House Price Prediction with Regression

CS 472 Machine Learning Spring 2024 Final Project Report

Karma Woeser

1. Introduction

Housing price predictions are crucial for the housing market. It's used in determining the right time to purchase a house. Being able to predict house prices accurately can be a challenge because of many different implications. Every small factor of the house plays a part in the price. This project is sourced from a Kaggle competition, Kaggle House Prices - Advanced Regression Techniques. The goal of this project is to use predictive models that can estimate the sale prices of houses. We want to estimate the continuous value of house prices, therefore we will use regression techniques. Regression is used because the target variable is a continuous numerical value that is predicted using features. This project will use feature engineering, random forest regression (RF), gradient boosting (GB) and cross-validation algorithms.

2. Data and Methods

The dataset is from https://www.kaggle.com/house-prices-advanced-regression-techniques/data. It focuses on residential houses in Ames, Iowa. Where each feature describes a different aspect of the house. The training dataset has 1460 house ID's and 79 features. Most importantly having the target variable (SalePrice). The features are stored as numerical and categorical values. Features like the year built, neighborhood, lot size, floor count, full bath, and half bath are all stored.

The dataset has some issues with missing features. Features with the value of NA are replaced with the average value of their respective column. For categorical features they are replaced with the value that appears most often. The dataset is split by an 80/20 training and validation set. All of this ensures that the algorithms can handle when there is data missing.

The baseline model used is a linear regression model. This project used two models: The Random Forest and Gradient Boosting regression models. The performance of these models are measured by their Root Mean Square Error (RMSE). This is a common performance indicator for regression models. RMSE measures the average difference between predicted values and the actual values by a model. It provides an accuracy to how well the model is able to predict its target variabel (SalePrice). The lower the RMSE value the better the model is.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

Random Forest (RF) use's multiple decision trees to improve predictive accuracy and control overfitting. Each tree makes its own prediction but the final prediction is based on the average or weighted averages of all the tree's predictions. RF trees use the best split strategy. Gradient boosting (GB) is an accuracy boosting technique that builds models sequentially and corrects errors made by previous models. I believe the GB model will perform better than RF since it builds off previous trees while RF is independent. Although, GB can lead to overfitting so precautions are needed.

