House Price Prediction with An Improved Stack Approach

House Price Prediction with An Improved Stack Approach

Hang Zhang[1], Kehan Wang[2*], Mengchu Li[3], Xinchen He[4] and Ruibo Zhang[5]

[1]College of Liberal Arts & Sciences, UIUC, Urbana, IL, 61801, US

[2]College of Art and Science, Boston University, Boston, MA, 02215, US

[3]Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, 47408, US

[4]College of Engineering, The Ohio State University, Columbus, OH, 43210, US

[5]TianJin YingHua international school, Tianjin, 301700, China

*Corresponding author's e-mail: wangkk2021@gmail.com, wangkk@bu.edu

Abstract. As a typical framework in ensemble learning, Stacking creates a multi-layer learner by integrating the learning results of different individual learners. Despite a general improvement in accuracy, multiple layers in stacking results in high computation cost. In addition, some individual learners do not perform better than others in single-model prediction task. To solve the above drawbacks of traditional stacking, this paper proposes an improved Stacking algorithm based on the concept of "weighted attribute". For each individual learner, a corresponding weight will be calculated based on its prediction performance on training data. Weights will then be used as evaluation criteria for model performance. Models with larger weights will be selected as attributes for meta-learner. When improved Stacking algorithm is applied on the same dataset, experiment results indicate a slight improvement in prediction accuracy and a relatively large reduction in computation cost. This demonstrates practical application value of the improvement.

1. Introduction

One widely accepted drawback of single learner is the lack of generality since each model has a limited scope of application. Compared to a single learner, ensemble learning usually achieves higher accuracy and better generality. Bagging and Boosting are two typical techniques in ensemble learning. Both combine multiple individual learners to form a stronger learner with the aim of improving variance. Taking Random Forest (RF) as an example of bagging, the algorithm builds a bunch of decision trees and use majority vote as a standard for prediction of each observation[1]. By contrast, in Boosting, following individual learner is built based on the performance of previous one. For example, if an observation is incorrectly classified in previous leaner, AdaBoost will increase its weights in a certain extent before the establishment of the following leaner[2].

Stacking is another relatively mature ensemble technique. In Stacking, the prediction for training data of a variety of base learners are combined as a new training set of another learner called meta-learner. Stacking combines the prediction of various learners and thus generating better accuracy than any single learner in general. However, in experiment testing, as the number of base learner increases, the computation cost of the whole algorithm increases significantly. In recent years, some researchers developed improvement ideas for sake of reducing the computation cost. For example, Xu developed a three-level Stacking structure and changed the representation of features[3]. Such improvement result in less computation cost in prediction of 15 different datasets.

This paper will propose an improvement of Stacking based on selection of weighted attributes. Firstly, the chosen dataset for algorithm implementation will be introduced. Procedure of data clean will follow the analysis of the raw data set. In the second part, some traditional algorithms will be introduced with each of them followed by corresponding single-learner prediction result. Next, the improvement will be clearly explained. This part will clarify how weight of each base learner is calculated and how it will be used as standard for selection. Testing results on the given dataset will also be showed. Finally, sensitivity analysis will be conducted and suggestions for future work will be discussed.

2. Data Process

The dataset used to evaluate the algorithm is collected by NYC Department of Finance that illustrates the prices of houses were sold. The collection was in a prosperous city and took 12-month, which makes the dataset more reliable and the results more accurate. Both commercial and residential ones are included in the dataset, which makes it suitable for the evaluation and left the dataset with lower bias. [4]

After importing all the required modules and data into working environment, it's necessary to check if there are any anomalies or missing data. It should rectify or drop at this stage. To achieve this, the implementation starts with checking if the dataset have any duplicated entries. Existing duplicates will be deleted. The next step is converting all categorical data into numerical one since some columns still have categorical data which cannot allow us to perform arithmetic (Figure 1). The implementation uses one-hot encoding since it removes the integer encoded variables and add a new binary variable for each unique integer variable[5].

