A PROJECT REPORT

ON

BANGALORE HOUSE PRICE PREDICTION

Submitted in partial fulfilment for the requirement of the award of

TRAINING

IN

Data Analytics, Machine Learning and AI using Python

Submitted By

AMEY THAKUR

(Student at University of Mumbai affiliated Institute Terna Engineering College)

Under the guidance of

Mr Bandenawaz Bagwan

ACKNOWLEDGEMENT

My sincere gratitude and thanks towards my project paper guide Mr. Bandenawaz Bagwan. It was only with his backing and support that I could complete the report. He provided me with all sorts of help and corrected me if ever seemed to make mistakes. I have no such words to express my gratitude. Without his support, I couldn't even start the work. So I am grateful to him. And last but not the least, I acknowledge my dearest parents for being such a nice source of encouragement and moral support that helped me tremendously in this aspect. I also declare to the best of my knowledge and belief that the Project Work has not been submitted anywhere else.

ABSTRACT

I propose to implement a house price prediction model of Bangalore, India. It's a Machine Learning project which integrates Data Science and Web Development. I intend to deploy the app on the Heroku Cloud Application Platform. Housing prices fluctuate on a daily basis and are sometimes exaggerated rather than based on worth. The major focus of this project is on predicting home prices using genuine factors. Here, I intend to base an evaluation on every basic criterion that is taken into account when establishing the pricing. This project is a part of my training and internship in Data Analytics, Machine Learning and AI using Python at Diginique TechLabs.

TABLE OF CONTENTS

INTRODUCTION

1.1 Context

This project was created as part of my training and internship at Diginique Techlabs for Data Science Machine Learning and AI using Python. My supervisor was Mr. Bandenawaz Bagwan, and I had one month to finish the job. We had two weeks of training sessions where we studied everything from the ground up.

1.2 Motivation

I am highly interested in anything related to Machine Learning, the independent project provided me with the opportunity to study and reaffirm my passion for this subject. The capacity to generate guesses, forecasts, and offer machines the ability to learn on their own is both powerful and infinite in terms of application possibilities. Machine Learning may be applied in finance, medicine, and virtually any other field. That is why I opted to base my idea on Machine Learning.

1.3 Objective

As a first project, I intended to make it as instructional as possible by tackling each stage of the machine learning process and attempting to comprehend it well. I picked Bangalore Real Estate Prediction as a method, which is known as a "toy issue," identifying problems that are not of immediate scientific relevance but are helpful to demonstrate and practice. The objective was to forecast the price of a specific apartment based on market pricing while accounting for various "features" that would be established in the following sections.

1.4 Implementation

I genuinely believe that sharing resources and knowledge allows us to not only benefit others but also ourselves. The project's sources can be found at the following URL:

Web Application: https://bangalorehousepriceprediction.herokuapp.com

Repository: https://github.com/Amey-Thakur/BANGALORE-HOUSE-PRICE-PREDICTION

Model: https://www.kaggle.com/ameythakur20/bangalore-house-price-prediction-model

LITERATURE SURVEY

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. As a result, anticipating the important estate price is an essential economic indicator. According to the 2011 census, the Asian country ranks second in the world in terms of the number of households, with a total of 24.67 crores. However, previous recessions have demonstrated that real estate costs cannot see the expenses of significant estate property are linked to the state's economic situation. Regardless, we don't have accurate standardized approaches to live the significant estate property values.

First, I looked at different articles and discussions about machine learning for housing price prediction. The title of the article is house price prediction, and it is based on machine learning and neural networks. The publication's description is minimal error and the highest accuracy. The aforementioned title of the paper is Hedonic models based on price data from Belfast infer that submarkets and residential valuation this model is used to identify over a larger spatial scale and implications for the evaluation process related to the selection of comparable evidence and the quality of variables that the values may require. Understanding current developments in house prices and homeownership are the subject of the study. In this article, they utilized a feedback mechanism or social pandemic that fosters a perception of property as an essential market investment.

METHODOLOGY

3.1 Data Collection

The statistics were gathered from Bangalore home prices. The information includes many variables such as area type, availability, location, BHK, society, total square feet, bathrooms, and balconies.

3.2 Linear Regression

Linear regression is a supervised learning technique. It is responsible for predicting the value of a dependent variable (Y) based on a given independent variable (X). It is the connection between the input (X) and the output (Y). It is one of the most well-known and well-understood machine learning algorithms. Simple linear regression, ordinary least squares, Gradient Descent, and Regularization are the linear regression models.

3.3 Decision Tree Regression

It is an object that trains a tree-structured model to predict data in the future in order to provide meaningful continuous output. The core principles of decision trees, Maximizing Information Gain, Classification trees, and Regression trees are the processes involved in decision tree regression. The essential notion of decision trees is that they are built via recursive partitioning. Each node can be divided into child nodes, beginning with the root node, which is known as the parent node. These nodes have the potential to become the parent nodes of their resulting offspring nodes. The nodes at the informative features are specified as the maximizing information gain, to establish an objective function that is to optimize the tree learning method.

3.4 Classification Trees

Classification trees are used to forecast the object into classes of a categorical dependent variable based on one or more predictor variables.

3.5 Regression Trees

It supports both continuous and categorical input variables. Regression trees are regarded as research with various machine algorithms for the regression issue, with the Decision Tree approach providing the lowest loss. The R-Squared value for the Decision Tree is 0.998, indicating that it is an excellent model. The Decision Tree was used to complete the web development.

3.6 Support Vector Regression

Supervised learning is linked with learning algorithms that examine data for classification and regression analysis.

3.7 Random Forest Regression

It is an essential learning approach for classification and regression to create a large number of decision trees. Preliminaries of decision trees are common approaches for a variety of machine learning problems. Tree learning is required for serving n off the self-produce for data mining since it is invariant despite scaling and several other changes. The trees are grown very deep in order to learn a high regular pattern. Random forest is a method of averaging several deep decision trees trained on various portions of the same training set. This comes at the price of a slight increase in bias and some interoperability.

PROJECT

4.1 Problem Statement

Create a model to estimate the price of houses in Bengaluru and host it on Heroku.

4.2 Data

The data is the most important aspect of a machine learning assignment, to which special attention should be paid. Indeed, the data will heavily affect the findings depending on where we found them, how they are presented, if they are consistent, if there is an outlier, and so on. Many questions must be addressed at this stage to ensure that the learning algorithm is efficient and correct.

To obtain, clean, and convert the data, many sub steps are required. I'll go through each one to illustrate how they've been used in my project and why they're helpful for the machine learning section.

4.3 Dataset

Dataset: https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data

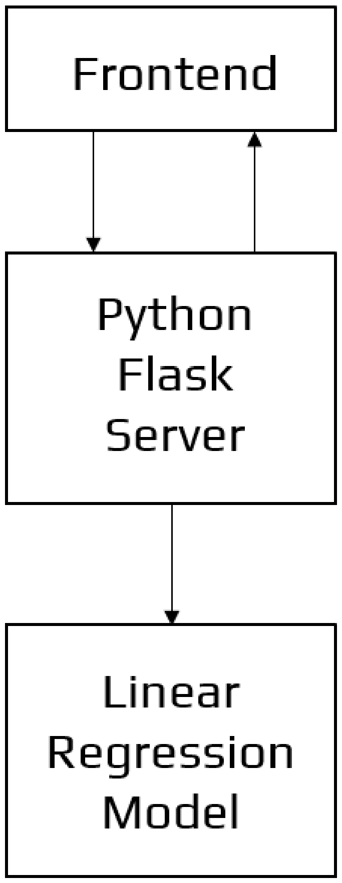
4.4 Model

Model: https://www.kaggle.com/ameythakur20/bangalore-house-price-prediction-model

4.5 Web App

Web Application: https://bangalorehousepriceprediction.herokuapp.com

4.6 Project Architecture



Architecture of the Application

4.7 Data Science

The first stage is standard data science work, in which we take a data set named ‘Bengaluru House pricing data' from Kaggle. We will do significant data cleaning on it to guarantee that it provides reliable predictions throughout prediction.

This jupyter notebook, ‘Bangalore-House-Price-Prediction-Model.ipynb,' is where we do all of our data science work. Because the jupyter notebook is self-explanatory, I'll only touch on the principles that I've implemented briefly. In terms of data cleansing, our dataset needs a significant amount of effort. In fact, 70% of the notebook is dedicated to data cleaning, in which we eliminate empty rows and remove superfluous columns that will not aid in prediction.

The process of obtaining valuable and significant information from a dataset that will contribute the most to a successful prediction is the next stage.

The final stage is to deal with outliers. Outliers are abnormalities that do massive damage to data and prediction. There is a lot to comprehend conceptually from the dataset in order to discover and eliminate these outliers.

Finally, the original dataset of over 13000 rows and 9 columns is reduced to about 7000 rows and 5 columns.

4.8 Machine Learning

The resulting data is fed into a machine learning model. To find the optimal procedure and parameters for the model, we will mostly employ K-fold Cross-Validation and the GridSearchCV approach.

It turns out that the linear regression model produces the best results for our data, with a score of more than 80%, which is not terrible.

Now, we need to export our model as a pickle file (Bengaluru House Data.pickle), which transforms Python objects into a character stream. Also, in order to interact with the locations(columns) from the frontend, we must export them into a JSON (columns.json) file.

4.9 Frontend

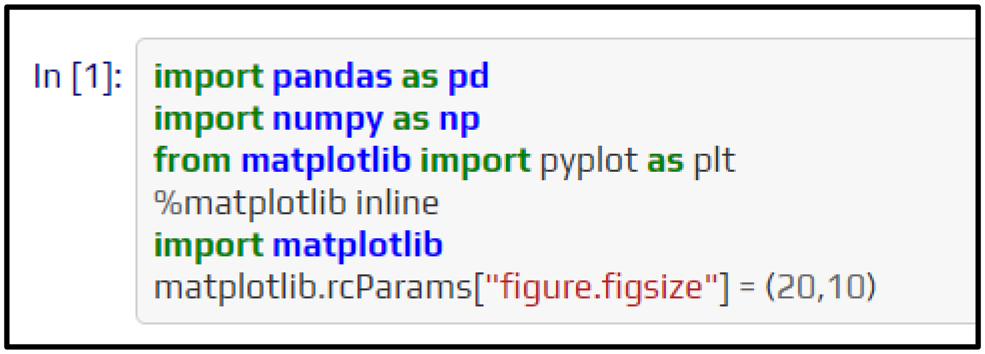
The front end is built up of straightforward HTML. To receive an estimated pricing, the user may fill-up the form with the number of square feet, BHK, bathrooms, and location and click the ‘ESTIMATE PRICE' button. The JavaScript file is in charge of connecting with the backend flask server routes as well as the frontend HTML. It takes the form data entered by the user and executes the function, which employs the prediction model to calculate the projected price in lakhs of rupees (1 lakh = 100000).

MODEL

5.1 Steps to create the model

1. Import Libraries
2. Load Dataset
3. Exploratory Data Analysis
4. Data Cleaning
5. Feature Engineering
6. Dimensionality Reductions
7. Outlier Removal using Business Logic
8. Outlier Removal using Standard Deviation & Mean
9. Data Visualization
10. Building a Model
11. Test the Model for few properties
12. Export the tested model to a pickle file

5.1.1 Import Libraries



```
In [1]: import pandas as pd
        import numpy as np
        from matplotlib import pyplot as plt
        %matplotlib inline
        import matplotlib
        matplotlib.rcParams["figure.figsize"] = (20,10)
```

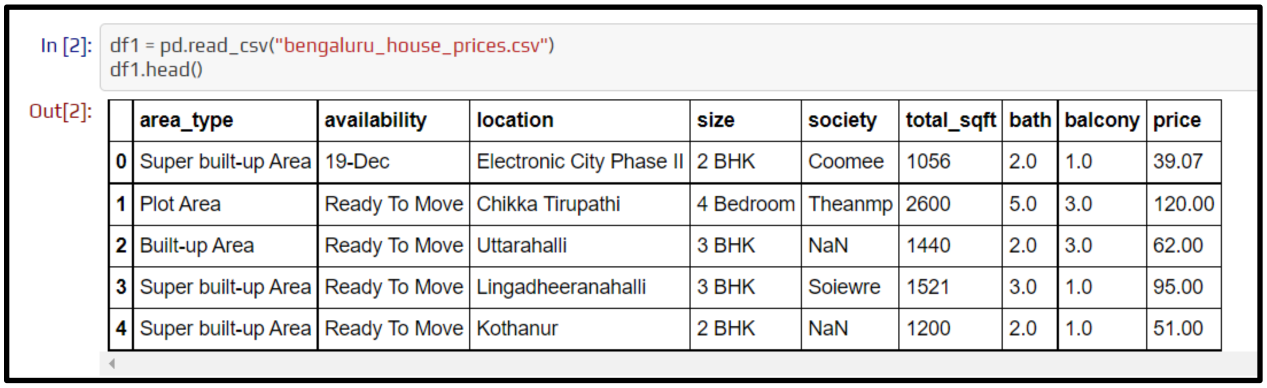
Pandas - Pandas is a data manipulation and analysis software package built-in Python. It specifically provides data structures and functions for manipulating numerical tables and time series.

NumPy - NumPy is an abbreviation for Numerical Python. NumPy is a Python package that is used to work with arrays. It also includes functions for working with linear algebra, Fourier transforms, and matrices.

Matplotlib - Matplotlib is a charting toolkit for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding charts into applications utilizing general-purpose GUI toolkits. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

5.1.2 Load Dataset

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



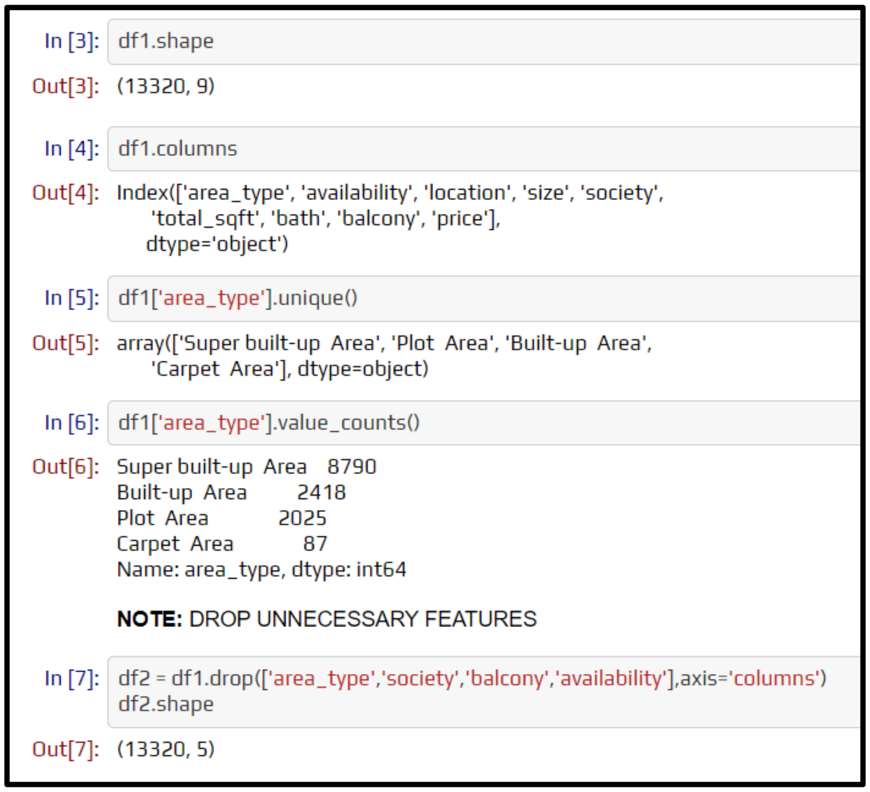
```
In [2]: df1 = pd.read_csv("bengaluru_house_prices.csv")
        df1.head()
Out[2]:
```

	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

read_csv - read_csv is an important pandas function to read CSV files and do operations on them.

head() - To obtain the first n rows, use the head() function. This function retrieves the object's first n rows based on location. It is handy for rapidly determining whether your object contains the correct type of data.

5.1.3 Exploratory Data Analysis



```
In [3]: df1.shape

Out[3]: (13320, 9)

In [4]: df1.columns

Out[4]: Index(['area_type', 'availability', 'location', 'size', 'society',
               'total_sqft', 'bath', 'balcony', 'price'],
              dtype='object')

In [5]: df1['area_type'].unique()

Out[5]: array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
               'Carpet  Area'], dtype=object)

In [6]: df1['area_type'].value_counts()

Out[6]: Super built-up  Area    8790
        Built-up  Area          2418
        Plot  Area              2025
        Carpet  Area              87
        Name: area_type, dtype: int64
```

NOTE: DROP UNNECESSARY FEATURES

```
In [7]: df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
        df2.shape

Out[7]: (13320, 5)
```

shape - The function "shape" returns the shape of an array. The shape is a tuple of numbers. These values represent the lengths of the appropriate array dimension. In other words, the "shape" of an array is a tuple with the number of items per axis (dimension).

columns - columns return the column labels of the provided Dataframe.

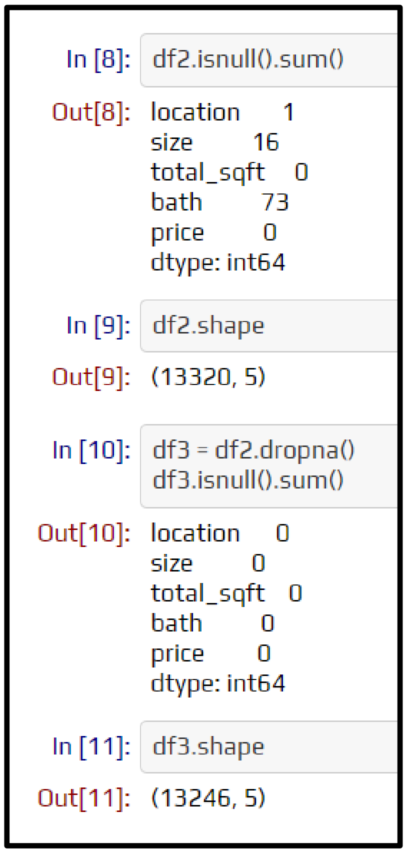
unique() - The unique() function is used to locate the array's unique items. Returns the array's sorted unique items.

NOTE: Drop unnecessary features.

counts() - The value counts() function returns an object with counts of unique values. The resultant object will be arranged in decreasing order, with the first element being the most often occurring. By default, NA values are excluded.

drop() - The drop() function is used to remove a set of labels from a row or column. By supplying label names and associated axes, or by explicitly specifying index or column names, you can remove rows or columns.

5.1.4 Data Cleaning



```
In [8]:    df2.isnull().sum()

Out[8]:    location      1
           size         16
           total_sqft    0
           bath         73
           price         0
           dtype: int64

In [9]:    df2.shape

Out[9]:    (13320, 5)

In [10]:   df3 = df2.dropna()
           df3.isnull().sum()

Out[10]:   location     0
           size         0
           total_sqft   0
           bath         0
           price        0
           dtype: int64

In [11]:   df3.shape

Out[11]:   (13246, 5)
```

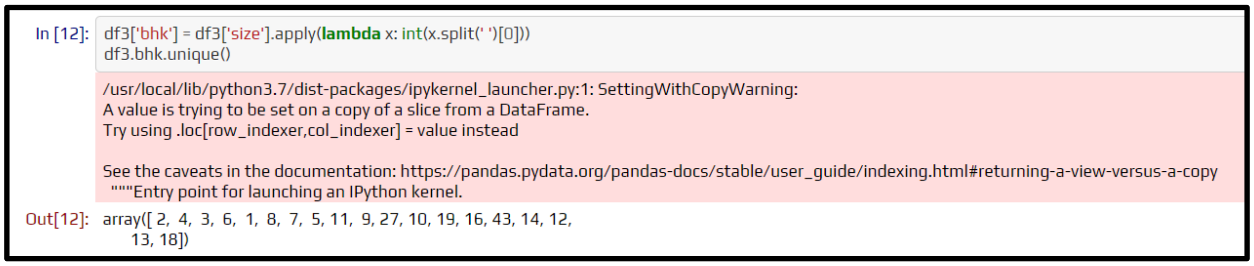
isnull() - The isnull() function finds missing values in a series object. It returns a boolean same-sized object that indicates if the values are NA or not. Missing values are translated to True, whereas non-missing values are assigned to False.

sum() - The sum() function in Python computes the sum of all numerical values in an iterable. sum() may be used with both integers and floating-point values. The sum() function in Python may be used to compute the sum of all values in an inerrable object.

dropna() - dropna() returns Index with no NA/NaN values. All missing values are deleted, and a new object is returned that does not include any NaN values.

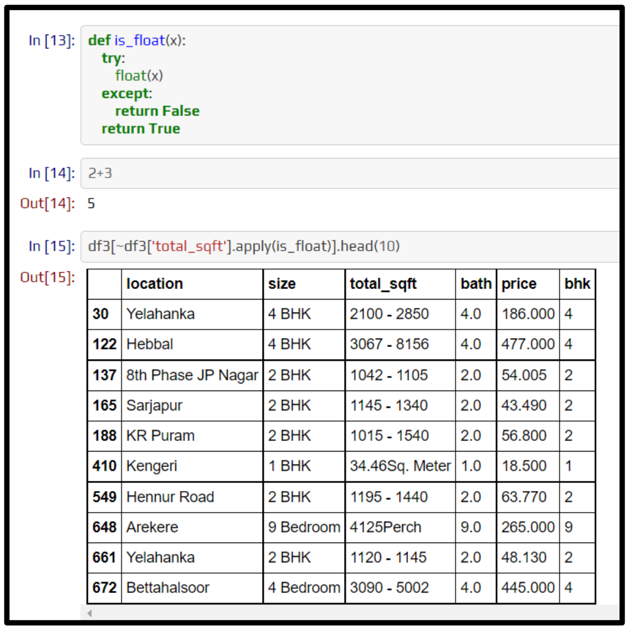
5.1.5 Feature Engineering

Feature engineering is the process of creating new features from existing data, which is frequently dispersed over many linked tables. Feature engineering is collecting important information from data and consolidating it into a single table that can subsequently be utilized to train a machine learning model.



```
In [12]: df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
         df3.bhk.unique()

         /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
         A value is trying to be set on a copy of a slice from a DataFrame.
         Try using .loc[row_indexer,col_indexer] = value instead

         See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
           """Entry point for launching an IPython kernel.

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

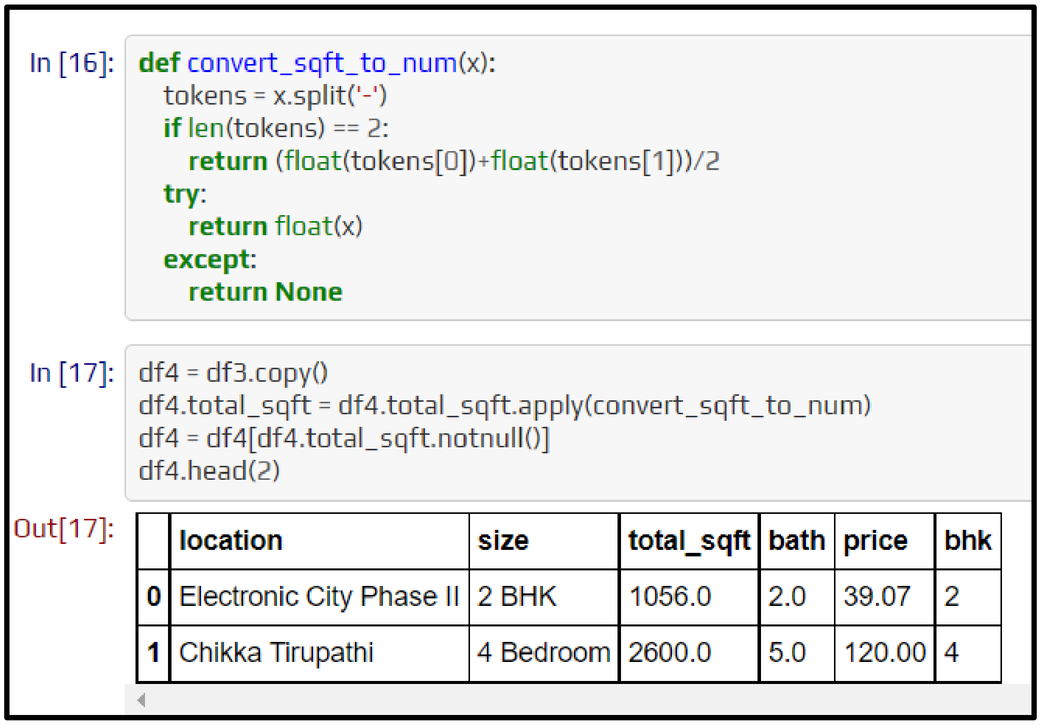
Explore total sqft features.



```
In [13]: def is_float(x):
             try:
                 float(x)
             except:
                 return False
             return True

In [14]: 2+3

Out[14]: 5

In [15]: df3[~df3['total_sqft'].apply(is_float)].head(10)

Out[15]:
```

	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
410	Kengeri	1 BHK	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	2 BHK	1195 - 1440	2.0	63.770	2
648	Arekere	9 Bedroom	4125Perch	9.0	265.000	9
661	Yelahanka	2 BHK	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	4 Bedroom	3090 - 5002	4.0	445.000	4

The above shows that total_sqft can be a range (e.g. 2100-2850). For such a case, we can just take an average of min and max value in the range. There are other cases such as 34.46Sq. The meter which one can convert to square ft using unit conversion. I am going to just drop such corner cases to keep things simple.



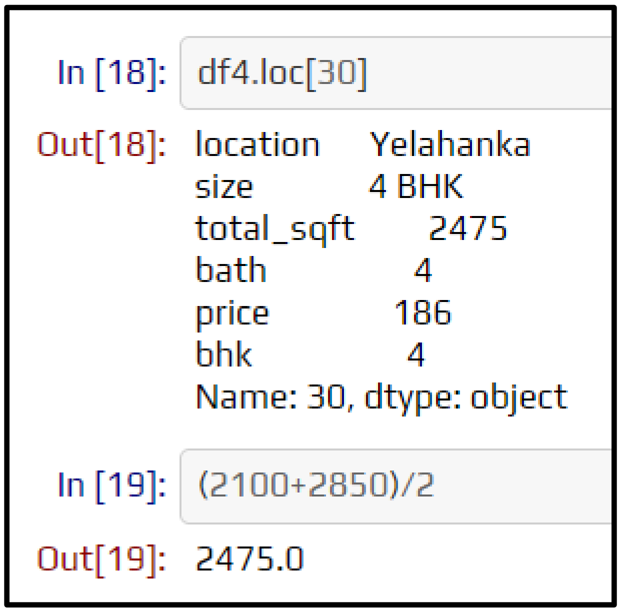
```
In [16]: def convert_sqft_to_num(x):
           tokens = x.split('-')
           if len(tokens) == 2:
             return (float(tokens[0])+float(tokens[1]))/2
           try:
             return float(x)
           except:
             return None
```

```
In [17]: df4 = df3.copy()
         df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
         df4 = df4[df4.total_sqft.notnull()]
         df4.head(2)
```

Out[17]:

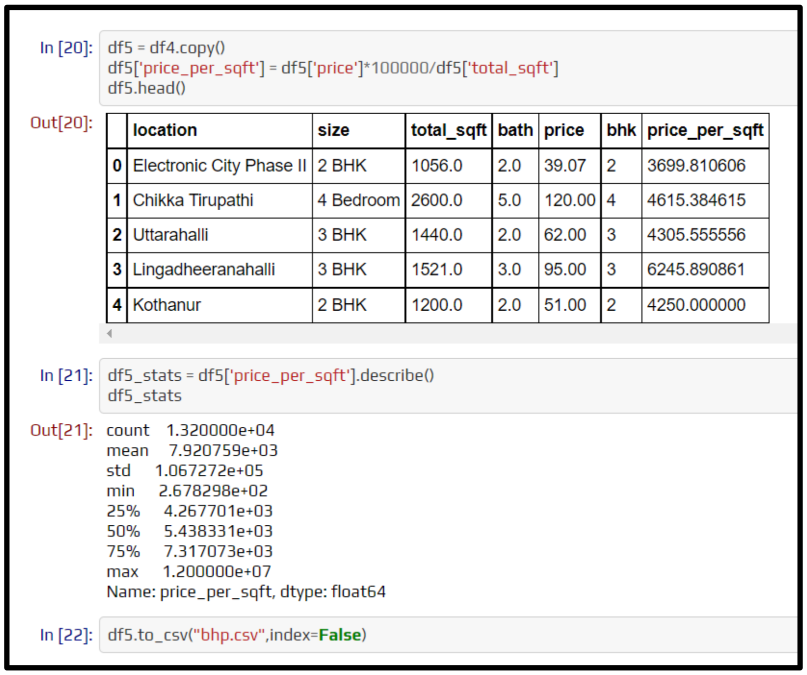
	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4

For the below row, it shows total_sqft as 2475 which is an average of the range 2100-2850.



