**Mini-Project Report**

\

|  |  |
| --- | --- |
| Department: | Mathematical and Computational Sciences |
| Specialization: | Computational and Data Science |
| Subject: | Machine Learning |
| Topic: | House Price Prediction Using Machine Learning |
| Course Instructor: | Dr. Jidesh P. |
| Date: | 08/05/2021 |



**Team Number 2**

* Gavali Deshabhakt Nagnath - 202CD005
* Mohammad Ahsan - 202CD016
* Shimpi Mayur Anil - 202CD027

***Index***

|  |  |  |
| --- | --- | --- |
| *Chapter No.* | *Chapter* | *Page No.* |
| I | Introduction & Methodology | 3 |
| II | Downloading Database |  |
| III | Data Pre-processing | 8 |
| IV | Feature Engineering | 13 |
| V | Removing Outliers | 19 |
| VI | Performing One-Hot-Encoding | 30 |
| VII | Building Machine Learning Model | 31 |
| VIII | Making GUI |  |
|  | References |  |
|  | Entire Python Code |  |

Chapter - I

**Introduction and Methodology**

Chapter - II

**Downloading Database**

The data base we used in the project is available on ‘Kaggle’ website.

The database containing useful attributes of Bangalore houses which help in developing a machine learning model for house prediction.

The database contains attributes as columns such as-

**‘area\_ type’:** 'Super built-up ', 'Plot', 'Built-up ', 'Carpet' areas.

**‘location’:** All important locations of Bangalore where house is available.

**‘size’:** It describe the type of apartment it is like: 2BHK, 3BHK, 2 Bedrooms etc.

**‘baths’:** This column describes the number of bathrooms that house have, which helps in removing the outliers.

**‘total\_sqft’:** It describe the area size of the house.

**‘price’:** Finally, the house price which act as dependent variable for training and testing of our machine learning model.

**‘Others’:** There are some unnecessary columns such as ‘availability’, ‘society’, ’balcony’, which we removed as they do not help in deciding the price of the house.

**The link to download the Bangalore-house database is available here:**

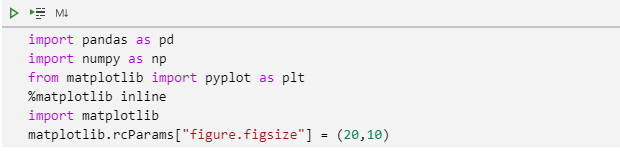
<https://www.kaggle.com/amitabhajoy/bengaluru-house-price-data>

Chapter - III

**Data Pre-processing**

We have done our project using ‘Jupyter Notebook’ in python programming, which includes importing libraries such as ‘pandas’, ’numpy’, ‘matplotlib’, etc.

**Importing Libraries:**

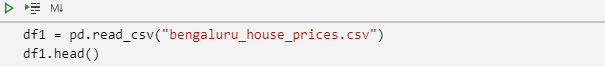
****

**Pandas:** Pandas helps in data manipulation and analysis.

**Numpy:** Numpy library comes handy while doing large sized matrix operations along with some important mathematical tools.

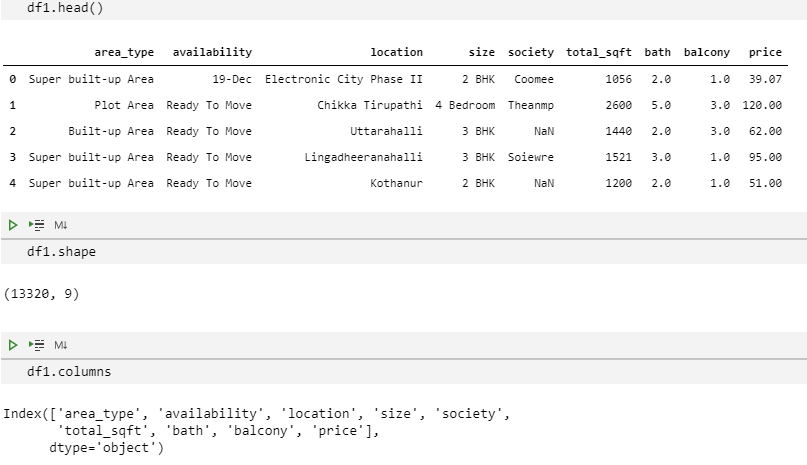
**Matplotlib:** This library provides sufficient functions to plot graphs which is very much required for analysis and understanding of large problem sets.

