# Apply various machine learning methods to predict house price

1. **Abstract**

In this report I have applied various machine learning methods to predict house price. After using different methods, I found that the ensemble methods like random forest and gradient boosting provide better accuracy and stability. In the end, I combined the XGBoost and Bagging methods to construct a new model for predicting house prices and achieved favorable results.

**Keywords:** machine learning, ensemble learning, regression problem

1. **Introduction**

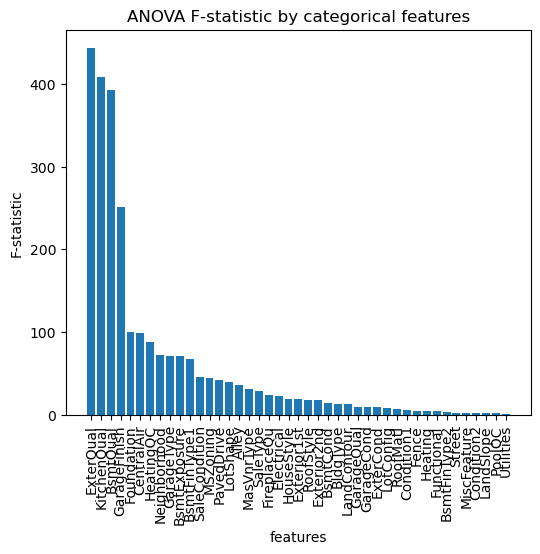
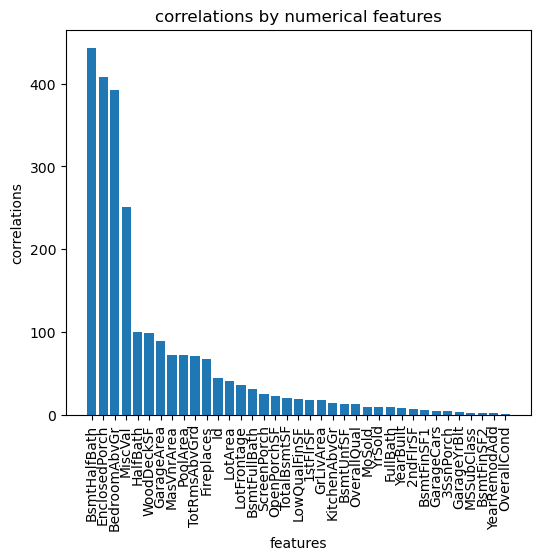
2.1purpose and background

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence with 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition is from Kaggle challenges us to predict the final price of each home base on the train dataset, choose one or multiple methods to construct a model and predict the test dataset.

2.2 Simple dataset description

the dataset I used is from Kaggle competition ‘House Prices -Advanced Regression Techniques’. It includes 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. The train dataset and test dataset have 1460 and 1459 data points respectively. Each data point has 79 features including 43 categorical features and 36 numerical features. Besides the last column is the sale price. For example, the categorical feature name ‘’ means ‘Identifies the type of dwelling involved in the sale.’ and the numerical feature name ‘’ means ‘Lot size in square feet’. You can learn more detail in the file ‘data\_describle.txt’.

1. **Approach**
   1. data preprocessing and analysis

I divided the features in the dataset into two groups based on their attributes: numerical features and categorical features. For categorical features, I used one-way analysis of variance to describle the effect degree of each feature, I sorted the features based on their F-statics and select the top 30 features. For numerical feature, I calculated the correlations between each feature and the sale prices, and select the top 27 features. Finally, I merged these selected features. After that, I performed one-hot encoding on the categorical data, standardized all the data, and finally filled in all missing values with 0, resulting in a training set with completed data preprocessing. Then I used the same operation to process the test set.

**Fig.2.**Correlations by numerical features

**Fig.1.** ANOVA F-statistic by categorical features

* 1. model selection and training

(1) multiple linear regression model

Multiple linear regression is an extension of linear regression model, which is used to describe the linear relationship between two or multiple independent variables and the dependent variable. In multiple linear regression, it is assumed that there is a linear relationship between the dependent variable Y and the independent variables , which can be described by a multiple linear function. The form of this function is usually written as:

(1)

Generally written in vector form:

(2)

In the task, the model is built using the LinearRegression class from the scikit-learn library. The model is trained by calling the fit method on the training data. After training, the model is used to predict the outcomes for both the training and testing data sets. The predicted values are stored. The root mean squared error (RMSE) is used to evaluate the performance of the model. The RMSE of training is small but the testing is very large. Obviously, the model is suffering from overfitting. Then I used the model to predict the prices of the test set and submitted the results to Kaggle. Of course, the score was not very satisfactory.

