

12 AUGUST 2022

HOUSE PRICE PREDICTION



B.TECH CSE CCET, SECTOR-26

SUBMITTED BY :

IQMAN SINGH BHATIA

ABHINAV BAINS

DIVYANSH AGGARWAL

SAKSHAM KAUSHIK

SUBMITTED TO :

MRS. URVASHI NAG

MR. AMRENDRA
SHARAN SIR

INTRODUCTION

- Due to increase in urbanisation, there is an increase in demand for renting houses and purchasing houses. Therefore, to determine a more effective way to calculate house price that accurately reflects the market price becomes a hot topic.
- The paper focuses on finding the house price accurately by using machine learning.
- This will be helpful for both the sellers and buyers for finding the best price for the house.

LANGUAGES AND LIBRARIES USED:

- PYTHON
- PANDAS
- NUMPY
- MATPLOTLIB
- The Jupyter Notebook is the original web application for creating and sharing computational documents



DATA SET SOURCE:

Our Data comes from a Kaggle competition named “[banglore_home_prices](#)”.

Bengaluru_House_Data									
area_type	availability	location	size	society	total_sqft	bath	balcony	price	
Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2	1	39.07	
Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5	3	120	
Built-up Area	Ready To Move	Uttarahalli	3 BHK		1440	2	3	62	
Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3	1	95	
Super built-up Area	Ready To Move	Kothanur	2 BHK		1200	2	1	51	
Super built-up Area	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2	1	38	
Super built-up Area	18-May	Old Airport Road	4 BHK	Jaades	2732	4		204	
Super built-up Area	Ready To Move	Rajaji Nagar	4 BHK	Brway G	3300	4		600	
Super built-up Area	Ready To Move	Marathahalli	3 BHK		1310	3	1	63.25	
Plot Area	Ready To Move	Gandhi Bazar	6 Bedroom		1020	6		370	
Super built-up Area	18-Feb	Whitefield	3 BHK		1800	2	2	70	
Plot Area	Ready To Move	Whitefield	4 Bedroom	Prrry M	2785	5	3	295	
Super built-up Area	Ready To Move	7th Phase JP Nagar	2 BHK	Shncyes	1000	2	1	38	
Built-up Area	Ready To Move	Gottigere	2 BHK		1100	2	2	40	
Plot Area	Ready To Move	Sarjapur	3 Bedroom	Skityer	2250	3	2	148	
Super built-up Area	Ready To Move	Mysore Road	2 BHK	PrntaEn	1175	2	2	73.5	
Super built-up Area	Ready To Move	Bisuvanahalli	3 BHK	Prtyel	1180	3	2	48	
Super built-up Area	Ready To Move	Raja Rajeshwari Nagar	3 BHK	GrrvaGr	1540	3	3	60	
Super built-up Area	Ready To Move	Ramakrishnappa Layout	3 BHK	PeBayle	2770	4	2	290	
Super built-up Area	Ready To Move	Manayata Tech Park	2 BHK		1100	2	2	48	
Built-up Area	Ready To Move	Kengeri	1 BHK		600	1	1	15	
Super built-up Area	19-Dec	Binny Pete	3 BHK	She 2rk	1755	3	1	122	
Plot Area	Ready To Move	Thanisandra	4 Bedroom	Soitya	2800	5	2	380	

**! NOW LETS START
WITH OUR MODEL !**

IMPORTING 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)
```

DATA LOAD: LOAD BANGLORE HOME PRICES INTO A DATAFRAME

```
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

```
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
```

Drop features that are not required to build our model

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

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

DATA CLEANING: HANDLE NA VALUES

```
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)
```

FEATURE ENGINEERING

Add new feature(integer) for bkh (Bedrooms Hall Kitchen)

In [12]:

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

```
C:\ProgramData\Anaconda3\lib\site-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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

Out[12]:

```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
       13, 18], dtype=int64)
```

Explore total_sqft feature

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

Above shows that total_sqft can be a range (e.g. 2100-2850). For such case we can just take average of min and max value in the range. There are other cases such as 34.46Sq. 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 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

FEATURE ENGINEERING

Add 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 [69]:

```
df5.to_csv("bhp.csv", index=False)
```

Examine locations which is a categorical variable. We need to apply dimensionality reduction technique here to reduce number of locations

In [22]:

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

```
Out[22]: Whitefield          533  
Sarjapur Road            392  
Electronic City           304  
Kanakpura Road           264  
Thanisandra                235  
Yelahanka                  210  
Uttarahalli                 186  
Hebbal                      176  
Marathahalli                 175  
Raja Rajeshwari Nagar      171  
Bannerghatta Road           151  
Hennur Road                  150  
7th Phase JP Nagar           148  
Haralur Road                  141  
Electronic City Phase II       131  
Rajaji Nagar                  106  
Chandapura                     98  
Bellandur                      96  
KR Puram                        88  
Hoodi                           88  
Electronics City Phase 1        87  
Yeshwanthpur                   85  
Begur Road                      84  
Sarjapur                         80  
Kasavanahalli                  79  
Harlur                           79  
Hormavu                          74  
Banashankari                    74  
Ramamurthy Nagar                  72  
Koramangala                      72  
...  
Ckikkakammana Halli             1  
Neelasandra                      1  
Gangondanahalli                  1  
Agara Village                      1  
Sundara Nagar                      1  
Binny Mills Employees Colony        1  
Adugodi                           1  
Uvce Layout                         1  
Kenchanehalli R R Nagar           1  
Whietfield,                      1  
manyata                            1  
Air View Colony                      1  
Thavarekere                         1  
Muthyala Nagar                      1  
Haralur Road,                      1  
Manonarayananapalya                 1  
GKW Layout                           1  
Marathalli bridge                  1  
Banashankari 6th Stage ,Subramanyapura 1  
anjananager magdi road               1  
akshaya nagar t c palya              1  
Indiranagar HAL 2nd Stage            1  
Maruthi HBCS Layout                  1  
Gopal Reddy Layout                  1  
High grounds                         1  
CMH Road                            1  
Chambenahalli                      1  
Sarvobhogam Nagar                  1  
Ex-Servicemen Colony Dinnur Main Road R.T.Nagar 1  
Bilal Nagar                         1  
Name: location, Length: 1287, dtype: int64
```

[1]

```
In [23]: location_stats.values.sum()  
Out[23]: 13200  
  
In [24]: len(location_stats[location_stats>10])  
Out[24]: 240  
  
In [25]: len(location_stats)  
Out[25]: 1287  
  
In [26]: len(location_stats[location_stats<=10])  
Out[26]: 1047
```

[2]

DIMENSIONALITY REDUCTION

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

