In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (20,10)
```

In [2]:
```
df1 = pd.read_csv("C:\\Users\\B Akhil\\OneDrive\\Desktop\\ML\\Projects\\Bengalur
df1.head()
```

Out[2]:

	area_type	availability	location	size	society	total_sqft	bath	balco
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	

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

Out[3]:
```
(13320, 9)
```

In [4]:
```
#data cleaning
```

In [5]:
```
df1.groupby('area_type')['area_type'].agg('count')
```

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

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

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
...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453	4.0	231.00
13316	Richards Town	4 BHK	3600	5.0	400.00
13317	Raja Rajeshwari Nagar	2 BHK	1141	2.0	60.00
13318	Padmanabhanagar	4 BHK	4689	4.0	488.00
13319	Doddathoguru	1 BHK	550	1.0	17.00

13320 rows × 5 columns

In [7]:
```
df2.isna().sum()#finds if any cell as None value
```

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

In [8]:
```
df3 = df2.dropna() #drops all rows, if that row contains None value
df3.isna().sum()
```

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

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]))
df3
```

```
C:\Users\B Akhil\AppData\Local\Temp\ipykernel_15568\746689020.py:1: SettingWithCo
pyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df3['bhk'] = df3['size'].apply(lambda x : int(x.split()[0]))
```

Out[10]:

	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
...	...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453	4.0	231.00	5
13316	Richards Town	4 BHK	3600	5.0	400.00	4
13317	Raja Rajeshwari Nagar	2 BHK	1141	2.0	60.00	2
13318	Padmanabhanagar	4 BHK	4689	4.0	488.00	4
13319	Doddathoguru	1 BHK	550	1.0	17.00	1

13246 rows × 6 columns

In [11]:
```
df3['bhk'].unique()
```

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

Out[12]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [13]:
```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

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 convertRangeToSqft(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]:
```
convertRangeToSqft('2166')
```

Out[16]: 2166.0

In [17]:
```
convertRangeToSqft('1500 - 1700')
```

Out[17]: 1600.0

In [18]:
```
df4 = df3.copy()
df4['total_sqft'] = df4['total_sqft'].apply(convertRangeToSqft)
df4
```

Out[18]:

	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
...	...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453.0	4.0	231.00	5
13316	Richards Town	4 BHK	3600.0	5.0	400.00	4
13317	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2
13318	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4
13319	Doddathoguru	1 BHK	550.0	1.0	17.00	1

13246 rows × 6 columns

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

Out[19]:
```
location      Yelahanka
size              4 BHK
total_sqft       2475.0
bath                4.0
price             186.0
bhk                   4
Name: 30, dtype: object
```

In [20]:
```
#feature engineering
```

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 [22]:
```
len(df5['location'].unique())
```

Out[22]:
```
1304
```

In [23]:
```
df5['location'] = df5['location'].apply(lambda x : x.strip())
location_stats = df5.groupby('location')['location'].agg('count').sort_values(as
```

```
location_stats
```

Out[23]:
```
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 [24]:
```
len(location_stats[location_stats <= 10])
```

Out[24]:
```
1052
```

In [25]:
```
location_stats_less_then_10 = location_stats[location_stats<=10]
location_stats_less_then_10
```

