ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

The Xgboost-Bagging Approach of Predicting House Prices

Xinchao Wu

Shanghai University of Sport, Shanghai, China
wuxinchaowuxinchao@outlook.com

Abstract: *In this paper we aim to establish appropriate models to predict housing prices that are affected by multiple factors and we use various machine learning methods to predict house prices. Boosting methods often aim to reduce the bias of a model but can increase its variance. Conversely, bagging methods tend to decrease the variance of a model. Therefore, we attempted to combine these two algorithms to improve the model's capability for predicting house prices. After comparing different methods, we combined the XGBoost and Bagging to construct a novel model for predicting house prices. The experimental results demonstrate that among all the methods, the proposed approach achieves the best prediction performance for housing prices. Furthermore, when compared to using XGBoost alone, our integrated model exhibits improved precision, indicating its efficacy in addressing the challenges associated with housing price prediction.*

Keywords: *house price prediction, machine learning, ensemble learning, XGBoost, bagging*

1. Introduction

Real estate is a crucial industry, with house price trends serving as a direct indicator of market stability and national economic growth. Predicting housing prices has been a longstanding concern in the real estate and financial markets, as it holds the key to informed decision-making and strategic planning. Current research on house price prediction primarily revolves around statistical and machine learning methods. Statistical methods use time series analysis, regression analysis, and other techniques to offer predictions. Machine learning methods, on the other hand, employ neural networks, decision trees, support vector machines, and other algorithms to achieve prediction goals. With the widespread application of machine learning methods across various domains, researchers are increasingly exploring how these methods can enhance the accuracy and stability of house price prediction. In this paper we aim to establish appropriate models to predict housing prices when the dataset is not sufficiently large. To accomplish the goal, we have explored and applied a diverse range of machine learning algorithms and related techniques. Here are some brief descriptions of our approach:

(1) We conducted data analysis and preprocessing, which encompassed Exploratory Data Analysis (EDA), data cleaning, data transformation, feature selection, and data partitioning.

(2) We applied the XGBoost algorithm to construct a basic model, and then trained and tuned its parameters. Subsequently, we utilized the trained model as the base model and applied the bagging method to obtain a new model. Then we conducted experiments on the test set using this model.

(3) Furthermore, we constructed and trained different models using other machine learning methods, and conducted the same experiments on the same test set, ensuring fairness and consistency in evaluation. We employed the Root Mean Square Error (RMSE) as a metric to objectively compare and evaluate the prediction performance of the various models. To further validate our finding, we conducted experiments on two additional house price datasets to further verify that the new model does indeed improve the accuracy of house price predictions compared to using a single XGBoost model.

Through experiments comparing different models, we observe that the combination of XGBoost and Bagging demonstrates performance improvement.

2. Related work

The utilization of hedonic regression methods and the application of artificial intelligence techniques

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

are two dominant research directions in constructing models for predicting housing prices [1].Currently, the research on house price prediction mainly focuses on machine learning methods. Machine Learning (ML) is a dynamic and interdisciplinary field at the intersection of computer science, statistics, and artificial intelligence. It empowers computers to learn patterns and make decisions or predictions without being explicitly programmed for a given task [2-3]. In the subsequent sections, we will discuss how machine learning methods have been utilized in the past to predict house prices.

