Bangalore House Price Predictions

What is Bangalore House Price Predictions dataset?

The dataset contains description of houses in the city of Bangalore.

Objective

To predict the price of house in the city of Bangalore, through analysing & working on available dataset.

In [1]:
```
#importing basic python libraries.
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
matplotlib.rcParams["figure.figsize"] = (20,10)
```

In [2]:
```
df1 = pd.read_csv (r'D:\Neel_Folder\Data Science\Bangalore Housing price Dataset\Bengaluru_House_Data.csv')
```

Observing the Dataset

In [3]:
```
# First explore the head of the dataset to get an idea of the dataset.
df1.head()
```

Out[3]:

	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 [4]:
```
df1.shape
# the dataset have 13320 rows and 9 columns.
```

Out[4]:
```
(13320, 9)
```

In [5]:
```
df1.columns
```

Out[5]:
```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

Observation

In the above code we can see there are 9 columns in dataset

1. area_type : Describes the type of the property
2. availability : Current status of the house available for moving in.
3. location : Locality of the house
4. size : Number of rooms
5. society : Name of the housing society
6. total_sqft : measure of the house in square feets
7. bath : Number of bathrooms
8. balcony : Number of balconies
9. price : price of the house

Price is dependant variable and other all are independent variable. We have to predict price.

Cleaning the Dataset

In [6]:
```
df1['area_type'].unique()
```

Out[6]:
```
array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
       'Carpet  Area'], dtype=object)
```

In [10]:
```
#df1['area_type'].value_counts()
```

To keep model very simple, drop the feature that are not require to build our model.

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

In [8]:
```
df2.head()
```

Out[8]:

	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

Find null (missing) values

In [9]:
```
#handle missing values
df2.isnull().sum()
```

Out[9]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

Observation :

As null values are very less in numbers, we can drop it.

In [41]:
```
#if we want to see the dataset with null values
#Null_size = pd.isnull(df2['size'])
#df2[Null_size]
```

In [10]:
```
# as null values are very less in numbers, we can drop it
df3 = df2.dropna()
```

In [11]:
```
df3.isnull().sum()
```

Out[11]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [12]:
```
df3.shape
```

Out[12]:
```
(13246, 5)
```

Feature Engineering

Feature engineering is the process of using domain knowledge to extract features from the raw data.

In [13]:
```
# Feature Engineering
df3['size'].unique()
```

Out[13]:
```
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)
```

When we explore the 'size' column we observe there is no uniformity in words. To solve this we will form a new column 'bhk', it will have only numeric values. Lambda function will be used here.

In [14]:
```
df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
df3.bhk.unique()
```

```
C:\Users\admin\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWi
thCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

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

In [15]:
```
df3.head()
```

Out[15]:

	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

In [16]:
```
df3['bhk'].unique()
# some house have 43 and 27 bedrooms
```

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

Observation

- When we look into unique values we see there is some house with 43 & 27 bedrooms.
- To tackle this we will first look into the 'total_sqft' column. The column shows that total_sqft is in 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. We are going to just drop such corner cases to keep things simple

In [17]:
```
df3[df3.bhk>20]
```

Out[17]:

	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

In [18]:
```
df3.total_sqft.unique()
#take average value for the total_sqft in range form
```

Out[18]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [19]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [20]:
```
df3[~df3['total_sqft'].apply(is_float)].head(10)
```

Out[20]:

	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

We create a new dataframe (df4) which is a copy of df3. The function 'convert_sqft_to_num' will be applied to df4.