```
In [18]: df4.loc[30]

Out[18]: location      Yelahanka
         size          4 BHK
         total_sqft       2475
         bath             4
         price           186
         bhk              4
         Name: 30, dtype: object

In [19]: (2100+2850)/2

Out[19]: 2475.0
```

Add a new feature called price per square feet.



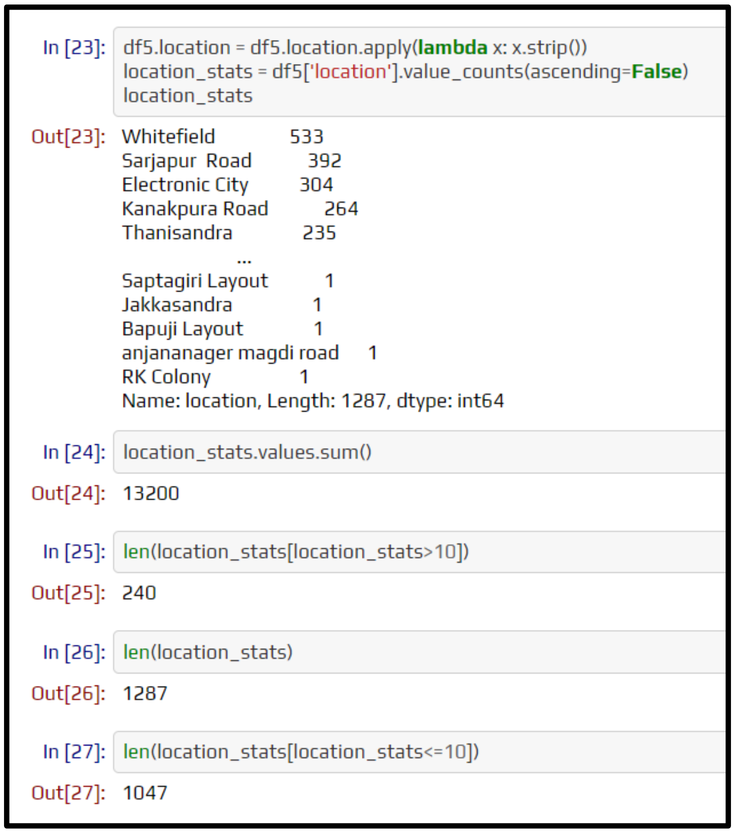
```
In [20]: df5 = df4.copy()
         df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
         df5.head()
```

Out[20]:

	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

```
In [21]: df5_stats = df5['price_per_sqft'].describe()
         df5_stats

Out[21]: count    1.320000e+04
         mean     7.920759e+03
         std      1.067272e+05
         min      2.678298e+02
         25%      4.267701e+03
         50%      5.438331e+03
         75%      7.317073e+03
         max      1.200000e+07
         Name: price_per_sqft, dtype: float64

In [22]: df5.to_csv("bhp.csv",index=False)
```

Examine locations that are categorical variables. We need to apply the dimensionality reduction technique here to reduce the number of locations.

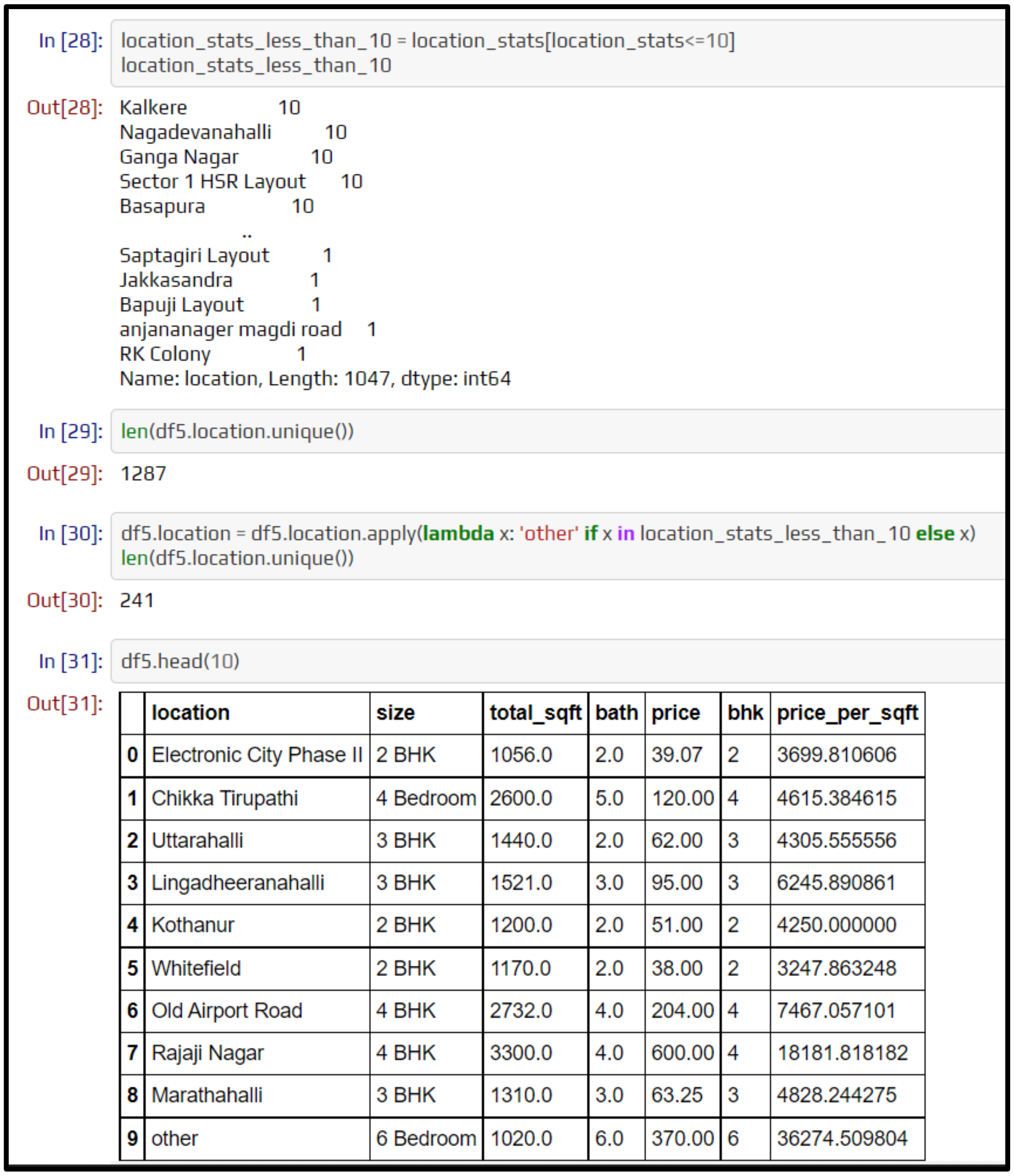


```
In [23]: df5.location = df5.location.apply(lambda x: x.strip())
         location_stats = df5['location'].value_counts(ascending=False)
         location_stats

Out[23]: Whitefield             533
         Sarjapur  Road         392
         Electronic City        304
         Kanakpura Road         264
         Thanisandra            235
                               ...
         Saptagiri Layout         1
         Jakkasandra              1
         Bapuji Layout            1
         anjananager magdi road   1
         RK Colony                1
         Name: location, Length: 1287, dtype: int64

In [24]: location_stats.values.sum()

Out[24]: 13200

In [25]: len(location_stats[location_stats>10])

Out[25]: 240

In [26]: len(location_stats)

Out[26]: 1287

In [27]: len(location_stats[location_stats<=10])

Out[27]: 1047
```

5.1.6 Dimensionality Reductions

Dimensionality reduction methods are used to reduce the number of input variables in training data. When working with high-dimensional data, it is frequently beneficial to reduce the dimensionality by projecting the data to a lower-dimensional subspace that captures the data's "essence."

Any location that has less than 10 data points should be tagged as another location. This way the number of categories can be reduced by a huge amount. Later on, when we do one-hot encoding, it will help us with having fewer dummy columns.

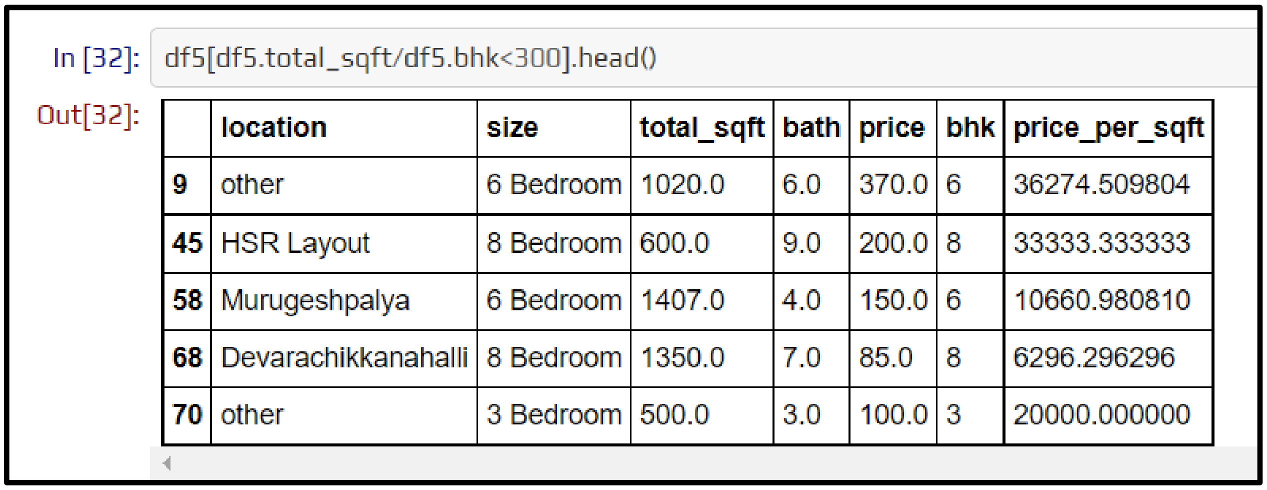


```
In [28]: location_stats_less_than_10 = location_stats[location_stats<=10]
         location_stats_less_than_10

Out[28]: Kalkere                  10
         Nagadevanahalli          10
         Ganga Nagar              10
         Sector 1 HSR Layout      10
         Basapura                 10
                                  ..
         Saptagiri Layout          1
         Jakkasandra               1
         Bapuji Layout             1
         anjananager magdi road    1
         RK Colony                 1
         Name: location, Length: 1047, dtype: int64

In [29]: len(df5.location.unique())

Out[29]: 1287

In [30]: df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
         len(df5.location.unique())

Out[30]: 241

In [31]: df5.head(10)

Out[31]:
```

	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

5.1.7 Outlier Removal using Business Logic

As a data scientist when you have a conversation with your business manager (who has expertise in real estate), he will tell you that normally square ft per bedroom is 300 (i.e. 2 bhk apartment is minimum 600 sqft. If you have for example 400 sqft apartment with 2 bhk then that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping our minimum threshold per bhk to be 300 sqft.

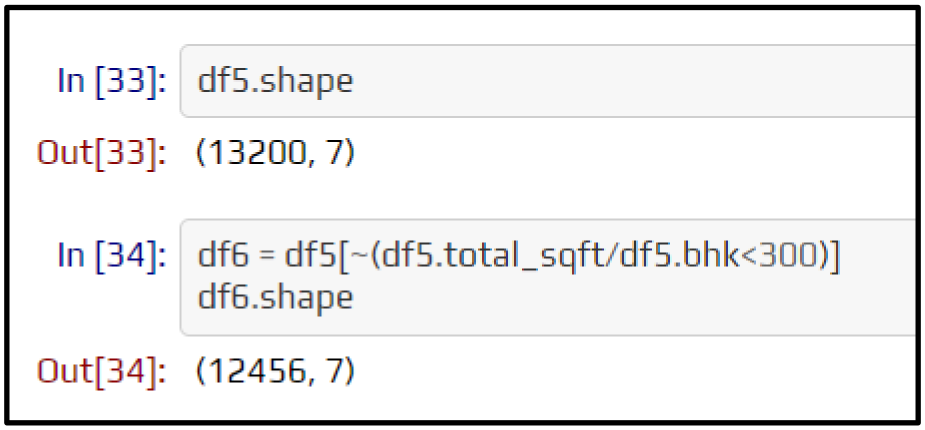


