**House Price Prediction using Machine Learning in Python**

**PHASE-1 DOCUMENT SUBMISSION**

# PROJECT: Problem definition at design thinking

**TEAM MEMBERS:**

110521106002-R.ABIRAMI

110521106003-K.ARULARASI

110521106014-C.MONISHA

110521106325-K.PONNARASI

**ABSTRACT:**

**PROJECT DESCRIPTION:**

**A. Problem Statement**

Thousands of houses are sold everyday. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price?

**B. Best Possible Solutions**

* Housing Expert
* Intuition About House
* Using Machine Learning

**C. Introduction About Project**

House Price prediction are very stressful work as we have to consider different things while buying a house like the structure and the rooms kitchen parking space and gardens. People don’t know about the factor which influence the house price. But by using the Machine learning we can easily find the house which is to be prefect for us and helps to predict the price accurately.

# [Housing Price Prediction In Beijing](https://github.com/bibek376/Housing_Price_Prediction#housing_price_prediction_in_beijing)

## [Installation Guide](https://github.com/bibek376/Housing_Price_Prediction#installation-guide)

1. Clone or Fork the Project
2. Create a Virtual Enviroment
3. go to same virtual enviroment and write below cmd
4. pip install -r requirements.txt

## [Project Highlights](https://github.com/bibek376/Housing_Price_Prediction#project-highlights)

1. Research Paper
2. User Friendly
3. Accuracy
4. Open Source

## [Table Of Content](https://github.com/bibek376/Housing_Price_Prediction#table-of-content)

1. [Project Description](https://www.kaggle.com/ruiqurm/lianjia)  
   A. Problem Statement  
   B. Best Possible Solutions  
   C. Introduction About Project  
   D. Tools and Libraries
2. [Data Collection](https://www.kaggle.com/ruiqurm/lianjia)
3. Generic Flow Of Project
4. [EDA](https://github.com/bibek376/Housing_Price_Prediction/blob/master/Housing_Price_Predication_Project_EDA.ipynb)  
   A. Data Cleaning  
   B. Feature Engineering  
   C. Data Normalization
5. [Choosing Best ML Model](https://github.com/bibek376/Housing_Price_Prediction/blob/master/Machine_Learning_Model_Creation.ipynb)
6. [Model Creation](https://github.com/bibek376/Housing_Price_Prediction/blob/master/Machine_Learning_Model_Creation.ipynb)
7. Model Deployment
8. Model Conclusion
9. Project Innovation
10. Limitation And Next Step
11. Working Project Video

#### [D. Tools and Libraries](https://github.com/bibek376/Housing_Price_Prediction#d-tools-and-libraries)

Tools:  
a.Python  
b.Jupyter Notebook  
c. Flask  
d. HTML  
e. CSS  
f. JS  
g. Heroku  
h. GitHub

Libraries:  
a.Pandas  
b.Scikit Learn  
c.Numpy  
d.Seaborn  
e.Matpoltlib

### [2. Data Collection](https://github.com/bibek376/Housing_Price_Prediction#2-data-collection)

For this project we used the data that is available on Kaggle. There are 26 columns and 318851 Rows. These are the major point about the data set.  
url: the url which fetches the data  
id: the id of transaction  
Lng: and Lat coordinates, using the BD09 protocol.  
Cid: community id  
tradeTime: the time of transaction  
DOM: active days on market  
followers: the number of people follow the transaction.  
total Price: the total price  
price: the average price by square  
square: the square of house  
living Room: the number of living room  
drawing Room: the number of drawing room  
kitchen: the number of kitchen  
bathroom the number of bathroom  
floor: the height of the house. I will turn the Chinese characters to English in the next version.  
building Type: including tower( 1 ) , bungalow( 2 )，combination of plate and tower( 3 ), plate( 4 ).  
construction Time: the time of construction  
renovation Condition: including other( 1 ), rough( 2 ),Simplicity( 3 ), hardcover( 4 )  
building Structure: including unknow( 1 ), mixed( 2 ), brick and wood( 3 ), brick and concrete( 4 ),steel( 5 ) and steel-concrete composite ( 6 ).  
ladder Ratio: the proportion between number of residents on the same floor and number of elevator of ladder. It describes how many ladders a resident have on average.  
elevator: have ( 1 ) or not have elevator( 0 )  
fiveYearsProperty: if the owner have the property for less than 5 years.

