SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA

Shri Vaishnav Institute of Information and Technology

SESSION: 2022-23

Class – 3nd Year 5th Sem (M)

Branch – Data Science

Subject : Introduction to Data Science

Subject Code: BTIBM505

Submitted To – Prof . Omkant Sharma

Submitted by – Mr. Suyog Sinnarkar (20100BTCSDSI07299)

Mr. Aradhya Solanki (20100BTCSDSI07262)

Mr. Himanshu Gehlot (20100BTCSDSI07275)

Mr. Vibhor Gupta (20100BTCSDSI07301)

1.INTRODUCTION

Investment is a business activity that most people are interested in this globalization era. There are several objects that are often used for investment, for example, gold, stocks and property. In particular, property investment has increased significantly since 2011, both on demand and property selling.

We need a proper prediction on the real estate and the houses in housing market.

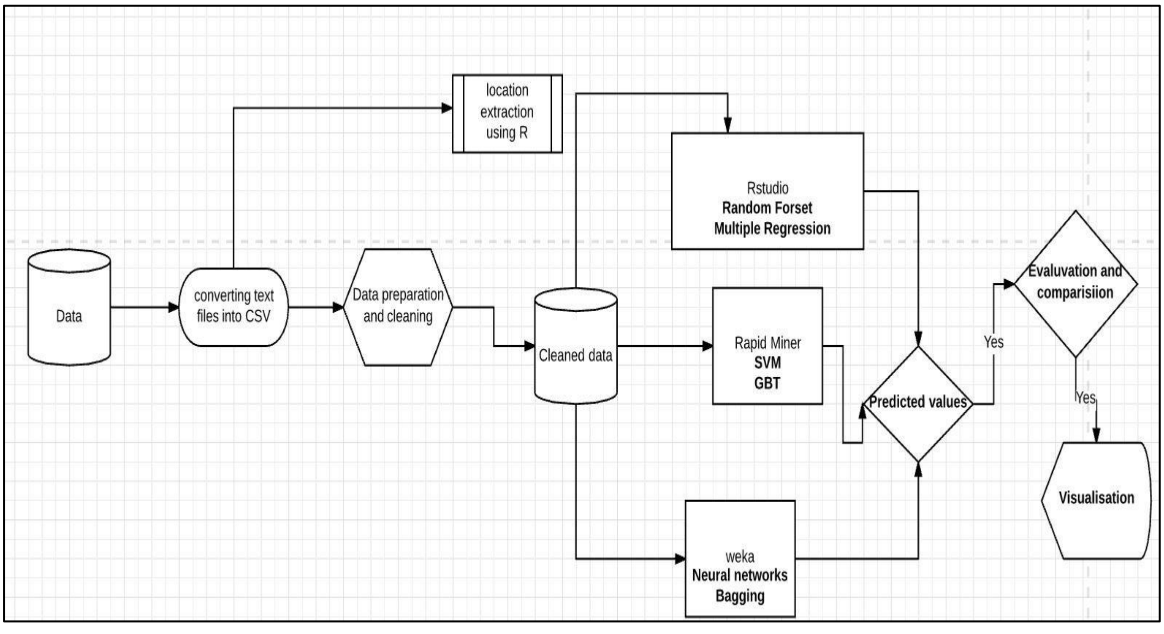
Many methods have been used in the price prediction like a hedonic regression in this we are trying to predict the predict the real estate price for the future using the machine learning techniques with the help of the previous works. We have used multiple regression and more algorithms with different tools to predict the house price So, it would be helpful for the people, so they will aware of both current and future situations, so it may avoid them in making mistakes.

2.PROBLEM

Prices of real estate properties are sophisticatedly linked with our economy. Despite this, we do not have accurate measures of house prices based on the vast amount of data available. Proper and justified prices of properties can bring in a lot of transparency and trust back to the real estate industry, which is very important for most consumers especially in India.

3.METHEDOLOGY

The below passages describe about the methodology used in the real estate



house price predictions and the architecture flow diagram is given.

3.1 Data Collection

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

3.2 Linear Regression

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

3.3 Decision Tree Regression

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

3.4 Classification Trees

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

4. RESULT

We created a function to predict the house price.Our function be like –

predict_price(location, sqft, bath,bhk) "

When we pass the values into our function, it will predict house price for us.

5.CONCLUSION

The main goal of this project is to determine the prediction for prices which we have successfully done using different machine learning algorithms like a Random forest, multiple regression ,so it's clear that linear regression have more accuracy in prediction when com- pared to the others and also my research provides to find the attributes contribution in prediction.