```
In [32]: df5[df5.total_sqft/df5.bhk<300].head()
```

Out[32]:

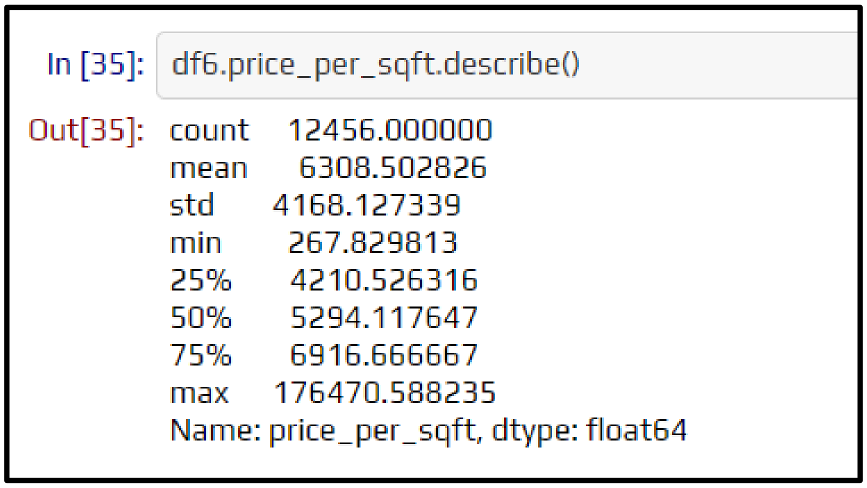
	location	size	total_sqft	bath	price	bhk	price_per_sqft
9	other	6 Bedroom	1020.0	6.0	370.0	6	36274.509804
45	HSR Layout	8 Bedroom	600.0	9.0	200.0	8	33333.333333
58	Murugeshpalya	6 Bedroom	1407.0	4.0	150.0	6	10660.980810
68	Devarachikkanahalli	8 Bedroom	1350.0	7.0	85.0	8	6296.296296
70	other	3 Bedroom	500.0	3.0	100.0	3	20000.000000

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



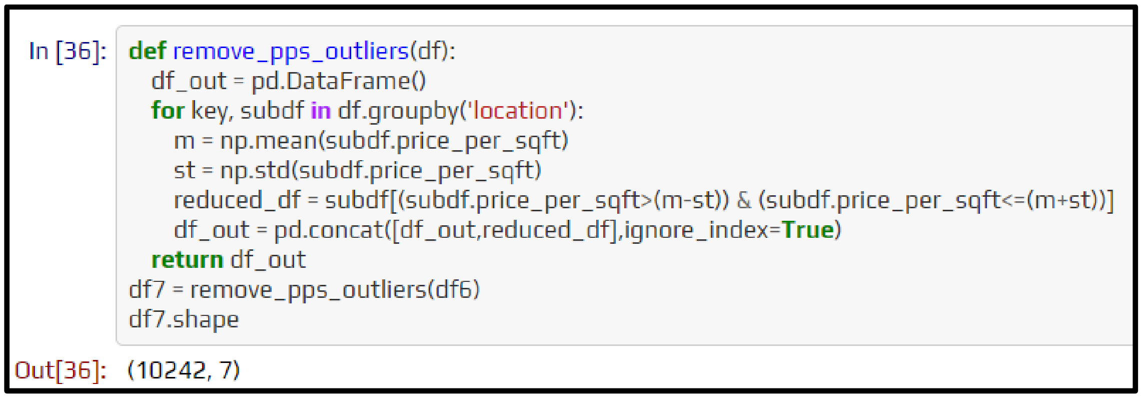
```
In [33]: df5.shape

Out[33]: (13200, 7)

In [34]: df6 = df5[~(df5.total_sqft/df5.bhk<300)]
         df6.shape

Out[34]: (12456, 7)
```

5.1.8 Outlier Removal using Standard Deviation & Mean



```
In [35]: df6.price_per_sqft.describe()

Out[35]: 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
```

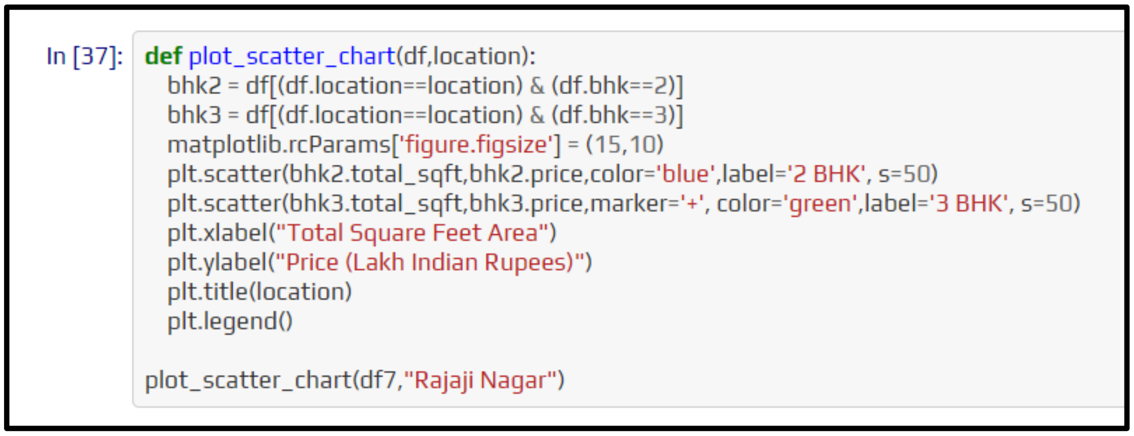
Here we find that the min price per sqft is 267 rs/sqft whereas the max is 12000000, which shows a wide variation in property prices. We should remove outliers per location using mean and one standard deviation.



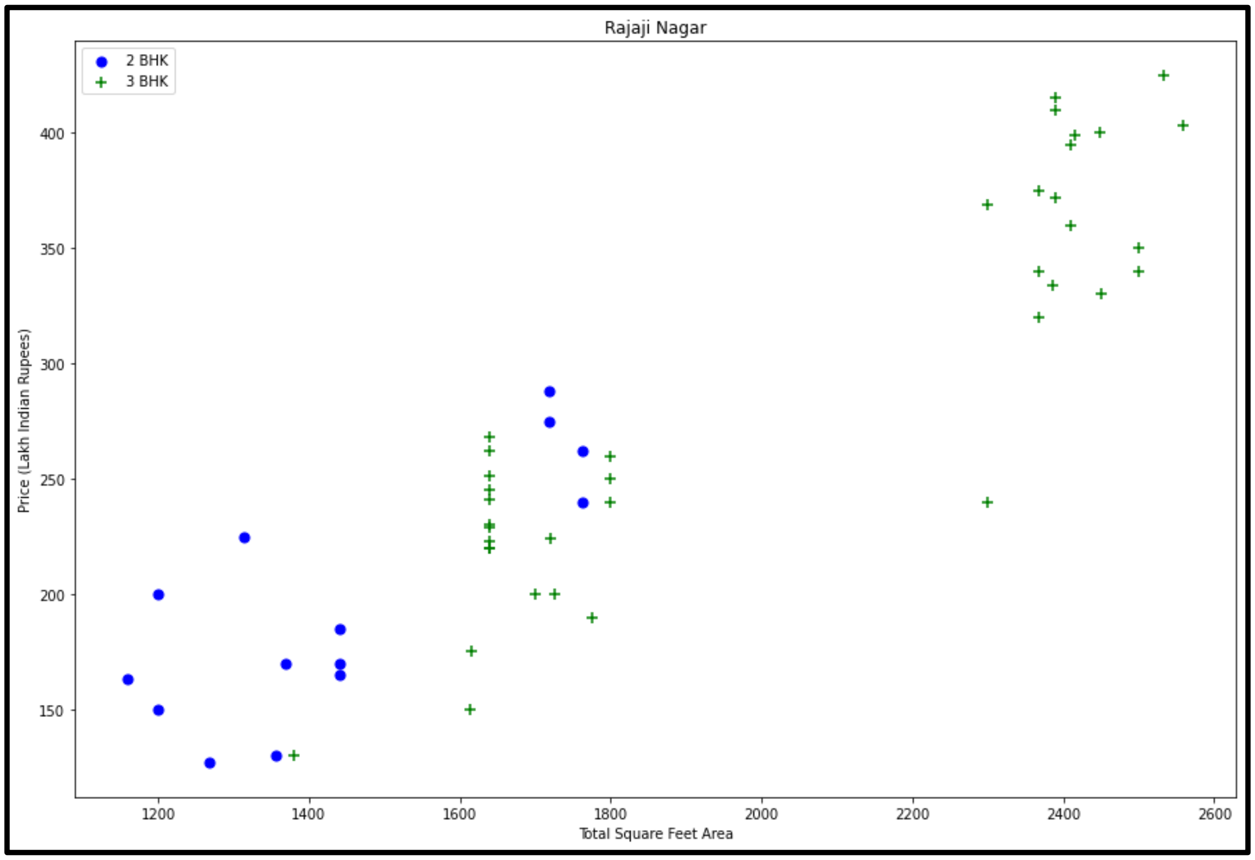
```
In [36]: def remove_pps_outliers(df):
           df_out = pd.DataFrame()
           for key, subdf in df.groupby('location'):
             m = np.mean(subdf.price_per_sqft)
             st = np.std(subdf.price_per_sqft)
             reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st))]
             df_out = pd.concat([df_out,reduced_df],ignore_index=True)
           return df_out
         df7 = remove_pps_outliers(df6)
         df7.shape

Out[36]: (10242, 7)
```

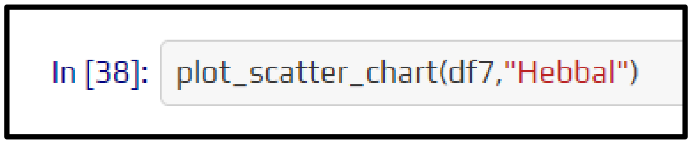
5.1.9 Data Visualization

Let's check for a given location how the 2 BHK and 3 BHK property prices look like.

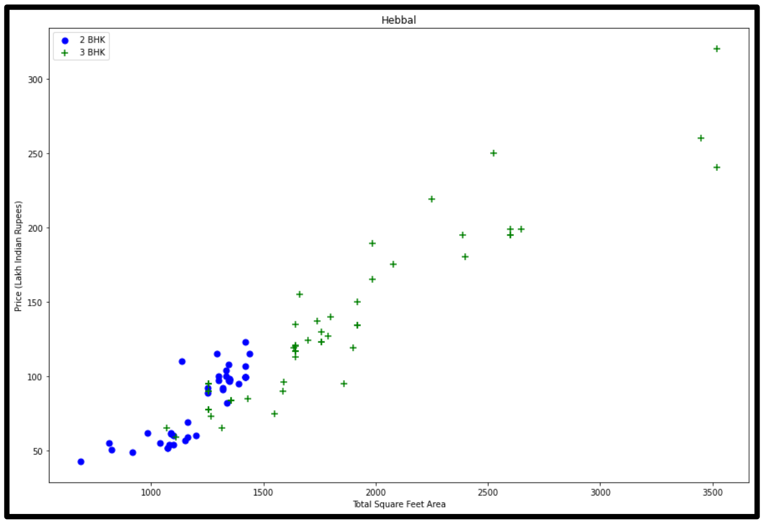


```
In [37]: def plot_scatter_chart(df,location):
             bhk2 = df[(df.location==location) & (df.bhk==2)]
             bhk3 = df[(df.location==location) & (df.bhk==3)]
             matplotlib.rcParams['figure.figsize'] = (15,10)
             plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 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 Indian Rupees)")
             plt.title(location)
             plt.legend()