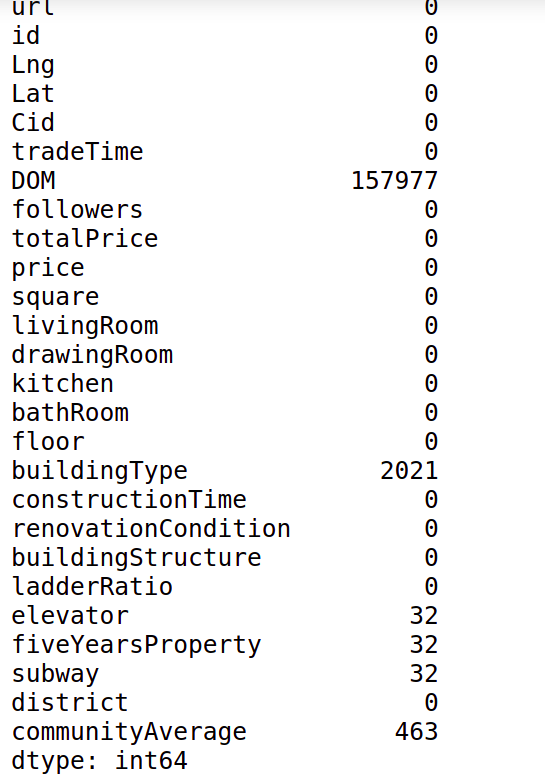
### [3. Generic Flow Of Project](https://github.com/bibek376/Housing_Price_Prediction#3-generic-flow-of-project):

### [4. EDA](https://github.com/bibek376/Housing_Price_Prediction#4-eda):

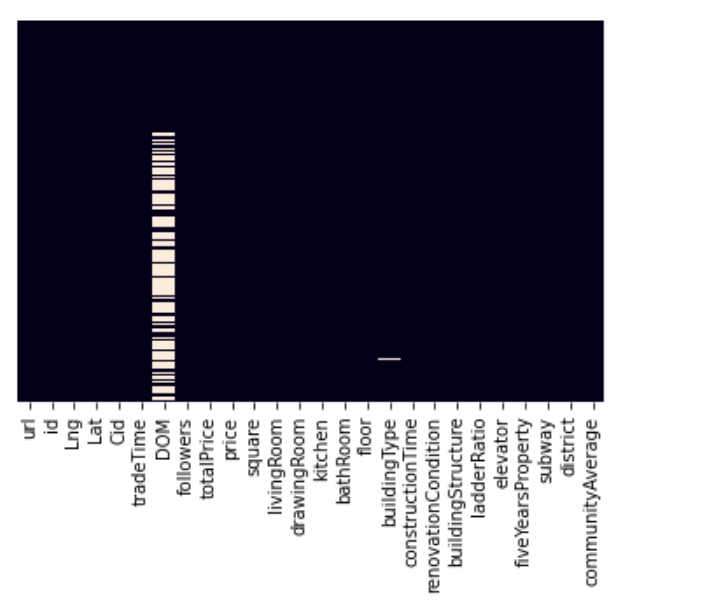
Index(['url', 'id', 'Lng', 'Lat', 'Cid', 'tradeTime', 'DOM', 'followers', 'totalPrice', 'price', 'square', 'livingRoom', 'drawingRoom', 'kitchen', 'bathRoom' 'floor', 'buildingType', 'constructionTime', ' renovationCondition', 'buildingStructure', 'ladderRatio', 'elevator', 'five YearsProperty', 'subway', 'district', 'communityAverage'], dtype='object')

#### [A.Data Cleaning](https://github.com/bibek376/Housing_Price_Prediction#adata-cleaning)

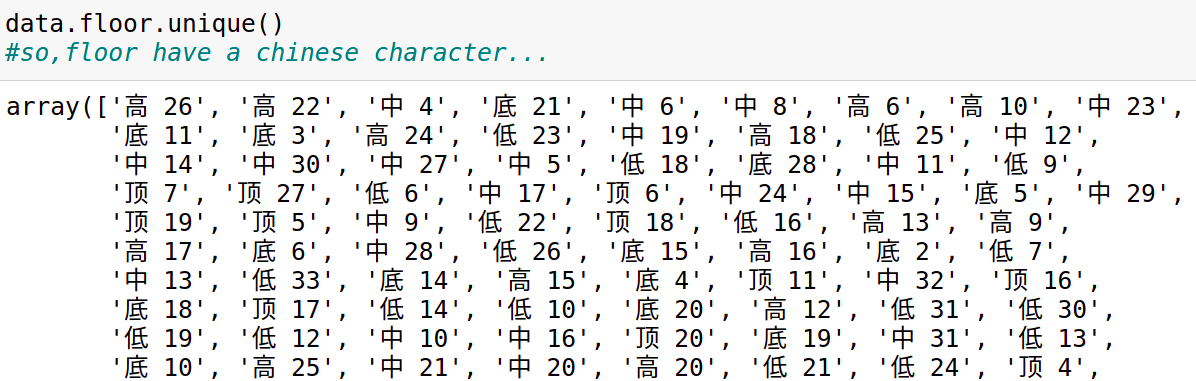
we have 26 columns ,from these we don't want some column(i.e. url,id,cid) then we will perform data cleaning wich involve following steps. our target variable is totalPrice  
a. Impute/Remove missing values or Null values (NaN)  
b. Remove unnecessary and corrupted data.  
c. Date/Text parsing if required.



we handle NAN value using appropriate solutions.

