House Price Prediction Project

Predicting the prices of the houses in bangalore basing on the location,sqft,size,bedrooms

In the dataset that we are using have both input and output variables thus making the mechine learning algotithm a supervised learning model

Lets try to predict the prices of the houses using Linear regression

In [1]:

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

In [2]:

```
df1 = pd.read_csv("C:\\Users\\saite\\Documents\\MLproject\\Bengaluru_House_Price.csv")
df1.head()
```

Out[2]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	1
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	12
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	0
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	9
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	5

In [3]:

```
df1.columns
```

Out[3]:

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

In [4]:

```
df1.groupby('area_type').agg('count')
```

Out[4]:

area_type	availability	location	size	society	total_sqft	bath	balcony	price
Built-up Area	2418	2418	2418	1215	2418	2410	2310	2418
Carpet Area	87	87	87	54	87	87	82	87
Plot Area	2025	2025	2009	311	2025	2009	1837	2025
Super built-up Area	8790	8789	8790	6238	8790	8741	8482	8790

In [5]:

```
df1['area_type'].value_counts() #returns the unique values within a column and no of val
```

Out[5]:

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

Remove the columns which are not useful for predicting the price

In [6]:

```
df2 = df1.drop(['area_type','society','balcony','availability'],axis=1)
df2.shape
```

Out[6]:

```
(13320, 5)
```

In [7]:

```
df2.head()
```

Out[7]:

	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

Data Cleaning: Handle NA values

In [8]:

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

Out[8]:

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

In [9]:

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

Out[9]:

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

In [10]:

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

Out[10]:

```
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)
```

Feature Engineering Add new feature(integer) for bhk (Bedrooms Hall Kitchen)

In [11]:

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

```
C:\Users\saite\AppData\Local\Temp\ipykernel_20816\2716584372.py:1: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df3['bhk'] = df3['size'].apply(lambda x: int(x.split(' ')[0]))
```

Out[11]:

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

In [12]:

```
df3.head()
```

Out[12]:

	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

In [13]:

```
df3[df3.bhk>20]
```

Out[13]:

	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

In [14]:

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

In [15]:

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

Out[15]:

	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 [16]:

```
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
```

In [17]:

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

Out[17]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3
4	Kothanur	2 BHK	1200.0	2.0	51.00	2

In [18]:

```
df4.iloc[30]
```

Out[18]:

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

Here we are adding a new feature "Price per square feet" which is price divided by total sq feet

In [19]:

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

Out[19]:

	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 [20]:

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

Out[20]:

```
1298
```

since there are 1300 unique locations in the dataset we cant one hot encode them, because that would result in ew 1300 columns this is called Dimensionality curse

In [21]:

```
df5.location = df5.location.apply(lambda x: x.strip()) #removing spaces at strarting if
location_stats = df5['location'].value_counts(ascending=False)
location_stats
```

Out[21]:

```
Whitefield                    533
Sarjapur  Road                392
Electronic City               304
Kanakpura Road                264
Thanisandra                   235
                             ...
Rajanna Layout                  1
Subramanyanagar                 1
Lakshmipura Vidyaanyapura       1
Malur Hosur Road                1
Abshot Layout                   1
Name: location, Length: 1287, dtype: int64
```

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 [22]:

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

Out[22]:

```
BTM 1st Stage                 10
Gunjur Palya                  10
Nagappa Reddy Layout          10
Sector 1 HSR Layout           10
Thyagaraja Nagar              10
                              ..
Rajanna Layout                 1
Subramanyanagar                1
Lakshmipura Vidyaanyapura      1
Malur Hosur Road               1
Abshot Layout                  1
Name: location, Length: 1047, dtype: int64
```

In [23]:

```
df5.head()
```

Out[23]:

	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 [24]:

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

Out[24]:

241

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 [25]:

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

Out[25]:

	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 [26]:

```
df5.shape
```

Out[26]:

(13200, 7)

In [27]:

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

```
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[28]:

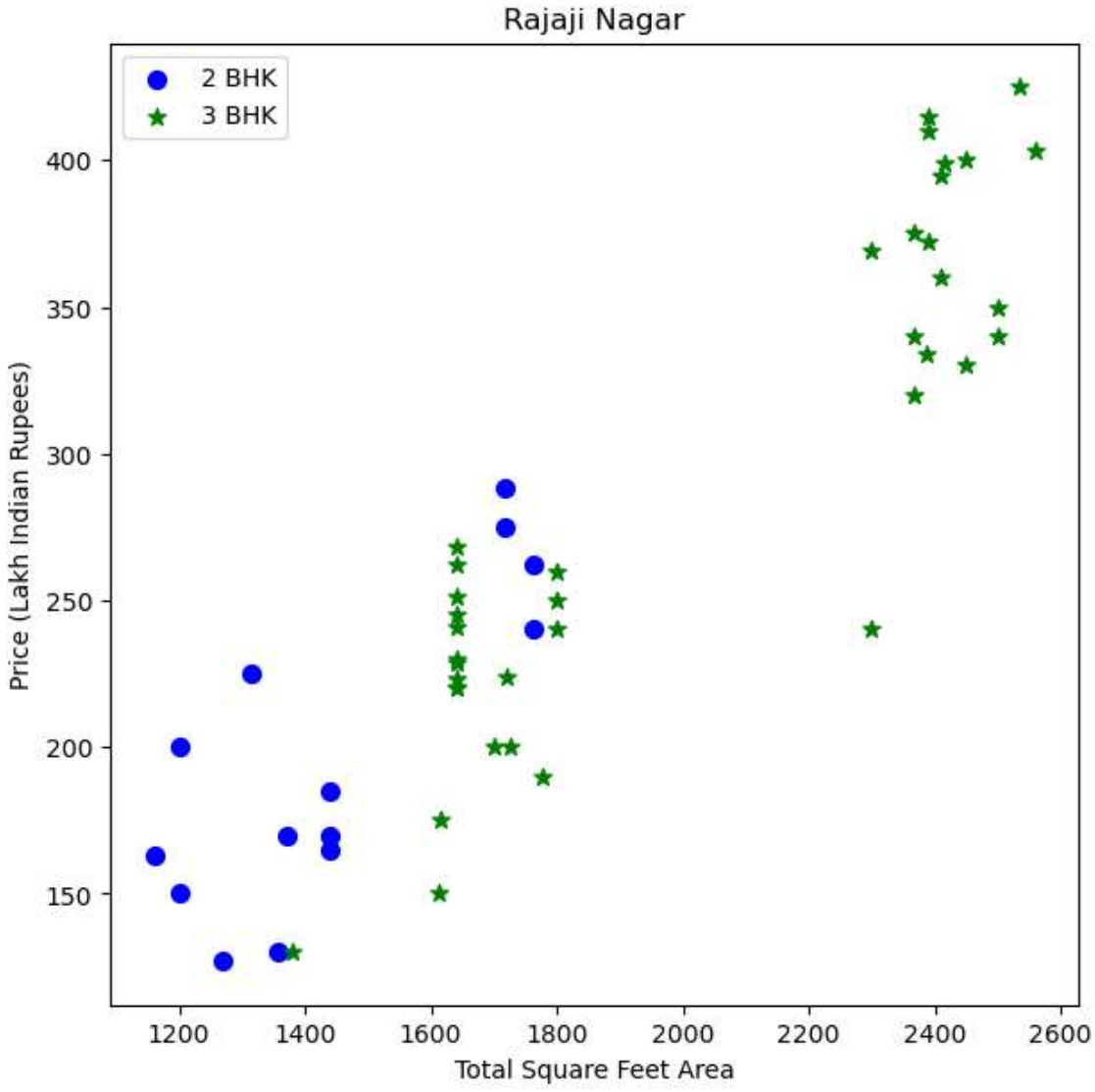
```
(10242, 7)
```

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

In [29]:

```
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'] = (7,7)
    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")
```



Here we can see that in the same location some 3bhk are less than 2bhk, which is an anamoly

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 [30]:

```
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_sqf
    return df.drop(exclude_indices,axis='index')
df8 = remove_bhk_outliers(df7)
df8.shape
```

Out[30]:

```
(7317, 7)
```

In [31]:

```
df8.describe()
```

Out[31]:

	total_sqft	bath	price	bhk	price_per_sqft
count	7317.000000	7317.000000	7317.000000	7317.000000	7317.000000
mean	1493.516501	2.452098	98.839331	2.499932	6126.119223
std	860.566085	1.015090	93.090156	0.926439	2410.348498
min	300.000000	1.000000	10.000000	1.000000	1300.000000
25%	1096.000000	2.000000	50.000000	2.000000	4596.273292
50%	1260.000000	2.000000	73.000000	2.000000	5681.818182
75%	1680.000000	3.000000	112.000000	3.000000	6896.551724
max	30000.000000	16.000000	2200.000000	16.000000	24509.803922

In [ ]:

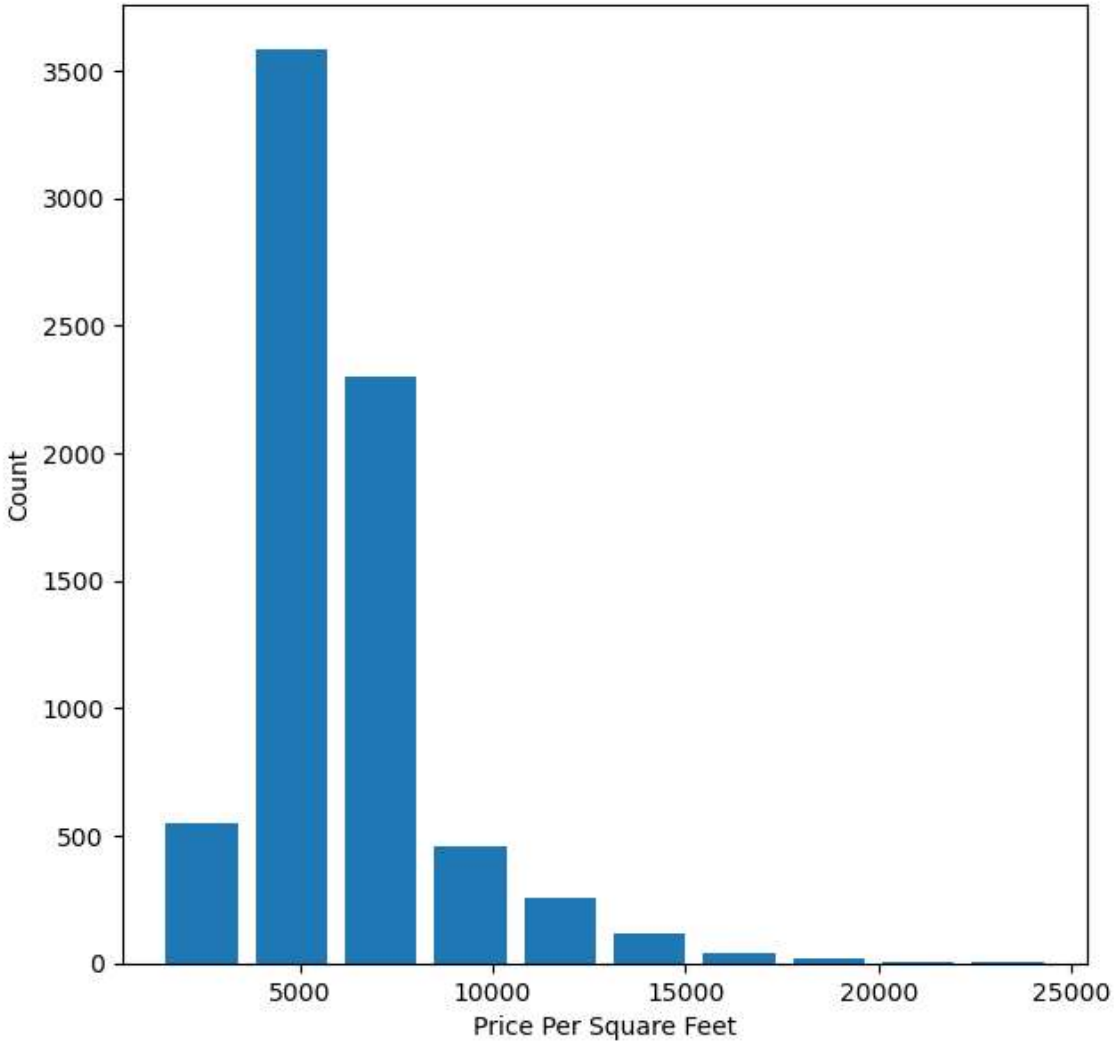
```
```

In [32]:

```
matplotlib.rcParams["figure.figsize"] = (7,7)
plt.hist(df8.price_per_sqft,rwidth=0.8)
plt.xlabel("Price Per Square Feet")
plt.ylabel("Count")
```

Out[32]:

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



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

In [33]:

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

Out[33]:

	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

Again the business manager has a conversation with you (i.e. a data scientist) that if you have 4 bedroom home and even if you have bathroom in all 4 rooms plus one guest bathroom, you will have total bath = total bed + 1 max. Anything above that is an outlier or a data error and can be removed

In [34]:

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

Out[34]:

```
(7239, 7)
```

In [35]:

```
df9.head()
```

Out[35]:

	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
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668

size, price_per_sqft can be dropped since they are not useful for predicting the final pice of the flats

In [36]:

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

Out[36]:

	location	total_sqft	bath	price	bhk
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3
3	1st Block Jayanagar	1200.0	2.0	130.0	3
4	1st Block Jayanagar	1235.0	2.0	148.0	2

In [37]:

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

Out[37]:

	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	9t Phas J Naga
0	1	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	.
10233	0	0	0	0	0	0	0	0	0	
10234	0	0	0	0	0	0	0	0	0	
10237	0	0	0	0	0	0	0	0	0	
10238	0	0	0	0	0	0	0	0	0	
10241	0	0	0	0	0	0	0	0	0	

7239 rows × 241 columns

In [38]:

```
df11 = pd.concat([df10,dummies], axis=1)
df11.head()
```

Out[38]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0
3	1st Block Jayanagar	1200.0	2.0	130.0	3	1	0	0	0	0
4	1st Block Jayanagar	1235.0	2.0	148.0	2	1	0	0	0	0

5 rows × 246 columns

In [ ]:

```

```

In [39]:

```
df11=df11.drop('other',axis=1)
df11.head(2)
```

Out[39]:

	location	total_sqft	bath	price	bhk	1st Block Jayanagar	1st Phase JP Nagar	2nd Phase Judicial Layout	2nd Stage Nagarbhavi	5th Block Hbr Layout
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0

2 rows × 245 columns

In [40]:

```
df12=df11.drop('location',axis=1)
df12.head(2)
```

Out[40]:

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

2 rows × 244 columns

In [41]:

```
df12.shape
```

Out[41]:

```
(7239, 244)
```

Building the Predictive Model

In [42]:

```
x = df12.drop('price',axis=1) # x is input variable which is all columns except target v
x.head(1)
```

Out[42]:

	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	...
0	2850.0	4.0	4	1	0	0	0	0	0	0	...

1 rows × 243 columns

In [43]:

```
y =df12.price
y.head(2)
```

Out[43]:

```
0    428.0
1    194.0
Name: price, dtype: float64
```

In [44]:

```
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 [45]:

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

Out[45]:

```
0.8629132245229442
```

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

In [46]:

```
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[46]:

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

Test the model for few properties

In [49]:

```
x.head()
```

Out[49]:

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

5 rows × 243 columns

In [50]:

```
def predict_price(location,sqft,bath,bhk):
    loc_index = np.where(x.columns==location)[0][0]

    z = np.zeros(len(x.columns))
    z[0] = sqft
    z[1] = bath
    z[2] = bhk
    if loc_index >= 0:
        z[loc_index] = 1

    return lr_clf.predict([z])[0]
```

In [51]:

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

```
C:\Users\saite\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
```

Out[51]:

```
83.8657025831235
```

In [52]:

```
predict_price('1st Phase JP Nagar',1000, 3, 3)
```

```
C:\Users\saite\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
```

Out[52]:

```
86.08062284987108
```

In [53]:

```
predict_price('Indira Nagar',1000, 3, 3)
```

```
C:\Users\saite\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarni
ng: X does not have valid feature names, but LinearRegression was fitted
with feature names
  warnings.warn(
```

Out[53]:

```
195.52689759854718
```

In [54]:

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

In [55]:

```
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 [ ]:

```

```