         plot_scatter_chart(df7,"Rajaji Nagar")
```





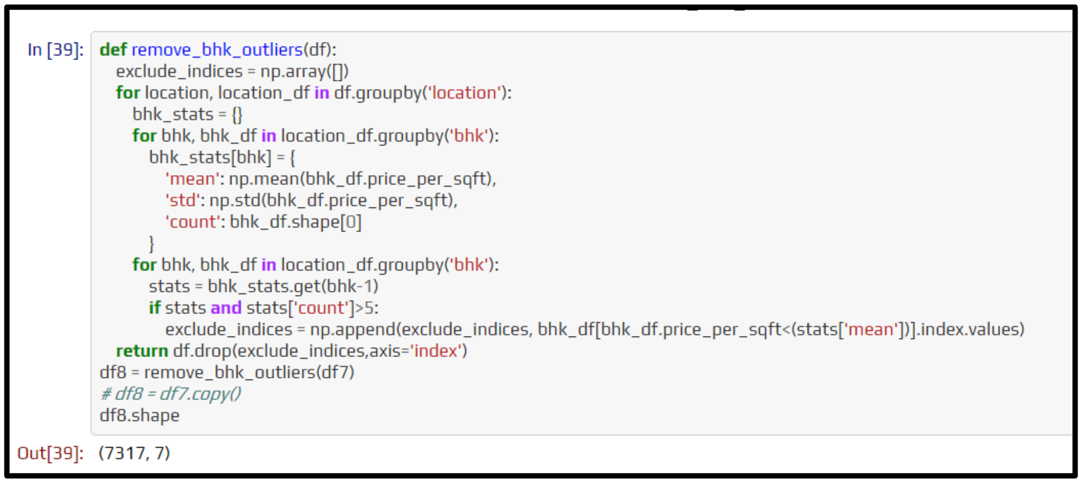
```
In [38]: plot_scatter_chart(df7,"Hebbal")
```



We should also remove properties where for the same location, the price of (for example) a 3 bedroom apartment is less than a 2 bedroom apartment (with the same square ft area). What we will do is for a given location, we will build a dictionary of stats per bhk, i.e.

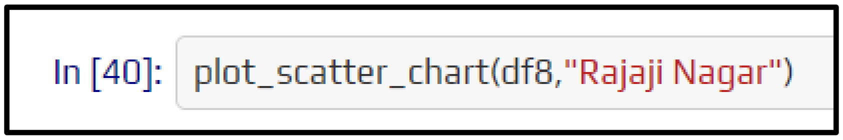
```
{
  '1' : {
    'mean': 4000,
    'std: 2000,
    'count': 34
  },
  '2' : {
    'mean': 4300,
    'std: 2300,
    'count': 22
  },
}
```

Now we can remove those 2 BHK apartments whose price_per_sqft is less than the mean price_per_sqft of 1 BHK apartment.

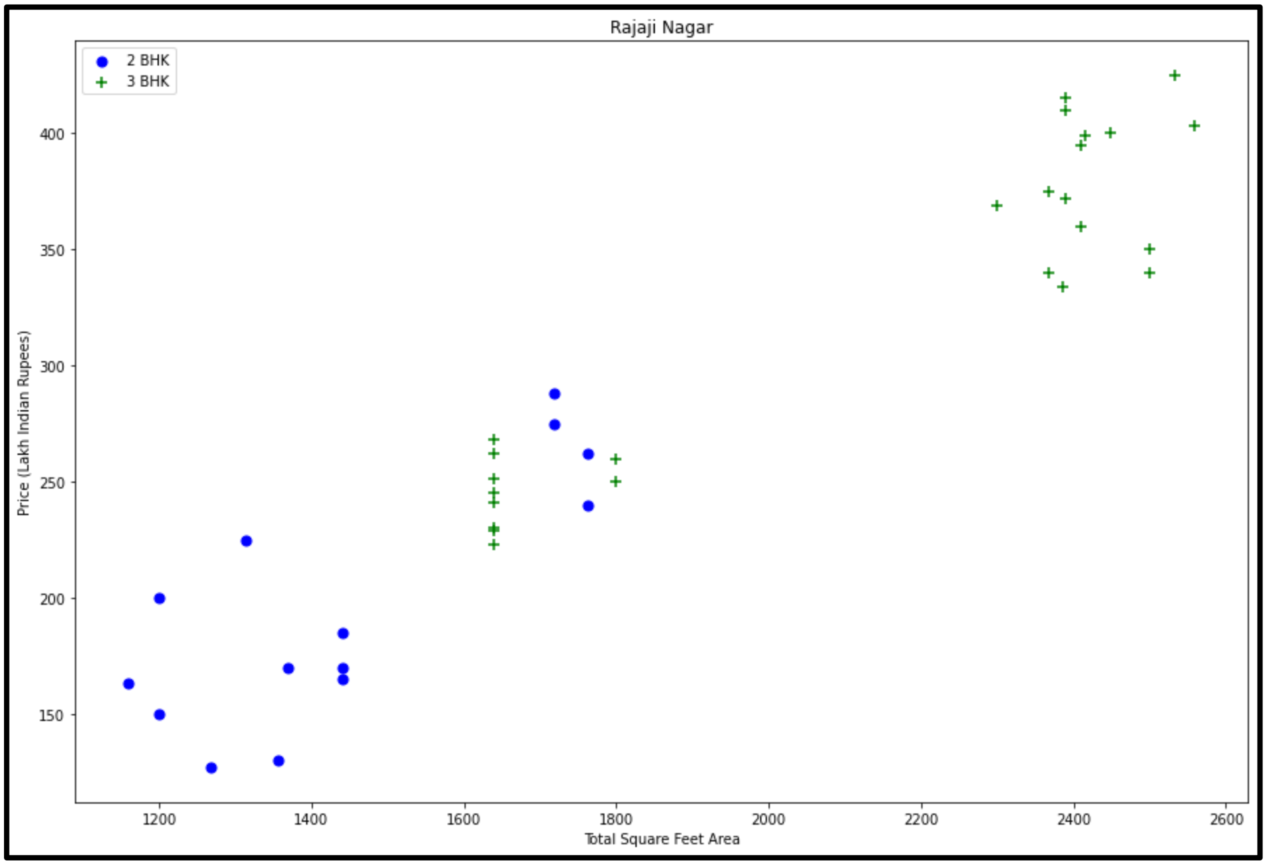


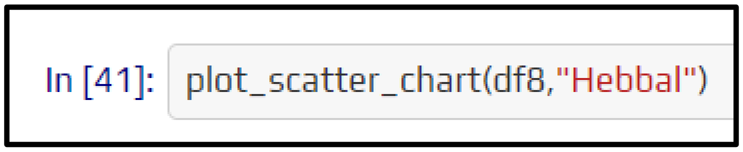
```
In [39]: def remove_bhk_outliers(df):
             exclude_indices = np.array([])
             for location, location_df in df.groupby('location'):
                 bhk_stats = {}
                 for bhk, bhk_df in location_df.groupby('bhk'):
                     bhk_stats[bhk] = {
                         '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

Out[39]: (7317, 7)
```

Plot the same scatter chart again to visualize price_per_sqft for 2 BHK and 3 BHK properties.

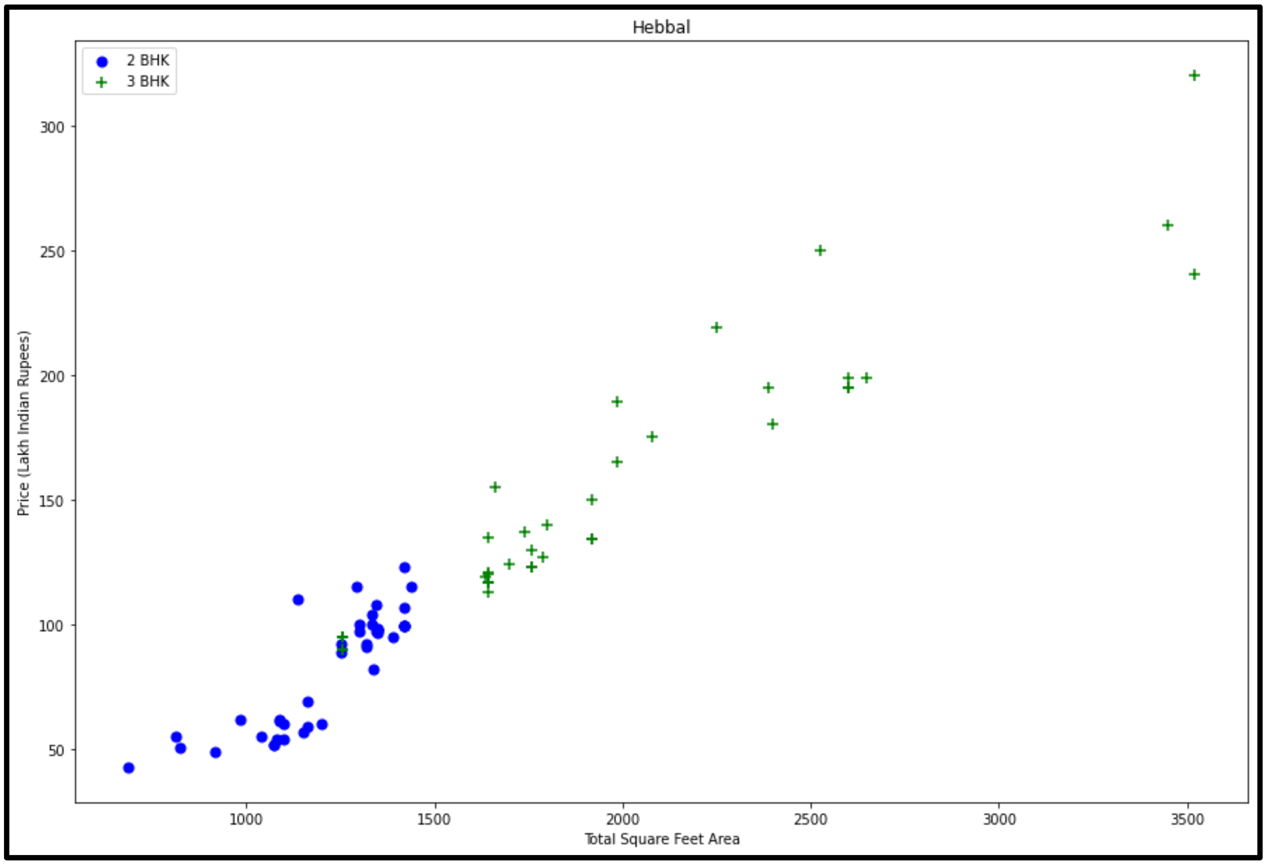


```
In [40]: plot_scatter_chart(df8,"Rajaji Nagar")
```



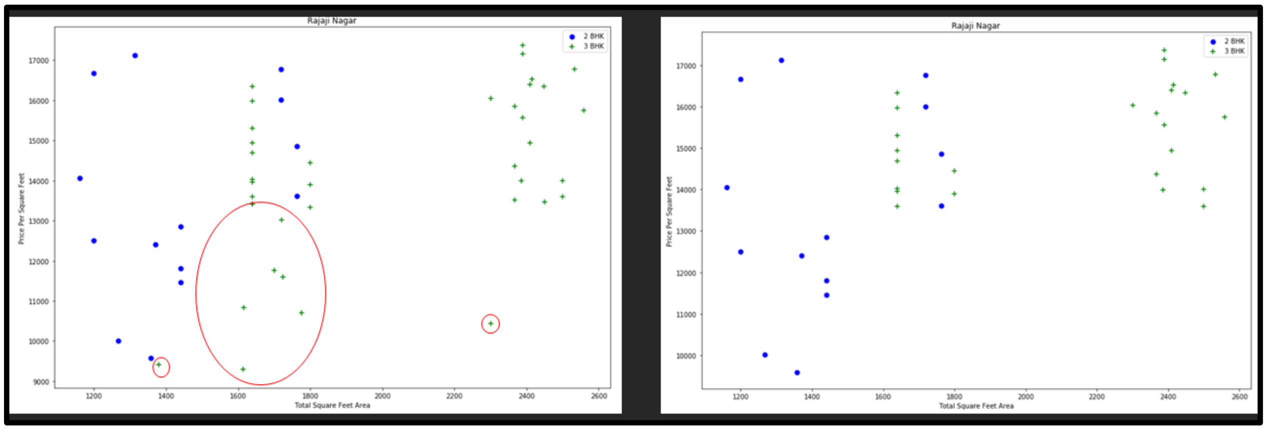