Out[25]:
```
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 [26]:
```
len(df5['location'].unique())
```

Out[26]:
```
1293
```

In [27]:
```
df5
```

Out[27]:

	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
...	...	...	...	...	...	...	...
13315	Whitefield	5 Bedroom	3453.0	4.0	231.00	5	6689.834926
13316	Richards Town	4 BHK	3600.0	5.0	400.00	4	11111.111111
13317	Raja Rajeshwari Nagar	2 BHK	1141.0	2.0	60.00	2	5258.545136
13318	Padmanabhanagar	4 BHK	4689.0	4.0	488.00	4	10407.336319
13319	Doddathoguru	1 BHK	550.0	1.0	17.00	1	3090.909091

13246 rows × 7 columns

In [28]:
```
df5['location'] = df5['location'].apply(lambda x : 'Other' if x in location_stat
len(df5['location'].unique())
```

Out[28]: 242

In [29]:
```
df5.head()
```

Out[29]:

	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 [30]:
```
df5[df5['location'] == 'Other']
```

Out[30]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
9	Other	6 Bedroom	1020.0	6.0	370.00	6	36274.509804
18	Other	3 BHK	2770.0	4.0	290.00	3	10469.314079
19	Other	2 BHK	1100.0	2.0	48.00	2	4363.636364
25	Other	3 BHK	1250.0	3.0	56.00	3	4480.000000
42	Other	1 BHK	600.0	1.0	38.00	1	6333.333333
...	...	...	...	...	...	...	...
13291	Other	1 Bedroom	812.0	1.0	26.00	1	3201.970443
13292	Other	3 BHK	1440.0	2.0	63.93	3	4439.583333
13302	Other	2 BHK	1075.0	2.0	48.00	2	4465.116279
13306	Other	4 Bedroom	1200.0	5.0	325.00	4	27083.333333
13316	Other	4 BHK	3600.0	5.0	400.00	4	11111.111111

2881 rows × 7 columns

In [31]:
```
#outlier detection and we'll remove them
```

In [32]:
```
df5[df5['total_sqft'] / df5['bhk'] < 300].head()
```

Out[32]:

	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 [33]:
```
df5.shape
```

Out[33]:
```
(13246, 7)
```

In [34]:
```
df6 = df5[~(df5['total_sqft'] / df5['bhk'] < 300)]
df6.shape
```

Out[34]:
```
(12502, 7)
```

In [35]:
```
df6['price_per_sqft'].describe()
```

Out[35]:
```
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
```

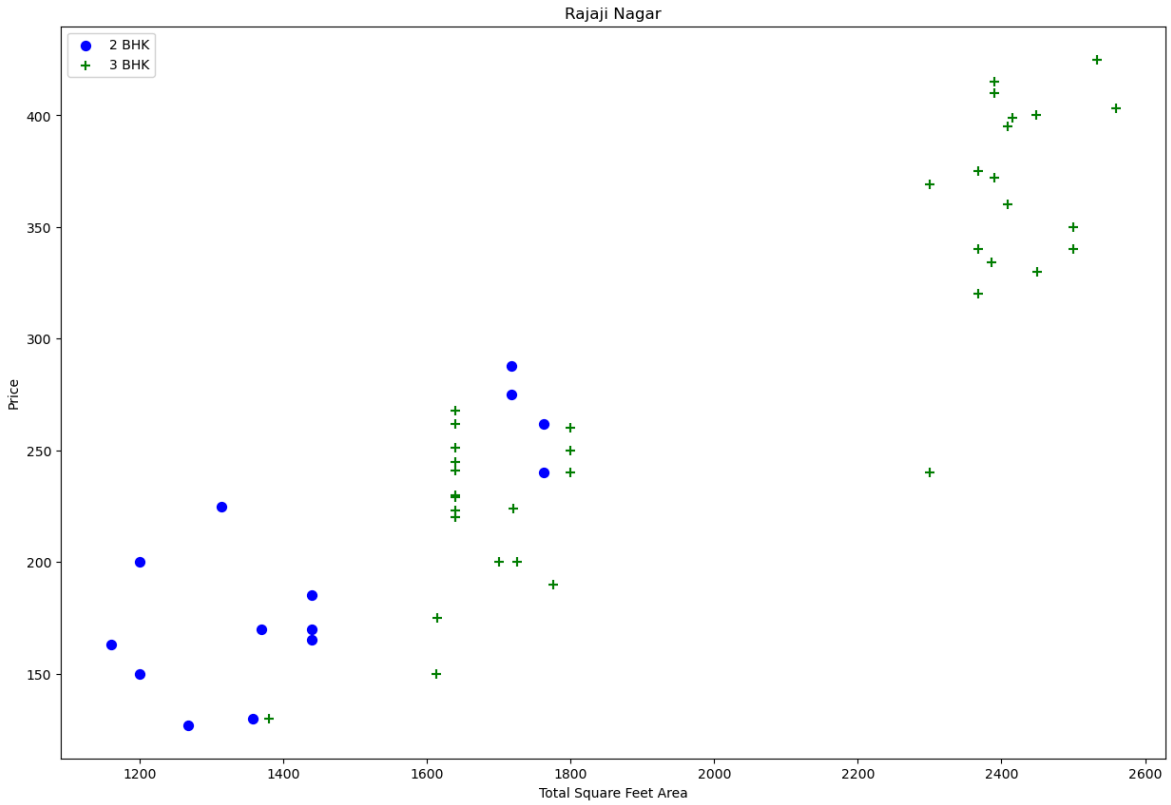
In [36]:
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st = np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft > (m - st)) & (subdf.price_per_
        df_out = pd.concat([df_out, reduced_df], ignore_index = True)
    return df_out

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

Out[36]:
```
(10241, 7)
```

In [37]:
```
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
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price")
    plt.title(location)
    plt.legend()