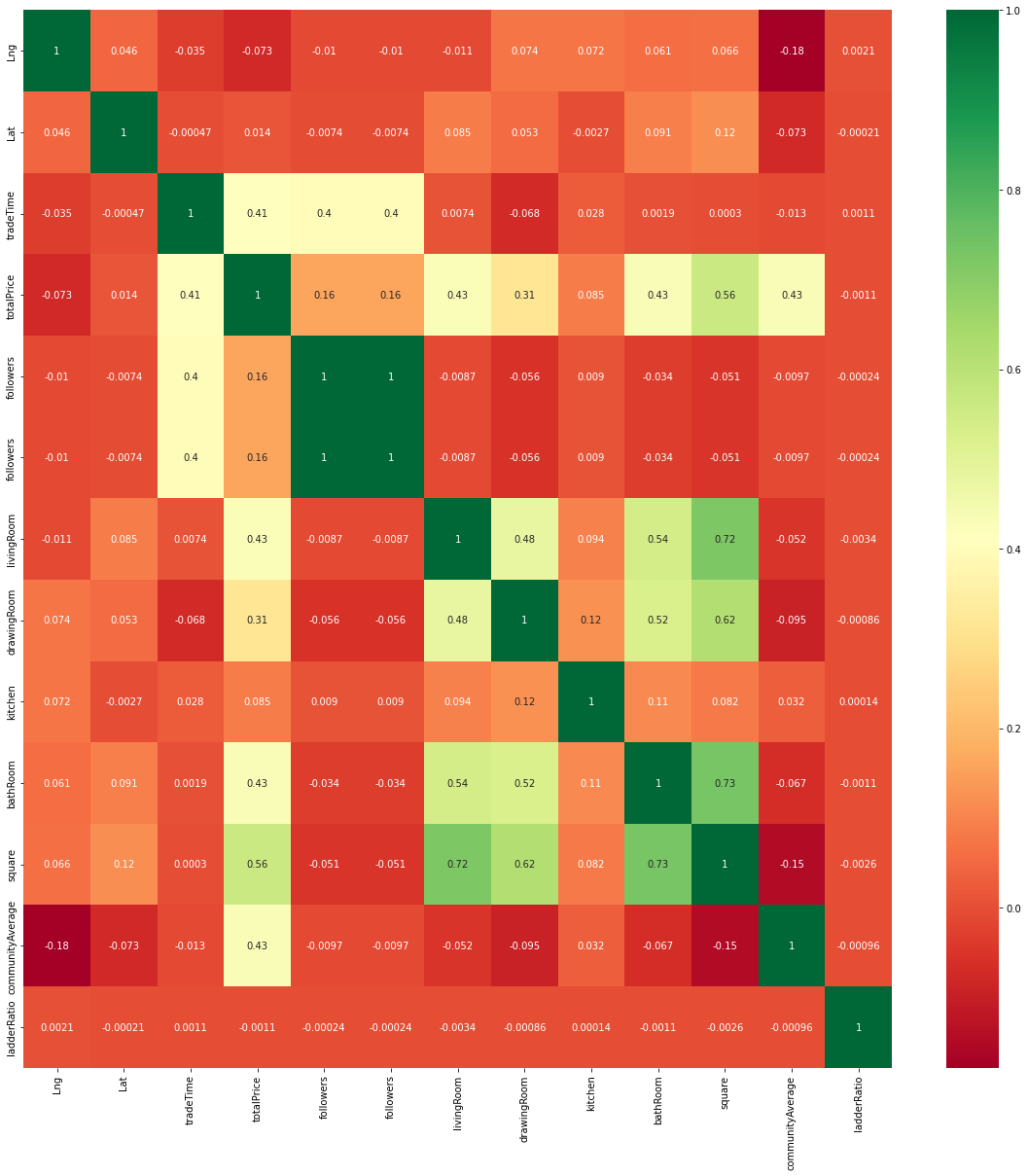


DOM Column have more than 50% value are missing it's better to delete that column

