

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

In [2]: df=pd.read_csv('D:\Machine learning\Bangalore House prediction\Bengaluru_House_Data.csv')

In [3]: df.head()

Out[3]:
   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 Trinpathi    4 Bedroom  Theampamp      2600   5.0    3.0    120.00
2    Built-up Area    Ready To Move      Uttarahalli      3 BHK  Na/N      1440   3.0    1.0    62.00
3  Super built-up Area    Ready To Move      Lingadheeranahalli      3 BHK  Solevere      1521   3.0    1.0    95.00
4  Super built-up Area    Ready To Move      Kothanur      2 BHK  Na/N      1200   2.0    1.0    51.00

In [4]: df1.shape
Out[4]: (13320, 9)

In [5]: df1.columns
Out[5]: Index(['area_type', 'availability', 'location', 'size', 'society', 'total_sqft', 'bath', 'balcony', 'price'],
      dtype='object')

In [6]: df1.groupby('area_type')['area_type'].agg('count')
Out[6]:
area_type
Built-up Area    8790
Carpet Area      87
Plot Area        2025
Super built-up Area    8790
Name: area_type, dtype: int64

In [7]: df1.area_type.unique()
Out[7]: array(['Super built-up Area', 'Plot Area', 'Built-up Area',
      'Carpet Area'], dtype=object)

In [8]: df1.area_type.value_counts()
Out[8]:
Super built-up Area    8790
Built-up Area          2418
Plot Area              2025
Carpet Area            87
Name: area_type, dtype: int64

In [9]: df2=df1.drop(columns=['area_type','availability','society','balcony'],axis=1)

In [10]: df2.head()
Out[10]:
   location      size  total_sqft  bath  price
0  Electronic City Phase II    2 BHK      1056   2.0    39.07
1    Chikka Trinpathi    4 Bedroom      2600   5.0    120.00
2    Uttarahalli      3 BHK      1440   3.0    62.00
3  Lingadheeranahalli      3 BHK      1521   3.0    95.00
4    Kothanur      2 BHK      1200   2.0    51.00

Data cleaning : Handle Na Values

In [11]: df2.isnull().sum()
Out[11]:
location      1
size          0
total_sqft    0
bath          73
price         0
dtype: object

In [12]: df2.shape
Out[12]: (13320, 5)

In [13]: df3=df2.dropna()
Out[13]:
location      0
size          0
total_sqft    0
bath          0
price         0
dtype: int64

In [14]: df3['size'].unique()
Out[14]: array(['12 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', '12 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)

Feature engineering

In [15]: df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
df3['bhk'].unique()

<ipython-input-15-1d3ab9857b13>:1: SettingWithCopyWarning:
A value is being 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[15]:
df3
0      2
1      4
2      3
3      3
4      2
dtype: int64

In [16]: df3[df3.bhk>20]
Out[16]:
   location      size  total_sqft  bath  price  bhk
1718  Electronic City Phase II    2 BHK      8000   2.0    39.07    2
4684  Munnakkal    43 Bedroom      2400   40.0    66.00    43

In [17]:
def isfloat(x):
    try:
        float(x)
    except:
        return False
    return True

df3[df3['total_sqft']!=df3.apply(isfloat,axis=1).head(20)]

Out[18]:
   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
186   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   Anekal     9 Bedroom      4125Perch   9.0    265.000    9
661   Yelahanka      2 BHK      1120 - 1145   2.0    48.130    2
672   Bellandur     4 Bedroom      3090 - 5002   4.0    445.000    4
772  Banashankari Stage VI      2 BHK      1160 - 1195   2.0    59.935    2
775   Basavanagara      1 BHK      1000Sq Meter   1.0    93.000    1
850   Bannerghatta Road      2 BHK      1115 - 1130   2.0    58.935    2
872   Singapura Village      2 BHK      1100Sq Yards   2.0    45.000    2
886   Chandapura      1 BHK      520 - 645   1.0    15.135    1
927   Thanisandra      2 BHK      1000 - 1285   2.0    43.415    2
959   Kammasandra      1 BHK      650 - 665   1.0    18.410    1
990   Sarjapur      1 BHK      633 - 666   1.0    17.535    1
1019  Marathi Layout    1 Bedroom      5.31Acres   1.0    110.000    1
1086  Narasapura      2 Bedroom      30Acres   2.0    29.500    2

