Housing Price Prediction using Python

Raghav Behal, VIT University
Ujjwal Prakash, VIT University
Udit Kanotra, VIT University
Prof. Naveen Kumar N, VIT University

Abstract

Housing is an essential part of our lives since the beginning of civilisation. It is commonly accepted that the well-being of both individuals and families is substantially affected when the need for satisfactory housing is not met. Access to adequate housing has long been viewed as a basic human right and is considered to be an integral factor for the enjoyment of other economic, social and cultural rights. Now, housing prices change every year, quite often in the upwards direction. As a result, there definitely rises a need for a system with the ability to predict these prices in the future. While the housing price prediction model can assist a particular developer in determining the selling price of a house, it can also certainly help a customer in deciding the right time to purchase a house. Generally, there are three factors that influence the price of a house, namely physical conditions, concept and location. When observed in depth, the factors that directly influence house prices are as follows - area of the house; location of the house; layout and design of the house; quality of construction material used to build the house; time since construction; time since being sold last; proximity to marketplaces, schools, hospitals, etc.; connectivity and accessibility of house location with transport; etc. The data set for this project will be taken from the Kaggle Housing Data Set Knowledge Competition. As

mentioned above, the data set is quite simple. The data set consists of 81 explanatory variables comprising of numeric, categorical, and ordinal variables. The target variable is Sale_Price. We create and add new features to the data set. Most of the ideas for these features come during the hypothesis generation stage. Model training is done with help of XGBoost, Neural Network, Lasso. Using these algorithms, we train the model on the given data set. Once the model is trained, we evaluate the model's performance using a suitable error metric. Here, we also look for variable importance, i.e., which variables have proved to be significant in determining the target variable. And, accordingly we can shortlist the best variables and train the model again. Finally, we test the model on the unseen data (test data) set. Once we get the scores, such as RMSE (root mean square error), for each model or algorithm, we can decide which one performs better. If needed, we can even try to ensemble them and see if there comes an improvement.

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

For building a prediction model, many experts use GBR(gradient boosting regression). It is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Keywords: Housing Price Prediction, Python, Machine Learning, Artificial Intelligence, Test, Train, XGBoost, Neural Network, Lasso, RMSE, Linear Regression, Gradient Boosting Regression

Introduction

House prices increase every year, so there is a need for a system to predict house prices in the future. House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location.

Some factors that directly influence house prices are the following: Area of House, how old is the house, Location of the house, how close/far is the market, Connectivity of house location with transport, how many floors does the house have, what material is used in the construction etc. The data set consists of 81 explanatory variables. The target variable is Sale_Price. As you can see the data set comprises numeric, categorical, and ordinal variables. Now, we create and add new features to the data set. Most of the ideas for these features come during the hypothesis generation stage. Model training is done with help of XGBoost, Neural Network, Lasso. Using these algorithms, we train the model on the given data set. Once the model is trained, we evaluate the model's performance using a suitable error metric. Here, we also look for variable importance, i.e., which variables have proved to be significant in determining the target variable. And, accordingly we can shortlist the best variables and train the model again. Finally, we test the model on the unseen data (test data) set.

Literature Survey

Artificial Intelligence plays a major role from past years in image detection, spam reorganization, normal speech command, product recommendation and medical diagnosis. Real properties possess value which is dependent on numerous factors. Investors and owners of the property are interested in the maximum returns, it would fetch. Considering the amount of money involved in real estate, there is a need of accurate prediction of returns and associated risks. This necessitates use of Artificial Intelligence prediction models. Comparison of various techniques shows that Artificial Neural Network and fuzzy logic are better suited if attributes and model parameters are appropriately selected. selection of variables is of paramount importance in the development and use of AI models in property rate forecasting. The presence of non-linear relationship between independent variables and value of property due to inadequate number of sample size could be the root of the poor performance of any AI model [1].

In [3] on the basis of the actual data, the values of Boston suburb houses are forecast by several machine learning methods. The paper uses the tools like support vector machine, least squares support vector machine, and partial least squares methods are used to forecast the home values. The paper finally concludes that Support vector machines and least squares support vector machine have better prediction effect, learning ability, and generalization ability than partial least squares methods. The paper [2] focuses on to predict house prices based on NJOP houses in Malang city with regression analysis and particle swarm optimization (PSO). PSO is used for selection of affect variables and regression analysis is used to determine the optimal coefficient in prediction. The paper [4] proposes a hybrid Lasso and Gradient boosting regression model to predict individual house price. Lasso is very useful for feature selection, and one more benefit of removing useless features is it could decrease Ridge.

In [5], the paper presents the reviews and findings of using machine learning algorithms for real data of housing prices in the area of Petaling Jaya, Selangor. The paper further demonstrate that feature selection is an important component of machine learning prediction. The pricing factor depends upon two important performances of machine learning prediction are accuracy of prediction, and averaging of errors or fitness. [8] presents a machine learning approach for solving the house price prediction problem in the classified advertisements. The study focuses on the Indian real estate market. The paper uses a technique known as PropTech and its applications. The business application of this algorithm is that classified websites can directly use this algorithm to predict prices of new properties that are going to be listed by taking some input variables and predicting the correct and justified price. The performance is promising as the latest score was ranked top 1% out of all competition teams and individuals.

[9] provides a study of 125 papers that use hedonic pricing model to estimate house prices in the past decade. The paper provides a list of 20 attributes that are frequently used to specify hedonic pricing models. This dataset contains 12 attributes on this list. Further paper [6] shows that the log price of houses is positively and significantly correlated with the number of bathrooms, bedrooms, fireplaces, garage spaces, stories and the total square feet of the house. Additionally, the paper adds three dummy variables, MAY, JUNE, and JULY, to account for seasonable factor with regards to the houses' prices.

[7] focuses on the point to compare the performance between Multiple Linear Regression model and Neural Network model on estimate house prices in New York. A sample of 1047 houses is randomly selected and finally concludes that Neural Network model is preferred to predict house price compared to MLR model and can be used as an alternative way to estimate house price in future.

[10] focuses on a structured Deep Neural Network model has been designed to learn from the most recently captured data points which allows the model to adapt to the latest market trends. Neurons in a structured Deep Neural Network Model are structurally connected, which makes the network time and space efficient; and thus, it requires fewer data points for training.

Proposed Methodology

“Keep tormenting data until it starts revealing its hidden secrets.” Yes, it can be done but there's a way around it. Making predictions using Machine Learning isn't just about grabbing the data and feeding it to algorithms. The algorithm might spit out some prediction but that's not what you are aiming for. The difference between good data science professionals and naive data science aspirants is that the former set follows this process religiously. The process is as follows:

- **Understand the problem:** Before getting the data, we need to understand the problem we are trying to solve. If you know the domain, think of which factors could play an epic role in solving the problem. If you don't know the domain, read about it.
- **Hypothesis Generation:** This is quite important, yet it is often forgotten. In simple words, hypothesis generation refers to creating a set of features which could influence the target variable given a confidence interval (taken as 95% all the time). We can do this before looking at the data to avoid biased thoughts.
- **Get Data:** Now, we download the data and look at it. Determine which features are available and which aren't, how many features we generated in hypothesis generation hit the

mark, and which ones could be created. Answering these questions will set us on the right track.

- **Data Exploration:** We can't determine everything by just looking at the data. We need to dig deeper. This step helps us understand the nature of variables (skewed, missing, zero variance feature) so that they can be treated properly. It involves creating charts, graphs (univariate and bivariate analysis), and cross-tables to understand the behaviour of features.
- **Data Pre-processing***:* Here, we impute missing values and clean string variables (remove space, irregular tabs, data time format) and anything that shouldn't be there.
- **Feature Engineering:** Now, we create and add new features to the data set. Most of the ideas for these features come during the hypothesis generation stage.
- **Model Training:** Using a suitable algorithm, we train the model on the given data set.
- **Model Evaluation:** Once the model is trained, we evaluate the model's performance using a suitable error metric. Here, we also look for variable importance, i.e., which variables have proved to be significant in determining the target variable. And, accordingly we can shortlist the best variables and train the model again.

- **Model Testing:** Finally, we test the model on the unseen data (test data) set.

<u>Proposed Algorithm for Linear Regression:</u>

- We import our dependencies, for linear regression we use sklearn (built in python library) and import linear regression from it.
- We then initialize Linear Regression to a variable reg.
- Now we know that prices are to be predicted, hence we set labels (output) as price columns and we also convert dates to 1's and 0's so that it doesn't influence our data much. We use 0 for houses which are new that is built after 2014.
- We again import another dependency to split our data into train and test.
- I've made my train data as 90% and 10% of the data to be my test data, and randomized the splitting of data by using random_state.
- So now, we have train data, test data and labels for both let us fit our train and test data into linear regression model.
- After fitting our data to the model, we can check the score of our data i.e., prediction, which in this case is 73%.

Proposed Algorithm for Gradient Boosting Regression:

- We first import the library from sklearn (trust me, it is the best library for all statistical related models)
- We create a variable where we define our gradient boosting regressor and set parameters to it, here
    - n_estimator — The number of boosting stages to perform. We should not set it too high which would overfit our model.
    - max_depth — The depth of the tree node.
    - learning_rate — Rate of learning the data.
    - loss — loss function to be optimized.
    - ls - least squares regression
    - minimum sample split — Number of samples to be split for learning the data
- We then fit our training data into the gradient boosting model and check for accuracy.
- We got an accuracy of 91.94%

Working Methodology

Boosting builds models from individual so called "weak learners" in an iterative way. In boosting, the individual models are not built on completely random subsets of data and features but sequentially by putting more weight on instances with wrong predictions and high errors. The general idea behind this is that instances, which are hard to predict correctly ("difficult" cases) will be focused on during learning, so that the model learns from past mistakes. When we train each ensemble on a subset of the training set, we also call this Stochastic Gradient Boosting, which can help improve generalizability of our model.

