

MACHINE LEARNING CA1 REPORT

on

House Price Prediction
Using Machine Learning

**School of Computer Science &
Engineering**

**LOVELY PROFESSIONAL
UNIVERSITY**

APRIL 2022



L LOVELY
P ROFESSIONAL
U NIVERSITY

SUBMITTED TO – Dr. Rahul

Name – Vasu Goel

Registration Number- 11907868

Roll Number – RKM045A05

Table of Content

S.No	Title	Page Number
1	Abstract	3
2	Introduction	4
3	Dataset	5
4	Features Description	5
5	Tools and Libraries used	6
6	Approach Taken	6-7
7	Data Cleaning	7-8
8	Exploratory Data Analysis	8
9	Model Selection	8-9
10	Screenshots	9-11
11	Conclusion	12
12	References	13
13	Github Link	13

Abstract

Real estate is the least transparent industry in our ecosystem. Housing prices keep changing day in and day out and sometimes are hyped rather than being based on valuation. It is one of the prime fields to apply the ideas of machine learning on how to enhance and foresee the costs with high accuracy. The objective of the paper is the prediction of the market value of a real estate property. This system helps find a starting price for a property based on the geographical variables. By breaking down past market patterns and value ranges, and coming advancements future costs will be anticipated. This examination means to predict house prices in Boston city with XGBoost regressor. It will help clients to put resources into a bequest without moving towards a broker. The result of this research proved that the XGBoost regressor gives an accuracy of 98%. Predicting housing prices with real factors is the main crux of our research project. Here we aim to make our evaluations based on every basic parameter that is considered while determining the price. We use various regression techniques in this pathway, and our results are not sole determination of one technique rather it is the weighted mean of various techniques to give most accurate results. The results proved that this approach yields minimum error and maximum accuracy than individual algorithms applied. We also propose to use real-time neighborhood details using Google maps to get exact real-world valuations.

House Price Index (HPI) is commonly used to estimate the changes in housing price. Since housing price is strongly correlated to other factors such as location, area, population, it requires other information apart from HPI to predict individual housing price. There has been a considerably large number of papers adopting traditional machine learning approaches to predict housing prices accurately, but they rarely concern about the performance of individual models and neglect the less popular yet complex models. As a result, to explore various impacts of features on prediction methods, this paper will apply both traditional and advanced machine learning approaches to investigate the difference among several advanced models. This paper will also comprehensively validate multiple techniques in model implementation on regression and provide an optimistic result for housing price prediction.

Introduction

House is one of human life's most essential needs, along with other fundamental needs such as food, water, and much more. Demand for houses grew rapidly over the years as people's living standards improved. While there are people who make their house as an investment and property, yet most people around the world are buying a house as their shelter or as their livelihood. Housing markets have a positive impact on a country's currency, which is an important national economy scale. Home owners will purchase goods such as furniture and household equipment for their home, and homebuilders or contractors will purchase raw material to build houses to satisfy house demand, which is an indication of the economic wave effect created by the new house supply. Besides that, consumers have capital to make a large investment, and the construction industry is in good condition can be seen through a country's high level of house supply.

House price prediction can be done by using a multiple prediction models (Machine Learning Model) such as support vector regression, artificial neural network, and more. There are many benefits that home buyers, property investors, and house builders can reap from the house-price model. This model will provide a lot of information and knowledge to home buyers, property investors and house builders, such as the valuation of house prices in the present market, which will help them determine house prices. Meanwhile, this model can help potential buyers decide the characteristics of a house they want according to their budget. Previous studies focused on analyzing the attributes that affect house price and predicting house price based on the model of machine learning separately. However, this article combines such a both predicting house price and attributes together.

The problem falls under the category of supervised learning algorithms. The dataset we'll be using is the “**Boston Housing**” dataset. The dataset comprises 13 input features and one target feature. The input features include features that may or may not impact the price.

➤ Dataset

The Boston data frame has 506 rows and 14 columns. Each row comprises one data-point and contains details about a plot. Various features affect the pricing of a house. The Boston housing dataset has 14 features, out of which we'll use 13 to train the model. The 14th feature is the price, which we'll use as our target variable. The table gives the list of features included in the dataset, along with their respective descriptions.

➤ Features Description

crim	<i>per capita crime rate by town</i>
zn	<i>proportion of residential land zoned for lots over 25,000 sq.ft.</i>
indus	<i>proportion of non-retail business acres per town.</i>
chas	<i>Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).</i>
nox	<i>nitrogen oxides concentration (parts per 10 million).</i>
rm	<i>average number of rooms per dwelling.</i>
age	<i>proportion of owner-occupied units built prior to 1940</i>
dis	<i>weighted mean of distances to five Boston employment centers.</i>
tax	<i>full-value property-tax rate per \$10,000.</i>
ptratio	<i>pupil-teacher ratio by town.</i>
black	<i>$1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town.</i>
lstat	<i>lower status of the population (percent).</i>
medv	<i>median value of owner-occupied homes in \$1000s.</i>

➤ Tools

I used Python and Jupyter notebooks for the competition. Jupyter notebooks are popular among data scientist because they are easy to follow and show your working steps. Please be aware this code is not for production purposes, it doesn't follow software engineering best practices. I've sacrificed that somewhat for explainability.

➤ Libraries

These are frameworks in python to handle commonly required tasks. I Implore any budding data scientists to familiarise themselves with these libraries:

Pandas: For handling structured data

Scikit Learn: For machine learning

Numpy: For linear algebra and mathematics

Seaborn: For data visualization and exploratory data analysis

Matplotlib: is a graphical plotting library for python

XGBoost: provides a high performance implementation of gradient boosted decision tree

➤ Approach Taken

We'll begin by loading the data. Since we're using an inbuilt dataset, we'll be calling the `load_boston` function from the `sklearn.datasets` module. We load the data into the data variable. Once the data is loaded, we separate the data and target attributes of the data variable. We store them in variables `data` and `target`, respectively. Once we have the data and target values in 2 different variables, we can divide the data into two parts: the testing data and training data. The theory behind dividing the dataset into two parts is to ensure the model doesn't overfit the training data. Otherwise, the model will perform well on the training data and perform poorly on the test data. This means that the model has learned the training data so well that it cannot generalize new data points. This should be avoided.

Once we have the dataset split into training and testing sets, we can pre-process the data. Pre-processing involves scaling the values and converting the categorical values into numerical values. For example, there is a variable in the given dataset that indicates whether the Charles river is close to the house or not. This variable takes the values Near and Far. We need to convert this into a numerical value. To do this, we can use the LabelEncoder function available in the pre-processing module of sklearn. This will replace the column with numerical values of 0 and 1, respectively. 0 indicates Near, and 1 indicates Far. Once we perform the pre-processing of the dataset, we can fit the data to the model. We begin with instantiating an object of the XGB Regressor class. This is available in the sklearn.ensemble module. We use the fit method to fit the data to the model. Once the model is fit, we evaluate the model's performance on the test set we got earlier. We use the predict method present in the XGB Regressor class. The predict method takes in the test input data and predicts an output. Using the predicted output and the actual output known from the dataset, we compute the test accuracy. Another useful evaluation metric is the Mean Squared Error. The Means Squared Error (MSE) loss estimates how far the prediction is from the mean of the output. Computing the MSE gives us an idea about the performance of the algorithm.

➤ **Data Cleaning**

- **Duplicates & NaNs:** I started by removing duplicates from the data, checked for missing or NaN (not a number) values. It's important to check for NaNs (and not just because it's socially moral) because these cause errors in the machine learning models.
- **Categorical Features:** There are a lot of categorical variables that are marked as N/A when a feature of the house is nonexistent. For example, when no alley is present. I identified all the cases where this was happening across the training and test data and replaced the N/As with something more descriptive. N/As can cause errors with machine learning later down the line so get rid of them.
- **Date Features:** For this exercise dates would be better used as categories and not integers. After all, it's not so much the magnitude that we care about but rather that the dates represent different years. Solving this problem is simple, just convert the numeric dates to strings.

- **Decoded Variables:** Some categorical variables had been number encoded. The problem here is that the machine learning algorithm could interpret the magnitude of the number to be important rather than just interpreting it as different categories of a feature. To solve the problem, I reverse engineered the categories and recoded them.

➤ **Exploratory Data Analysis (EDA)**

This is where our data visualisation journey often begins. The purpose of EDA in machine learning is to explore the quality of our data. A question to keep in mind is; are there any strange patterns that leave us scratching our heads?

