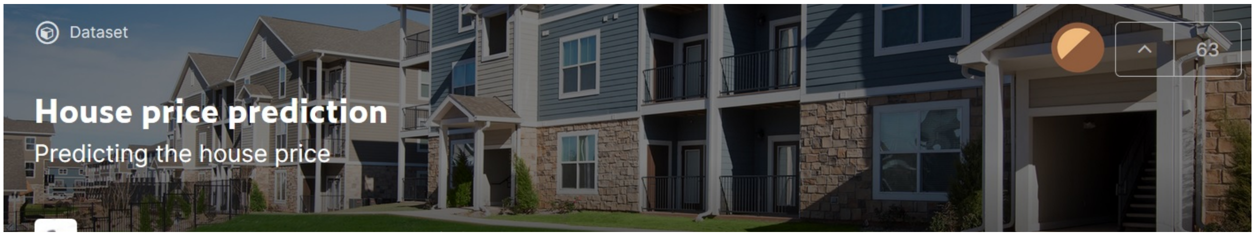


House Price Prediction

VALUATION OF HOUSE PRICES USING PREDICTION TECHNIQUE FOR GLOBAL CHALLENGE FOR AI AND DATA SCIENCE

Youssouf Sougueh
Data Scientist and Machine Learning | 2020

INTRODUCTION

INTRODUCTION

RELATED WORK

METHODLOGY

RESULTAT

CONCLUSION

I. INTRODUCTION

In this project, we present a web application that can generate predictions for future housing prices in Ottawa.

With the improvement of people's living standards, the demand for houses increases.

The average price for all residential homes, which includes houses and condos was $494,929, an increase of 17.64% over 2019.

Below are the average prices for Different types of Freehold & Condominium class properties last month

Housing Type	Average Price	% Change in Price
Bungalow	$575,774	28.1%
1 1/2 storey	$442,000	-8.7%
2 storey	$565,210	13.5%
3 storey	$670,581	25%
Split-Level	$584,552	5.44%
High-Ranch	$570,700	14.9%
Condo 2 Storey	$310,583	26.6%
Condo Apartment	$365,567	18.21%
3 Storey Condo	$346,766	19.8%

Figure 1: Average SOLD Home Prices by Property Type

Below are the average prices for Different types of Freehold & Condominium class properties last month

In order to select a prediction method various regression methods we explored and compared.

By training the model and generating predictions on the server-side of the application, we are able to offload computationally intensive tasks and focus on generating visualisations on the client-side. Our results demonstrate that our approach to the problem has been largely successful, and is able to produce predictions that are competitive to other housing price prediction models.

II. RELATIED WORK

House is usually as a heterogeneous goad, defined by a bundle of utility bearing features. Therefore, the house price can be considered as a quantitative representation of a set of these features. Over the past decades, a large amount of studies has examined the relationship between house price and house features.

	Data Examples	References														Ours
		[5]	[6]	[7]	[8]	[11]	[12]	[13]	[15]	[27]	[28]	[29]	[30]	[31]	[32]	
Scale of data	100	✔	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✔	✘	✘
	1, 000	✘	✘	✔	✔	✘	✘	✘	✔	✘	✔	✘	✔	✘	✔	✘
	10, 000	✘	✔	✘	✘	✔	✔	✔	✘	✔	✘	✘	✘	✘	✘	✘
	100, 000	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔
House profile	floor area, number of bedrooms	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
	geo-information, address, suburb	✘	✘	✔	✔	✔	✔	✔	✘	✘	✔	✔	✘	✔	✔	✔
	air condition, water, heating views	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘	✔	✘	✔	✔
Education profile	nearby schools	✘	✘	✔	✘	✔	✘	✘	✘	✘	✘	✘	✘	✔	✘	✔
	school districts	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔
	school rankings	✘	✘	✘	✘	✔	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔
Transportation profile	nearby public trans-port	✘	✘	✔	✘	✘	✘	✘	✘	✘	✘	✔	✘	✔	✔	✔
	travel time to work	✘	✔	✘	✘	✘	✘	✘	✘	✘	✘	✔	✔	✔	✘	✔
Facility profile	hospitals, shops	✘	✘	✔	✘	✘	✘	✘	✘	✔	✔	✔	✔	✔	✘	✔
	distance to nearest hospitals	✘	✘	✘	✘	✘	✘	✘	✘	✔	✘	✔	✔	✔	✘	✔

Figure 2: Summary of our data profiles

Independent Variables	Dependent Variable
Location (Geo-information Address) Size (Number of bedrooms) Total_Sqft (The land size of the house) Bath (Nb fo bathrooms_ Price per sqft	Price of property