**Loading ‘Bangalore\_House\_Price’ database:**

****

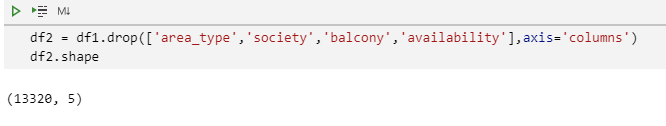
We here import our ‘bangaluru\_house\_price.csv’ database to our python program, using pandas.read\_csv().

**Structure of the imported database:**



As you can clearly see here that there are ‘13320 rows’ and ‘9 columns’ available with our database, which even contains some of the unnecessary rows and thus required cleaning of the data to reduce the dimension and speed the processing capability.

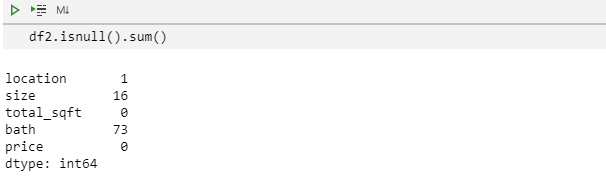
**Dropping unnecessary columns:**

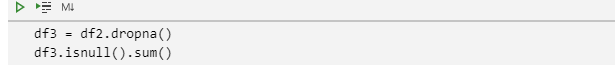
****

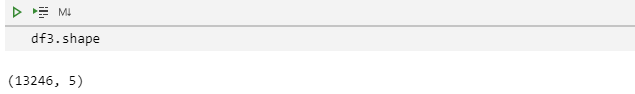
Here we dropped columns such as ‘area\_type’, ‘society’, ‘balcony’, ‘availability’

from our database, because these were not needed for analysis.

**Checking for ‘Null’ values in database:**







As we can see that there exist some ‘Null’ values in respective columns. As they are few in number hence can be safely removed from database.

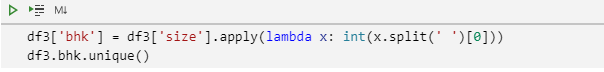
Chapter - IV

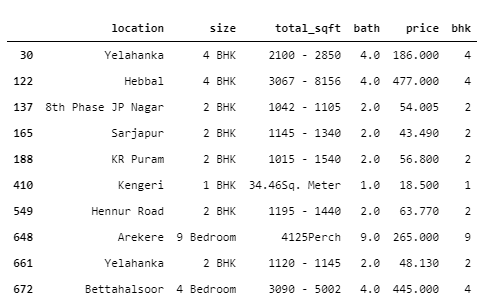
**Feature Engineering**

After cleaning of database, the most important part of our data pipeline is to create some useful features to have better analysis of the data. So, in this chapter we have discussed the processes of generation of useful features.

**Adding new feature ‘bhk’:**

As far of analysis we found that the ‘size’ column have alpha-numeric format and the numerical values determine the size of the apartment/house. So from this we extract the numeric values and assigned a new column for that. i.e. ‘bhk’.



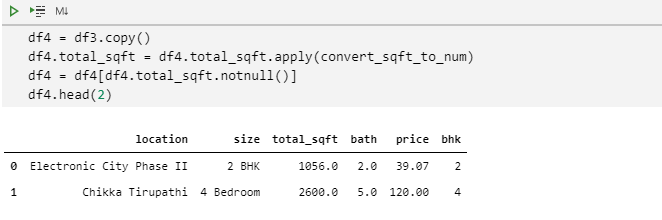


Here we separated the bhk (number) from size column using simple lambda function.

