REAL ESTATE PRICE PREDICTION

by

Usama Fazal

A Project Report submitted to the
DEPARTMENT OF COMPUTER SCIENCE
in partial fulfillment of the requirements for the degree of
BACHELORS OF SCIENCE IN COMPUTER SCIENCE

Faculty of Computing
Capital University of Science & Technology,
Islamabad

Table of Contents

CHAPTER 1

INTRODUCTION

Introduction of Project:

Machine Learning is a vital aspect of present-day business and research. It progressively improves the performance of computer systems by using algorithms and neural network models. Machine Learning algorithms automatically build a mathematical model using sample data also referred to as “training data” which form decisions without being specifically programmed to make those decisions. People and real estate agencies buy or sell houses, people buy to live in or as an investment and the agencies buy to run a business. Either way, we believe everyone should get exactly what they pay for. over-valuation and under-valuation in housing markets has always been an issue and there is a lack of proper detection measures. Broad measures, like house/Real-estate price-to-rent ratios, give a primary pass. However, to decide about this issue an in-depth analysis and judgment are necessary. Here’s where machine learning comes in, by training an ML model with hundreds and thousands of data a solution can be developed which will be powerful enough to predict prices accurately and can cater to everyone’s needs.

Our system is a real-life data science product and we are using the concepts like:

1) Data Cleaning
2) Feature Engineering
3) One Hot Encoding
4) Outlier Detection
5) Dimensionality Reduction
6) Model Building

Explanation of Dataset:

First of all, we have nine features in our dataset and 13000 records and price is the main feature that we have to predict and it is also called the class attribute and the other features are:

1) Area Type
2) Location
3) Availability
4) Size
5) Society
6) Balcony
7) Bath
8) Total_sqft
9) Price

In Area type feature we have different we have different areas such as Super Built-up are and plot area and also there is a society feature that have different societies like Jades, Soiewre, Coomee, Theanmp and also there is the other feature size which tells us that how many rooms are there in the house and also No of bathrooms is the feature and total square feet of the house is the main feature that tells us the house covers how much area and also the other feature is balcony that how many balconies are present and location feature in the data set is very important because it affects the price much more than the other feature and availability is also an other feature but according to me it is not very important feature in predicting the price.

Also, in our data set we have null values and too much large values which are called outliers and they affect the price feature and our prediction will be wrong. So, before predicting the price we perform “Data Pre-Processing” and through this we clean our data by removing the null values and outliers by outlier detection and also do the dimensionality reduction and at last from the clean data build a model and perform analysis on different types of machine learning model and check that which will gives the best score values and who will predict the price better and accurate.

Also, the data set we choose is of the Bengaluru city and of different locations and different home prices information is present in dataset.

CHAPTER 2

METHODOLOGY

Data Preprocessing:

Data preprocessing is a step in the data mining and data analysis process that takes raw data and transforms it into a format that can be understood and analyzed by computers and machine learning. Raw, real-world data in the form of text, images, video, etc., is messy. Not only may it contain errors and inconsistencies, but it is often incomplete, and doesn't have a regular, uniform design. Machines like to process nice and tidy information they read data as 1s and 0s. So, calculating structured data, like whole numbers and percentages is easy. However, unstructured data, in the form of text and images must first be cleaned and formatted before analysis.

Importance of Data Preprocessing:

When using data sets to train machine learning models, you'll often hear the phrase "garbage in, garbage out" This means that if you use bad or "dirty" data to train your model, you'll end up with a bad, improperly trained model that won't actually be relevant to your analysis. Good, preprocessed data is even more important than the most powerful algorithms, to the point that machine learning models trained with bad data could actually be harmful to the analysis you're trying to do giving you "garbage" results.

Data Cleaning:

We are using supervised learning and we use on data set basically the input and output value. Libraries like pandas are basic block for cleaning the data. Import basic libraries like python, matplotlib and NumPy. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries

First of all, **shape ()** function is used to check the columns and rows in the data.

```
[4]  df1.shape

     (13320, 9)
```

Secondly, we drop some features **'area_type','society','balcony','availability'** and they are not necessary in predicting the price. After removing the features, the head will be like this:

```
[8]  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

Thirdly, we have to remove the Null values so we find the aggregate count of all features values which are null by **df2.isnull(). sum ()** and the output will be like this:

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

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

As these values are so small and our date set is too large so remove these values and this will be:

```
[10] df3=df2.dropna()
```

```
[11] df3.isnull().sum()

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

```
[12] df3.shape

(13246, 5)
```

Now in size feature we two type of values like 4 bedrooms and bhk so we separate those values and made a new feature called bhk.

