

Import Libraries

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

#Load dataset
df1=pd.read_csv("/content/Bengaluru_House_Data (1).csv")
```

df1.head(6) #Returns first 6 rows , default is 5 when no number is given

	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
5	Super built-up Area	Ready To Move	Whitefield	2 BHK	DuenaTa	1170	2.0	1.0	38.00

df1.shape # returns rows and columns

(13320, 9)

df1.groupby('area_type')['area_type'].agg('count') #count all objects of same area type

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

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

	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

df2.isnull().sum() #gives which columns have NA values with count , then we will drop those rows if they are small in number

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

```
df3=df2.dropna()
df3.isnull().sum() # we could also replace na values with median values instead of dropping

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

Columns Dropped

df3['size'].unique() #we need to write in one format , either bhk or bedroom so we use create a new column

```
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)
```

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

<ipython-input-13-c379116b8702>: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-
df3['bhk']=df3['size'].apply(lambda x: int(x.split(' ')[0]))

```
df3.head()
```

	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

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

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

```
df3[df3.bhk>20] #no. of bedrooms as 43 was strange , so we'll check
```

	location	size	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	27 BHK	8000	27.0	230.0	27
4684	Munnekollal	43 Bedroom	2400	40.0	660.0	43

There is something weird , a 43 bedroom flat can not have 2400 sq ft. area ,we will correct late

```
df3.total_sqft.unique()
```

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

```
#we see there is a range and we need numbers
```

```
def is_float(x):
    try:
        float(x)
    except:
        return False
    return True
```

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

	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 410 , the format is not correct , so we can ignore these rows or do unit conversion

```
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
```

```
convert_sqft_to_num('2166')
```

```
2166.0
```

```
convert_sqft_to_num('34.46Sq. Meter')
```

```
# Didn't return anything which is what we want
```

```
df4=df3.copy()
df4['total_sqft']=df4['total_sqft'].apply(convert_sqft_to_num)
```

```
df4.head()
```

	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

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

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

```
(2100+2850)/2 #so correct
```

```
2475.0
```

```
# By now , handled NAN and cleaned sqft column and also removed some unnecessary features
# Now , we learn feature engineering techniques
```

```
df5=df4.copy()
df5['price_per_sqft']=df5['price']*100000/df5['total_sqft'] #price is an important feature in real estate
df5.head()
```

	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

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

```
1304
```

```
# Too many columns , dimensionality curse
```

```
df5.location=df5.location.apply(lambda x:x.strip()) #removes extra white spaces
location_stats = df5.groupby('location')['location'].agg('count').sort_values(ascending=False)
location_stats
```

```
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
```

```
#the distribution is very uneven
```

```
len(location_stats[location_stats<=10]) #to find a threshold
```

1052

#Since 1052 is a big number , we create a new column

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



```
location_stats_less_than_10
```

```
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
whitefield         1
Name: location, Length: 1052, dtype: int64
```



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

242

```
df5.head(10) #notice 10th row
```

	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	

```
df5[df5.total_sqft/df5.bhk<300].head() #average 1 bhk size is 300 so , the following data is unsual
```

	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	

We found some price outliers using price column

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

```
(12502, 7)
```

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

```
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
```

