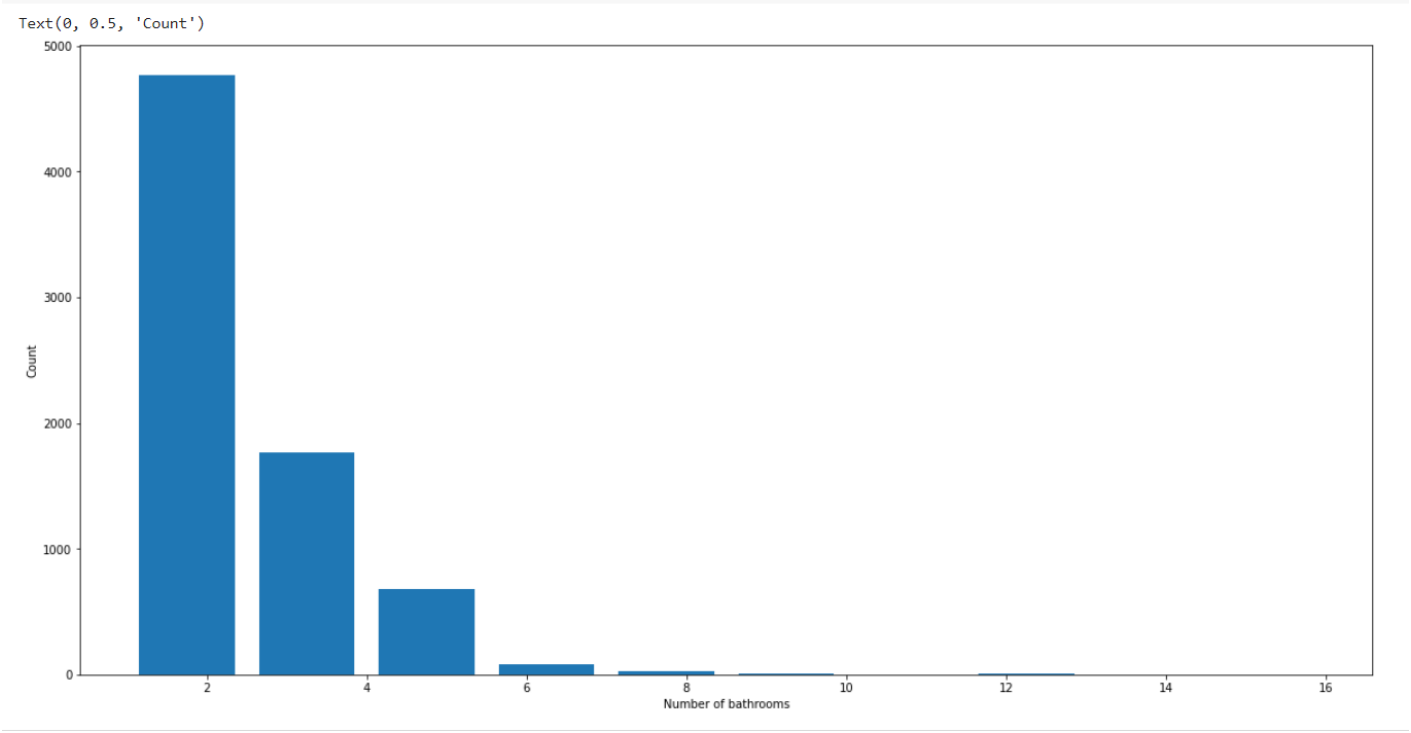

Fig. Plot of Number of bathrooms vs count

Hence, the data points where the number of bathrooms > the number of bedrooms +2, will be removed.

```
df8[df8.bath>df8.bhk+2]
#where number of bathrooms are greater than number of (bedrooms +2)  ->outliers
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1626	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5238	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6711	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8411	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

```
#removing bathroom outliers
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

```
(7251, 7)
```

```
df9.head(5)
```

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668

Lastly we drop the price_per_sqft column as it was needed only for outlier detection.

```
df10 = df9.drop(['size','price_per_sqft'],axis='columns')
df10.head(3)
#we dropped size as the bhk column did similar work
# we dropped price persqft as that was essential only for outlier detection
```

	location	total_sqft	bath	price	bhk
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3

3.8 One Hot Encoding

Machine Learning Model Cannot Interpret Text Data, So we use dummies for locations.

```
dummies = pd.get_dummies(df10.location)
dummies.head()
```

	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	...	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenahalli
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 242 columns

```
#appending dummies dataset to df
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenahalli
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	0	0	0	0	0	0
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0	...	0	0	0	0	0	0
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0	...	0	0	0	0	0	0
3	1st Block Jayanagar	1200.0	2.0	130.0	3	1	0	0	0	0	...	0	0	0	0	0	0
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...	0	0	0	0	0	0

5 rows × 246 columns

```
#dropping last col (here 'others') to avoid dummy variable trap
df12 = df11.drop('location',axis='columns')
df12.head()
```

	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenahalli
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	0	0	0	0	0
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	0	0	0	0	0
2	1875.0	2.0	235.0	3	1	0	0	0	0	0	...	0	0	0	0	0	0
3	1200.0	2.0	130.0	3	1	0	0	0	0	0	...	0	0	0	0	0	0
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	0	0	0	0	0	0

5 rows × 245 columns

The data is cleaned using the above techniques, now the next stage is to build the regression model.

4. Building Model

Price column is dropped and the entire list of independent variables is stored in X and price in Y. The linear regression model will design and build our model using sklearn.