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

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py: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
  """Entry point for launching an IPython kernel.
```

```
[15] 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])
```

We are having the range in some cases in total_sqft not the single value so we have to convert this into a single number and one of the ways is to take the average of these two numbers. We are checking the values are in float or not?. If the values are not in float, they will be returning false. Just apply **is float ()** function.

Now in total_sqft we got all the ranges in which we will get cases like which are not unform and they are unstructured containing the outliers.

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

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

We are using the **convert_sqft_to_num ()** to overcome these problems. We take the particular range and then convert it into the single numeric value. We are creating the new data frame. Now the data set will be looking more accurate as compared to before.

```
[23] convert_sqft_to_num('2100-21850')

     11975.0
```

```
[24] df4=df3.copy()
     df4['total_sqft']= df4['total_sqft'].apply(convert_sqft_to_num)
     df4.head(3)
```

	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

Feature Engineering:

We are copying the data frame into a new data frame by using **copy ()** function. We are doing the feature engineering to make the other steps further easy. So, to check we are finding unique location by calling the **location. unique ()** function and grouping it by calling the dfs. groupby () function for the location with aggregation function.

Also, we made a new feature here which is price per square feet.

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

Also, now in locations We have 1300 columns for locations and if we keep them than this is called curse of dimensionality and that's why we use a different technique and some locations there will be one or two data points and we called them other locations.

```
[35] 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)
```

	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

Outlier Detection:

We are just detecting and then removing the outliers, sometimes they are just not data errors but also, they are causing extreme variation in the dataset. We are using different techniques to detect the outliers like STANDARD DEVIATION or SIMPLE DOMAIN KNOWLEDGE.

Data errors are clearly visible in the dataset. So, for removing them we are creating a new dataset and filtering the rows according to the criteria like calling the function **dfs[~(dfs.total_sqft/dfs.bhk<300)]**

	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

After calling it the number of rows is clearly reduced. Now describing the price per square foot, we are receiving the maximum. minimum, 25%,75%, mean etc. values all over the output. According to these properties there are some extreme cases we have to remove these extreme cases on the basis of NORMAL DISTRIBUTION.

```
[40] #Remove some outliers
     df6 = df5[~(df5.total_sqft/df5.bhk<300)]
     df6.shape

     (12502, 7)

[41] 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
```

To remove the outliers the function, **remove_pps_outliers ()** is used. We are taking data frame as an input and grouping them by location first and by it we are calculating mean and standard deviation. So, anything above means -1 standard deviation and below mean +10 standard deviation is kept in the reduced data frame and appending these data frames which is getting the output data frame.

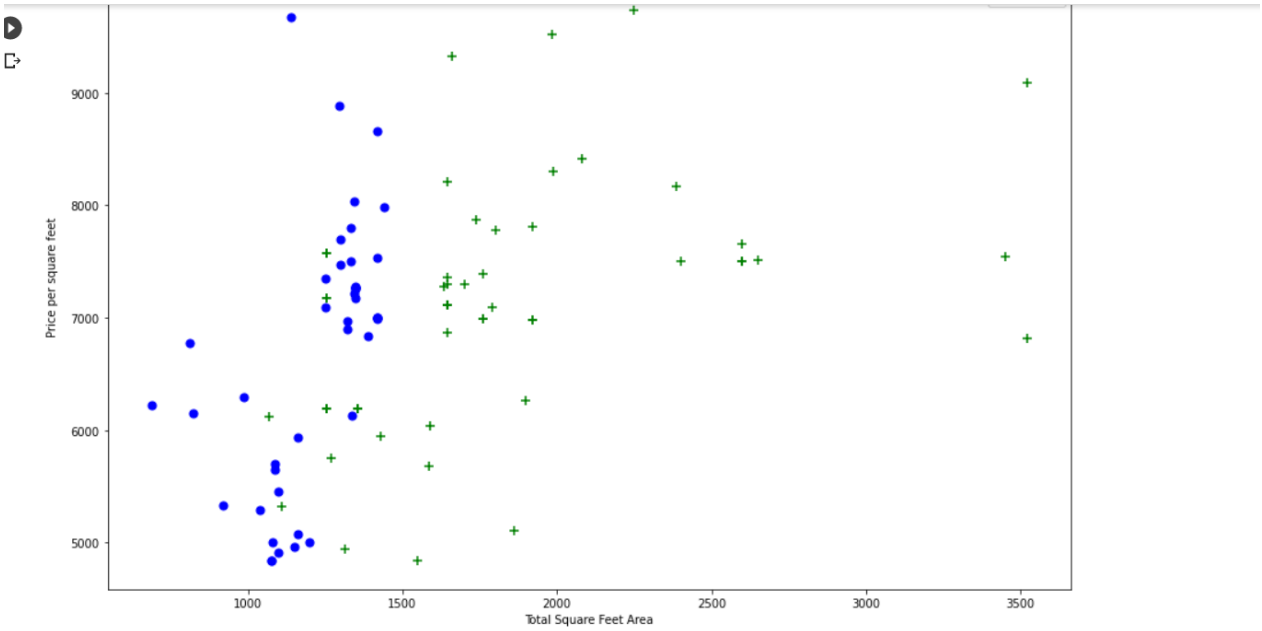
```
[42] def remove_pps_outliers(df):
       df_out=pd.DataFrame()
       for key , subdf in df.groupby('location'):
         m=np.mean(subdf.price_per_sqft)
         st=np.std(subdf.price_per_sqft)
         reduced_df = subdf[(subdf.price_per_sqft>(m-st)) & (subdf.price_per_sqft<=(m+st)) ]
        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