The gradient is used to minimize a loss function, similar to how Neural Nets utilize gradient descent to optimize ("learn") weights. In each round of training, the weak learner is built and its predictions are compared to the correct outcome that we expect. The distance between prediction and truth represents the error rate of our model. These errors can now be used to calculate the gradient. The gradient is nothing fancy, it is basically the partial derivative of our loss function - so it describes the steepness of our error function. The gradient can be used to find the direction in which to change the model parameters in order to (maximally) reduce the error in the next round of training by "descending the gradient".

In Neural nets, gradient descent is used to look for the minimum of the loss function, i.e. learning the model parameters (e.g. weights) for which the prediction error is lowest in a single model. In

Gradient Boosting we are combining the predictions of multiple models, so we are not optimizing the model parameters directly but the boosted model predictions. Therefore, the gradients will be added to the running training process by fitting the next tree also to these values.

Because we apply gradient descent, we will find learning rate (the "step size" with which we descend the gradient), shrinkage (reduction of the learning rate) and loss function as hyperparameters in Gradient Boosting models - just as with Neural Nets. Other hyperparameters of Gradient Boosting are similar to those of Random Forests:

- the number of iterations (i.e. the number of trees to ensemble),
- the number of observations in each leaf,
- tree complexity and depth,
- the proportion of samples and
- the proportion of features on which to train on.

Gradient Boosting Machines vs. XGBoost

XGBoost stands for Extreme Gradient Boosting; it is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model. It employs a number of nifty tricks that make it exceptionally successful, particularly with structured data.

The most important are:

1. Computing second-order gradients, i.e. second partial derivatives of the loss function, which provides more information about the direction of gradients and how to get to the minimum of our loss function. While regular gradient boosting uses the loss function of our base model (e.g. decision tree) as a proxy for minimizing the error of the overall model, XGBoost uses the 2nd order derivative as an approximation.
2. Advanced regularization (L1 & L2), which improves model generalization.

XGBoost has additional advantages: training is very fast and can be parallelized / distributed across clusters.

Project Description

List of Modules

- Hypothesis Generation
- Get Data
- Data Exploration
- Data Pre-processing
- Feature Engineering
- Model Training – XGBoost, Neural Network, Lasso and Model Evaluation

Software Requirements

- Operating System: Ubuntu 12/ Windows 7 or above
- Languages: Python

Hardware Requirements

- CPU: Pentium IV or above with 1.6 GHz
- Mouse Type: Optical
- RAM: 2GB or above
- Internet Connectivity Required

Module Description

- Hypothesis Generation: Well, this is going to be interesting. What factors can you think of right now which can influence house prices? As you read this, I want you to write down your factors as well, then we can match them with the data set. Defining a hypothesis has two parts: Null Hypothesis (Ho) and Alternate Hypothesis (Ha). They can be understood as: Ho - There exists no impact of a particular feature on the dependent variable. Ha - There exists a direct impact of a particular feature on the dependent variable. Based on a decision criterion (say, 5% significance level), we always 'reject' or 'fail to reject' the null hypothesis in statistical parlance. Practically, while model building, we look for probability (p) values. If p value < 0.05, we reject the null hypothesis. If p > 0.05, we fail to reject the null hypothesis.

Some factors which we can think of that directly influence house prices are the following:

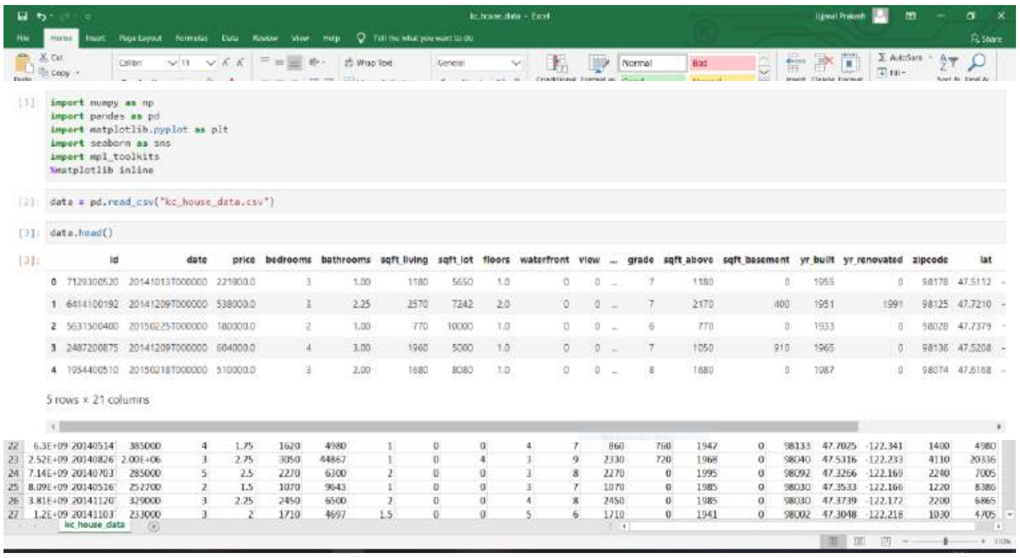
- Area of House
- Age of the House
- Location of House
- Transport Connectivity
- Material used for Construction
- Water/Electricity availability
- Car Parking

- Get Data: Download the data and load it in your python IDE. Also, check the competition page where all the details about the data and variables are given. The data set consists of 81 explanatory variables. The data set comprises numeric, categorical, and ordinal variables. Without further ado, let's start with hands-on coding.
- Data Exploration: Data Exploration is the key to getting insights from data. Practitioners say a good data exploration strategy can solve even complicated problems in a few hours. A good data exploration strategy comprises the following:
    - Univariate Analysis: It is used to visualize one variable in one plot.
    - Bivariate Analysis: It is used to visualize two variables (x and y axis) in one plot.
    - Multivariate Analysis: As the name suggests, it is used to visualize more than two variables at once.

  - o Cross Tables: They are used to compare the behavior of two categorical variables (used in pivot tables as well).

- Data Pre-processing: In this stage, we deal with outlier values, encode variables, impute missing values, and take every possible initiative which can remove inconsistencies from the data set.
- Feature Engineering: There are no libraries or sets of functions one can use to engineer features. It's majorly a manual task. Feature engineering requires domain knowledge and lots of creative ideas. The ideas for new features usually develop during the data exploration and hypothesis generation stages. The motive of feature engineering is to create new features which can help make predictions better. There's a massive scope of feature engineering in this data set. Most categorical variables have near-zero variance distribution. Near-zero variance distribution is when one of the categories in a variable has >90% of the values. Some binary variables depicting the presence or absence of a category are created. The new features will contain 0 or 1 values. In addition, some more variables which are self-explanatory with comments will be created.
- Model Training and Evaluation: The training model starts now. Standard neural networks are used, along with XGBoost. The models are made into ensembles.

Implementation



```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import mpl_toolkits
     %matplotlib inline

[2]: data = pd.read_csv("kc_house_data.csv")

[3]: data.head()

[3]:
           id            date     price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  waterfront  view  ...  grade  sqft_above  sqft_basement  yr_built  yr_renovated  zipcode      lat
0  7129300520  20141013T000000  221900.0        3       1.00         1180      5650     1.0           0     0  ...      7        1180              0      1955             0    98178  47.5112  -
1  6414100192  20141209T000000  538000.0        3       2.25         2570      7242     2.0           0     0  ...      7        2170            400      1951          1991    98125  47.7210  -
2  5631500400  20150225T000000  180000.0        2       1.00          770     10000     1.0           0     0  ...      6         770              0      1933             0    98028  47.7379  -
3  2487200875  20141209T000000  604000.0        4       3.00         1960      5000     1.0           0     0  ...      7        1050            910      1965             0    98136  47.5208  -
4  1954400510  20150218T000000  510000.0        3       2.00         1680      8080     1.0           0     0  ...      8        1680              0      1987             0    98074  47.6168  -

5 rows × 21 columns
```

22	6.3E+09	20140514	385000	4	1.75	1620	4980	1	0	0	4	7	860	760	1947	0	98133	47.7025	-122.341	1400	4980
23	2.52E+09	20140826	2.00E+06	3	2.75	3050	44867	1	0	4	3	9	2330	720	1968	0	98040	47.5316	-122.233	4110	20336
24	7.14E+09	20140703	285000	5	2.5	2270	6300	2	0	0	3	8	2270	0	1995	0	98092	47.3266	-122.169	2240	7005
25	8.09E+09	20140516	252700	2	1.5	1070	9643	1	0	0	3	7	1070	0	1985	0	98030	47.3533	-122.166	1220	8386
26	3.81E+09	20141120	329000	3	2.25	2450	6500	2	0	0	4	8	2450	0	1985	0	98030	47.3739	-122.172	2200	6865
27	1.2E+09	20141103	233000	3	2	1710	4697	1.5	0	0	5	6	1710	0	1941	0	98002	47.3048	-122.218	1030	4705

Fig. A sample of data (kc_house_data.csv)