```
In [41]: plot_scatter_chart(df8,"Hebbal")
```

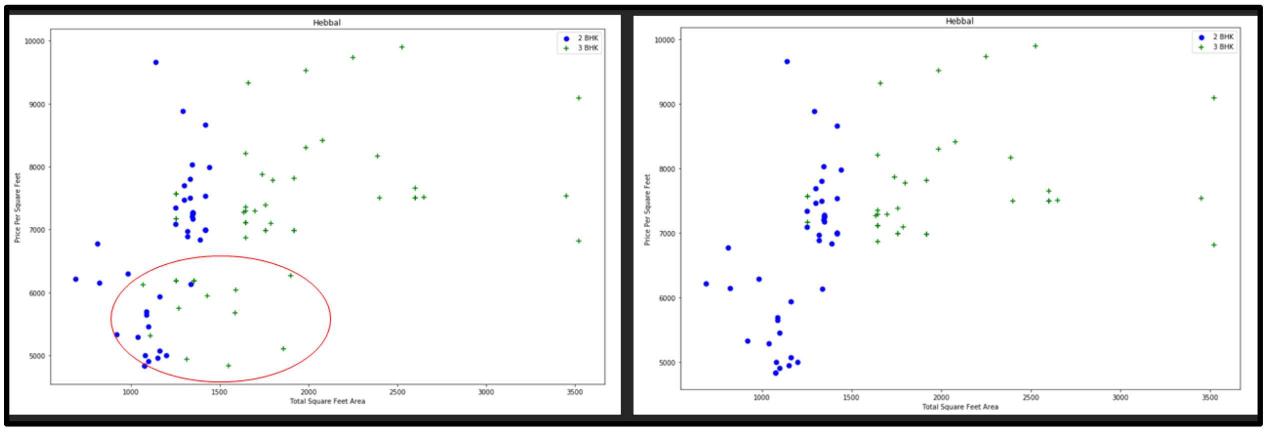


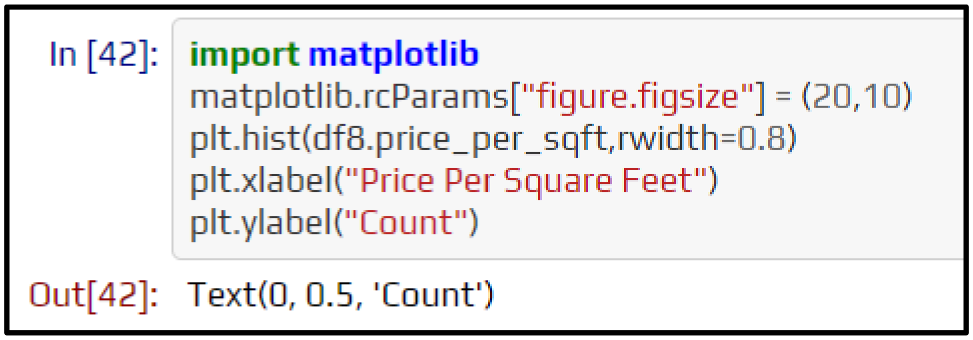
Based on the above charts we can see that data points highlighted in red below are outliers and they are being removed due to the remove_bhk_outliers function.

Before and after outlier removal: Rajaji Nagar

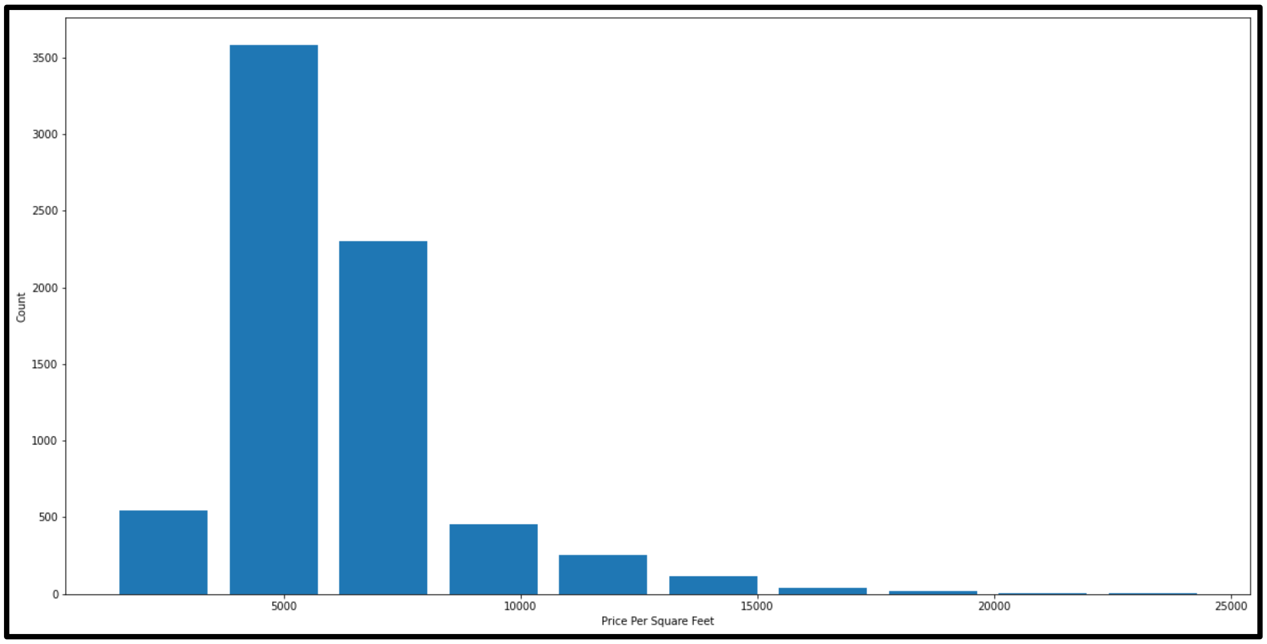


Before and after outlier removal: Hebbal

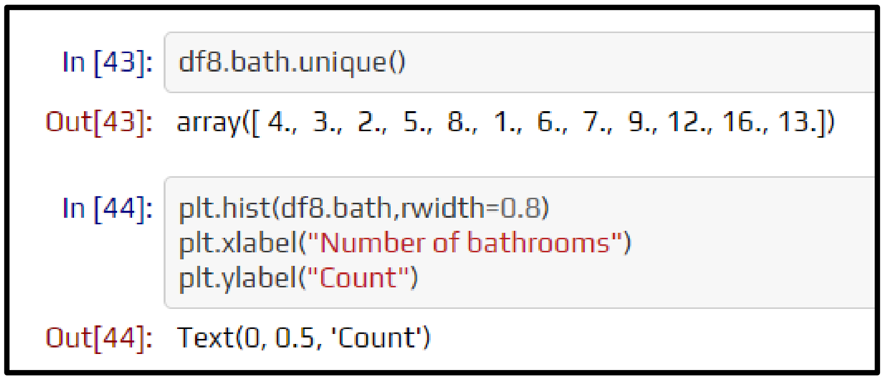




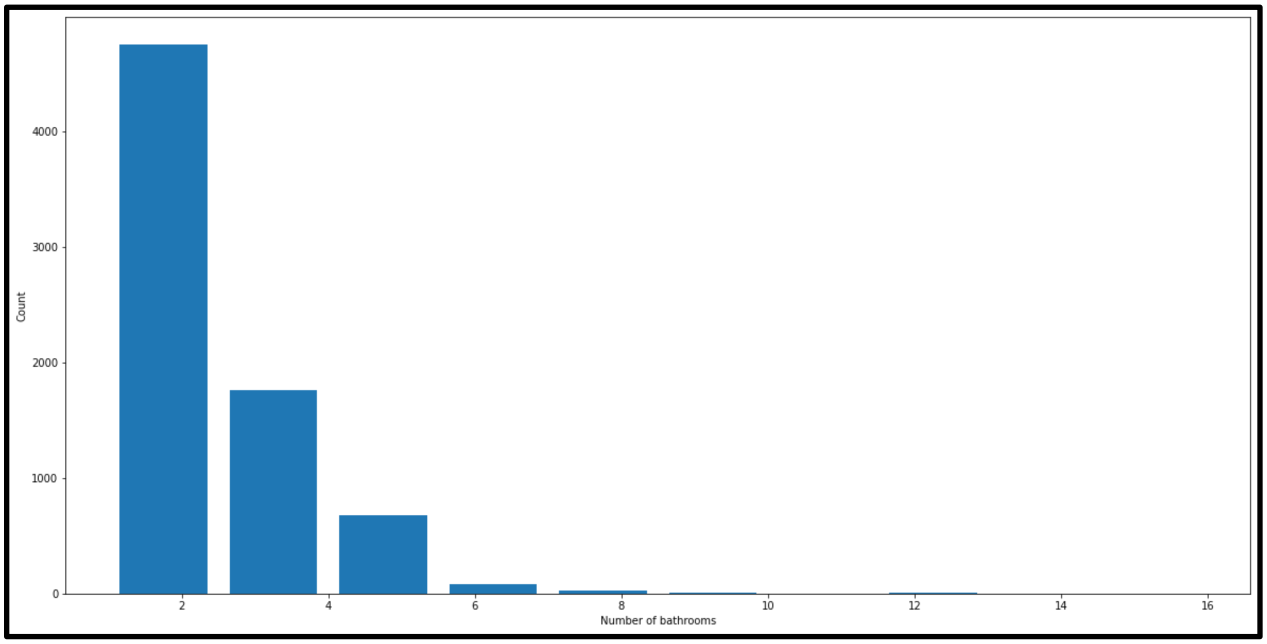
```
In [42]: import matplotlib
         matplotlib.rcParams["figure.figsize"] = (20,10)
         plt.hist(df8.price_per_sqft,rwidth=0.8)
         plt.xlabel("Price Per Square Feet")
         plt.ylabel("Count")

Out[42]: Text(0, 0.5, 'Count')
```

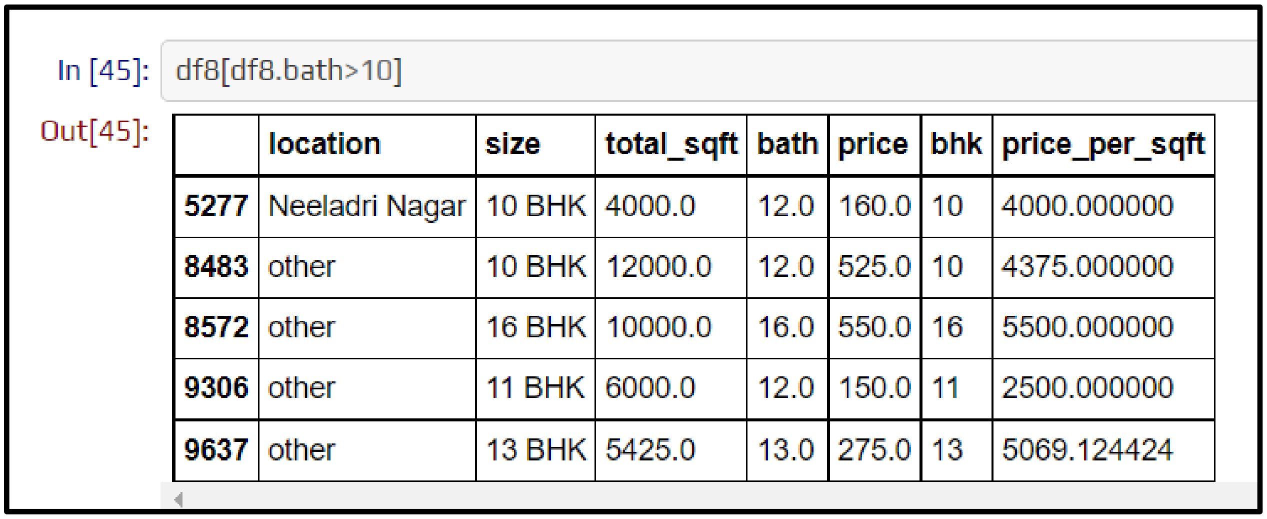


Outlier Removal Using Bathrooms Feature.



```
In [43]: df8.bath.unique()
Out[43]: array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])

In [44]: plt.hist(df8.bath,rwidth=0.8)
         plt.xlabel("Number of bathrooms")
         plt.ylabel("Count")
Out[44]: Text(0, 0.5, 'Count')
```

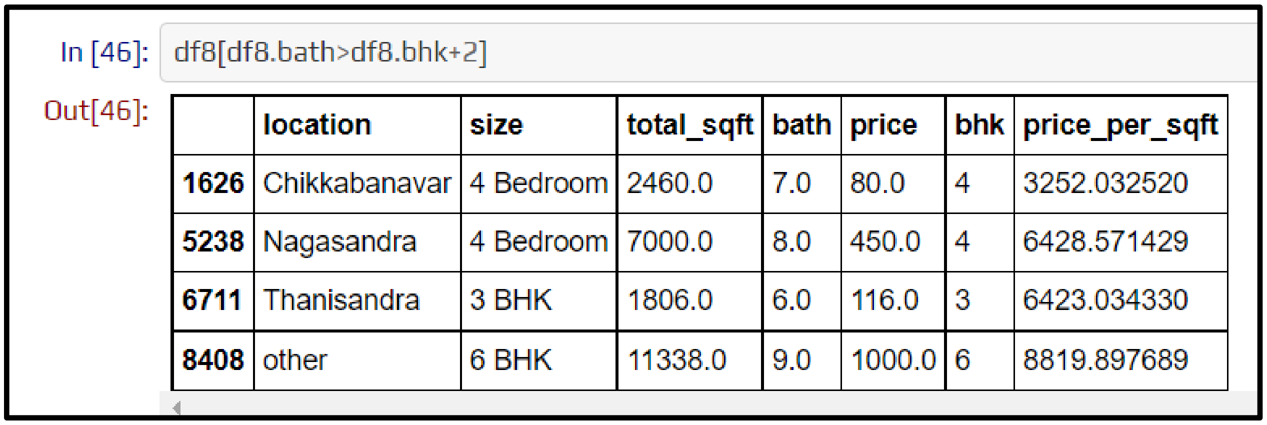




```
In [45]: df8[df8.bath>10]
Out[45]:
```

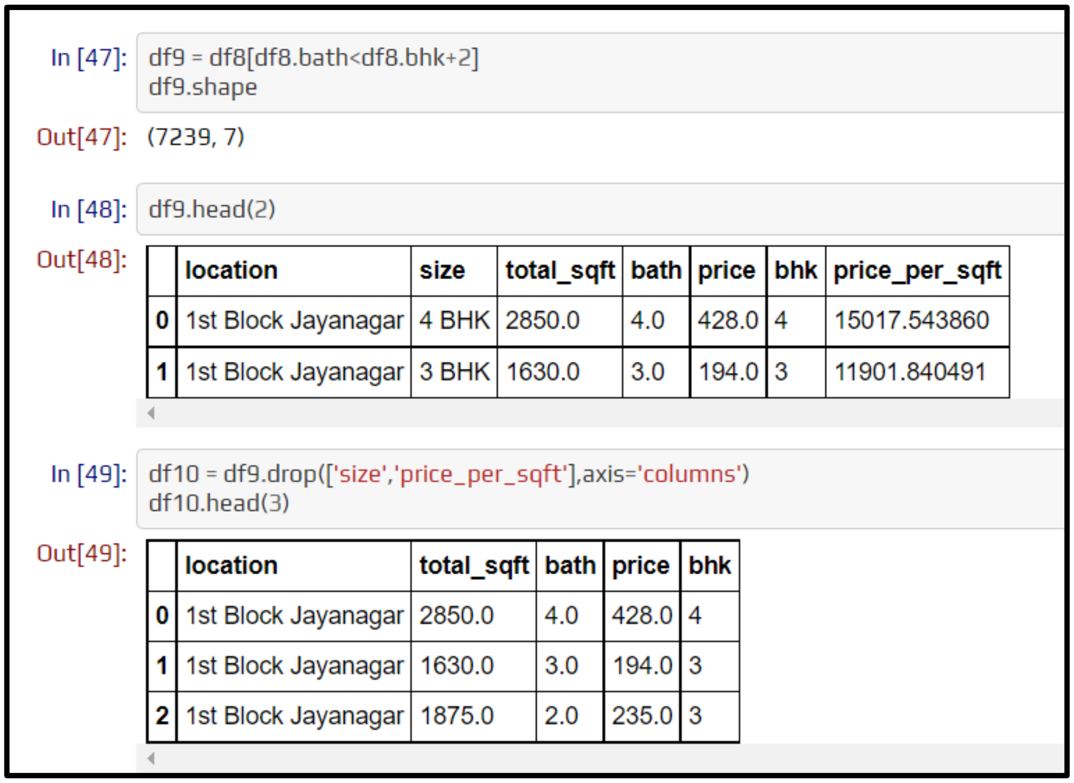
	location	size	total_sqft	bath	price	bhk	price_per_sqft
5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8483	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8572	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9306	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9637	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

It is unusual to have 2 more bathrooms than several bedrooms in a home.