```
In [27]: location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

```
Out[27]:
BTM 1st Stage          10
Sector 1 HSR Layout    10
Ganga Nagar            10
Naganathapura          10
1st Block Koramangala  10
Thyagaraja Nagar       10
Dairy Circle           10
Nagadevanahalli        10
Sadashiva Nagar        10
Gunjur Palya           10
Dodsworth Layout       10
Basapura                10
Kalkere                10
Nagappa Reddy Layout   10
2nd Phase JP Nagar     9
Yemlur                  9
Medahalli              9
Kaverappa Layout       9
Ejipura                 9
Mathikere               9
Lingarajapuram         9
Peenya                  9
Vignana Nagar          9
B Narayananpura        9
Chandra Layout         9
Jakkur Plantation      9
Banagiri Nagar          9
Chennammana Kere        9
Richmond Town           9
Vishwanatha Nagenahalli 9
.
.
.
Ckikkakammana Halli    1
Neelasandra             1
Gangondanahalli         1
Agara Village            1
Sundara Nagar           1
Binny Mills Employees Colony 1
Adugodi                 1
Uvce Layout              1
Kenchanehalli R R Nagar 1
Whietfield,              1
manyata                  1
Air View Colony          1
Thavarekere              1
Muthyala Nagar           1
Haralur Road,             1
Manonarayanapalya        1
GKW Layout                1
Marathalli bridge         1
Bananashankari 6th Stage ,Subramanyapura 1
anjananager magdi road    1
akshaya nagar t c palya    1
Indiranagar HAL 2nd Stage 1
Maruthi HBCS Layout       1
Gopal Reddy Layout        1
High grounds              1
CMH Road                  1
Chambenahalli             1
Sarvobhogam Nagar         1
Ex-Servicemen Colony Dinnur Main Road R.T.Nagar 1
Bilal Nagar               1
Name: location, Length: 1047, dtype: int64
```

[1]

```
In [28]: len(df5.location.unique())
Out[28]: 1287
In [29]: df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
len(df5.location.unique())
Out[29]: 241
In [30]: df5.head(10)
Out[30]:
   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
```

[2]

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 than 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 [31]:

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

Out[31]:

	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	Devarachikkannahalli	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 above data points. We have 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

In [32]:

```
df5.shape
```

Out[32]:

```
(13200, 7)
```

In [33]:

```
df6 = df5[-(df5.total_sqft/df5.bhk<300)]
df6.shape
```

Out[33]:

```
(12456, 7)
```

OUTLIER REMOVAL USING STANDARD DEVIATION AND MEAN

```
In [34]: df6.price_per_sqft.describe()
```

```
Out[34]: 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 min price per sqft is 267 rs/sqft whereas max is 12000000, this shows a wide variation in property prices. We should remove outliers per location using mean and one standard deviation

```
In [35]: 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
```

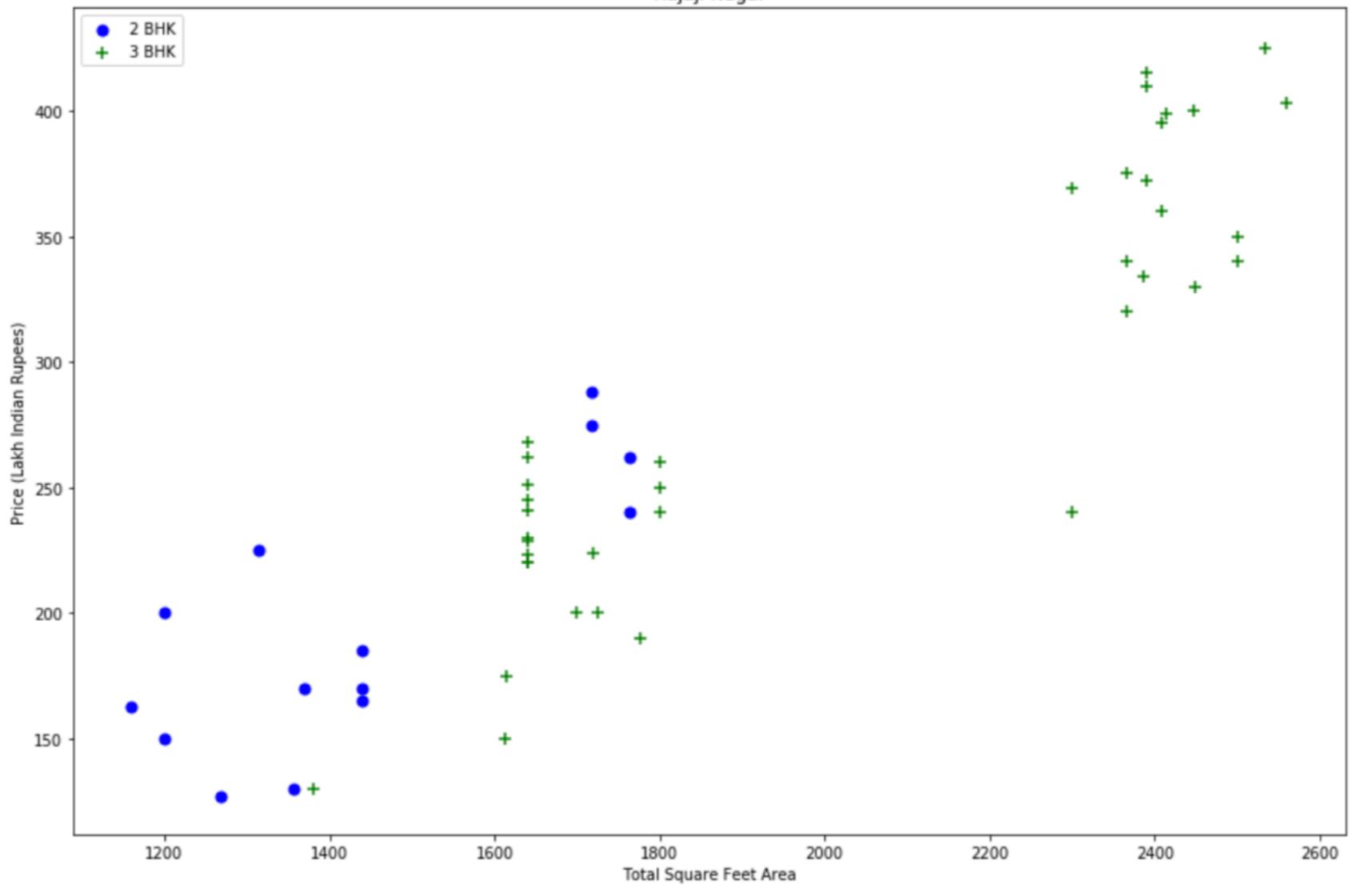
```
Out[35]: (10242, 7)
```

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

```
In [36]: 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")
```

Rajaji Nagar