**Updating the ‘total\_sqft’ column values to ‘float’ data type:**

As the values present in each cell of the ‘total\_sqft’ column contain numeric value in ‘string’ format. Thus, we are required to convert it to float so as to make some mathematical analysis over that column when required.





For this purpose we use ‘convert\_sqft\_to\_num()’ to convert it to float data type and updated the column and formed a new data frame by the name of ‘df4’ where do not exist any ‘null’ value in ‘total\_sqft’ column.

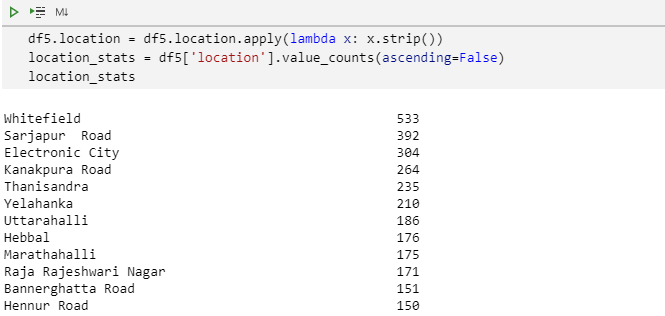
**Adding a new column ‘price\_per\_sqft’:**

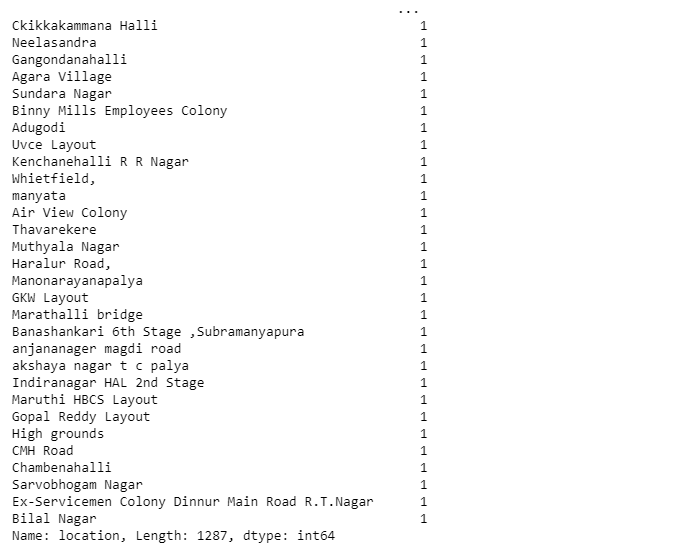
We are required this column to remove some outliers whose ‘price\_per\_sqft’ is less than the threshold value i.e., ‘300’ (in rupees) (our assumption).

For that we used this formula: price\_per\_sqft= (price/total\_sqft)



**Checking for the all-possible unique locations available in database:**

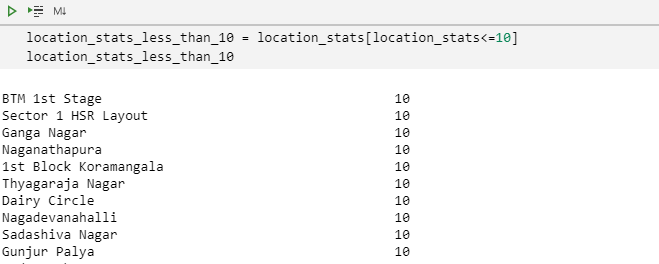




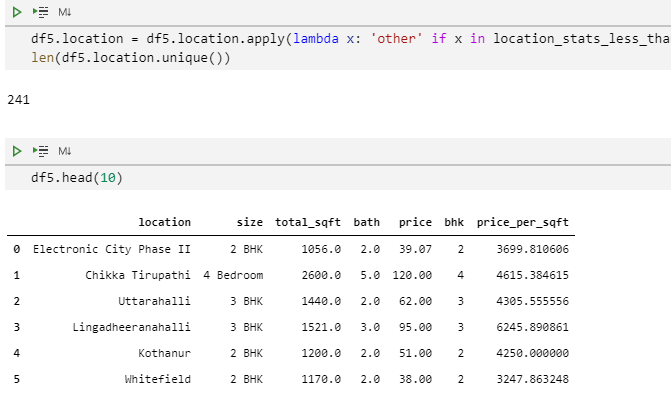
So you can see here that there are exactly ‘1287’ unique locations with their respective counts of number of available houses to each location. If we have to do analysis and train our ML model we are required to create ‘1287’ new columns while doing ‘one-hot-encoding’. Thus, it will create high dimensional complexity and consume huge resources to process such a large dataset.

