In [2]:

```
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 [3]:

```
df1= pd.read_csv("E:/Python Project/Bengaluru_House_Data.csv")
```

In [4]:

```
df1.head()
```

Out[4]:

	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 [5]:

```
df1.shape
```

Out[5]:

```
(13320, 9)
```

In [7]:

```
df1.columns
```

Out[7]:

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

In [8]:

```
df1['area_type'].unique()
```

Out[8]:

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

In [9]:

```
df1['area_type'].value_counts()
```

Out[9]:

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

In [10]:

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

Out[10]:

```
(13320, 5)
```

In [11]:

```
df2.isnull().sum()
```

Out[11]:

```
location       1
size          16
total_sqft     0
bath          73
price          0
dtype: int64
```

In [12]:

```
df3=df2.dropna()
```

In [13]:

```
df3.isnull().sum()
```

Out[13]:

```
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64
```

In [14]:

```
df3.shape
```

Out[14]:

```
(13246, 5)
```

In [15]:

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

```
C:\Users\Chinmoy Hazra\AppData\Local\Temp\ipykernel_27048\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#ret
urning-a-view-versus-a-copy
  df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

```
Out[15]: array([ 2,  4,  3,  6,  1,  8,  7,  5, 11,  9, 27, 10, 19, 16, 43, 14, 12,
                13, 18], dtype=int64)
```

```
In [18]: def is_float(x):
             try:
                 float(x)
             except:
                 return False
             return True
```

```
In [22]: df4 = df3[~df3['total_sqft'].apply(is_float)]
```

```
In [24]: df4['total_sqft'].unique()
```

```
Out[24]: array(['2100 - 2850', '3067 - 8156', '1042 - 1105', '1145 - 1340',
                '1015 - 1540', '34.46Sq. Meter', '1195 - 1440', '4125Perch',
                '1120 - 1145', '3090 - 5002', '1160 - 1195', '1000Sq. Meter',
                '1115 - 1130', '1100Sq. Yards', '520 - 645', '1000 - 1285',
                '650 - 665', '633 - 666', '5.31Acres', '30Acres', '1445 - 1455',
                '884 - 1116', '850 - 1093', '716Sq. Meter', '547.34 - 827.31',
                '580 - 650', '3425 - 3435', '1804 - 2273', '3630 - 3800',
                '4000 - 5249', '1500Sq. Meter', '142.61Sq. Meter', '1574Sq. Yards',
                '1250 - 1305', '670 - 980', '1005.03 - 1252.49', '1004 - 1204',
                '361.33Sq. Yards', '645 - 936', '2710 - 3360', '2830 - 2882',
                '596 - 804', '1255 - 1863', '1300 - 1405', '117Sq. Yards',
                '934 - 1437', '980 - 1030', '2249.81 - 4112.19', '1070 - 1315',
                '3040Sq. Meter', '500Sq. Yards', '2806 - 3019', '613 - 648',
                '704 - 730', '1210 - 1477', '3369 - 3464', '1125 - 1500',
                '167Sq. Meter', '1076 - 1199', '381 - 535', '524 - 894',
                '540 - 670', '315Sq. Yards', '2725 - 3250', '888 - 1290',
                '660 - 700', '385 - 440', '770 - 841', '3Cents', '188.89Sq. Yards',
                '1469 - 1766', '204Sq. Meter', '1255 - 1350', '870 - 1080',
                '45Sq. Yards', '133.3Sq. Yards', '2580 - 2591', '2563 - 2733',
                '605 - 624', '1349 - 3324', '78.03Sq. Meter', '3300 - 3335',
                '1180 - 1630', '1365 - 1700', '122Sq. Yards', '84.53Sq. Meter',
                '2.09Acres', '981 - 1249', '1565 - 1595', '24Guntha',
                '1270 - 1275', '840 - 1010', '697Sq. Meter', '655 - 742',
                '1408 - 1455', '942 - 1117', '598 - 958', '1500Cents',
                '132Sq. Yards', '1010 - 1300', '2Acres', '1450 - 1950',
                '1100Sq. Meter', '15Acres', '763 - 805', '3307 - 3464',
                '1.26Acres', '620 - 934', '2462 - 2467', '540 - 740',
                '3508 - 4201', '4900 - 4940', '755 - 770', '664 - 722',
                '151.11Sq. Yards', '596 - 861', '615 - 985', '540 - 565',
                '750 - 800', '1660 - 1805', '1079 - 1183', '2800 - 2870',
                '1230 - 1290', '943 - 1220', '2041 - 2090', '527 - 639',
                '1Grounds', '1160 - 1315', '706 - 716', '2940Sq. Yards',
                '45.06Sq. Meter', '799 - 803', '2470 - 2790', '783 - 943',
                '4500 - 5540', '1255 - 1375', '610 - 615', '854 - 960',
                '2650 - 2990', '1.25Acres', '86.72Sq. Meter', '1230 - 1490',
                '660 - 780', '1150 - 1194', '684 - 810', '1510 - 1670',
                '1550 - 1590', '1235 - 1410', '38Guntha', '929 - 1078',
                '2150 - 2225', '1520 - 1759', '629 - 1026', '1215 - 1495',
                '6Acres', '1140 - 1250', '2400 - 2600', '1052 - 1322',
                '5666 - 5669', '712 - 938', '1783 - 1878', '120Sq. Yards',
                '24Sq. Meter', '2528 - 3188', '650 - 760', '1400 - 1421',
                '4000 - 4450', '142.84Sq. Meter', '300Sq. Yards', '1437 - 1629',
                '850 - 1060', '1200 - 1470', '1133 - 1384'], dtype=object)
```

```
In [25]: import re
         def convert_sqft_to_num(x):
             tokens= x.split('-')
             if len(tokens) == 2:
                 return (float(tokens[0])+float(tokens[1]))/2
             try:
                 head = re.split("[^0-9]",x)
                 tail = "".join(re.split("[^a-zA-Z]",x))


                 if tail == "Perch":
                     if head[1] != "":
                         return round(((float(head[0]) * 272.25) + (float(head[1]) * 2.7225)),2)
                     else:
                         return round((float(head[0]) * 272.25),2)
                 if tail == "SqMeter":
                     if head[1]!="":
                        return round(((float(head[0])*10.7639104)+(float(head[1])*0.107639104)),2)
                     else:
                         return round((float(head[0])*10.7639104),2)
                 if tail == "SqYards":
                     if head[1]!="":
                        return round(((float(head[0])*9)+(float(head[1])*0.09)),2)
                     else:
                         return round((float(head[0])*9),2)

                 if tail == "Guntha":
                     if head[1]!="":
                        return round(((float(head[0])*1089)+(float(head[1])*10.89)),2)
```

```
            else:
                return round((float(head[0])*1089),2)
        if tail == "Acres":
            if head[1]!="":
               return round(((float(head[0])*43560)+(float(head[1])*435.60)),2)
            else:
                return round((float(head[0])*43560),2)

        if tail == "Cents":
            if head[1]!="":
               return round(((float(head[0])*435.6)+(float(head[1])*4.3560)),2)
            else:
                return round((float(head[0])*435.60),2)

        if tail == "Cents":
            if head[1] != "":
                return round(((float(head[0]) * 435.6) + (float(head[1]) * 4.3560)), 2)
            else:
                return round((float(head[0]) * 435.60), 2)

        if tail == "":
            return float(head[0])
    except:
        return None
```