Figure 3: Variables to be used in our prediction model

Category	Name of features	Descriptions	Min	Max
House	#BEDROOM	The number of bedrooms	1	10
	#BATHROOM	The number of bathrooms	1	10
	LANDSIZE	The land size of the house	173	52 272
SALES_DATE		The sales time of the houses (Updated 2 years ago)		
HOUSE_PRICE		The sales price of the houses	600	3 600

Figure 4: List of selected house features and statistics of our house data

III. METHODOLOGY

Methodology represents a description about the framework that is undertaken. It consists of various milestones that need to be achieved in order to fulfill the objective. We have undertaken different data mining and machine learning concepts. The following diagram (figure) represents step-wise tasks that need to be completed:

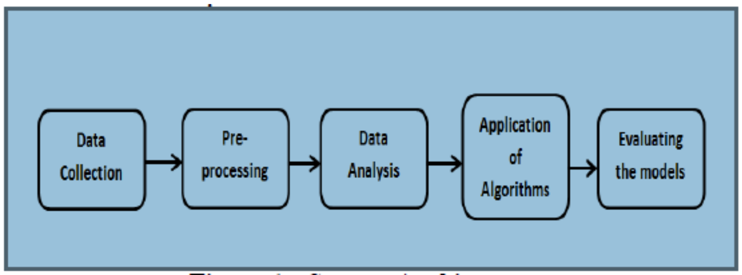


Figure 5: System Architecture

1) Data Collection

The dataset used in this project was an open source dataset from Kaggle Inc (https://www.kaggle.com/). It consists of 13 320 records with 9 parameters that have the possibility of affection the property prices.

However out of these 9 parameters only 5 were chosen which are bound to affect the housing prices

Parameters	Description	Datatype
Location	(Geo-information – Location)	Text
Size	(Number of bedrooms)	Numerical
Total_sqft	Total square feet of basement area	Numerical
Bath	Bathrooms	Numerical
Balcony		Numerical
Price	Selling Price of the house	Numerical

Figure 6: The Parameters

2) Data Preprocessing

It is a process of transforming the raw, complex data into systematic understandable knowledge. It involves the process of finding out missing and redundant data in the dataset. Entire dataset is checked for NaN and whichever observation consists of NaN will be deleted. Thus, this brings uniformity in the dataset. However in our dataset, there was no missing values found meaning that every record was constituted its corresponding feature values.

Data Cleaning and Missing Data

3) Data Analysis

Before applying any model to our dataset, we need to find out characteristics of our dataset. Thus, we need to analyze our dataset and study the different parameters and relationship between these parameters. We can also find out the outliers present in our dataset. Outliers occur due to some kind of experimental errors and they need to be excluded from the dataset

4) Application of Algorithms

Once the data is clean and we have gained insights about the dataset, we can apply an appropriate machine learning model that fits our dataset. We have selected three algorithms to predict the dependent variable in our datasets. The algorithms that we have selected are basically used as classifiers but we are training them to predict the continuous values. The three algorithms are Linear Regression, Lassor Regression and Decision Tree. These algorithms were implemented with the help of python's SciKit-learn Library

a. Linear Regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

b. Lasso Regression

Lasso is a powerful regression technique. It works by penalizing the magnitude of coefficients of features along with minimizing the error between predicted and actual observations. Lasso is called as L1 Regularization technique. Lasso attempts to minimize the cost function. The cost function is given as Cost(W)= RSS(W) + ⍰ (Sum of squares of weight)

c. Decision Tree

Decision trees are considered to be the best and most widely used supervised learning algorithm. This model has the ability to predict the output with at most accuracy and stability. It is used to predict any kind of problems such as classification or regression. However, in our case we want to predict a continuous target value hence our problem is of regression type. In this model, the available dataset can be continuous or categorical.

IV. RESSULTAT

On comparing the various models, we find that Lenear Regression works the best with highest accuracy of 81.83%

	model	best_score	best_params
0	linear_regression	0.818354	{'normalize': False}
1	lasso	0.687429	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.725165	{'criterion': 'mse', 'splitter': 'best'}

Figure 7 : Accuracy Values

V. CONCLUSION

In this research paper, we have used machine learning algorithms to predict the house prices. We have mentioned the step by step procedure to analyze the dataset and finding the correlation between the parameters.

By improving the error values this research work can be useful for development of applications for various respective cities.