```
In [37]: plot_scatter_chart(df7, "Hebbal")
```

[1]

We should also remove properties where for same location, the price of (for example) 3 bedroom apartment is less than 2 bedroom apartment (with same square ft area). What we will do is for a given location, we will build a dictionary of stats per bkh, 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 mean price_per_sqft of 1 BHK apartment

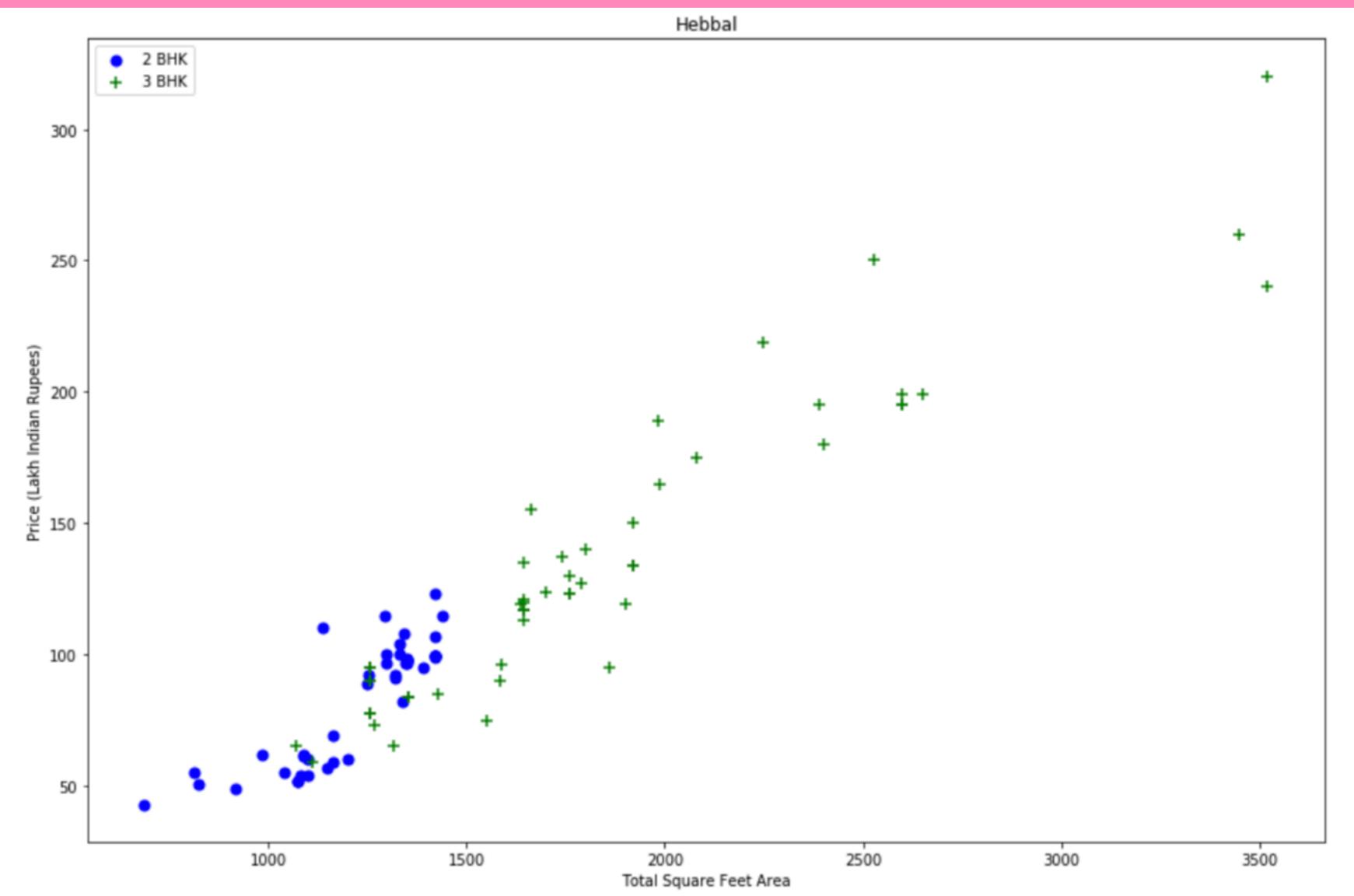
```
In [38]: def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bkh_stats = {}
        for bkh, bkh_df in location_df.groupby('bkh'):
            bkh_stats[bkh] = {
                'mean': np.mean(bkh_df.price_per_sqft),
                'std': np.std(bkh_df.price_per_sqft),
                'count': bkh_df.shape[0]
            }
        for bkh, bkh_df in location_df.groupby('bkh'):
            stats = bkh_stats.get(bkh-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices, bkh_df[bkh_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[38]: (7317, 7)

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

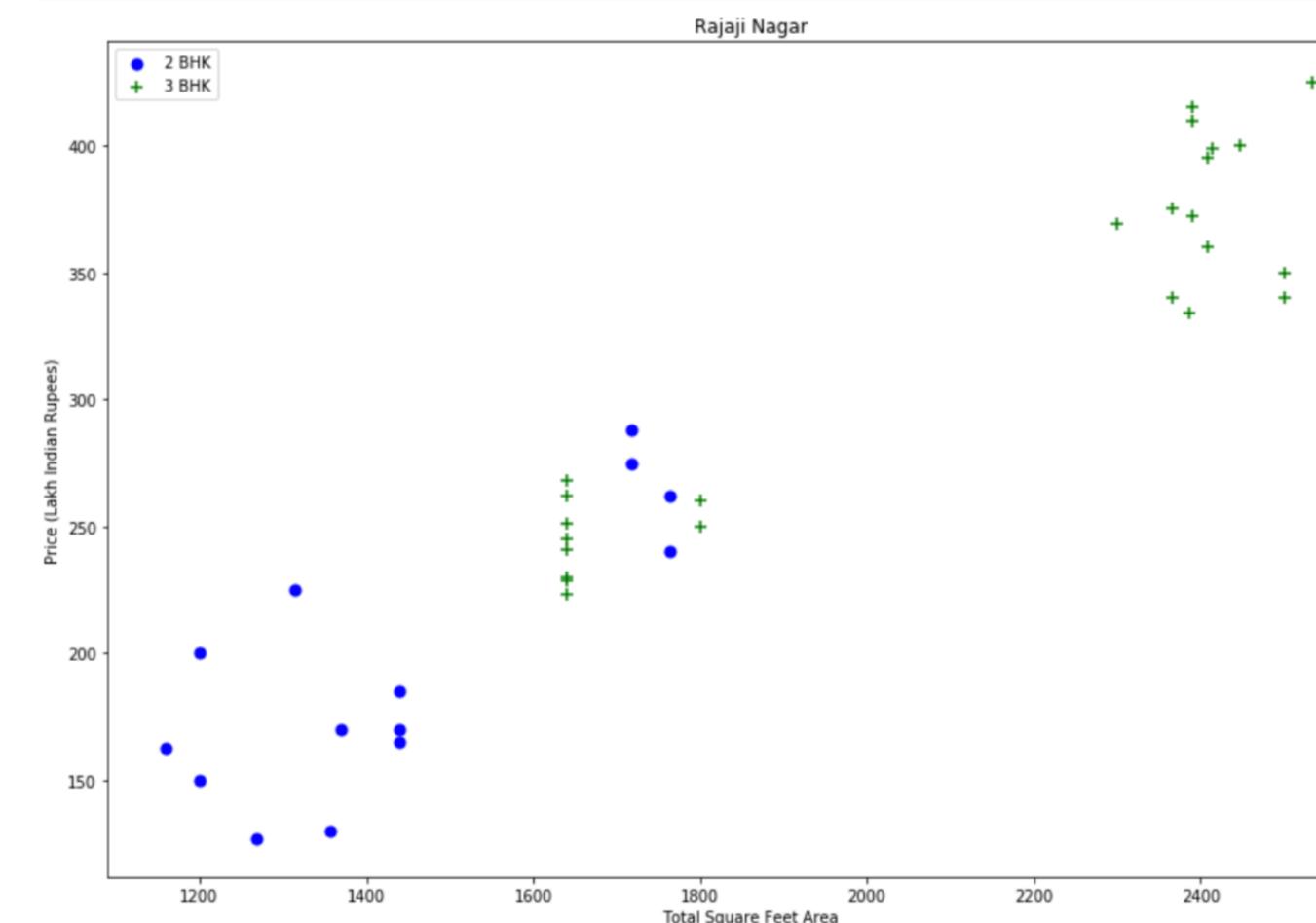
[3]