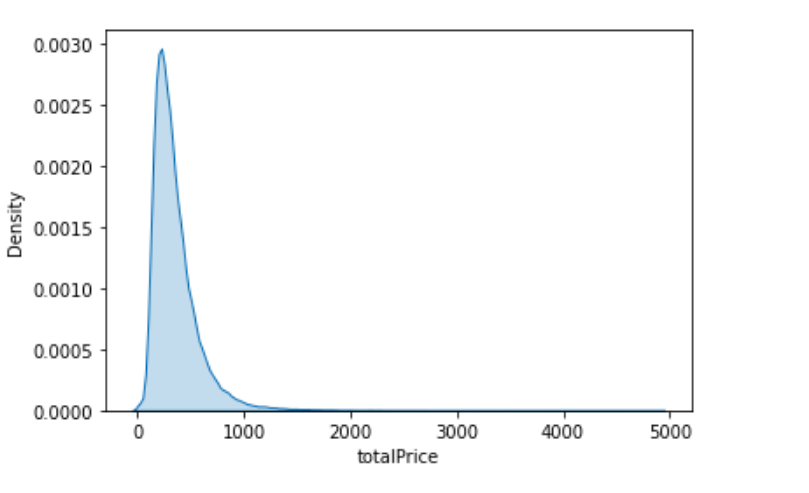


some column have unique character. we solve these problem using split method and create seprate column for unique character.

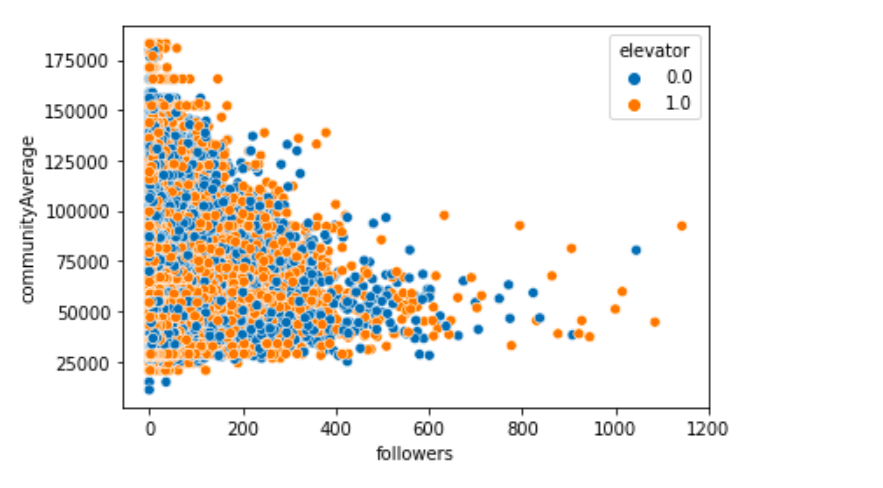
We also have a categorical data we handle such kind of data using dummies variable concept. following are the columns which have categorical data.  
a. renovationCondition  
b. buildingStructure  
c. buildingType  
d. district  
e. elevator  
f. floor\_type



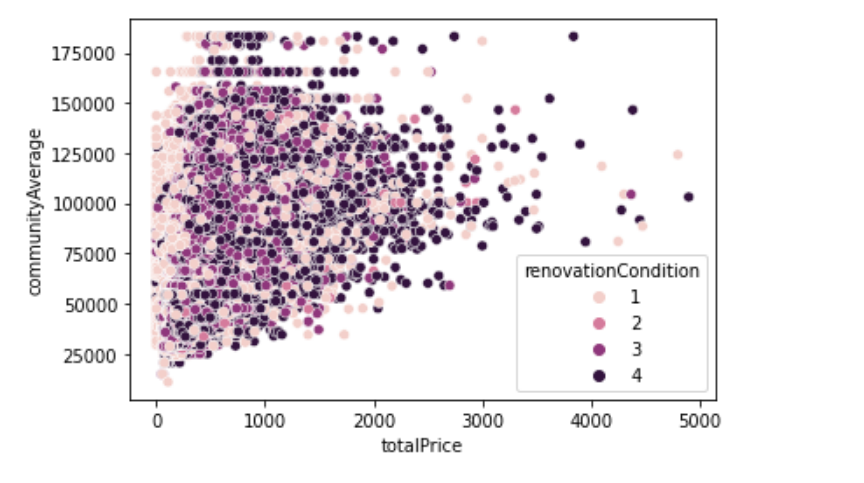
Summary of the Heat-Map  
a. totalPrice is highly corellated with community average, square, bathroom, Livingroom and Trde Time.  
b. total price is highly negative corellated with ladderRatio,lat and lng.



