House Price Predictor

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

Machine Learning Project

Submitted By:

Japjot Singh
Roll no. - 102103174

Submitted To:

Ms. Kudratdeep Aulakh

Index

1. Introduction

1.1 Name of the dataset
Dataset Link

Bengaluru_House_Data (from Kaggle)

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

1.2 Description

Linear Regression:

Linear Regression is a supervised machine learning model that attempts to model a linear relationship between dependent variables (Y) and independent variables (X). Every evaluated observation with a model, the target (Y)'s actual value is compared to the target (Y)'s predicted value, and the major differences in these values are called residuals. The Linear Regression model aims to minimize the sum of all squared residuals. Here is the mathematical representation of the linear regression:

$$Y= a0+a1X+ \varepsilon$$

The values of X and Y variables are training datasets for the model representation of linear regression. When a user implements a linear regression, algorithms start to find the best fit line using a0 and a1. In such a way, it becomes more accurate to actual data points; since we recognize the value of a0 and a1, we can use a model for predicting the response.

- People looking to buy a new home tend to be more conservative with their budgets and market strategies.

- This project aims to analyse various parameters like average income, average area etc. and predict the house price accordingly.

- This application will help customers to invest in an estate without approaching an agent
- To provide a better and fast way of performing operations.
- To provide proper house price to the customers.
- To eliminate need of real estate agent to gain information regarding house prices.
- To provide best price to user without getting cheated.
- To enable user to search home as per the budget.
- The aim is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. By analyzing previous market trends and price ranges, and also upcoming developments future prices will be predicted.
- House prices increase every year, so there is a need for a system to predict house prices in the future.
- House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house.
- We use linear regression algorithm in machine learning for predicting the house price trends

2. Libraries Used

NumPy (import numpy as np):-

NumPy is used for numerical operations in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these elements.

Pandas (import pandas as pd):

Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrame, which is particularly useful for working with structured data, such as tabular data.

Matplotlib (import matplotlib.pyplot as plt):

Matplotlib is a plotting library that produces static, animated, and interactive visualizations in Python. Here, it is likely used for creating various plots and charts to visualize data.

Scikit-learn:

Scikit-learn (from sklearn.tree import DecisionTreeRegressor, from sklearn.ensemble import RandomForestRegressor, from sklearn.linear_model import LinearRegression, from sklearn.metrics import mean_squared_error, from sklearn.model_selection import train_test_split, from sklearn.metrics import r2_score, from sklearn import preprocessing):

3. Algorithm(s) Used

Linear Regression:

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

k fold cross validation:

K-fold cross-validation is a technique for evaluating predictive models. The dataset is divided into k subsets or folds. The model is trained and evaluated k times, using a different fold as the validation set each time. Performance metrics from each fold are averaged to estimate the model's generalization performance.

Decision tree (GridSearchCV):

A decision tree is a decision support hierarchical model that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

4. Code and Screenshots

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

In [2]:
```
#Reading the csv file
df1 = pd.read_csv("Desktop/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
```

In [5]:
```
#Dropping out the unnecessary parameters
df2 = df1.drop(['area_type', 'society', 'balcony', 'availability'], axis = 'columns')
df2.head()
```

Out[5]:

	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

In [6]:
```
#Checking the values with null
df2.isnull().sum()
```

Out[6]:
```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [7]:
```
#Dropping the values which are null
df3 = df2.dropna()
df3.head()
```

Out[7]:

	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

In [8]:
```
df3.shape
```

Out[8]:
```
(13246, 5)
```

In [9]:
```
df3['size'].unique()
```

Out[9]:
```
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 [10]:
```
df3['bhk'] = df3['size'].apply(lambda x : int(x.split(' ')[0]))
```

In [11]:
```
df3.head()
```

Out[11]:

	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 [12]:
```
df3[df3.bhk>20]
```

Out[12]:

	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 [13]:
```
df3.total_sqft.unique()
```