In [19]:
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens)==2:
        return (float(tokens[0])+float(tokens[1])/2)
    else:
        return None

convert_sqft_to_num('5.31Acres')

In [20]: df4 = df3.copy()

In [21]: df4['total_sqft'] = df4['total_sqft'].apply(convert_sqft_to_num)

In [22]: df4.isna().sum()
Out[22]:
location      0
size          0
total_sqft    0
bath          0
price         0
bhk          0
dtype: int64

In [23]: df4=df4.dropna()
Out[23]:
location      0
size          0
total_sqft    0
bath          0
price         0
bhk          0
dtype: int64

In [24]: df4.isna().sum()
Out[24]:
location      0
size          0
total_sqft    0
bath          0
price         0
bhk          0
dtype: int64

In [25]: df4.head(20)
Out[25]:
   location      size  total_sqft  bath  price  bhk
0  Electronic City Phase II    2 BHK      1056.0   2.0    39.07    2
1    Chikka Trinpathi    4 Bedroom      2600.0   5.0    120.00    4
2    Uttarahalli      3 BHK      1440.0   3.0    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
5    Whitefield      2 BHK      1170.0   2.0    38.00    2
6    Old Airport Road      4 BHK      3300.0   4.0    60.00    4
7    Marathahalli      3 BHK      1310.0   3.0    63.25    3
8    Gandhinagar     6 Bedroom      1020.0   6.0    37.00    6
9    Whitefield      3 BHK      1800.0   3.0    70.00    3
10   Whitefield     4 Bedroom      2785.0   5.0    295.00    4
11   7th Phase JP Nagar      2 BHK      1000.0   2.0    38.00    2
12   Gottigere      2 BHK      1100.0   2.0    40.00    2
13   Sarjapur      3 Bedroom      2250.0   3.0    148.00    3
14   Mysore Road      2 BHK      1175.0   2.0    73.50    2
15   Bisuvanahalli      3 BHK      1180.0   3.0    48.00    3
16   Raja Rajeshwari Nagar      3 BHK      1540.0   3.0    60.00    3
17   Ramakrishnappa Layout      3 BHK      2770.0   4.0    290.00    3
18   Manayata Tech Park      2 BHK      1100.0   2.0    48.00    2

In [26]: df4.loc[30]
Out[26]:
location      Yelahanka
size          4 BHK
total_sqft    2475
bath          4
price         186
bhk           4
Name: 30, dtype: object

In [27]: df4.head(4)
Out[27]:
   location      size  total_sqft  bath  price  bhk
0  Electronic City Phase II    2 BHK      1056.0   2.0    39.07    2
1    Chikka Trinpathi    4 Bedroom      2600.0   5.0    120.00    4
2    Uttarahalli      3 BHK      1440.0   3.0    62.00    3
3  Lingadheeranahalli      3 BHK      1521.0   3.0    95.00    3

In [28]: df5=df4.copy()

In [29]: df5['price_per_sqft']=df5['price']*100000/df5['total_sqft']
df5

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 Trinpathi    4 Bedroom      2600.0   5.0    120.00    4    4615.384615
2    Uttarahalli      3 BHK      1440.0   3.0    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    150.00    4    4615.384615
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  Doddanahallur      1 BHK      550.0   1.0    17.00    1    3090.909091

13200 rows x 7 columns

Examine locations which is a categorical variable. We need to apply dimensionality reduction technique here to reduce number of locations

In [30]: df5.location.nunique()
Out[30]: 1298

In [31]: df5.location=df5.location.apply(lambda x: x.strip())
location_stats=df5['location'].value_counts(ascending=False)
location_stats.head(7)