Summary of the Density Plot  
a. most of the output features is lies between 0-2500



Summary of Scatterplot  
a. Most of the House Followers 0-400.

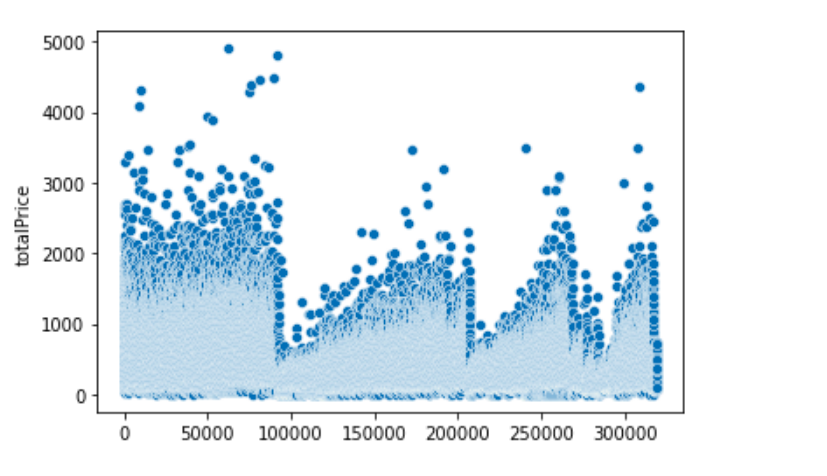


Summary of Scatterplot with respect to renovation Condition  
a. most of the expensive houses have Hardcover as a renovation condition

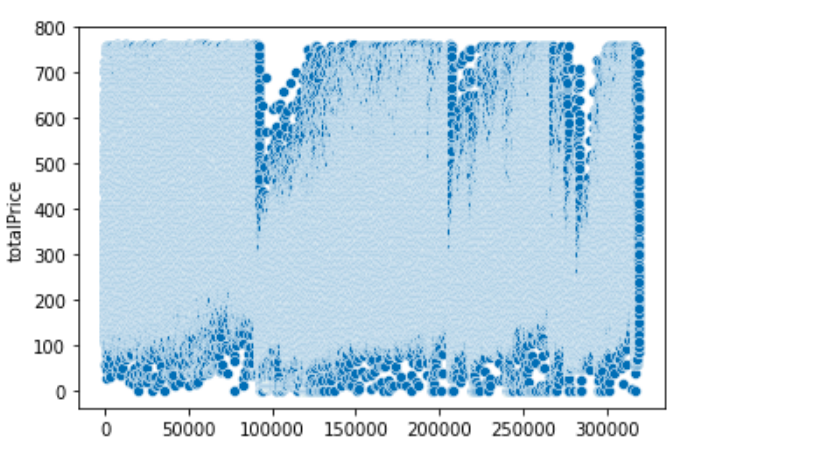


Summary of lineplot  
a. Most of the peoples average are lies in 12500-150000 ...

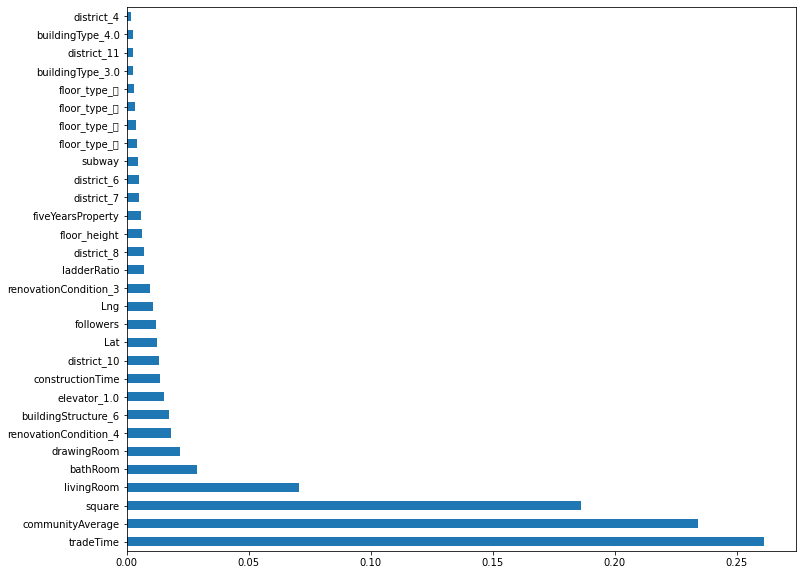
#### [B. Feature Engineering](https://github.com/bibek376/Housing_Price_Prediction#b-feature-engineering)

we found outlier in our data ..

from the above figure we can notice that we have an outlier present in our dataset.  
for outlier we can use IQR method and after using IQR method. Now, our data looks fine.



using the feature engineering we got out top 30 features with respect to totalPrice .