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)
```

In [2]:
```
df1 = pd.read_csv("Bengaluru_House_Data.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.groupby('area_type')['area_type'].agg("count")
```

Out[4]:
```
area_type
Built-up  Area          2418
Carpet  Area              87
Plot  Area              2025
Super built-up  Area    8790
Name: area_type, dtype: int64
```

Drop the columns

In [5]:
```
df2 = df1.drop(['area_type', 'society', 'balcony', 'availability'],axis=
'columns')
df2.shape
```

Out[5]: (13320, 5)

In [6]:
```
df2.head()
```

Out[6]:

	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

Data Cleaning (Real Estate Price Prediction Project)

Missing data

In [7]:
```
df2.isnull().sum()
```

Out[7]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

Drop missing data

In [8]:
```
df3 = df2.dropna()
df3.isnull().sum()
```

Out[8]:
```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [9]:
```
df3.shape
```

Out[9]:
```
(13246, 5)
```

See size of Bedroom

In [10]:
```
df3['size'].unique()
```

Out[10]:
```
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)
```

In [11]:
```
df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Sett
ingWithCopyWarning:
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-do
cs/stable/indexing.html#indexing-view-versus-copy
```

```
  """Entry point for launching an IPython kernel.
```

In [12]:
```
df3.head()
```

Out[12]:

	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 [23]:
```
df3['bhk'].unique()
```

Out[23]:
```
array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 1
2,
       13, 18], dtype=int64)
```

In [14]:
```
df3.total_sqft.unique()
```

Out[14]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [15]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [16]:
```
df3[~df3['total_sqft'].apply(is_float)].head(10)
```

Out[16]:

	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

In [43]:
```
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 [44]:
```
df4 = df3.copy()
df4['total_sqft'] = df4['total_sqft'].apply(convert_sqft_to_num)
df4.head()
```

Out[44]:

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

In [19]:
```
df4.loc[30]
```

Out[19]:
```
location      Yelahanka
size              4 BHK
total_sqft         2475
bath                  4
price               186
bhk                   4
Name: 30, dtype: object
```

In [20]:
```
(2100+2850)/2
```

Out[20]:
```
2475.0
```

Feature Engineering (Real Estate Price Prediction Project)

In [21]:
```
df5 = df4.copy()
df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
df5.head()
```

Out[21]:

	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 [42]:
```
df5.location.unique()
```

Out[42]:
```
array(['Electronic City Phase II', 'Chikka Tirupathi', 'Uttarahalli', ..
.,
       '12th cross srinivas nagar banshankari 3rd stage',
       'Havanur extension', 'Abshot Layout'], dtype=object)
```

Count location: high dimensionality problem : technique available to reduce the dimension

In [45]:
```
len(df5.location.unique())
```
Out[45]:
```
1304
```

In [47]:
```
df5.location = df5.location.apply(lambda x: x.strip())

location_stats = df5.groupby('location')['location'].agg('count')
location_stats.head()
```
Out[47]:
```
location
1 Annasandrapalya                                  1
1 Giri Nagar                                       1
1 Immadihalli                                      1
1 Ramamurthy Nagar                                 1
12th cross srinivas nagar banshankari 3rd stage    1
Name: location, dtype: int64
```

In [49]:
```
df5.location = df5.location.apply(lambda x: x.strip())