Thus, it’s the prime requirement to reduce look for some dimensionality reduction option. And for that we are putting all such location for which count is less than ‘10’ into a separate category i.e. ‘others’.

**Performing dimensionality reduction over ‘location’ column:**



These are some of the columns showing the locations which has house counts less than or equal to 10. As these locations contains lesser number of houses, we put all these location into ‘other’ category. Doing this we will reduce number of columns significantly.

So, using above functions we reduce the locations to just ‘241’ from ‘1287’.

Chapter - V

**Removing Outliers**

**1. Removing outliers based on ‘price per square feet’:**

As we previously made assumption that ‘price\_per\_sqft’ can’t be less than 300 Rs. Thus, we need to remove any such values present in ‘price\_per\_sqft’ column.

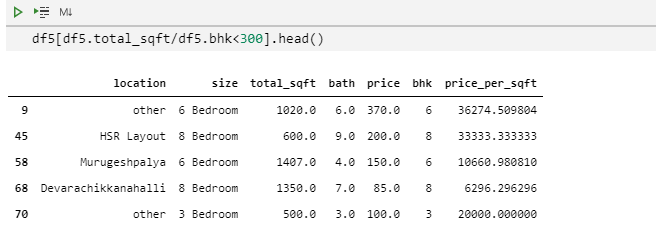
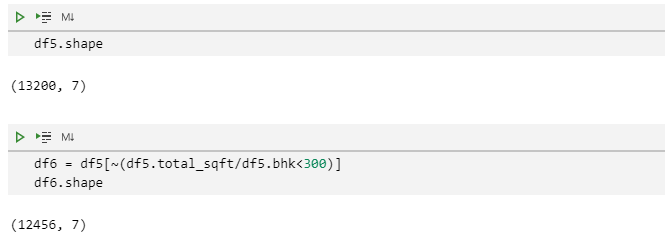
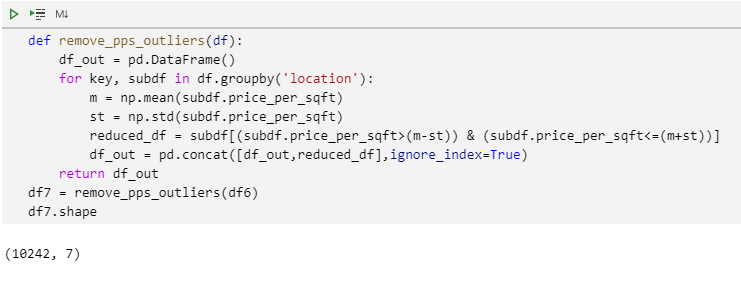


Illustration of such houses whose price\_per\_sqft are less than 300.



So, you can see that some of the outliers corresponding (price\_per\_sqft) are removed. And we saved to new data frame ‘df6’.

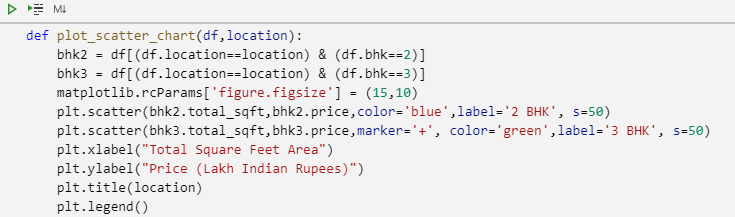
**2. Removing outliers using ‘statistical technique’:**



It is not possible to have huge variation in ‘price\_per\_sqft’ values for houses at same location. So we find mean and standard deviation grouping the data frame location-wise and removing any such data which having variation of more than (mean + standard deviation) and lesser than (mean – standard deviation) from mean value of the particular location.

