Ilkogretim Online - Elementary Education Online, 2021; Vol 20 (Issue 3): pp. 3706-3716
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2021.03.386

House Price Prediction Using Regression Analysis

Aditya Joshi Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India, aditya.joshi.15@hotmail.com

Bhawnesh Kumar Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India, bhawneshmca@gmail.com

Vandana Rawat Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India, vandanarawat2405@gmail.com

Mansi Srivastava Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India, mansisrivastava0109@gmail.com

Prof. (Dr) C. S. Yadav , Professor, School of Management, Graphic Era Hill University, Dehraudun

Abstract— A house is one of the basic necessities of a family and one of the most crucial long-term purchases made. House as a property is also one of the best investments. With the high scope and demand, real estate is a highly profitable business. While purchasing a house a person looks through look for their preferred price range, and has a good idea about the kind of features they will look for in their desired house such as the number of rooms, furnishing, locality, accessibility to market, hospital, and other essential services, etc. The person then decides if the house they are considering is worth the mentioned price or not. For this, a trustable price prediction system is needed. Similarly, if a person needs to sell a house, they need a prediction system to decide the price of the house with the specifications that the house has. The prediction system can give a good idea to a seller and help them set a good price by adding all the favorable specifications of their house. A price prediction system can help both the buyer and seller by predicting the price of the house according to the features it holds. It is known that we can create such predictions through Regression in Machine Learning.

Keywords— House Price Prediction, Machine Learning, Regression

I. INTRODUCTION

While looking for a house, people look at all the basic requirements and other specifications that they can get at a reasonable price. A price prediction model that can give an approximate or precise prediction of the price of a house is needed to help both buyers and sellers so that they can get predictions about the price of the house by analyzing its features. The house price prediction through the regression model will help them to decide whether the house that they want to buy is worth the price or not. While selling the house, a person can add all the specifications of the house and check through the price prediction model to get the maximum profit by selling the house.

This paper's aim is to predict the house prices of Boston using Boston Housing Data on the basis of various features or parameters and find the most accurate model among three different regression models. We then choose the best and most accurate model according to it. This will allow the buyer to get an idea of what amount of money they need to spend in order to buy a certain property or house. It will also help the seller to analyze the information regarding what is the property's real value and how they can maximize the profit by selling it.

The three regression models whose accuracy we try to analyze through the Boston Housing Data are Linear, Decision Tree, and Random Forest Regression models.

II. LITERATURE REVIEW

Over a long period of time, various studies have been conducted regarding the thorough analysis and prediction of housing prices. A regression model which was helpful in understanding and analyzing the trends of the housing prices in an area was developed [2]. House prices by using a back propagation neural network model were predicted [4]. An artificial neural network was developed that has helped in analyzing and predicting the future trends of house prices in England [1]. Multiple linear regression techniques were used to predict the price of a house in a region, and they also find the increase in the price of the land annually [5]. Why and how the time series regression techniques are helpful for the analysis and prediction of house prices was stated [6]. The real-estate prices using an autoregressive integrated moving average model were predicted [3]. Researchers used neural networks to predict house price trends [7]. A neural network was also used to find the house prices in Taipei. Rather than using normal parameters, they used economical parameters to make their house price prediction model [9]. The prediction results of the random forest machine learning approach generally utilized hedonic models consisting of multiple regression for the prediction of housing prices [10]. This paper come up with a deep learning-based house price prediction system that is carried through on the TensorFlow framework. The Adam optimizer is utilized to train the model, with the Relu activation function. The ARIMA model is then used to forecast house price trends [11]. The housing resale price prediction is carried out using various classification techniques were used [12]. Deep learning combined with XGBoost for real estate assessment with a visual perspective to foresee house prices were explored [13]. The literature attempts to glean useful information from historical house market data. Machine learning methods are used to analyze historical dealing in order to develop useful models for home buyers and sellers in Australia [14].

III. METHODOLOGY AND IMPLEMENTATION

The simple model followed in Machine Learning for Regression is illustrated in Fig. 1 This is the methodology that we would use in the implementation of our regression model in this paper.