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83190 entries, 0 to 84547
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   BOROUGH                         83190 non-null  int64
 1   NEIGHBORHOOD                    83190 non-null  object
 2   BUILDING CLASS CATEGORY         83190 non-null  object
 3   TAX CLASS AT PRESENT            83190 non-null  object
 4   BLOCK                           83190 non-null  int64
 5   LOT                             83190 non-null  int64
 6   BUILDING CLASS AT PRESENT       83190 non-null  object
 7   ADDRESS                         83190 non-null  object
 8   APARTMENT NUMBER                83190 non-null  object
 9   ZIP CODE                        83190 non-null  int64
 10  RESIDENTIAL UNITS               83190 non-null  int64
 11  COMMERCIAL UNITS                83190 non-null  int64
 12  TOTAL UNITS                     83190 non-null  int64
 13  LAND SQUARE FEET                83190 non-null  object
 14  GROSS SQUARE FEET               83190 non-null  object
 15  YEAR BUILT                      83190 non-null  int64
 16  TAX CLASS AT TIME OF SALE       83190 non-null  int64
 17  BUILDING CLASS AT TIME OF SALE  83190 non-null  object
 18  SALE PRICE                      83190 non-null  object
dtypes: int64(9), object(10)
memory usage: 12.7+ MB
```

Figure 1. Variable List

According to its description, the sale price of 0 indicates that houses are transfers of deeds between parties: for example, parents transferring ownership to their home to a child after moving out for retirement, and there were plenty of null value inputs (e.g: "-") in the dataset, which cannot be used to fit the model: SALE PRICE, GROSS SQUARE FEET and LAND SQUARE are three features with null value. The SALE PRICE is the target variable, and it was not a good idea to have missing data in that column. Hence, the processing strategy will be dropping the rows with a zero value for the SALE PRICE column and fill the missing values in the other columns with the mean values.

3. Data Analysis

3.1. Correlation Heatmap

After data processing, finding out characteristics of dataset is essential. The implementation uses Sklearn heatmap function to analyze the dataset and study different parameters. In Figure 2, the values shown in the correlation heatmap are the score of two associating attributes. The value is between +1 and -1. A negative number represents a negative correlation between two attributes and vice versa. Comparing the

absolute values of the numbers shown in the heatmap can help getting the relationship of all the attributes. TOTAL UNITS and RESIENDITAL UNITS have the highest correlation, and GROSSSQUARE FEET and LAND SQUARE FEET have relatively higher correlation in all the attributes. Basing on common sense, either of these highly correlated features should be neglected to minimize the number of parameters. However, both of these columns are related to sale price in real life, these columns should not be neglected. Section 3.3 will use Principle Component Analysis(PCA) to solve this poential influence.

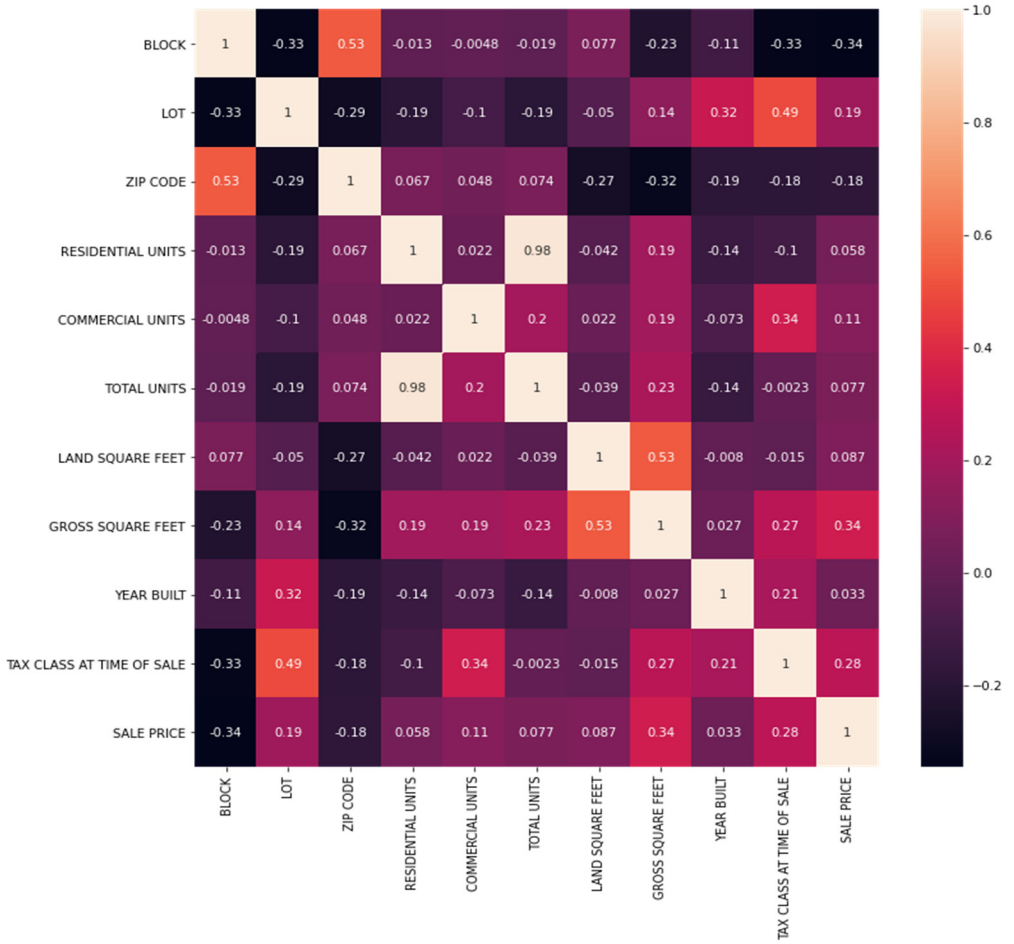

Figure 2. Correlation Heatmap of All Attributes

3.2. Univariate Selection

The implementation also uses univariate selection which measures by KBest from sklearn using score function chi-squared. The calculation will help us find out the most important attributes, and Table 1 presents the result of top 8 suitiable features for prediction. TOTAL UNITS is positioned at eight and RESIDENTIAL UNITS is not one of the top 8. At this stage, it should be reasonable to conclude that TOTAL UNITS and RESIDENTIAL UNITS should be deleted since it cannot make a big influence for final prediction based on KBest feature selection.

Table 1. Top 8 Best Features.

Features	Scores
BLOCK	3.625129e+07
LOT	1.506336e+07
GROSS SQUARE FEET	1.506336e+07

LAND SQUARE FEET	8.030508e+06
ZIP CODE	2.699177e+05
COMMERCIAL UNITS	1.118886e+04
YEAR BUILT	9.437283e+03
TOTAL UNITS	8.160029e+03

3.3. Principle Component Analysis (PCA)