So, after following the above defined function you can see we arrived at just 10k rose from 12.5k and the result is finally stored to new data frame ‘df7’.

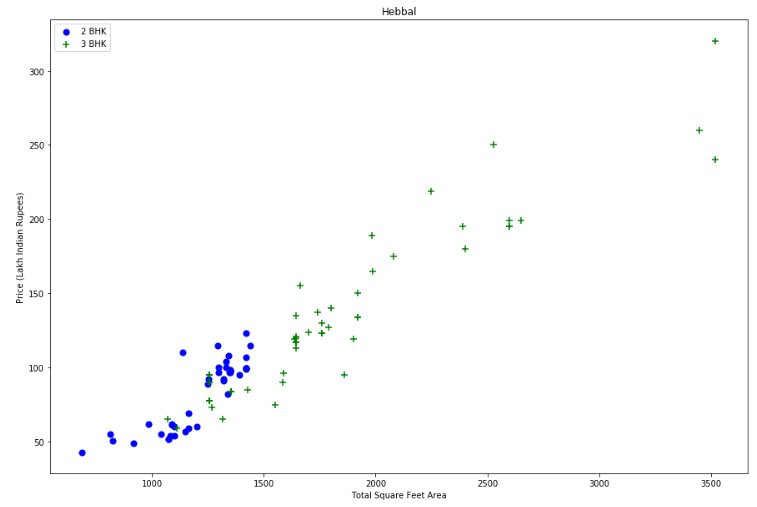
**3. Detecting the outlier and their removal by plotting as scatter plot:**





We can follow from the scatter plot shown on next page, and can see the comparison of 2-bhk and 3-bhk houses at same location.

We observe that some of the 3-bhk house at same location with same square feet have price lower than 2-bhk houses , thus we treat them as outlier as such situation is not possible. And it can decrease our machine learning model accuracy.

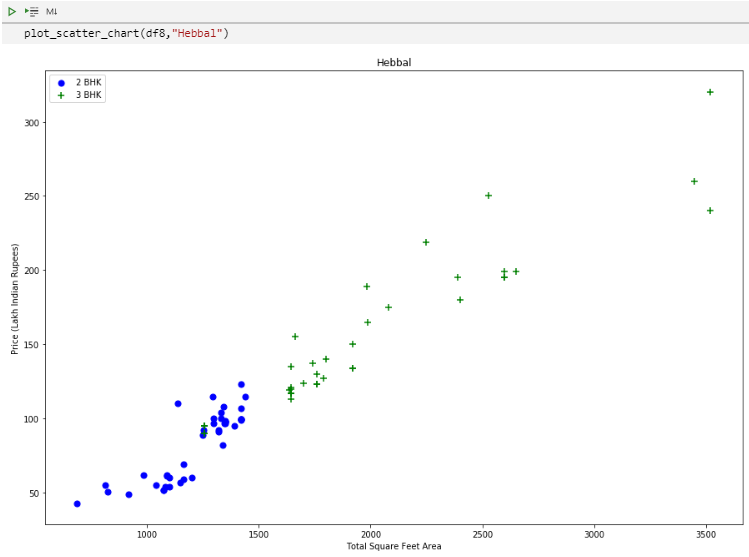
*Scatter plot comparing price of 2-bhk and 3-bhk at different Total Square Feet Area*



Here as you can find that we grouped the data location-wise and for each sub-data frame we are finding the mean, standard-deviation and count. And later comparing the house with one less bhk value to the current bhk value.

If the price per square feet for current bhk is found to be less than ‘1 bhk’ house in the same location. Then we consider such rows as outliers and removing such rows later.

So at the end we can see that only ‘7317’ rows are now available with us as some of the outliers are removed to make a good ML model.