     df7= remove_pps_outliers(df6)
     df7.shape
```

```
(10241, 7)
```

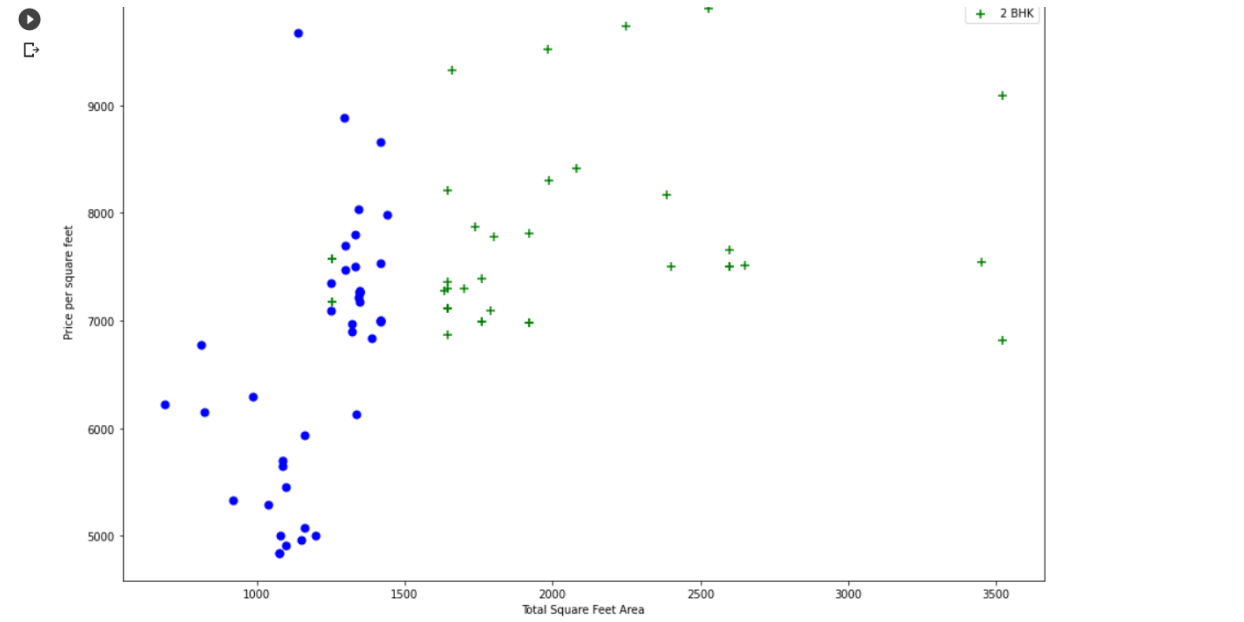
As the price in some cases of two bed apartments is greater than the three bed apartment and also there is a location factor. So, we can't predict this so we use scatter plot for the visualization.

Now some data is having same property specs but different prices, this could e due to any reason like of location or anything as visible in our data set. To visualize this, we just have to use the scatter plot function like it is **plot_scattter_chart ():** it will draw the scatter plot on the suppose bhk 2 or 3 apartments. Two different colored dots will be appeared showing 2- and 3-bedrooms apartments respectively on x and y axis. This plot will show some 2 bed rooms apartments will be having more prices than the 3 bedrooms apartments. We are removing these because they are the outliers. To remove these outliers, we made a dictionary which is calculating mean STD and count. We are comparing it with second apartment and seeing if the value of it is less than the mean. The function used for this purpose is remove bhk_outliers (): for the location of BHK.