Out[31]:
Whitefield      533
Sarjapur Road    392
Electronic City    304
Kanakpura Road    264
Thanisandra      235
...
9th Phase JP Nagar      43
Subramanyapura      43
Vittasandra      43
Vittasandra Apsara      42
Kanakpura      42
Name: location, Length: 70, dtype: int64

In [32]: location_stats.values.sum()
Out[32]: 13200

In [33]: len(location_stats[location_stats>10])
Out[33]: 240

In [34]: len(location_stats[location_stats>40])
Out[34]: 1033

In [35]: len(location_stats)
Out[35]: 1287

Dimensionality Reduction

Any location having less than 10 data points should be tagged as 'other' location. This way number of categories can be reduced by huge amount. Later on when we do one hot encoding, it will help us with having fewer dummy columns

In [36]: location_stats_less_than_10=location_stats[location_stats<=10]

In [37]: location_stats_less_than_10
Out[37]:
Nagappa Reddy Layout      10
Nagadevanahalli          10
Thyagaraja Nagar          10
1st Block Koramangala     10
1 Ramamurthy Nagar        1
Chokkiahalli              1
Ramdenu Nagar             1
Manonayanaapalya          1
Electronic city phase 1    1
Name: location, Length: 1047, dtype: int64

In [38]: len(df5.location.unique())
Out[38]: 1287

In [39]: df5.location=df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
len(df5.location.unique())

Out[39]: 241

In [40]: df5.head()
Out[40]:
   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 Trinpathi    4 Bedroom      2600.0   5.0    120.00    4    4615.384615
2    Uttarahalli      3 BHK      1440.0   3.0    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

Outlier Removal Using Business Logic

As a data scientist when you have a conversation with your business manager (who has expertise in real estate), he will tell you that normally square ft per bedroom is 300 (i.e. 2 BHK apartment is minimum 600 sqft. If you have for example 400 sqft apartment with 2 bhk then that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping our minimum threshold per bhk to be 300 sqft

In [41]: df5[df5.total_sqft/df5.bhk<300]
Out[41]:
   location      size  total_sqft  bath  price  bhk  price_per_sqft
9      other     6 Bedroom      1020.0   6.0    37.00    6    3627.4509804
48     HSR Layout     8 Bedroom      600.0   9.0    20.00    8    3627.4509804
55   Murugesapalya     6 Bedroom      1407.0   4.0    150.00    6    10660.980810
68   Devarachikkanahalli     8 Bedroom      1350.0   7.0    85.00    8    6296.296296
70      other     3 Bedroom      500.0   3.0    100.00    3    2000.000000
...
...
13277  other     7 Bedroom      1400.0   7.0    218.00    7    15571.428571
13278  other     6 Bedroom      1200.0   6.0    130.00    6    10833.333333
13281  Margondanahalli     5 Bedroom      1375.0   5.0    125.00    5    9090.909091
13303  Vidyanagarapalya     5 Bedroom      775.0   5.0    70.00    5    9043.927649
13311  Ramamurthy Nagar     7 Bedroom      1500.0   7.0    250.00    7    16666.666667

744 rows x 7 columns

In [42]: df5.shape
Out[42]: (13200, 7)

In [43]: df6=df5[df5.total_sqft/df5.bhk<300]
df6.shape
Out[43]: (12456, 7)

Outlier Removal Using Standard Deviation and Mean

In [44]: df6.price_per_sqft.describe()
Out[44]:
count      12456.000000
mean       6308.502824
std        4168.127339
min         267.829813
5%         4210.526316
50%        5294.117647
95%        69316.666667
max       176470.588525
Name: price_per_sqft, dtype: float64

Here we find that min price per sqft is 267 rs/sqft whereas max is 1200000, this shows a wide variation in property prices. We should remove outliers per location using mean and one standard deviation