In [26]:

```
df3['total_sqft']= df3['total_sqft'].apply(convert_sqft_to_num)
```

```
C:\Users\Chinmoy Hazra\AppData\Local\Temp\ipykernel_27048\3436377115.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#ret
urning-a-view-versus-a-copy
  df3['total_sqft']= df3['total_sqft'].apply(convert_sqft_to_num)
```

In [27]:

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

Out[27]:

```
array([1056. , 2600. , 1440. , ..., 1258.5,  774. , 4689. ])
```

In [28]:

```
df5 = df3.copy()
```

In [30]:

```
df5['price_per_sqft']=(df5['price']*100000)/df5['total_sqft']
```

In [31]:

```
df5.head()
```

Out[31]:

	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 [32]:

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

Out[32]:

```
count    1.324500e+04
mean     7.915743e+03
std      1.065492e+05
min      2.257423e+00
25%      4.262295e+03
50%      5.433830e+03
75%      7.317073e+03
max      1.200000e+07
Name: price_per_sqft, dtype: float64
```

In [33]:

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

Out[33]:

```
Whitefield                            535
Sarjapur  Road                        392
Electronic City                       304
Kanakpura Road                        266
Thanisandra                           236
                                     ... 
Rajanna Layout                          1
Kengeri Satellite Town ( BDA SITE)      1
Lakshmipura Vidyaanyapura               1
Malur Hosur Road                        1
Abshot Layout                           1
Name: location, Length: 1294, dtype: int64
```

In [34]:

```
location_stats.values.sum()
```

Out[34]:

```
13246
```

In [36]:

```
len(location_stats[location_stats>10])
```

Out[36]:

```
241
```

In [37]:

```
len(location_stats)
```

Out[37]:

```
1294
```

In [38]:

```
len(location_stats[location_stats<=10])
```

Out[38]:

```
1053
```

In [40]:

```
location_stats_less_than_10 = location_stats[location_stats<=10]
location_stats_less_than_10
```