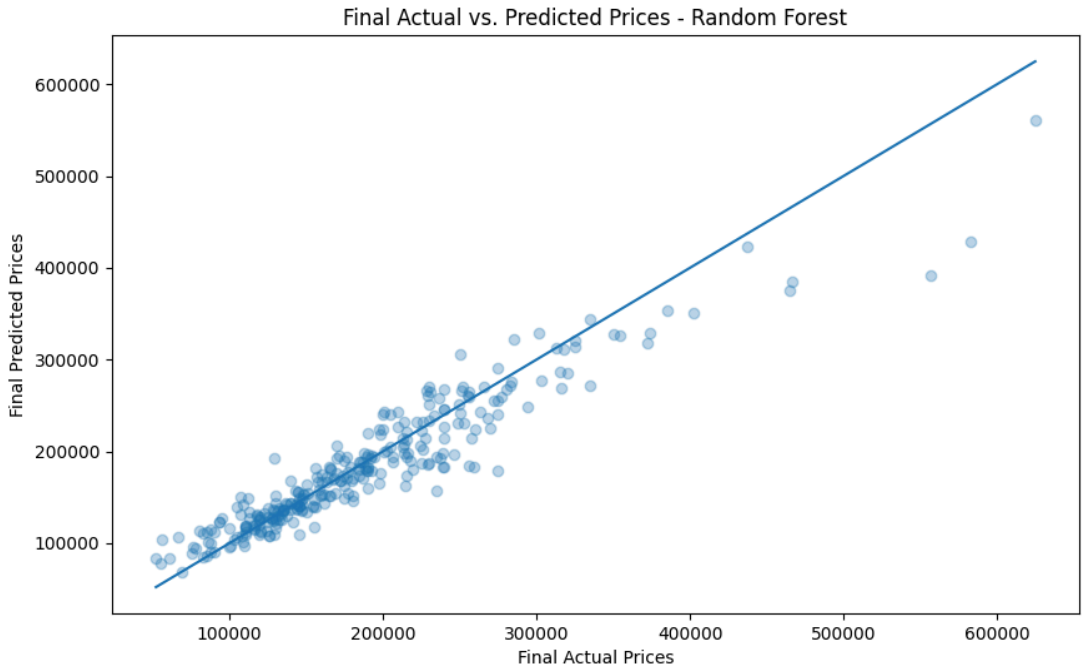
3. Results and Discussion

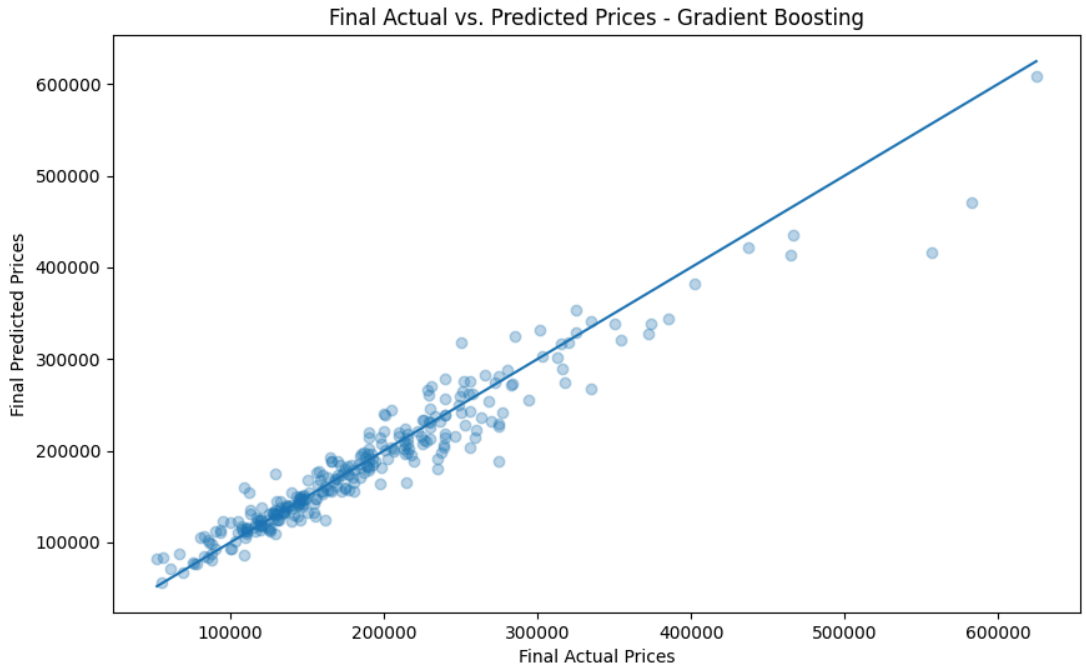
The initial performance of the models were decent. The RMSE wasn't too high but there was definite room for improvement. I focused on improving the performance of the current models, instead of trying other models. I wanted to improve the models using hyperparameter tuning. I used a scikit-learn tool called GridSearchCV. It's a tool that searches for the best parameter values from a set of parameters. It uses the k-fold cross-validation method. The purpose of this was to see how the model would perform when new data was given. The models were set k to 5. GridSearchCV does increase the model's run time but it's accurate at choosing good parameters based on RMSE scores. Below is the RMSE of before and after hyperparameter tuning.

Model	Initial RMSE	RMSE after hyperparameter tuning	Hyperparameters used
Random Forest	27592.57227	27931.863244565367	max_depth: 10, n_estimators: 200
Gradient Boosting	24001.66996	22355.204339593838	max_depth: 3, n_estimators: 300

From the RMSE results we can see that the Random forest model doesn't actually improve after tuning. It actually becomes slightly worse. For RF there were multiple times where the RMSE became worse after tuning. Gradient Boosting on the other hand had a much better RMSE value both before and after tuning. Tests that were run using higher max_depth's always lead to higher RMSE values. A lower max_depth count leads to a better RMSE value. Which might indicate that complex trees overfit the data.

Below are graphs that show the relationship between the actual sale prices from the dataset and the predicted sale prices. These are graphs after tuning has been done. The blue points are the predicted sale prices. The blue line is the optimal price, so the closer the blue points are to the blue line the better the accuracy. Both models shared similar results. The higher the sale price becomes the less accurate the model becomes. The outliers are very similar in both graphs. On average Gradient Boosting had a better RMSE value and led to more accurate house price predictions.





Both models are able to predict the sale price of a house fairly well. I wanted to improve the random forest model by adding and trying different hyperparameters. Also wanted to improve the Gradient desiccant model by adding learning rate parameters. Most of the tests I ran didn't help or I wasn't able to fully analyze. Any tests that did improve the accuracy didn't have enough of an improvement for me to consider it as better hyperparameters. GB tends to outperform models like RF because of the fact that RF builds each tree independently. While GB builds trees based on others and corrects errors of previous trees. RF gets the average of the trees. GB optimizes its loss function using gradient descent.

Comparing my models with others from the same kaggle competition showed me that my results were fairly similar to others. The top pinned kaggle competition notebook used TFDF to predict house prices. Comparing both of our first 5 ID sale prices you can see that it's similar. The right example is my SalePrice predictions.

	Id	SalePrice
0	1461	123554.718750
1	1462	153939.062500
2	1463	176793.765625
3	1464	183828.296875
4	1465	193644.484375

```
Id,SalePrice
1461,124786.6648583476
1462,163304.85397059977
1463,187134.14402991463
1464,188555.37282906738
1465,181248.18980221328
```

4. Conclusion

This project's aim was to accurately predict house prices using machine learning techniques. Mainly Random Forest and Gradient Boosting regression models. I used data processing, cleaning and hyperparameter tuning to improve the prediction accuracy of the models. The accuracy of the models were determined by its Root Mean Squared Error. The RMSE was tested before and after tuning. Tuning was optimized by a scikit-learn tool called GridSearchCV. The RMSE for RF before tuning was 27592.57227. The RMSE for RF after was 27931.86324. The RMSE for GB before tuning was 24001.66996. The RMSE for GB after was 22355.20403. The graphs of both models showed us that they both run into issues when dealing with higher sale prices. From our results we can conclude that the Gradient Boosting model outperforms the Random Forest model before and after tuning. Although the results of this project weren't perfect or complex. We were able to compare two different machine learning models and find which one had a better accuracy. I plan to improve on this project in the future by exploring other machine learning techniques that could have even better prediction accuracy.

5. References

[1] “House Prices - Advanced Regression Techniques.” *Kaggle*, www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview. Accessed 12 June 2024.

[2] Saini, Anshul. “Gradient Boosting Algorithm: A Complete Guide for Beginners.” *Analytics Vidhya*, 25 May 2024, www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/.

[3] “User Guide.” *Scikit*, scikit-learn.org/stable/user_guide.html. Accessed 12 June 2024.