location_stats = df5.groupby('location')['location'].agg('count').sort_v
alues(ascending=False)
location_stats.head()
```
Out[49]:
```
location
Whitefield          535
Sarjapur  Road      392
Electronic City     304
Kanakpura Road      266
Thanisandra         236
Name: location, dtype: int64
```

In [50]:
```
len(location_stats[location_stats<=10])
```
Out[50]:
```
1052
```

In [52]:
```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10.head()
```
Out[52]:
```
location
BTM 1st Stage           10
Basapura                10
Sector 1 HSR Layout     10
Naganathapura           10
Kalkere                 10
Name: location, dtype: int64
```

In [53]:
```
len(df5.location.unique())
```
Out[53]:
```
1293
```

In [54]:
```
df5.location = df5.location.apply(lambda x: 'other' if x in location_sta
ts_less_than_10 else x)
len(df5.location.unique())
```
Out[54]:
```
242
```

In [55]:
```
df5.head(10)
```
Out[55]:

	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 : Data Error Detection and Correction

In [56]:
```
df5[df5.total_sqft/df5.bhk<300].head()
```
Out[56]:

	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 [57]:
```
df5.shape
```
Out[57]:
```
(13246, 7)
```

Removing outliers

In [58]:
```
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.shape
```
Out[58]:
```
(12502, 7)
```

In [59]:
```
df6.price_per_sqft.describe()
```
Out[59]:
```
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
```

To calculate the standard deviation

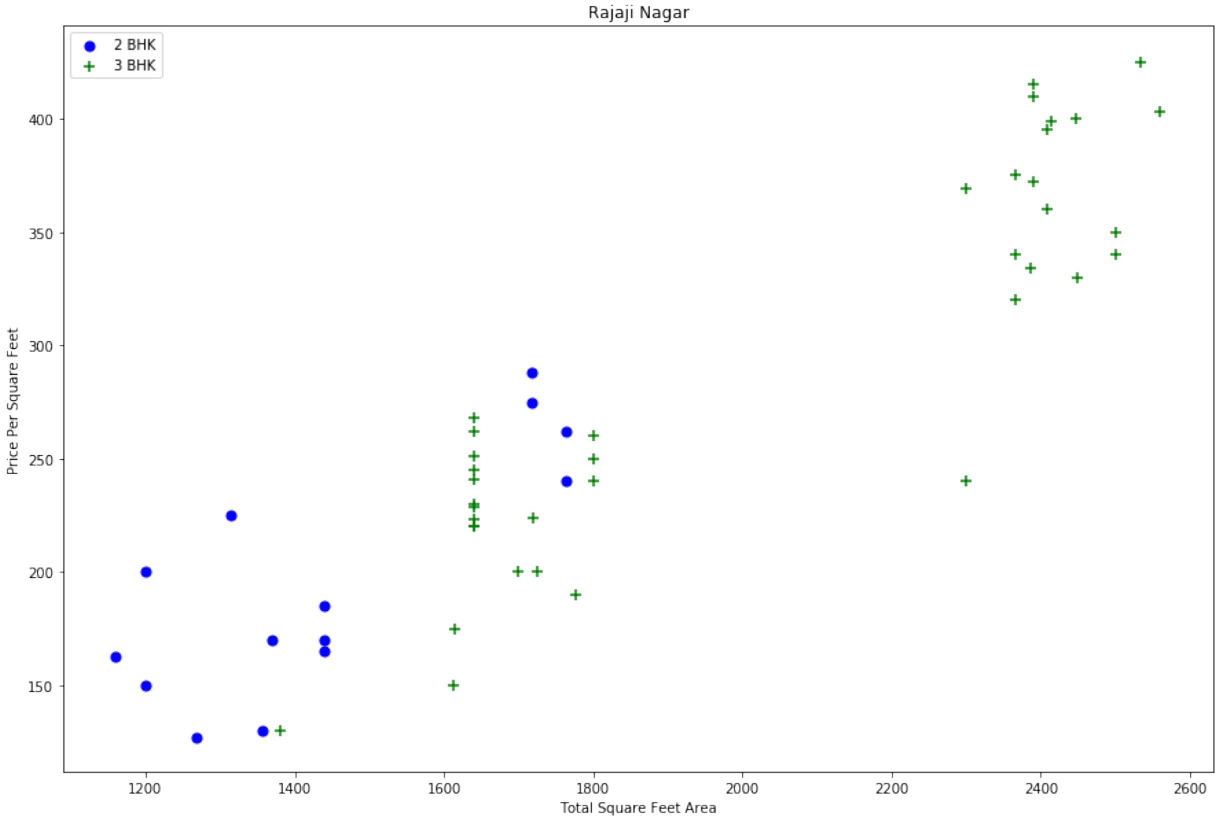
In [60]:
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_
per_sqft<=(m+st))]
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)
    return df_out

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

Out[60]:
```
(10241, 7)
```

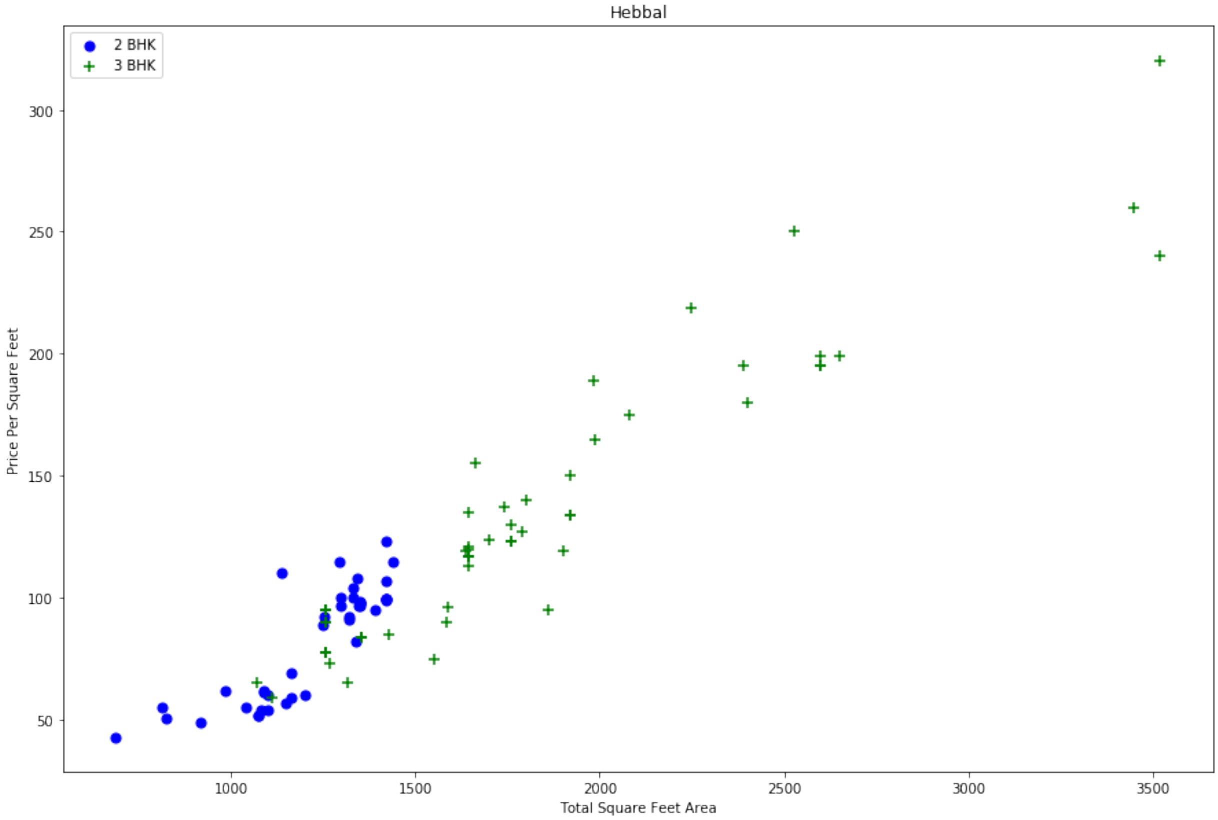
In [61]:
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s
=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+',color='green',labe
l='3 BHK', s=50)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price Per Square Feet")
    plt.title(location)
    plt.legend()

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



In [62]:
```
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',labe
l='3 BHK', s=50)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price Per Square Feet")
    plt.title(location)
    plt.legend()