Hebbal



[2]

```
In [39]: plot_scatter_chart(df8, "Rajaji Nagar")
```

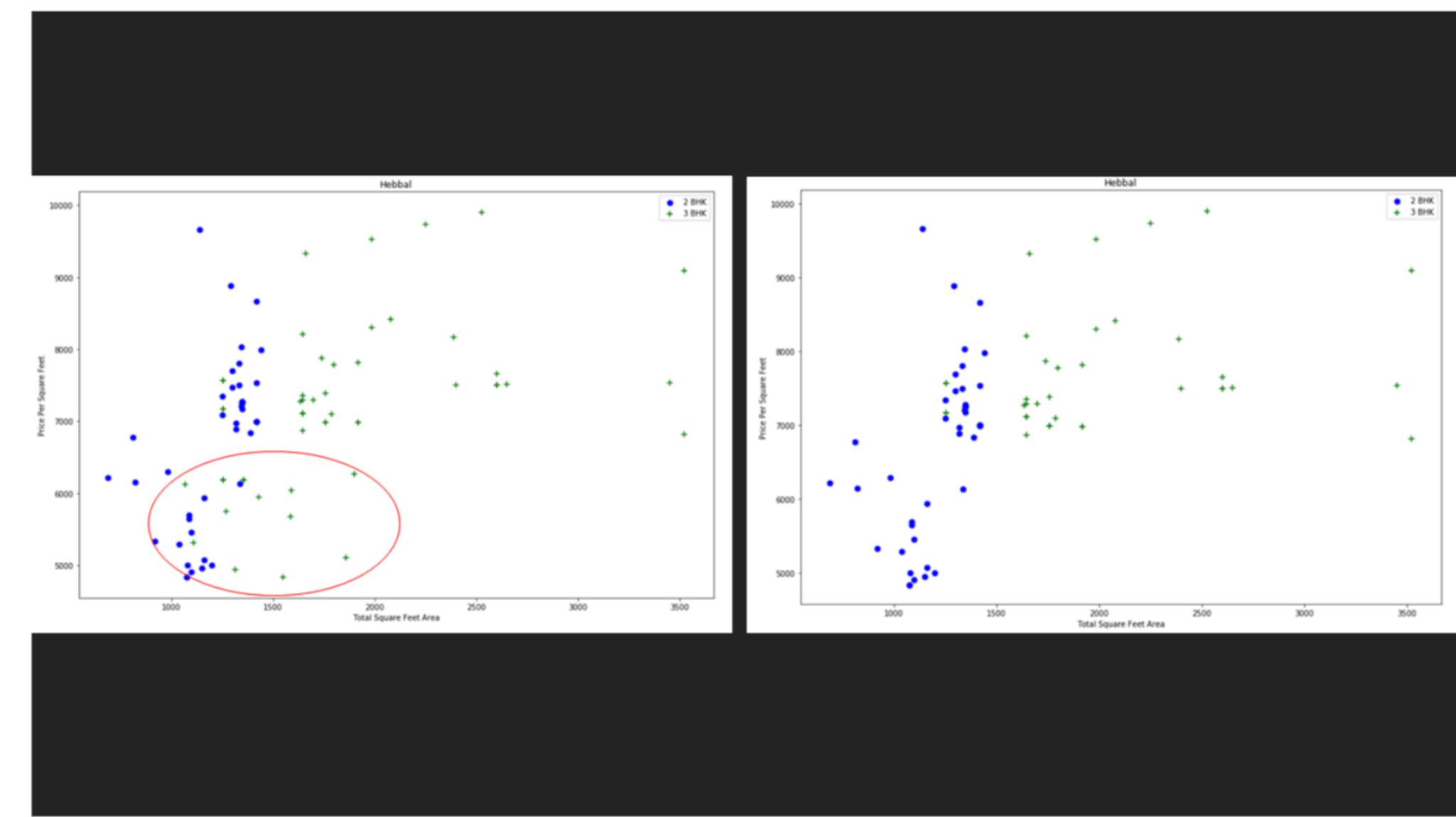


```
In [40]: plot_scatter_chart(df8, "Hebbal")
```