Out[13]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [14]:
```
#For getting the values which are not in float
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

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

Out[15]:

	location	size	total_sqft	bath	price	bhk
30	Yelahanka	4 BHK	2100 - 2850	4.0	186.000	4
122	Hebbal	4 BHK	3067 - 8156	4.0	477.000	4
137	8th Phase JP Nagar	2 BHK	1042 - 1105	2.0	54.005	2
165	Sarjapur	2 BHK	1145 - 1340	2.0	43.490	2
188	KR Puram	2 BHK	1015 - 1540	2.0	56.800	2
410	Kengeri	1 BHK	34.46Sq. Meter	1.0	18.500	1
549	Hennur Road	2 BHK	1195 - 1440	2.0	63.770	2
648	Arekere	9 Bedroom	4125Perch	9.0	265.000	9
661	Yelahanka	2 BHK	1120 - 1145	2.0	48.130	2
672	Bettahalsoor	4 Bedroom	3090 - 5002	4.0	445.000	4

In [16]:
```
#Getting the mean values which are present in range(For example 2100-2850)
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0]) + float(tokens[1])) / 2
    try:
        return float(x)
    except:
        return None
```

In [17]:
```
df4 = df3.copy()
df4['total_sqft'] = df4['total_sqft'].apply(convert_sqft_to_num)
df4.head(3)
```

In [18]:
```
#Getting a new coloumn ('price per square fit')
df5 = df4.copy()
df5['price_per_sqft'] = df5['price'] * 100000 / df5['total_sqft']
```

In [19]:
```
len(df5.location.unique())
```

Out[19]:
```
1304
```

In [20]:
```
#Grouping data with respect to location
df5.location = df5.location.apply(lambda x : x.strip())

location_stats = df5.groupby('location')['location'].agg('count').sort_values(ascending = False)
location_stats
```

Out[20]:
```
location
Whitefield               535
Sarjapur  Road           392
Electronic City          304
Kanakpura Road           266
Thanisandra              236
                        ...
1 Giri Nagar               1
Kanakapura Road,           1
Kanakapura main  Road      1
Karnataka Shabarimala      1
whitefiled                 1
Name: location, Length: 1293, dtype: int64
```

In [21]:
```
#Getting the number of loations with less than equal to 10 examples
len(location_stats[location_stats<=10])
```

Out[21]:
```
1052
```

In [22]:
```
#Displaying the locations with less examples
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[22]:
```
location
Basapura                10
1st Block Koramangala   10
Gunjur Palya            10
Kalkere                 10
Sector 1 HSR Layout     10
                        ..
1 Giri Nagar             1
Kanakapura Road,         1
Kanakapura main  Road    1
Karnataka Shabarimala    1
whitefiled               1
Name: location, Length: 1052, dtype: int64
```

In [23]:
```
#Getting the number of unique location
len(df5.location.unique())
```

Out[23]:
```
1293
```

In [24]:
```
df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
len(df5.location.unique())
```

Out[24]:
```
242
```

In [25]:
```
df5.head(10)
```

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

In [26]:
```
df5[df5.total_sqft/df5.bhk<300].head()
```

Out[26]:

	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 [27]:
```
df5.shape
```

Out[27]:
```
(13246, 7)
```

In [28]:
```
#Removing the outliers
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.shape
```

Out[28]:
```
(12502, 7)
```

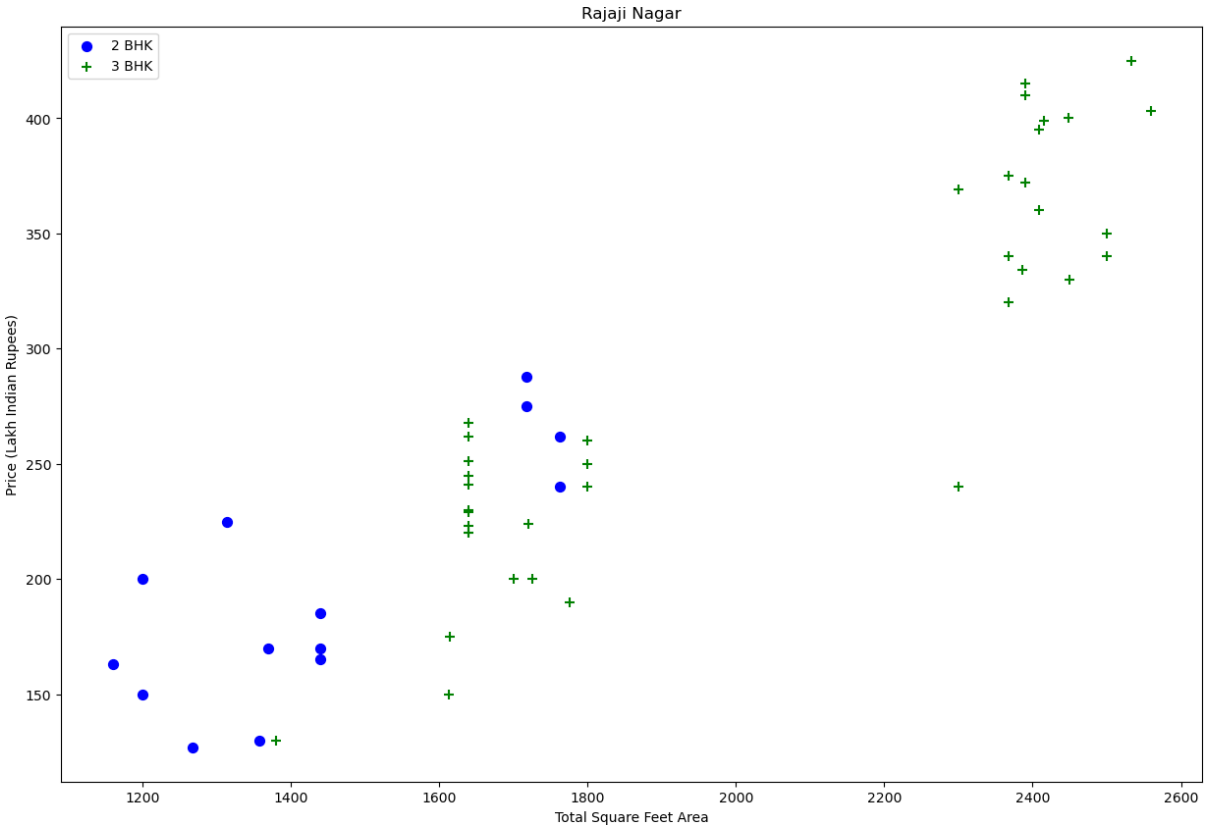
In [29]:
```
df6.price_per_sqft.describe()
```