We have performed step by step procedure to analyze the dataset and found the correlation between the parameters. The manually collected Real-time Dataset has been collected which contains 1635 entries and independent variables. We analyze and pre- process this dataset before performing Exploratory Data Analysis. This analyzed feature set was given as an input to machine learning algorithms and calculated the performance of each model to compare based on Accuracy score. We found that Linear Regression fits our dataset and gives the highest accuracy of 85.64%. Decision Tree gives the least accuracy of 56.02%. Support Vector Regression gives an accuracy of 62.81%. Thus we conclude that we implemented regression techniques to check how well an algorithm fits to given problem statement of House price prediction.

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)
df1 = pd.read_csv("C:/Users/Asus/OneDrive/Desktop/Bengaluru_House_Data.csv")
df1.head()
```

Out[1]:

	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 [2]:
```
df1.shape
```

Out[2]:
```
(13320, 9)
```

In [3]:
```
df1.columns
```

Out[3]:
```
Index(['area_type', 'availability', 'location', 'size', 'society',
       'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')
```

In [4]:
```
df1['area_type'].unique()
```

Out[4]:
```
array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
       'Carpet  Area'], dtype=object)
```

In [5]:
```
df1['area_type'].value_counts()
```

Out[5]:
```
Super built-up  Area    8790
Built-up  Area          2418
Plot  Area              2025
Carpet  Area              87
Name: area_type, dtype: int64
```

In [6]:
```
df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
df2.shape
```

Out[6]:
```
(13320, 5)
```

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

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

In [8]:
```
df2.shape
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[8]: (13320, 5)

In [9]:
```
df3 = df2.dropna()
df3.isnull().sum()
```

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

In [10]:
```
df3.shape
```

Out[10]: (13246, 5)

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

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_3652\2716584372.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

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

In [12]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

In [13]:
```
2+3
```

Out[13]: 5

In [14]:
```
df3[~df3['total_sqft'].apply(is_float)].head(10)
```

Out[14]:

	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 [15]:
```
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 [16]:

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

Out[16]:

	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 [17]:

```
df4.loc[30]
```

Out[17]:

```
location      Yelahanka
size              4 BHK
total_sqft       2475.0
bath                4.0
price             186.0
bhk                   4
Name: 30, dtype: object
```

In [18]:

```
(2100+2850)/2
```

Out[18]:

```
2475.0
```

In [19]:

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

Out[19]:

	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 [20]:

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

Out[20]:

```
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 [21]:

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

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
                               ...
Rajanna Layout                    1
Subramanyanagar                   1
Lakshmipura Vidyaanyapura         1
Malur Hosur Road                  1
Abshot Layout                     1
Name: location, Length: 1287, dtype: int64
```

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
```

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

Out[27]:
```
BTM 1st Stage                   10
Gunjur Palya                    10
Nagappa Reddy Layout            10
Sector 1 HSR Layout             10
Thyagaraja Nagar                10
                                ..
Rajanna Layout                   1
Subramanyanagar                  1
Lakshmipura Vidyaanyapura        1
Malur Hosur Road                 1
Abshot Layout                    1
Name: location, Length: 1047, dtype: int64
```

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
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

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	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 [32]:

```
df5.shape
```

Out[32]:

```
(13200, 7)
```

In [33]:

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

Out[33]:

```
(12456, 7)
```

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
```

In [35]:

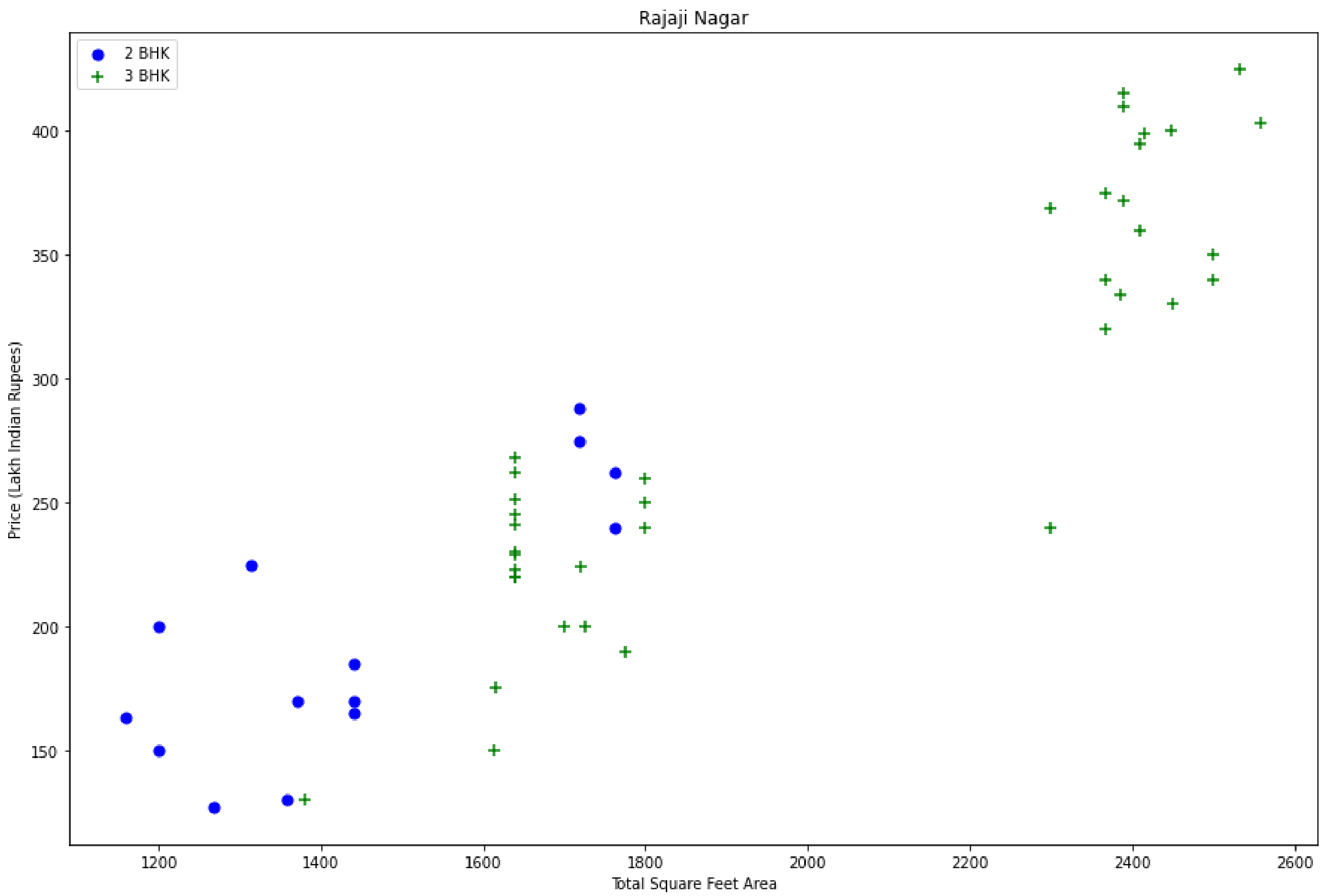
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st)
        df_out = pd.concat([df_out,reduced_df],ignore_index=True)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

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[35]: (10242, 7)

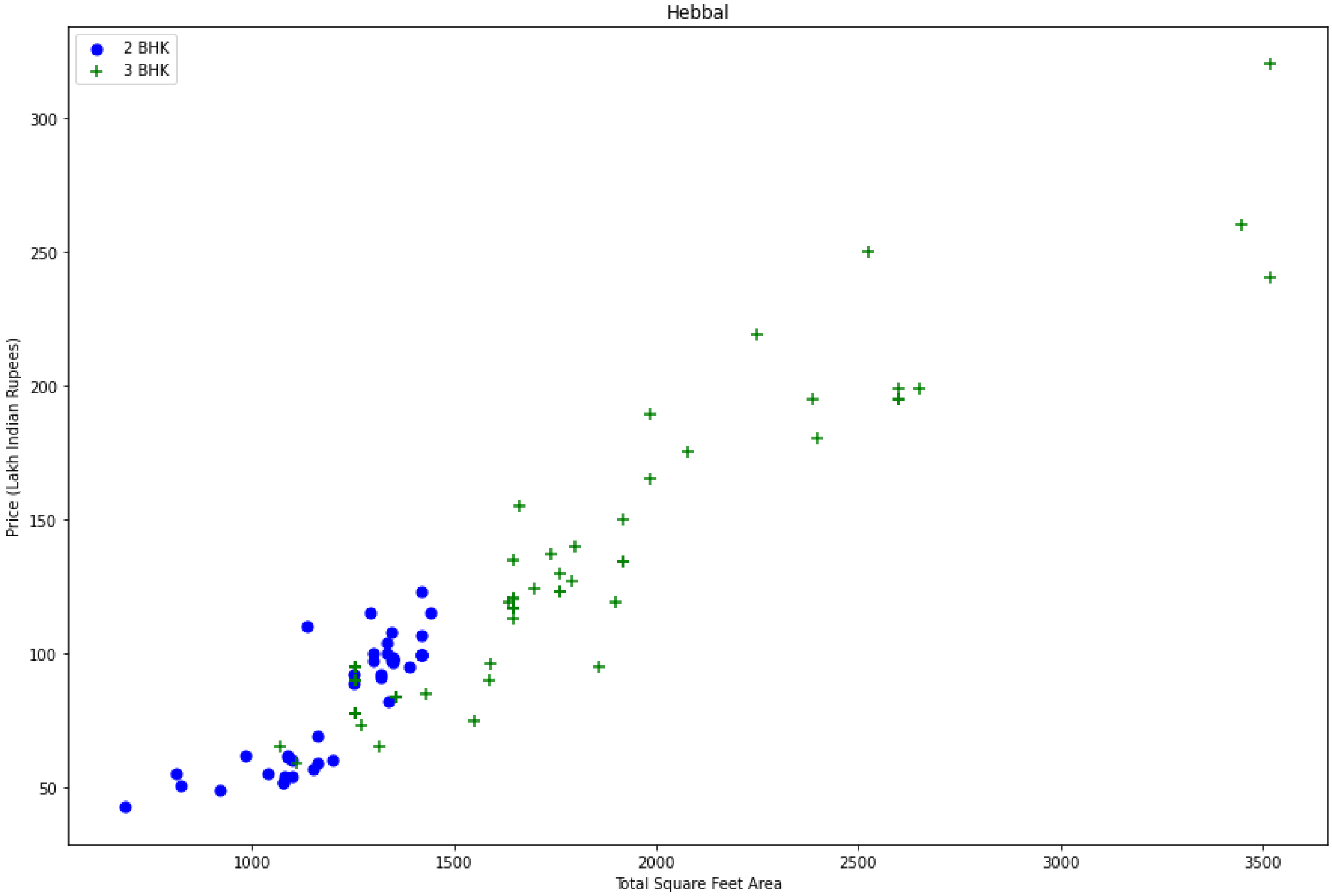
In [36]:
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', 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")
```