The Jupyter notebook proceeds as follows:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import mpl_toolkits
%matplotlib inline

data = pd.read_csv("kc_house_data.csv")

data.head()

           id             date     price  bedrooms  bathrooms
sqft_living  \
0  7129300520  20141013T000000  221900.0         3       1.00
1180
1  6414100192  20141209T000000  538000.0         3       2.25
2570
2  5631500400  20150225T000000  180000.0         2       1.00
770
3  2487200875  20141209T000000  604000.0         4       3.00
1960
4  1954400510  20150218T000000  510000.0         3       2.00
1680

   sqft_lot  floors  waterfront  view  ...  grade  sqft_above
sqft_basement  \
0      5650     1.0             0     0  ...      7         1180
0
1      7242     2.0             0     0  ...      7         2170
400
2     10000     1.0             0     0  ...      6          770
0
3      5000     1.0             0     0  ...      7         1050
910
4      8080     1.0             0     0  ...      8         1680
0

   yr_built  yr_renovated  zipcode      lat     long  sqft_liv
ing15  \
0      1955             0    98178  47.5112 -122.257
1340
1      1951          1991    98125  47.7210 -122.319
1690
2      1933             0    98028  47.7379 -122.233
2720
3      1965             0    98136  47.5208 -122.393
1360
4      1987             0    98074  47.6168 -122.045
1800

   sqft_lot15
0        5650
1        7639
2        8062
3        5000
4        7503

[5 rows x 21 columns]
```

```
data.describe()
```

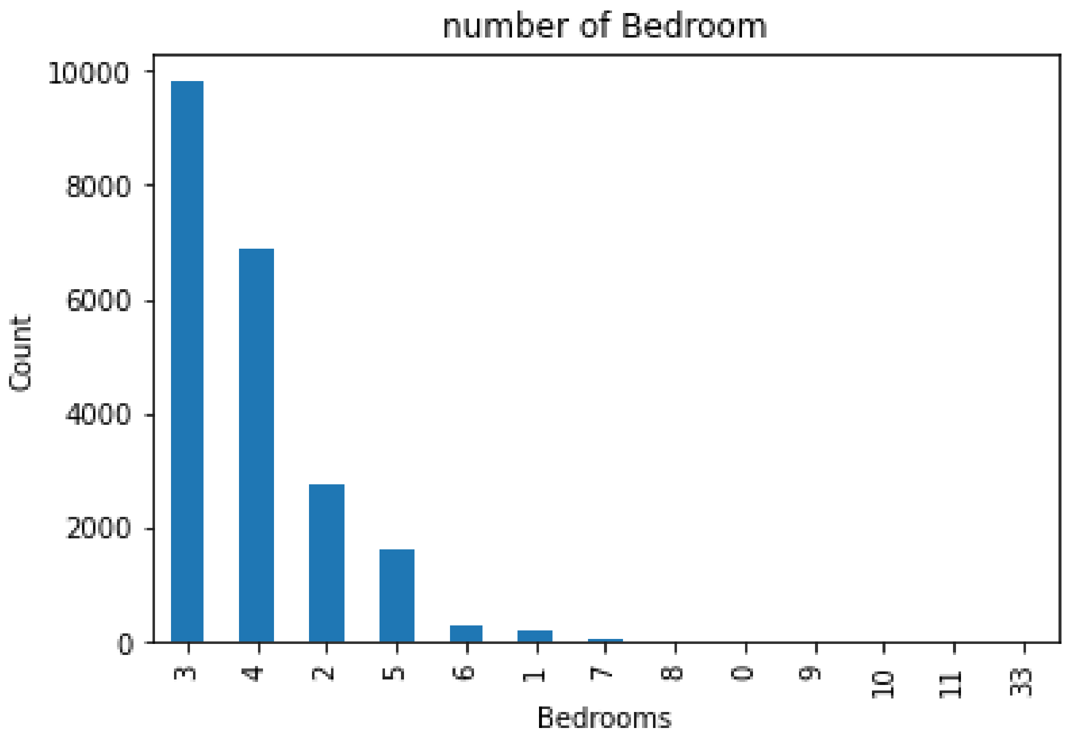
```
                 id          price      bedrooms     bathrooms
sqft_living  \
count  2.161300e+04  2.161300e+04  21613.000000  21613.000000
21613.000000
mean   4.580302e+09  5.401822e+05      3.370842      2.114757
2079.899736
std    2.876566e+09  3.673622e+05      0.930062      0.770163
918.440897
min    1.000102e+06  7.500000e+04      0.000000      0.000000
290.000000
25%    2.123049e+09  3.219500e+05      3.000000      1.750000
1427.000000
50%    3.904930e+09  4.500000e+05      3.000000      2.250000
1910.000000
```

```
75%    7.308900e+09  6.450000e+05      4.000000      2.500000
2550.000000
max    9.900000e+09  7.700000e+06     33.000000      8.000000
13540.000000

           sqft_lot        floors    waterfront          view
condition  \
count  2.161300e+04  21613.000000  21613.000000  21613.000000
21613.000000
mean   1.510697e+04      1.494309      0.007542      0.234303
3.409430
std    4.142051e+04      0.539989      0.086517      0.766318
0.650743
min    5.200000e+02      1.000000      0.000000      0.000000
1.000000
25%    5.040000e+03      1.000000      0.000000      0.000000
3.000000
50%    7.618000e+03      1.500000      0.000000      0.000000
3.000000
75%    1.068800e+04      2.000000      0.000000      0.000000
4.000000
max    1.651359e+06      3.500000      1.000000      4.000000
5.000000

              grade    sqft_above  sqft_basement      yr_built
yr_renovated  \
count  21613.000000  21613.000000   21613.000000  21613.000000
21613.000000
mean       7.656873   1788.390691     291.509045   1971.005136
84.402258
std        1.175459    828.090978     442.575043     29.373411
401.679240
min        1.000000    290.000000       0.000000   1900.000000
0.000000
25%        7.000000   1190.000000       0.000000   1951.000000
0.000000
50%        7.000000   1560.000000       0.000000   1975.000000
0.000000
75%        8.000000   2210.000000     560.000000   1997.000000
0.000000
max       13.000000   9410.000000    4820.000000   2015.000000
2015.000000

            zipcode           lat          long  sqft_living15
sqft_lot15
count  21613.000000  21613.000000  21613.000000   21613.000000
21613.000000
mean   98077.939805     47.560053   -122.213896    1986.552492
```

```
12768.455652
std       53.505026      0.138564      0.140828     685.391304
27304.179631
min    98001.000000     47.155900   -122.519000     399.000000
651.000000
25%    98033.000000     47.471000   -122.328000    1490.000000
5100.000000
50%    98065.000000     47.571800   -122.230000    1840.000000
7620.000000
75%    98118.000000     47.678000   -122.125000    2360.000000
10083.000000
max    98199.000000     47.777600   -121.315000    6210.000000
871200.000000
```

```
data['bedrooms'].value_counts().plot(kind='bar')
plt.title('number of Bedroom')
plt.xlabel('Bedrooms')
plt.ylabel('Count')
sns.despine
```

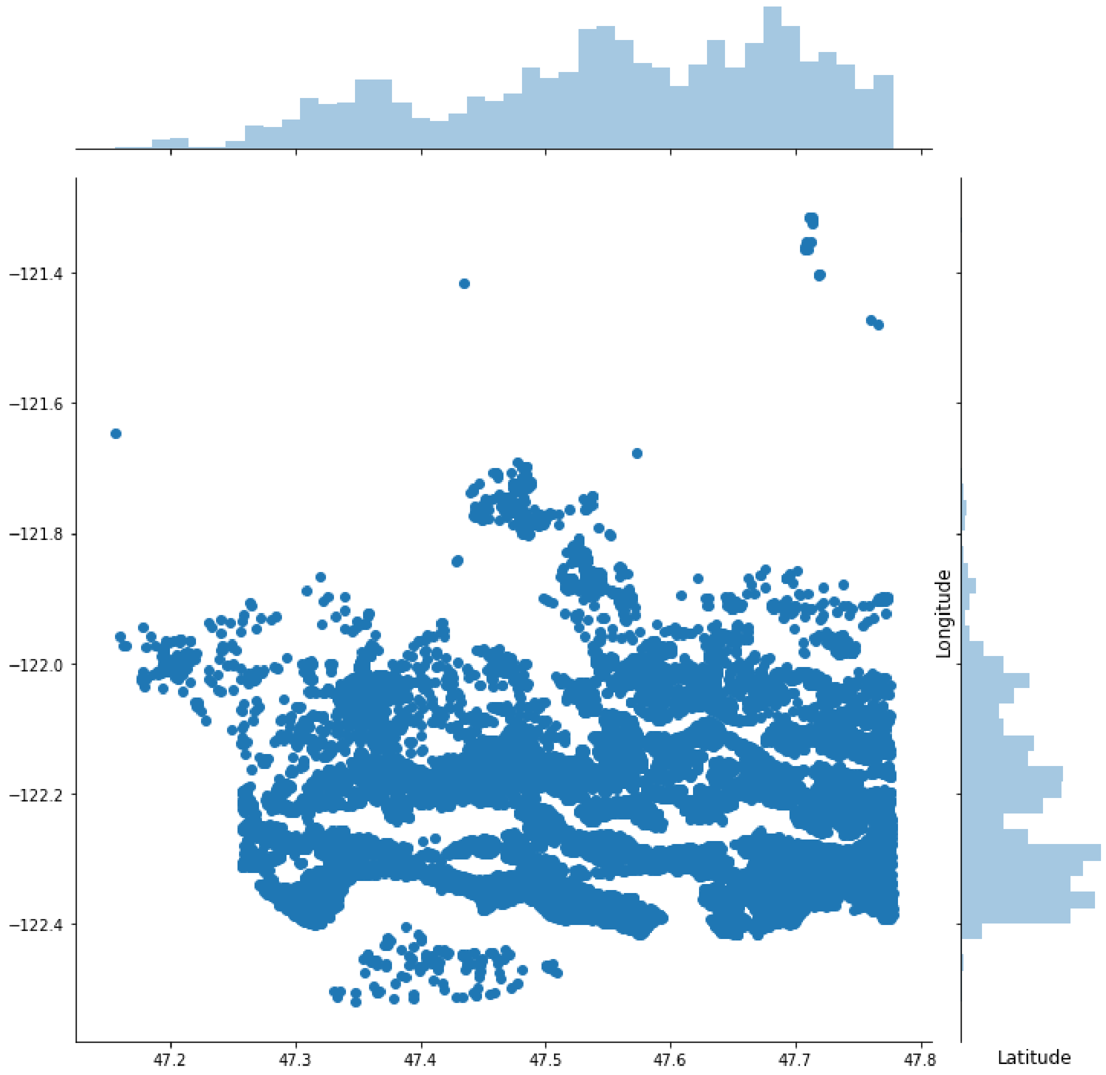
```
<function seaborn.utils.despine(fig=None, ax=None, top=True, r
ight=True, left=False, bottom=False, offset=None, trim=False)>
```