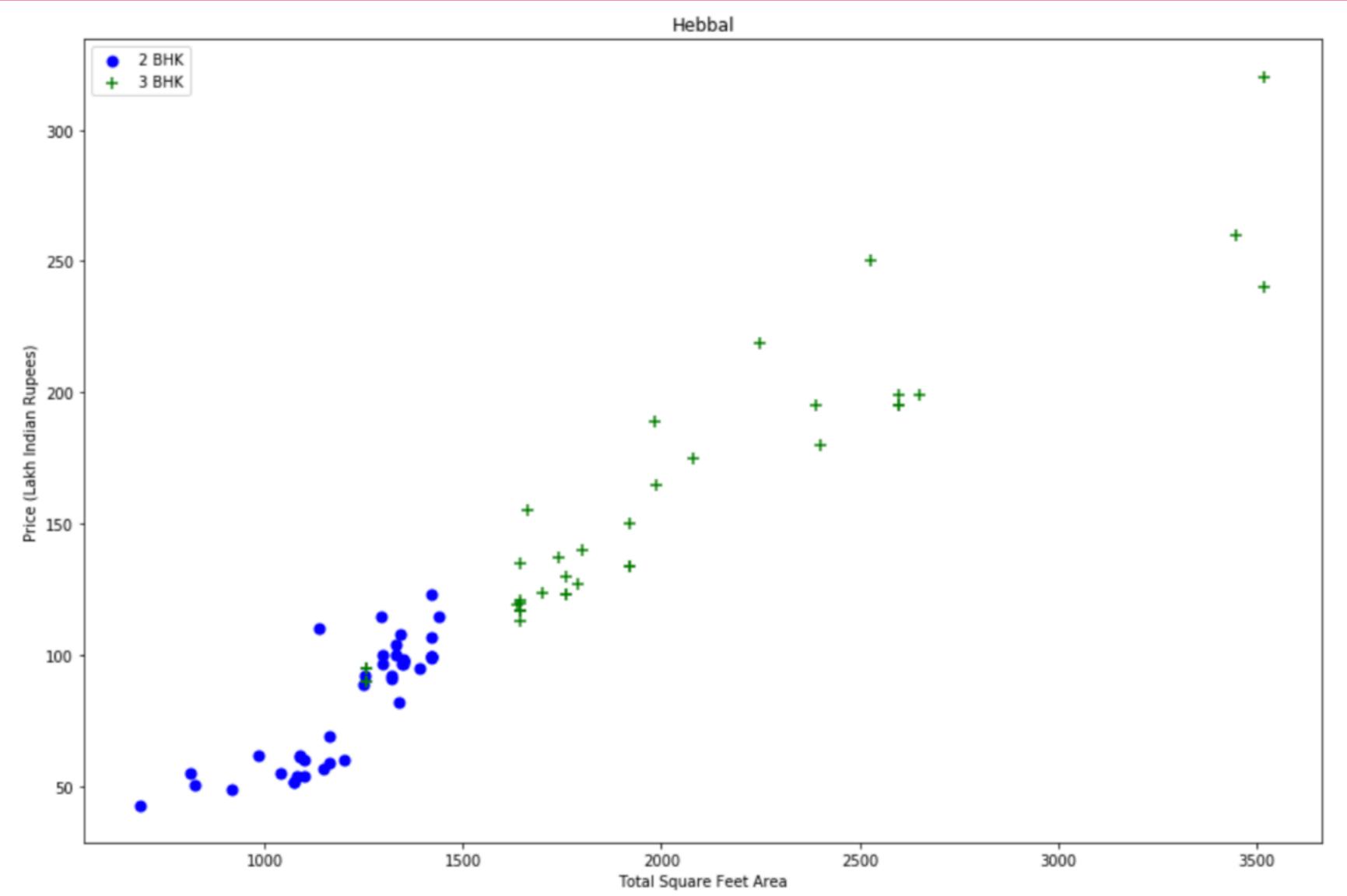
[4]

[3]

Before and after outlier removal: Hebbal

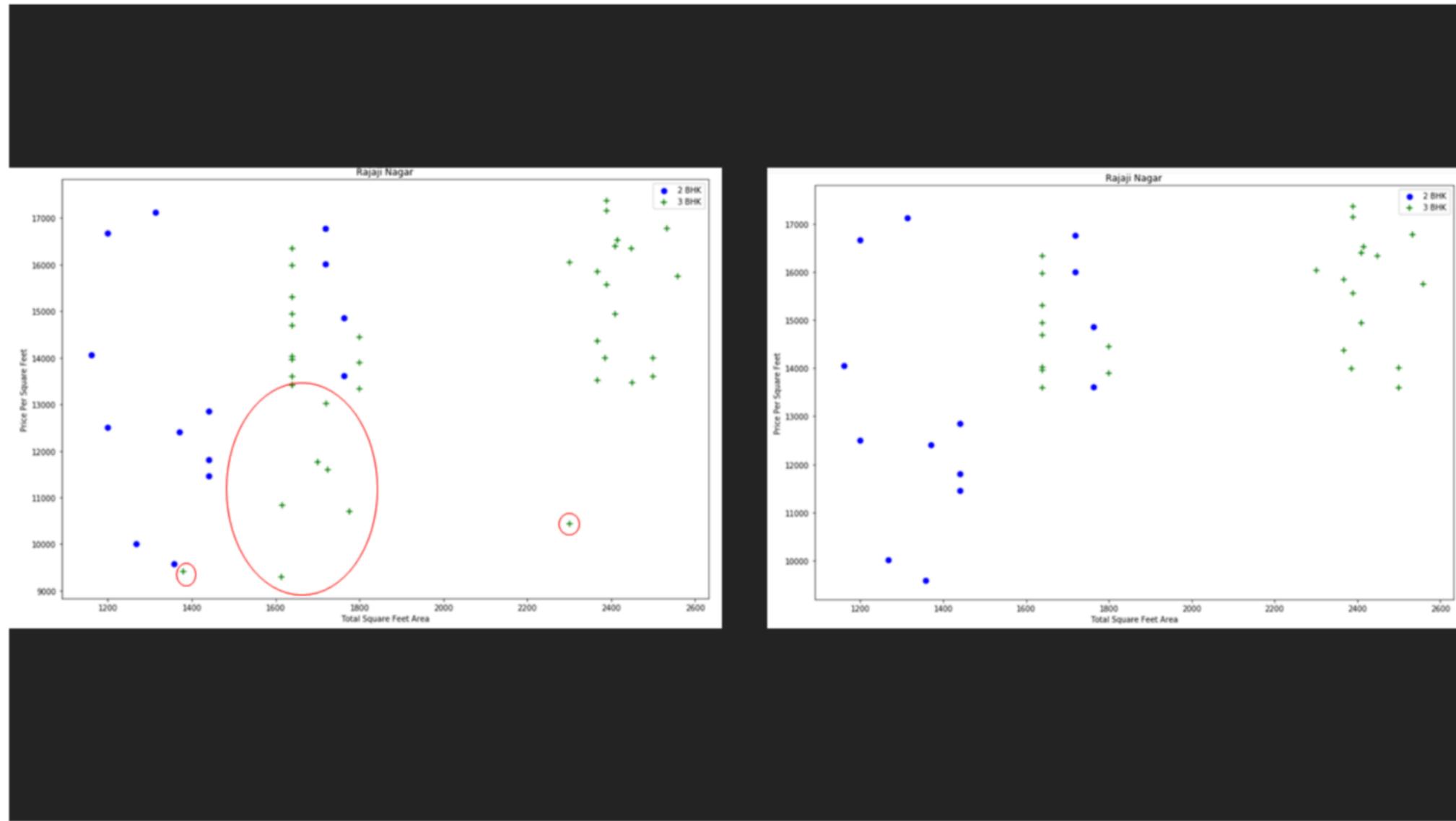


[1]