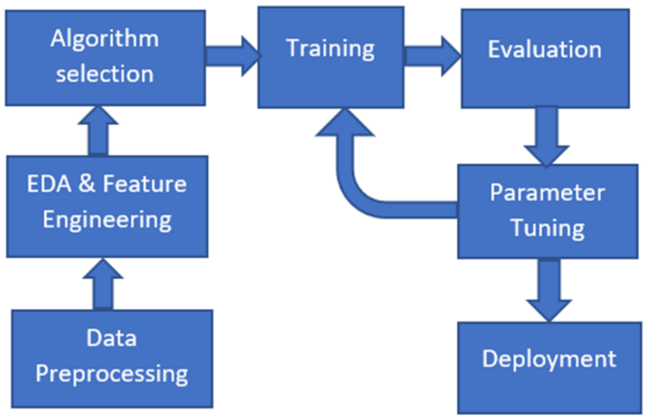


Figure 1. Proposed Model for Regression

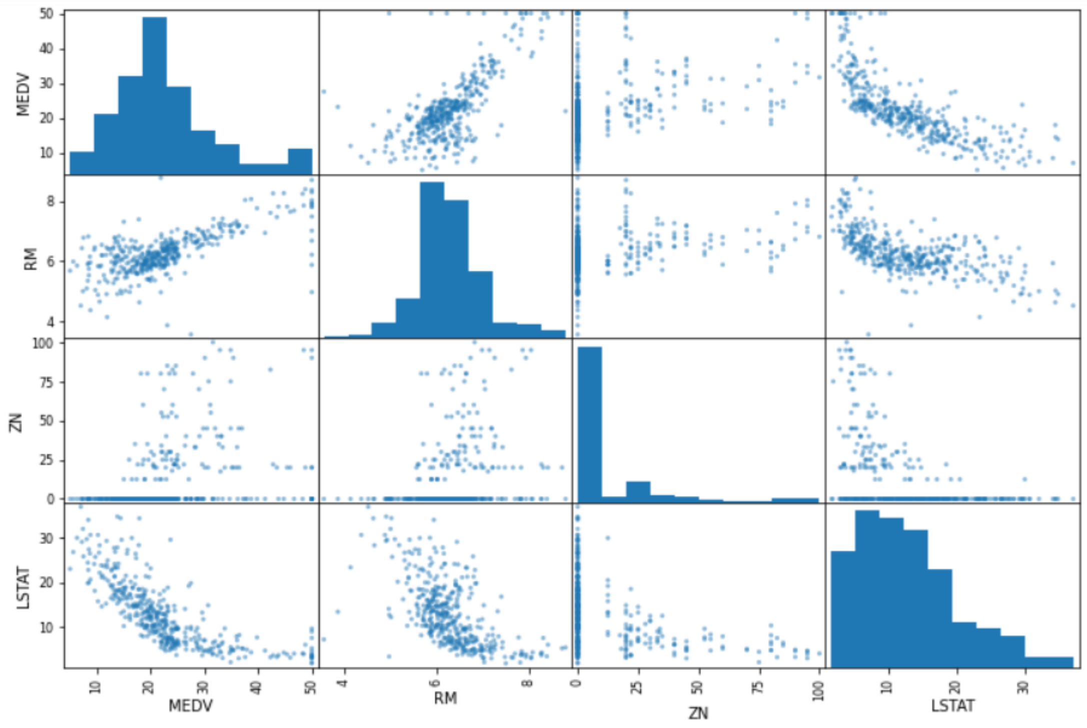


Figure 2. scatter plot of some variables like LSTAT, ZN, RM, and MEDV

A. Data Pre-processing

This approach can be divided into several phases. The first step is data collection. Here we are using the Boston Housing Dataset from the USI repository. It is used to train machine learning models. The dataset collected during this phase consists of raw and unstructured data. The dataset has 506 rows (from 0 to 505) and 14 columns. According to the dataset, the price represented by MEDV is the Median value of owner-occupied homes at $1000's. The MEDV column of the dataset is the dependent variable which is also referred to as a label, and the remaining columns are independent variables also known as features. It is necessary to make sure that all the features are storing information in the form of numbers and not in textual format. Categorical data is tough for the machine learning algorithms to understand and interpret. Techniques like One hot coding can be used to convert categorical data to numerical form. We can use the scikit-learn python library's classes and functions to do the same. Since Boston Housing Dataset already has all its data in numerical format, we don't need to change anything.

B. **EDA and Feature Engineering**

EDA is also known as Exploratory Data Analysis. It is one of the most important steps to be performed while solving any machine learning problem. It involves steps as below:

1. Analyzing the data

Understanding and examining each feature variable and making an analysis of its importance and importance to this problem. Then we just focus on the dependent variable MDV and try to know a little bit more about it. We then try to understand how the dependent variable and independent variables relate. We can make scatter plots and see the trends between the dependent label MDV and other features and see the negative and positive trends. We look at the strong-positive (increasing positively from 0) and strong-negative (deflecting negatively on the axis) features in comparison with the label MDV. These are the foremost features in predicting the price of a house. We can also correlate between any two features. Here we can see a scatter plot in Fig. 2 of some variables like LSTAT, ZN, RM, and MEDV against each other.

Through the above graphs we can conclude that feature like RM (Average number of rooms per dwelling) are strong-positive which means with the increase of number of rooms, the price of the house MEDV will increase. Similarly, the feature LSTAT (Percentage lower status of the population) is a strong negative which means that with increase of LSTAT, the price of the house MEDV will go down which is depicted in Fig. 3 and 4.

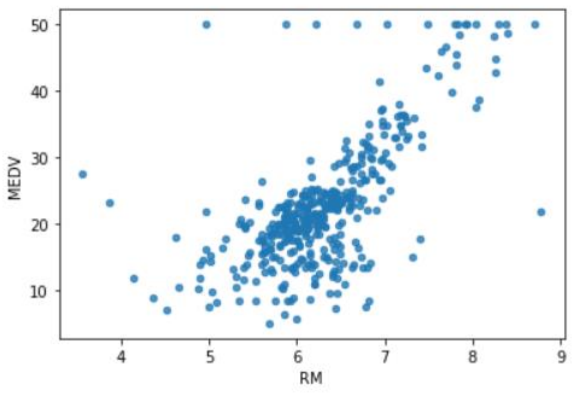


Figure 3. Scatter plot of strong positive value RM against MEDV

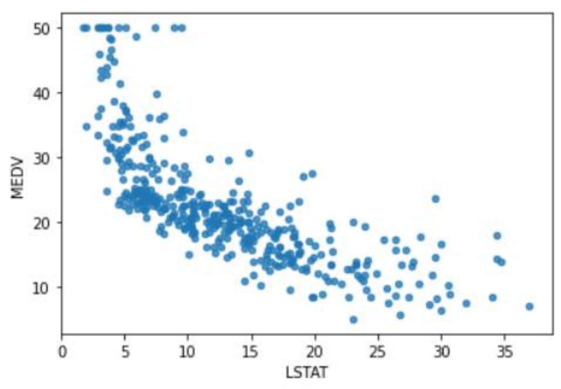


Figure 4. Scatter plot of strong negative value LSTAT against MEDV

2. Handling the missing values

After seeing the correlation between data, we should now handle any missing values in the dataset. If there are any missing data points in the datasets, we can handle them in three ways:

- Get rid of a missing data point- If the dataset is large enough to be processed in machine learning, we can get rid of the missing data points. We can remove the whole row of the data. This step couldn't and shouldn't be used if the dataset is already small.
- Get rid of the whole attribute- If the attribute is not of much significance to calculate the price of the house, we can delete the whole attribute or column. This includes the values are nearing 0 that is, they are neither strong-positive nor strong-negative attributes. But if the attribute having missing values is an important such as RM in our dataset, we shouldn't opt for this step.
- Set the value of missing value to some other value- We can set a value of the missing data with mean, median or 0. This can be done by fitting the values using the scikit-learn library's SimpleImputer class.

Scikit-learn library provides different classes and techniques for Feature Scaling such as Estimators, Transformers, Serialization etc. It can also be used to create a pipeline.

3. Splitting Data into Training and Testing Data

We now split the data into training data and testing data. We use a larger part of data to train our dataset over machine learning algorithm and a smaller part of data is used to test the data later. Most of the time, 80 percent of data is used to train on the algorithm and 20 percent data is used to test that data as depicted in Fig. 5.