In one study [4], the author utilize linear regression, random forests, and boosting methods to fit the given dataset, aiming to enhance the prediction accuracy. Random forests is introduced by Breiman [5], leverage bagging [6-7] to construct diverse tree predictors with a random subset of features. Gradient boosting, first proposed by Friedman, employs a sequential ensemble building approach by approximating a target function through gradient descent [8]. In one article [9], the authors utilized various machine learning techniques such as C4.5, RIPPER, Naive Bayesian, and AdaBoost to build a housing price prediction model. They compare the performance of these algorithms in terms of classification accuracy and identify the most effective method for predicting housing prices in the given dataset. The experiments conducted by the authors demonstrated that the RIPPER algorithm consistently outperformed the other models, offering the highest accuracy in predicting housing prices. In one paper [10],The author first employed a RF(random forest)model to handle redundant data, and then utilized the XGB(XGBoost) model to fit the new dataset, aiming to improve the prediction accuracy. Neural networks are also frequently utilized in building house price prediction models. In a study [11], three different models were constructed using Support Vector Machines (SVM), feedforward artificial neural networks (MLP), and Generalized Regression Neural Networks (GRNN). The Mean Absolute Percentage Error (MAPE) was chosen as the error metric to evaluate their performance. The results demonstrate that the MLP model achieved the best performance. However, another study find that feedforward neural networks are unstable and expensive to train. On the contrary, XGBoost [12], an extension of gradient boosting, has become a widely adopted solution in ML competitions, including Kaggle, is superior to traditional regression models. Past researches have typically focused on comparing various methods, selecting a few excellent ones, and then combining them through ensemble learning to create a predictive model. Numerous studies have demonstrated the good performance of XGBoost in house price prediction. However, one article [13] mentioned that boosting algorithms may suffer from overfitting issues, and one paper [14] have proposed hybrid boosting-bagging algorithms to address this problem. Inspired by this, I attempted to combine XGBoost with bagging to obtain a superior model.

3. Research approach

We embarked on a comprehensive data analysis and preprocessing journey, encompassing Exploratory Data Analysis (EDA), data cleaning, transformation, feature selection, and partitioning. Utilizing the XGBoost algorithm, we constructed a basic model, followed by parameter training and tuning. To further enhance its performance, we employed the trained model as the base and integrated the bagging method to derive a refined model. Conducted experiments on two additional house price datasets. These experiments aimed to confirm if the hybrid model indeed enhances the accuracy of house price predictions beyond using a standalone XGBoost model. Figure 1 illustrates the construction process of the model. We also constructed and trained various models using other machine learning techniques, subjecting them to identical experiments on the same test set. We relied on RMSE (Root Mean Square Error) as the metric to assess and contrast the predictive capabilities of each model.

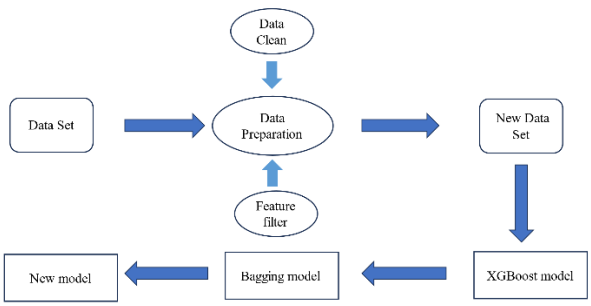

Figure 1: The construction process of the model

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

3.1. Xgboost

XGBoost, an extension of gradient boosting, has become a widely adopted solution in ML competitions, including Kaggle. Its scalability and performance make it a valuable tool for enhancing predictive accuracy in regression problems. To appreciate the significance of XGBoost, it's crucial to first understand Gradient Boosting. Gradient Boost is actually just a framework or design concept that can be applied to different algorithms. Its core idea is using multiple weak classifiers to construct a strong classifier. The general process is to first establish multiple decision trees for integration, and then accumulate the output results (learned optimal parameters) of all trees. Alternatively, each tree (taking GBRT as an example) learns the residual of the sum of all previous tree conclusions, which is the accumulated amount of true values after adding the predicted values. The mathematical expression can be written as:

$$f(\vec{x}) = \sum_{k=1}^{k} h_k(\vec{x};\ \Theta_k) \tag{1}$$

$h_k(\vec{x};\Theta_k)$ is the kth tree(Iak classifier),Θ_k is the parameter of the kth decision tree, and k is the number of decision trees.The loss function of the tree is:

$$L\big(y, f(\vec{x})\big) = \big(r - h_k(\vec{x};\Theta_k)\big) \tag{2}$$

Where

$$\mathrm{r}=y - f_{k-1}(\vec{x}) \tag{3}$$

r is the residual of the current model after it fitting the data.So the problem is transformed into finding the parameter values that minimize the cumulative loss function:

$$\Theta_k = argmin_{\Theta_k} \sum_{i=1}^{n} L\big(y_i, f_k(\vec{x_i})\big) \tag{4}$$