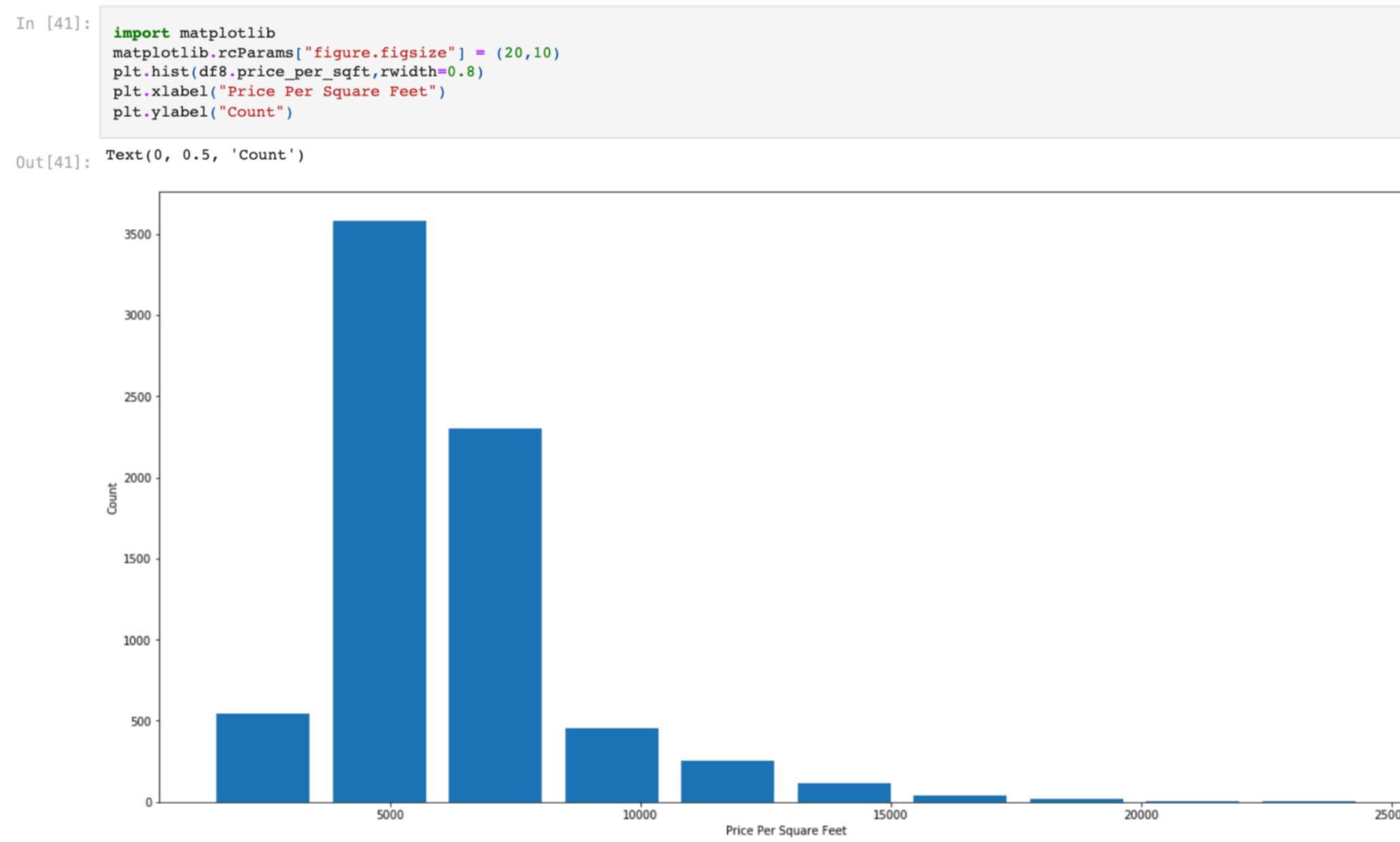


Based on above charts we can see that data points highlighted in red below are outliers and they are being removed due to remove_bhk_outliers function

Before and after outlier removal: Rajaji Nagar



[2]



[4]

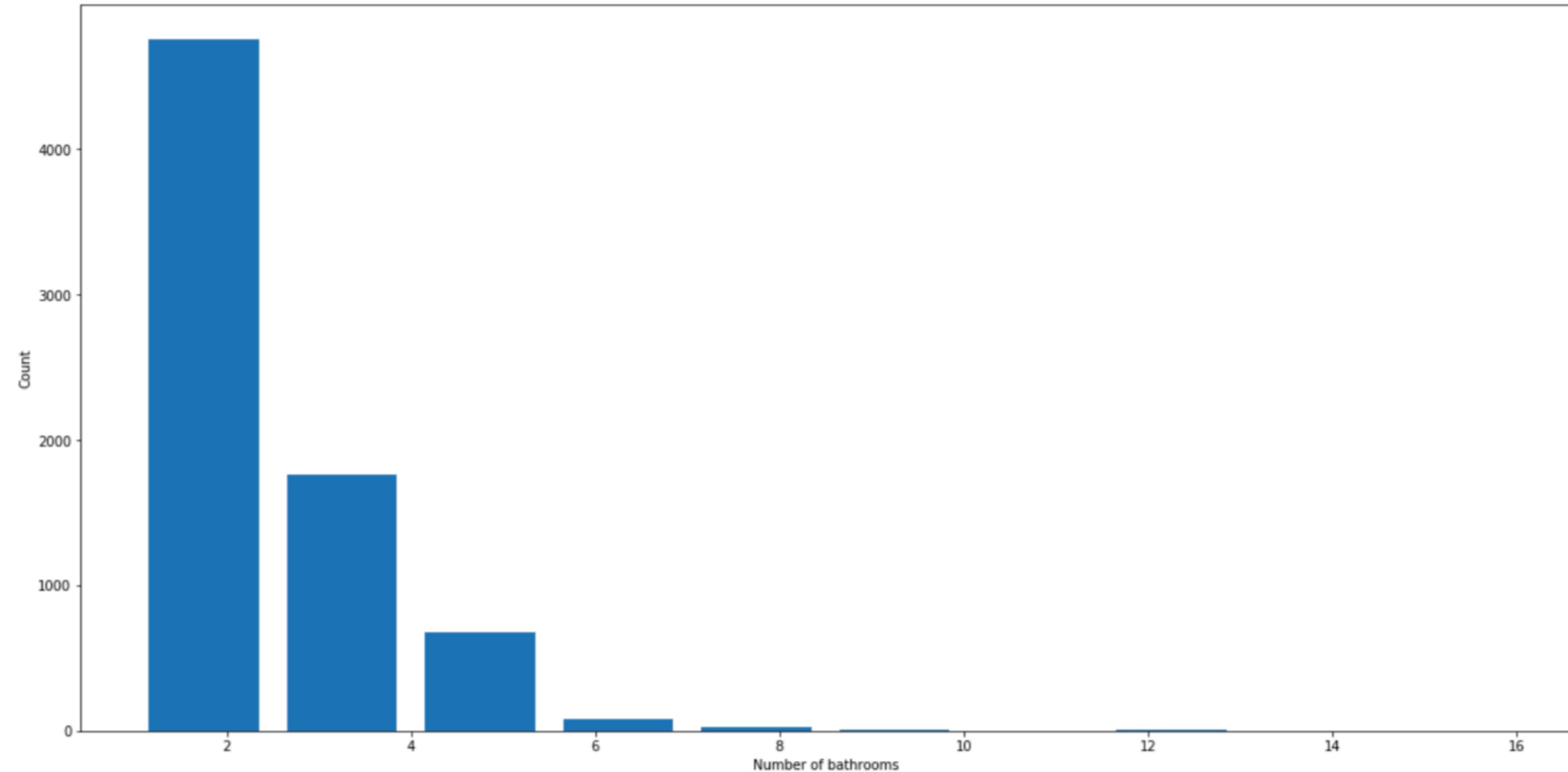
OUTLIER REMOVAL USING BATHROOMS FEATURE

```
In [42]: df8.bath.unique()
```

```
Out[42]: array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