In [45]:
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key, subdf in df.groupby('location'):
        mpp = mean(subdf.price_per_sqft)
        std = std(subdf.price_per_sqft)
        reduced_out = subdf[(subdf.price_per_sqft>=m-std) & (subdf.price_per_sqft<=m+std)]
        df_out = pd.concat([df_out, reduced_out], ignore_index=True)
    return df_out
df7=remove_pps_outliers(df6)
df7.shape
Out[45]: (10242, 7)

Lets check if for a given location how does the 2 BHK and 3 BHK property prices look like

In [46]:
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,color='green',label='3 BHK', s =50)
    plt.xlabel("Total Square Feet Area")
    plt.ylabel("Price (Lakh Indian Rupees)")
    plt.title(location)
    plt.legend()

In [47]: plot_scatter_chart(df7,"Rajajinagar")
Out[47]:
In [48]: plot_scatter_chart(df7,"Hebbal")
Out[48]:

In [49]:
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'):
            'mean': np.mean(bhk_df.price_per_sqft),
            'std': np.std(bhk_df.price_per_sqft),
            'count': len(bhk_df)
        )
        for 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']-1)*std)].index)
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
df8.shape
Out[49]: (7317, 7)

In [50]: plot_scatter_chart(df8,"Rajajinagar")
Out[50]:

In [51]: plot_scatter_chart(df8,"Hebbal")
Out[51]:

In [52]:
import matplotlib
matplotlib.rcParams['figure.figsize']=(20,10)
plt.hist(df8.price_per_sqft,width=50)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")

Out[52]:
Text(0.5, 0, 'Count')

In [53]:
df9=df8[df8.bath>1]

In [54]:
df9=df8[df8.bath>1]

In [55]:
df9=df8[df8.bath>1]

In [56]:
df9=df8[df8.bath>1]

In [57]:
df9=df8[df8.bath>1]

In [58]:
df9=df8[df8.bath>1]

In [59]:
df9=df8[df8.bath>1]

In [60]:
df9=df8[df8.bath>1]

In [61]:
df9=df8[df8.bath>1]

In [62]:
df9=df8[df8.bath>1]

In [63]:
df9=df8[df8.bath>1]

In [64]:
df9=df8[df8.bath>1]

In [65]:
df9=df8[df8.bath>1]

In [66]:
df9=df8[df8.bath>1]

In [67]:
df9=df8[df8.bath>1]

In [68]:
df9=df8[df8.bath>1]

In [69]:
df9=df8[df8.bath>1]

In [70]:
df9=df8[df8.bath>1]

In [71]:
df9=df8[df8.bath>1]

In [72]:
df9=df8[df8.bath>1]

In [73]:
df9=df8[df8.bath>1]

In [74]:
df9=df8[df8.bath>1]

In [75]:
df9=df8[df8.bath>1]

In [76]:
df9=df8[df8.bath>1]

In [77]:
df9=df8[df8.bath>1]

In [78]:
df9=df8[df8.bath>1]

In [79]:
df9=df8[df8.bath>1]

In [80]:
df9=df8[df8.bath>1]

In [81]:
df9=df8[df8.bath>1]

In [82]:
df9=df8[df8.bath>1]

In [83]:
df9=df8[df8.bath>1]

In [84]:
df9=df8[df8.bath>1]

In [85]:
df9=df8[df8.bath>1]

In [86]:
df9=df8[df8.bath>1]

In [87]:
df9=df8[df8.bath>1]

In [88]:
df9=df8[df8.bath>1]

In [89]:
df9=df8[df8.bath>1]

In [90]:
df9=df8[df8.bath>1]

In [91]:
df9=df8[df8.bath>1]

In [92]:
df9=df8[df8.bath>1]

In [93]:
df9=df8[df8.bath>1]

In [94]:
df9=df8[df8.bath>1]

In [95]:
df9=df8[df8.bath>1]

In [96]:
df9=df8[df8.bath>1]

In [97]:
df9=df8[df8.bath>1]

In [98]:
df9=df8[df8.bath>1]

In [99]:
df9=df8[df8.bath>1]

In [100]:
df9=df8[df8.bath>1]