```
In [46]: df8[df8.bath>df8.bhk+2]
```
Out[46]:

	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
8408	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

Again the business manager has a conversation with you (i.e. a data scientist) that if you have a 4 bedroom home and even if you have a bathroom in all 4 rooms plus one guest bathroom, you will have a total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed.

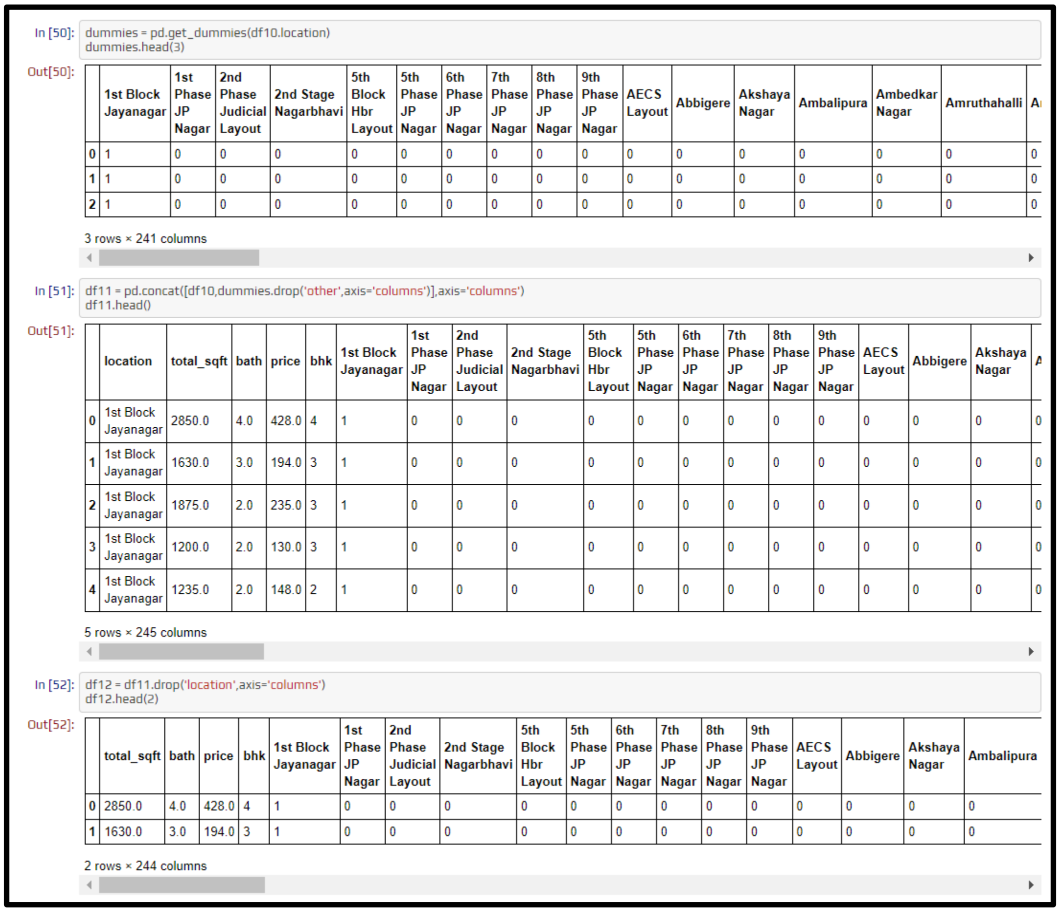

```
In [47]: df9 = df8[df8.bath<df8.bhk+2]
         df9.shape
Out[47]: (7239, 7)

In [48]: df9.head(2)
```
Out[48]:

	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

```
In [49]: df10 = df9.drop(['size','price_per_sqft'],axis='columns')
         df10.head(3)
```
Out[49]:

	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

Use One Hot Encoding For Location.



```
In [50]: dummies = pd.get_dummies(df10.location)
         dummies.head(3)
```

Out[50]:

	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	AECS Layout	Abbigere	Akshaya Nagar	Ambalipura	Ambedkar Nagar	Amruthahalli	A
0	1	0	0	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	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3 rows × 241 columns

```
In [51]: df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
         df11.head()
```

Out[51]:

	location	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	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	AECS Layout	Abbigere	Akshaya Nagar	A
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	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	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	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	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	0	0	0

5 rows × 245 columns

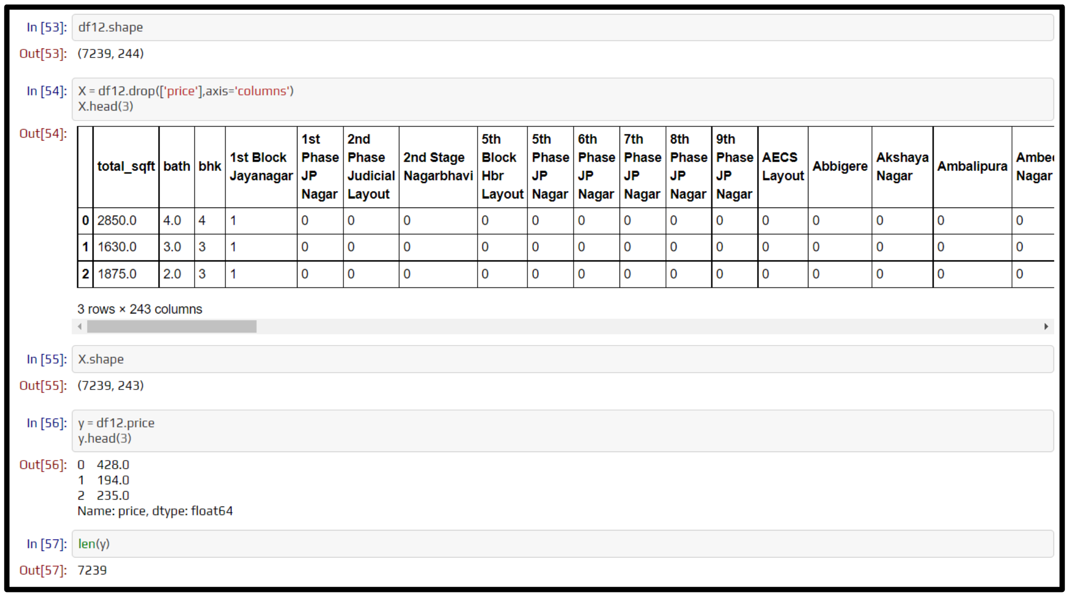
```
In [52]: df12 = df11.drop('location',axis='columns')
         df12.head(2)
```

Out[52]:

	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	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	AECS Layout	Abbigere	Akshaya Nagar	Ambalipura
0	2850.0	4.0	428.0	4	1	0	0	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	0	0

2 rows × 244 columns

4.1.10 Building a Model



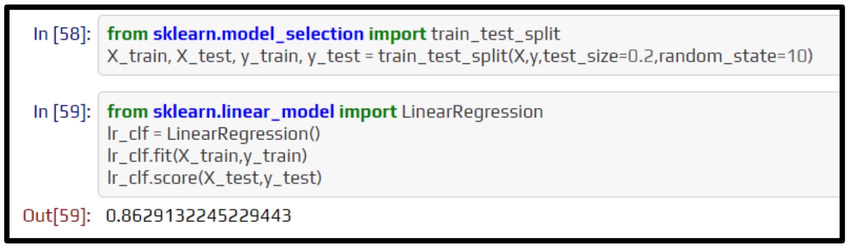
```
In [53]: df12.shape
Out[53]: (7239, 244)

In [54]: X = df12.drop(['price'],axis='columns')
         X.head(3)
```

Out[54]:

	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	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	AECS Layout	Abbigere	Akshaya Nagar	Ambalipura	Ambe Nagar
0	2850.0	4.0	4	1	0	0	0	0	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	0	0	0	0
2	1875.0	2.0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

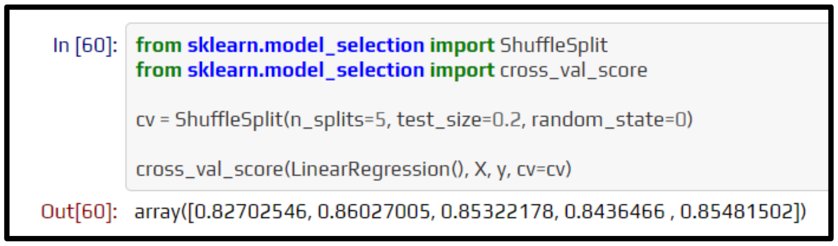
3 rows × 243 columns

```
In [55]: X.shape
Out[55]: (7239, 243)

In [56]: y = df12.price
         y.head(3)
Out[56]: 0    428.0
         1    194.0
         2    235.0
         Name: price, dtype: float64

In [57]: len(y)
Out[57]: 7239
```



```
In [58]: 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)

In [59]: from sklearn.linear_model import LinearRegression
         lr_clf = LinearRegression()
         lr_clf.fit(X_train,y_train)
         lr_clf.score(X_test,y_test)

Out[59]: 0.8629132245229443
```

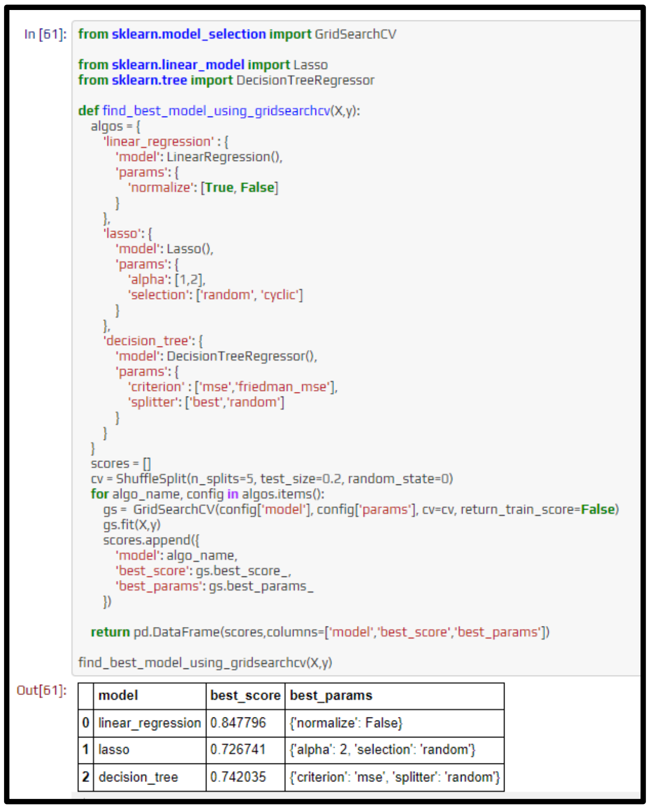
Use K Fold cross-validation to measure the accuracy of our LinearRegression model.



```
In [60]: 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)

         cross_val_score(LinearRegression(), X, y, cv=cv)

Out[60]: array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])
```

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

Find the best model using GridSearchCV.



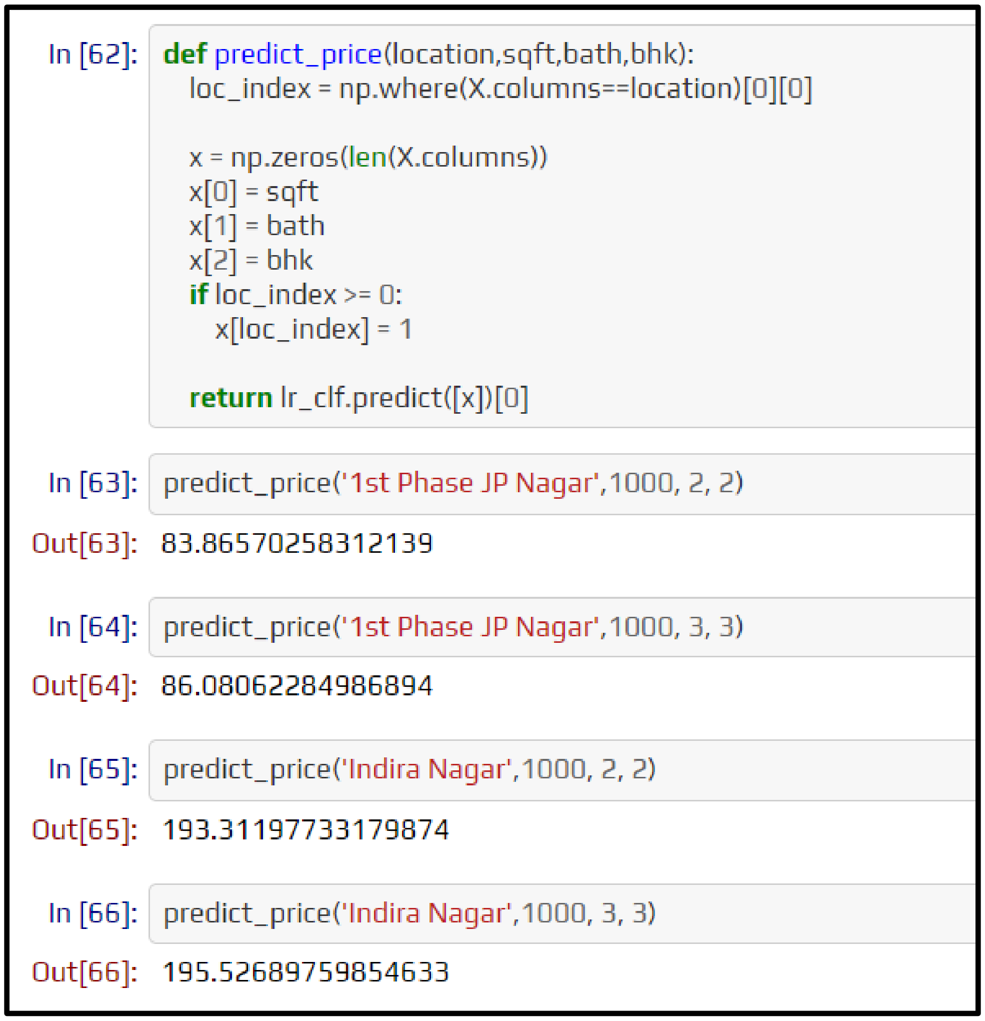
```
In [61]: from sklearn.model_selection import GridSearchCV

         from sklearn.linear_model import Lasso
         from sklearn.tree import DecisionTreeRegressor