In [21]:
```
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 [22]:
```
convert_sqft_to_num('2100 - 2850')
```

Out[22]:
```
2475.0
```

In [23]:
```
df4 = df3.copy()
df4.total_sqft = df4.total_sqft.apply(convert_sqft_to_num)
df4 = df4[df4.total_sqft.notnull()]
df4.head(2)
```

Out[23]:

	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

In [24]:
```
df4.loc[30]
```

Out[24]:
```
location      Yelahanka
size              4 BHK
total_sqft         2475
bath                  4
price               186
bhk                   4
Name: 30, dtype: object
```

In real estate market price per sq.ft is very important. A new column is created 'price per sq.ft' (derived by dividing price by total_sqft). This new column will help us in outlier cleaning.

In [25]:
```
#a new column is created 'price per sq.ft' (derived by dividing price by total
_sqft)
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
```

Out[25]:

	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 [26]:
```
df5_stats = df5['price_per_sqft'].describe()
df5_stats
```

Out[26]:
```
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
```

Dimensionality Reduction

In [27]:
```
len(df5.location.unique())
```

Out[27]:
```
1298
```

Method to Dimensionality Reduction

- To handle text data we convert it into dummy columns using one hot encoding, if we keep all the location we will have around 1300 columns which is too many features. This is called Dimensionality Curse.
- There will be many location which will have less than 10 data points. We searched unique location name and those whose name was appeared less than 10 we categorized them in 'Others'.This way number of category can be reduced by huge amount.Later when we do one hot encoding, it will help us with having fewer dummy columns.

In [28]:
```
df5.location = df5.location.apply(lambda x: x.strip())
location_stats = df5['location'].value_counts(ascending=False)
location_stats
```

Out[28]:
```
Whitefield                                    533
Sarjapur  Road                                392
Electronic City                               304
Kanakpura Road                                264
Thanisandra                                   235
                                             ...
Banashankari 6th Stage ,Subramanyapura          1
2nd phase jp nagar, jp nagar                    1
Banashankari3rd stage bigbazar                  1
Gubbi Cross, Hennur Main Road                   1
Kanakapura  Rod                                 1
Name: location, Length: 1287, dtype: int64
```

In [29]:
```
location_stats.values.sum()
```

Out[29]:
```
13200
```

In [30]:
```
len(location_stats[location_stats>10])
```

Out[30]:
```
240
```

In [31]:
```
len(location_stats)
```

Out[31]:
```
1287
```

In [32]:
```
len(location_stats[location_stats<=10])
```

Out[32]:
```
1047
```

In [33]:
```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[33]:
```
1st Block Koramangala                      10
Ganga Nagar                                10
Kalkere                                    10
Basapura                                   10
Naganathapura                              10
                                           ..
Banashankari 6th Stage ,Subramanyapura      1
2nd phase jp nagar, jp nagar                1
Banashankari3rd stage bigbazar              1
Gubbi Cross, Hennur Main Road               1
Kanakapura  Rod                             1
Name: location, Length: 1047, dtype: int64
```

In [34]:
```
len(df5.location.unique())
```

Out[34]:
```
1287
```

In [35]:
```
df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_les
s_than_10 else x)
len(df5.location.unique())
```

Out[35]: 241

In [36]:
```
df5.head(10)
```

Out[36]:

	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

Outlier Removal

Outlier Removal Using Business Logic

According to real estate expert a room is of 300sq ft. approx. Now we will divide the ‘total_sqft’ with ‘bhk’ and see which value is less than 300. After obtaining the values we will remove them from the data.

In [37]:
```
df5[df5.total_sqft/df5.bhk<300].head()
```

Out[37]:

	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

In [38]:
```
df5.shape
```

Out[38]: (13200, 7)

In [39]:
```
df6 = df5[~(df5.total_sqft/df5.bhk<300)] # '~' means filter
df6.shape
```

Out[39]:
```
(12456, 7)
```

Outlier Removal Using Standard Deviation and Mean

In [40]:
```
df6.price_per_sqft.describe()
```

Out[40]:
```
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
```

- When we use ‘describe’ function on price per sq.ft. we see the minimum price per sq.ft. is Rs.267. which is impossible in Bangalore. Same with maximum value.
- We will remove this by using mean and standard deviation. We want our dataset to have normal distribution, in which 68% of data is between Mean and 1 std deviation. We will filter everything beyond 1 standard deviation per location.

In [41]:
```
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_sq
ft<=(m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out
df7 = remove_pps_outliers(df6)
df7.shape
```