plot_scatter_chart(df7,"Hebbal")
```



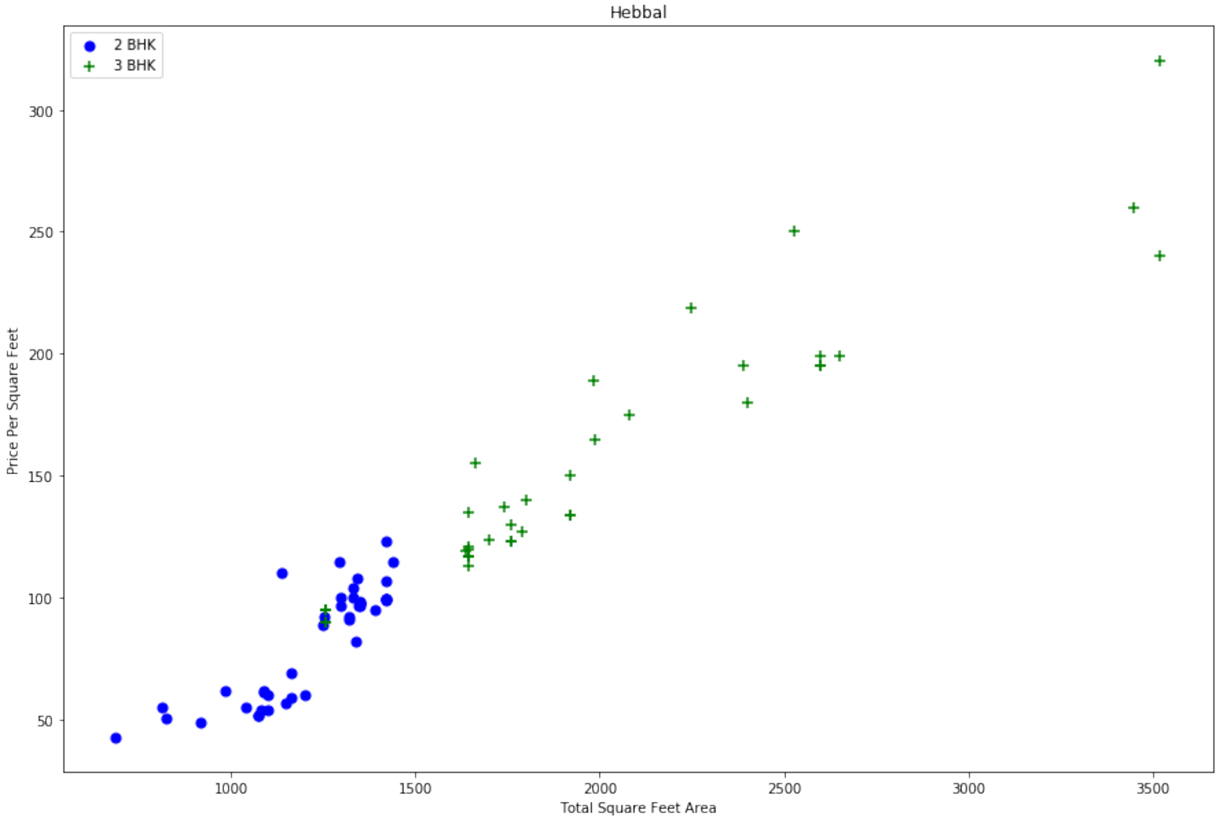
Now we can remove those 2 BHK aprtments whose price_per_sqft is less than mean price_price_sqft of 1 BHK apartment

In [63]:
```
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.shape
```

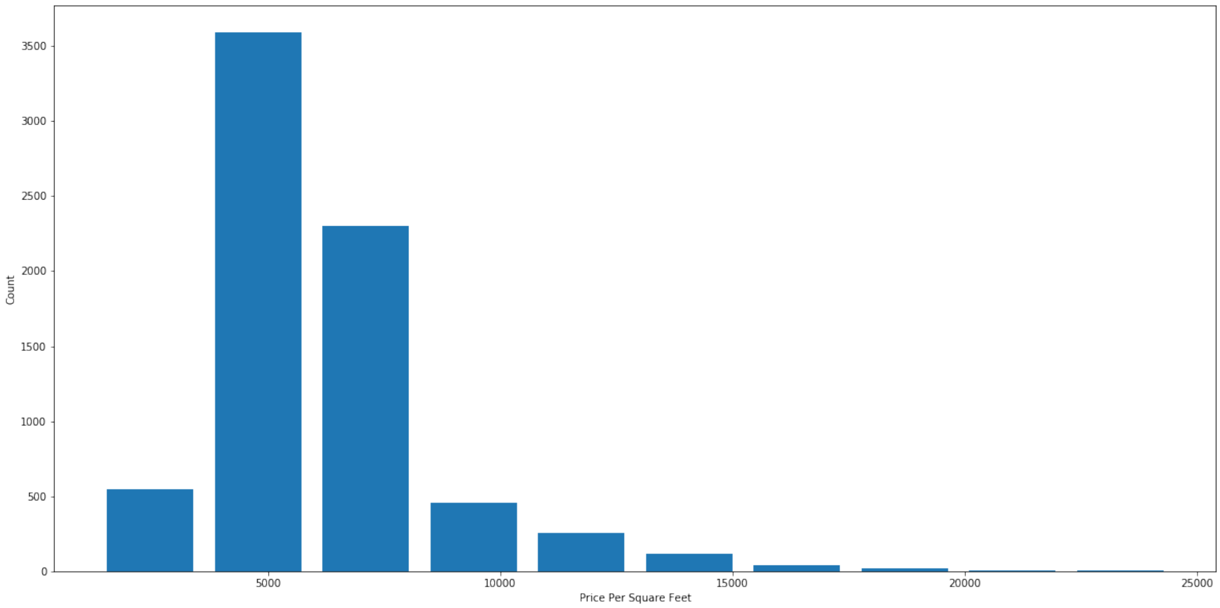
Out[63]: (7329, 7)

In [64]:
```
plot_scatter_chart(df8,"Hebbal")
```



In [65]:
```
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[65]: Text(0, 0.5, 'Count')