Since some features can be unequally distributed between the training and testing data for example, CHAS has a value of 1 or 0 in the dataset but if while splitting, all the 0 gets on one side of split and 1 value on the other, it can result in unequal distribution of data in both split. Hence, we can use StratifiedShuffledSplit class of Scikit-learn. It is a combination of both ShuffleSplit and StratifiedKFold and keeps an equal proportion of class labels among the testing and training data.

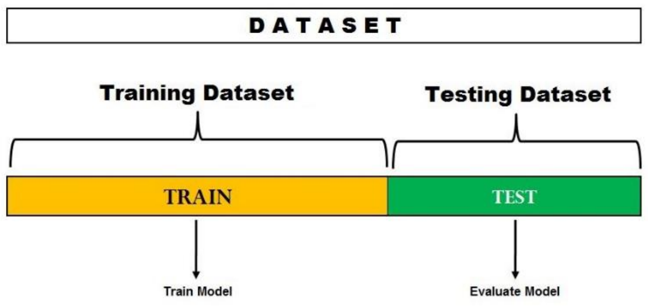


Figure 5. Splitting of Data into Train and Test set

4. Feature Engineering and Scaling

Normalization and Standardization are a part of data processing, feature scaling and cleansing techniques.

Normalization is part of data processing and cleansing techniques. The main purpose of normalization is to homogenize the data across all records and fields. This helps to create links between the input data and helps to clean up and improve the data quality. Data standardization is the process of arranging different characteristics on the same scale. In other words, standardized data can be defined as rescaling the attribute so that the mean is 0 and the standard deviation is 1.

Scikit-learn also provides the class named Standard Scaler for Standardization. Standardization is necessary as variables measured at different scales do not contribute equally to model fit and model learning capabilities and can eventually introduce bias. Therefore, to solve this potential problem, functional standardization ($\mu = 0$, $\sigma = 1$) is generally used before fitting the model.

5. Creating a pipeline

A machine learning process is a point-to-point structure that coordinates the flow of data to a machine learning model (or set of multiple models) and the output from the machine learning model (or combining models). This consist of raw data inputs, features, outputs, machine learning models and model parameters, and predictive outputs.
Building a machine learning model involves many steps, including pre-processing, standardization, etc. The scikit learn pipeline simplifies the overall machine learning modeling and testing flow. The sklearn pipeline connects everything to one object. The pipeline model can be trained and used for testing, and all pre-processing and transformations are also applied to the test data.

C. Algorithm Selection

By doing pre-processing and EDA on our data, we have data analysis, fitting the missing values and splitting the data into train and test set. Now we have to decide the algorithm or the Regression model that we are going to use. Here, we will be training our data on three different Regression models namely Linear Regressor model, Decision Tree Regressor model and Random Forest Regressor model.

D. Training and Evaluation

The purpose of training the regression model is to find a weight value that can minimize the loss function. The difference among the predicted value and the actual specification is reduced as much as possible. We can use various regression models to train the data on and check the accuracy we receive. We try our three models that is, Linear Regression model, Decision Tree Regression model and Random Forest Regression model. We can easily use these models for training our data through scikit-learn library's classes. For evaluating our model, we can use one of the evaluation metrics to find the difference between the actual and predicted values of the model. Here we will be using RMSE or Root Mean Square Error to calculate the accuracy of our predictions against the actual values. But before calculating RMSE, let's understand first both MSE (Mean Square Error) and RMSE (Root Mean Square Error)

- Mean Square Error (MSE)

MSE is the most commonly used and very simple metric, with slight variations in mean absolute error. The mean square error indicates that the square of the difference between the actual value and the predicted value has been found.
We find the absolute difference and then squared difference. MSE represents the square of the distance between the actual and predicted values. The squared value is calculated so as not to cancel the negative situation. This is the merit of MSE.

$$\mathrm{MSE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} (\mathrm{fi} - \mathrm{yi})^2 \qquad (1)$$

Where N is the number of data points
f_i the value returned by the model and
y_i the actual value for data point i.

- Root Mean Square Error (RMSE)

RMSE is as the name suggests, the square root of the mean squared error. RMSE is a measure of how well these residuals are distributed. Residual is a measure of how far a data point is from the regression line. In other words, it shows how the data points are concentrated around the line of best fit.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} (\mathrm{Predicted}_\mathrm{i} - \mathrm{Actual}_\mathrm{i})^2}{\mathrm{N}}} \qquad (2)$$

We calculate the RMSE values for all the three models. The lower the RMSE while being a good fit model (neither underfitted model or overfitted model) the better the model is.