We found some more outliers

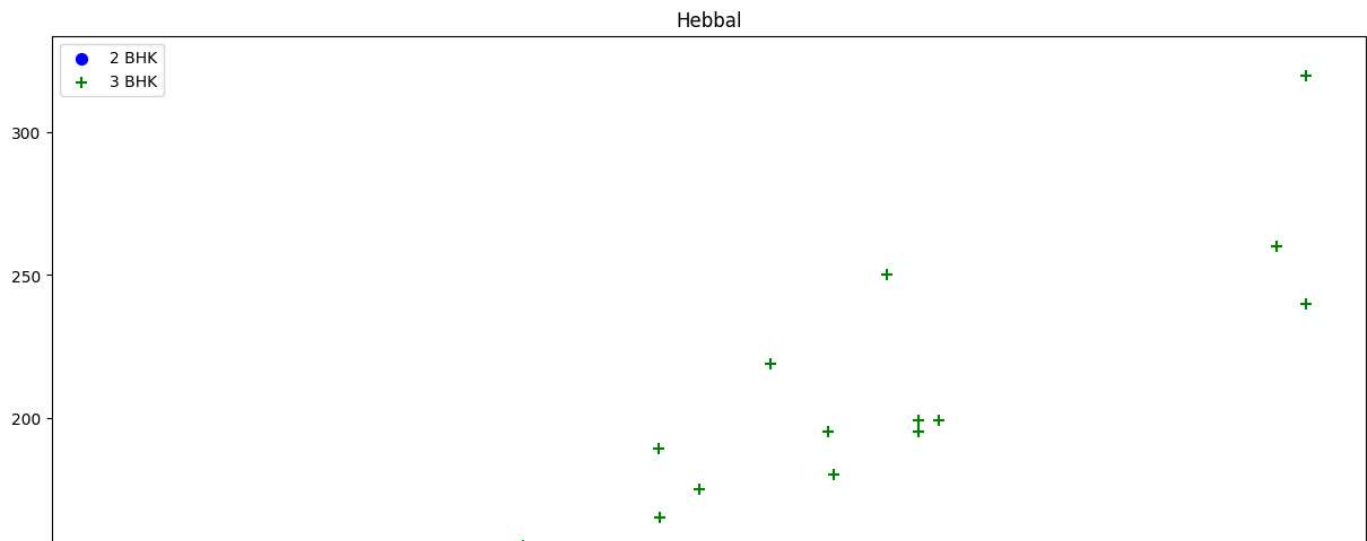
```
def remove_pps_outliers(df):
    df_out = pd.DataFrame()
    for key , subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)
        st= np.std(subdf.price_per_sqft)
        reduced_df = subdf[(subdf.price_per_sqft > (m-st)) & (subdf.price_per_sqft <= (m+st))]
        df_out = pd.concat([df_out , reduced_df],ignore_index = True)
    return df_out
df7=remove_pps_outliers(df6)
df7.shape
```

```
(10241, 7)
```

We removed close to 2000 outliers

```
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 Per Square Feet")
    plt.title(location)
    plt.legend()
```

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



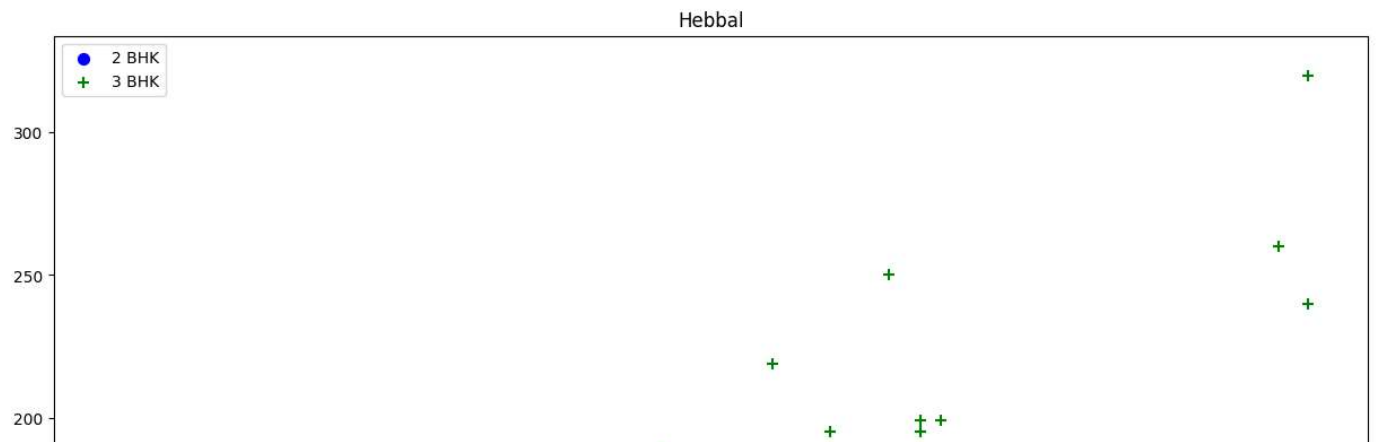
Now we can remove those 2 BHK apartments whose price_per_sqft is less than the mean price_per_sqft of 1 BHK apartment

```
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.shape #more data points removed
```

```
(7329, 7)
```