```
In [43]: plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

```
Out[43]: Text(0, 0.5, 'Count')
```



```
In [44]: df8[df8.bath>10]
```

```
Out[44]:
```

	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 number of bedrooms in a home

```
In [45]:
```

```
df8[df8.bath>df8.bhk+2]
```

```
Out[45]:
```

	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 4 bedroom home and even if you have bathroom in all 4 rooms plus one guest bathroom, you will have total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed

```
In [46]:
```

```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

```
Out[46]:
```

```
(7239, 7)
```

```
In [47]:
```

```
df9.head(2)
```

```
Out[47]:
```

	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 [48]:
```

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

```
Out[48]:
```

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

```
Out[49]:
```

	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi Layout	5th Block Hbr	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	Yelahanka
0	1	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
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

3 rows × 241 columns

```
In [50]: df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

```
Out[50]:
```

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

5 rows × 245 columns

```
In [51]: df12 = df11.drop('location',axis='columns')
df12.head(2)
```

```
Out[51]:
```

	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi Layout	5th Block Hbr	5th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenahalli	Yelahanka
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

BUILD A MODEL NOW...

```
In [52]: df12.shape
```

```
Out[52]: (7239, 244)
```

```
In [53]: X = df12.drop(['price'],axis='columns')  
X.head(3)
```

```
Out[53]:
```

	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	Yelache
0	2850.0	4.0	4	1	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
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0	0

3 rows x 243 columns

```
In [54]: X.shape
```

```
Out[54]: (7239, 243)
```

```
In [55]: y = df12.price  
y.head(3)
```

```
Out[55]:
```

0	428.0
1	194.0
2	235.0

Name: price, dtype: float64

```
In [56]: len(y)
```

```
Out[56]: 7239
```

```
In [57]: 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 [58]: from sklearn.linear_model import LinearRegression  
lr_clf = LinearRegression()  
lr_clf.fit(X_train,y_train)  
lr_clf.score(X_test,y_test)
```

```
Out[58]: 0.8629132245229449
```

USE K FOLD CROSS VALIDATION TO MEASURE ACCURACY OF OUR LINEARREGRESSION MODEL

In [59]:

```
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[59]: 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 few other algorithms for regression to see if we can get even better score. We will use GridSearchCV for this purpose

FIND BEST MODEL USING GRIDSEARCHCV

In [60]:

```
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[60]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726738	{'alpha': 2, 'selection': 'cyclic'}
2	decision_tree	0.716064	{'criterion': 'friedman_mse', 'splitter': 'best'}

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

TEST THE MODEL FOR FEW PROPERTIES

In [61]:

```
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 [62]:

```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

Out[62]:

```
83.86570258311222
```

In [63]:

```
predict_price('1st Phase JP Nagar',1000, 3, 3)
```

Out[63]:

```
86.08062284985995
```

In [64]:

```
predict_price('Indira Nagar',1000, 2, 2)
```

Out[64]:

```
193.31197733179556
```

In [65]:

```
predict_price('Indira Nagar',1000, 3, 3)
```

Out[65]:

```
195.52689759854331
```

EXPORT THE TESTED MODEL TO A PICKLE FILE

In [66]:

```
import pickle
with open('banglore_home_prices_model.pkl', '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 [67]:

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

SETTING OF FLASK SERVER

```
1  from flask import Flask, request, jsonify
2  import util
3
4  app = Flask(__name__)
5
6  @app.route('/get_location_names', methods=['GET'])
7  def get_location_names():
8      response = jsonify({
9          'locations': util.get_location_names()
10     })
11    response.headers.add('Access-Control-Allow-Origin', '*')
12
13    return response
14
15 @app.route('/predict_home_price', methods=['GET', 'POST'])
16 def predict_home_price():
17     total_sqft = float(request.form['total_sqft'])
18     location = request.form['location']
19     bhk = int(request.form['bhk'])
20     bath = int(request.form['bath'])
21
22     response = jsonify({
23         'estimated_price': util.get_estimated_price(location,total_sqft,bhk,bath)
24     })
25     response.headers.add('Access-Control-Allow-Origin', '*')
26
27     return response
28
29 if __name__ == "__main__":
30     print("Starting Python Flask Server For Home Price Prediction...")
31     util.load_saved_artifacts()
32     app.run()
```

FLASK DATA FUNCTIONS

```
1 import pickle
2 import json
3 import numpy as np
4
5 __locations = None
6 __data_columns = None
7 __model = None
8
9 def get_estimated_price(location,sqft,bhk,bath):
10     try:
11         loc_index = __data_columns.index(location.lower())
12     except:
13         loc_index = -1
14
15     x = np.zeros(len(__data_columns))
16     x[0] = sqft
17     x[1] = bath
18     x[2] = bhk
19     if loc_index>=0:
20         x[loc_index] = 1
21
22     return round(__model.predict([x])[0],2)
23
24
25 def load_saved_artifacts():
26     print("loading saved artifacts...start")
27     global __data_columns
28     global __locations
29
30     with open("./artifacts/columns.json", "r") as f:
31         __data_columns = json.load(f)['data_columns']
32         __locations = __data_columns[3:] # first 3 columns are sqft, bath, bhk
33
34     global __model
35     if __model is None:
36         with open('./artifacts/banglore_home_prices_model.pickle', 'rb') as f:
37             __model = pickle.load(f)
38     print("loading saved artifacts...done")
39
40 def get_location_names():
41     return __locations
42
43 def get_data_columns():
44     return __data_columns
45
46 if __name__ == '__main__':
47     load_saved_artifacts()
48     print(get_location_names())
49     print(get_estimated_price('1st Phase JP Nagar',1000, 3, 3))
50     print(get_estimated_price('1st Phase JP Nagar', 1000, 2, 2))
51     print(get_estimated_price('Kalhalli', 1000, 2, 2)) # other location
52     print(get_estimated_price('Ejipura', 1000, 2, 2)) # other location
```