In general, XGBoost and GBDT are consistent in the core gradient boosting decision tree algorithm, but differ in some key implementation details. XGBoost has more advantages in optimization and functional features compared to GBDT.The loss function of GBDT is a first derivative, while XGBoost uses a second derivative. In addition, XGBoost adds a regularization term to the loss function, which helps prevent overfitting.

3.2. Bagging

Bagging, short for Bootstrap Aggregating, is a highly effective ensemble learning technique that aims to enhance the stability and predictive accuracy of machine learning models. This approach operates by generating multiple instances of the same base model, each trained on different subsets of the training data. These subsets are typically created through random sampling with replacement, a process known as bootstrapping. By training multiple models on different data subsets, Bagging aims to decorrelate the predictions made by individual models, reducing the variance and enhancing the overall performance of the ensemble. The main advantage of Bagging is its ability to improve the stability and reliability of machine learning models. By training multiple models on different subsets of the data, Bagging reduces the influence of outliers and noise, making the model more robust to changes in the training data. Additionally, Bagging can also help to mitigate overfitting, as it prevents individual models from overfitting to specific patterns in the training data.

4. Experimental analysis

4.1. Data preprocessing and analysis

The dataset utilized in our study originates from the Kaggle competition titled 'House Prices: Advanced Regression Techniques'. This comprehensive dataset encompasses 79 explanatory variables, encapsulating nearly every facet of residential properties in Ames, Iowa. The training dataset comprises 1460 data points, while the testing dataset has 1459 data points. Each data point is characterized by 79 features, including 43 categorical and 36 numerical attributes. Notably, the final column represents the sale price of the respective property.For instance, the categorical feature 'MSSubClass' signifies the type of dwelling involved in the sale, while 'LotArea' denotes the lot size in square feet as a numerical feature. To gain a deeper understanding of the dataset,we can refer to the 'data_describe.txt' file for detailed

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

information. In our analysis, we segregated the features into two categories: numerical and categorical. For categorical features, we employed one-way analysis of variance (ANOVA) to quantify the influence of each feature. Based on their F-statistics, we prioritized and selected the top 30 most influential features. For numerical features, we calculated the correlation between each attribute and the sale prices, resulting in the selection of the top 27 most correlated features.

By narrowing down the features to these top-performing ones, we aim to enhance the efficiency and accuracy of our regression models, focusing on the most predictive and representative characteristics of the dataset.

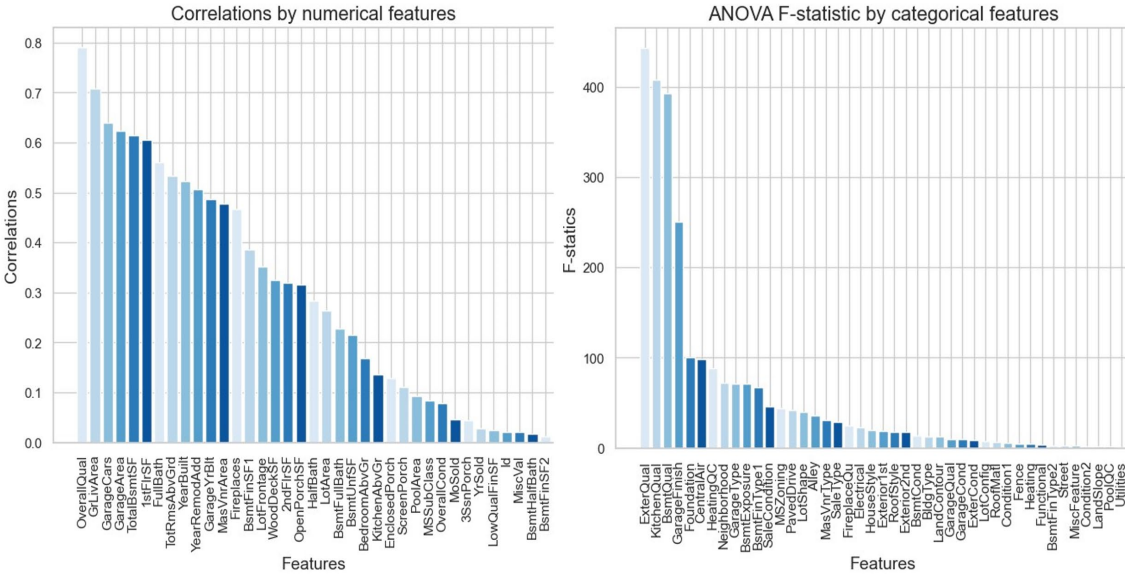

Figure 2: ANOVA F-statistic by categorical features and Correlations by numerical features.

Finally, we merged the selected features to create a comprehensive dataset for analysis. Prior to model training, we undertook necessary preprocessing steps. Categorical data underwent one-hot encoding, while numerical data was standardized. All missing values were replaced with zeros, ensuring a complete and ready-to-use dataset for training. We applied the same preprocessing pipeline to the test set to maintain consistency. For the training set, we utilized the 'train_test_split' function from scikit-learn [15] to divide the dataset into training and validation sets. This function randomly shuffles the data, ensuring randomness in the splitting process. Subsequently, the first 75% of the shuffled data was designated as the training set, while the remaining 25% constituted the validation set.

4.2. Model training

4.2.1. XGBoost and Bagging

In the experiment, we utilized the XGBoost algorithm for regression, importing the XGBRegressor module from the XGBoost library. After fine-tuning the model parameters, including setting the maximum depth to 3 and the number of estimators to 117 for improved performance, we instantiated the XGBRegressor as `xgb_reg`. The parameters of XGB regressor is provided in Table 1.

(1) ‘n_estimators’: The number of boosting rounds or the number of trees to build.

(2) ‘max_depth’: The maximum depth of a tree.

(3) ‘learning_rate’: Also known as the 'eta' parameter, it controls the step size during each boosting iteration.

(4) ‘objective’: Specifies the learning task and the corresponding learning objective. For regression problems, common objectives include 'reg:squarederror' (mean squared error) or 'reg:squaredlogerror' (squared logarithmic error).

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

(5) ‘eval_metric’: The evaluation metric used to monitor the model's performance during training. For regression tasks, common evaluation metrics include 'rmse' (root mean squared error) or 'mae' (mean absolute error).

Table 1: The parameters of XGB regressor

Parameters	Optimal values
‘n_estimators’	117
‘max_depth’	3
‘learning_rate’	0.3
‘objective’	‘reg:squarederror’
‘eval_metric’	‘rmse’

The training process involved fitting the XGBoostRegressor to the preprocessed training data. The model was trained to learn the underlying patterns and relationships in the training dataset. Once the training was complete, we used the trained model to make predictions on both the training and validation datasets. To assess the model's performance, we calculated the root mean squared error (RMSE) for both the training and testing predictions. The RMSE provides a quantitative measure of the model's accuracy, indicating the average deviation of predicted values from the actual values. After that, we employed 155 pre-trained XGBoost models as the base models for Bagging. Each of these base models was trained independently on a randomly sampled subset of the original training data. The random subsets, known as bootstrap samples, are generated by sampling with replacement from the training dataset. This process introduces diversity among the base models, as each one has been exposed to a slightly different perspective of the data. The parameters of Bagging regressor is provided in Table 2.

(1) ‘n_estimators’: The number of base estimators to be trained and combined within the bagging ensemble.