- **Labels:** I plotted sales price on a histogram. The distribution of sale prices is right skewed, something that is expected. In your neighborhood it might not be unusual to see a few houses that are relatively expensive. Here I perform my first bit of feature engineering (told you the process was messy). I'll apply a log transform to sales price to compress outliers making the distribution normal. Outliers can have devastating effects on models that use loss functions minimising squared error. Instead of removing outliers try applying a transformation.
- **Correlation:** It's often good to plot a correlation matrix to give you an idea of relationships that exist in your data. It can also guide your model building. For example, if you see a lot of your features are correlated with each other you might want to avoid linear regression.
- **Categorical Relation:** Sales price appears to be approximately normally distributed within each level of each category. No observations appear, untoward. Some categories contain little to no data, whilst other show little to no distinguishing ability between sales class. See full project on GitHub for data visualisation.

➤ **Model Selection**

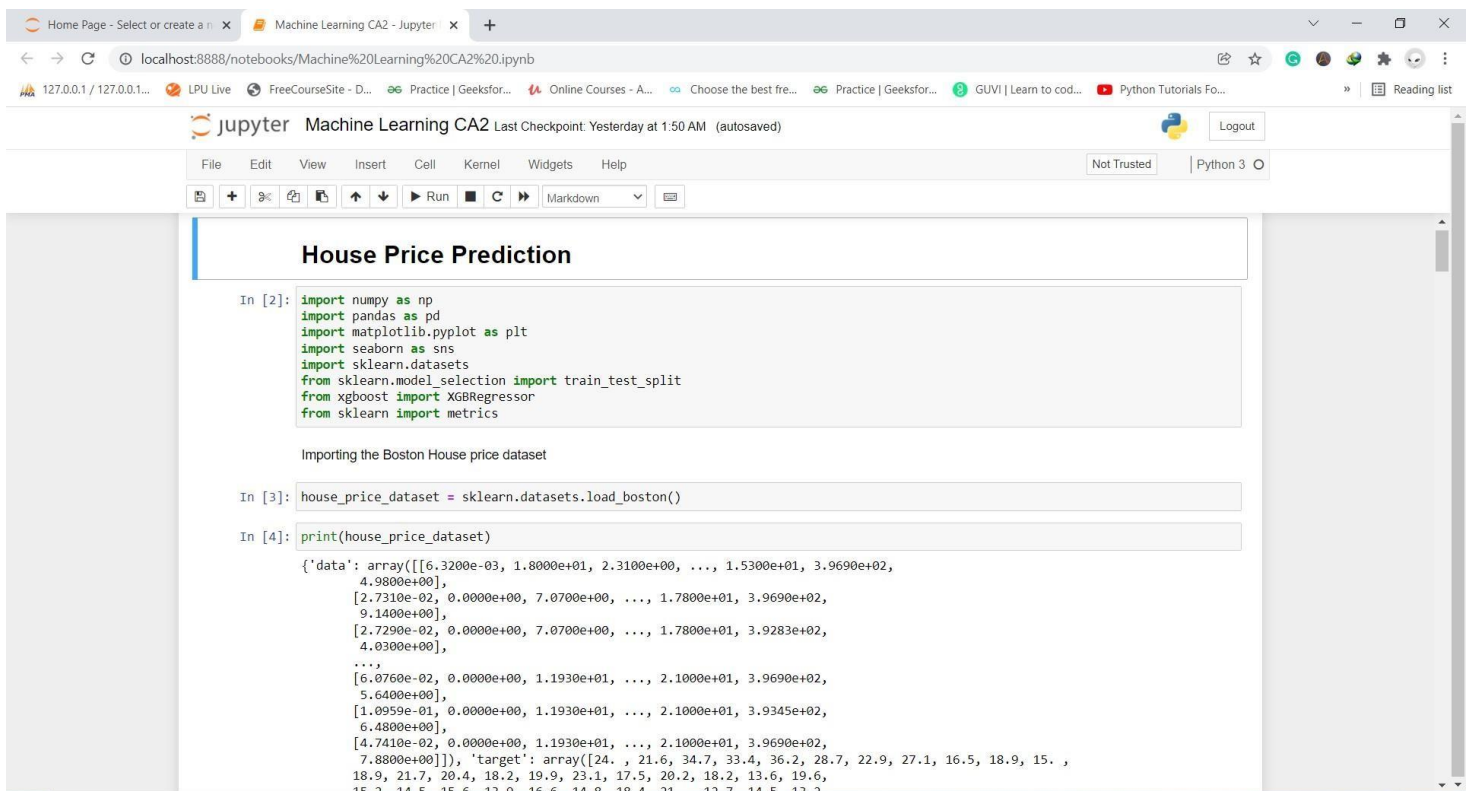
As mentioned at the start of the article the task is supervised machine learning. We know it's a regression task because we are being asked to predict a numerical outcome (sale price).

