Bandırma Onyadi Eylül Üniversitesi

Veri Madenciliği Final Ödevi – 2021

Md Nasrullah **Öğranci No : 205210030**

Bu Veri set kaggle web sitesinden alındı. Bu çalışmada veri set Makina öğrenme üzeinde bir kaç algorıthma ile sınıflandımaya çalıştım.

Bu Çalışma Booyunca kullnacağım :

- Python 3.8
- Jupyter Notebook

Yapılan adımlar :

- Kütüphaneler Eklmek
- Veri seti Çağırmak
- Veri içinde Boş veri bakmak ve temizlemek
- Bazen String(Sözel) Sayısal a dönüştürmek
- Feature Seçmek
- Outlier Düşürmek
- OneHot encoder kullanmak
- Model
- K Fold cross kullanmak
- Ve pickle kullanarak model kayıt etmek

Veri Birim proje: Ev Fiyati Tahmini

Dataset is downloaded from here (Veriye bu linkten indirebilirsiniz): https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data

In [2]:

```
# Kütüphaneler Ekleyelim
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
import matplotlib
matplotlib.rcParams["figure.figsize"] = (20,10)
```

Veriye Çarmak

In [3]:

```
df1 = pd.read_csv("bengaluru_house_prices.csv")
df1.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 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

```
In [4]: df1.shape

Out[4]: (13320, 9)
```

```
In [5]: df1.columns # Sütünler

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

```
In [6]: df1['area_type'].unique()

Out[6]: array(['Super built-up  Area', 'Plot  Area', 'Built-up  Area',
               'Carpet  Area'], dtype=object)
```

```
In [7]: df1['area_type'].value_counts()  # Hangi çeşit Veri DeğiŞimler var

Out[7]: Super built-up  Area    8790
        Built-up  Area          2418
        Plot  Area              2025
        Carpet  Area              87
        Name: area_type, dtype: int64
```

Gereksiz sütünler(Feature) Düşürmek

```
In [15]: df2 = df1.drop(['area_type','society','balcony','availability'],axis='columns')
         df2.shape

Out[15]: (13320, 5)
```

Data Cleaning: Veri Temizleme

```
In [9]: df2.isnull().sum() # Boş veriler bulmak

Out[9]: location       1
        size          16
        total_sqft     0
        bath          73
        price          0
        dtype: int64
```

```
In [10]: df2.shape

Out[10]: (13320, 5)
```

```
In [11]: df3 = df2.dropna() # Boş veriler ortadan kaldırmak
         df3.isnull().sum()

Out[11]: location      0
         size          0
         total_sqft    0
```

```
In [12]: df3.shape
```

```
Out[12]: (13246, 5)
```

Feature Engineering

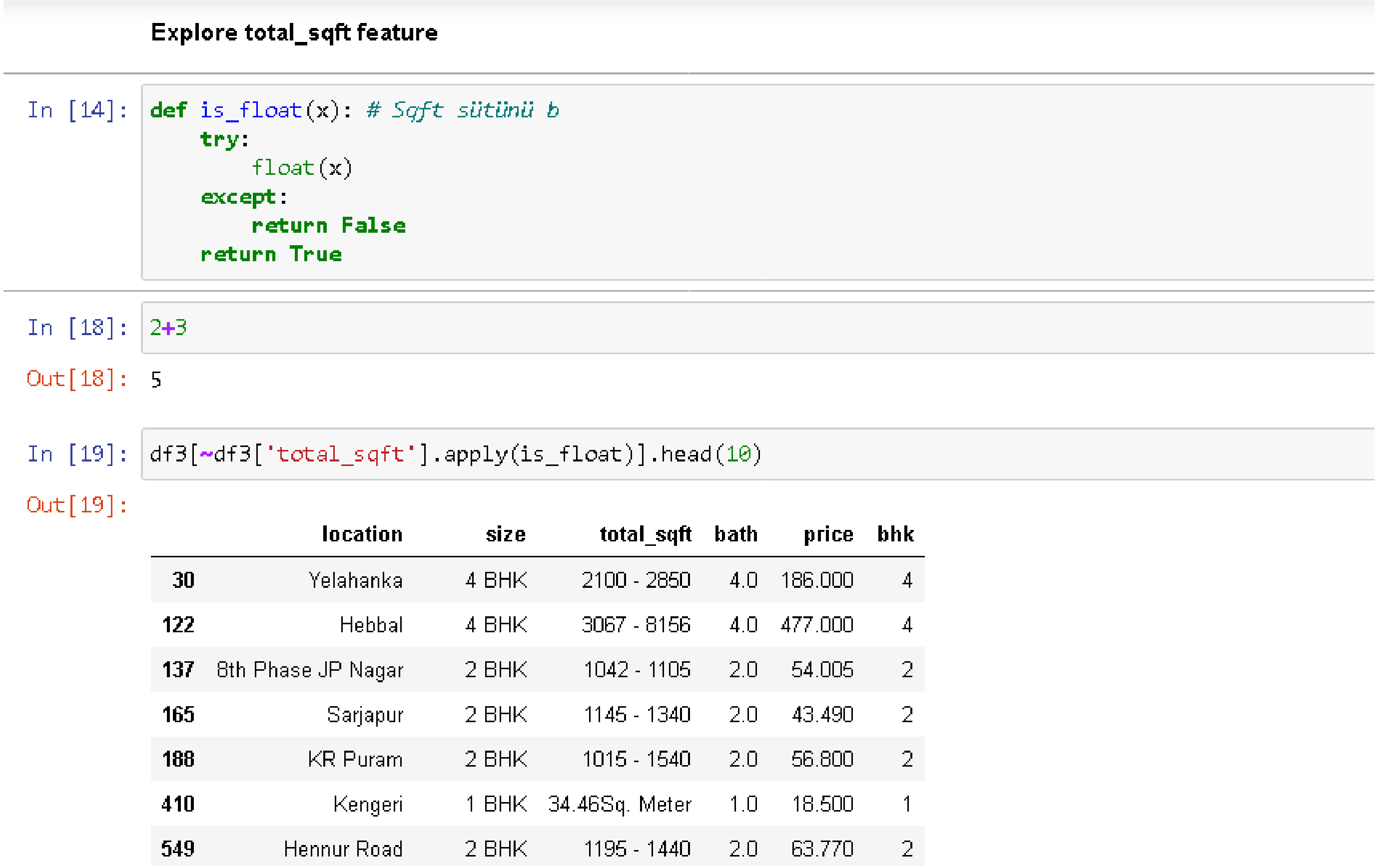
*Add new feature(integer) for bhk (Bedrooms Hall Kitchen) Yani bhk i sayıya dönüştürmek *

```
In [13]: df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
         df3.bhk.unique()
```

```
<ipython-input-13-681cf3aca53d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

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



Explore total_sqft feature