Out[29]:
```
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 [30]:
```
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[30]:
```
(10241, 7)
```

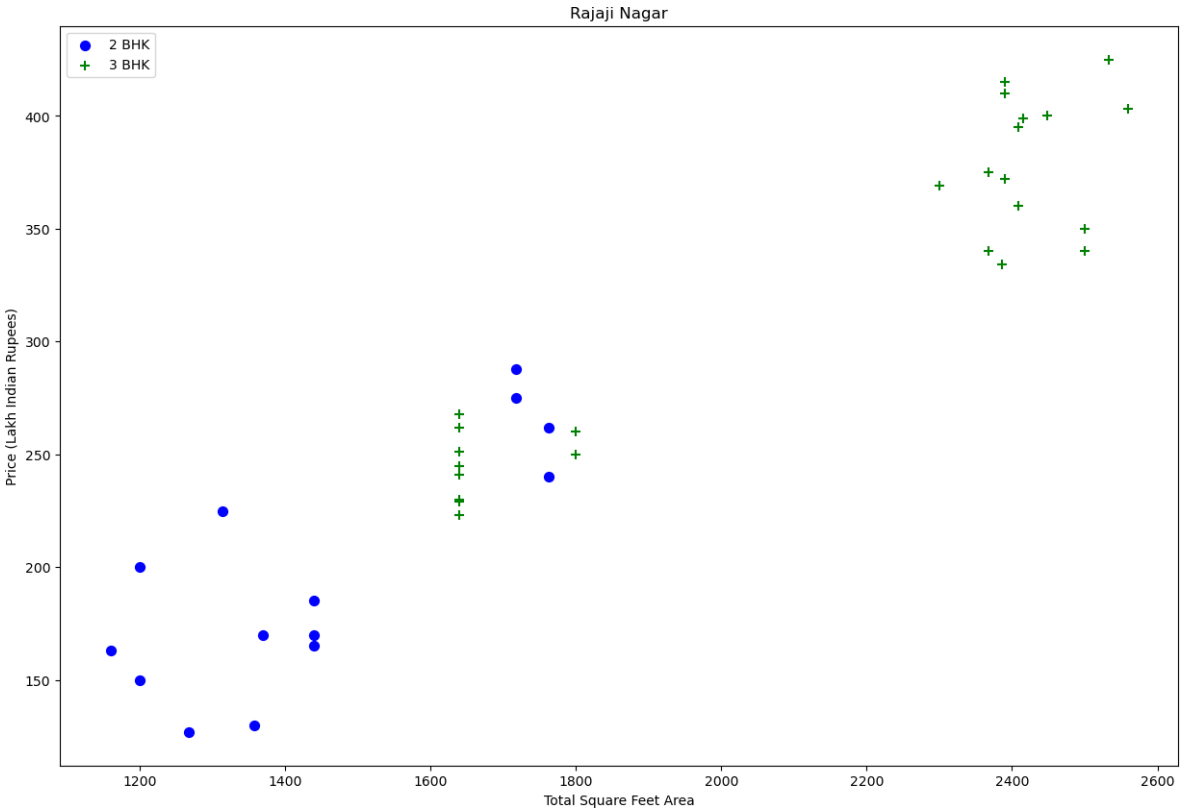
In [31]:
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', 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")
```



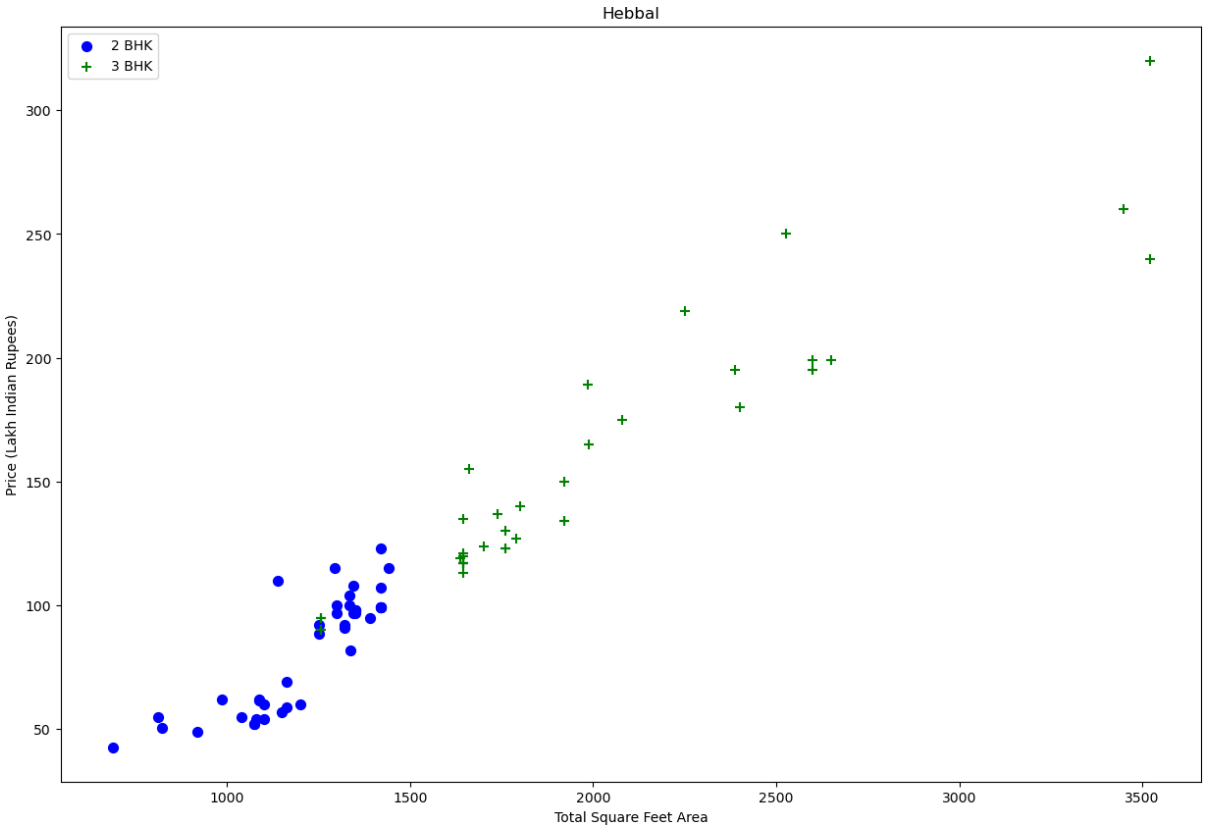
In [32]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                'std': np.std(bhk_df.price_per_sqft),
                'count': bhk_df.shape[0]
            }
        for bhk, bhk_df in location_df.groupby('bhk'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.price_per_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
# df8 = df7.copy()
df8.shape
```

Out[32]: (7329, 7)

In [33]:
```
#Plot same scatter chart again to visualize price_per_sqft for 2 BHK and 3 BHK properties
plot_scatter_chart(df8,"Rajaji Nagar")
```