Therefore, I approached this problem with three machine learning models. Decision tree and gradient boosting machines. I used the decision tree as my baseline model then built on this experience to tune my candidate models. This approach saves a lot of time as decision trees are quick to train and can give you an idea of how to tune the hyper parameters for my candidate models.

- Decision Tree: A tree algorithm used in machine learning to find patterns in data by learning decision rules.
- Gradient Boosting Machines: A type of boosting method that uses a combination of decision tree in series. Each tree is used to predict and correct the errors by the preceding tree additively.

Screenshots

➤ Importing libraries and loading the dataset



The screenshot shows a Jupyter Notebook interface with the title 'Machine Learning CA2'. The notebook contains the following code cells:

```
In [2]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

Importing the Boston House price dataset

```
In [3]: house_price_dataset = sklearn.datasets.load_boston()

In [4]: print(house_price_dataset)
```

The output of the print statement shows the structure of the dataset:

```
{'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
4.9800e+00],
[2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
9.1400e+00],
[2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
4.0300e+00],
...,
[6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
5.6400e+00],
[1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
6.4800e+00],
[4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
7.8800e+00]]), 'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
15.2, 14.5, 15.6, 12.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2])}
```

➤ Loading the dataset to a pandas data frame

Machine Learning CA2 - Jupyter

localhost:8888/notebooks/Machine%20Learning%20CA2%20.ipynb

Machine Learning CA2 Last Checkpoint: Yesterday at 1:50 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
kages\\sklearn\\datasets\\data\\boston_house_prices.csv'})
```

In [5]: `# Loading the dataset to a pandas dataframe`
`house_price_dataframe = pd.DataFrame(house_price_dataset.data, columns = house_price_dataset.feature_names)`

In [6]: `#Printing first 5 rows of the DataFrame`
`house_price_dataframe.head()`

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [7]: `# add the target (price) column to the DataFrame`
`house_price_dataframe['Price'] = house_price_dataset.target`

In [8]: `house_price_dataframe.head()`

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [9]: `#Checking the number of rows and columns in the data frame`

➤ Understanding the correlation between various features in the dataset

Machine Learning CA2 - Jupyter

localhost:8888/notebooks/Machine%20Learning%20CA2%20.ipynb

Machine Learning CA2 Last Checkpoint: Yesterday at 1:50 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
In [12]: correlation = house_price_dataframe.corr()
```

In [13]: `#Constructing the heatmap to understand the correlation`
`plt.figure(figsize=(10,10))`
`sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues')`

Out[13]: <AxesSubplot:>

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B		
CRIM	1.0	-0.2	0.4	-0.1	0.4	-0.2	0.4	-0.4	0.6	0.6	0.3	-0.4	0.5	-0.4
ZN	-0.2	1.0	-0.5	-0.0	-0.5	0.3	-0.6	0.7	-0.3	-0.3	-0.4	0.2	-0.4	0.4
INDUS	0.4	-0.5	1.0	0.1	0.8	-0.4	0.6	-0.7	0.6	0.7	0.4	-0.4	0.6	-0.5
CHAS	-0.1	-0.0	0.1	1.0	0.1	0.1	0.1	-0.1	-0.0	-0.0	-0.1	0.0	-0.1	0.2
NOX	0.4	-0.5	0.8	0.1	1.0	-0.3	0.7	-0.8	0.6	0.7	0.2	-0.4	0.6	-0.4
RM	-0.2	0.3	-0.4	0.1	-0.3	1.0	-0.2	0.2	-0.2	-0.3	-0.4	0.1	-0.6	0.7
AGE	0.4	-0.6	0.6	0.1	0.7	-0.2	1.0	-0.7	0.5	0.5	0.3	-0.3	0.6	-0.4
DIS	-0.4	0.7	-0.7	-0.1	-0.8	0.2	-0.7	1.0	-0.5	-0.5	-0.2	0.3	-0.5	0.2
RAD	0.6	-0.3	0.6	-0.0	0.6	-0.2	0.5	-0.5	1.0	0.9	0.5	-0.4	0.5	-0.4
TAX	0.6	-0.3	0.7	-0.0	0.7	-0.3	0.5	-0.5	0.9	1.0	0.5	-0.4	0.5	-0.5
PTRATIO	0.3	-0.4	0.4	-0.1	0.2	-0.4	0.3	-0.2	0.5	0.5	1.0	-0.2	0.4	-0.5
B	-0.4	0.2	-0.4	0.0	-0.4	0.1	-0.3	0.3	-0.4	-0.4	-0.2	1.0	-0.4	0.3

➤ Splitting the data into training and test data and applying XGBoost Regressor

Machine Learning CA2 - Jupyter

localhost:8888/notebooks/Machine%20Learning%20CA2%20.ipynb

Machine Learning CA2 Last Checkpoint: Yesterday at 1:50 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

Name: Price, Length: 506, dtype: float64

Splitting the data into training data and test data