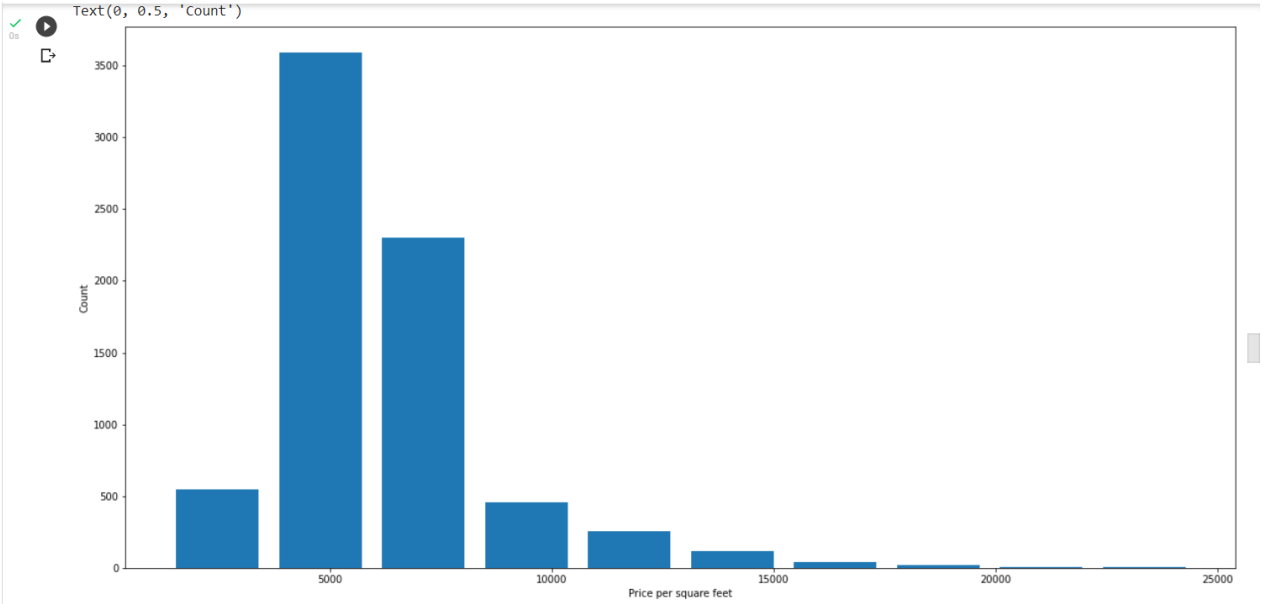
✓ 0s completed at 9:12 AM

We are running the same for loop again and again to check the values of previous BHK. After running this function, we will see many more data is removed from the dataset which is the outlier basically. Now again calling the scatter plot we will watch that many green data points are disappeared.



After removing the outliers, we have to produce a histogram with the help of imported library matplotlib and calling the function **plt. hist ():** we can give any parameters in this function as

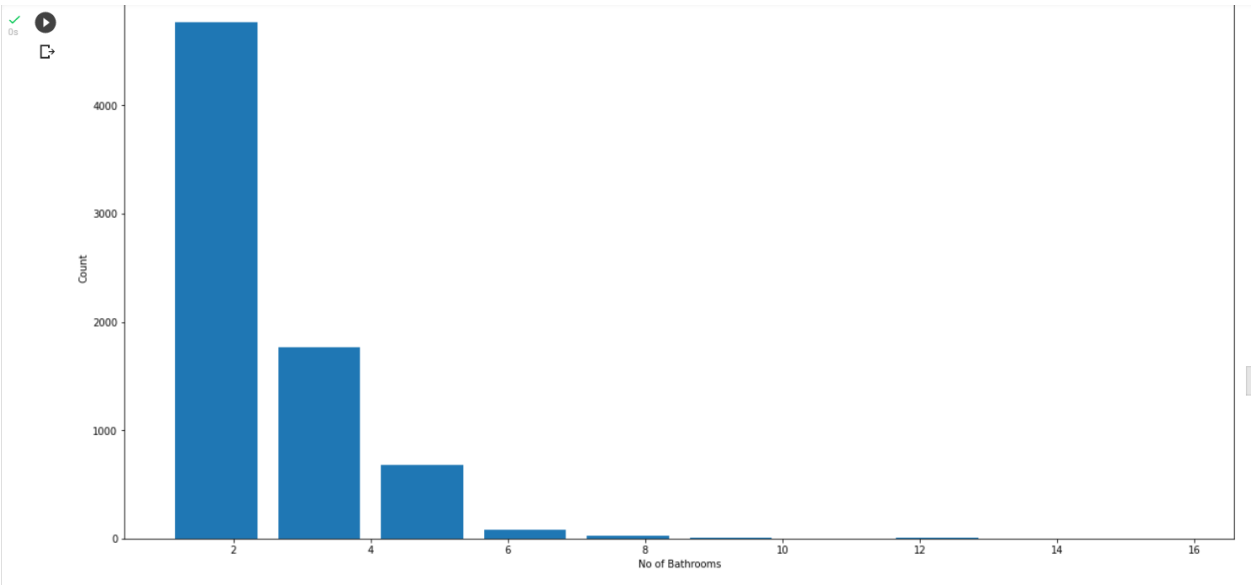
if we want to produce the histogram of some other features. After this function the histogram will be appeared which is showing number of points are lying at one bar. This is the normal distribution.



In the category of bathroom, we are having very less outliers in the histogram. Most of the properties are having two beds and two bathrooms. Greater than this criterion we are marking them outliers and we can safely remove them. Now we are having the new data frame and now it is much looking neat and clean. BHK and price per sqft can now be dropped because they are used for detecting outliers and we have done it.

