Assignment #1

Report on House Price Prediction using Machine Learning

Kavan Shah, Rutvikkumar Patel, Shrey Agrawal, Shweta Limbachiya

Cloud Computing for Big Data, Lambton College

2021S_CBD 2214_5: Big Data Fundamentals – Data Storage Networking

Dr. Debashish Roy

July 15, 2021

Abstract

Machine learning plays a major role from past years in image detection, spam reorganization, normal speech command, product recommendation and medical diagnosis. Present machine learning algorithm helps us in enhancing security alerts, ensuring public safety and improve medical enhancements. Machine learning system also provides better customer service and safer automobile systems. In the present paper, we discuss about the prediction of future housing prices that is generated by machine learning algorithm. For the selection of prediction methods, we compare and explore various prediction methods. We utilize Random Forest as our model because of its adaptable and probabilistic methodology on model selection. Our result exhibit that our approach of the issue needs to be successful and can process predictions that would be comparative with other house cost prediction models. Moreover, on other hand housing value indices, the advancement of a housing cost prediction that tend to the advancement of real estate policies schemes. This study utilizes machine learning algorithms as a research method that develops housing price prediction models. We create a housing cost prediction model, in view of machine learning algorithm models For Example - Linear Regression, XGBoost, Decision Tree and Random Forest, on look at their order precision execution. Those examinations exhibit that Random Forest algorithm, in view of accuracy, reliably outperforms alternate models in the execution of housing cost prediction.

Keywords: House prediction, Linear regression, Random Forest, Decision tree, XG boosts, analysis of models

Dataset Description

- Link to the dataset: https://www.kaggle.com/lespin/house-prices-dataset
- This dataset consists of 2920 records(rows) and 82 features(columns). Here, we are using supervised learning and our targeted column is SalePrice.

Solution Pipeline

Data Preparation:

- Firstly, we are doing data cleaning in which we are dealing with the missing values in dataset. For that, we are finding the missing values in dataset first and then assign the value accordingly. For example, Lot Frontage in dataset represents linear feet of street connected to house. So, missing value in this column means the house is not connected to any street due to which we are filling that with 0.
- Secondly, we are removing outliers from dataset. This means we are removing the values that already present in the dataset for house price which are majorly scattered from the other house price.
- Finally, we are deleting the unnecessary columns/features which means the columns which have not much effect on house price. For our database, these columns are “Order” and “PID”.

Feature Engineering:

- First, we have determined the target value distribution. From this, we have received the where the most data points in the graph will come and what is the highest and lowest possible value for house price.

- Second, we have found correlation between other features and targeted feature so that we got an idea about the features which have how much impact on house price.
- Third, we have deleted one of the two features which are highly correlated with each other to reduce the dimensions for the model.
- Fourth, we have formatted data to learn the model. For example, we are having a column named 'Bsmt Cond' which represents basement condition and has six unique values ('Gd' = Good, 'TA' = Typical, 'No Basement', 'Po' = Poor, 'Fa' = Fair, 'Ex' = Excellent). As these are ranking for the condition, we have assigned values to these ('No Basement' = 0, 'Po' = 1, 'Fa' = 2, 'Ta' = 3, 'Gd' = 4, 'Ex' = 5) so that machine can understand the feature.
- Last but not the least, we have done one-hot encoding for categorial features. This means transforming the features by assigning them binary numbers. For instance, there is one feature in our dataset named 'Paved Drive' that indicates how drive is paved. It is having three possible values: Y means 'Paved', P means 'Partial Pavement', and N means 'Dirt/Gravel'. So, we applied one-hot encoding on this feature, and it will replace the 'Paved Drive' with 'Paved Drive_Y', 'Paved Drive_P', and 'Paved Drive_N'.

Data Modelling:

- Here, we want to predict the price of house and the feature we want to predict is real number and continuous which means the prediction type for this problem is regression.

Performance Measure:

- First, we have performed 80% (Train) – 20% (Test) split on the given dataset.
- To find the suitable algorithm having higher efficiency and lower error, we have checked the model with four different algorithms listed below:

1. Linear Regression
2. XG Boost
3. Decision Tree
4. Random Forest

- We have used GridSearchCV/RandomizedSearchCV functions to find the best parameters for algorithms.
- After finding optimal parameters, we have applied **different algorithms** to learn the model.

1. Linear Regression:

This model is having following syntax:

```
model = LinearRegression().fit(X_train, y_train)
```

To find the error in this model, we have used mean absolute error function as below:

```
y_preds = model.predict(X_test)
mae = mean_absolute_error(y_test,y_preds)
print ("Error in testing using Linear Regression:", mae)
```

By using this model, we are getting **MAE = 22517.43**.

2. XG Boost:

This model is having following syntax:

```
model = XGBRegressor(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    booster='gbtree',
    gamma=0,
```

```
    subsample=1,
    colsample_bytree=1,
    colsample_bylevel=1,
    reg_alpha=0,
    reg_lambda=1
)
model.fit(X_train, y_train)
```

To find the error in this model, we have used mean absolute error function as below:

```
y_preds = model.predict(X_test)
mae = mean_absolute_error(y_test, y_preds)
print ("Error in testing using XG Boost:", mae)
```

By using this model, we are getting **MAE = 19331.86**.