In [101]:
df9=df8[df8.bath>1]

In [102]:
df9=df8[df8.bath>1]

In [103]:
df9=df8[df8.bath>1]

In [104]:
df9=df8[df8.bath>1]

In [105]:
df9=df8[df8.bath>1]

In [106]:
df9=df8[df8.bath>1]

In [107]:
df9=df8[df8.bath>1]

In [108]:
df9=df8[df8.bath>1]

In [109]:
df9=df8[df8.bath>1]

In [110]:
df9=df8[df8.bath>1]

In [111]:
df9=df8[df8.bath>1]

In [112]:
df9=df8[df8.bath>1]

In [113]:
df9=df8[df8.bath>1]

In [114]:
df9=df8[df8.bath>1]

In [115]:
df9=df8[df8.bath>1]

In [116]:
df9=df8[df8.bath>1]

In [117]:
df9=df8[df8.bath>1]

In [118]:
df9=df8[df8.bath>1]

In [119]:
df9=df8[df8.bath>1]

In [120]:
df9=df8[df8.bath>1]

In [121]:
df9=df8[df8.bath>1]

In [122]:
df9=df8[df8.bath>1]

In [123]:
df9=df8[df8.bath>1]

In [124]:
df9=df8[df8.bath>1]

In [125]:
df9=df8[df8.bath>1]

In [126]:
df9=df8[df8.bath>1]

In [127]:
df9=df8[df8.bath>1]

In [128]:
df9=df8[df8.bath>1]

In [129]:
df9=df8[df8.bath>1]

In [130]:
df9=df8[df8.bath>1]

In [131]:
df9=df8[df8.bath>1]

In [132]:
df9=df8[df8.bath>1]

In [133]:
df9=df8[df8.bath>1]

In [134]:
df9=df8[df8.bath>1]

In [135]:
df9=df8[df8.bath>1]

In [136]:
df9=df8[df8.bath>1]

In [137]:
df9=df8[df8.bath>1]

In [138]:
df9=df8[df8.bath>1]

In [139]:
df9=df8[df8.bath>1]

In [140]:
df9=df8[df8.bath>1]

In [141]:
df9=df8[df8.bath>1]

In [142]:
df9=df8[df8.bath>1]

In [143]:
df9=df8[df8.bath>1]

In [144]:
df9=df8[df8.bath>1]

In [145]:
df9=df8[df8.bath>1]

In [146]:
df9=df8[df8.bath>1]

In [147]:
df9=df8[df8.bath>1]

In [148]:
df9=df8[df8.bath>1]

In [149]:
df9=df8[df8.bath>1]

In [150]:
df9=df8[df8.bath>1]

In [151]:
df9=df8[df8.bath>1]

In [152]:
df9=df8[df8.bath>1]

In [153]:
df9=df8[df8.bath>1]

In [154]:
df9=df8[df8.bath>1]

In [155]:
df9=df8[df8.bath>1]

In [156]:
df9=df8[df8.bath>1]

In [157]:
df9=df8[df8.bath>1]

In [158]:
df9=df8[df8.bath>1]

In [159]:
df9=df8[df8.bath>1]

In [160]:
df9=df8[df8.bath>1]

In [161]:
df9=df8[df8.bath>1]

In [162]:
df9=df8[df8.bath>1]

In [163]:
df9=df8[df8.bath>1]

In [164]:
df9=df8[df8.bath>1]

In [165]:
df9=df8[df8.bath>1]

In [166]:
df9=df8[df8.bath>1]

In [167]:
df9=df8[df8.bath>1]

In [168]:
df9=df8[df8.bath>1]

In [169]:
df9=df8[df8.bath>1]

In [170]:
df9=df8[df8.bath>1]

In [171]:
df9=df8[df8.bath>1]

In [172]:
df9=df8[df8.bath>1]

In [173]:
df9=df8[df8.bath>1]

In [174]:
df9=df8[df8.bath>1]

In [175]:
df9=df8[df8.bath>1]