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

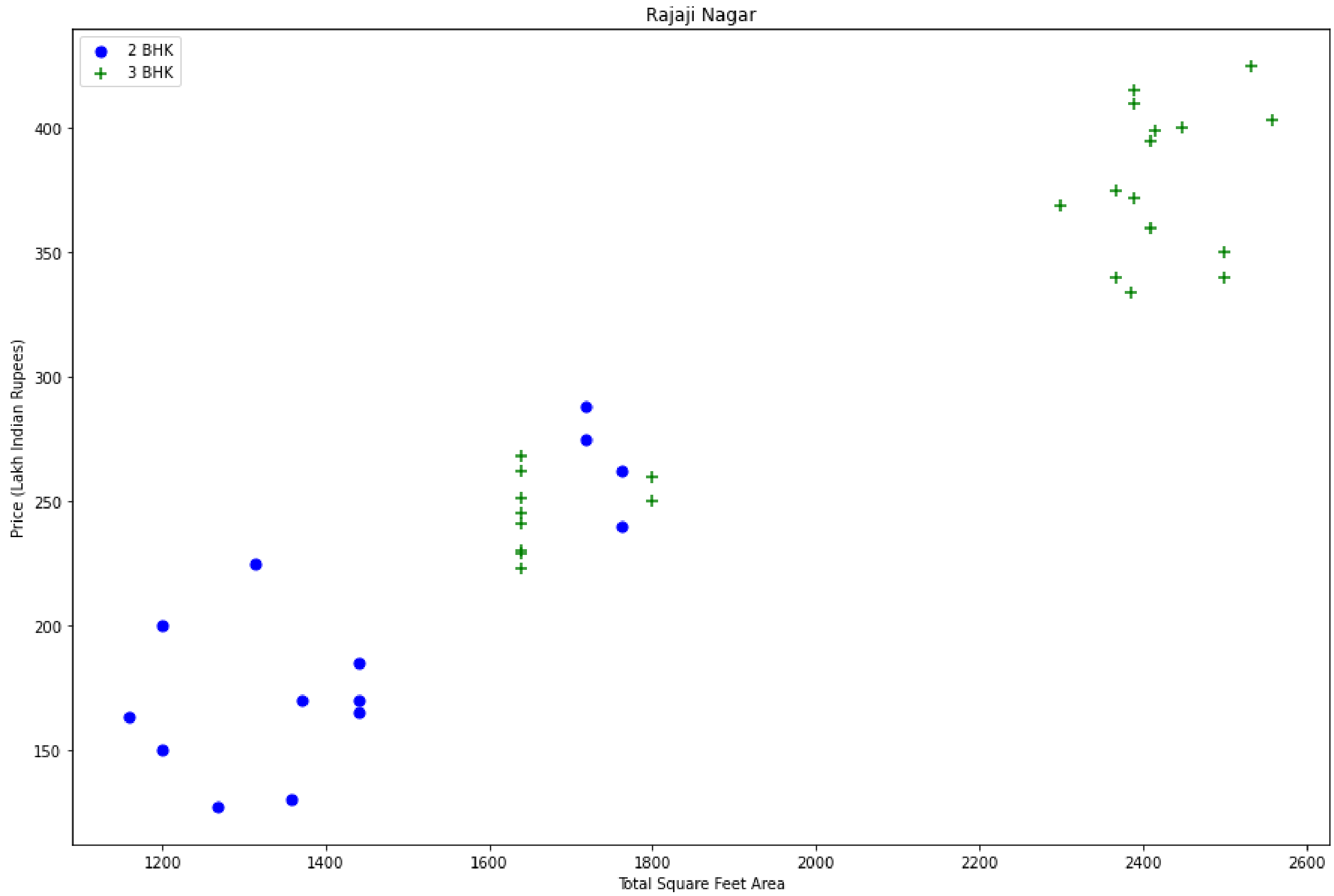
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