```
In [14]: def is_float(x): # Sqft sütünü b
             try:
                 float(x)
             except:
                 return False
             return True
```

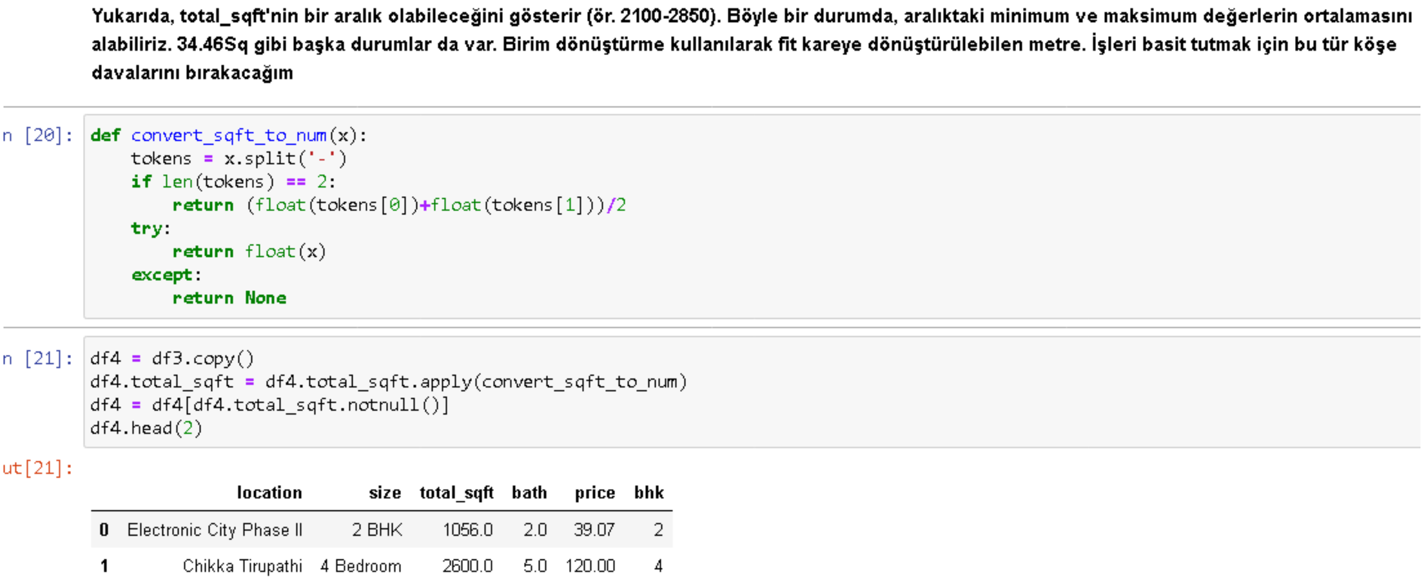
```
In [18]: 2+3
```

```
Out[18]: 5
```

```
In [19]: df3[~df3['total_sqft'].apply(is_float)].head(10)
```

Out[19]:

	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



Yukarıda, total_sqft'nin bir aralık olabileceğini gösterir (ör. 2100-2850). Böyle bir durumda, aralıktaki minimum ve maksimum değerlerin ortalamasını alabiliriz. 34.46Sq gibi başka durumlar da var. Birim dönüştürme kullanılarak fit kareye dönüştürülebilen metre. İşleri basit tutmak için bu tür köşe davalarını bırakacağım

n [20]:

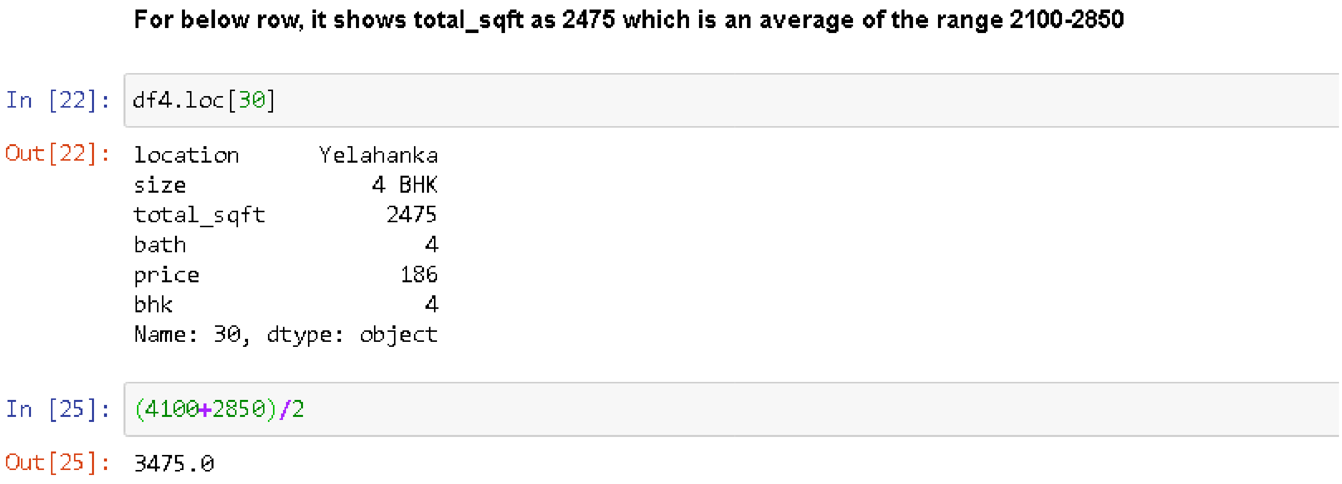
```
def convert_sqft_to_num(x):
    tokens = x.split('-')
    if len(tokens) == 2:
        return (float(tokens[0])+float(tokens[1]))/2
    try:
        return float(x)
    except:
        return None
```

n [21]:

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

ut[21]:

	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



For below row, it shows total_sqft as 2475 which is an average of the range 2100-2850

In [22]:

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

Out[22]:

```
location      Yelahanka
size              4 BHK
total_sqft         2475
bath                  4
price               186
bhk                   4
Name: 30, dtype: object
```

In [25]:

```
(4100+2850)/2
```

Out[25]:

```
3475.0
```

Feature Engineering

Add new feature called price per square feet

```
In [27]: df5 = df4.copy()
         df5['price_per_sqft'] = df5['price']*100000/df5['total_sqft']
         df5.head()
```

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

```
In [28]: df5_stats = df5['price_per_sqft'].describe()
         df5_stats
```

```
Out[28]: count    1.320000e+04
         mean     7.920759e+03
         std      1.067272e+05
         min      2.678298e+02
         25%      4.267701e+03
         50%      5.438331e+03
         75%      7.317073e+03
         max      1.200000e+07
         Name: price_per_sqft, dtype: float64
```

```
In [29]: df5.to_csv("bhp.csv",index=False)
```

Kategorik bir değişken olan konumları inceleyin. Konum sayısını azaltmak için burada boyutsallık azaltma tekniğini uygulamamız gerekiyor.

```
In [30]: df5.location = df5.location.apply(lambda x: x.strip())
         location_stats = df5['location'].value_counts(ascending=False)
         location_stats
```

```
Out[30]: Whitefield               533
         Sarjapur  Road           392
         Electronic City          304
```

```
In [31]: location_stats.values.sum()
```

```
Out[31]: 13200
```

```
In [32]: len(location_stats[location_stats>10])
```

```
Out[32]: 240
```

```
In [33]: len(location_stats)
```

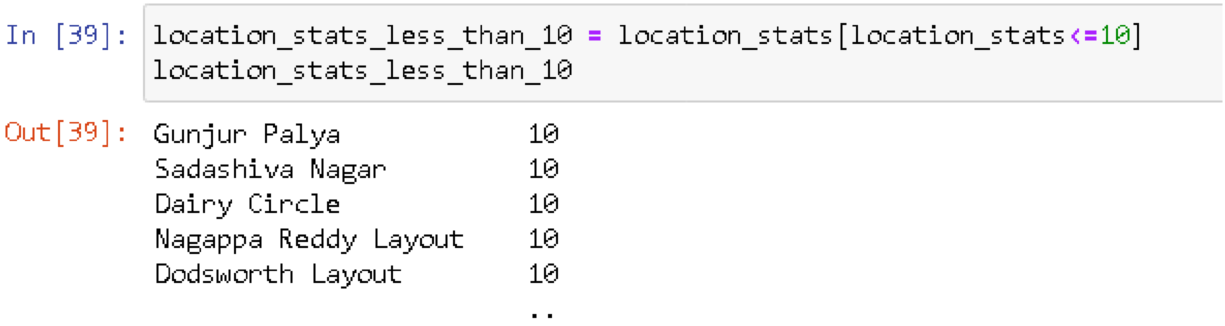
```
Out[33]: 1287
```

```
In [34]: len(location_stats[location_stats<=10])
```

```
Out[34]: 1047
```

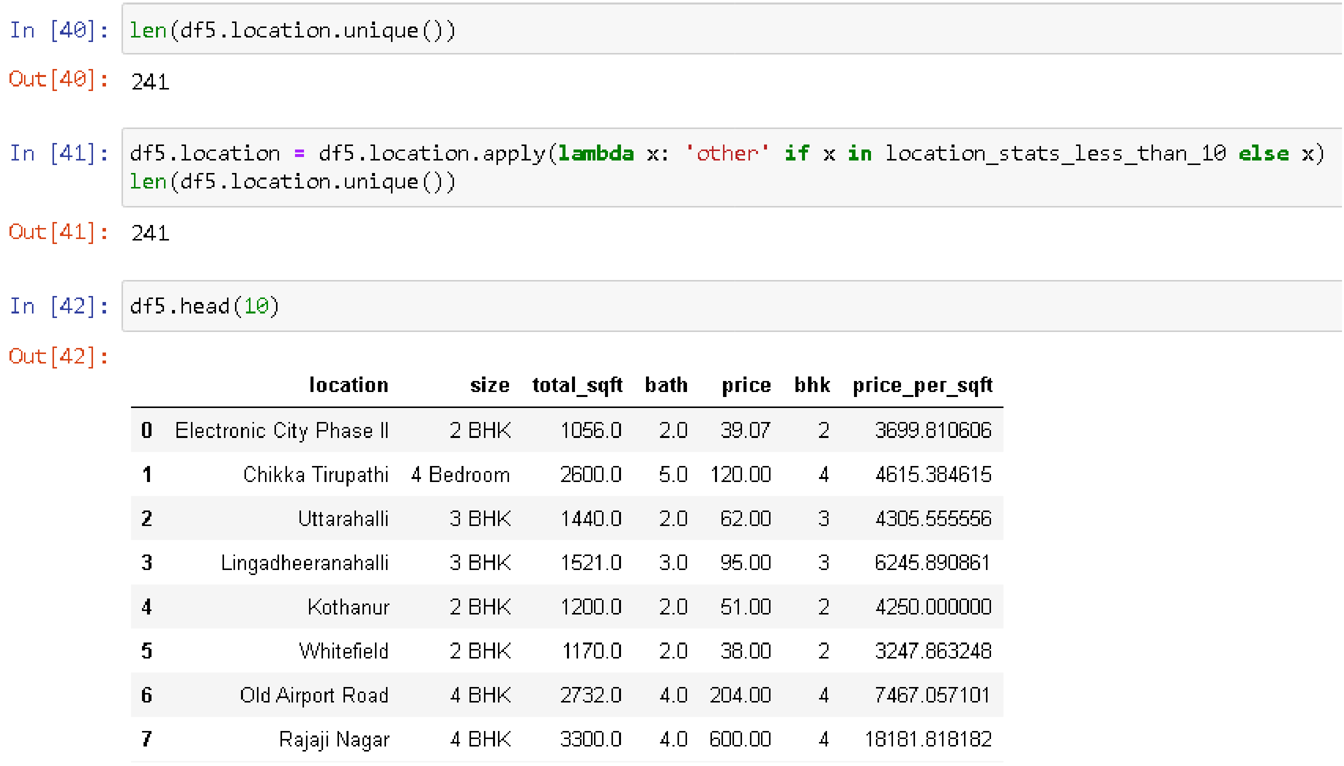
Dimensionality Reduction

10'dan az veri noktasına sahip herhangi bir konum "diğer" konum olarak etiketlenmelidir. Bu şekilde kategori sayısı büyük miktarda azaltılabilir. Daha sonra bir sıcak kodlama yaptığımızda, daha az boş sütuna sahip olmamıza yardımcı olacaktır



```
In [39]: location_stats_less_than_10 = location_stats[location_stats<=10]
         location_stats_less_than_10
```

```
Out[39]: Gunjur Palya            10
         Sadashiva Nagar         10
         Dairy Circle            10
         Nagappa Reddy Layout    10
         Dodsworth Layout        10
                                 ..
```



```
In [40]: len(df5.location.unique())
```

```
Out[40]: 241
```

```
In [41]: df5.location = df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else x)
         len(df5.location.unique())
```

```
Out[41]: 241
```

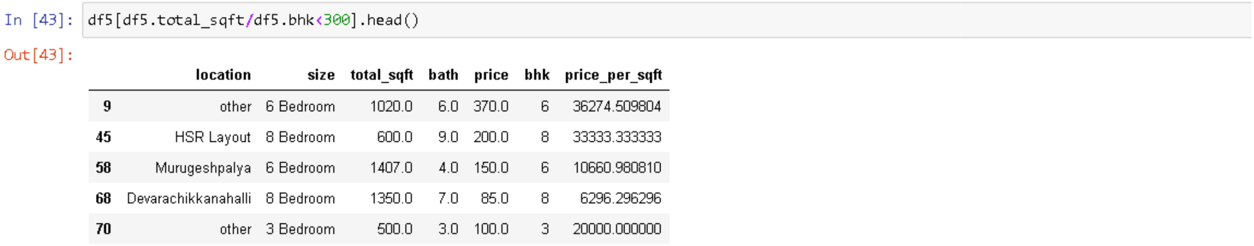
```
In [42]: df5.head(10)
```

Out[42]:

	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

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 than that seems suspicious and can be removed as an outlier. We will remove such outliers by keeping our minimum thresold per bhk to be 300 sqft



```
In [43]: df5[df5.total_sqft/df5.bhk<300].head()
```

Out[43]:

	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

Yukarıdaki veri noktalarını kontrol edin. 1020 m2'lik 6 bhk dairemiz var. Bir diğeri 8 bhk ve toplam sqft 600'dür. Bunlar güvenle giderilebilecek net veri hatalarıdır.

Yukarıdaki veri noktalarını kontrol edin. 1020 m2'lik 6 bhk dairemiz var. Bir diğeri 8 bhk ve toplam sqft 600'dür. Bunlar güvenle giderilebilecek net veri hatalarıdır.

```
In [44]: df5.shape
```

```
Out[44]: (13200, 7)
```

```
In [45]: df6 = df5[~(df5.total_sqft/df5.bhk<300)]
         df6.shape
```

```
Out[45]: (12456, 7)
```

Outlier Removal Using Standard Deviation and Mean

```
In [46]: df6.price_per_sqft.describe()
```

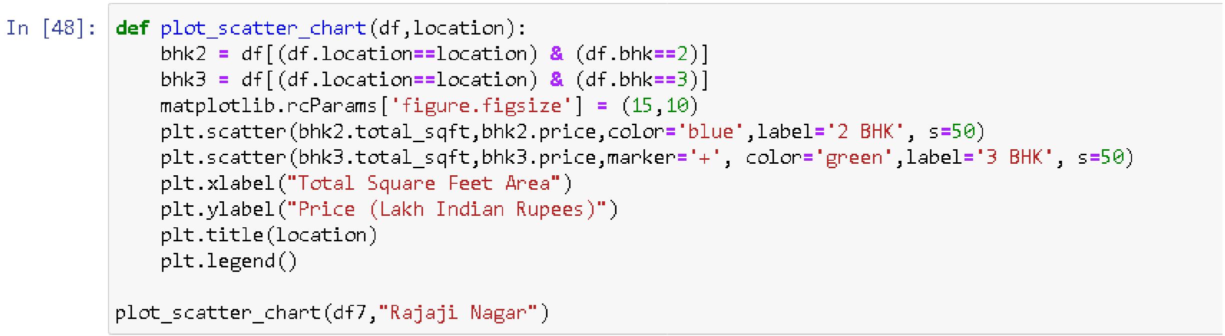
```
Out[46]: count     12456.000000
         mean       6308.502826
         std        4168.127339
         min         267.829813
         25%        4210.526316
         50%        5294.117647
         75%        6916.666667
         max      176470.588235
         Name: price_per_sqft, dtype: float64
```

Burada metrekare başına minimum fiyatın 267 rs/sqft olduğunu, maksimum 12000000 olduğunu görüyoruz, bu emlak fiyatlarında geniş bir değişiklik gösteriyor. Ortalama ve bir standart sapma kullanarak konum başına aykırı değerleri kaldırmalıyız

```
In [47]: 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
```

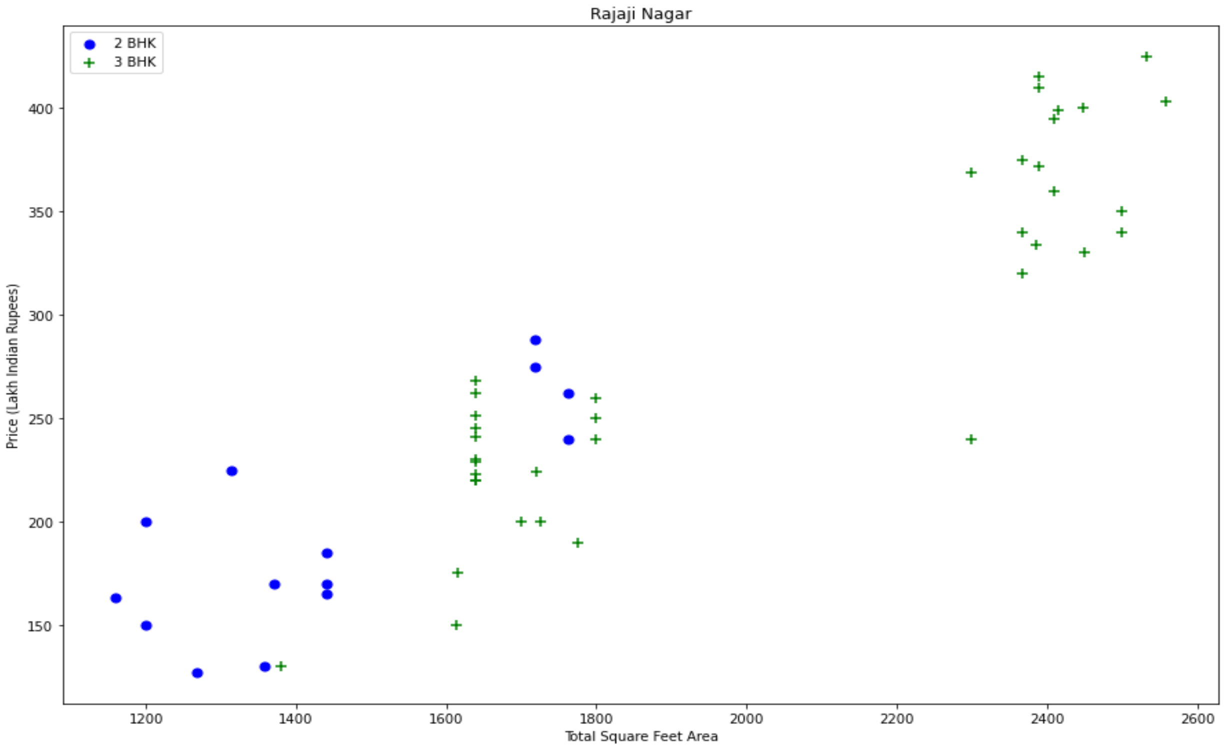
```
Out[47]: (10242, 7)
```

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

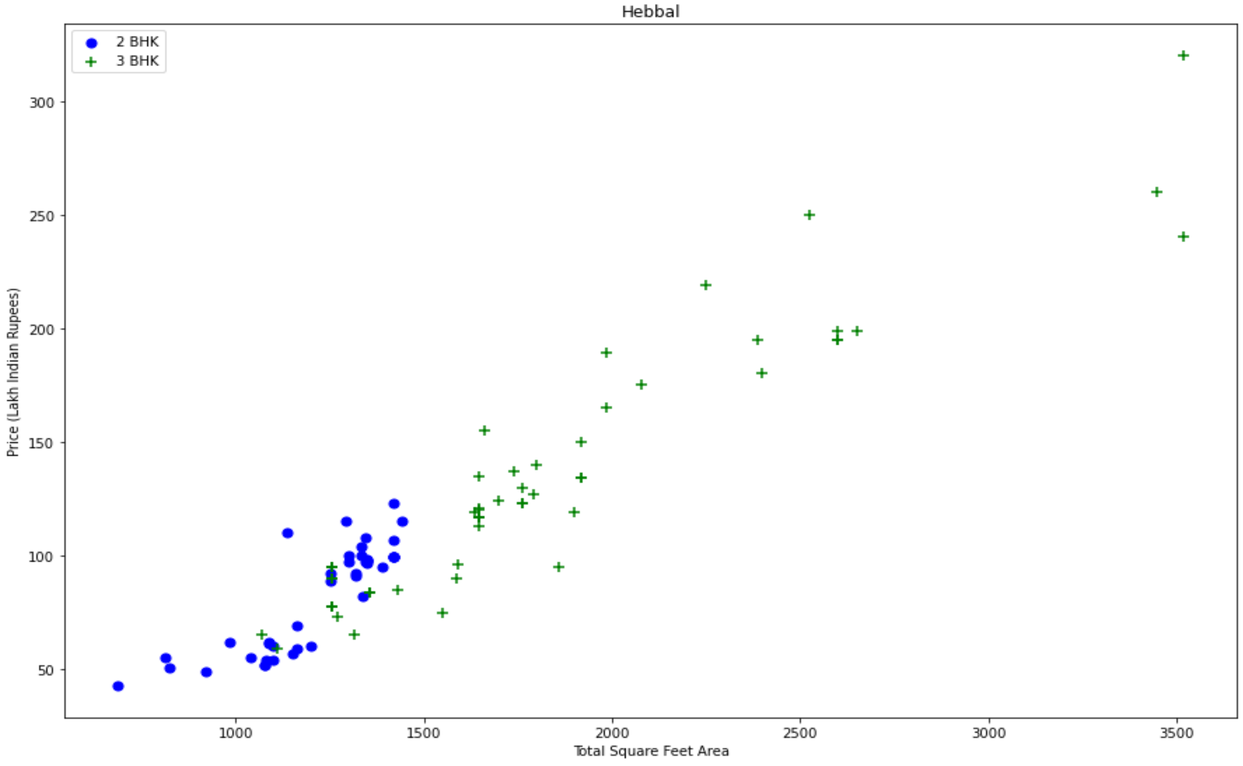