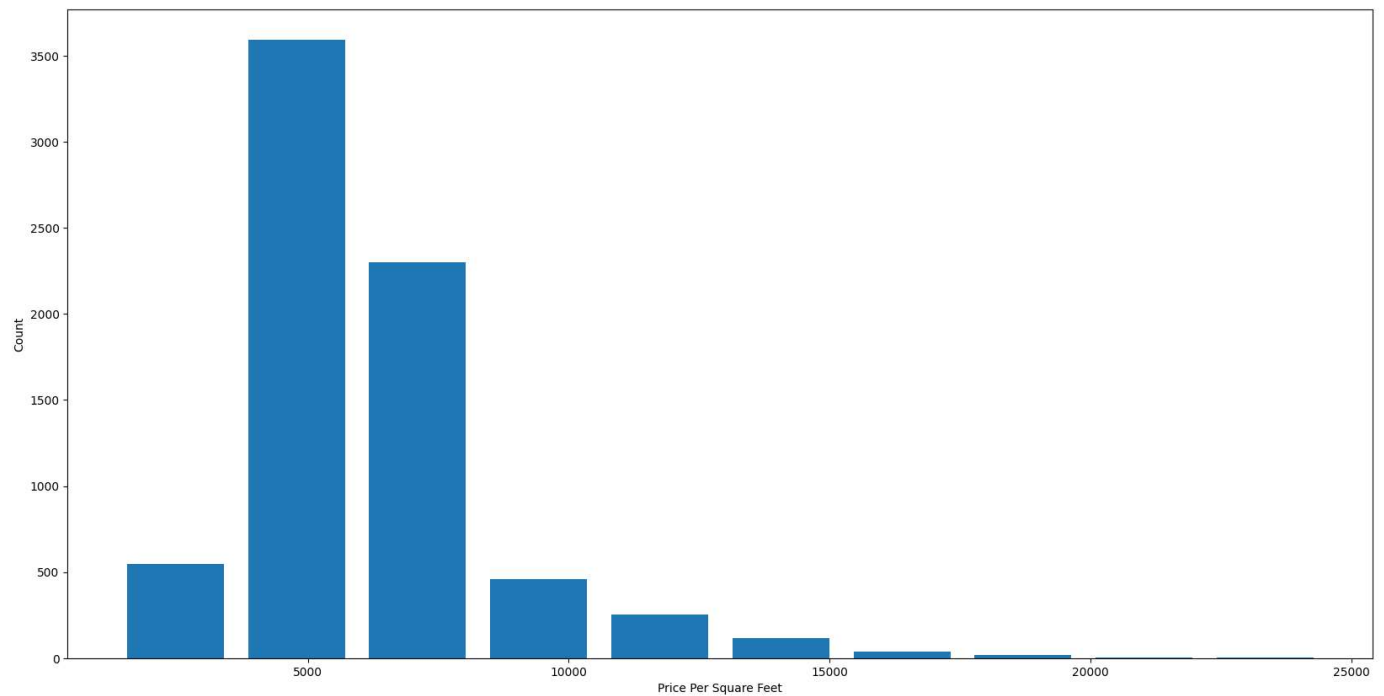
```
plot_scatter_chart(df7,"Hebbal")
```



Abnormalities are still present but less in number

```
matplotlib.rcParams["figure.figsize"] = (20,10)
plt.hist(df8.price_per_sqft, rwidth = 0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

Text(0, 0.5, 'Count')



Now , let's explore the bathroom features