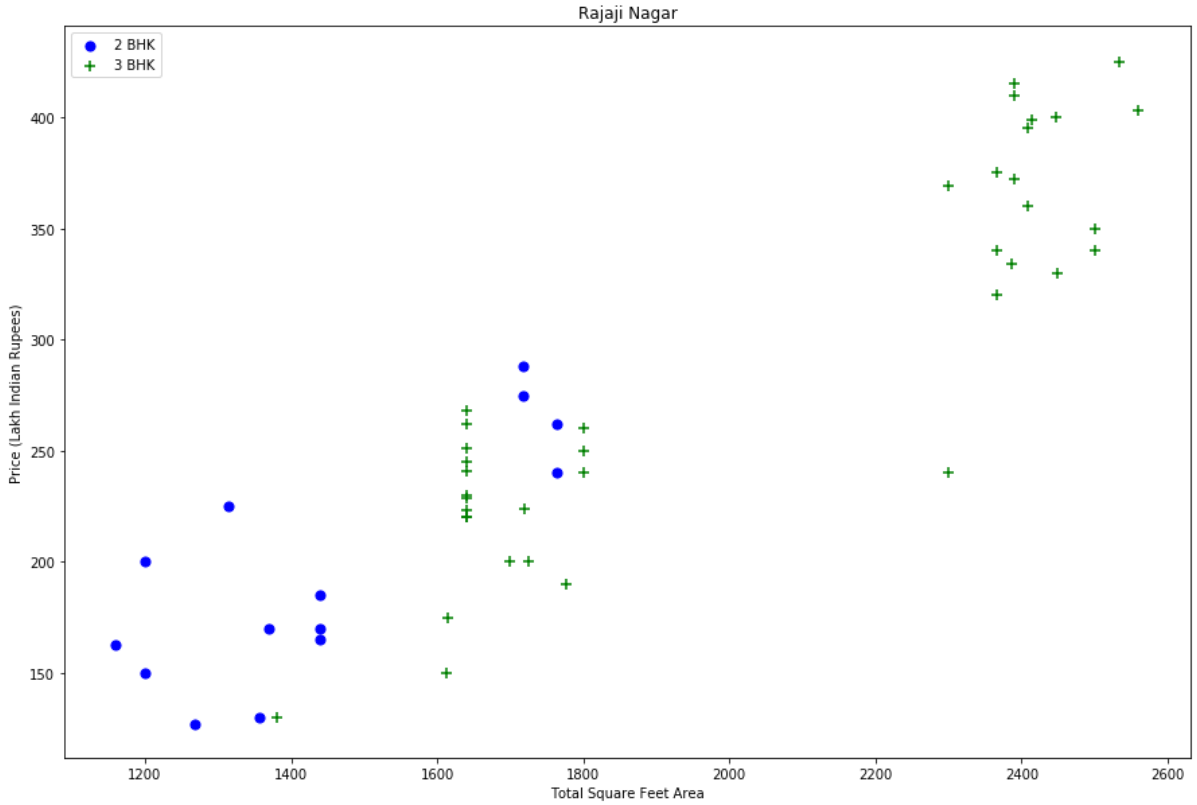
Out[41]:
```
(10242, 7)
```

Outlier removing using scatter plot

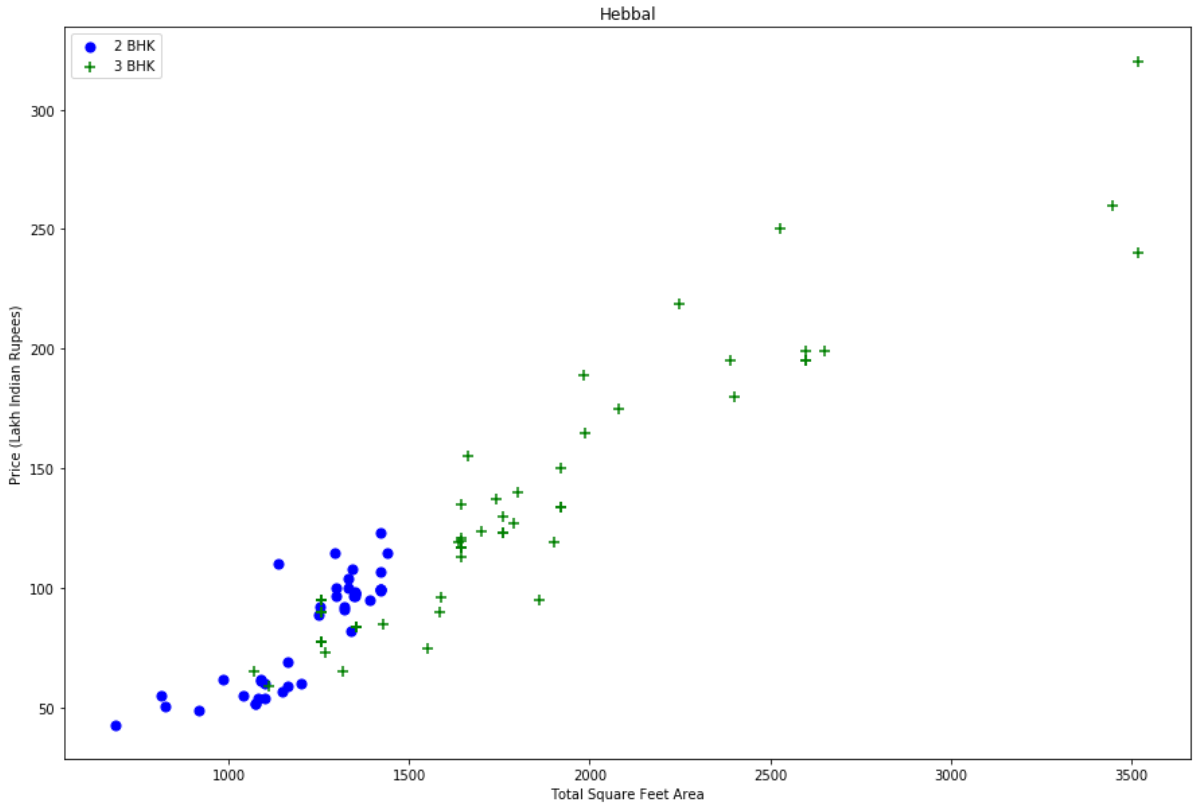
In [48]:

```
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 [49]:
```
plot_scatter_chart(df7,"Hebbal")
```



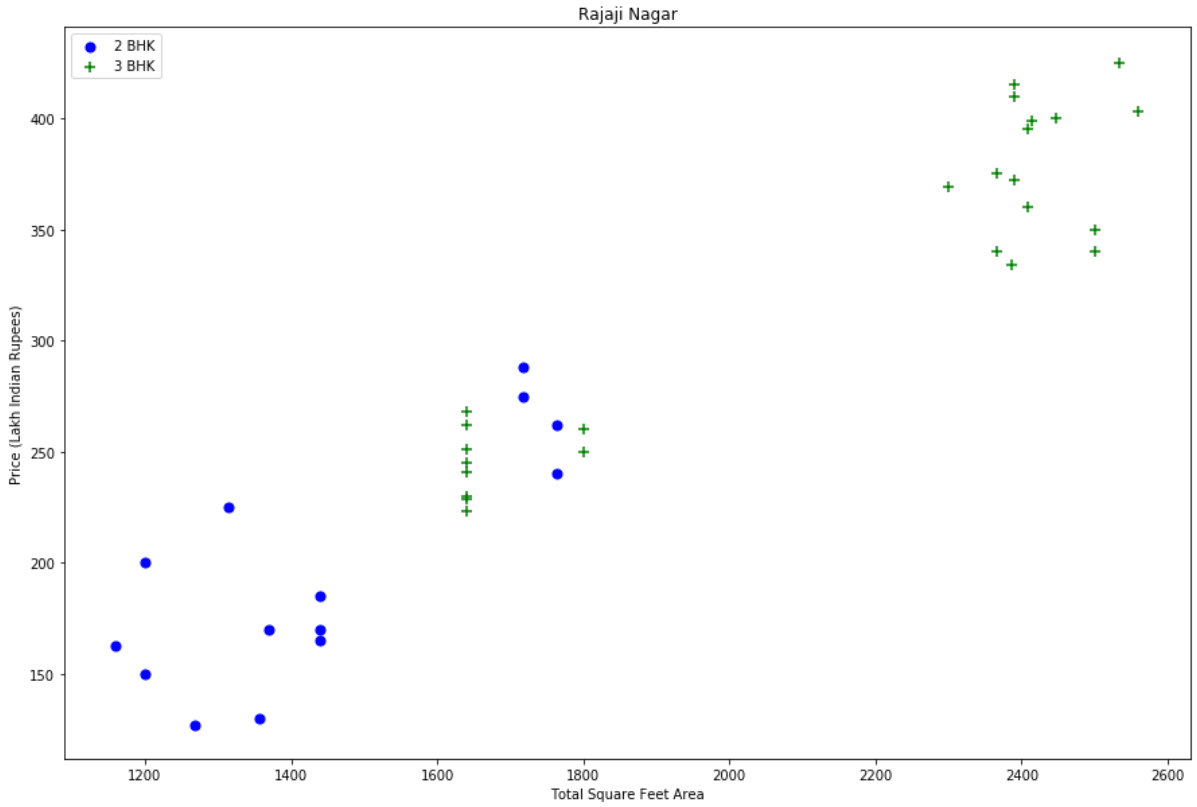