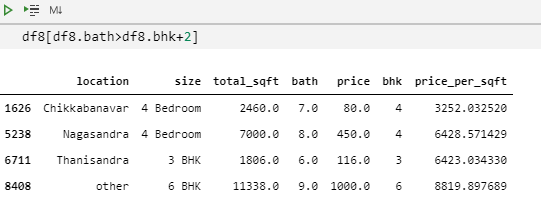
*Scatter-Plot after the removal of the outlier of same location (Hebbal)*

**4. Removing outliers using ‘bath’ feature:**

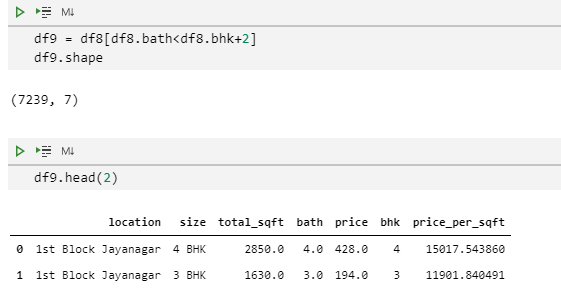
It is very unlikely to have a greater number of bathroom than bedroom.

So, for such condition we have to remove such data giving condition where number of bathrooms are more than two than number of bedrooms.

Mathematically saying, we will remove such data points where, (bath) > (bhk+2).



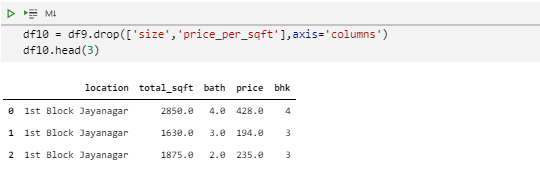
The rows shown above need to be removed from our data frame.



Now after removal of such rows, we only have ‘7239’ rows.

**Dropping ‘size’ and ‘price\_per\_sqft’ columns:**

Dropping ‘size’ and ‘price\_per\_sqft’ as they are not required anymore.



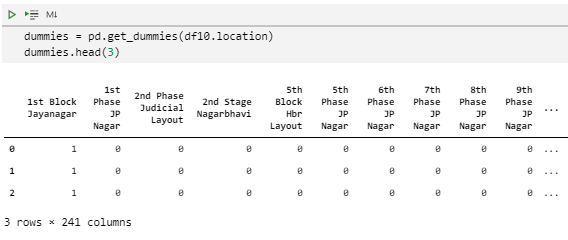
Chapter - VI

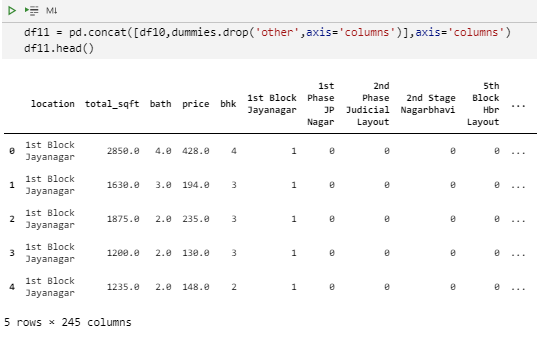
**Performing One-Hot-Encoding**

We need to perform one-hot-encoding on location columns. It will create separate new columns with unique name of locations and assign ‘1’ to rows containing that particular location and ‘0’ to other rows for each new location columns.

This is required to train our machine learning model using supervised learning technique.

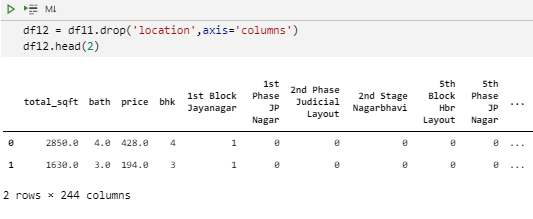
We do it by ‘get\_dummies()’ of pandas applied on ‘location’ column of our data frame. You can see in the image below.





Now concatenating the newly created ‘dummies’ data frame to ‘df10’ using pandas.concat() as you can see here.