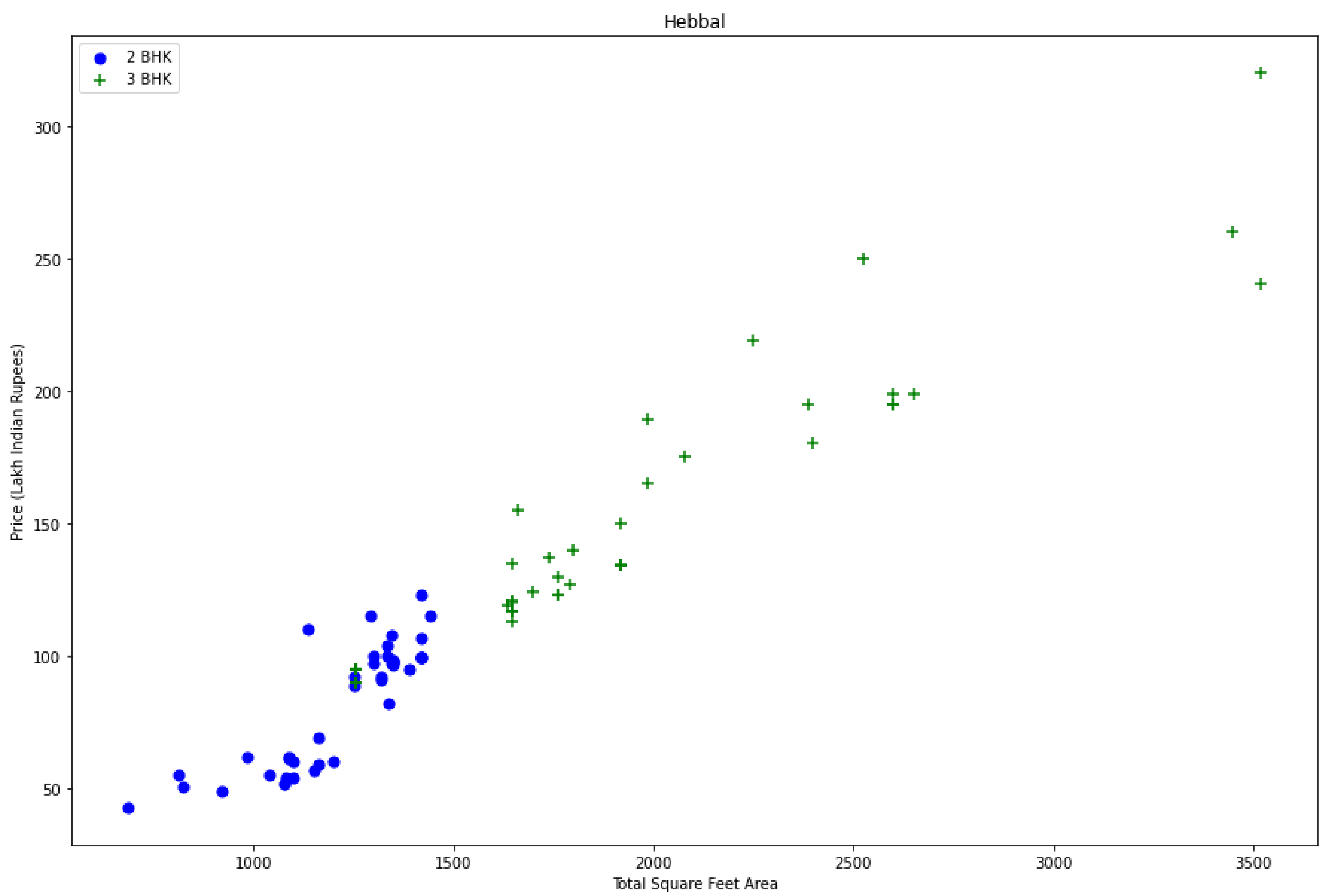
In [38]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                '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_sqf
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
```

Out[38]: (7317, 7)

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




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

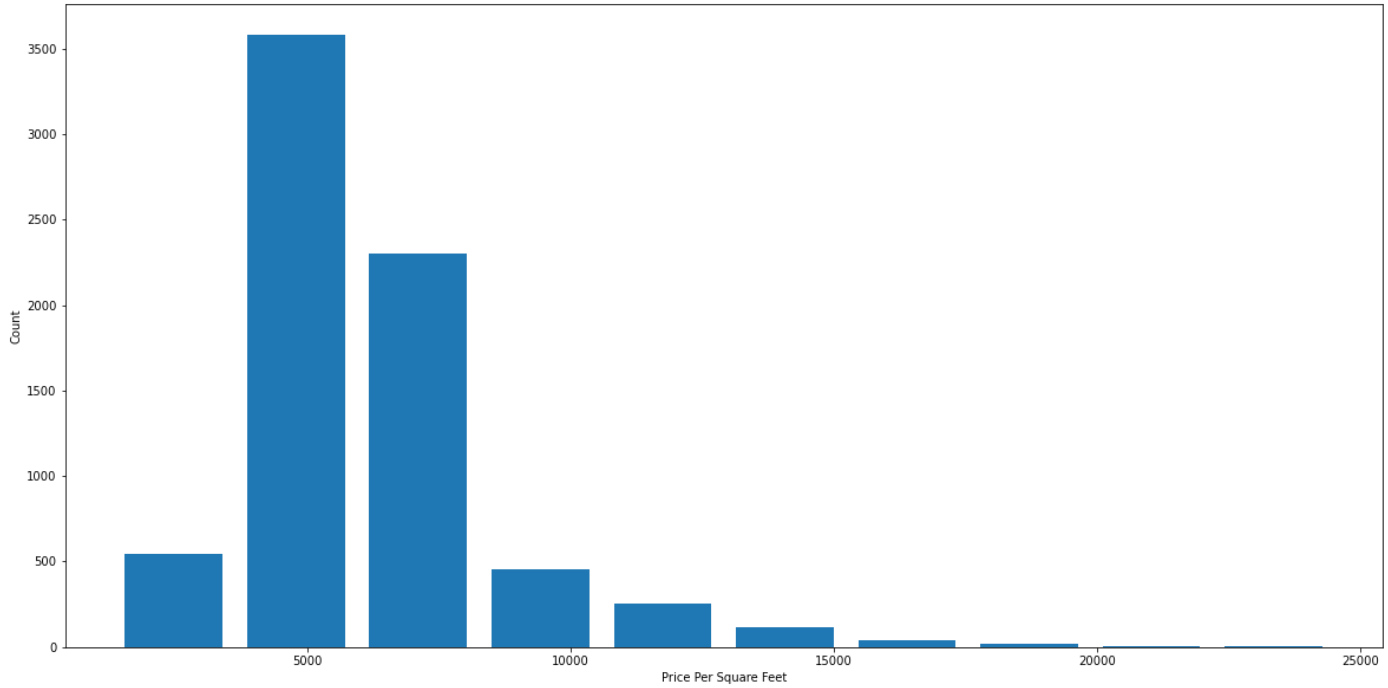