- When we compare price of 2bhk and 3bhk house from Rajaji nagar the price differ.For this we used scatter plot.
- While studying scatter plot in some cases the price of 2bhk & 3bhk is similar, for this We should remove properties where for same location the price of (for example) 3 bedroom apartment 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. By using dictionary we will remove 2 BHK apartments whose price_per_sqft is less than mean price_per_sqft of 1 BHK apartment (the dictionary works on all bhk-1).

In [50]:
```
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.pri
ce_per_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
```

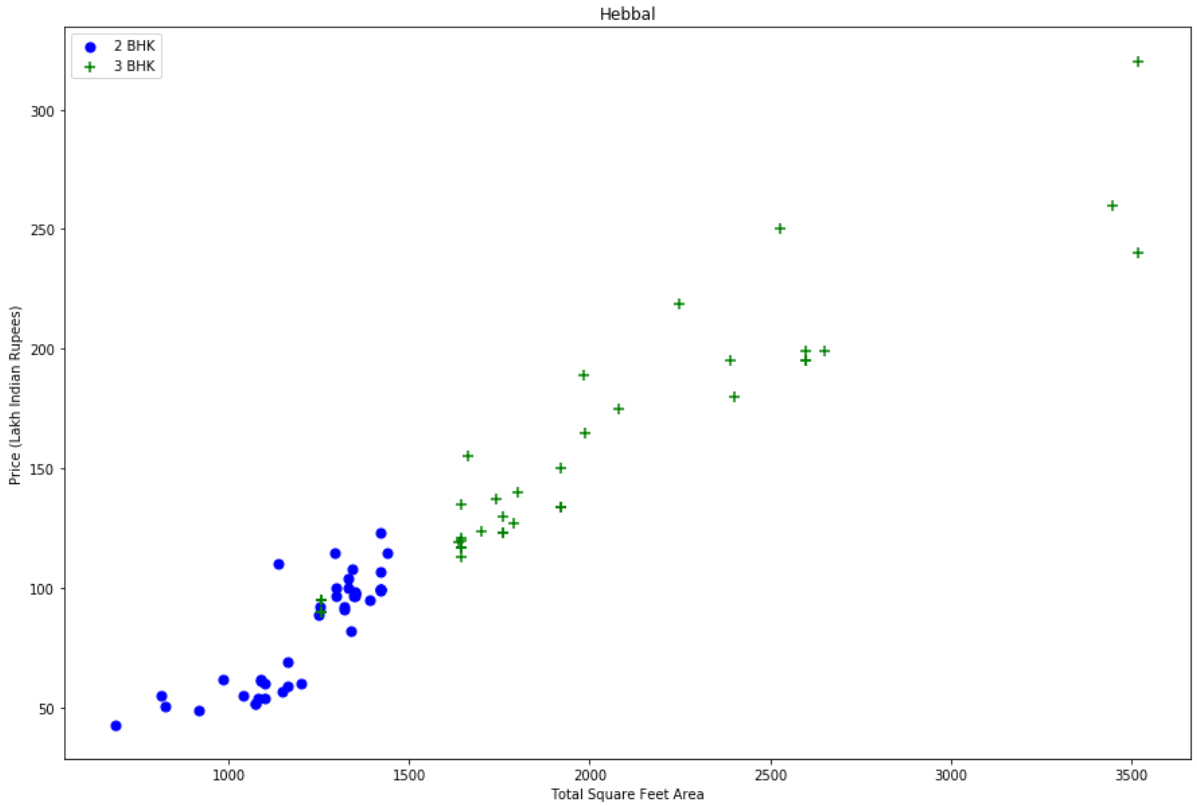
Out[50]: (7317, 7)

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