```
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("No of Bathrooms")
plt.ylabel("Count")
```

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



Now our model is ready for machine learning.

Model Building:

We are building the machine learning model and then using the K fold cross validation and grid search CV to come up with the best algorithm and the best parameter.

Machine learning model can't interpret the text data. So, we have a text data in our data set (location) and we have to convert it into the numeric values.

We are using the pandas dummy model to do it, the function called is like pd.get_dummies (df10.location). as the feature location is having the data in the form of text so this function will convert this data into the numeric values. When we have the 1st location, all the column will be 1 and every other value in data set is zero.

```
dumies = pd.get_dummies(df10.location)
dumies.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	Vittasandra	Whitefield	Yel
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 × 242 columns

We will calculate the new data frame and we are concatenating the df11, df10 and dummies. Dummies will be stored in separate data frame. We are dropping the last column because we can live without it. We are having the new data frame.

```
[56] df11=pd.concat([df10,dumies.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	Vishveshwarya Layout	Vishwapriya Layout	Vittasand
0	1st Block Jayanagar	2850.0	4.0	428.0	4	1	0	0	0	0	...	0	0	0	
1	1st Block Jayanagar	1630.0	3.0	194.0	3	1	0	0	0	0	...	0	0	0	
2	1st Block Jayanagar	1875.0	2.0	235.0	3	1	0	0	0	0	...	0	0	0	

3 rows × 246 columns

We are creating an X variable which Is having the only independent variables so the dependent variable price will be dropped from this data frame. Now head () will not be having the price feature. And the variable Y with the only variable price.

```
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	Vishwapriya Layout	Vittasandra
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	1200.0	2.0	3	1	0	0	0	0	0	0	...	0	0	0	0
4	1235.0	2.0	2	1	0	0	0	0	0	0	...	0	0	0	0

5 rows × 244 columns

```
[60] y=df12.price
     y.head()
```

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

Now for training data set we are importing the train_test_split () from sklearn. model selection

20% will be the test samples and 80% will be the training dataset. We are checking the fitting and scoring of the model with the help of functions in the part of linear regression by importing the library of Linear Regression from sklearn. linear model.

Some needed methods are also shuffle split and cross_val_score so the target must not be the same area using shuffle split.