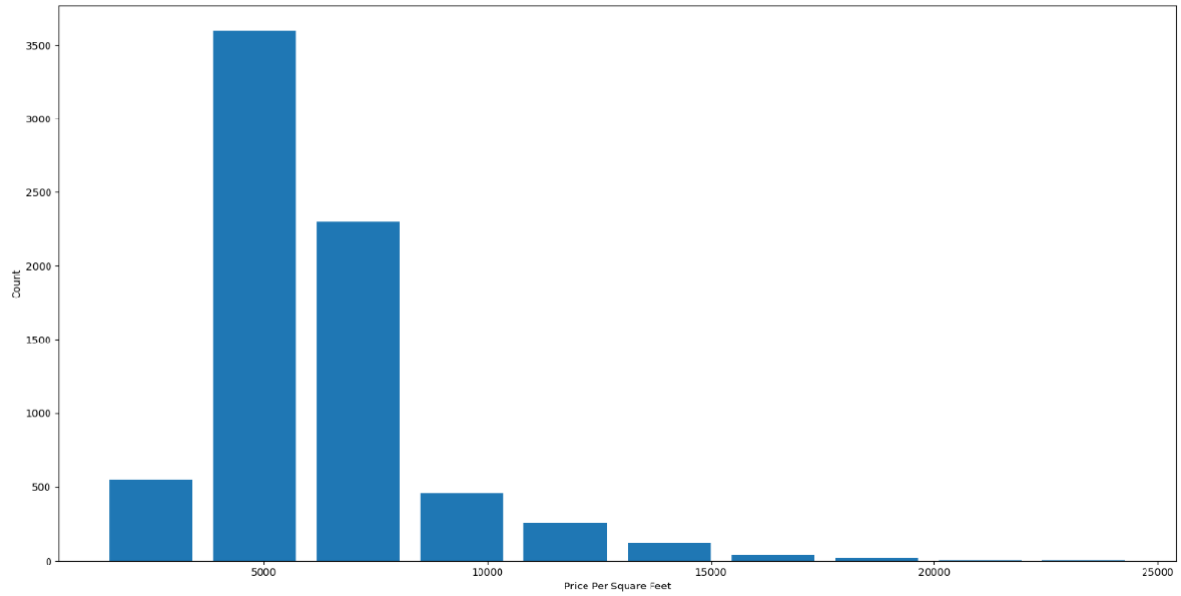

Rajaji Nagar
2 BHK
3 BHK
Price (Lakh Indian Rupees)
Total Square Feet Area
150
200
250
300
350
400
1200
1400
1600
1800
2000
2200
2400
2600

```
In [34]: plot_scatter_chart(df8, "Hebbal")
```




```
In [35]: 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[35]: Text(0, 0.5, 'Count')
```

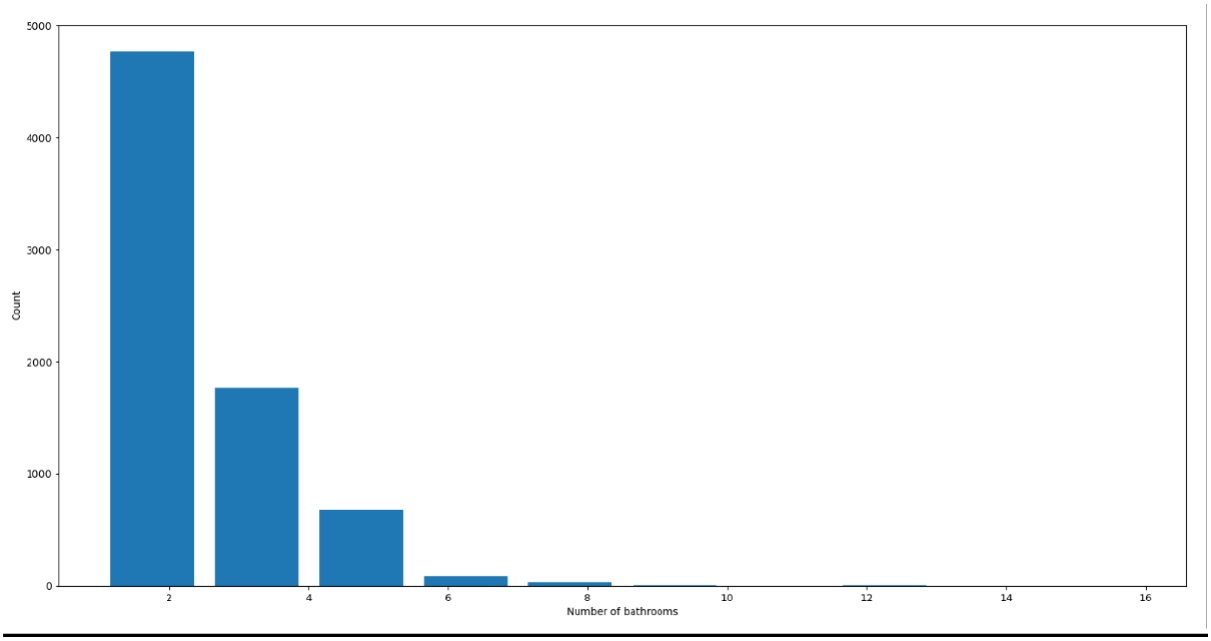


In [36]:
```
df8.bath.unique()
```

Out[36]:
```
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

In [37]:
```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[37]:
```
Text(0, 0.5, 'Count')
```