So, these are the top 20 features for our model  
a. tradeTime  
b. CommunityAverage  
c. square  
d. livingRoom  
e. bathRoom  
f. drawingRoom  
g. renovationCondition  
h. buildingStructure  
i. elevator  
j. constructionTime  
k. Followers

#### [C. Data Normalization](https://github.com/bibek376/Housing_Price_Prediction#c-data-normalization)

Normalization (min-max Normalization)  
In this approach we scale down the feature in between 0 to 1

we have numerical column where we can apply min-max Normalization.

**col\_for\_normalization= [ 'Lng', 'Lat', 'followers', 'square', 'livingRoom', 'drawingRoom', 'kitchen', 'bathRoom', 'ladderRatio', 'fiveYears Property', 'subway', 'communityAverage', 'floor\_height']**

### [5. Choosing Best ML Model](https://github.com/bibek376/Housing_Price_Prediction#5-choosing-best-ml-model)

List of the model that we can use for our problem  
a. Linear Regression model  
b. KNN Model  
c. Decision Tree  
d. Random Forest

**By using Linear Regression we got: Training data accuracy 0.7574698746154127 Testing data accuracy 0.7576318049777959**

Using the linearRegression we got only 75 % accuracy.

**print (rfm.score (X\_train,y\_train)) print (rfm.score (X\_test,y\_test))**

**0.9848032043888097**

**0.895634280563376**

Using the Random Forest we got 98 % accuracy on train data and 89 % on test data .so,we can consider Random Forest as a Best Algorithm for this problem.

### [6. Model Creation](https://github.com/bibek376/Housing_Price_Prediction#6-model-creation)

So, using a Random Forest we got good accuracy , we can Hyperparameter tuning for best accuracy.

Algorithm that can be used for Hyperparameter tuning are :-

a. GridSearchCV  
b. RandomizedSearchCV  
c. Bayesian Optimization-Automate Hyperparameter Tuning (Hyperopt)  
d. Sequential model-based optimization  
e. Optuna-Automate Hyperparameter Tuning  
f. Genetic Algorithm

Main parameters used by Random Forest Algorithm are :-

a. n\_estimators ---> The number of trees in the forest.  
b. criterion--->{"mse", "mae"}-->The function to measure the quality of a split  
c. max\_features--->{"auto", "sqrt", "log2"}--> The number of features to consider when looking for the best split:

So, After Hyperparameter Tuning we got 90 % accuracy on test data and 94 % accuracy on train data.