In [66]:
```
df8.bath.unique()
```

Out[66]: array([4., 3., 2., 5., 8., 1., 6., 7., 9., 12., 16., 13.])

In [67]:
```
df8[df8.bath>10]
```

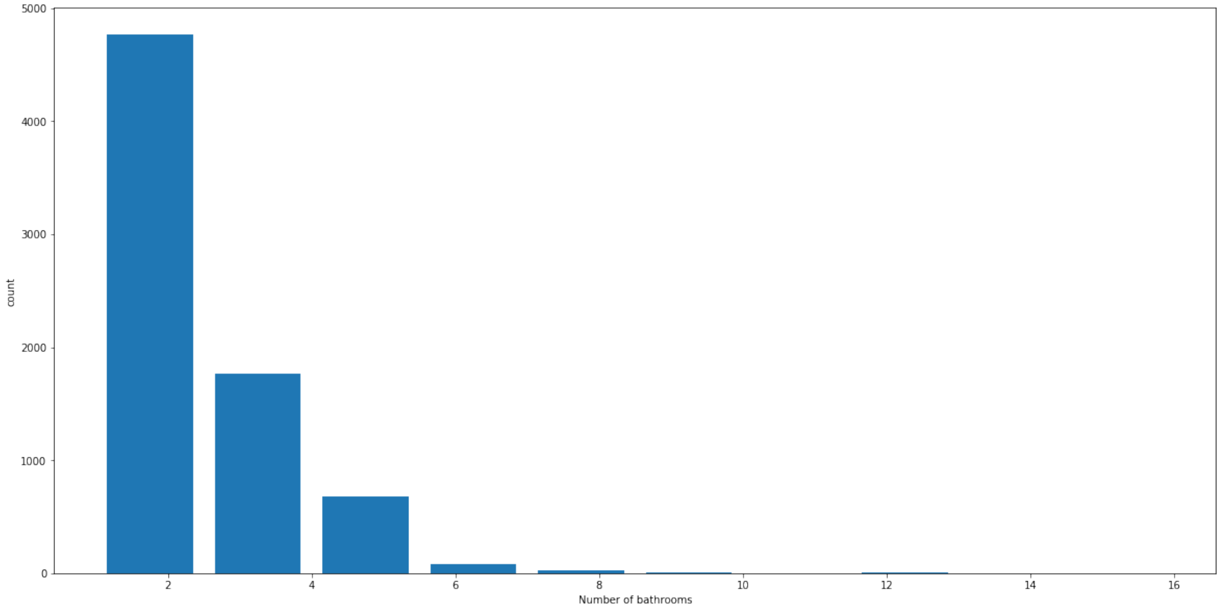
Out[67]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft

5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8486	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8575	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9308	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9639	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

In [68]:
```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("count")
```

Out[68]: `Text(0, 0.5, 'count')`



In [69]:
```
corr = df8.corr()
corr
```

Out[69]:

	total_sqft	bath	price	bhk	price_per_sqft
total_sqft	1.000000	0.712380	0.840862	0.675220	0.354010
bath	0.712380	1.000000	0.613515	0.881427	0.358671
price	0.840862	0.613515	1.000000	0.568565	0.710216
bhk	0.675220	0.881427	0.568565	1.000000	0.342726
price_per_sqft	0.354010	0.358671	0.710216	0.342726	1.000000

In [70]:
```
#import Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [71]:
```
sns.heatmap(df8.corr(), annot=True)
```

Out[71]: `<matplotlib.axes._subplots.AxesSubplot at 0x184704990b8>`

In [72]:
```
df8[df8.bath>df8.bhk+2]
```

Out[72]:

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

In [73]:
```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

Out[73]: (7251, 7)

column removal not necessary

In [74]:
```
df10 = df9.drop(['size', 'price_per_sqft'],axis='columns')
df10.head(5)
```

Out[74]:

	location	total_sqft	bath	price	bhk
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3
3	1st Block Jayanagar	1200.0	2.0	130.0	3
4	1st Block Jayanagar	1235.0	2.0	148.0	2

Model Building

Convert text to numeric

In [78]:
```
pd.get_dummies(df10.location).head()
```

Out[78]:

	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	...	Vish
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	1	0	0	0	0	0	0	0	0	0	...	
4	1	0	0	0	0	0	0	0	0	0	...	

5 rows × 242 columns

In [79]:
```
dummies = pd.get_dummies(df10.location)
dummies.head(5)
```

Out[79]:

	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	...	Vish
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	1	0	0	0	0	0	0	0	0	0	...	
4	1	0	0	0	0	0	0	0	0	0	...	

5 rows × 242 columns

In [80]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='colum
ns')
df11.head(5)
```

Out[80]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	...	Vi
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	1200.0	2.0	130.0	3	1	0	0	0	0	...	

	location	total_sqft	bath	price	bhk							
	Jayanagar											
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...

5 rows × 246 columns

In [81]:
```
df12 = df11.drop('location',axis='columns')
df12.head(5)
```

Out[81]:

	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	...	Vijaya
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	1875.0	2.0	235.0	3	1	0	0	0	0	0	...	
3	1200.0	2.0	130.0	3	1	0	0	0	0	0	...	
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	

5 rows × 245 columns

In [82]:
```
df12.shape
```

Out[82]:
```
(7251, 245)
```

In [83]:
```
X = df12.drop('price',axis='columns')
X.head()
```

Out[83]:

	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	...	Vijay
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	1200.0	2.0	3	1	0	0	0	0	0	0	...	
4	1235.0	2.0	2	1	0	0	0	0	0	0	...	

5 rows × 244 columns

In [84]:
```
y = df12.price
y.head()
```

Out[84]:
```
0    428.0
1    194.0
2    235.0
```