Out[40]:

```
BTM 1st Stage                          10
Ganga Nagar                            10
Gunjur Palya                           10
Naganathapura                          10
Dodsworth Layout                       10
                                       ..
Rajanna Layout                          1
Kengeri Satellite Town ( BDA SITE)      1
Lakshmipura Vidyaanyapura               1
Malur Hosur Road                        1
Abshot Layout                           1
Name: location, Length: 1053, dtype: int64
```

In [41]:

```
len(df5.location.unique())
```

Out[41]:

```
1294
```

In [42]:

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

Out[42]:

```
242
```

In [43]:

```
df5.head(10)
```

Out[43]:

	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 [45]:

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

Out[45]:

	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 [46]:

```
df5.shape
```

Out[46]:

```
(13246, 7)
```

In [47]:

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

Out[47]:

```
(12498, 7)
```

In [49]:

```
df6.price_per_sqft.describe()
```

Out[49]:

```
count     12497.000000
mean       6299.256398
std        4169.126919
min           2.257423
25%        4203.984064
50%        5291.005291
75%        6916.666667
max      176470.588235
Name: price_per_sqft, dtype: float64
```

In [50]:

```
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
```

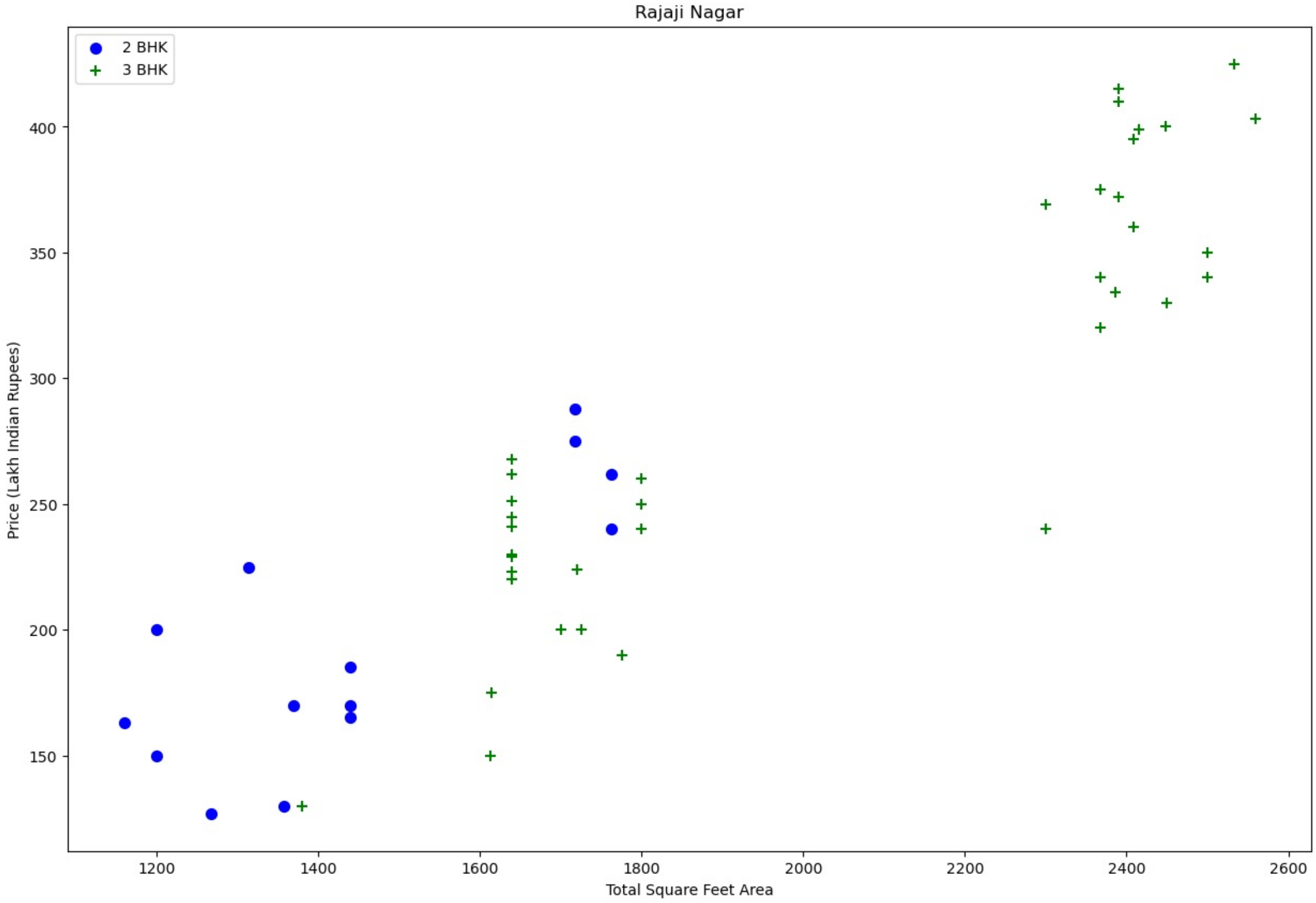
Out[50]:

```
(10268, 7)
```

In [51]:

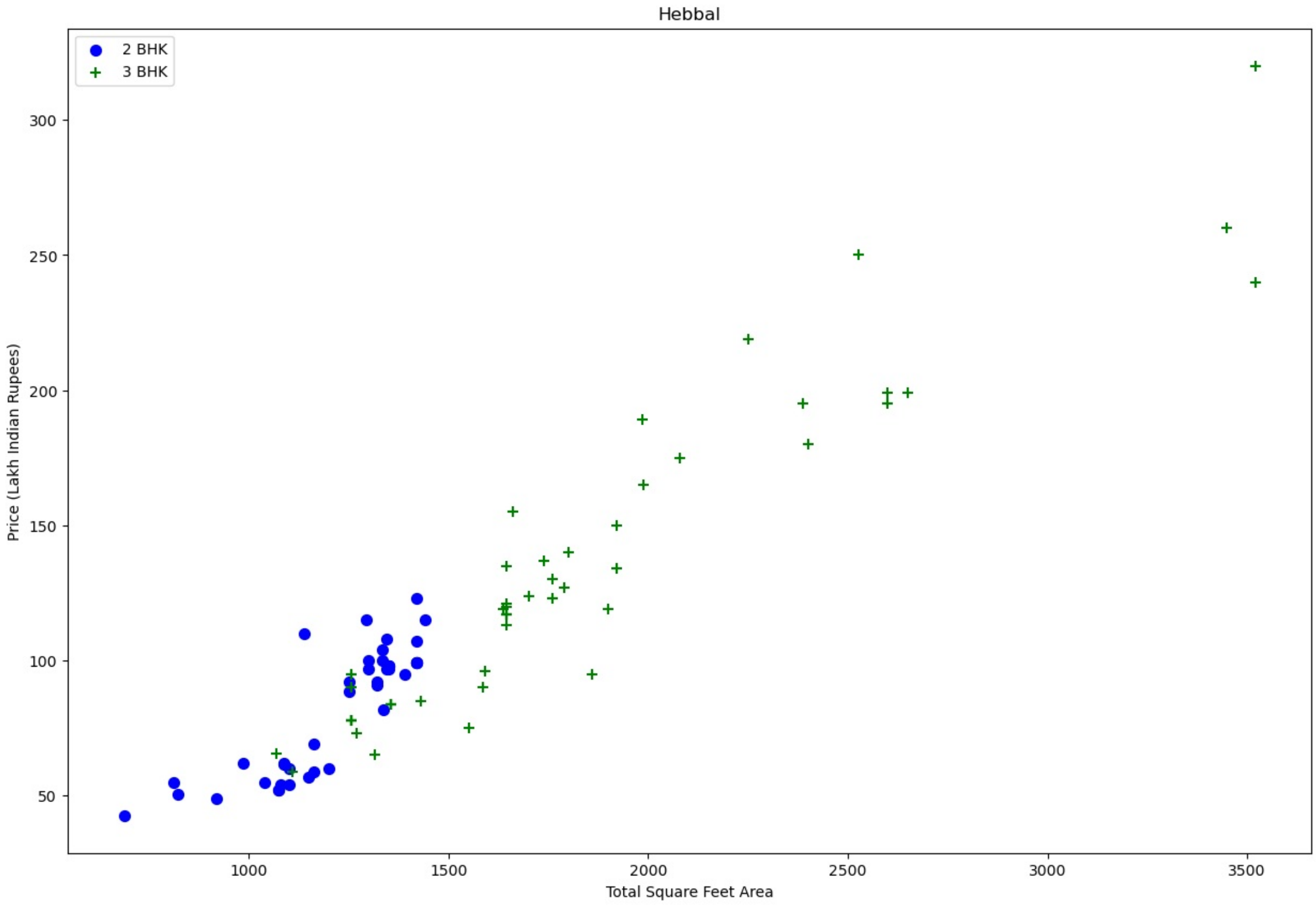
```
def plot_scatter_chart(df,location):
    bhk2 = df[(df.location==location) & (df.bhk==2)]
    bhk3 = df[(df.location==location) & (df.bhk==3)]
    matplotlib.rcParams['figure.figsize'] = (15,10)
    plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 BHK', s=50)
    plt.scatter(bhk3.total_sqft,bhk3.price,marker='+', 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 [52]:

```
plot_scatter_chart(df7,"Hebbal")
```



In [53]:

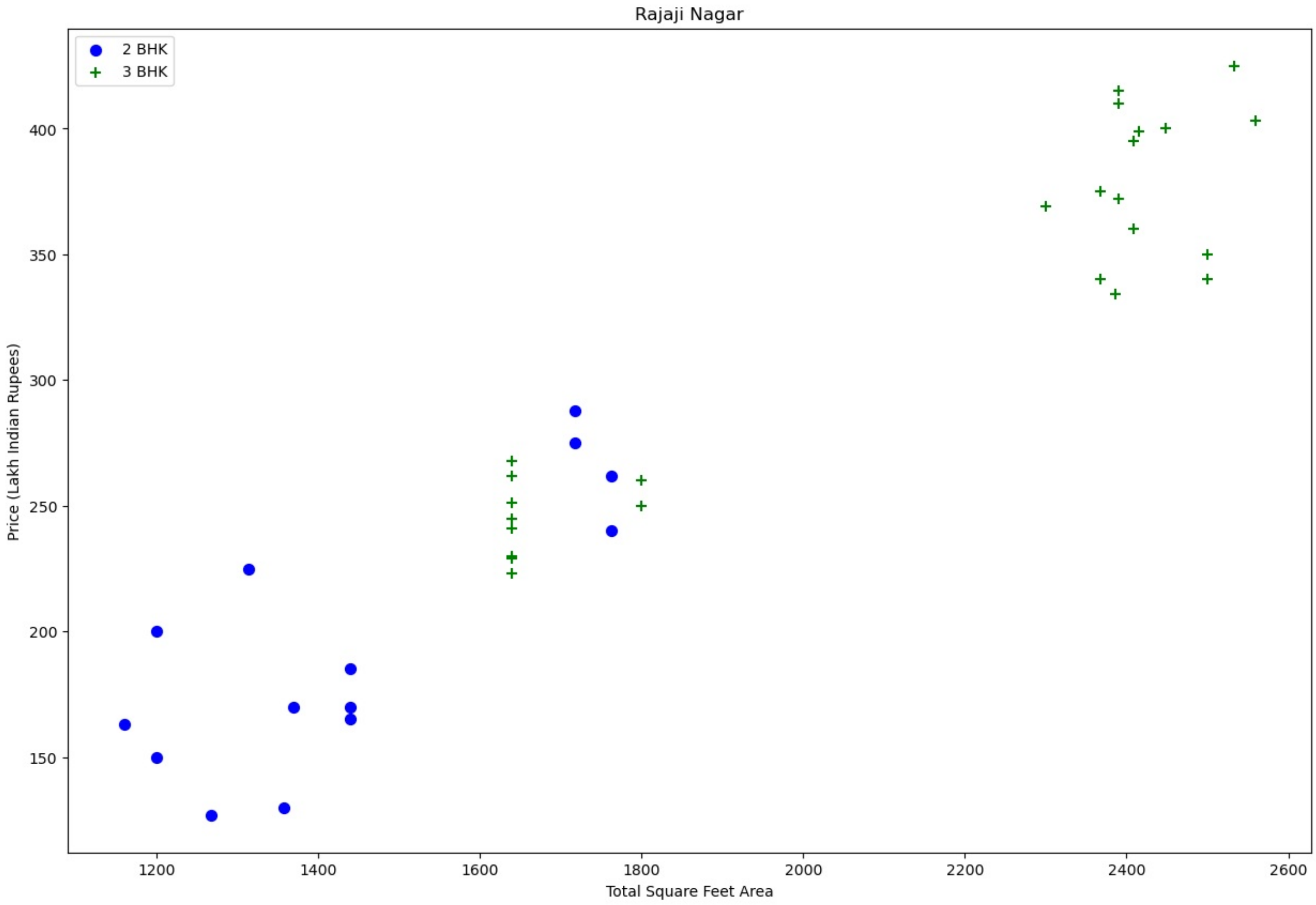
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
          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

Out[53]:

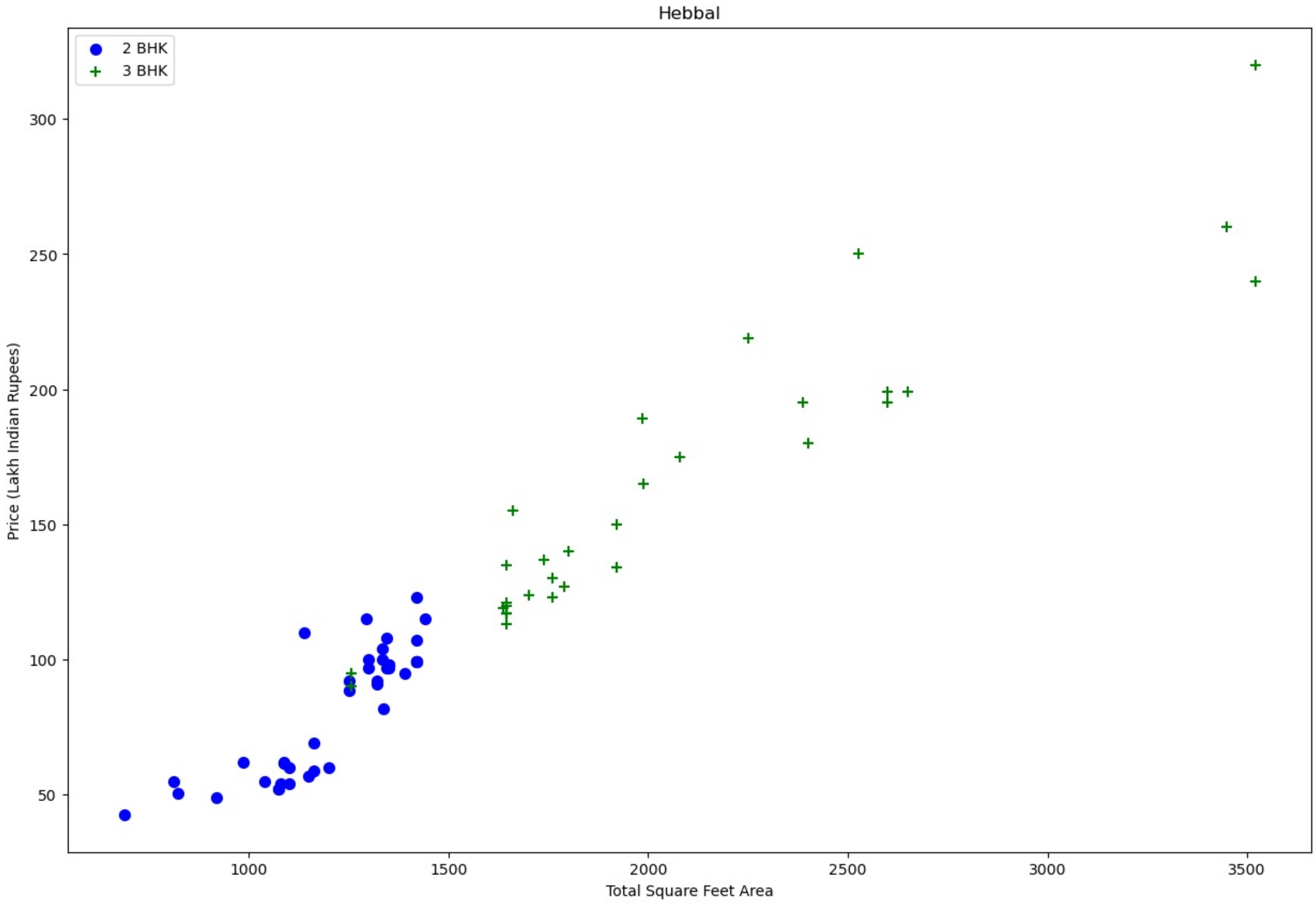
```
(7344, 7)
```

In [54]:

```
plot_scatter_chart(df8,"Rajaji Nagar")
```

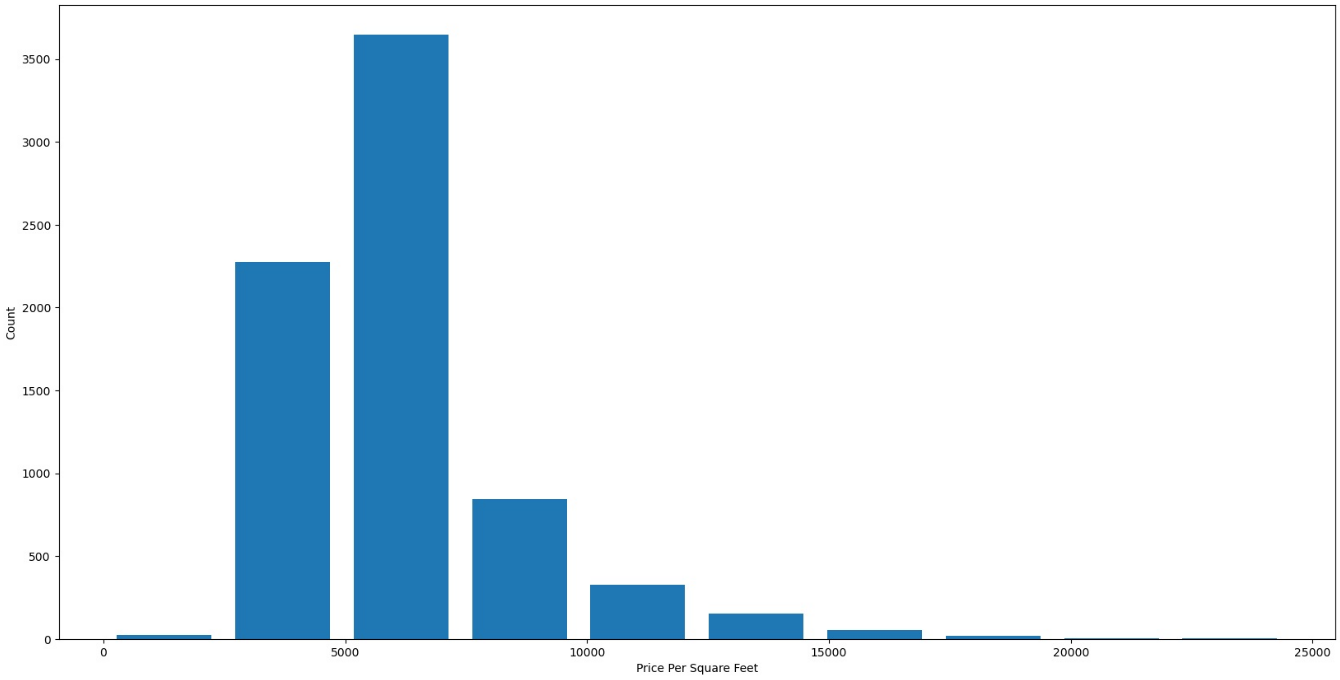


In [56]:
```
plot_scatter_chart(df8,"Hebbal")
```



In [57]:
```
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[57]:
```
Text(0, 0.5, 'Count')
```