```
1 X = df12.drop(['price'],axis='columns')   #independent variables
2 X.head(3)
```

	total_sqft	bath	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	0	0	0
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	0	0	0
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0

3 rows × 244 columns

```
1 X.columns
```

```
Index(['total_sqft', 'bath', 'bhk', '1st Block Jayanagar',
       '1st Phase JP Nagar', '2nd Phase Judicial Layout',
       '2nd Stage Nagarbhavi', '5th Block Hbr Layout', '5th Phase JP Nagar',
       '6th Phase JP Nagar',
       ...
       'Vijayanagar', 'Vishveshwarya Layout', 'Vishwapriya Layout',
       'Vittasandra', 'Whitefield', 'Yelachenahalli', 'Yelahanka',
       'Yelahanka New Town', 'Yelenahalli', 'Yeshwanthpur'],
      dtype='object', length=244)
```

```
1 y = df12.price   #dependent variable
2 y.head(3)
```

```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

```
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=10)
```

```
# Linear Regression
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

```
0.8452277697874304
```

5. K-Fold Cross Validation to measure the accuracy of our Linear Regression Model

```
# 5 fold cross validation
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0) # randomize the sample

cross_val_score(LinearRegression(), X, y, cv=cv)
# majority of the score is more than 80%
```

```
array([0.82430186, 0.77166234, 0.85089567, 0.80837764, 0.83653286])
```

We can see that in 5 iterations we get a score above 80% most of the time. This is good but we want to test few other algorithms for regression to see if we can get even better score. We will use GridSearchCV for this purpose.

6. GridSearchCV is implemented.

The accuracy of various datasets is then checked with GridSearchCV.

The machine learning models used here are Linear Regression, Lasso Regression, and Decision Tree. GridSearchCV provides us with the model accuracy and the parameters that should be used to get the accuracy.

```
# to run model on diffrent models
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
# gird serach cv will select the best algorithm and the best parameters - called hyperparameter tuning
def find_best_model_using_gridsearchcv(X,y):
    algos = {
        'linear_regression' : {
            'model': LinearRegression(),
            'params': {
                'normalize': [True, False]
            }
        },
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1,2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion' : ['mse','friedman_mse'],
                'splitter': ['best','random']
            }
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    for algo_name, config in algos.items():
        gs =  GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores,columns=['model','best_score','best_params'])

find_best_model_using_gridsearchcv(X,y)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_base.py:145: FutureWarning: 'normalize' was deprec
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing stage. To reproduce the pr

from sklearn.pipeline import make_pipeline

model = make_pipeline(StandardScaler(with_mean=False), LinearRegression())

If you wish to pass a sample_weight parameter, you need to pass it as a fit parameter to each step of the pipel

kwargs = {s[0] + '__sample_weight': sample_weight for s in model.steps}
model.fit(X, y, **kwargs)
```

	model	best_score	best_params
0	linear_regression	0.818354	{'normalize': False}
1	lasso	0.687431	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.728048	{'criterion': 'mse', 'splitter': 'best'}

Based on the above result we can say that Linear Regression gives the best accuracy score. Hence, we shall use Linear Regression to predict the real estate prices(in lacs.

7. The model is pickled to be used for the generation of server requests in the flask.

```
import pickle
with open('real_estate_price_model.pickle','wb') as f:
  pickle.dump(lr_clf,f)
```

```
import json
columns = {
    'data_columns' : [col.lower() for col in X.columns]
}
with open("columns.json","w") as f:
  f.write(json.dumps(columns))
```

8. Flask server request is generated

```
Starting Python Flask Server For Home Price Prediction...
loading saved artifacts...start
loading saved artifacts...done
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