This section introduces how the implementation uses PCA to process the columns of GROSS SQUARE FEET and LAND SQUARE FEET since they have risk to affect the result based on section 3.1 and 3.2. PCA is a method of computing tranformations of the data to get a lower-dimensional set of axes. To achieve that data lies along the directions with highest variation, assume data matrix X multiplies a rotation matrix (called P^T) such that $Y = P^T * X$. The choice of P is significant such that the corvariance matrix of Y is diagonal. Using eigenvector equation $\lambda * p_i = Z * p_i$, $Z = Cov(X)$ to get all eigenvalues, eigenvectors and matrix P. Using $cov(Y) = cov(P^T X)$ to get covariance matrix of Y decides which dimensions should leave.[6] The implementation grabs all the useful information from these two columns and do a dimensionality reduction to be one principal component. It solves the problem of these two columns with high correlations that influences the final prediction

4. Base Learners

4.1. RMSE Introduction

Root Mean Square Error (RMSE) is the standard deviation of points from the linear regression points. It is a popular way to measure how well data is distributed along the line of best fit. The lower the values of the RMSE, the better the model fits with current data. In the experiement, RMSE is used to evaluate the performance of models.

4.2. Regression

Linear regression is a fundamental machine learning model. Using the most highly correlated variables, the linear regression model can make good predictions based on available data. It models the relationship between independent and dependent variables by creating a linear equation. On chosen data set, implementation of linear model resulted in an RMSE of 0.798.

Compared to linear regression, Lasso and Ridge regression are stronger regression techniques. The cost function with respect to them contains two parts. In Ridge, besides the same part as linear regression, a penalty term equivalent to the sum of squared magnitude of coefficients is added to the cost function. For Lasso, the added penalty is simply changed to the sum of magnitude of coefficients. These two algorithms help lower the risk of overfitting.[7] With a few rounds of parameter tuning, the best achieved RMSE is 0.79 for Lasso, and 0.795 for Ridge.

4.3. Random Forest

Random Forest is a method of ensemble learning constructed by multiple decision trees during training. It combines the idea of bagging, random selection and bootstrap. For each tree in the random forest, first, create a new bootstrap from the training dataset; second, use this bootstrap sample to train a decision tree; third, for each node of this decision tree,random selection m features to compute information gain (using Gini Impurity or Entropy) and select the optimal one; finally, repeat all these steps until completing the tree. [5, 6] While fitting Random Forest Rregrssion with default parameters, the mean RMSE is about 0.63, which is a big promotion from Regressions.

4.4. Gradient Boosting

Gradient Boosting Machine is an ensemble learning technique. Like other boosting algorithms, it combines weak learners to form a strong learner, which turns out to yield more accurate results than a

single learner. In each iteration of Gradient Boosting, based on current prediction, a residual with respect to each observation is calculated. A decision tree that classifies these residuals is then built. Before new learner is combined with all previous learners, a numerical value called learning rate, which is defined mainly to reduce the risk of overfitting, is assigned to the new learner. Final prediction thus depends on combination of all built trees.

Gradient Boosting was implemented on chosen data set. When using default setting of parameters, the mean RMSE is about 0.69. Two rounds of parameter tuning reduced the value to 0.61. With such accuracy improvement, Gradient Boosting performed the best among all individual algorithms.

4.5. Stacked Generalization