In [176]:
df9=df8[df8.bath>1]

In [177]:
df9=df8[df8.bath>1]

In [178]:
df9=df8[df8.bath>1]

In [179]:
df9=df8[df8.bath>1]

In [180]:
df9=df8[df8.bath>1]

In [181]:
df9=df8[df8.bath>1]

In [182]:
df9=df8[df8.bath>1]

In [183]:
df9=df8[df8.bath>1]

In [184]:
df9=df8[df8.bath>1]

In [185]:
df9=df8[df8.bath>1]

In [186]:
df9=df8[df8.bath>1]

In [187]:
df9=df8[df8.bath>1]

In [188]:
df9=df8[df8.bath>1]

In [189]:
df9=df8[df8.bath>1]

In [190]:
df9=df8[df8.bath>1]

In [191]:
df9=df8[df8.bath>1]

In [192]:
df9=df8[df8.bath>1]

In [193]:
df9=df8[df8.bath>1]

In [194]:
df9=df8[df8.bath>1]

In [195]:
df9=df8[df8.bath>1]

In [196]:
df9=df8[df8.bath>1]

In [197]:
df9=df8[df8.bath>1]

In [198]:
df9=df8[df8.bath>1]

In [199]:
df9=df8[df8.bath>1]

In [200]:
df9=df8[df8.bath>1]

In [201]:
df9=df8[df8.bath>1]

In [202]:
df9=df8[df8.bath>1]

In [203]:
df9=df8[df8.bath>1]

In [204]:
df9=df8[df8.bath>1]

In [205]:
df9=df8[df8.bath>1]

In [206]:
df9=df8[df8.bath>1]

In [207]:
df9=df8[df8.bath>1]

In [208]:
df9=df8[df8.bath>1]

In [209]:
df9=df8[df8.bath>1]

In [210]:
df9=df8[df8.bath>1]

In [211]:
df9=df8[df8.bath>1]

In [212]:
df9=df8[df8.bath>1]

In [213]:
df9=df8[df8.bath>1]

In [214]:
df9=df8[df8.bath>1]

In [215]:
df9=df8[df8.bath>1]

In [216]:
df9=df8[df8.bath>1]

In [217]:
df9=df8[df8.bath>1]

In [218]:
df9=df8[df8.bath>1]

In [219]:
df9=df8[df8.bath>1]

In [220]:
df9=df8[df8.bath>1]

In [221]:
df9=df8[df8.bath>1]

In [222]:
df9=df8[df8.bath>1]

In [223]:
df9=df8[df8.bath>1]

In [224]:
df9=df8[df8.bath>1]

In [225]:
df9=df8[df8.bath>1]

In [226]:
df9=df8[df8.bath>1]

In [227]:
df9=df8[df8.bath>1]

In [228]:
df9=df8[df8.bath>1]

In [229]:
df9=df8[df8.bath>1]

In [230]:
df9=df8[df8.bath>1]

In [231]:
df9=df8[df8.bath>1]

In [232]:
df9=df8[df8.bath>1]

In [233]:
df9=df8[df8.bath>1]

In [234]:
df9=df8[df8.bath>1]

In [235]:
df9=df8[df8.bath>1]

In [236]:
df9=df8[df8.bath>1]

In [237]:
df9=df8[df8.bath>1]

In [238]:
df9=df8[df8.bath>1]

In [239]:
df9=df8[df8.bath>1]

In [240]:
df9=df8[df8.bath>1]

In [241]:
df9=df8[df8.bath>1]

In [242]:
df9=df8[df8.bath>1]

In [243]:
df9=df8[df8.bath>1]

In [244]:
df9=df8[df8.bath>1]

In [245]:
df9=df8[df8.bath>1]

In [246]:
df9=df8[df8.bath>1]

In [247]:
df9=df8[df8.bath>1]

In [248]:
df9=df8[df8.bath>1]

In [249]:
df9=df8[df8.bath>1]

In [250]:
df9=df8[df8.bath>1]

In [251]:
df9=df8[df8.bath>1]