E. Parameter Tuning

Cross-validation, a widely used statistical technique, is used to calculate the performance (or accuracy) of a machine learning model. It is used to protect predictive models from overfitting, particularly when the available data is limited. Cross-validation requires a set number of data convolutions (or partitions), analyses each convolution, and then averages the total error estimates.

Cross-validation helps when there is a case of overfitting. Overfitting occurs when the machine learning model attempts to cover all data points, or data points that exceed the required data points in a particular dataset. In such a case, the model starts showing a zero error. As an outcome, the model begins to catch noise and faulty values in the dataset, all of which reduce the effectiveness and accuracy of the model. The overfitting model has a low bias and a large variance.
While training data on Decision Tree Regressor here, on evaluating the model, the MSE comes out to be zero. This means that the data is overfitted in the model and the model has learned the noise too. We want our model to learn the trend not the noise. We want our data to be the best fit that is, neither overfitted nor underfitted. Underfitting is when our data doesn't even learn the trend of the data. This case is illustrated in Fig. 6

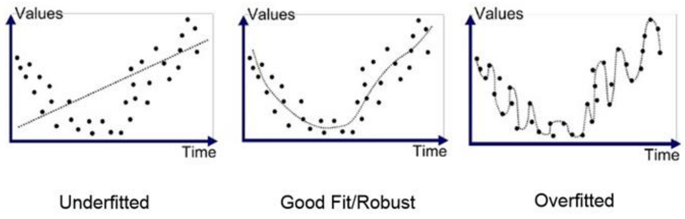


Figure 6. Visualization of underfitting, best fit, and overfitting

There are many ways to perform cross-validation such as Hold out, K-fold cross-validation, etc. Here we will be using K-Fold cross-validation and creating 10 folds. K-fold cross-validation is a data partitioning strategy that allows you to effectively utilize datasets to create more general techniques. The prime purpose of all kinds of machine learning is to develop more general models that will be efficient with invisible data. You can build a complete model on your training data with 100% accuracy or 0 errors, but it may not be generalized to hidden data. So, it's not a good model. Overwrite training data. Machine learning is all about generalization. That is, model performance can only be measured using data points that have never been used during the training process. Therefore, we often split the data into training sets and test sets.

F. Re-evaluating the Model

After re-evaluating the model again, we get the RMSE scores of the 10 folds of cross validation. We calculate the mean and standard deviation using the RMSE scores for the Decision Tree Regressor model. Similarly, we calculate the scores for the other two models. We now got the scores of all the three models. The model with the smallest mean and standard deviation will be considered the best. On calculating the mean and deviation of all the three models, we find that the model scores of Random Forest regressor have the least error and standard deviation. Hence, among the three, Random Forest Regressor is the most accurate and works the best followed by Decision Tree Regressor and then Linear Regression.

1. Linear Regression:

```
  Scores: [4.22235612 4.26438649 5.09424333 3.83081183 5.37600331 4.41092152 7.47272243
5.48554135 4.14606627   6.0717752]
   Mean:  5.037482786117751
   Standard deviation:  1.0594382405606955
```

2. Decision Tree:

Scores: [3.98986521 5.38106436 5.00209712 4.04969135 4.00543381 3.17423219 5.1558462 3.88252495 3.45666024 4.09206549]
Mean: 4.218948091685954
Standard deviation: 0.6906608179946986

3. Random Forest Regressor:

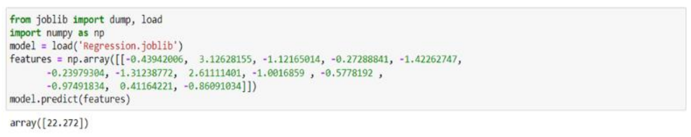
Scores: [2.89105431 2.74750626 4.39672493 2.582941 3.48965374 2.64489624 4.64366757 3.34816971 3.11595216 3.30343453]
Mean: 3.316400045347007
Standard deviation: 0.6703397062559692

G. Deployment

The joblib library can be used to save and load machine learning models. Joblib also allows sklearn to take full advantage of multiple cores of the computer and speed up training.
You can save the machine learning model using the dump () method available in the joblib library. Objects are serialized to disk. This accepts two parameters.