In [51]:
```
plot_scatter_chart(df8,"Rajaji Nagar")
```



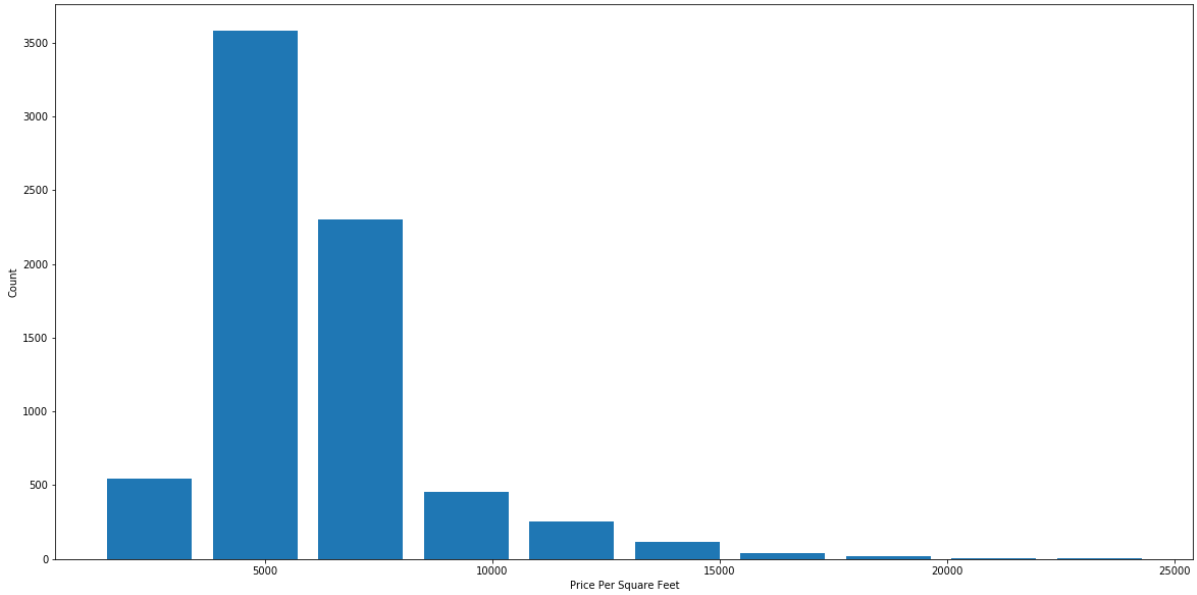
In [52]:
```
plot_scatter_chart(df8,"Hebbal")
```



When we compare before and after applying dictionary, we can see the outliers being removed.

In [54]:
```
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[54]:
```
Text(0, 0.5, 'Count')
```