In [38]:
```
df8[df8.bath>10]
```

Out[38]:

	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 [39]:
```
df8[df8.bath>df8.bhk+2]
```

Out[39]:

	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 [40]:
```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

Out[40]:
```
(7251, 7)
```

In [41]:
```
df9.head(2)
```

Out[41]:

	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 [42]:
```
df10 = df9.drop(['size','price_per_sqft'],axis='columns')
df10.head(3)
```

Out[42]:

	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 [43]:
```
dummies = pd.get_dummies(df10.location)
dummies.head(3)
```

Out[43]:

	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	Vishwapriya Layout	Vittasandra	Whitefield	Yelachenahal
0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

3 rows × 242 columns

In [44]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[44]:

	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	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	0	0	0	0	0
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0	...	0	0	0	0	0
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0	...	0	0	0	0	0
3	1st Block Jayanagar	1200.0	2.0	130.0	3	1	0	0	0	0	...	0	0	0	0	0
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...	0	0	0	0	0

5 rows × 246 columns

In [45]:
```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

Out[45]:

In [45]:
```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

Out[45]:

	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	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield	Y
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	0	0	0	0	
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	0	0	0	0	

2 rows × 245 columns

In [46]:
```
df12.shape
```

Out[46]: (7251, 245)

In [47]:
```
X = df12.drop(['price'],axis='columns')
X.head(3)
```

Out[47]:

	total_sqft	bath	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	6th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapriya Layout	Vittasandra	Whitefield
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	0	0	0	0
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0	0

3 rows × 244 columns

In [48]:
```
X.shape
```

Out[48]: (7251, 244)

In [49]:
```
y = df12.price
y.head(3)
```

Out[49]:
```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

In [50]:
```
len(y)
```

Out[50]: 7251

In [51]:
```
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)
```

```
E:\anaconda\lib\site-packages\scipy\__init__.py:155: UserWarning: A NumPy version >=1.1
sion of SciPy (detected version 1.26.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [52]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[52]: 0.845227769787429

In [52]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

Out[52]:
```
0.845227769787429
```

In [53]:
```
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[53]:
```
array([0.82430186, 0.77166234, 0.85089567, 0.80837764, 0.83653286])
```

In [ ]:
```

```

In [54]:
```
import pandas as pd
from sklearn.model_selection import GridSearchCV, ShuffleSplit
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.tree import DecisionTreeRegressor

def find_best_model_using_gridsearchcv(X, y):
    algos = {
        'linear_regression': {
            'model': LinearRegression(),
            'params': {
                'fit_intercept': [True, False],
                'copy_X': [True, False],
                'positive': [True, False],
                'n_jobs': [-1]  # Assuming you want to specify this parameter
            }
        },
```

```
        'lasso': {
            'model': Lasso(),
            'params': {
                'alpha': [1, 2],
                'selection': ['random', 'cyclic']
            }
        },
        'decision_tree': {
            'model': DecisionTreeRegressor(),
            'params': {
                'criterion': ['mse', 'friedman_mse'],
                'splitter': ['best', 'random']
            }
        }
    }
    scores = []
    cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
    for algo_name, config in algos.items():
        gs = GridSearchCV(config['model'], config['params'], cv=cv, return_train_score=False)
        gs.fit(X, y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

    return pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])

# Assuming X and y are your feature matrix and target variable, respectively
result = find_best_model_using_gridsearchcv(X, y)
print(result)
```

In [55]:
```
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 [56]:
```
predict_price('1st Phase JP Nagar',1000, 2, 2)
```

```
E:\anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does
itted with feature names
  warnings.warn(
```

Out[56]:
```
83.49904677172414
```

In [57]:
```
predict_price('1st Phase JP Nagar',1000, 3, 3)
```

```
E:\anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does
itted with feature names
  warnings.warn(
```

Out[57]:
```
86.80519395198999
```