```
In [16]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
```

```
In [17]: print(X.shape, X_train.shape, X_test.shape)
```

```
(506, 13) (404, 13) (102, 13)
```

Model Training

XGBoost Regressor

```
In [18]: #Loading the model
model = XGBRegressor()
```

```
In [19]: #Training the model with X_train
model.fit(X_train, Y_train)
```

```
Out[19]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
gamma=0, gpu_id=-1, importance_type=None,
interaction_constraints='', learning_rate=0.300000012,
max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
monotone_constraints='()', n_estimators=100, n_jobs=4,
num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
validate_parameters=1, verbosity=None)
```

Evaluation

➤ Checking the r squared error and also visualizing the actual and predicted prices

Machine Learning CA2 - Jupyter

localhost:8888/notebooks/Machine%20Learning%20CA2%20.ipynb

Machine Learning CA2 Last Checkpoint: Yesterday at 1:50 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
In [22]: #R squared error
score_1 = metrics.r2_score(Y_train, training_data_prediction)
```


```
#Mean absolute error
score_2 = metrics.mean_absolute_error(Y_train, training_data_prediction)
```

```
print("R squared error: ",score_1)
print("Mean absolute error: ",score_2)
```

```
R squared error: 0.9999948236320982
Mean absolute error: 0.0145848437110976
```

Visualizing the actual prices and predicted prices

```
In [26]: plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Price vs Predicted Price")
plt.show()
```



Conclusion

Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in property research. This study is an exploratory attempt to use three machine learning algorithms in estimating housing prices, and then compare their results.

In this study, our models are trained on housing property data utilising Decision tree(DT) and gradient boosting machine (GBM). We have demonstrated that advanced machine learning algorithms can achieve very accurate prediction of property prices, as evaluated by the performance metrics. Given our dataset used in this paper, our main conclusion is that DT and GBM are able to generate comparably accurate price estimations with lower prediction errors.

We have gone through how to implement the entire machine learning pipeline, and we have an intuitive understanding of machine learning algorithms. The larger the dataset gets, the more complex each of the mentioned steps gets. Therefore, using this as a base will help while you build your knowledge of machine learning pipelines.

To conclude, the application of machine learning in property research is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to property appraisal, and presenting an alternative approach to the valuation of housing prices. Future direction of research may consider incorporating additional property transaction data from a larger geographical location with more features, or analysing other property types beyond housing development.

References

- https://en.wikipedia.org/wiki/Machine_learning
 - <https://www.javatpoint.com/regression-analysis-in-machine-learning>
 - <https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/>
 - https://www.tutorialspoint.com/machine_learning/machine_learning_scikit_learn_algorithm.htm
 - <https://machinelearningmastery.com/xgboost-for-regression/>
 - <https://stackoverflow.com/questions/59166308/how-to-measure-xgboost-regressor-accuracy-using-accuracy-score-or-other-suggest>
- Github Project Link:- <https://github.com/Vasugoe125/Machine-Learning-House-Price-Prediction-Assignment>