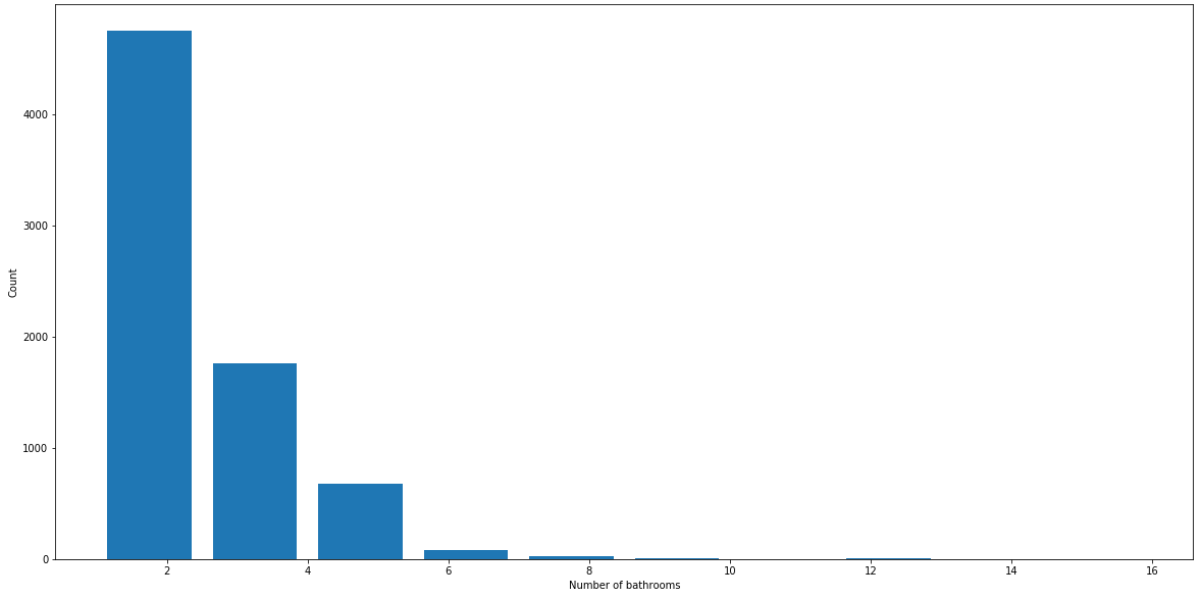
Outlier Removal Using Bathrooms Feature

In [55]:
```
df8.bath.unique()
```

Out[55]:
```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

In [56]:
```
#Lets just get an idea, number of bathroom
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[56]:
```
Text(0, 0.5, 'Count')
```



In [63]:
```
df8[df8.bath>10]
```

Out[63]:

	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

- Normally the number of bathroom is equal to number of bedrooms. We will remove data which has +2 bathrooms than bedroom. It is unusual to have 2 more bathrooms than number of bedrooms in a home.

In [57]:
```
df8[df8.bath>df8.bhk+2]
```

Out[57]:

	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

In [58]:
```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

Out[58]:
```
(7239, 7)
```

In [59]:
```
df9.head(2)
```

Out[59]:

	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

Now we will remove some extra features. Now we don't want size as we have bhk, we don't want price_per_sqft as it was used for outlier detection.

In [60]:
```
df10 = df9.drop(['size','price_per_sqft'],axis='columns')
df10.head(3)
```

Out[60]:

	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

Machine learning models cannot process the text data. So we have to convert it into numeric column. One of the method to convert categorical column to numeric is one hot encoding (using pandas dummies method).

In [64]:
```
dummies = pd.get_dummies(df10.location)
dummies.head(3)
```

Out[64]:

	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	...	Vi
0	1	0	0	0	0	0	0	0	0	0	...	
1	1	0	0	0	0	0	0	0	0	0	...	
2	1	0	0	0	0	0	0	0	0	0	...	

3 rows × 241 columns

In [66]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[66]:

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

5 rows × 245 columns

Delete Location Column as we have dummies

In [67]:
```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

Out[67]:

	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	...	Vija
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	

2 rows × 244 columns

Building a Model

In [69]:
```
df12.shape
```

Out[69]: (7239, 244)

We will create a X variable and it should contain only independent variable, so drop price from the dataframe. Create Y with price features.

In [70]:
```
X = df12.drop(['price'],axis='columns')
X.head(3)
```

Out[70]:

	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	...	Vij
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	

3 rows × 243 columns

In [71]:
```
X.shape
```

Out[71]: (7239, 243)

In [72]:
```
y = df12.price
y.head(3)
```

Out[72]:
```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

In [73]:
```
len(y)
```

Out[73]:
```
7239
```

Now use 'Train test split', test size will 20% and remaining 80% for model training.

In [74]:
```
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_s
tate=10)
```

Create a linear regression model, call fit method and once it is train we will check the scores.

Use K-fold cross validation to measure accuracy of our Linear Regression model.

In [75]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[75]:
```
0.8629132245229453
```

In [76]:
```
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[76]:
```
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.

In [78]:
```
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_tr
ain_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[78]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726739	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.736505	{'criterion': 'mse', 'splitter': 'random'}

Based on the results we can say that Linear Regression gives the best score. Hence we will use that.

Lets predict price by using linear regression.

In [79]:
```
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]
```

The values are entered in the following sequence: location, sqft, bath & bhk.

In [80]:
```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

Out[80]: 83.86570258310776

In [81]:
```
predict_price('1st Phase JP Nagar',1000, 3, 3)
```

Out[81]: 86.0806228498554

In [82]:
```
predict_price('Indira Nagar',1000, 2, 2)
```

Out[82]: 193.31197733179425

In [83]:
```
predict_price('Indira Nagar',1000, 3, 3)
```

Out[83]: 195.5268975985419

In [84]:
```
predict_price('1st Phase JP Nagar',500, 1, 1)
```

Out[84]: 41.59183071478516

In [89]:
```
predict_price('Electronic City Phase II',1056,2,2)
```

Out[89]: 35.44588171398833

Conclusion

By using various data science methods we prepared a dataset on which the linear regression model was used to predict the house rate.

(Source : 'Codebasic' youtube channel)