```
plt.figure(figsize=(10,10))
sns.jointplot(x=data.lat.values, y=data.long.values, height=10
)
```

```
plt.ylabel('Longitude', fontsize=12)
plt.xlabel('Latitude', fontsize=12)
plt.show()
#plt1 = plt()
sns.despine
```

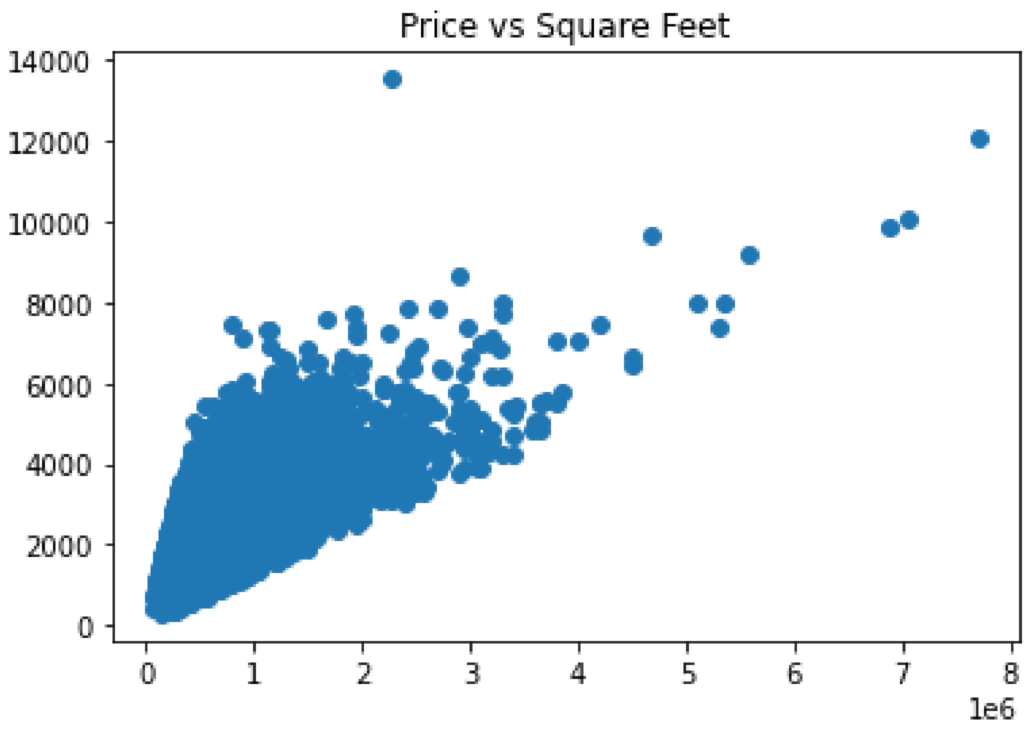
```
<Figure size 720x720 with 0 Axes>
```



```
<function seaborn.utils.despine(fig=None, ax=None, top=True, r
ight=True, left=False, bottom=False, offset=None, trim=False)>
```

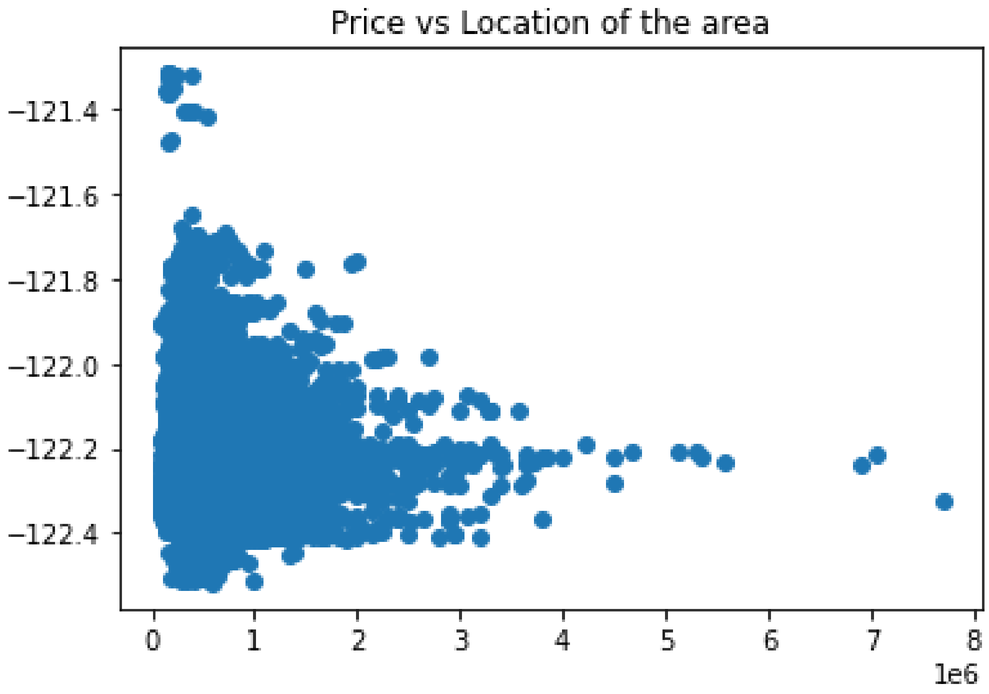
```
plt.scatter(data.price,data.sqft_living)
plt.title("Price vs Square Feet")
```

```
Text(0.5, 1.0, 'Price vs Square Feet')
```



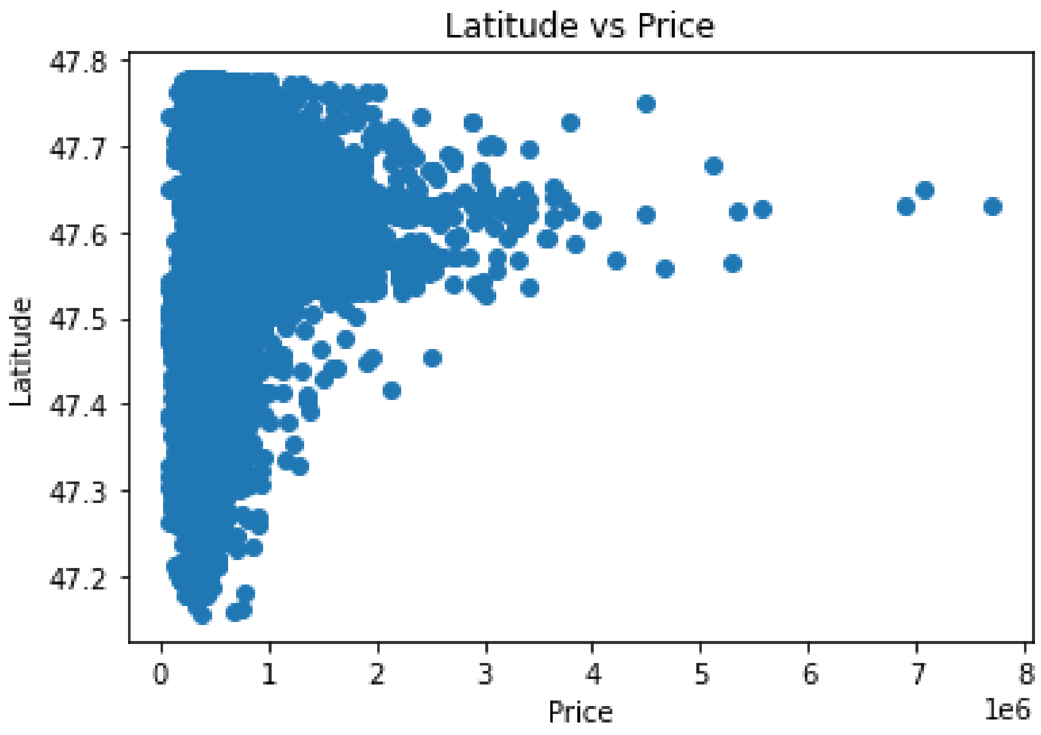
```
plt.scatter(data.price,data.long)
plt.title("Price vs Location of the area")
```