In [252]:
df9=df8[df8.bath>1]

In [253]:
df9=df8[df8.bath>1]

In [254]:
df9=df8[df8.bath>1]

In [255]:
df9=df8[df8.bath>1]

In [256]:
df9=df8[df8.bath>1]

In [257]:
df9=df8[df8.bath>1]

In [258]:
df9=df8[df8.bath>1]

In [259]:
df9=df8[df8.bath>1]

In [260]:
df9=df8[df8.bath>1]

In [261]:
df9=df8[df8.bath>1]

In [262]:
df9=df8[df8.bath>1]

In [263]:
df9=df8[df8.bath>1]

In [264]:
df9=df8[df8.bath>1]

In [265]:
df9=df8[df8.bath>1]

In [266]:
df9=df8[df8.bath>1]

In [267]:
df9=df8[df8.bath>1]

In [268]:
df9=df8[df8.bath>1]

In [269]:
df9=df8[df8.bath>1]

In [270]:
df9=df8[df8.bath>1]

In [271]:
df9=df8[df8.bath>1]

In [272]:
df9=df8[df8.bath>1]

In [273]:
df9=df8[df8.bath>1]

In [274]:
df9=df8[df8.bath>1]

In [275]:
df9=df8[df8.bath>1]

In [276]:
df9=df8[df8.bath>1]

In [277]:
df9=df8[df8.bath>1]

In [278]:
df9=df8[df8.bath>1]

In [279]:
df9=df8[df8.bath>1]

In [280]:
df9=df8[df8.bath>1]

In [281]:
df9=df8[df8.bath>1]

In [282]:
df9=df8[df8.bath>1]

In [283]:
df9=df8[df8.bath>1]

In [284]:
df9=df8[df8.bath>1]

In [285]:
df9=df8[df8.bath>1]

In [286]:
df9=df8[df8.bath>1]

In [287]:
df9=df8[df8.bath>1]

In [288]:
df9=df8[df8.bath>1]

In [289]:
df9=df8[df8.bath>1]

In [290]:
df9=df8[df8.bath>1]

In [291]:
df9=df8[df8.bath>1]

In [292]:
df9=df8[df8.bath>1]

In [293]:
df9=df8[df8.bath>1]

In [294]:
df9=df8[df8.bath>1]

In [295]:
df9=df8[df8.bath>1]

In [296]:
df9=df8[df8.bath>1]

In [297]:
df9=df8[df8.bath>1]

In [298]:
df9=df8[df8.bath>1]

In [299]:
df9=df8[df8.bath>1]

In [300]:
df9=df8[df8.bath>1]

In [301]:
df9=df8[df8.bath>1]

In [302]:
df9=df8[df8.bath>1]

In [303]:
df9=df8[df8.bath>1]

In [304]:
df9=df8[df8.bath>1]

In [305]:
df9=df8[df8.bath>1]

In [306]:
df9=df8[df8.bath>1]

In [307]:
df9=df8[df8.bath>1]

In [308]:
df9=df8[df8.bath>1]

In [309]:
df9=df8[df8.bath>1]

In [310]:
df9=df8[df8.bath>1]

In [311]:
df9=df8[df8.bath>1]

In [312]:
df9=df8[df8.bath>1]

In [313]:
df9=df8[df8.bath>1]

In [314]:
df9=df8[df8.bath>1]

In [315]:
df9=df8[df8.bath>1]

In [316]:
df9=df8[df8.bath>1]

In [317]:
df9=df8[df8.bath>1]

In [318]:
df9=df8[df8.bath>1]

In [319]:
df9=df8[df8.bath>1]

In [320]:
df9=df8[df8.bath>1]

In [321]:
df9=df8[df8.bath>1]

In [322]:
df9=df8[df8.bath>1]

In [323]:
df9=df8[df8.bath>1]

In [324]:
df9=df8[df8.bath>1]

In [325]:
df9=df8[df8.bath>1]

In [326]:
df9=df8[df8.bath>1]