- object_to_be_serialized- Model object serialized to disk.
- File_name-The name of the target file to save the model to disk. We can just pass the file name. We don't need to create a file object. We can now model and predict the price for the house by entering the values of the features.

The code can be written as below in Fig. 7:



```
from joblib import dump, load
import numpy as np
model = load('Regression.joblib')
features = np.array([[-0.43942006,  3.12628155, -1.12165014, -0.27288841, -1.42262747,
       -0.23979304, -1.31238772,  2.61111401, -1.0016859 , -0.5778192 ,
       -0.97491834,  0.41164221, -0.86091034]])
model.predict(features)
```

```
array([22.272])
```

Figure 7. Dumping and Saving the Model

IV. CONCLUSION

In this paper, three different regression models namely, Linear Regression Model, Decision Tree Regressor, and Random Forest Regressor are adopted to predict the price of different houses. We checked for the most accurate model among the three and found that Random Forest Regressor gave the lowest mean error and standard deviation and hence, is the best and most accurate model to predict the price of the houses. It comes under the area of supervised learning which is one of the types of machine learning. All the steps required for the successful completion of the house price prediction system have been completed. Some improvements and additions are possible, and many other models are much more accurate than we have found. We used the Boston dataset, which is a very popular dataset, so you can use other datasets to predict home prices. Another thing you can do is increase the ability to update records on a regular basis. This makes the prediction system much more accurate and correct.

V. References

[1] I. D. Wilson, S. D. Paris, J. A. Ware, and D. H. Jenkins, “Residential property price time series forecasting with neural networks,” in Applications and Innovations in Intelligent Systems IX. Springer, 2002, pp. 17–28.

[2] J. Birch and M. Sunderman, “Estimating price paths for residential real estate,” Journal of Real Estate Research, vol. 25, no. 3, pp. 277–300, 2003.3

[3] O. Haoting, “Real estate price index based on arma model,” Statistics and Decision, no. 7, 2007.

[4] H. Zhangming, “Research on forecasting real estate price index based on neural networks,” Journal of the Graduates Sun Yat Sen University, vol. 27, 2006.

[5] V. Sampathkumar and H. Santhi, “Artificial neural network modelling of land price at sowcarpet in chennai city,” International Journal of Computer Science & Emerging Technologies, vol. 1, pp. 44–49, 2010.

[6] B. Mundy and J. A. Kilpatrick, “Factors influencing cbd land prices,” Real Estate Issues, vol. 25, no. 3, pp. 39–39, 2000.

[7] J. Wang and P. Tian, “Real estate price indices forecast by using wavelet neural network,” Computer Simulation, vol. 2, 2005.

[8] N. Bhagat, A. Mohokar, and S. Mane, “House price forecasting using data mining,” International Journal of Computer Applications, vol. 152, no. 2, pp. 23–26, 2016.

[9] L. Li and K.-H. Chu, “Prediction of real estate price variation based on economic parameters,” in 2017 International Conference on Applied System Innovation (ICASI). IEEE, 2017, pp. 87–90.

[10] M. ˇCeh, M. Kilibarda, A. Lisec, and B. Bajat, “Estimating the performance of random forest versus multiple regression for predicting prices of the apartments,” ISPRS international journal of geo-information, vol. 7, no. 5, p. 168, 2018.

[11] F. Wang, Y. Zou, H. Zhang, and H. Shi, “House price prediction approach based on deep learning and arima model,” in 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT). IEEE, 2019, pp. 303–307.

[12] P. Durganjali and M. V. Pujitha, “House resale price prediction using classification algorithms,” in 2019 International Conference on Smart Structures and Systems (ICSSS). IEEE, 2019, pp. 1–4

[13] Y. Zhao, G. Chetty, and D. Tran, “Deep learning with xgboost for real estate appraisal,” in 2019 IEEE symposium series on computational intelligence (SSCI). IEEE, 2019, pp. 1396–1401

[14] T. D. Phan, “Housing price prediction using machine learning algorithms: The case of melbourne city, australia,” in 2018 International conference on machine learning and data engineering (iCMLDE). IEEE, 2018, pp. 35–42