9. The web page is ready to be operated.

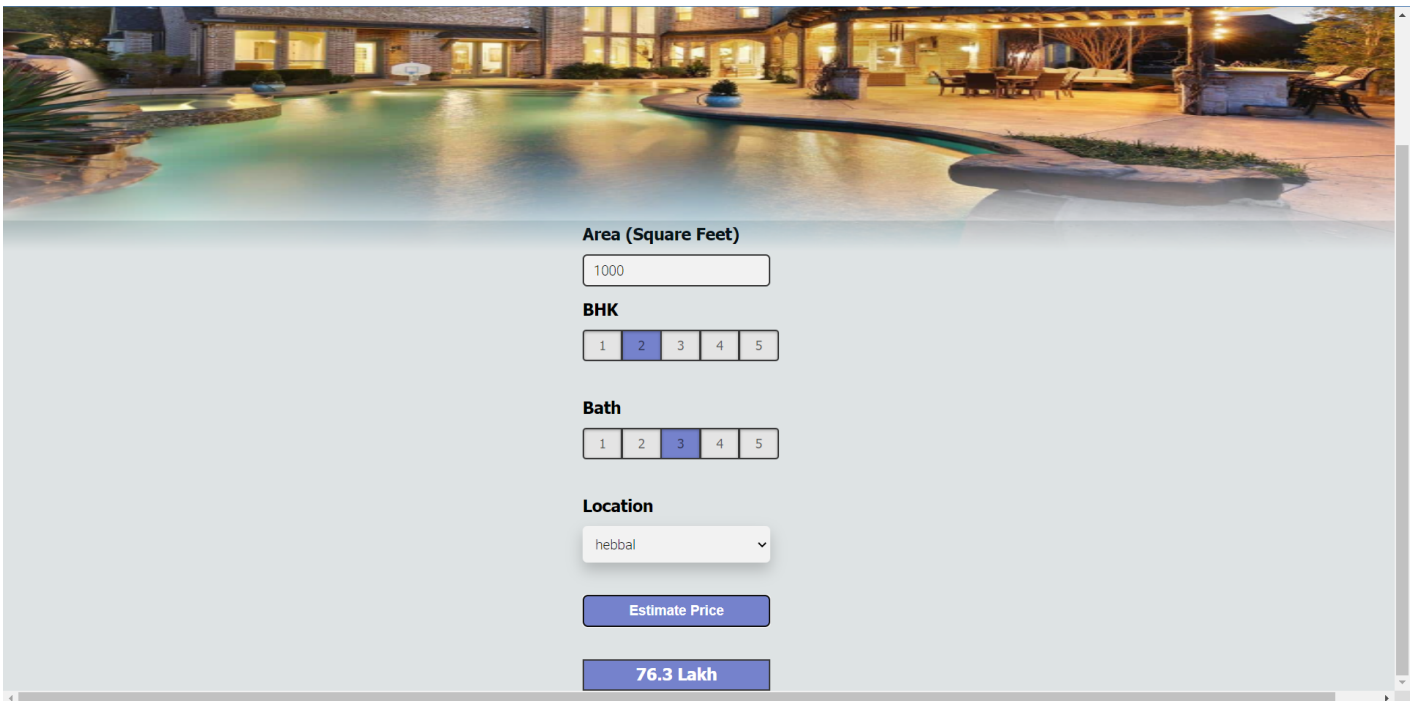

Fig. 11: Visual of the web page

The location will give a drop-down of all the locations in the city and once the 'Estimate Price' button is clicked, we shall obtain the price of the real estate of our specifications.

Chapter 4: Results

4.1 Discussion on the Results Achieved

The model has successfully been able to estimate the price of real estate. On cleaning the data, it becomes structured and hence it becomes easy to perform manipulations on the dataset. The efficiency of the linear regression model is much higher as compared to other models.

	model	best_score	best_params
0	linear_regression	0.818354	{'normalize': False}
1	lasso	0.687431	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.728048	{'criterion': 'mse', 'splitter': 'best'}

Fig. 15: Comparison of the efficiency scores of different regression models

The web page runs on the flask-generated server. The UI designed for the customer is feasible content to work on.

The price is estimated in a span of time and the interface offers the customer to make multiple selections and predict the price of the real estate.

4.2 Application of the Project

The project incurs to the present demand of the market. The real estate customers wish to know the price of the flat or the building while just exploring the different locations in a city. The model offers the customer to select the specifications of his desired flat and estimate the price accordingly. He could maintain the authenticity of the record shared by his data analyst and offer relevant amendments by observing the visual analysis.

4.3 Limitations of the Project

No model can be a perfect implementation of the idea in the first attempt, our model has certain limitations too.

i. The webpage runs on the server request generated by the flask for localhost and is not a domain purchased model.

ii. The price estimation of the real estate on the User Interface is not an immediate process but takes a span of 4-5 seconds.

4.4 Future Work

We have a list of certain improvements and future work to be performed on the project:
i. The website shall be made live and globally operative by purchasing a domain name.

ii. The website shall be made responsive.

Conclusion

The project that we have created shall find a suitable position in the cyber market world catering to the need for a platform to explore the desired property in a city and the owner or the dealer gets an appropriate platform to keep a check on his sales chronologically.

This opportunity provided to us has accomplished its motive of letting us explore our knowledge and implement it.

References

[1] https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

[2] https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data

[3] Rana, V., Mondal, J., Sharma, A., & Kashyap, I. (2020). *House Price Prediction Using Optimal Regression Techniques*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9362864&isnumber=9362727

[4] Fan, C., Cui, Z., & Zhong, X. (2018). *House Prices Prediction with Machine Learning Algorithms*. Association for Computing Machinery. https://doi.org/10.1145/3195106.3195133

[5] https://realpython.com/linear-regression-in-python/