Stacked generalization allows a new model in learning how to best combine predictions from the multiple existing models. It uses k-fold for training base models, through which predictions are made on the left-out fold. Such so-called out-of-fold predictions are then utilized to train another model, meta-model, where produced information can be used to make final predictions by the base models. [8] For instance, if there are having a 5-fold stacking, then, first of all, the training data need to be split into 5 folds. Then training each base model in 4 folds and make predictions on the remaining fold. After 5 iterations, the entire data will be used to get out-of-folds predictions, which will further apply as a new feature to train meta-model. For the prediction part, the result will just take an average of predictions of all.

In addition to the above mentioned models, Elastic Net and AdaBoosting will also be used for ensembling, and Lasso is chosen as a meta-model because it helps generating the best result —— the RMSE is reduced to 0.60, which outperforms performance than Gradient Boosting.

While, the main limitation of this approach is that each model contributes the same amount for ensembling prediction, regardless of how well a model has performed; The following section will keep discussing a variation of this approach.

5. Improvement on Stacking

5.1. Model Building

5.1.1. Weight of Base Learners.

In first-level learning process of traditional Stacking, cross validation is applied on each base learner. In one round of a K-fold cross validation, besides the prediction for validation data set, the training data will also be used to predict the raw testing data. Therefore, if a 5-fold cross validation is applied on a linear regression model, 5 predictions of the raw testing data based on five different parts of the raw training data will be obtained. The mean value of these 5 predictions will serve as potential testing data of the meta-learner. As each base learner represents a feature of the new data, the number of features will be equal to the number of base learners. In the imporved algorithm, the above process of traditional Stacking is maintained. For the same data set, predictions for different individual learner are usually different. Through certain evaluation criteria like RMSE, how learners performed differently in the same prediction task can be found. Inspired by performance variations, weight is introduced to evaluate the importance of a specific model. Performing 5-fold cross validation generates a training data prediction vector for each base learner. Each vector will then be assigned a weight that represents its importance. All weights form a new vector which will be named “weight vector”. It is defined as follows:

$$\vec{w} = argmin|\sum_{i=1}^{n} w_i\vec{y}_i - \vec{y}| \quad (1)$$

The norm here represents the Euclidean norm. Each v represents a training data prediction vector of a base learner, while y represents the target vector in the raw training data. In other words, the goal is to find the weight vector that minimizes the Euclidean distance between the linear combination of prediction vectors and target vector. To maintain the significance of weight, some extra restrictions are added:

$$\sum_{i=1}^{n} w_i = 1, w_i \geq 0 \quad (2)$$

5.1.2. Elimination of Attributes.
This approach reflects how much each base learner contributes to the prediction of target in training data. After obtaining the weight vector, those learners with relatively higher weights are kept as attributes of meta-learner. Learners that will possibly contribute more in second-level learning can thus be kept and those with less importance can be eliminated. It is reasonable to assume that a reduction in running time results from decline in attributes number. To improve the generality of the algorithm, there are no restrictions on the number of attributes to be retained. The specific number can be determined in testing phase.

5.2. Model Validation