(2)Decision Tree and Random Forest

Random Forest is an ensemble learning method based on decision trees. It consists of multiple decision trees, each of which trains on a subset of samples and predicts the output. Each tree selects a random subset of samples and features, which helps to avoid overfitting and improves the noise resistance and stability of the model. In the task, the model consists of 100 decision trees, each decision tree applies the CART (Classification and Regression Trees) algorithm. CART is a binary decision tree with the Gini index of diversity as the splitting criterion. For a dataset D, its purity can be measured by Gini value:

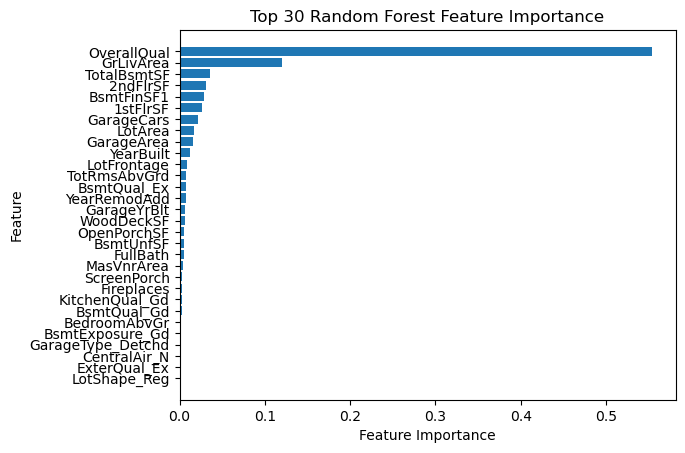
(3)D is the dataset, p(xi) is the probability of occurrence, n is the number of classifications. Gini(D) reflects the probability that two samples randomly drawn from the dataset. For sample D, the number is |D|. Based on whether feature A takes a possible value a, sample D is divided into two parts. We define Gini index as:

(4)

we select the attribute that minimizes the Gini index after partitioning from the candidate attribute set A as the optimal partitioning attribute. Random forests for regression are formed by growing trees depending on a random vector such that the tree predictor h(x, ) takes on numerical values as opposed to class labels.

Experimental result demonstrates if only a decision tree without growth restriction is used as a model, the prediction result is better than a linear model but there is obvious overfitting .But if use random forest ,the result obviously get better.

The model is built using the ‘RandomForestRegression’ class from the scikit-learn library. The model includes 100 decision trees, it uses square error as criterion.it sets the max depth to none so that the tree model can grow freely. The model evaluation is same as the multiple linear regression model above.



**Fig.3.** Top 30 Random Forest Feature importance

(3) XGBoost

Before talking about XGBoost ,we have to first mention Gradient Boost.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:

(5)

is the kth tree(weak classifier) is the parameter of the kth decision tree, and k is the number of decision trees.

The loss function of the tree is:

(6)

Where r=

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:

(7)

In general, XGBoost and GBT 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 GBT.The loss function of GBT 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.

The model training process is as follows: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 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 testing 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.The final step involved printing the RMSE scores for both the training and testing datasets, allowing us to evaluate and compare the model's performance on the two sets of data.

(4) Random forest+AdaBoost

AdaBoost (adaptive boosting) algorithm is a type of boosting algorithm. It starts from weak learners and repeatedly learns a series of weak classifiers, then combines these weak classifiers. In each round, AdaBoost changes the weight or probability distribution of the training data to better learn the samples. The combination of weak classifiers uses a weighted majority voting method to increase the weight of weak classifiers with small classification error rates, so that they play a larger role in voting, and decrease the weight of weak classifiers with large classification error rates, so that they play a smaller role in voting. The final classifier output depends on the weighted voting of all weak classifiers. For a train dataset T with N samples:

(8)

the weight distribution of the training data should be initialized such that the weight of each sample is equal. The initial weight distribution is as follows:

(9)

Based on the D(M),we get the classifier , The actual category of each sample is ,we define error rate as:

(10)

Each classifier has a weight :

(11)

Then we update the weight distribution of the train dataset:

(12)

At last we get a classifier:

(13)

In the experiment, I use Adaboost for regression. The model is built using the AdaboostRegression class from the scikit-learn library. The model use previous random forest model as the base estimator,and the number of the estimator is 100.I define learning rate as 0.01. it uses square error as crit erion. The model training process is similar to the previous models, so I won't elaborate on it here. The model's performance is relatively satisfactory compared to the previous models.