```
Text(0.5, 1.0, 'Price vs Location of the area’)
```

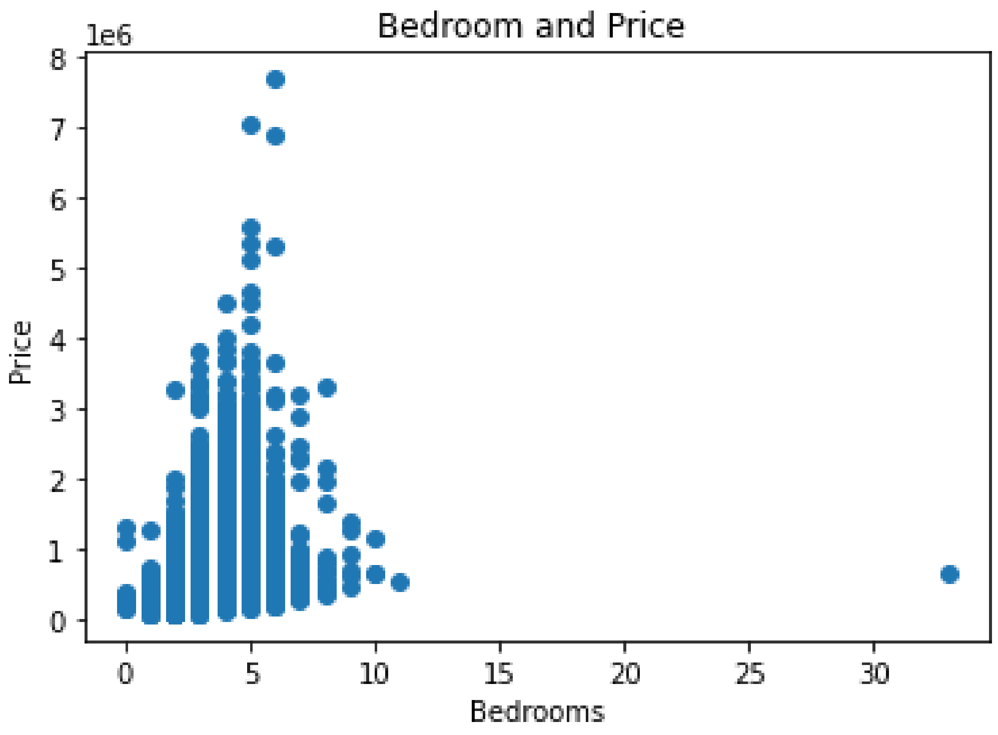


```
plt.scatter(data.price,data.lat)
plt.xlabel("Price")
plt.ylabel('Latitude')
plt.title("Latitude vs Price")
```

```
Text(0.5, 1.0, 'Latitude vs Price')
```



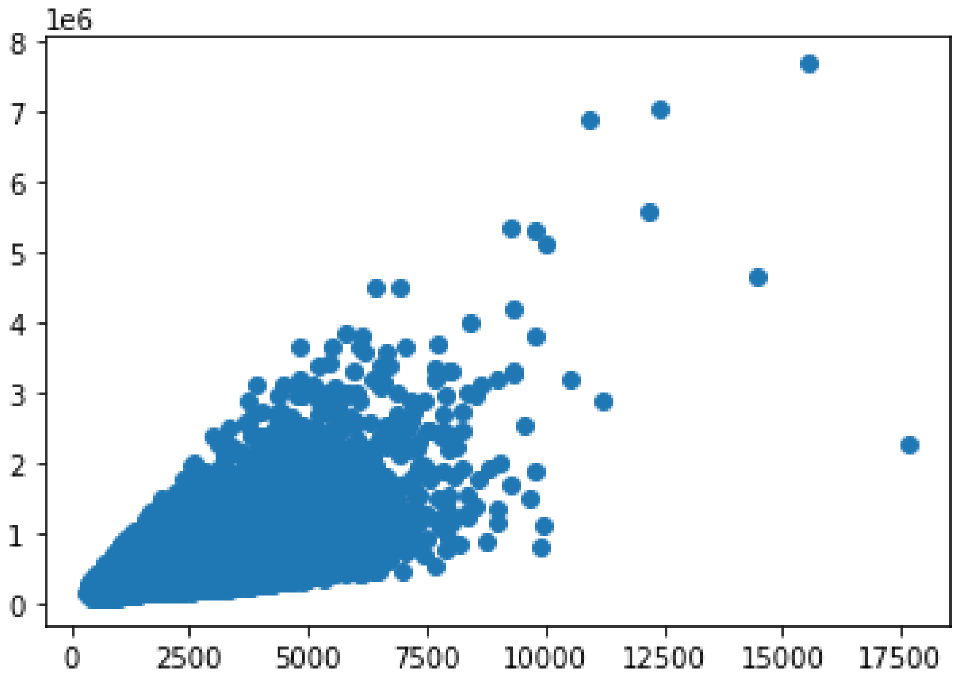
```
plt.scatter(data.bedrooms,data.price)
plt.title("Bedroom and Price ")
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
sns.despine
```



```
<function seaborn.utils.despine(fig=None, ax=None, top=True, r
ight=True, left=False, bottom=False, offset=None, trim=False)>
```

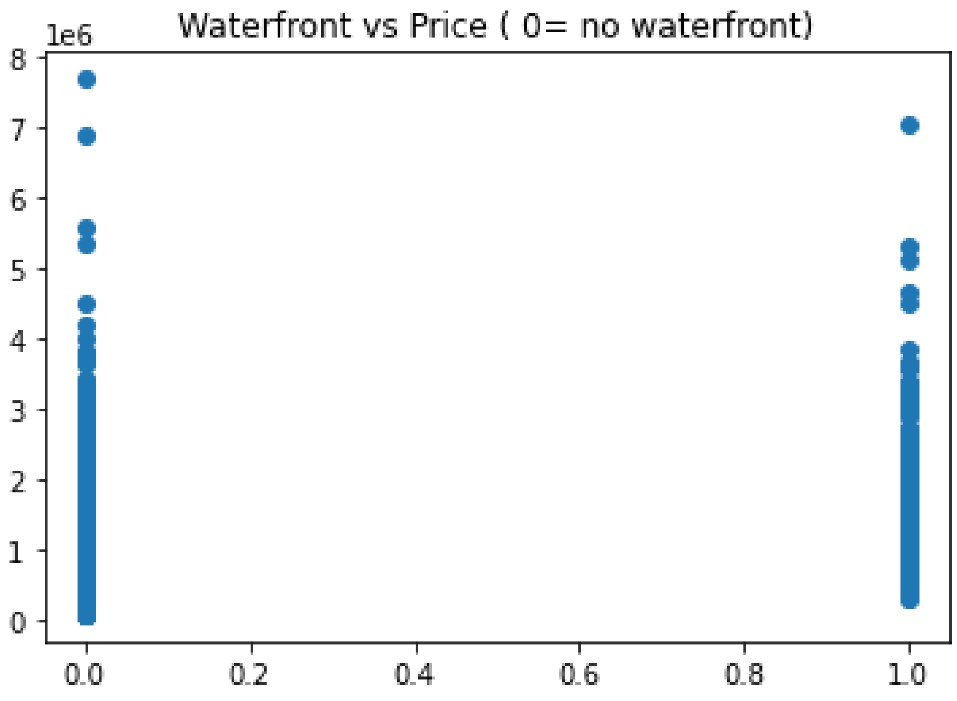
```
plt.scatter((data['sqft_living']+data['sqft_basement']),data['
price'])
```

```
<matplotlib.collections.PathCollection at 0x1452c778>
```

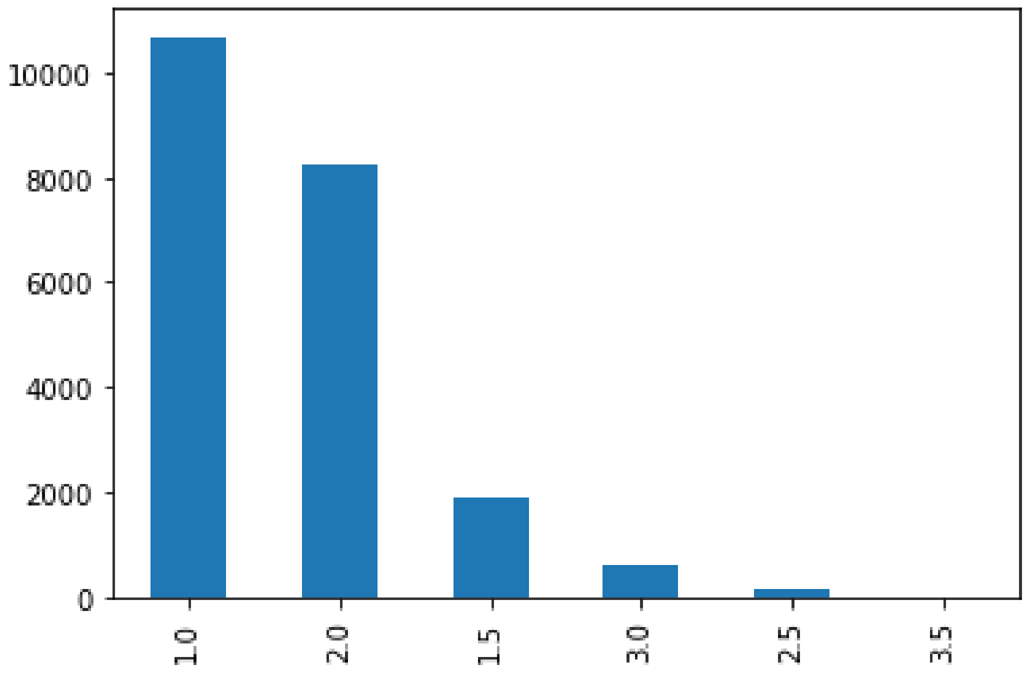