In [58]:

```
df8.bath.unique()
```

Out[58]:

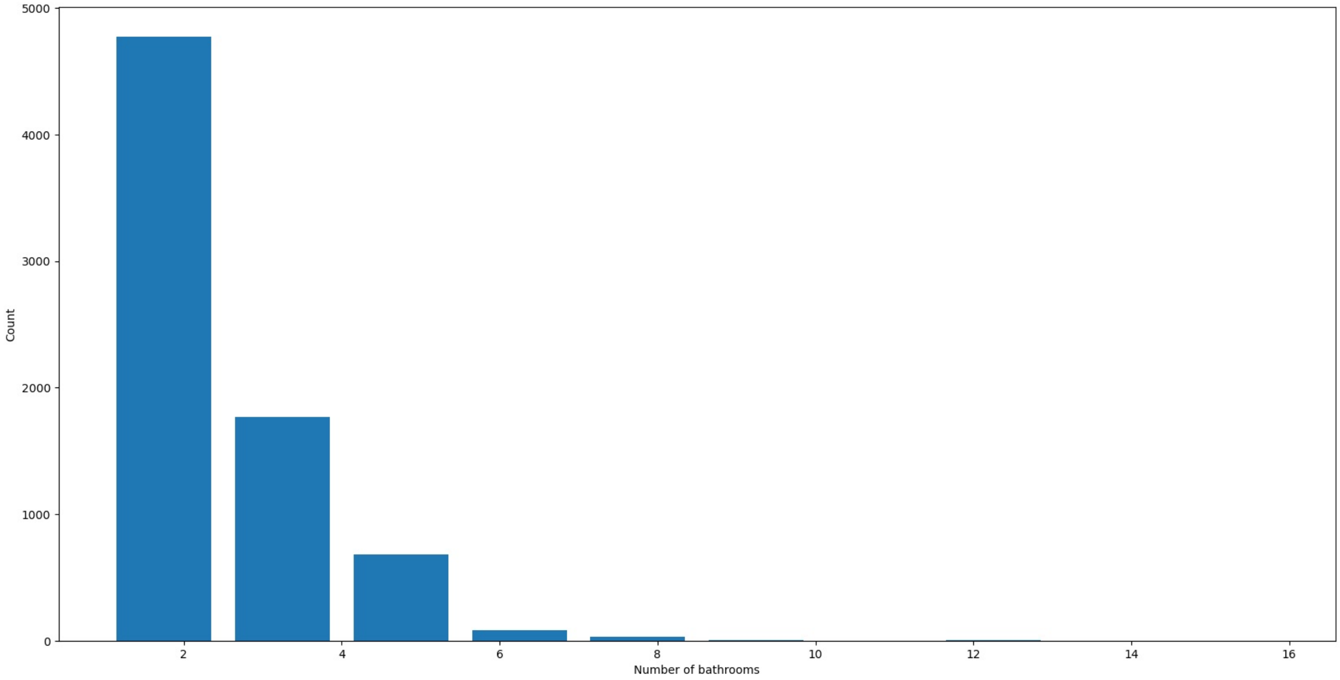
```
array([ 4.,  3.,  2.,  5.,  1.,  8.,  6.,  7.,  9., 12., 16., 13.])
```