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



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
    return df.drop(exclude_indices, axis='index')

df8 = remove_bhk_outliers(df7)
df8.shape
```

Out[38]: (7329, 7)

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
5850	Other	6 BHK	11338.0	9.0	1000.0	6	8819.897689
9012	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330

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

Out[40]: (7251, 7)

In [41]:
```
df10 = df9.drop(['size', 'price_per_sqft'], axis = 'columns')
df10
```

Out[41]:

	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
...	...	...	...	...	...
10230	Yeshwanthpur	1195.0	2.0	100.0	2
10231	Yeshwanthpur	1692.0	3.0	108.0	3
10233	Yeshwanthpur	2500.0	5.0	185.0	6
10238	Yeshwanthpur	1855.0	3.0	135.0	3
10239	Yeshwanthpur	1876.0	3.0	160.0	3

7251 rows × 5 columns

In [42]:
```
#one hot encoding or dummies
```

In [43]:
```
dummies = pd.get_dummies(df10['location'])

dummies
```

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
0	True	False	False	False	False	False	False	False	False
1	True	False	False	False	False	False	False	False	False
2	True	False	False	False	False	False	False	False	False
3	True	False	False	False	False	False	False	False	False
4	True	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
10230	False	False	False	False	False	False	False	False	False
10231	False	False	False	False	False	False	False	False	False
10233	False	False	False	False	False	False	False	False	False
10238	False	False	False	False	False	False	False	False	False
10239	False	False	False	False	False	False	False	False	False

7251 rows × 242 columns

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

Out[44]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	Bl Lay
0	1st Block Jayanagar	2850.0	4.0	428.0	4	True	False	False	False	F
1	1st Block Jayanagar	1630.0	3.0	194.0	3	True	False	False	False	F
2	1st Block Jayanagar	1875.0	2.0	235.0	3	True	False	False	False	F
3	1st Block Jayanagar	1200.0	2.0	130.0	3	True	False	False	False	F
4	1st Block Jayanagar	1235.0	2.0	148.0	2	True	False	False	False	F

5 rows × 246 columns

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

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	I N
0	2850.0	4.0	428.0	4	True	False	False	False	False	
1	1630.0	3.0	194.0	3	True	False	False	False	False	
2	1875.0	2.0	235.0	3	True	False	False	False	False	
3	1200.0	2.0	130.0	3	True	False	False	False	False	
4	1235.0	2.0	148.0	2	True	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
10230	1195.0	2.0	100.0	2	False	False	False	False	False	
10231	1692.0	3.0	108.0	3	False	False	False	False	False	
10233	2500.0	5.0	185.0	6	False	False	False	False	False	
10238	1855.0	3.0	135.0	3	False	False	False	False	False	
10239	1876.0	3.0	160.0	3	False	False	False	False	False	

7251 rows × 245 columns

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

Out[46]: (7251, 245)

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

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
0	2850.0	4.0	4	True	False	False	False	False	False
1	1630.0	3.0	3	True	False	False	False	False	False
2	1875.0	2.0	3	True	False	False	False	False	False
3	1200.0	2.0	3	True	False	False	False	False	False
4	1235.0	2.0	2	True	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
10230	1195.0	2.0	2	False	False	False	False	False	False
10231	1692.0	3.0	3	False	False	False	False	False	False
10233	2500.0	5.0	6	False	False	False	False	False	False
10238	1855.0	3.0	3	False	False	False	False	False	False
10239	1876.0	3.0	3	False	False	False	False	False	False

7251 rows × 244 columns

In [48]:
```
y = df12['price']
y
```

Out[48]:
```
0        428.0
1        194.0
2        235.0
3        130.0
4        148.0
         ...
10230    100.0
10231    108.0
10233    185.0
10238    135.0
10239    160.0
Name: price, Length: 7251, dtype: float64
```

In [49]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, random_s
```