```
plt.scatter(data.waterfront,data.price)
plt.title("Waterfront vs Price ( 0= no waterfront)")
```

```
Text(0.5, 1.0, 'Waterfront vs Price ( 0= no waterfront)')
```

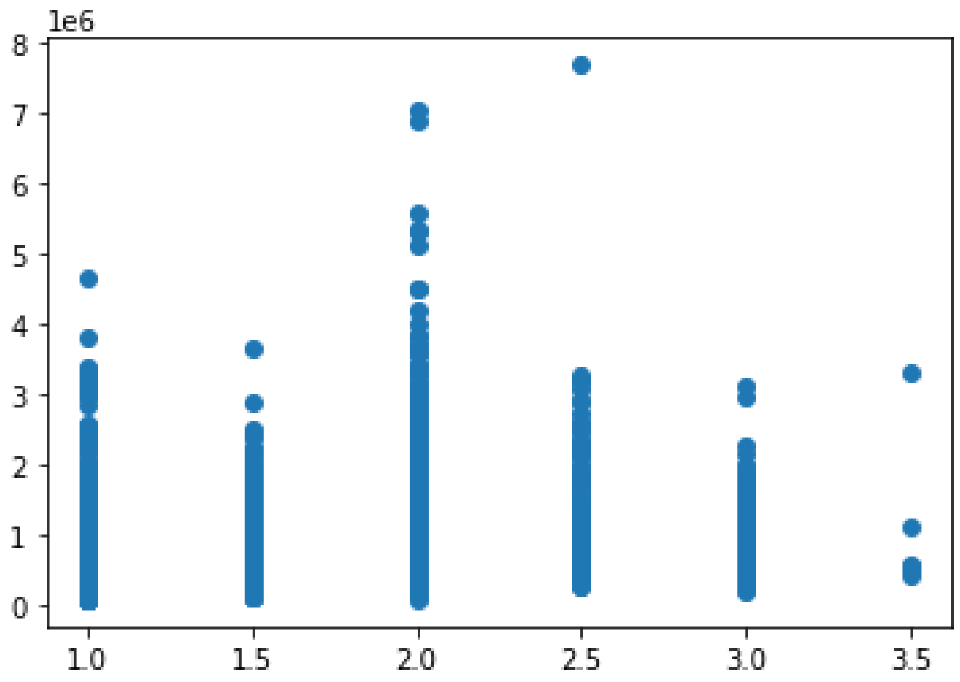


```
train1 = data.drop(['id', 'price'],axis=1)
train1.head()

              date  bedrooms  bathrooms  sqft_living  sqft_lot
floors  \
0  20141013T000000         3       1.00         1180      5650
1.0
1  20141209T000000         3       2.25         2570      7242
2.0
2  20150225T000000         2       1.00          770     10000
1.0
3  20141209T000000         4       3.00         1960      5000
1.0
4  20150218T000000         3       2.00         1680      8080
1.0

   waterfront  view  condition  grade  sqft_above  sqft_baseme
nt  yr_built  \
0           0     0          3      7        1180
0      1955
1           0     0          3      7        2170            4
00      1951
2           0     0          3      6         770
0      1933
3           0     0          5      7        1050            9
10      1965
4           0     0          3      8        1680
0      1987

   yr_renovated  zipcode      lat     long  sqft_living15  sqf
t_lot15
0             0    98178  47.5112 -122.257           1340
5650
1          1991    98125  47.7210 -122.319           1690
7639
2             0    98028  47.7379 -122.233           2720
8062
3             0    98136  47.5208 -122.393           1360
5000
4             0    98074  47.6168 -122.045           1800
7503

data.floors.value_counts().plot(kind='bar')

<matplotlib.axes._subplots.AxesSubplot at 0x144e3b50>
```



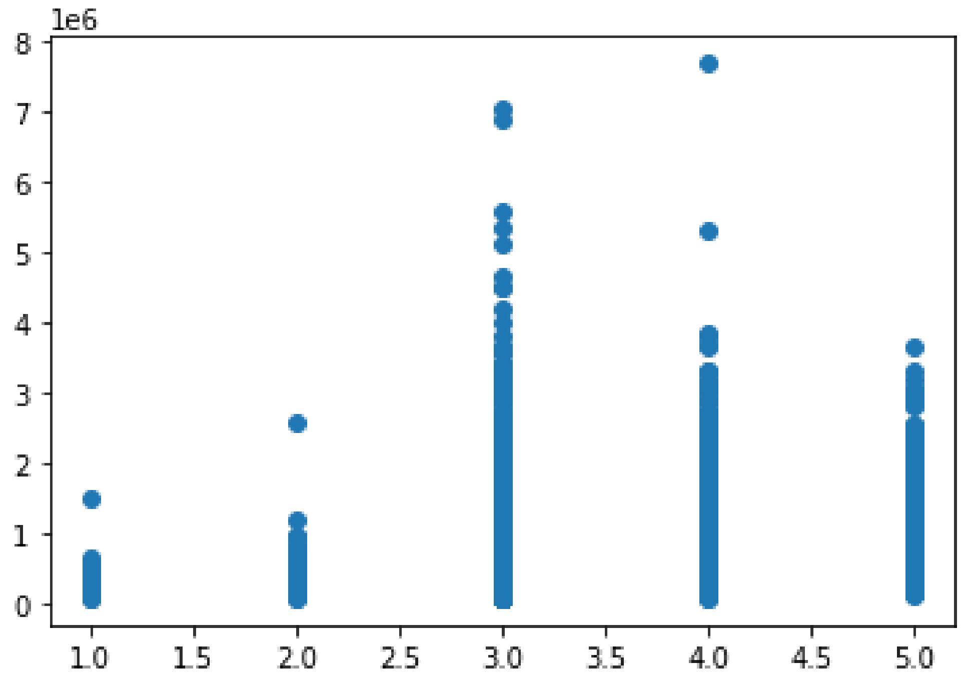
```
plt.scatter(data.floors,data.price)
```

```
<matplotlib.collections.PathCollection at 0x144ee3d0>
```



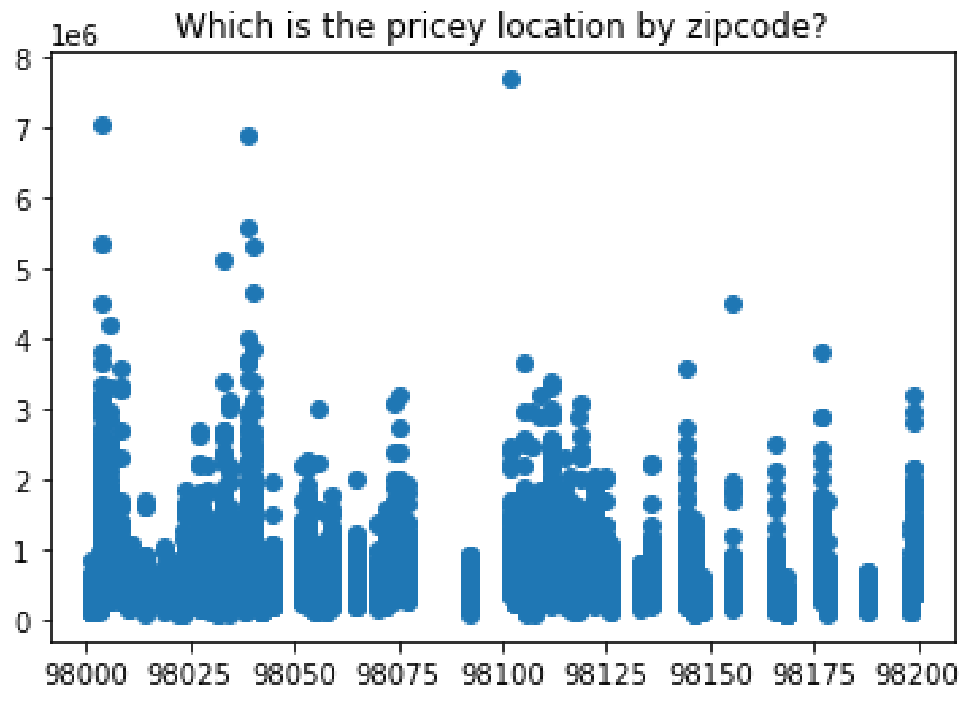
```
plt.scatter(data.condition,data.price)
```

```
<matplotlib.collections.PathCollection at 0x1271d448>
```



```
plt.scatter(data.zipcode,data.price)
plt.title("Which is the pricey location by zipcode?")
```

```
Text(0.5, 1.0, 'Which is the pricey location by zipcode?')
```



```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
labels = data['price']
conv_dates = [1 if values == 2014 else 0 for values in data.da
te ]
data['date'] = conv_dates
train1 = data.drop(['id', 'price'],axis=1)
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(train1
, labels , test_size = 0.10,random_state =2)
```

```
reg.fit(x_train,y_train)
```

```
LinearRegression()
```

```
reg.score(x_test,y_test)
```

```
0.732072105606804
```

```
params = {'n_estimators' : 400, 'max_depth' : 5, 'min_samples_
split' : 2, 'learning_rate' : 0.1, 'loss' : 'ls'}
```

```
from sklearn import ensemble
clf = ensemble.GradientBoostingRegressor(**params)
clf.fit(x_train, y_train)
```