```
df8.bath.unique()
array([ 4.,  3.,  2.,  5.,  8.,  1.,  6.,  7.,  9., 12., 16., 13.])
```

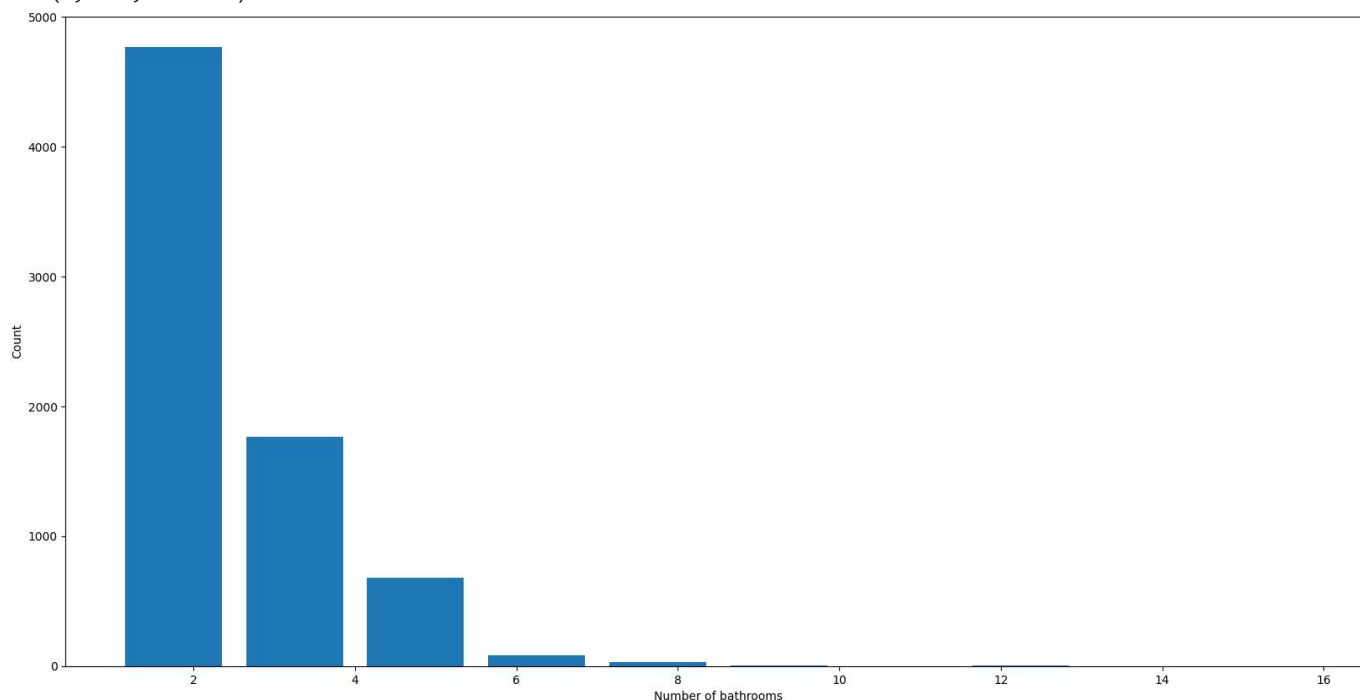
```
df8[df8.bath>10]
```


	location	size	total_sqft	bath	price	bhk	price_per_sqft
5277	Neeladri Nagar	10 BHK	4000.0	12.0	160.0	10	4000.000000
8486	other	10 BHK	12000.0	12.0	525.0	10	4375.000000

Criteria of removing number of bathroom outliers as set by the manager : if no. of bathrooms are more than the no. of bedrooms , then remove it .

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

Text(0, 0.5, 'Count')



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

	location	size	total_sqft	bath	price	bhk	price_per_sqft
1626	Chikkabanavar	4 Bedroom	2460.0	7.0	80.0	4	3252.032520
5238	Nagasandra	4 Bedroom	7000.0	8.0	450.0	4	6428.571429
6711	Thanisandra	3 BHK	1806.0	6.0	116.0	3	6423.034330
8411	other	6 BHK	11338.0	9.0	1000.0	6	8819.897689



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

(7251, 7)

Outlier Detection And Removal is Complete , Now we prepare our data for machine learning (4 finished)

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

```
df10.head(3)
```

	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	

ML can't interpret text data , so we convert into numeric data using one hot encoding

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

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

3 rows × 242 columns

```
df11= pd.concat([df10,dummies.drop('other',axis = 'columns')],axis='columns')
df11.head(3)
```

	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	Vishveshwa Lay
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	0	
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0	...	0	
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0	...	0	

3 rows × 246 columns

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

	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
0	2850.0	4.0	428.0	4	1	0	0	0	0	0	...	0	0
1	1630.0	3.0	194.0	3	1	0	0	0	0	0	...	0	0

2 rows × 245 columns