```
In [48]: def plot_scatter_chart(df,location):
             bhk2 = df[(df.location==location) & (df.bhk==2)]
             bhk3 = df[(df.location==location) & (df.bhk==3)]
             matplotlib.rcParams['figure.figsize'] = (15,10)
             plt.scatter(bhk2.total_sqft,bhk2.price,color='blue',label='2 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 [49]: plot_scatter_chart(df7,"Hebbal")
```



We should also remove properties where for same location, the price of (for example) 3 bedroom apartment is less than 2 bedroom apartment (with same square ft area). What we will do is for a given location, we will build a dictionary of stats per bhk, i.e.

```
{
    '1' : {
        'mean': 4000,
        'std: 2000,
        'count': 34
    },
    '2' : {
        'mean': 4300,
        'std: 2300,
        'count': 22
    },
}
```

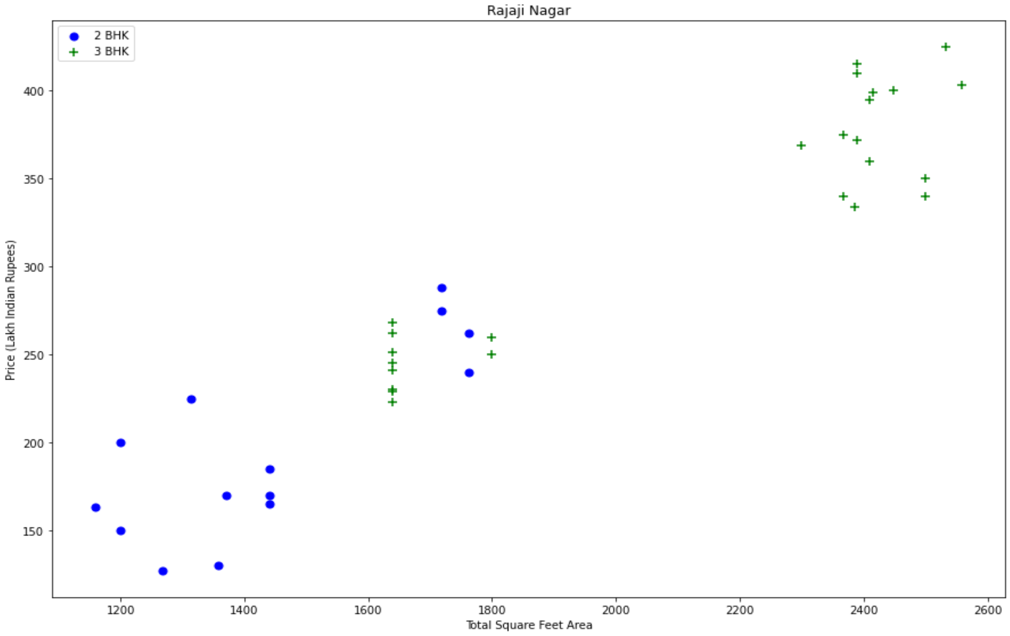
Now we can remove those 2 BHK apartments whose price_per_sqft is less than mean price_per_sqft of 1 BHK apartment

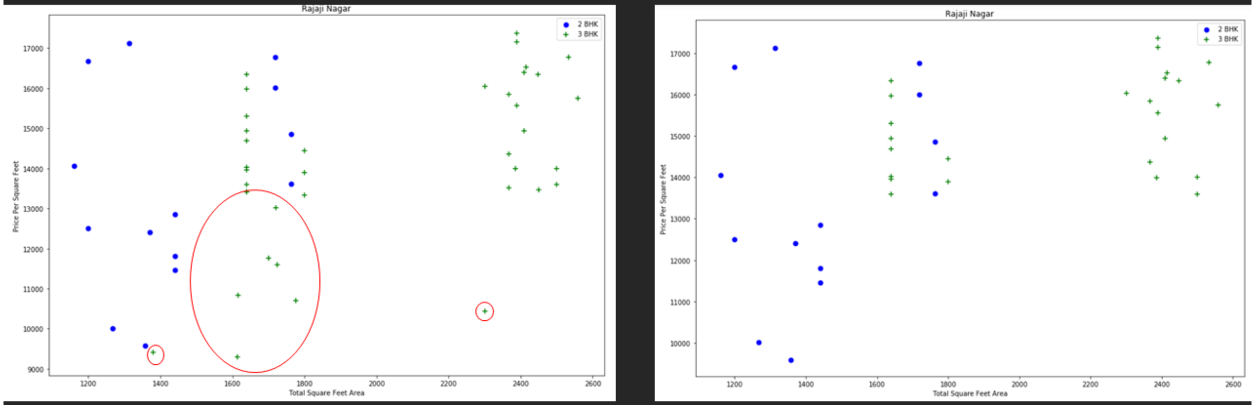
In [50]:
```
def remove_bhk_outliers(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean': np.mean(bhk_df.price_per_sqft),
                '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[50]:
```
(7317, 7)
```