**print (rfm.score (X\_train, y\_train))**

**print (rfm.score (X\_test, y\_test))**

**[Parallel (n\_jobs=12)]: Using backend ThreadingBackend with 12 concurrent workers.**

**[Parallel (n\_jobs=12)]: Done 17 tasks | elapsed 0.2s**

**[Parallel (n\_jobs=12)]: Done 120 out of 120 | elapsed: 0.9s finished**

**[Parallel (n\_jobs=12)]: Using backend ThreadingBackend with 12 concurrent workers.**

**[Parallel (n\_jobs=12)]: Done 17 tasks elapsed: 0.1s**

**0.944283757229526**

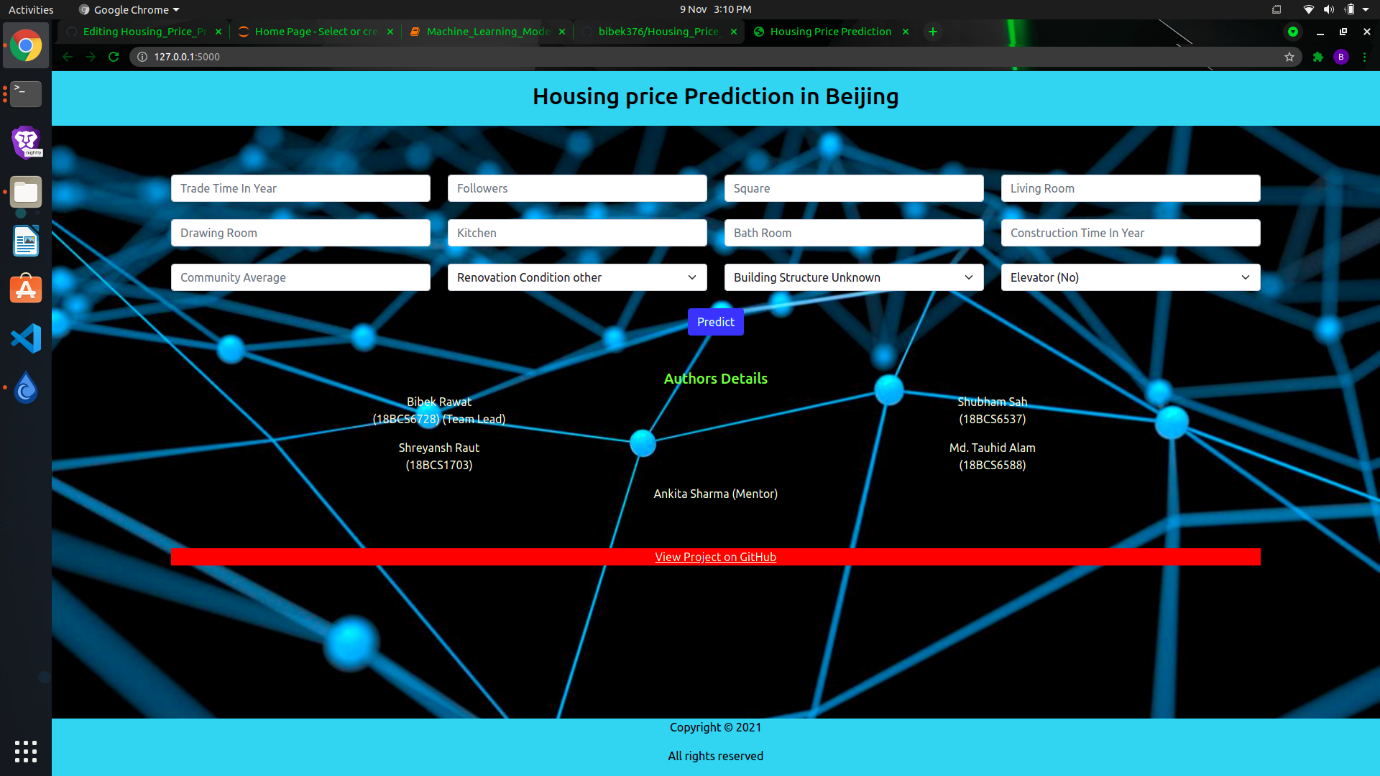
**0.8966624691850038**

**[Parallel (n\_jobs=12)]: Done 120 out of 120 | elapsed: 0.4s finished**

Now, Accuracy of model seems to be very good .so we can save the model using pickle.

### [7. Model Deployment](https://github.com/bibek376/Housing_Price_Prediction#7-model-deployment)

After creating model, we integrate that model with beautiful UI. for the UI part we used HTML, CSS, JS and Flask



### [8. Model Conclusion](https://github.com/bibek376/Housing_Price_Prediction#8-model-conclusion)

Model predict 90% accurately on test data and 94% accurately on train data .