```
GradientBoostingRegressor(max_depth=5, n_estimators=400)
```

```
clf.score(x_test,y_test)
```

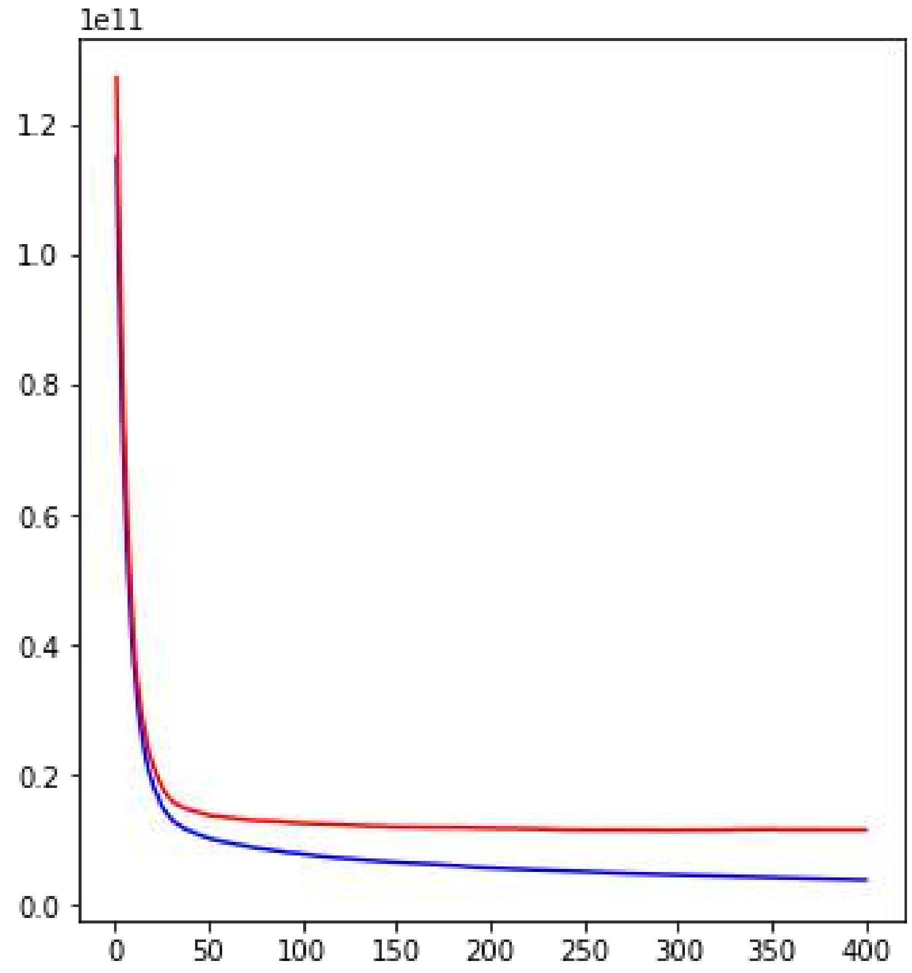
```
0.9208803759940887
```

```
t_sc = np.zeros((params['n_estimators']),dtype=np.float64)
y_pred = reg.predict(x_test)
for i,y_pred in enumerate(clf.staged_predict(x_test)):
    t_sc[i]=clf.loss_(y_test,y_pred)
```

```
testsc = np.arange((params['n_estimators']))+1
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(testsc,clf.train_score_,'b-',label= 'Set dev train')
plt.plot(testsc,t_sc,'r-',label = 'set dev test')
```

```
[<matplotlib.lines.Line2D at 0x14298718>]
```



```
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
pca = PCA()
pca.fit_transform(scale(train1))
```