**Dropping ‘location’ column:**



From here onwards we are ready to train and test our model using this data frame containing ‘7239 rows’ and ‘244 columns’.

Chapter - VII

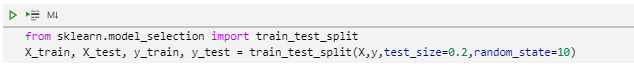
**Building Machine Learning Model**

To use machine learning we first to train our model with database. After training our model we again have to test it on some database. So, in order to achieve this, we have split our database in train and testing sets.

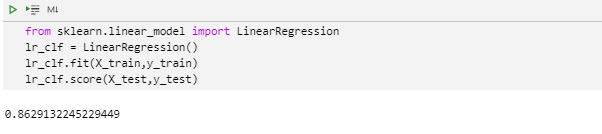
**Splitting database into training and testing database:**

We need to divide our database into two different portion of database.

One on which we can train the machine learning model and other is used to test the model. We choses 20% of data points to test our models.



**Applying Linear Regression technique to out training examples:**

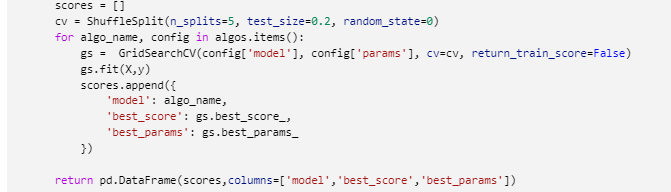


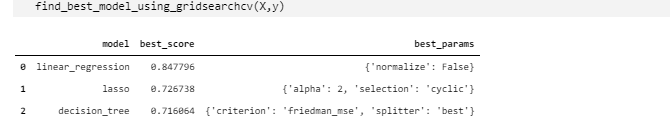
We used here ‘scikit learn’ library to implement Linear Regression model on our training dataset and then cross-validating our result, which give around 86% accuracy, which can be one of the potential models for predicting the house price.

**Finding best fitting model for our created database:**

Here we used Linear Regression, Lasso Regression, Decision Tree Regression as three different machine learning models and comparing their result using GridSearchCV available in sklearn module, to find the best fitting model.

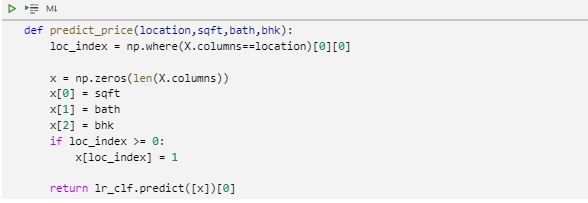




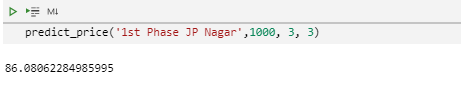
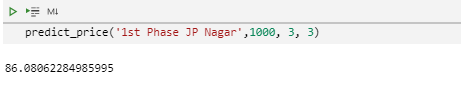


Based on the above result, we can say that Liner Regression model is giving best result among all three. And thus, we will use Linear Regression model as our machine learning model to predict the house price.

**Testing our model by giving few properties:**



For this purpose, we created a function taking ‘Location’, ’Area’, ’Bathroom’, ’BHK’ as parameter to estimate the house price.



The example here shows that with the given parameter the expected house price is around 86 Lakh.

Also, some of the predictions are as follows,



Chapter - VIII

**Making GUI**

As our database contains around 243 locations (after preprocessing), so, it becomes very hard to type in location name while predicting the price of house in that particular location. So, in order to overcome this difficult we made a simple graphical user interface using ‘gradio’ library which enables us to give 4 inputs – area/size (in sqft), BHK, bathrooms, location and displays cost corresponding to input parameters in output section. But first we have to do following things.

1. **Storing trained model in ML file:**

Also, as every time we open our program we need to run/train our model which becomes a tedious task. So, to avoid this we stored our model in a ‘pickle file’ using ‘\_pickle’ library of python.