         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)
```

Out[61]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726741	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.742035	{'criterion': 'mse', 'splitter': 'random'}

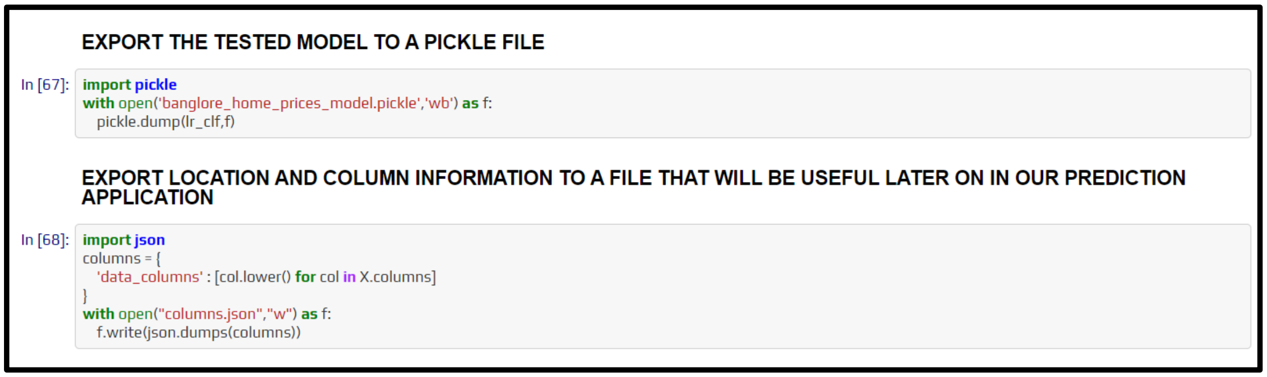
Based on the above results we can say that LinearRegression gives the best score. Hence, we will use that.

5.1.11 Test the Model for few properties



```
In [62]: def predict_price(location,sqft,bath,bhk):
             loc_index = np.where(X.columns==location)[0][0]

             x = np.zeros(len(X.columns))
             x[0] = sqft
             x[1] = bath
             x[2] = bhk
             if loc_index >= 0:
                 x[loc_index] = 1

             return lr_clf.predict([x])[0]

In [63]: predict_price('1st Phase JP Nagar',1000, 2, 2)

Out[63]: 83.86570258312139

In [64]: predict_price('1st Phase JP Nagar',1000, 3, 3)

Out[64]: 86.08062284986894

In [65]: predict_price('Indira Nagar',1000, 2, 2)

Out[65]: 193.31197733179874

In [66]: predict_price('Indira Nagar',1000, 3, 3)

Out[66]: 195.52689759854633
```

5.1.12 Export the tested model to a pickle file



EXPORT THE TESTED MODEL TO A PICKLE FILE

```
In [67]: import pickle
         with open('banglore_home_prices_model.pickle','wb') as f:
             pickle.dump(lr_clf,f)
```

EXPORT LOCATION AND COLUMN INFORMATION TO A FILE THAT WILL BE USEFUL LATER ON IN OUR PREDICTION APPLICATION

```
In [68]: import json
         columns = {
             'data_columns' : [col.lower() for col in X.columns]
         }
         with open("columns.json","w") as f:
             f.write(json.dumps(columns))
```

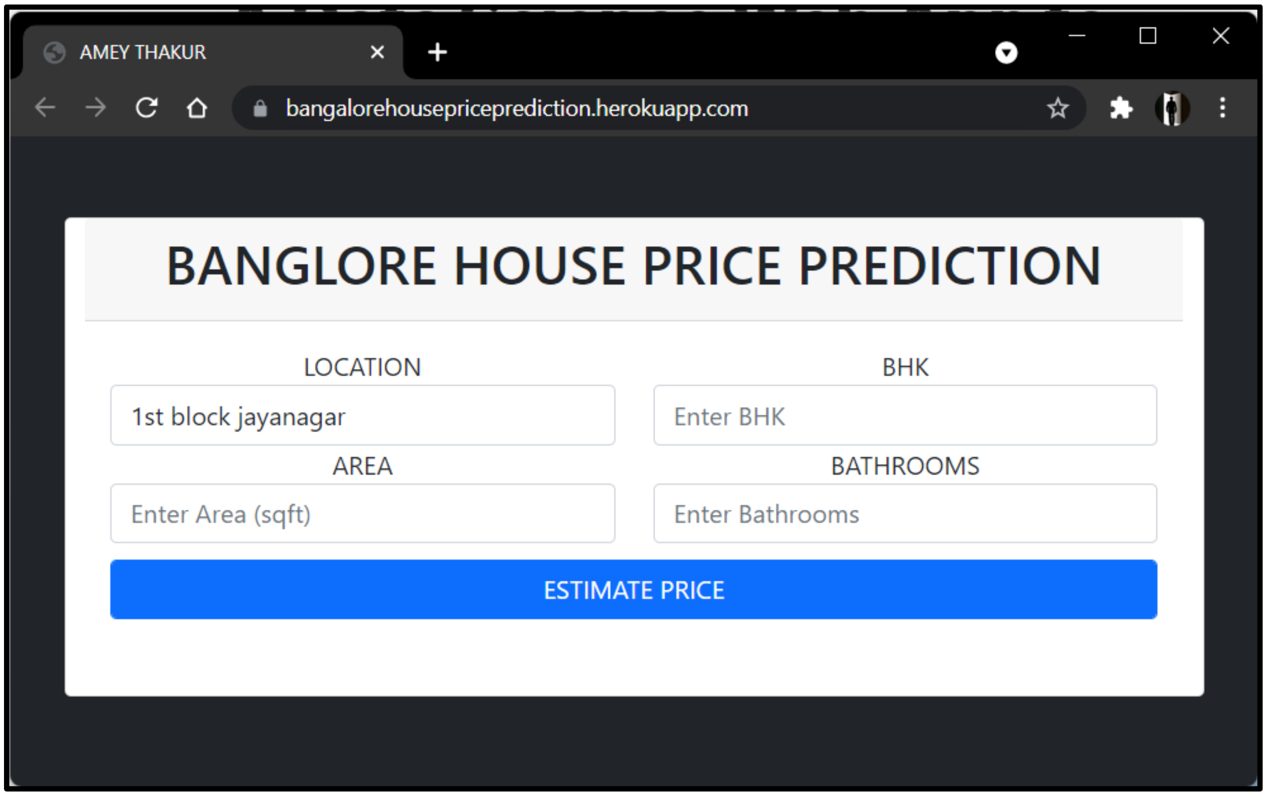
DEPLOYMENT

Heroku is a cloud Platform as a container-based Service (PaaS). Heroku is used by developers to launch, manage, and grow contemporary programs.

6.1 Steps to deploy the model on Heroku

```
heroku login

git init

heroku create bangalorehousepriceprediction

git add .

git commit -am "initial commit"

git push heroku master
```

RESULTS


AMEY THAKUR

bangalorehousepriceprediction.herokuapp.com

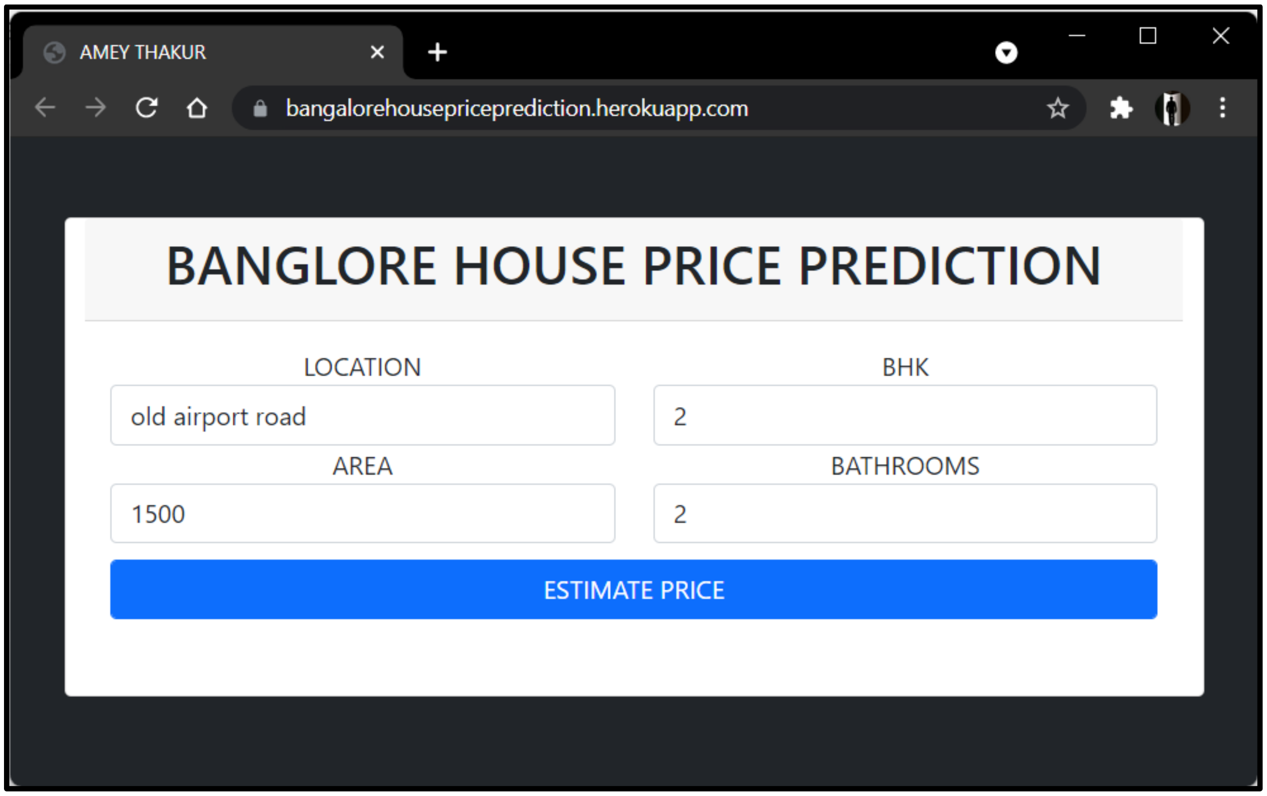
BANGLORE HOUSE PRICE PREDICTION

LOCATION: 1st block jayanagar

BHK: Enter BHK

AREA: Enter Area (sqft)

BATHROOMS: Enter Bathrooms


AMEY THAKUR

bangalorehousepriceprediction.herokuapp.com

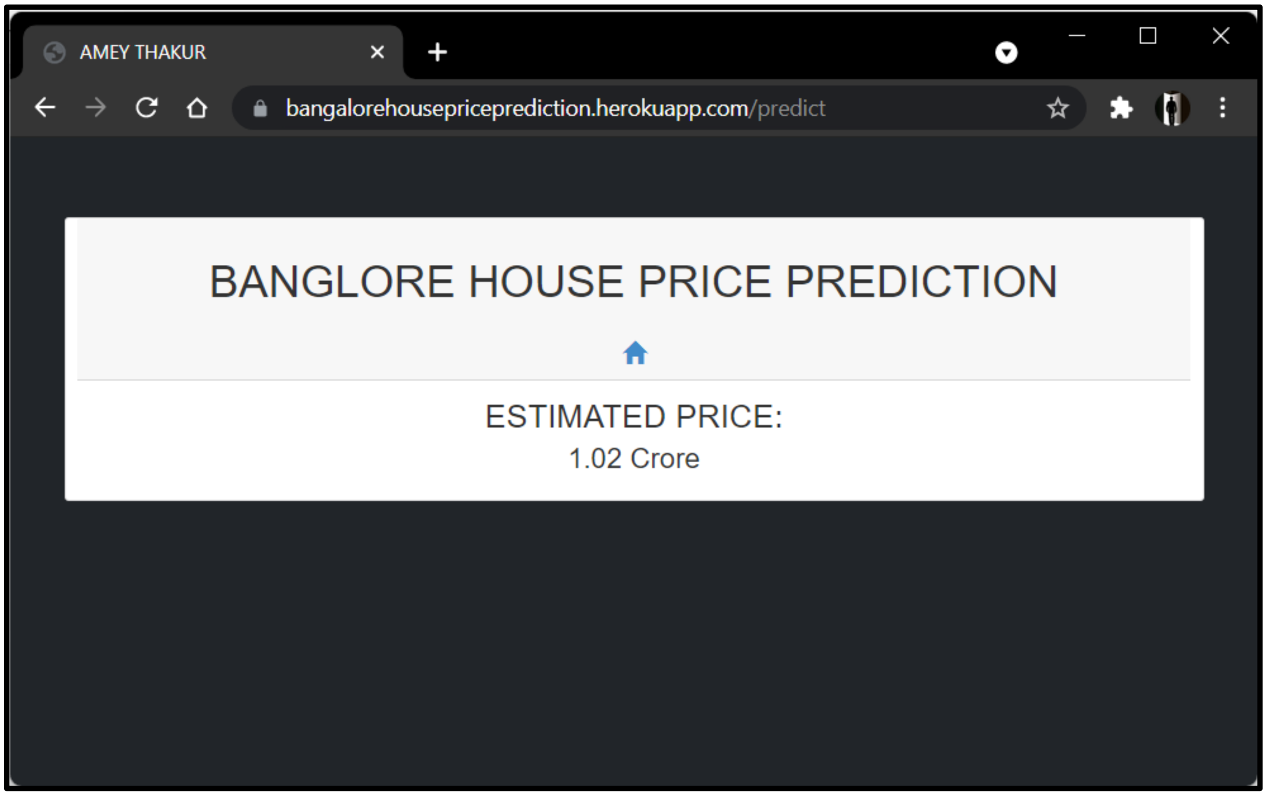
BANGLORE HOUSE PRICE PREDICTION

LOCATION: old airport road

BHK: 2

AREA: 1500

BATHROOMS: 2



BANGLORE HOUSE PRICE PREDICTION

ESTIMATED PRICE:
1.02 Crore

CONCLUSION

With several characteristics, the suggested method predicts the property price in Bangalore. We experimented with different Machine Learning algorithms to get the best model. When compared to all other algorithms, the Decision Tree Algorithm achieved the lowest loss and the greatest R-squared. Flask was used to create the website.

Let's see how our project pans out. Open the HTML web page we generated and run the app.py file in the backend. Input the property's square footage, the number of bedrooms, the number of bathrooms, and the location, then click 'ESTIMATE PRICE.' Yay! We forecasted the cost of what may be someone's ideal home.