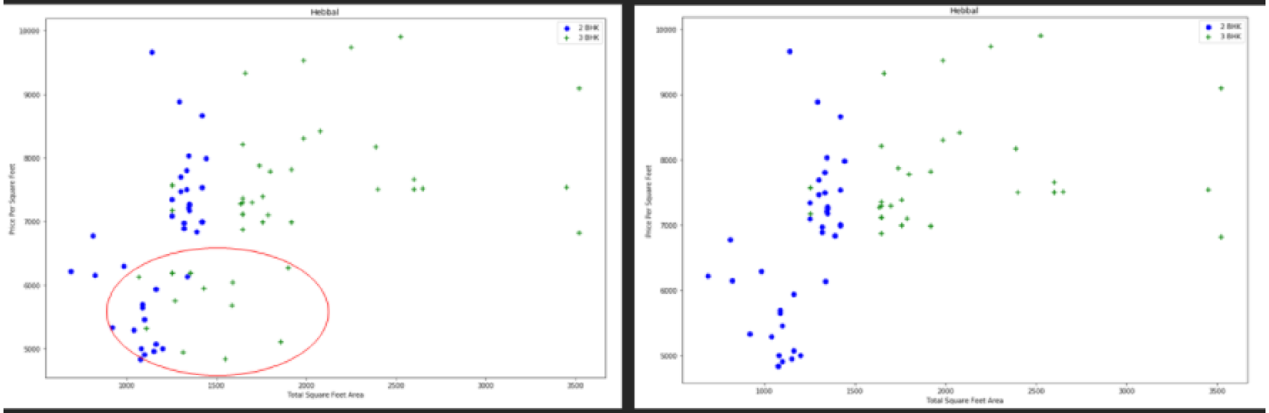
Plot same scatter chart again to visualize price_per_sqft for 2 BHK and 3 BHK properties

In [51]:
```
plot_scatter_chart(df8,"Rajaji Nagar")
```



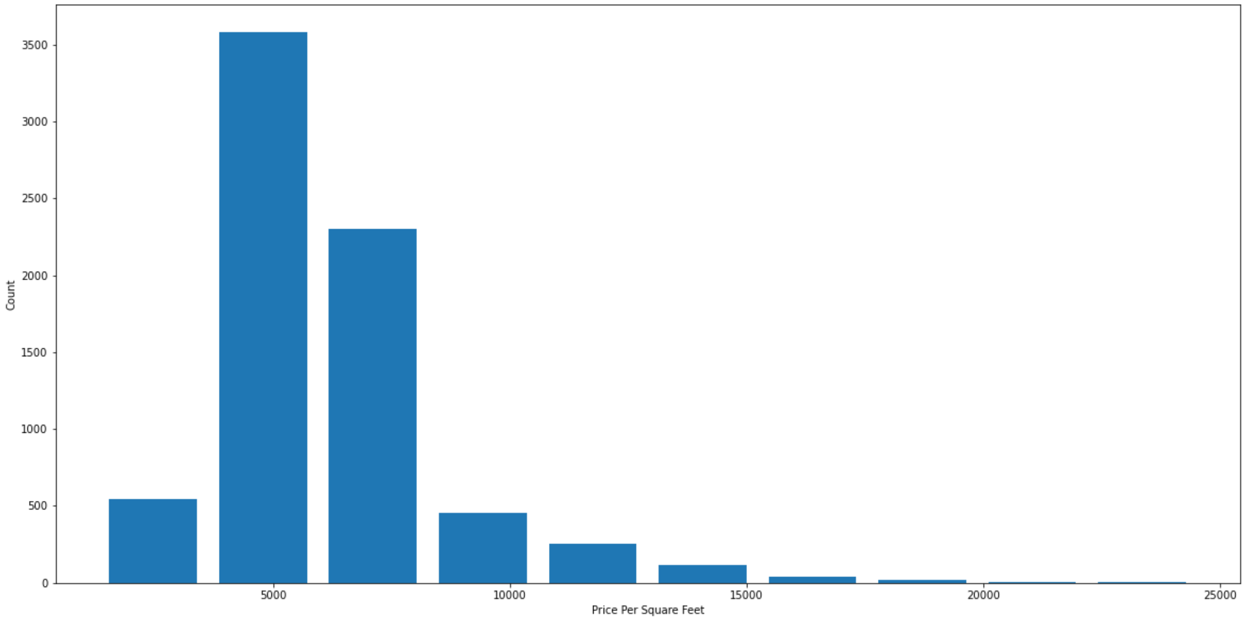


Before and after outlier removal: Hebbal



```
In [53]: 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[53]: Text(0, 0.5, 'Count')
```



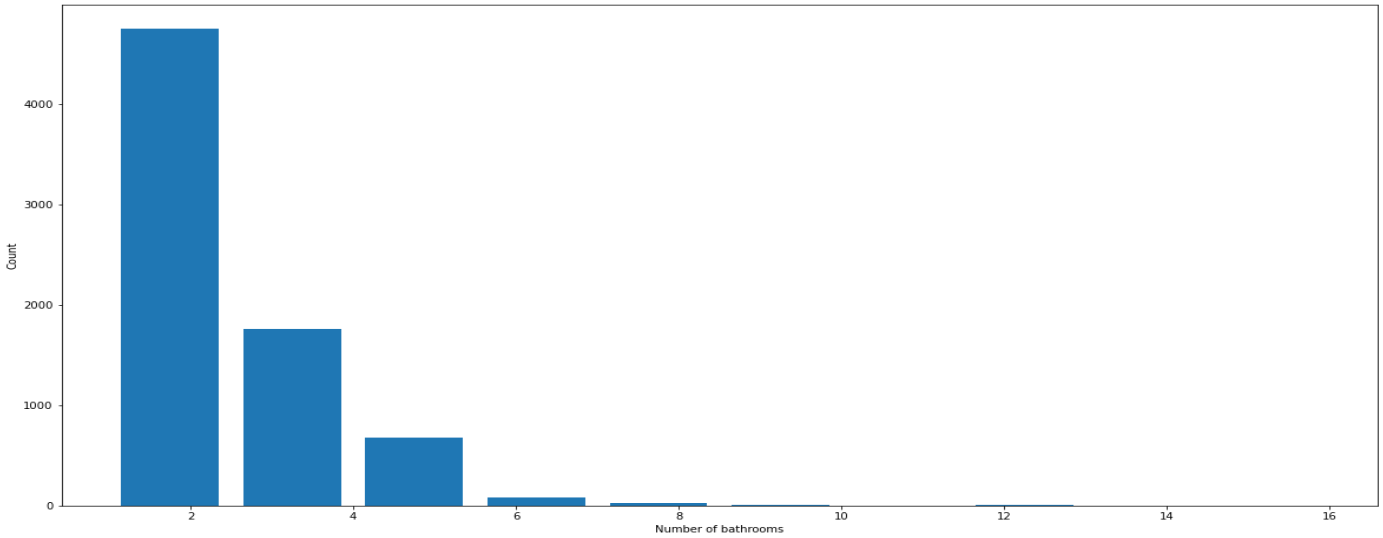
Outlier Removal Using Bathrooms Feature

```
In [54]: df8.bath.unique()
```

```
Out[54]: array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