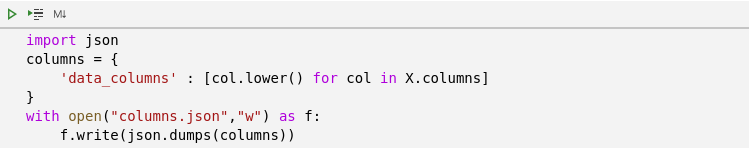
Using this code, we stored our trained ML-model in pickle file.

We can read our model from above stored file using following code.



1. **Storing column name information in text file:**

We have created the ML model for predicting house prices in which we’re supposed to give location of house as one of the inputs. So, in order to retain location names from our database we stored those in a json file. Doing this we can then simply load this file to get back all location names. Also, as we did one hot encoding on locations, we formed column corresponding to each location name. So, for retaining location we can simply store column names.



Here we can see following is data stored in our ‘data\_column.json’ file



We can read above json file using following code.



1. **Making Predict price function for price prediction:**

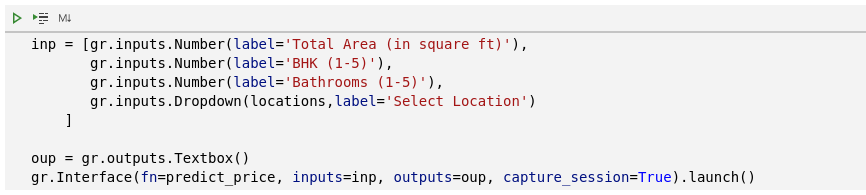
This function takes area size (in sqft), number of bathrooms, number of bhk and location as input and returns predicted price as output.



1. **Making GUI:**

As discussed earlier, Making GUI is crucial for making price prediction easy and user friendly. So, we made a GUI using gradio library.

The code for GUI is as below,



The ‘inp’ variable contains a list of gradio inputs. Those are as follows,

1. Total Area in square ft
2. Number of BHK
3. Number of Bathrooms
4. Location (can be selected from a drop-down menu)

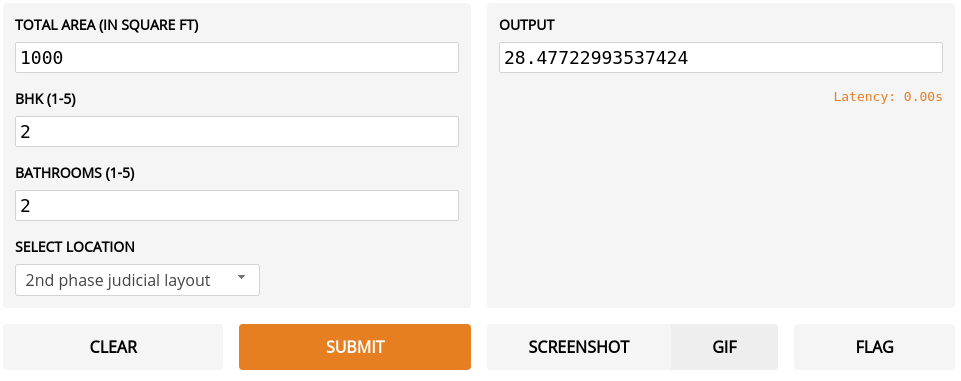
This list is passed as input to interface function of gradio library.

The ‘oup’ is output variable which will collect predicted price (in lakhs).

The interface function takes 3 main parameters. Those are as follows,

1. fn – function that is supposed to be used to perform operation of given inputs
2. inputs – variable to store inputs to be taken from user
3. outputs – variable to store output

Our GUI looks like this,



\*The output is in lakhs.

**References**

1. [Kaggle](https://www.kaggle.com/)
2. [Pandas Documentation](https://pandas.pydata.org/docs/)
3. [Scikit learn Documentation](https://scikit-learn.org/0.21/documentation.html)
4. [Matplotlib Documentation](https://matplotlib.org/stable/contents.html)
5. [Gradio Documentation](https://www.gradio.app/docs)