### [9. Project Innovation](https://github.com/bibek376/Housing_Price_Prediction#9-project-innovation)

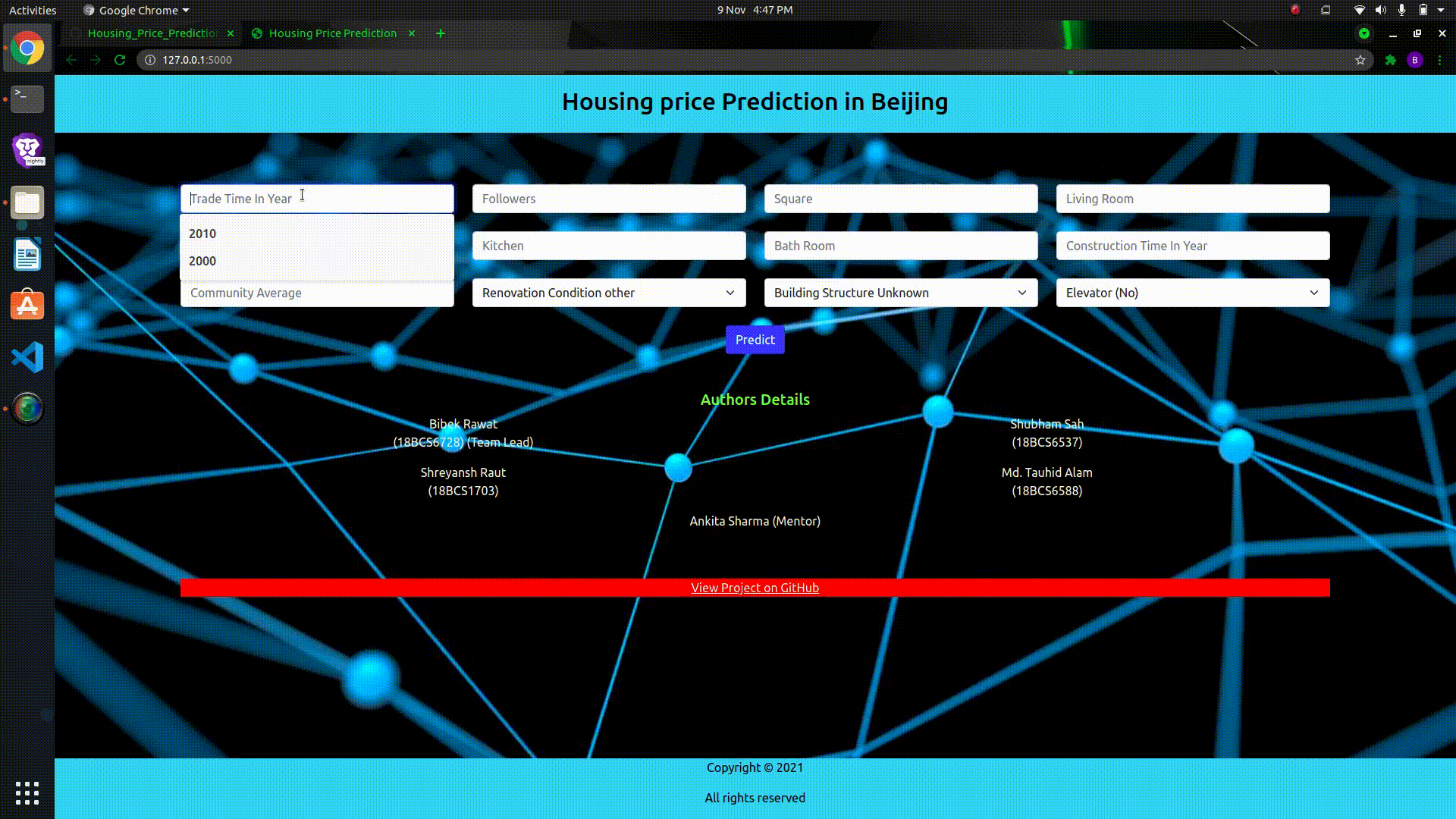
a. Easy to use  
b. open source  
c. Best accuracy  
d. GUI Based Application

### [10. Limitation And Next Step](https://github.com/bibek376/Housing_Price_Prediction#10-limitation-and-next-step)

Limitation are :-  
a. Mobile Application  
b. Accuracy can be improve  
c. Model Size is heavy(~310 mb )  
d. Feature is limited

Next Step are :-  
a. we will work on mobile application  
b. we will reduce the size of model using PCA .

### [11. Working Project Video](https://github.com/bibek376/Housing_Price_Prediction#11-working-project-video)



**PYTHON PROGRAMMING:**

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

import pandas as pdd

# Loading the dataset

data\_h = pdd.read\_csv('kc\_house\_data.csv')

# Selecting the features and target variable

Features1 = ['bedrooms', 'bathrooms', 'sqft\_living', 'sqft\_lot', 'floors', 'zipcode']

target = 'price'

X1 = data\_h[features1]

y1 = data\_h[target]

# We will perform the data splitting into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X1, y1, test\_size=0.2, random\_state=42)

# instance of the Linear Regression model creation

model = LinearRegression()

# Training the model

model.fit(X\_train, y\_train)

# Making predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluating the model

score = model.score(X\_test, y\_test)

print("Model R^2 Score:", score)

# Predicting the price of a new house

new\_house = pdd.DataFrame({'bedrooms': [2], 'bathrooms': [2.5], 'sqft\_living': [600], 'sqft\_lot': [600], 'floors': [2], 'zipcode': [98008]})

predicted\_price = model.predict(new\_house)

print("Predicted Price:", predicted\_price[0])

### **OUTPUT:**

Model R^2 Score: 0.5152176902631012

Predicted Price: 121215.61449578404

## **CONCLUSION:**

In conclusion, using machine learning in Python is a powerful tool for predicting house prices. By gathering and cleaning data, visualizing patterns, and training and evaluating our models, we can make informed decisions in the dynamic world of real estate.

By leveraging advanced algorithms and data analysis, we can make accurate predictions and inform decision-making processes. This approach empowers buyers, sellers, and investors to make informed choices in a dynamic and competitive market, ultimately maximizing their opportunities and outcomes.