```
In [55]: plt.hist(df8.bath,rwidth=0.8)
         plt.xlabel("Number of bathrooms")
         plt.ylabel("Count")
```

```
Out[55]: Text(0, 0.5, 'Count')
```



In [56]:
```
df8[df8.bath>10]
```

Out[56]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8483	other	10 BHK	12000.0	12.0	525.0	10	4375.000000
8572	other	16 BHK	10000.0	16.0	550.0	16	5500.000000
9306	other	11 BHK	6000.0	12.0	150.0	11	2500.000000
9637	other	13 BHK	5425.0	13.0	275.0	13	5069.124424

It is unusual to have 2 more bathrooms than number of bedrooms in a home

In [57]:
```
df8[df8.bath>df8.bhk+2]
```

Out[57]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1626	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5238	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6711	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8408	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689

Yine işletme müdürü sizinle (yani bir veri bilimcisi) 4 yatak odalı eviniz varsa ve 4 odanın hepsinde banyo artı bir misafir banyonuz olsa bile toplam banyo = toplam yatak + 1 maks. . Bunun üzerindeki herhangi bir şey aykırı değer veya veri hatasıdır ve kaldırılabilir

Yine işletme müdürü sizinle (yani bir veri bilimcisi) 4 yatak odalı eviniz varsa ve 4 odanın hepsinde banyo artı bir misafir banyonuz olsa bile toplam banyo = toplam yatak + 1 maks. . Bunun üzerindeki herhangi bir şey aykırı değer veya veri hatasıdır ve kaldırılabilir

In [58]:
```
df9 = df8[df8.bath<df8.bhk+2]
df9.shape
```

Out[58]:
```
(7239, 7)
```

In [59]:
```
df9.head(2)
```

Out[59]:

	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 [60]:
```
df10 = df9.drop(['size','price_per_sqft'],axis='columns')
df10.head(3)
```

Out[60]:

	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

Use One Hot Encoding For Location

In [61]:
```
dummies = pd.get_dummies(df10.location)
dummies.head(3)
```

Out[61]:

	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	Yela
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 × 241 columns

In [62]:
```
df11 = pd.concat([df10,dummies.drop('other',axis='columns')],axis='columns')
df11.head()
```

Out[62]:

	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	W
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	0	0	0	0	

In [63]:
```
df12 = df11.drop('location',axis='columns')
df12.head(2)
```

Out[63]:

	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	Whit
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 rows × 244 columns

Build a Model Now...

In [64]:
```
df12.shape
```
Out[64]:
```
(7239, 244)
```

In [65]:
```
X = df12.drop(['price'],axis='columns')
X.head(3)
```
Out[65]:

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

3 rows × 243 columns

In [66]:
```
X.shape
```
Out[66]:
```
(7239, 243)
```

In [67]:
```
y = df12.price
y.head(3)
```
Out[67]:
```
0    428.0
1    194.0
2    235.0
Name: price, dtype: float64
```

In [68]:
```
0len(y)
```
Out[68]:
```
7239
```

In [69]:
```
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 [70]:
```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```
Out[70]:
```
0.8629132245229522
```

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

In [71]:
```
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[71]:
```
array([0.82702546, 0.86027005, 0.85322178, 0.8436466 , 0.85481502])
```

We can see that in 5 iterations we get a score above 80% all the time. This is pretty good but we want to test few other algorithms for regression to see if we can get even better score. We will use GridSearchCV for this purpose

Find best model using GridSearchCV

In [72]:

```
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[72]:

	model	best_score	best_params
0	linear_regression	0.847796	{'normalize': False}
1	lasso	0.726823	{'alpha': 2, 'selection': 'random'}
2	decision_tree	0.728107	{'criterion': 'friedman_mse', 'splitter': 'ran...

Based on above results we can say that LinearRegression gives the best score. Hence we will use that.

Test the model for few properties

```
In [93]: 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 [74]: predict_price('1st Phase JP Nagar',1000, 2, 2)

Out[74]: 83.8657025833251
```

```
In [75]: predict_price('1st Phase JP Nagar',1000, 3, 3)

Out[75]: 86.08062285007186
```

```
In [76]: predict_price('Indira Nagar',1000, 2, 2)

Out[76]: 193.31197733179096
```

```
In [77]: predict_price('Indira Nagar',1000, 3, 3)

Out[77]: 195.52689759853774
```

Export the tested model to a pickle file

```
In [84]: import pickle
         with open('banglore_my ','wb') as f:
             pickle.dump(lr_clf,f)
```

```
In [95]: import pickle
```

```
In [96]: with open ('banglore001','wb') as f:
             pickle.dump(lr_clf,f)
```

Export location and column information to a file that will be useful later on in our prediction application

```
In [85]: import json
         columns = {
             'data_columns' : [col.lower() for col in X.columns]
         }
         with open("columns1.json","w") as f:
             f.write(json.dumps(columns))
```

```
In [97]: with open('banglore001','rb') as f :
             mp=pickle.load(f)
```