3. **Decision Tree:**

This model is having following syntax:

```
model = DecisionTreeRegressor(
    criterion='mse',
    max_features=85,
    min_samples_leaf=7,
    min_samples_split=18
)
model.fit(X_train, y_train)
```

To find the error in this model, we have used mean absolute error function as below:

```
y_preds = model.predict(X_test)
mae = mean_absolute_error(y_test, y_preds)
print ("Error in testing using Decision Tree: ", mae)
```

By using this model, we are getting **MAE = 24738.96**.

4. **Random Forest:**

This model is having following syntax:

```
model = RandomForestRegressor(
    n_estimators=600,
    criterion='mse',
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features='auto',
    bootstrap=True
)
model.fit(X_train, y_train)
```

To find the error in this model, we have used mean absolute error function as below:

```
y_preds = model.predict(X_test)
mae = mean_absolute_error(y_test, y_preds)
print("Error in testing using Random Forest: ", mae)
```

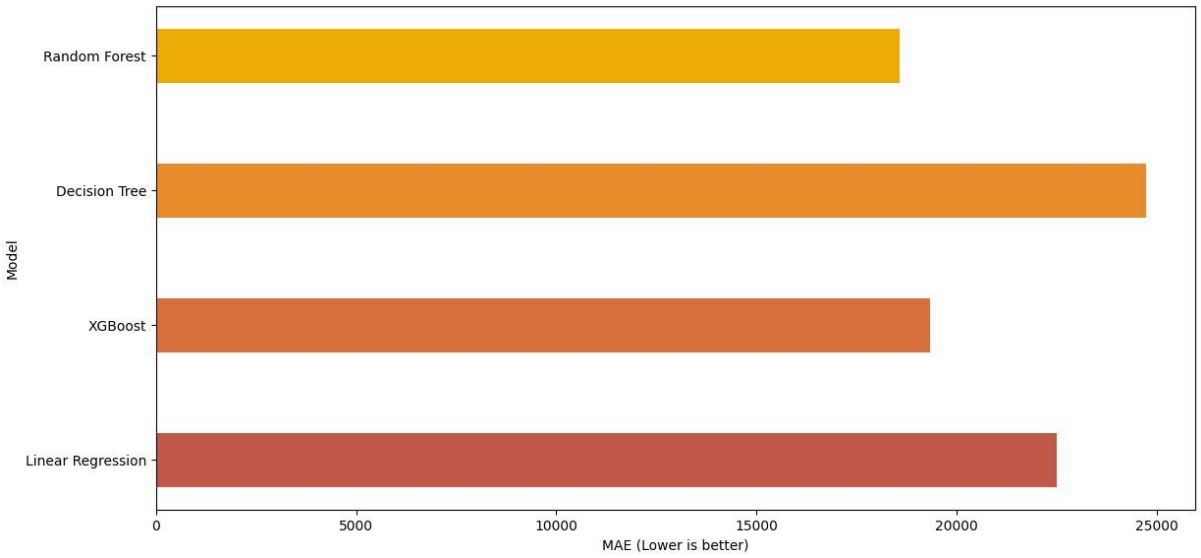
By using this model, we are getting **MAE = 18578.86**. This MAE value keeps changing in every single execution of the model, but it remains lower compared to other models.

Analysis of Models:

We have created different models using different regression algorithms. The table shown below describes the mean absolute error in every model and from that value we can evaluate the performance of the model. Mean absolute error for all models have been shown in table below:

MODEL	MAE
Linear Regression	**22517.43**
XGBoost	**19331.86**
Decision Tree	**24738.96**
Random Forest	**18578.86**

The graph below visualizes the contents of the table:



By examining both graph and table, it is clearly seen that the Random Forest is having the lowest error, 18578.86 followed by XGBoost with the error 19331.86. After that, Linear Regression has error of 22517.43 and Decision Tree is the worst model, and it has error of 24738.96.

In a nutshell, in our calculation, the best model for this problem is Random Forest and the worst model is Decision Tree.

Conclusion

In this report, we built different regression models to predict the house price according to several given features of house. We have followed the data science process starting with data preparation, then feature engineering, followed by data modelling, and lastly the performance measure. We evaluated every model with visualization and later derived the highest performing model by comparing them. This model can be used by people to predict the house price to get an idea about what can be the actual price if the house is having similar features mentioned in the dataset. This model can also be used in different areas or cities if that data contains the same features as dataset.

References

1. Kaggle. "Kaggle datasets".
2. Python Package Index. "Learn about installing packages".
3. W3Schools. "Python Tutorial".
4. Stack Overflow. "Error Correction and Learning".
5. Hegazy, O., Soliman, O.S., & Salam, M. A. (2014). A machine learning model for stock market prediction. https://arxiv.org/abs/1402.7351