In [59]:

```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("Number of bathrooms")
plt.ylabel("Count")
```

Out[59]:

```
Text(0, 0.5, 'Count')
```



In [60]:

```
df8[df8.bath>10]
```

Out[60]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
5291	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8507	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8596	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9332	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9664	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

In [61]:

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

Out[61]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1632	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5252	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6727	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8431	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

In [62]:

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

Out[62]:

```
(7266, 7)
```

In [64]:

```
df9.head(2)
```

Out[64]:

	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 [65]:

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

Out[65]:

	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 [66]:

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

Out[66]:

	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	White
0	1	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	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	

3 rows × 242 columns

In [67]:

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

Out[67]:

	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	Vitt
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	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	
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	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	
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0	...	0	0	0	

5 rows × 246 columns

In [69]:

```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

Out[69]:

	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	Vittasa
0	2850.0	4.0	428.0	4	1	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	

2 rows × 245 columns

In [71]:
```
df12.shape
```

Out[71]:
```
(7266, 245)
```

In [72]:
```
X = df12.drop(['price'],axis='columns')
X.head(3)
```

Out[72]:

	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	Vittas
0	2850.0	4.0	4	1	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	
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	

3 rows × 244 columns

In [73]:
```
X.shape
```

Out[73]:
```
(7266, 244)
```

In [75]:
```
y = df12.price
y.head(3)
```

Out[75]:
```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

In [76]:
```
len(y)
```

Out[76]:
```
7266
```

In [90]:
```
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[90]:
```
array([0.63989066, 0.62381148, 0.62305156, 0.60649799, 0.66420308])
```

In [91]:
```
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=False)
        gs.fit(X,y)
        scores.append({
            'model': algo_name,
            'best_score': gs.best_score_,
            'best_params': gs.best_params_
        })

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

find_best_model_using_gridsearchcv(X,y)
```

```
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarning: '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 stage. To reproduce the pr
evious 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 pipel
ine as follows:

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


  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarning: '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 stage. To reproduce the pr
evious 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 pipel
ine as follows:

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


  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarning: '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 stage. To reproduce the pr
evious 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 pipel
ine as follows:

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


  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarning: '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 stage. To reproduce the pr
evious 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 pipel
ine as follows:

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


  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarning: '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 stage. To reproduce the pr
evious 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 pipel
ine as follows:

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


  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarning: '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 behavior of this estimator is to not do any normalization. If normal
ization is needed please use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarning: '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 behavior of this estimator is to not do any normalization. If normal
ization is needed please use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarning: '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 behavior of this estimator is to not do any normalization. If normal
ization is needed please use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarning: '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 behavior of this estimator is to not do any normalization. If normal
ization is needed please use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:148: FutureWarning: '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 behavior of this estimator is to not do any normalization. If normal
ization is needed please use sklearn.preprocessing.StandardScaler instead.
  warnings.warn(
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\linear_model\_base.py:141: FutureWarning: '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 stage. To reproduce the pr
evious 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 pipel
ine as follows:

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


  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
C:\Users\Chinmoy Hazra\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 `criterion='squared_error'` which is equivalent.
  warnings.warn(
```

Out[91]:

	model	best_score	best_params
0	linear_regression	0.631491	{'normalize': True}
1	lasso	0.418363	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.706191	{'criterion': 'friedman_mse', 'splitter': 'best'}

In [86]:

```
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 [92]:

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

```
C:\Users\Chinmoy Hazra\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feat
ure names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[92]:

```
99.14792843683288
```

In [93]:

```
def run_latex(self, filename):
    """Run xelatex self.latex_count times."""

    def log_error(command, out):
        self.log.critical(u"%s failed: %s\n%s", command[0], command, out)

    return self.run_command(self.latex_command, filename,
        self.latex_count, log_error)
```

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