```
array([[-2.64785461e+00, -4.54699955e-02, -3.16665762e-01, ...
,
        -7.94687728e-02, -8.63406595e-15,  0.00000000e+00],
       [-2.34485164e-01,  1.68297114e+00, -7.61521725e-01, ...
,
         9.81487761e-01,  1.53456462e-12,  0.00000000e+00],
       [-2.57007792e+00, -6.14344122e-01,  3.49292423e-01, ...
,
        -1.38570764e-01,  4.33715426e-13,  0.00000000e+00],
       ...,
       [-2.41985641e+00, -1.10027662e+00, -1.46293798e+00, ...
,
         9.66785881e-01, -9.95088775e-18, -0.00000000e+00],
       [ 3.32183025e-01, -1.88043103e+00, -1.04412760e+00, ...
,
        -3.97449542e-01,  2.71307522e-16,  0.00000000e+00],
       [-2.43180432e+00, -1.08505981e+00, -1.47248379e+00, ...
,
         9.53674385e-01, -4.25162784e-17, -0.00000000e+00]])
```

Result & Discussion

Model training is done with help of XGBoost, Neural Network, Las so. Using these algorithms, we train the model on the given data set . Once the model is trained, we evaluate the model's performance u sing a suitable error metric. Here, we also look for variable importa nce, i.e., which variables have proved to be significant in determini ng the target variable. And, accordingly we can shortlist the best va riables and train the model again. Finally, we test the model on the unseen data (test data) set. RMSE score of lasso model is 0.11859. We see that the lasso model has outperformed the formidable XGB oost algorithm. For further improvement, we tried to ensemble the p redictions from XGBoost and lasso model. We get RMSE as 0.117 92, which is the best score of all and hence this ensembled model is more accurate to predict the house prices.

Conclusion & Future Work

We conclude that while a standard neural network is a good tool for training a model, it is not accurate enough to be applied in real life s ituations. We thus need better supporting mechanisms that can prov ide a higher degree of accuracy. For future improvement, we can tr y to improve the accuracy of the model by combining frameworks f ruitfully in order to get the possible features of the frameworks wor king.

References

[1] G. Naga Satish, Ch. V. Raghavendran, M.D.Sugnana Rao, Ch.S rinivasulu, House Price Prediction Using Machine Learning, Intern ational Journal of Innovative Technology and Exploring Engineerin g (IJITEE)(2019)

[2] Adyan Nur Alfiyatin, Hilman Taufiq , Modeling House Price Pr ediction using Regression Analysis and Particle Swarm Optimizatio n(2017)

[3] Jingyi Mu, Fang Wu and Aihua Zhang, Housing Value Forecast ing Based on Machine Learning Methods, Hindawi Publishing Cor poration(2014)

[4] Sifei Lu, Zengxiang Li, Zheng Qin, Xulei Yang, Rick Siow Mo ng Goh, A Hybrid Regression Technique for House Prices Predictio n, Institute of High Performance Computing (IHPC)()

[5] Thuraiya Mohd, Suraya Masrom, Noraini Johari, Machine Lear ning Housing Price Prediction in Petaling Jaya, Selangor, Malaysia, International Journal of Recent Technology and Engineering (IJRT E) 2019

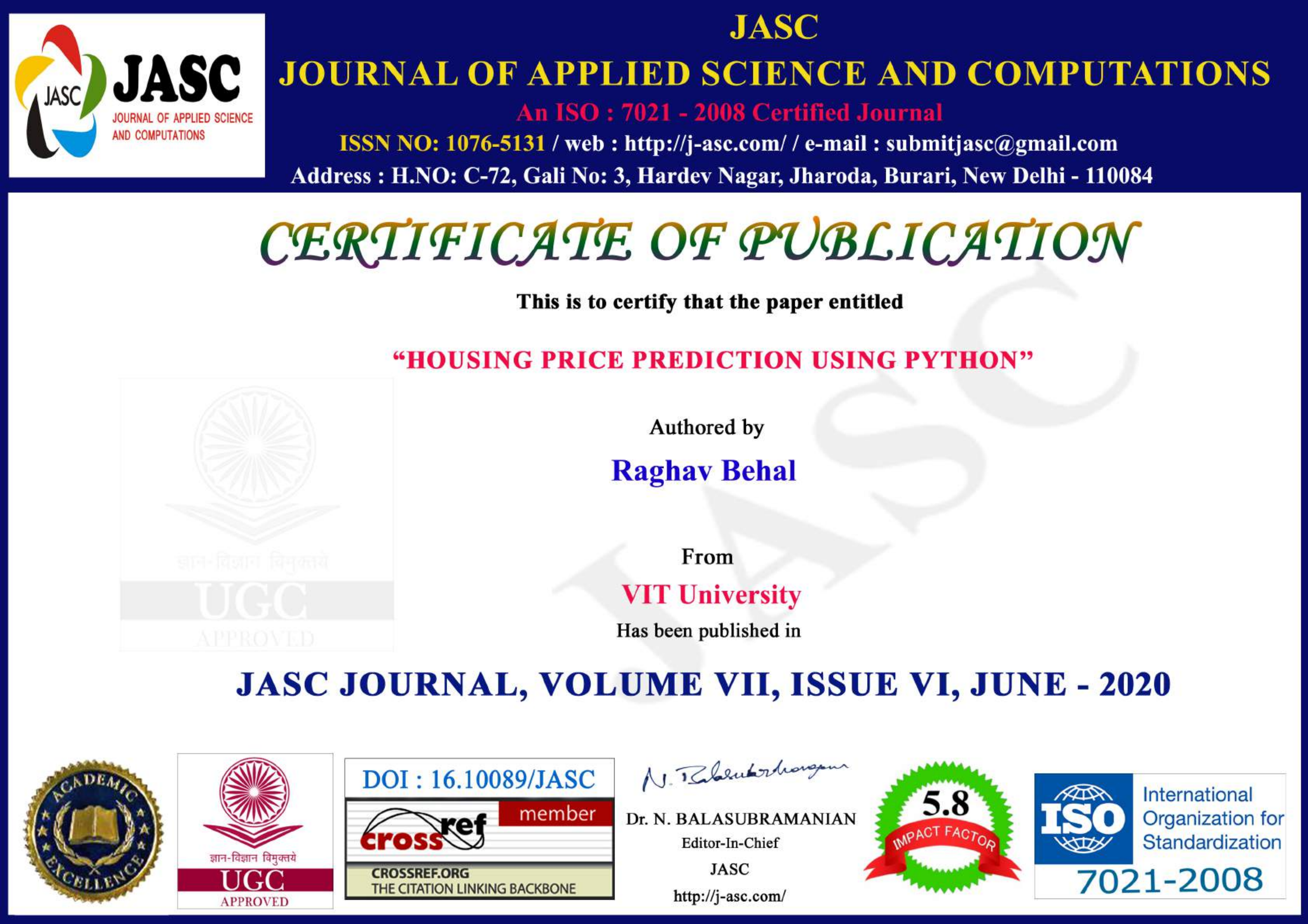
[6] Rochard J. Cebula. "The Hedonic Pricing Model Applied to the Housing Market of the City of Savannah and Its Savannah Historic Landmark District". In: The Review of Regional Studies 39.1 (2009)

[7] Azme Bin Khamis, Nur Khalidah Khalilah Binti Kamarudin, Co mparative Study On Estimate House Price Using Statistical And Ne ural Network Model, International Journal Of Scientific & Technol ogy Research Volume 3, Issue 12, December 2014

[8] Sayan Putatunda, PropTech for Proactive Pricing of Houses in C lassified Advertisements in the Indian Real Estate Market, IEEE(20 16)

[9] G. Stacy Sirmans, David A. Macpherson, and Emily N. Zietz. " The Composition of Hedonic Pricing Models". In: Journal of Real E state Literature 13.1 (2005), pp.3–43.

[10] Predicting housing prices for real estate companies, 2016



JASC

JOURNAL OF APPLIED SCIENCE AND COMPUTATIONS

An ISO : 7021 - 2008 Certified Journal

ISSN NO: 1076-5131 / web : http://j-asc.com/ / e-mail : submitjasc@gmail.com

Address : H.NO: C-72, Gali No: 3, Hardev Nagar, Jharoda, Burari, New Delhi - 110084

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"HOUSING PRICE PREDICTION USING PYTHON"

Authored by

Raghav Behal

From

VIT University

Has been published in

JASC JOURNAL, VOLUME VII, ISSUE VI, JUNE - 2020

DOI : 16.10089/JASC

crossref member

CROSSREF.ORG
THE CITATION LINKING BACKBONE

[illegible]

Dr. N. BALASUBRAMANIAN
Editor-In-Chief
JASC
http://j-asc.com/

ACADEMIC EXCELLENCE

ज्ञान-विज्ञान विमुक्तये
UGC
APPROVED

5.8
IMPACT FACTOR

ISO International Organization for Standardization
7021-2008

JASC
JOURNAL OF APPLIED SCIENCE AND COMPUTATIONS
An ISO : 7021 - 2008 Certified Journal
ISSN NO: 1076-5131 / web : http://j-asc.com/ / e-mail : submitjasc@gmail.com
Address : H.NO: C-72, Gali No: 3, Hardev Nagar, Jharoda, Burari, New Delhi - 110084

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"HOUSING PRICE PREDICTION USING PYTHON"

Authored by

Ujjwal Prakash

From

VIT University

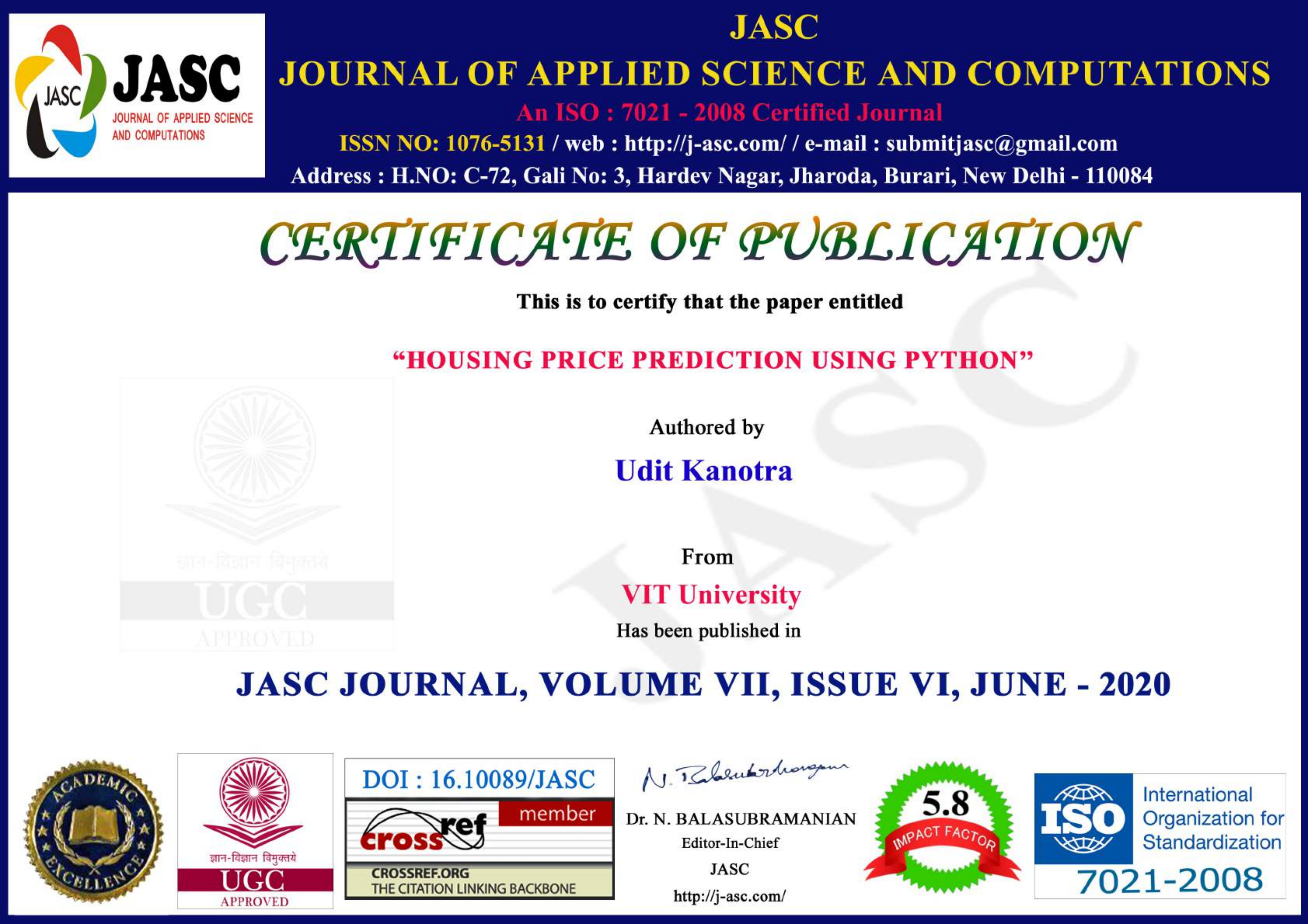
Has been published in

JASC JOURNAL, VOLUME VII, ISSUE VI, JUNE - 2020

ACADEMIC EXCELLENCE
ज्ञान-विज्ञान विमुक्तये UGC APPROVED
DOI : 16.10089/JASC
crossref member
CROSSREF.ORG THE CITATION LINKING BACKBONE

[illegible]
Dr. N. BALASUBRAMANIAN
Editor-In-Chief
JASC
http://j-asc.com/

5.8 IMPACT FACTOR
ISO International Organization for Standardization 7021-2008



JASC

JOURNAL OF APPLIED SCIENCE AND COMPUTATIONS

JASC JOURNAL OF APPLIED SCIENCE AND COMPUTATIONS

An ISO : 7021 - 2008 Certified Journal

ISSN NO: 1076-5131 / web : http://j-asc.com/ / e-mail : submitjasc@gmail.com

Address : H.NO: C-72, Gali No: 3, Hardev Nagar, Jharoda, Burari, New Delhi - 110084

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"HOUSING PRICE PREDICTION USING PYTHON"

Authored by

Udit Kanotra

From

VIT University

Has been published in

JASC JOURNAL, VOLUME VII, ISSUE VI, JUNE - 2020

ACADEMIC EXCELLENCE

ज्ञान-विज्ञान विमुक्तये UGC APPROVED

DOI : 16.10089/JASC

crossref member

CROSSREF.ORG THE CITATION LINKING BACKBONE

[illegible]

Dr. N. BALASUBRAMANIAN
Editor-In-Chief
JASC
http://j-asc.com/

5.8 IMPACT FACTOR

ISO International Organization for Standardization 7021-2008

JASC
JOURNAL OF APPLIED SCIENCE AND COMPUTATIONS
An ISO : 7021 - 2008 Certified Journal
ISSN NO: 1076-5131 / web : http://j-asc.com/ / e-mail : submitjasc@gmail.com
Address : H.NO: C-72, Gali No: 3, Hardev Nagar, Jharoda, Burari, New Delhi - 110084

CERTIFICATE OF PUBLICATION

This is to certify that the paper entitled

"HOUSING PRICE PREDICTION USING PYTHON"

Authored by

Prof. Naveen Kumar N

From

VIT University

Has been published in

JASC JOURNAL, VOLUME VII, ISSUE VI, JUNE - 2020

DOI : 16.10089/JASC
crossref member
CROSSREF.ORG
THE CITATION LINKING BACKBONE

[illegible]
Dr. N. BALASUBRAMANIAN
Editor-In-Chief
JASC
http://j-asc.com/