(2) ‘estimator’: The base estimator to be used in the bagging ensemble.

(3) ‘boostrap’: Whether bootstrap samples are used when building each base estimator

(4) ‘max_features’: The number of features to consider when looking for the best split

(5) ‘max_samples’: The number of samples to draw from the training set when building each base estimator.

(6) ‘oob_score’: Whether to use out-of-bag samples to estimate the generalization error.

Table 2: The parameters of Bagging regressor

Parameters	Optimal values
‘n_estimators’	155
‘estimator’	‘xgb_reg’
‘boostrap’	‘true’
‘max_features’	1.0
‘max_samples’	1.0
‘oob_score’	‘false’

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

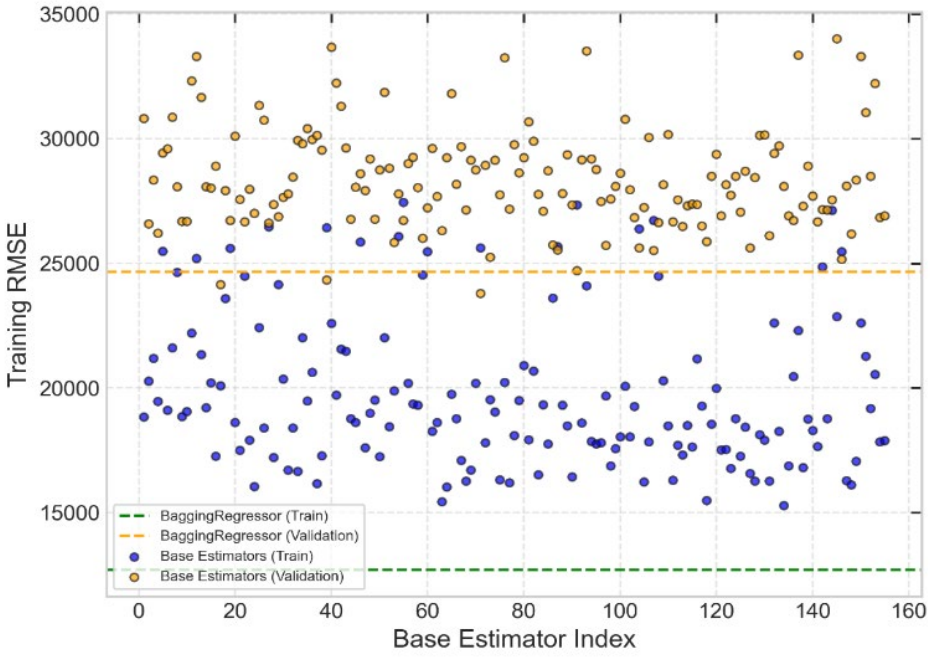

Figure 3: Train and validation RMSE Distribution of Base Estimators

We visualized the training and validation RMSE for each base model (Figure 3), revealing substantial variations in the predictive variance among different base models. Then we applying the Bagging technique, the experimental results demonstrated an improvement. The root mean squared error (RMSE) of the model predictions was reduced by nearly 2000 in comparison to the performance of the standalone XGBoost model. This Bagging approach has proven to be the most effective among all the experimental models conducted, showcasing its capability to mitigate overfitting and enhance the overall robustness of the predictive model.

4.2.2. Other models

We constructed three models for analysis: a Multiple Linear Regression Model utilizing the Linear Regression class from scikit-learn, a Decision Tree Regression model using the DecisionTreeRegressor from sklearn.tree, and a Random Forest Regression model from the same library. Each model underwent training on the designated training dataset and made predictions for both the training and validation/test datasets. To assess their performance, we employed the root mean squared error (RMSE) as the evaluation metric. This allowed us to compare and contrast the performance of each model effectively.

4.3. The perfomance of models on validation set

After training various machine learning models, their performance has evaluated on a validation set to assess their predictive capabilities. The models were assessed based on the root mean squared error (RMSE), which measures the average deviation of predicted values from the actual values in the validation set. The results(Figure 4) indicate that the combination of XGBoost and Bagging, significantly improves the model's predictive performance compared to using individual model.