```
In [41]: import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
plt.hist(df8.price_per_sqft,rwidth=0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

Out[41]: Text(0, 0.5, 'Count')

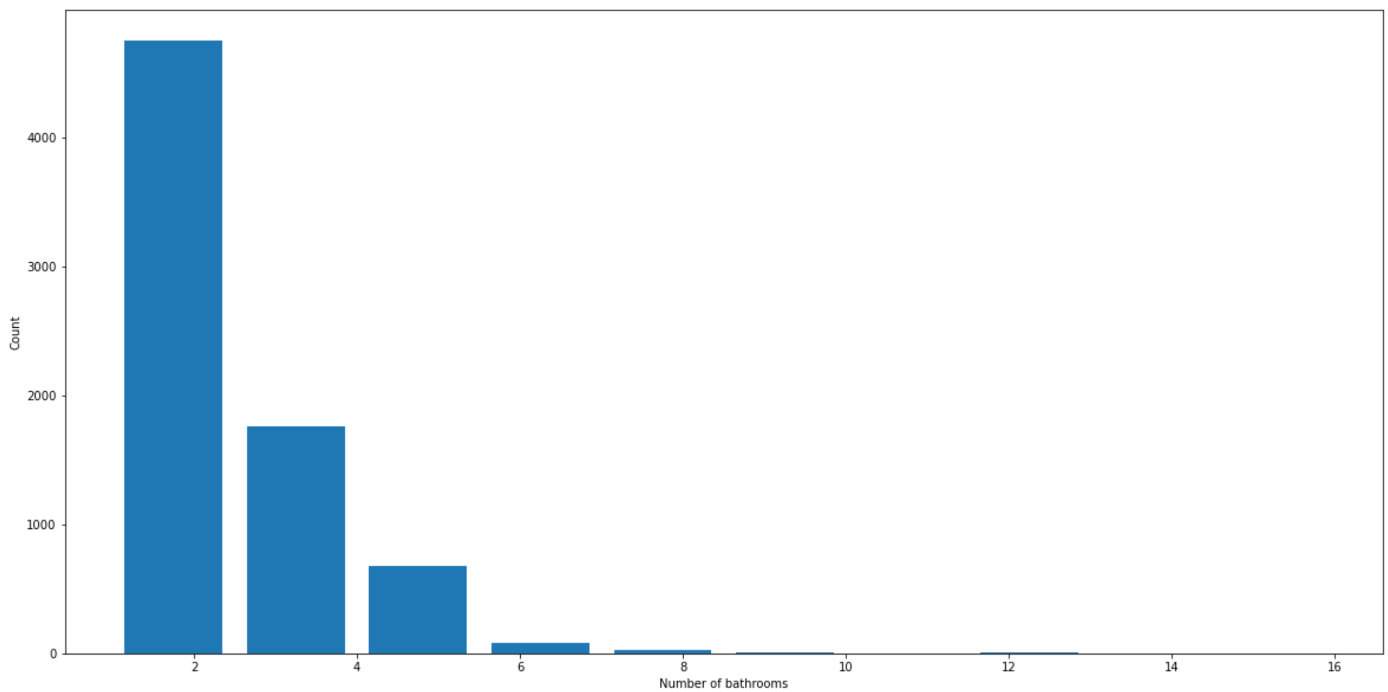


In [42]:
```
df8.bath.unique()
```

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]
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

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

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

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

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	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	...	Vishveshwarya Layout
0	1	0	0	0	0	0	0	0	0	0	...	0
1	1	0	0	0	0	0	0	0	0	0	...	0
2	1	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')
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```python
df11.head()
```

Out[50]:

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

5 rows × 245 columns

In [51]:

```python
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	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishv
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	

2 rows × 244 columns

In [52]:

```python
df12.shape
```

Out[52]:

```
(7239, 244)
```

In [53]:

```python
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	Vish
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	

3 rows × 243 columns

In [54]:

```python
X.shape
```

Out[54]:

```
(7239, 243)
```

In [55]:

```python
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]:
```
0.8629132245229485
```

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])
```

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=
        gs.fit(X,y)
        scores.append({
```

```
            '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)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing sta
ge. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

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

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

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


  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing sta
ge. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

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

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

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


  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing sta
ge. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

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

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

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


  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing sta
ge. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

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

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

kwargs = {s[0] + '__sample_weight': sample_weight for s in model.steps}
```

```
model.fit(X, y, **kwargs)


  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2.
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing sta
ge. To reproduce the previous behavior:

from sklearn.pipeline import make_pipeline

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

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

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


  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2. Please leave
the normalize parameter to its default value to silence this warning. The default behavi
or of this estimator is to not do any normalization. If normalization is needed please u
se sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2. Please leave
the normalize parameter to its default value to silence this warning. The default behavi
or of this estimator is to not do any normalization. If normalization is needed please u
se sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2. Please leave
the normalize parameter to its default value to silence this warning. The default behavi
or of this estimator is to not do any normalization. If normalization is needed please u
se sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2. Please leave
the normalize parameter to its default value to silence this warning. The default behavi
or of this estimator is to not do any normalization. If normalization is needed please u
se sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2. Please leave
the normalize parameter to its default value to silence this warning. The default behavi
or of this estimator is to not do any normalization. If normalization is needed please u
se sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarn
ing: 'normalize' was deprecated in version 1.0 and will be removed in 1.2. Please leave
the normalize parameter to its default value to silence this warning. The default behavi
or of this estimator is to not do any normalization. If normalization is needed please u
se sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
```

```
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\tree\_classes.py:359: FutureWarning:
Criterion 'mse' was deprecated in v1.0 and will be removed in version 1.2. Use `criterio
n='squared_error'` which is equivalent.
  warnings.warn(
```

Out[60]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726752	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.717160	{'criterion': 'friedman_mse', 'splitter': 'best'}

In [61]:

```python
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]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not
have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[62]: 83.86570258324036

In [63]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[63]:
```
86.08062284998763
```

In [64]:
```python
predict_price('Indira Nagar',1000, 2, 2)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[64]:
```
193.31197733179548
```