（5）XGBoost+Bagging

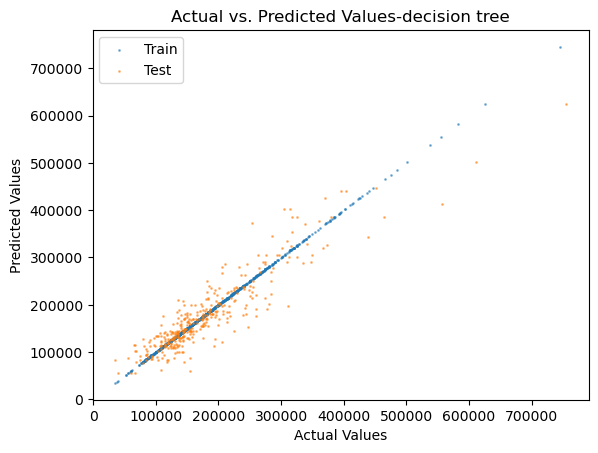
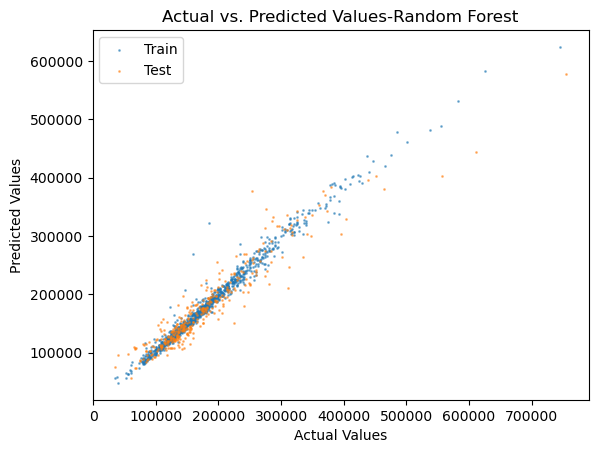
Bagging, or Bootstrap Aggregating, is an ensemble learning technique designed to improve the stability and accuracy of machine learning models. It works by training multiple instances of the same base model on different subsets of the training data, and then combining their predictions to reduce variance and enhance overall performance.I took the pre-trained XGBoost model as the base model and applied the Bagging method.after fine-tune the model, The experimental results show that the RMSE error of the model prediction is reduced by nearly 2000 compared to the former XGBoost model. And this model is also the best performance among all my experimental models.

* 1. predict the house price in test set

Finally, I used the trained model to predict the missing 1459 housing prices in the test set, and submitted the prediction results to Kaggle, obtaining the scores for each model. It should be noted that a lower score indicates a better model.

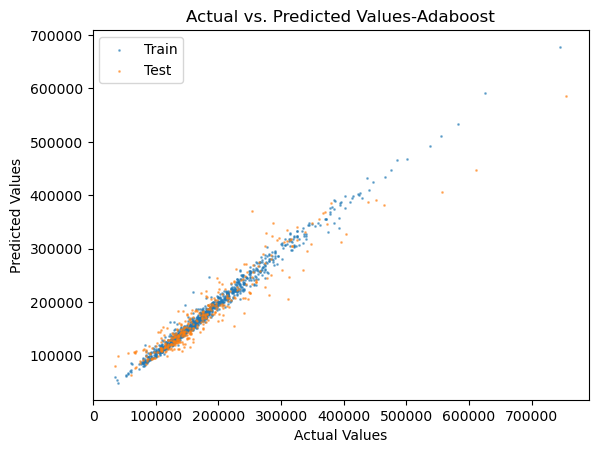
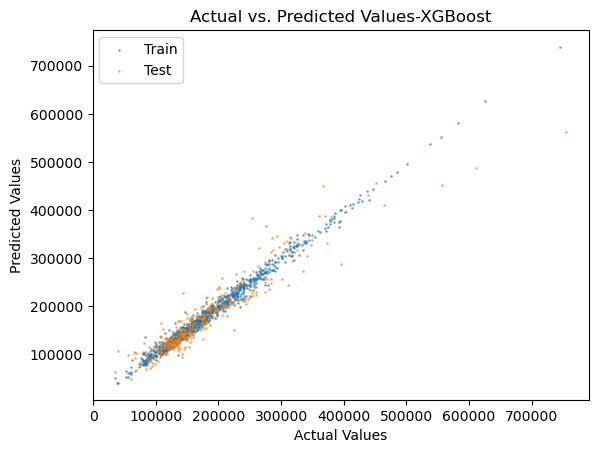
**Table 1** the performance of different model

|  |  |  |
| --- | --- | --- |
| Model | Train RMSE | validation RMSE Kaggle score training time(s) |
| Multiple Linear Model | 25863.2 | 97307.7e10 1.5802 0.0 |
| Decision tree  Random Forest | 0  11903.2 | 33846.9 0.4116 0.1  27793.5 0.1488 4.8 |
| AdaBoost | 6867.4 | 27097.2 0.1432 360.0 |
| XGboost | 8977.3 | 26607.3 0.1356 0.2 |
| XGBoost+Bagging | 12700.3 | 24654.0 0.1320 24.3 |



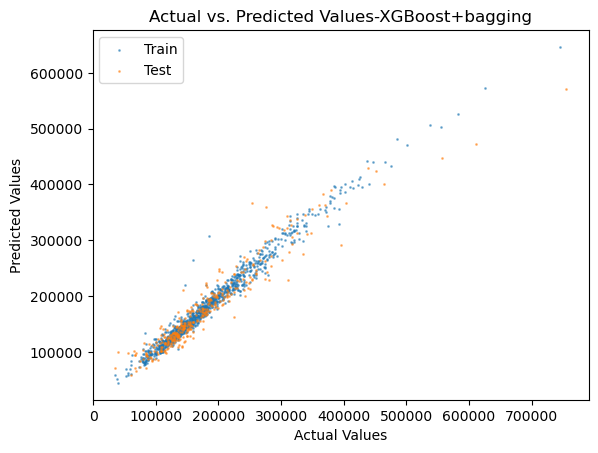
**Fig.5.**Actual vs. Predicted Values-Random Forest

**Fig.4.** Actual vs. Predicted Values-decision tree

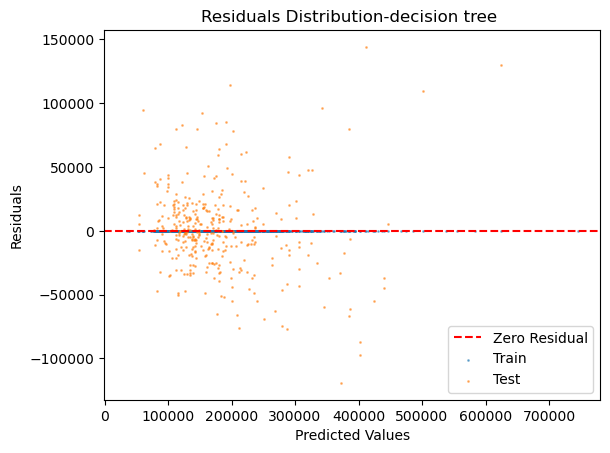
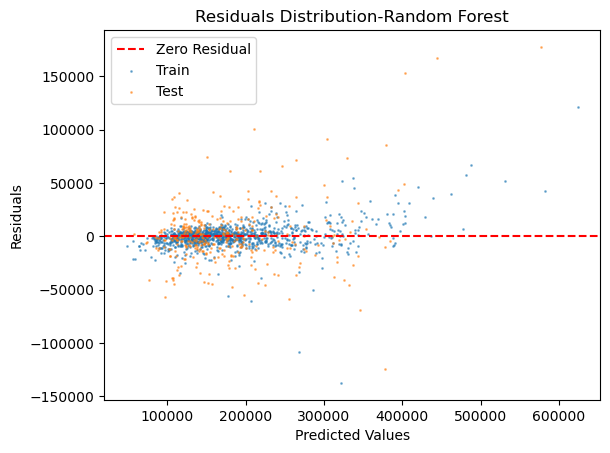


**Fig.7.**Actual vs. Predicted Values-XGBoost

**Fig.6.**Actual vs. Predicted Values-Adaboost

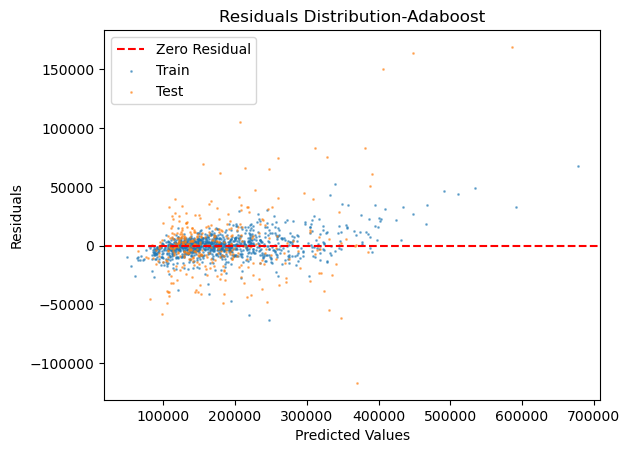
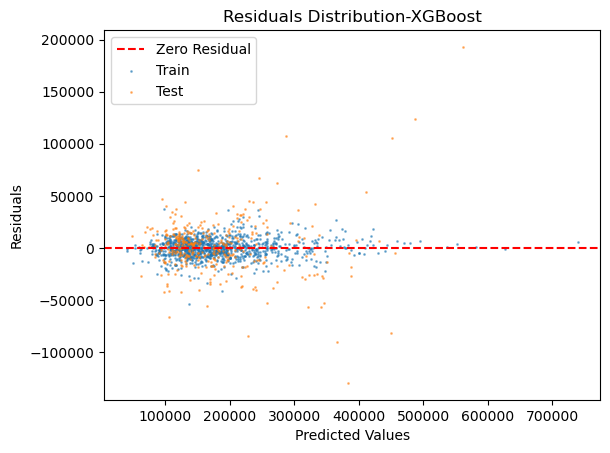


**Fig.8.**Actual vs. Predicted Values-XGBoost+bagging



**Fig.10.**Residuals Distribution-Random Forest

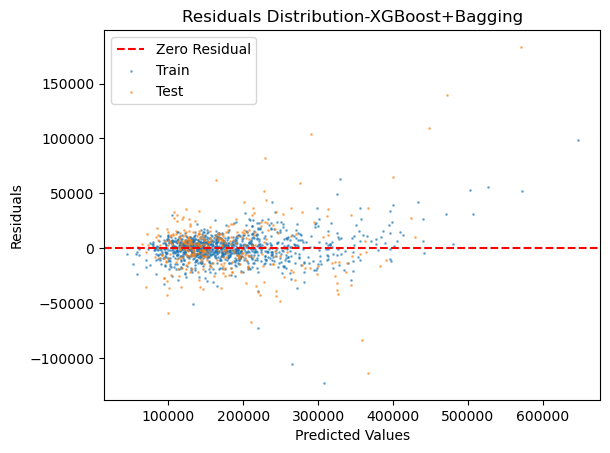
**Fig.9.** Residuals Distribution-decision tree



**Fig.12.**Residuals Distribution-XGBoost

**Fig.11.**Residuals Distribution-Adaboost

**Fig.13.**Residuals Distribution-XGBoost+Bagging



1. **Conclusion**

From the result of the experiment, ensemble learning has a good effect on the prediction of housing prices under this dataset. In the last method, the model combines two common ensemble learning methods, boosting and bagging, using xgboost as the base learner and then using bagging for learning. The performance of the model is improved compared to using a single ensemble learning method.

References：

1. 李航.统计学习方法[M].清华大学出版社,2012.
2. 周志华.机器学习[M].清华大学出版社,2016.

3.Freund, Y., Schapire, R.E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P. (eds) Computational Learning Theory. EuroCOLT 1995. Lecture Notes in Computer Science, vol 904. Springer, Berlin, Heidelberg.

4. Breiman, L. (2001) Random Forests. Machine Learning, 45, 5-35.

5. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, P. J. (1984). Classification and regression trees. CA: Wadsworth International Group. Chen, M. S., Han

6. Chen T , Guestrin C .XGBoost: A Scalable Tree Boosting

System[J].ACM, 2016.DOI:10.1145/2939672.2939785.

7. Breiman, Leo . "Bagging Predictors." Machine Learning (1996).