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

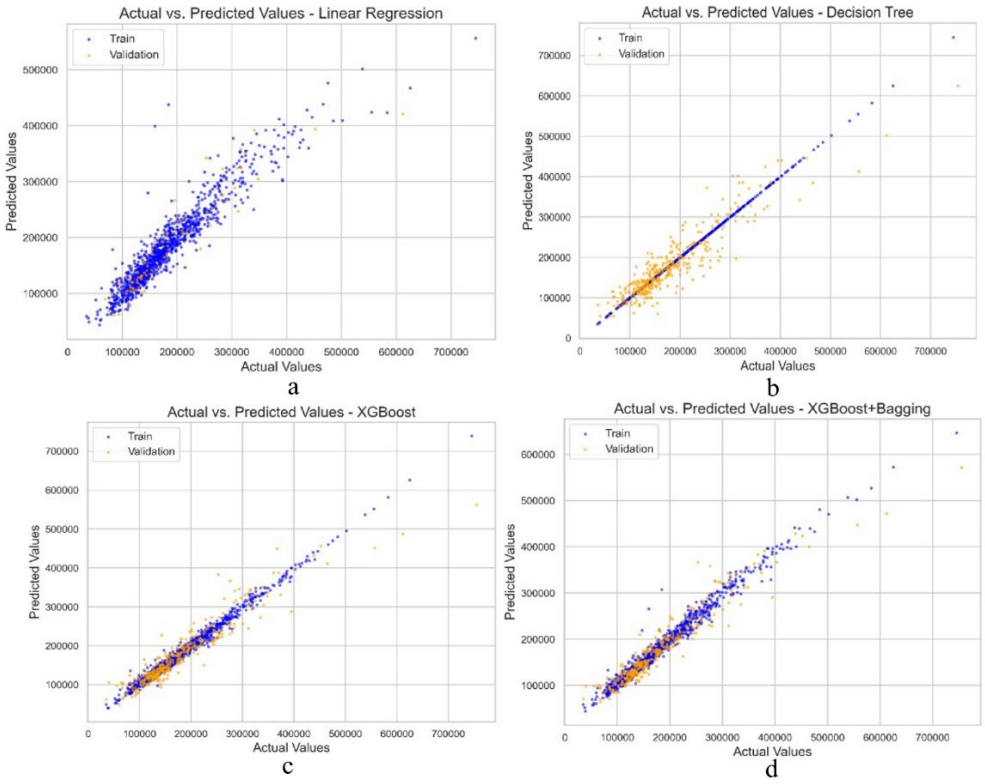

Figure 4: Actual vs. Predicted Values of some models

The following table(Table 3) summarizes the performance of different models:

Table 4: The performance of different model

Model	Train RMSE	validation RMSE
Multiple Linear Model	25863.2	97307.7e10
Decision tree	**0**	33846.9
Random Forest	11903.2	27793.5
XGBoost	8977.3	26607.3
XGBoost+Bagging	12700.3	**24654.0**

4.4. Performance of the XGBoost-Bagging model on other two datasets

After conducting experiments on the model using the aforementioned dataset, I obtained two additional datasets for performance testing. One dataset has fewer dimensions but a larger sample size, while the other is a smaller dataset. The experiments revealed that the new model performs better on the large-sample dataset. However, for the small-sample dataset. The performance improvement of the new model, relative to the XGBoost model, is not very significant.

The California Housing Dataset is a widely used dataset in machine learning and statistics. The dataset is freely available for download on the official website of scikit-learn (sklearn).It contains data on housing districts in California, with various features such as median housing price, median income, average rooms per dwelling, and more. The target variable is typically the median house value for California districts. Researchers and practitioners often use this dataset for regression tasks, aiming to predict housing prices based on the provided features. It serves as a benchmark for evaluating regression models.