In [65]:
```python
predict_price('Indira Nagar',1000, 3, 3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[65]:
```
195.52689759854277
```

In [ ]:

Property Price Prediction

using Data Science and Machine Learning

Guided by-

Prof. Omkant Sharma

Team Members

01 Suyog Sinnarkar

02 Himanshu Gehlot

03 Aradhya Solanki

04 Vibhor Gupta

5-Step Plan

01	Introduction
02	Problem Statement
03	Project Specification
04	Data Set
05	Pipeline
06	Result

- # INTRODUCTION

- In this project, we have built a machine learning model to predict the house prices of an Indian city Bengaluru

- his project will very helpful for the real estate market. Our model can be used by both house sellers and house buyers

- Multiple Linear Regression algorithm is used to create a model with a great accuracy score.

- # PROBLEM STATEMENT

- Prices of real estate properties are sophisticatedly linked with our economy.

- Despite this, we do not have accurate measures of house prices based on the vast amount of data available.

- Proper and justified prices of properties can bring in a lot of transparency and trust back to the real estate industry, which is very important for most consumers especially in India

• PROJECT SPECIFICATION

- The goal of this project is to predict house prices in Bengaluru city based on some features such as location, size/area, number of bedrooms, and number of bathrooms.

- Bengaluru house price dataset is used to create the model.

- We are using Machine Learning Algorithm to create a predictive model.

- Multiple Linear Regression algorithm is used to train and test the model in our project.

• DATASET

- The data set comes from Kaggle.com

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

- There are 13320 number of observations in our dataset.

- There are a total of 9 columns/attributes in our dataset.

- The all 9 columns are area_type, availability, location, size, total_sqft, bath, society, balcony, and price.

PIPELINE

DATA CLEANING

FEATURE ENGINEERING

ONE HOT ENCODING

OUTLIER DETECTION

OUTLIER REMOVAL

MODEL CREATION

• DATA CLEANING

- The main aim of Data Cleaning is to identify and remove errors & duplicate data, in order to create a reliable dataset.
- The process of data cleaning is done by using a very famous library pandas.
- Initially, those columns/features are dropped from our dataset who are not important in deciding the final price.
- . The rows having a null value in any columns are dropped from our dataset

- # FEATURE ENGINEERING

- Feature engineering is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithms. Feature engineering can be considered as applied machine learning itself.

- Dimensionality reduction techniques are used in our dataset to reduce those rows who are not very much important to decide the house price.

• OUTLIER DETECTION

- In simple words, an outlier is an observation that diverges from an overall pattern on a sample.

- There are many types of outlier detection techniques such as Z-Score or Extreme Value Analysis,

- Probabilistic and Statistical Modeling, Information Theory Models, Standard Deviation etc.

- We have used simple domain knowledge of real estate market to detect the outliers in our dataset.

- # OUTLIER REMOVAL

- After detecting the outlier, correct that errors if possible and if you can not fix it, then remove that observation.
- In our dataset, we observed variations in the relation between values of some attributes.
- So that these type of rows are dropped from the dataset.
- Scatter plots are used to detect some more outliers and they are also removed from our dataset.

• ONE HOT ENCODING

- This technique is used to convert the categorical variables into numeric values.

- Our dataset contains a categorical variable which is "location".

- We have used one hot encoding method to convert them as numeric values.

As we can see the location name 1st Block Jayanagar having the value 1 and the rest of the locations are treated as 0 .

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

49]:

	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	7th Phase JP Nagar	8th Phase JP Nagar	9th Phase JP Nagar	...	Vishveshwarya Layout
0	1	0	0	0	0	0	0	0	0	0	...	0
1	1	0	0	0	0	0	0	0	0	0	...	0
2	1	0	0	0	0	0	0	0	0	0	...	0

3 rows × 241 columns

• MODEL CREATION

- The process of modeling means training a machine learning algorithm to predict the labels from the features.
- We have used Linear Regression algorithm for trainig and testing of the model.
- The accuracy rate of our model is 87% which is pretty good.

```
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)
```

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

```
0.8629132245229485
```

• GRID SEARCH VALIDATION

- This is a technique which is used to find the best algorithm for modeling and give the best parameters as well.

- We applied grid search cross validation method on our dataset with Linear Regression, Lasso Regression, and Decision Tree algorithms.

- We find the Linear Regression algorithm is giving the best accuracy score as more than 80%.

60]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726752	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.717160	{'criterion': 'friedman_mse', 'splitter': 'best'}

61]:

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

• RESULT

- We created a function to predict the house price.

- Our function be like " predict_price(location, sqft, bath,bhk) "
  When we pass the values into our function, it will predict house price for us

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

nJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

C:\ProgramData\Anaconda3\lib\site-packages\sklearn
have valid feature names, but LinearRegression was
  warnings.warn(
]: 86.08062284998763
```