In [327]:
df9=df8[df8.bath>1]

In [328]:
df9=df8[df8.bath>1]

In [329]:
df9=df8[df8.bath>1]

In [330]:
df9=df8[df8.bath>1]

In [331]:
df9=df8[df8.bath>1]

In [332]:
df9=df8[df8.bath>1]

In [333]:
df9=df8[df8.bath>1]

In [334]:
df9=df8[df8.bath>1]

In [335]:
df9=df8[df8.bath>1]

In [336]:
df9=df8[df8.bath>1]

In [3
```


df12

	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Nagar	2nd Stage Nagarbhavi	5th Block Hbr Layout	5th Phase JP Nagar	...	Vijayanagar	Vishveshwarya Layout	Vishwapiya Layout	Vit
0	2850.0	4.0	428.0	4	1	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
2	1875.0	2.0	235.0	3	1	0	0	0	0	0	...	0	0	0	0
3	1200.0	2.0	130.0	3	1	0	0	0	0	0	...	0	0	0	0
4	1235.0	2.0	148.0	2	1	0	0	0	0	0	...	0	0	0	0
...
10233	1200.0	2.0	700	2	0	0	0	0	0	0	...	0	0	0	0
10234	1800.0	1.0	200.0	1	0	0	0	0	0	0	...	0	0	0	0
10237	1353.0	2.0	110.0	2	0	0	0	0	0	0	...	0	0	0	0
10238	812.0	1.0	26.0	1	0	0	0	0	0	0	...	0	0	0	0
10241	3600.0	5.0	400.0	4	0	0	0	0	0	0	...	0	0	0	0

7239 rows x 244 columns

```
In [68]: df12.shape
Out[68]: (7239, 244)

In [69]: X=df12.drop(['price'],axis='columns')
Out[69]: X.head(3)

Out[69]:
total_sqft  bath  bhk  1st Block  1st  2nd  2nd Stage  5th  5th  6th  ...  Vijayanagar  Vishveshwarya  Vishwapiya  Vittasa
Jayanagar  Phase JP Nagar  Phase Judicial Nagarbhavi  Block Hbr Layout  Phase JP Nagar  Phase JP Nagar
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
3 130.0
4 148.0
Name: price, dtype: float64

In [72]: len(y)
Out[72]: 7239

In [73]: 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 [74]: from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)

Out[74]: 0.8629132245229443
```

Use K Fold cross validation to measure accuracy of our LinearRegression model

```
In [75]: from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit

cvc= ShuffleSplit(n_splits=9,test_size=0.2, random_state=0)
cross_val_score(LinearRegression(),X,y,cv=cvc)

Out[75]: array([0.82702546, 0.86027005, 0.8532178, 0.8436466, 0.85481502,
0.7996843, 0.8546295, 0.84180048, 0.79241964])

In [76]: 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)

Out[76]:
model best_score best_params
0 linear_regression 0.847796 {normalize: False}
1 lasso 0.726738 {'alpha': 2, 'selection': 'cyclic'}
2 decision_tree 0.716838 {'criterion': 'friedman_mse', 'splitter': 'ran...
```

```
In [77]: 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 [78]: df12.columns
Out[78]: Index(['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', 'Vishwapiya Layout',
'Vittasandra', 'Whitfield', 'Yelachenahalli', 'Yelahanka',
'Yelahanka New Town', 'Yelenahalli', 'Yeshwantpur'],
dtype='object', length=244)

In [79]: predict_price('Whitefield',5000,4,5)
Out[79]: 377.12579897851015

In [80]: predict_price('Indira Nagar',2000,3,4)
Out[80]: 274.14277436167635

In [86]: predict_price('Ulsoor',800,1,2)
Out[86]: 68.49165579594116

In [84]: predict_price('Yelahanka',800,1,2)
Out[84]: 26.519683081756575

In [82]: # import pickle
# with open('model.pickle','wb') as f:
#     pickle.dump(lr_clf,f)

In [83]: # 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 [ ]:
```