APP.HTML

APP.CSS

APP.JS

```
<!DOCTYPE html>

<html>
<head>
    <title>Banglore Home Price Prediction</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="app.js"></script>
    <link rel="stylesheet" href="app.css">
</head>
<body>
<div class="img"></div>
<form class="form">
    <h2>Area (Square Feet)</h2>
    <input class="area" type="text" id="uiSqft" class="floatLabel" name="Squareft" value="1000">
    <h2>BHK</h2>
    <div class="switch-field">
        <input type="radio" id="radio-bhk-1" name="uiBHK" value="1"/>
        <label for="radio-bhk-1">1</label>
        <input type="radio" id="radio-bhk-2" name="uiBHK" value="2" checked/>
        <label for="radio-bhk-2">2</label>
        <input type="radio" id="radio-bhk-3" name="uiBHK" value="3"/>
        <label for="radio-bhk-3">3</label>
        <input type="radio" id="radio-bhk-4" name="uiBHK" value="4"/>
        <label for="radio-bhk-4">4</label>
        <input type="radio" id="radio-bhk-5" name="uiBHK" value="5"/>
        <label for="radio-bhk-5">5</label>
    </div>
    </form>
<form class="form">
    <h2>Bath</h2>
    <div class="switch-field">
        <input type="radio" id="radio-bath-1" name="uiBathrooms" value="1"/>
        <label for="radio-bath-1">1</label>
        <input type="radio" id="radio-bath-2" name="uiBathrooms" value="2" checked/>
```

```
@import url("https://fonts.googleapis.com/css?family=Roboto:300");

.switch-field {
    display: flex;
    margin-bottom: 36px;
    overflow: hidden;
}

.switch-field input {
    position: absolute !important;
    clip: rect(0, 0, 0, 0);
    height: 1px;
    width: 1px;
    border: 0;
    overflow: hidden;
}

.switch-field label {
    background-color: #e4e4e4;
    color: rgba(0, 0, 0, 0.6);
    font-size: 14px;
    line-height: 1;
    text-align: center;
    padding: 8px 16px;
    margin-right: -1px;
    border: 1px solid rgba(0, 0, 0, 0.2);
    box-shadow: inset 0 1px 3px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255, 0.1);
    transition: all 0.1s ease-in-out;
}

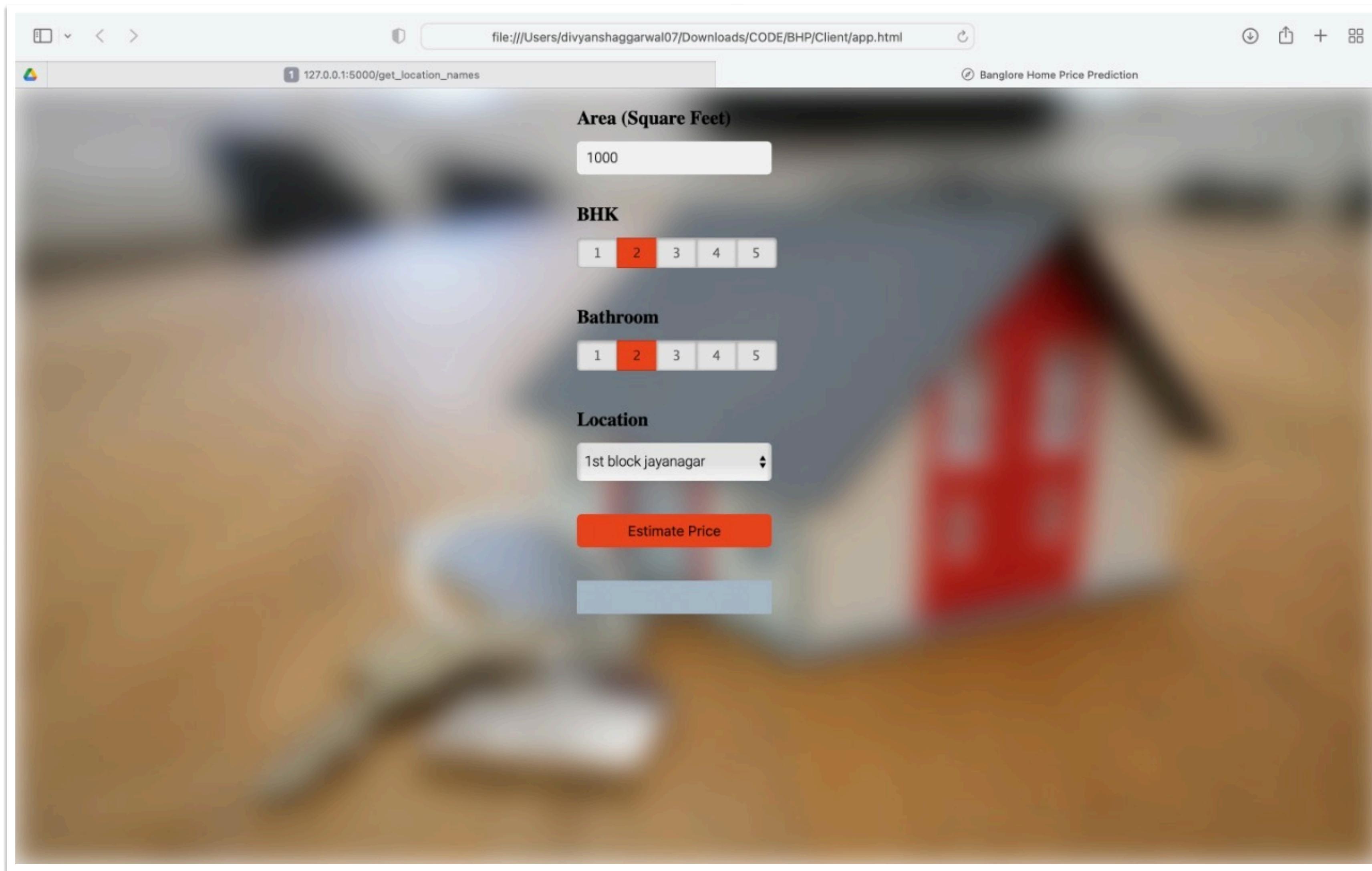
.switch-field label:hover {
    cursor: pointer;
}
```

```
function getBathValue() {
    var uiBathrooms = document.getElementsByName("uiBathrooms");
    for(var i in uiBathrooms) {
        if(uiBathrooms[i].checked) {
            return parseInt(i)+1;
        }
    }
    return -1; // Invalid Value
}

function getBHKValue() {
    var uiBHK = document.getElementsByName("uiBHK");
    for(var i in uiBHK) {
        if(uiBHK[i].checked) {
            return parseInt(i)+1;
        }
    }
    return -1; // Invalid Value
}

function onClickedEstimatePrice() {
    console.log("Estimate price button clicked");
    var sqft = document.getElementById("uiSqft");
    var bhk = getBHKValue();
    var bathrooms = getBathValue();
    var location = document.getElementById("uiLocations");
    var estPrice = document.getElementById("uiEstimatedPrice");
    // var url = "http://127.0.0.1:5000/predict_home_price"; //Use this if you are NOT using nginx which is first 7 tutorials
    var url = "/api/predict_home_price"; // Use this if you are using nginx. i.e tutorial 8 and onwards
    $.post(url, {
        total_sqft: parseFloat(sqft.value),
```

WEB APPLICATION



CONCLUSION AND FUTURE SCOPE:

**TILL NOW WE HAVE TAKEN THE DATA ON BANGLORE ALONE.
IN FUTURE WE HOPE TO TAKE THE DATA OF OTHER STATES
ALSO**

**OUR MODEL HAS AN ACCURACY OF 86 % TILL TODAY.
WE HOPE TO INCREASE THE ACCURACY IN NEAR FUTURE.**

THANK YOU