The improved algorithm was tested on the dataset given by Section 2, and the initial group of base learners includes the exact same type of models mentioned in Section 4.5. Table 2 shows the corresponding relationship between learners and weights:

Table 2. Learners with Corresponding Weights

Model	Weight
Linear, Lasso, AdaBoost, Elastic Net	0
Ridge	0.0579
Random Forest	0.4477
GBM	0.4944

A value of 0 might be the result of underflow. Nevertheless, Ridge Regression, Random Forest and Gradient Boosting Machine were considered to be more important than others given the result. Therefore, three base learners with positive weights was kept as attributes of meta-learner at first. In addition, since both Gradient Boosting Machine and Random Forest have much larger weights than Ridge Regression, Ridge Regression was also eliminated to see how results will be affected. Five rounds of testing were conducted in both cases. The results are as follows:

Table 3. Result with Ridge, Random Forest, GBM

Models	Mean RMSE	Mean Running Time(s)
All 7 models included	0.58904	346.962
Ridge, RF, GBM	0.57744	344.105
Random Forest, GBM	0.58891	321.083

Although the improved algorithm may not help promote prediction accuracy compared to traditional Stacking, it generated small enough error while benefitting computation efficiency.

5.3. Sensitivity Analysis
In testing phase, it is found that the initial number of base learners could impact the stability of improved algorithm. When all seven base learners were considerd initially, the algorithm will eliminate four of them, which finally result in a reduction in RMSE of 0.0116 on average. However, if only six of them were included, the algorithm will yield a reduction of 0.00019 on average. The following graph shows how reduction in RMSE correlates with initial number of base learners:

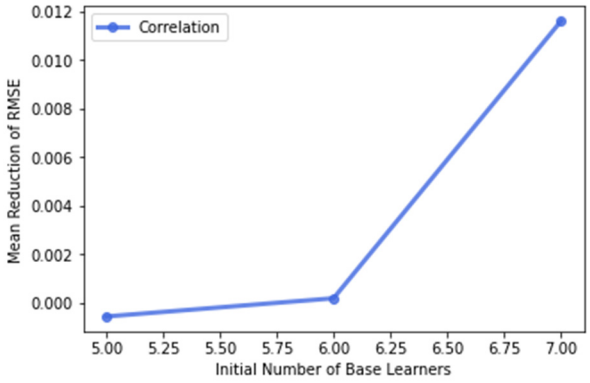

Figure 3. Reduction in RMSE vs Initial Number of Base Learners

As has been shown, having less initial base learners reduces the effect of the algorithm in improving precision. Therefore, it is suggested that large variety of base learners should be considered when using the improved algorithm.

6. Conclusion

In this research paper, comprehensive data processing work on a dataset was firstly conducted. Several machine learning algorithms in making predictions were implemented. When evaluating the prediction accuracy of these models, Gradient Boosting performs the best on the given dataset among all individual learners. Finally, the paper proposes wean improved Stacking approach, which requires less time in computation compared to traditional Stacking while maintaining the same level of precision. As a crucial parameter of the algorithm, the initial number of base learners are found to affect the stability of results. Although the modified algorithm performed well on the given dataset, its generality needs further verification. This improved Stacking is encouraged to be implemented on more datasets. Moreover, as dropping some less important attributes can be an approach, better strategy of using weighted attributes is of great research value.

References

[1] Tin Kam Ho. (1998) The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20, 8 (1998), 832–844.
[2] Yoav Freund and Robert E. Schapire. (1999) A Short Introduction to Boosting. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, 1401–1406.
[3] Huili Xu. (2018) Research and Improvement of Stacking Algorithm. CDMD:2.1018.872852.
[4] New York City Department of Finance. (2017) New York Property Sales https://www.kaggle.com/new-york-city/nyc-property-sales.
[5] Jason Brownlee. (2017) Why One-Hot Encode Data in Machine Learning? Machine Learning Mastery https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/
[6] Stephen Marsland, (2014) Machine Learning: An Algorithmic Perspective. CRC Press. Boca Raton. pp. 275-286.
[7] Robert Tibshirani. (1996) Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological). Volume 8, Issue 1, 267-288S.
[8] David Wolpert. (1992) Stacked Generalization. Neural Networks 5 (12 1992), 241–259.