```
df12.shape

(7251, 245)
```

```
X = df12.drop('price',axis = 'columns')
X.head()
```

	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
0	2850.0	4.0	4	1	0	0	0	0	0	0	...	0	0
1	1630.0	3.0	3	1	0	0	0	0	0	0	...	0	0
2	1875.0	2.0	3	1	0	0	0	0	0	0	...	0	0
3	1200.0	2.0	3	1	0	0	0	0	0	0	...	0	0
4	1235.0	2.0	2	1	0	0	0	0	0	0	...	0	0

5 rows × 244 columns

```
y=df12.price
y.head()
```

```
0    428.0
1    194.0
2    235.0
3    130.0
4    148.0
```

Name: price, dtype: float64

```
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)
```

```
from sklearn.linear_model import LinearRegression
lr_clf = LinearRegression()
lr_clf.fit(X_train,y_train)
lr_clf.score(X_test,y_test)
```

0.8452277697874376

84% is a decent score

```
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 ) #5 fold cross validation
cross_val_score(LinearRegression(),X,y,cv=cv)
cross_val_score
```

```
<function sklearn.model_selection.validation.cross_val_score(estimator, X, y=None, *, groups=None, scoring=None,
cv=None, n_jobs=None, verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score=nan)>
```

#read about k4 corss validation . Here the score is near to 80% always , so it's decent



#We don't know if this regression technique is the best to apply here , so we capply others to see which one works best

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
```

```
def find_bestmodel_using_gridsearchcv(X,y):
    algos = {'linear_regression' :
        {'model' : LinearRegression() , 'params' :{
            'fit_intercept': [True, False],
            'copy_X': [True, False],
            'positive': [True, False],
            'n_jobs': [-1]
        }} ,
        'lasso':
        {'model':Lasso(),'params':{'alpha':[1,2] , 'selection' : ['random','cyclic'] }} ,
        'decision_tree':
        {'model':DecisionTreeRegressor(),'params':
            {'criterion':['friedman_mse','poisson'] , '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_bestmodel_using_gridsearchcv(X,y)
```

	model	best_score	best_params	
0	linear_regression	0.819001	{'copy_X': True, 'fit_intercept': False, 'n_jo...	
1	lasso	0.687429	{'alpha': 1, 'selection': 'cyclic'}	
2	decision_tree	0.728160	{'criterion': 'poisson', 'splitter': 'best'}	

```
X.columns
```

```
Index(['total_sqft', 'bath', 'bhk', '1st Block Jayanagar',
      '1st Phase JP Nagar', '2nd Phase Judicial Layout',
      '2nd Stage Nagarbhavi', '5th Block Hbr Layout', '5th Phase JP Nagar',
      '6th Phase JP Nagar',
      ...,
      'Vijayanagar', 'Vishveshwarya Layout', 'Vishwapriya Layout',
      'Vittasandra', 'Whitefield', 'Yelachenahalli', 'Yelahanka',
      'Yelahanka New Town', 'Yelenahalli', 'Yeshwanthpur'],
      dtype='object', length=244)
```

```
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]
```

```
np.where(X.columns=='2nd Phase Judicial Layout')[0][0]
```

```
5
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lin
warnings.warn(
83.49904677206221
```

```
predict_price('1st Phase JP Nagar',1000,3,3) ##something is not right ,less bathrooms and price is more but it could be th  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lin  
warnings.warn(  
86.80519395233001
```



```
predict_price('2nd Stage Nagarbhavi',1000,2,2)  
  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lin  
warnings.warn(  
182.63400447991722
```



```
predict_price('2nd Stage Nagarbhavi',10000,3,2)  
  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lin  
warnings.warn(  
902.4513947443014
```



```
predict_price('Indira Nagar',1000,2,2)  
  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lin  
warnings.warn(  
181.2781548400639
```



```
predict_price('Indira Nagar',1000,3,3)  
  
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but Lin  
warnings.warn(  
184.5843020203317
```