The Boston Housing Dataset is another popular dataset from Kaggle. It comprises data related to housing in various neighborhoods in Boston. The dataset includes features such as crime rate, average number of rooms per dwelling, and accessibility to employment centers. The target variable is the median value of owner-occupied homes. Researchers frequently utilize this dataset to develop regression models

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

for predicting housing prices, making it a standard dataset in the field of machine learning and housing market analysis.

The following table(Table 5) summarizes the dataset we used in the experiment:

Table 6: Data set summary

Data set	Samples	Variables	Train size	Test size
Lowa	2919	80	1460	1459
Clifornia	20640	10	18576	2064
Boston	506	14	506	506

The following table(Table 7) exhibits the performance of the XGBoost-Bagging model compare to XGBoost:

Table 8: The performance of the XGBoost-Bagging model compare to XGBoost

Data set	XGBoost-validation RMSE	XGBoost+Bagging-validation RMSE
Lowa	26607.3	**24654.0**
Clifornia	48884.6	**45270.6**
Boston	3.0549	**3.0472**

5. Conclusion and future work

In this study, we aim to establish appropriate model to predict housing prices. We combined the XGBoost and Bagging to construct a novel model for predicting house prices. From the result of the experiment, the XGBoost-Bagging model has a good effect on the prediction of housing prices under various housing prices datasets. For some relatively large datasets, the performance of the model is improved compared to using a single XGBoost model.

Our study has the following limitations which future research could examine further. Firstly, the number of datasets used in our experiments is still not sufficient. According to the experimental results, the dataset itself, including its size and dimensions, has varying degrees of impact on the prediction performance of the new model. We are not yet certain that our model outperforms the original single XGBoost model in predicting all housing price datasets. Therefore, in the future, we will conduct experiments on more datasets. Secondly, compared to using a single XGBoost model, although our model has reduced prediction error, it has also increased the training time of the model. Finally, we have not yet provided a mathematical theoretical explanation for the new model, which is also a future research direction.

References

[1] Park, B., and Bae, J. K. (2015). "Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data." Expert Systems with Applications, 42(12), 2928-2934.
[2] Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc.
[3] Li, H. (2012). "Statistical Learning Methods." Tsinghua University Press.
[4] Varma, A., Sarma, A., Doshi, S., & Nair, R. (2018). "House Price Prediction Using Machine Learning and Neural Networks." In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 1936-1939). IEEE.
[5] Breiman, L. (2001) Random Forests. Machine Learning, 45, 5-35.
[6] Breiman, L. (2001). Using iterated bagging to debias regressions. Machine Learning, 45(3), 261-277.
[7] Breiman, Leo. (1996). "Bagging Predictors." Machine Learning.
[8] Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189-1232.

ISSN 2616-5775 Vol. 7, Issue 4: 1-9, DOI: 10.25236/AJCIS.2024.070401

[9] Ravindra Ravindra, M. P., Meghana, K., Bhavitha, G., et al. (2020). "House price prediction using advanced regression techniques." JEngSci, 11, 1084.
[10] Tao, R. (2022). "Optimized Housing Price Prediction Based on XGBoost." J Sichuan Univ:Nat Sci Ed, 59, 037001.
[11] Erkek, Mehmet, Çayırlı, Kamil, & Hepsen, Ali. (2020). "Predicting House Prices in Turkey by Using Machine Learning Algorithms." Journal of Statistical and Econometric Methods, 9(4), 31-38.
[12] Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM. (DOI: 10.1145/2939672.2939785)
[13] Rogozhnikov, A., & Likhomanenko, T.(2017). Infinite Boost: building infinite ensembles with gradient descent. ArXiv, abs/1706.01109.
[14] Vinayak, R.K., & Gilad-Bachrach, R. (2015). DART: Dropouts meet Multiple Additive Regression Trees. ArXiv, abs/1505.01866.
[15] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. (2011). Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 12, null (2/1/2011), 2825–2830.