In [50]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train, y_train)
lr_clf.score(X_test, y_test)
```

Out[50]:
```
0.8691914452174369
```

In [51]:
```
#lr_clf.predict([['Indira Nagar', 1000, 3, 3]])
```

In [52]:
```
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[52]:
```
array([0.85430675, 0.84187647, 0.84728412, 0.85171729, 0.87168018])
```

In [53]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor

def find_best_model_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
        gs.fit(X,y)
        scores.append({
            'model' : algo_name,
            'best_score' : gs.best_score_,
            'best_params' : gs.best_params_
        })
    return pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])

find_best_model_gridsearchcv(X,y)
```

```
C:\Anaconda\Lib\site-packages\sklearn\model_selection\_validation.py:547: FitFailedWarning:
10 fits failed out of a total of 20.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Anaconda\Lib\site-packages\sklearn\model_selection\_validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Anaconda\Lib\site-packages\sklearn\base.py", line 1467, in wrapper
    estimator._validate_params()
  File "C:\Anaconda\Lib\site-packages\sklearn\base.py", line 666, in _validate_params
    validate_parameter_constraints(
  File "C:\Anaconda\Lib\site-packages\sklearn\utils\_param_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'criterion' parameter of DecisionTreeRegressor must be a str among {'absolute_error', 'friedman_mse', 'squared_error', 'poisson'}. Got 'mse' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Anaconda\Lib\site-packages\sklearn\model_selection\_search.py:1051: UserWarning: One or more of the test scores are non-finite: [       nan        nan 0.69169442 0.68423749]
  warnings.warn(
```

Out[53]:

	model	best_score	best_params
0	linear_regression	0.853373	{}
1	lasso	0.727543	{'alpha': 1, 'selection': 'cyclic'}
2	decision_tree	0.691694	{'criterion': 'friedman_mse', 'splitter': 'best'}

In [54]:
```
#winner is LinearRegression
```

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)
```

```
C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[56]:
```
82.8198103134903
```

In [57]:
```
predict_price('1st Phase JP Nagar', 1000, 2, 3)
```

```
C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[57]: 79.65568746323761

In [58]:
```
predict_price('Indira Nagar', 1000, 2, 2)
```

```
C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[58]: 179.37066882807497

In [59]:
```
predict_price('Indira Nagar', 1000, 3, 3)
```

```
C:\Anaconda\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have v
alid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[59]: 177.68734072817432

In [60]:
```
#export model as pickle file

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

In [61]:
```
import json
columns = {
    'data_columns' : [col.lower() for col in X.columns]
}
with open('columns.json', 'w') as f:
    f.write(json.dumps(columns))
```

In [ ]:
```

```