We are using the method grid search CV that is a very good API that sklearn provide which can learn your model on different parameters. So by importing the GridSearchCV from sklearn We wrote a function combined of linear regression, lasso and decision tree and at the last we are seeing the scores and comparing them into the data frame. So, the maximum score is given from linear regression which is the best one to apply. So, we are using the LR.

FutureWarning,

	model	best_score	best_params
0	linear_regression	0.818354	{'normalize': False}
1	Lasso	0.687610	{'alpha': 1, 'selection': 'random'}
2	decision_tree	0.722018	{'criterion': 'friedman_mse', 'splitter': 'best'}

We are predicting the price by using the predict price function which is having the parameters as location, sqft, bath and bhk. In this function giving the location exact with the sqfts and bath and bhk we will be receiving the cost of the apartment. But like for same area if we are having bathrooms more than the price is definitely going to be high. So that's why there are some variations in the models' outcomes.

```
[66]
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]
```

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

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LinearRegression w
  "X does not have valid feature names, but"
83.49904677176957
```

CHAPTER 3

IMPLEMENTATION

Code:

Call libraries and read the csv file

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['figure.figsize']
df1=pd.read_csv('./Bang.csv')
print(df1)
df1.shape
```

Data Cleaning

```
df1.shape
df1.groupby('area_type')['area_type'].agg('count')
df2=df1.drop(['area_type','society','balcony','availability'],axis='columns')
print(df2)
df2.isnull().sum()
df3=df2.dropna()
df3['size'].unique()
df3['bhk']=df3['size'].apply(lambda x: int(x.split(' ')[0]))
df3['bhk'].unique()
df3[df3.bhk>20]
df3.total_sqft.unique()
def is_float(x):
  try:
    float(x)
  except:
  return False
```

```
  return True
df3[~df3['total_sqft'].apply(is_float)].head(10)
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
df4=df3.copy()
df4['total_sqft']= df4['total_sqft'].apply(convert_sqft_to_num)
df4.head(3)
```

NOW FEATURE ENGINEERING STARTS

```
df5= df4.copy()
df5['price_per_sqft']=df5['price']*100000/df5['total_sqft']
df5.head()
df5.location.unique()
len(df5.location.unique())
df5.location=df5.location.apply(lambda x: x.strip())
location_stats =
df5.groupby('location')['location'].agg('count').sort_values(ascending=False)
location_stats
len(location_stats[location_stats<=10])
len(df5.location.unique())
df5.location= df5.location.apply(lambda x: 'other' if x in location_stats_less_than_10 else
x)
len(df5.location.unique())
df5.head(10)
```

OUTLIER DETECTION AND MODEL BUILDING

```
df5[df5.total_sqft/df5.bhk<300].head()
df5.shape
df6 = df5[~(df5.total_sqft/df5.bhk<300)]
df6.shape
df6.price_per_sqft.describe()
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
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_per_sqft, color='blue', label='2 BHK',s=50 )
plt.scatter(bhk3.total_sqft,  bhk3.price_per_sqft,marker='+',  color='green',  label='2
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")
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

plot_scatter_chart(df8,"Hebbal")
matplotlib.rcParams["figure.figsize"]=(20,10)
plt.hist(df8.price_per_sqft,rwidth=0.8)
plt.xlabel("Price per square feet")
plt.ylabel("Count")
plt.hist(df8.bath,rwidth=0.8)
plt.xlabel("No of Bathrooms")
plt.ylabel("Count")
df8[df8.bath>df8.bhk+2]
df9=df8[df8.bath<df8.bhk+2]
df9.shape
```

```
df10=df9.drop(['size','price_per_sqft'], axis='columns')
df10.head()
dumies = pd.get_dummies(df10.location)
 dumies.head(3)
df11=pd.concat([df10,dumies.drop('other',axis='columns')],axis='columns')
df11.head(3)
df12=df11.drop('location',axis='columns')
df12.head(3)
df12.shape
X=df12.drop('price',axis='columns')
X.head()
y=df12.price
y.head()
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.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)
import sklearn
from pandas.core.common import random_state
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)
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
from sklearn.pipeline import make_pipeline
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)
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]
```

REFERENCES

[1]. https://www.ijert.org/research/real-estate-price-prediction-IJERTV10IS040322.pdf, last visited 3rd July, 2022.