Machine Learning Project : Predicting Home Prices in Banglore

In [1]:

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

Data Load: Load Banglore Home Prices into a Dataframe

In [2]:

```
df=pd.read_csv("downloads/bengaluru_house_prices.csv")
df.head()
```

Out[2]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

In [3]:

```
df.groupby('area_type')['area_type'].agg('count')
```

Out[3]:

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

In [4]:

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

Out[4]:

	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 [5]:

```
df2=df1.dropna()
df2.isnull().sum()
```

Out[5]:

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

Feature Engineering

In [6]:

```
df2['bhk']=df2['size'].apply(lambda x : int(x.split(' ')[0]))
```

```
<ipython-input-6-7d950b2d6685>: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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  df2['bhk']=df2['size'].apply(lambda x : int(x.split(' ')[0]))
```

In [7]:

```
df2.drop('size',axis="columns")
```

Out[7]:

	location	total_sqft	bath	price	bhk
0	Electronic City Phase II	1056	2.0	39.07	2
1	Chikka Tirupathi	2600	5.0	120.00	4
2	Uttarahalli	1440	2.0	62.00	3
3	Lingadheeranahalli	1521	3.0	95.00	3
4	Kothanur	1200	2.0	51.00	2
...	...	...	...	...	...
13315	Whitefield	3453	4.0	231.00	5
13316	Richards Town	3600	5.0	400.00	4
13317	Raja Rajeshwari Nagar	1141	2.0	60.00	2
13318	Padmanabhanagar	4689	4.0	488.00	4
13319	Doddathoguru	550	1.0	17.00	1

13246 rows × 5 columns

In [8]:

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

In [9]:

```
df2[~df2['total_sqft'].apply(is_float)].head()
```

Out[9]:

	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

In [10]:

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

In [11]:

```
df3=df2.copy()
df3['total_sqft']=df3['total_sqft'].apply(convert)
```

In [12]:

```
df4=df3.copy()
df4['price_per_sqft']=df4['price']*100000/df4['total_sqft']
```

In [13]:

```
df4.location= df4.location.apply(lambda x : x.strip())
location_stats= df4.groupby('location')['location'].agg('count').sort_values(ascendi
location_stats
```

Out[13]:

```
location
Whitefield            535
Sarjapur  Road        392
Electronic City       304
Kanakpura Road        266
Thanisandra           236
                     ... 
LIC Colony              1
Kuvempu Layout          1
Kumbhena Agrahara       1
Kudlu Village,          1
1 Annasandrapalya       1
Name: location, Length: 1293, dtype: int64
```

Dimensionality Reduction

In [14]:

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

In [15]:

```
df4.location = df4.location.apply(lambda x : 'others' if x in location_stats_less_th
```

Outlier Removal

In [16]:

```
df5 = df4[~(df4.total_sqft/df4.bhk<300)]
```

In [17]:

```
df5.shape
```

Out[17]:

```
(12502, 7)
```

In [18]:

```
def remove_outliers_price(df):
    df_final=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+s
        df_final= pd.concat([df_final,reduced_df],ignore_index = True)
    return df_final
```

In [19]:

```
df5.shape
```

Out[19]:

```
(12502, 7)
```

In [20]:

```
df6 = remove_outliers_price(df5)
```

In [21]:

```
df6.shape
```

Out[21]:

```
(10241, 7)
```

In [22]:

```
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
    return df.drop(exclude_indices,axis='index')
df7 = remove_bhk_outliers(df6)
df7.shape
```

Out[22]:

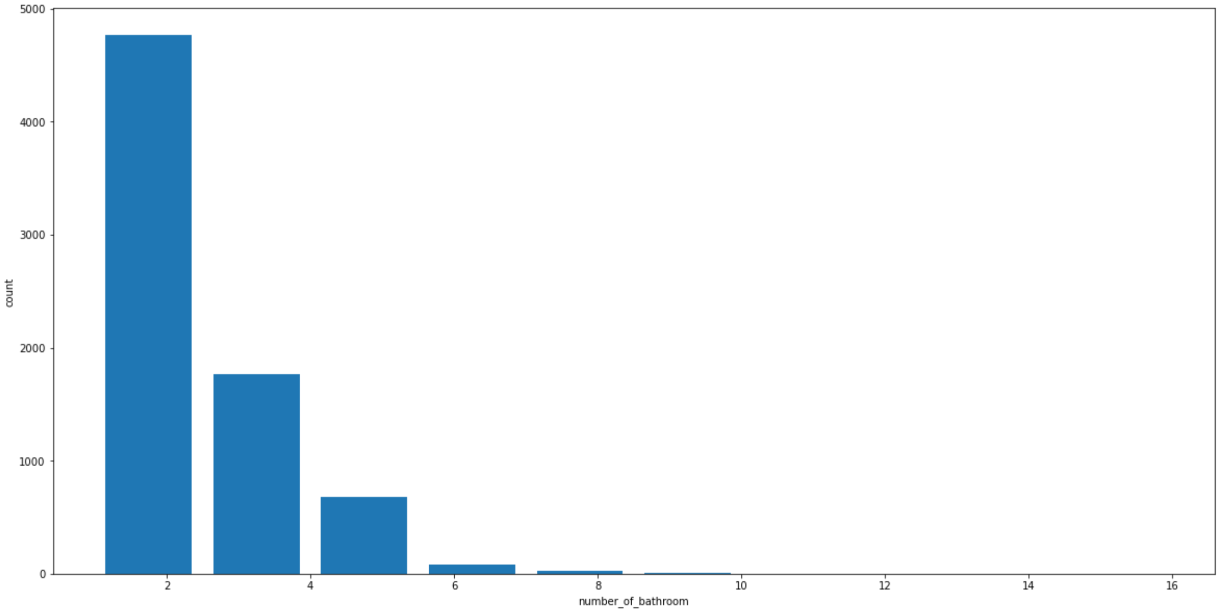
```
(7329, 7)
```

In [23]:

```
plt.hist(df7.bath,rwidth=0.8)
plt.xlabel("number_of_bathroom")
plt.ylabel('count')
```

Out[23]:

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



In [24]:

```
df8 = df7[df7.bath < df7.bhk+2]
df8.shape
```

Out[24]:

```
(7251, 7)
```

In [25]:

```
df9 = df8.drop("price_per_sqft",axis ="columns")
df9.head()
```

Out[25]:

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

Use One Hot Encoding For Location

In [26]:

```
dummies = pd.get_dummies(df9.location)
dummies.head()
```

Out[26]:

	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	...	Vishve
0	1	0	0	0	0	0	0	0	0	0	...	
1	1	0	0	0	0	0	0	0	0	0	...	
2	1	0	0	0	0	0	0	0	0	0	...	
3	1	0	0	0	0	0	0	0	0	0	...	
4	1	0	0	0	0	0	0	0	0	0	...	

5 rows × 242 columns

In [27]:

```
df10= pd.concat([df9,dummies.drop('others',axis="columns")],axis = "columns")
df10.head()
```

Out[27]:

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

5 rows × 247 columns

In [28]:

```
df11= df10.drop(['location','size'],axis = "columns")
df11.head(2)
```

Out[28]:

	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	...	Vijayana
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 × 245 columns

In [29]:

```
X = df11.drop('price',axis="columns")
X.head()
```

Out[29]:

	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	...	Vijayai
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 × 244 columns

In [30]:

```
Y = df11.price
Y.head()
```

Out[30]:

```
0    428.0
1    194.0
2    235.0
3    130.0
4    148.0
Name: price, dtype: float64
```

Build a Model

In [31]:

```
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_model = LinearRegression()
lr_model.fit(x_train,y_train)
lr_model.score(x_test,y_test)
```

Out[31]:

```
0.8452277697874312
```

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

In [32]:

```
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[32]:

```
array([0.82430186, 0.77166234, 0.85089567, 0.80837764, 0.83653286])
```

In [33]:

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

    x = np.zeros(len(X.columns))
    x[0]=total_sqft
    x[1]=bath
    x[2]=bhk
    if loc_index >=0:
        x[loc_index] = 1
    return lr_model.predict([x])[0]
```

Price Prediction

In [34]:

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

Out[34]:

```
83.49904677179224
```

In [35]:

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

Out[35]:

```
86.80519395205835
```

In [36]:

```
predict_price('Indira Nagar',1000, 2, 2)
```

Out[36]:

```
181.27815484006857
```

Export the tested model to a pickle file

In [37]:

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

In [38]:

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

The End

In [ ]: