House Price Prediction System Based on Ensemble Learning

Shenghan Gao, Pengyu Long

October 2, 2024

Abstract

In this project, we propose a novel system for predicting house prices using ensemble learning techniques. Leveraging the dataset from the Kaggle competition "House Prices - Advanced Regression Techniques,[1]" we aim to enhance the accuracy and robustness of price predictions by combining multiple models. Our approach integrates various machine learning algorithms, including linear regression, random forest, and gradient boosting, to capture different aspects of the data. By optimizing the ensemble model, we achieve superior performance compared to individual models. This report details the data preprocessing steps, model selection, and the ensemble strategy employed. The results demonstrate that our ensemble learning approach significantly improves predictive accuracy comparing with single model.

1 Introduction

Accurate house price prediction is a critical task with significant implications in various domains such as real estate, urban planning, and financial services. The Kaggle competition "House Prices - Advanced Regression Techniques[1]" provides a rich dataset that challenges participants to predict the final price of residential homes. The dataset includes various features ranging from property characteristics to neighborhood attributes, making it an excellent candidate for advanced predictive modeling.

In this project, we introduce a novel ensemble learning system designed to enhance the accuracy and robustness of house price predictions. Ensemble learning, which combines multiple models to produce a single superior prediction, has proven effective in numerous machine learning applications. Our approach leverages this technique by integrating several base models to capture different aspects of the dataset and improve predictive performance.

2 Problem Analysis and Data Preprocessing

Our goal is to predict the sales price for each house given 79 explanatory variables describing (almost) every aspect of residential homes. Predicting house prices is a complex regression problem that involves understanding various features of residential properties and their impact on the final sale price. The key challenges include dealing with diverse data types (categorical and numerical), handling missing values, and capturing non-linear relationships among features.

Given the complexity and high dimension of the dataset, a robust approach is required to preprocess the data effectively and to select suitable models that can capture the intricate relationships among the features.

2.1 Missing Value Handling

We observed missing value in several features. For numerical features, missing values were imputed using the minimum value of the respective feature to minimize the impact of outliers. Missing values in categorical features were imputed using the most frequent value.

2.2 Feature Encoding

Categorical variables like MSSubClass were transformed into binary vectors using one-hot encoding, which helps models interpret categorical data without assuming any ordinal relationship. For ordinal categorical

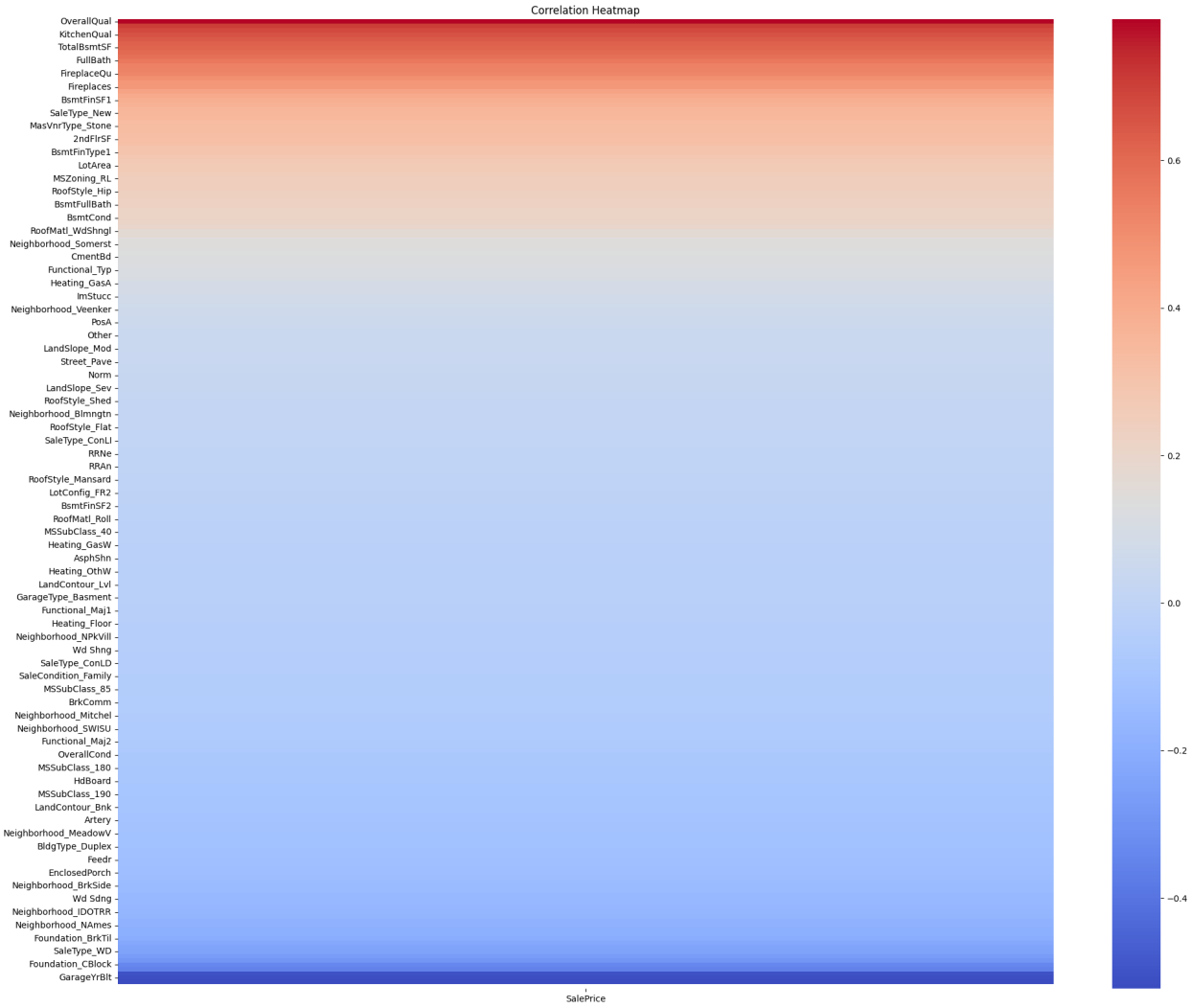

Figure 1: Correlation heatmap. We removed features with a correlation coefficient lower than 0.1 to retain only those features that have a meaningful relationship with the house prices.

variables like LotShape, Likert scaling was applied to retain the ordinal relationship among categories. After encoding, Z-score normalization is applied to ensure they have a mean of 0 and a standard deviation of 1. This step is crucial for models sensitive to feature scaling, such as Ridge and Lasso regression. After this step, we transform a 79-dimension raw data into 223-dimension features.

2.3 Feature Engineering

We calculated the covariance of each feature with the target variable, the sale price. This helped us identify the features that have the strongest linear relationship with the house prices, which can be crucial in predicting the final sale price.

2.4 Dimensionality Reduction

To reduce the dimensionality of the feature space and to mitigate multicollinearity, Principal Component Analysis(PCA) with $PoV > 0.9$ was applied. This technique helps in summarizing the information content of the original features into a smaller set of uncorrelated components, thereby enhancing the computational efficiency of the models. Afte this step, feature dimension of our data is reduced to 123 dimensions.

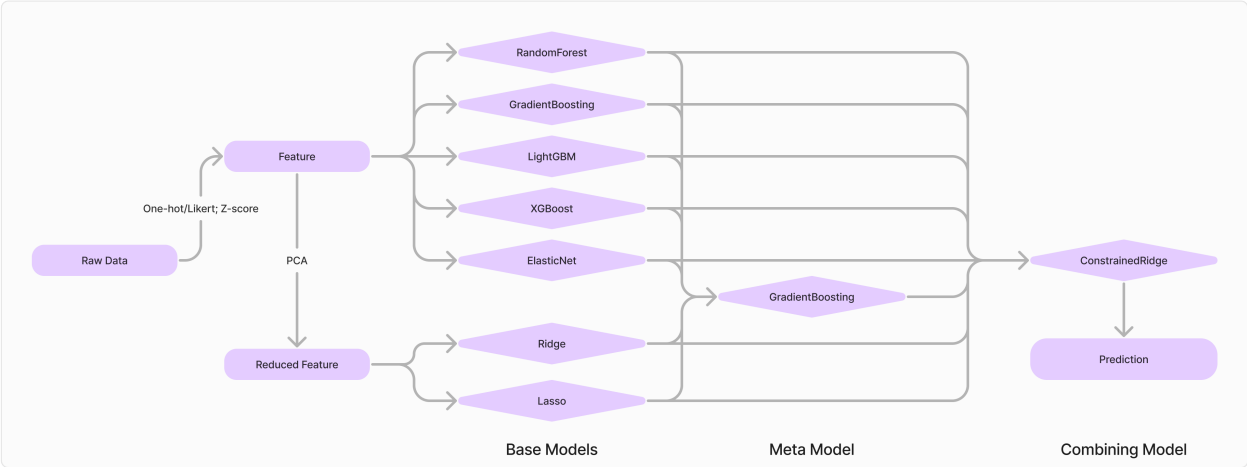

Figure 2: Pipeline

	Score(Red)↓	Score(All)↓
RandomForest	0.17884	0.14378
GradientBoost	0.17027	0.12678
XGBoost	0.14198	0.12751
LightGBM	0.15875	0.12662
Ridge	0.20840	0.44823
Lasso	0.20886	0.46003
ElasticNet	0.20886	0.15347

Table 1: Experiment for base models.

3 Method

Figure 2 provides an overview of our system. The system leverages the concepts of stacking and voting. Initially, base models are employed to make fundamental predictions. Inspired by Nanashi's work[4] the stacking method [2], these predictions are treated as meta-features and are fed into a meta-model. To further enhance our model, we integrate all features into a modified ridge model, obtaining the final prediction as the ultimate result.

3.1 Base Models

We first try various model like lasso regression, RandomForest etc. To ensure the efficiency of dimensionality reduction, we validate all models on both non-reduced and reduced features. The detail is shown in Table 1. As shown in table, we find some models, like ElasticNet, are more suitable for high-dimension data. Hence, based on our experiment result, we decide to apply reduced features on Ridge and Lasso regression, while other models, like RandomForest, show better performance with non-reduced features.

3.2 Stacking

To further strengthen our model, we utilize stacking, a popular method in ensemble learning. The main idea is to combine several different base models to generate a more powerful meta-model. We choose Gradient Boosting as the meta-model because it shows great potential in handling non-linear relationships and strong robustness, thus avoiding the overfitting problem. As shown in Table 2, compared with MLP and other ensemble methods, stacking produces better results.

	Score
MLP	0.18008
Voting	0.13389
LinearReg	0.13116
Stacking	0.12655

Table 2: Experiment for base models. In this experiments, we test model performance on both

3.3 Combining

To further explore the potential of our models, we attempted to combine the results of all previous models. However, determining the optimal hyperparameters for this combination proved to be challenging. To address this, we implemented a constrained ridge regression.

$$\begin{aligned} \text{Minimize} \quad & \text{Score} \\ \text{Minimize during training} \quad & \text{Loss} + \alpha \sum w^2 \\ \text{Subject to} \quad & \sum w = 1 \\ & w > 0 \end{aligned}$$

where, α is a hyperparameter to balance error and regularization. The corresponding weights and results are shown in Table 4.

4 Evaluation

4.1 Experimental Setting

We participated in the competition on Kaggle[1], obtaining the training dataset with labels and the test dataset without labels. We trained our entire pipeline using the root mean square error (RMSE) metric:

$$L = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

During the training process, we trained models individually and utilized K-fold cross-validation to obtain the best hyperparameters ($K = 10$ in our training).

Subsequently, we applied the test dataset to generate predictions, which were then submitted to the Kaggle competition platform. The scores obtained from these submissions indicated the performance of the corresponding models, with lower scores representing better performance.

All code was executed on an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz, equipped with two NVIDIA GeForce RTX 3090 GPUs. The software environment was based on Python 3.11.

4.2 Result Analysis

The experimental results of our model are summarized in Table 4. The table presents the weights assigned to each model and their corresponding scores.

The results indicate that the Stacking model received the highest weight (0.2999), contributing significantly to the combined model. GradientBoost and XGBoost also had substantial weights of 0.2199 and 0.2170, respectively. LightGBM had a moderate weight of 0.1110.

On the other hand, Ridge, Lasso, and ElasticNet received relatively low weights of 0.0352, 0.0355, and 0.0486, respectively. This suggests that their individual contributions to the combined model were less significant.

The final combined model achieved a score of 0.12176(Rank 227/4650, 5.5%, 6/17/2024), which is an improvement over the individual models, demonstrating the effectiveness of our approach in leveraging the strengths of different models to enhance overall performance.

5 Discussion and Limitation

5.1 Discussions

5.1.1 Pros and Cons of Dimensionality Reduction

Dimensionality reduction presents both advantages and disadvantages in our house price prediction model.

Dimensionality reduction reduced risk of overfitting. With fewer features, models are less likely to overfit the training data, leading to better generalization on unseen data, especially for simple models like Lasso regression.