```
3    130.0
4    148.0
Name: price, dtype: float64
```

I use training data set for the model training To evaluate the model performance I use the test data set

In [85]:
```
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,ra
ndom_state=10)
```

In [86]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[86]:
```
0.845227769787428
```

Evaluate metric(s) by cross-validation

In [87]:
```
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[87]:
```
array([0.82430186, 0.77166234, 0.85089567, 0.80837764, 0.83653286])
```

In [88]:
```
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, retu
rn_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_param
s'])

find_best_model_using_gridsearchcv(X,y)
```

Out[88]:

	model	best_score	best_params
0	linear_regression	0.818354	{'normalize': False}
1	lasso	0.687479	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.726649	{'criterion': 'mse', 'splitter': 'best'}

I conclud the regression model is the best one

In [89]:

```
X.columns
```

Out[89]:

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

In [90]:

```
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 [97]:

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

Out[97]:

```
83.49904677167721
```

In [98]:

```
predict_price('1st Phase JP Nagar',1000, 3, 2)
```

Out[98]:

```
88.57807171618973
```

In [99]:
```
predict_price('Indira Nagar',1000, 2, 2)
```

Out[99]: 181.27815484007024

In [94]:
```
predict_price('Indira Nagar',1000, 3, 2)
```

Out[94]: 186.35717978458274

Export the model to pickle file: Save Model Using Pickle

In [100]:
```
import pickle
with open('banglore_home_prices_model.pickle','wb') as f:
    pickle.dump(lr_clf,f)
```

In [101]:
```
import json
columns = {
    'data_columns': [col.lower() for col in X.columns]
}
with open("columns.json", "w") as f:
    f.write(json.dumps(columns))
```

Build Python Flask Server to use these 2 artifacts

In [ ]:
```
```

sponsor

Abu Dhabi UAE, 14-16th of March 2020

Make your AI ideas real in UAE *pitch - hack - win $100 000*
A competition featuring top AI teams *$310 000 prizes*
Make your AI ideas real in UAE *pitch - hack - win $100 000*
A competition featuring top AI teams *$310 000 prizes*

PRIZES

DATA SCIENCE CHALLENGE
Turn data into winning solution

DATA

- You'll be using the data provided by local entities
- Data will be prepared for the hackathon
- You can use external data
- Local representatives will be there to answer your data questions

EXPECTED OUTCOME

- Working web application (desktop) using provided data
- Presentation explaining general idea and how application works

BENEFITS

Pitch your great idea and make it real

A unique opportunity to bring your great ideas to life together with representatives of the top global AI society

Win prizes from the pool of $310 000

Be recognised at the exclusive ISNR Abu Dhabi 2020 conference (25 000 participants)

Secure funding for your ideas Develop long-lasting working relationships with key executives from the UAE

AGENDA

Day 1: Present your team and idea
Day 2: Solve our problem during 16-hour hackathon based on real datasets
Day 3: Shortlisted teams present their solutions

PART OF ISNR Abu Dhabi 2020

the must-attend event for the National & Global Cyber Security Communities to do business, drive innovation, promote thought leadership and raise public awareness.

ISNR
INTERNATIONAL EXHIBITION FOR NATIONAL SECURITY & RESILIENCE

17-19 MARCH 2020
ADNEC, ABU DHABI, U.A.E

[ISNR website](#)

ISNR Abu Dhabi 2020 outlook

600 + Participating companies
4 Exhibition communities
3 Dedicated conferences
25 000 Participants

GLOBAL CHALLENGE FOR AI AND DATA SCIENCE TEAMS

Team requirements:

Have a team of 4-6 people

Be a company or a team of experts

Small company with annual turnover <$5M

Minimum 2 AI or Data Science experts

SOLVE ONE OF THREE CHALLENGES

AND BECOME A WINNER!
Challenge 1: SECURITY
Challenge 2: CIVIL DEFENSE
Challenge 3: SERVICES

Examples:

- **Patrol enhancement system. Predicting "property crime" from a city's topography**
- **Recognizing family-friendly content in Internet video**
- **Detecting fraud in online payments**
- **Injury prediction. Predicting overload in athletes based on their health and profile data**

Examples:

- **Robot or Human? Chatbot recognition**
- **Beat the pirates. Detecting ships in satellite images.**
- **Evacute.me Object recognition, generating potential evacuation routes.**
- **Detecting faulty gas pipeline.**
- **Anomaly detection based on IoT sensors in the energy industry**

Examples

- **Pricing real estate based on topography data**

- **Using NLP methods in stochastic grammar analysis.**
- **Predicting the probability of promotion based on employee profiles.**
- **TravelAI. Creating a destination recommendation system based on personal preferences like budget, season, duration length.**

There will be three challenges to solve - one in each of the three challenge groups above.
Exact challenges will be announced during the event

The location: Abu Dhabi National Exhibition Centre

Contact us at hello@betterworldhack.com