However, dimensionality reduction can also lead to potential loss of information. Important information might be lost during the transformation, especially if too many dimensions are reduced, which can reduce performance of complex models since they can handle the complexity of high-dimensional data. Also, the new features generated by PCA are linear combinations of the original features, making them harder to interpret.

As shown in Table 3, for Lasso Regression, dimensionality reduction improved model performance. Although training Lasso Regression on all features gives a lower RMSE, the performance on final score is reduced, which can be considered as an overfitting. In contrast, for Random Forest, dimensionality reduction reduce the model performance.

5.1.2 Ensemble learning vs. Single Model

Ensembles often perform better than single models by combining the strengths of multiple models. Also, by aggregating predictions from several models, ensemble methods are less sensitive to the weaknesses of individual models.

However, ensemble models are more complex to implement and interpret compared to single models. Also, ensemble models usually require tuning more hyperparameters, which can be computationally expensive.

5.1.3 Similarity with Residual Connections[3]

In our ensemble learning system, the outputs of the base models are passed through a meta-model (Gradient Boosting) and then combined with the base model outputs before being fed into the final combining model. This process can be likened to residual connections in deep learning. Residual connections help correct errors by learning the residuals (differences) between the predicted and actual values. In our ensemble, Gradient Boosting learns from the errors of the base models, improving the final prediction accuracy. Also, just as residual connections allow information to bypass certain layers, combining base model outputs with the meta-model output ensures that valuable information is retained and directly influences the final prediction.

5.2 Limitations

hyperparameter Tuning: Our approach involves multiple models, each with numerous hyperparameters, making the fine-tuning process highly time-consuming. Due to the submission limits of Kaggle, we were unable to exhaustively search for the optimal hyperparameters. Consequently, our model still has untapped potential for further improvement.

Time Complexity: The overall time complexity of our method is the cumulative time complexity of all individual models. Although the training dataset comprises only 1.6k records, the computational demands become significant when scaling to larger datasets. Despite all of our models being machine learning-based, managing the training process on a large-scale dataset remains a challenge.

	Reduced Features		All Features	
	RMSE↓	Score↓	RMSE↓	Score↓
Lasso	26977.03	0.20886	23592.56	0.46003
RandomForest	11380.59	0.15519	10347.46	0.14378

Table 3: Comparison for dimensionality reduction

	Weight	Score↓
RandomForest	0.0327	0.14378
GradientBoost	0.2199	0.12678
XGBoost	0.2170	0.12751
LightGBM	0.1110	0.12662
Ridge	0.0352	0.20840
Lasso	0.0355	0.20886
ElasticNet	0.0486	0.15347
Stacking	0.2999	0.12655
Combining		0.12176

Table 4: combiningModel

6 Conclusion

In this project, we developed an ensemble learning system to predict house prices, leveraging the rich dataset provided by the Kaggle competition "House Prices - Advanced Regression Techniques."[1] Our approach involved a multi-step process including meticulous data pre-processing, dimensionality reduction, and a sophisticated ensemble modeling technique.

By integrating various machine learning models through a layered ensemble approach, we effectively harnessed the strengths of individual base models. The use of dimensionality reduction via PCA helped streamline the feature set, improving computational efficiency and reducing the risk of overfitting. Our system's architecture, inspired by principles similar to residual connections[3], allowed us to combine the raw outputs of base models with the meta-model (Gradient Boosting) outputs, capturing a comprehensive range of data patterns and enhancing overall predictive performance.

The comparison between ensemble and single models highlighted the superior accuracy and robustness of ensemble methods, albeit with increased complexity and computational requirements. This complexity, particularly in hyperparameter tuning, remains a significant challenge and an area for potential optimization.

While the high-dimensional nature of the dataset posed challenges, our ensemble learning system demonstrated significant improvements in predictive accuracy, making it a valuable tool for real-world applications in house price prediction. Future work could focus on further optimizing hyperparameter tuning and exploring additional dimensionality reduction techniques to balance model complexity and performance.

In summary, our ensemble learning approach successfully tackled the complexities of house price prediction, providing a robust and accurate model that leverages the combined power of multiple machine learning techniques.

References

[1] House prices - advanced regression techniques. `https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview`.

[2] [model fusion for machine learning] stacking stacking method. `https://blog.csdn.net/weixin_46803857/article/details/128700297`.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] NANASHI. House prices solution [top 1%]. https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/discussion/83751.