Predicting House Prices using Advanced Regression Techniques

1st Joshua Bae
Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA
joshbae@rice.edu

***Abstract*—For the final project in ELEC 478, a Kaggle data science problem was completed to dig deeper and explore an aspect of machine learning that was previously unfamiliar. This project had the goal of satisfying the criteria of scoring within the top 10% of the Kaggle leaderboard. The project involved predicting house sale prices in Ames, Iowa, using the intricacies of real-estate data. While the goal was simple, given the complexity of the data (78 explanatory variables), a creative combination of domain knowledge, feature engineering, and advanced regression techniques (model stacking) was necessary to identify the most important features of the dataset and satisfy the overall objective. Unfortunately, the desired leaderboard ranking was not achieved, reaching a ranking within the top 20%, which seems reputable given that the competition is already a few years old and the enormous amount of computational resources required for marginal increases.**

I. Introduction

Purchasing a home remains one of the biggest purchasing decision that individuals make in their lifetime. This project aimed to predict house sale prices of houses in Ames, Iowa, by using various aspects of residential homes. In doing so, insight is gathered on what features can be used by individuals to complement their decision-making process when purchasing/selling a house. On one side, these key discoveries can help maximize the potential value that homebuyers gain under a budget; on the other hand, property agents can improve the probability of a sale through proper marketing of key variables.

The Ames Housing dataset was compiled by Dean De Cock for use in data science education though Kaggle [1]. It is often cited as a good alternative for data scientists looking for a modernized and expanded version of the more popular Boston Housing dataset. 79 variables describe (almost) every aspect of residential homes in Ames, Iowa, and consists of a mix of numerical and categorical features.

The graded submissions on Kaggle are evaluated based on the Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. (Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.)

II. Data Cleaning

A. Dataset

The training dataset has a total of 1460 observations with 81 variables, whereas the test dataset contains the same number of observations but one less variable with the exclusion of the target variable, SalePrice.

While not technically counting as outliers, per se, there were a few features in the training dataset whose values failed to be represented in the testing dataset. For example, in the HouseStyle category, which indicates the style of dwelling, the testing data does not have any houses catgorized as 2.5Fin, which alludes to a two and one-half story house with the 2nd level finished. As can be seen in the figure below, the 2.5Fin bar is colored only by the blue training data. This situation was one that could not be amended due to the sparsity of some of features of the overall dataset. For the most part, these features were left in for the ensemble models to remedy.

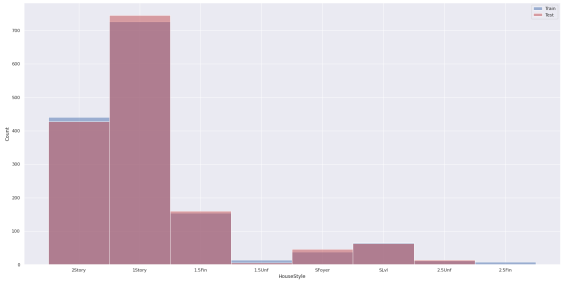

Fig. 1. Four Outliers Eliminated Based on Living Area

Another unfortunate circumstance of the dataset was the lack of information on how some of the variables in the datasets were computed. To give an example, it was found that OverallQual, which rates the overall material and finish of the house, is graded on a highly subjective scale from 1 (Very Poor) to 10 (Very Excellent). Meanwhile, the accompanying text file that is supposed to explain the significance of each real-estate feature fails to justify the metric with any objective measurement. This reason alone generated some skepticism and doubt as to whether the feature should be included in the machine learning models; however, given its high correlation to the sale price, it was ultimately kept for model training.

B. Missing Values

Unique to the dataset, many of the "NaNs" values were recorded due to the fact that particular features did not apply to certain houses.

These were remedied using imputation methods based on the surrounding context of the data. Therefore, these values (mostly from categorical variables) were replaced with a value of "0" to indicate the lack of that aspect's presence. For a few other variables, we replaced the null values with the most frequent (categorical) or median (continuous) value that appeared for the values of that feature. By implementing this technique, most of the missing data were properly filled in to avoid completely eliminating any more observations of the already small dataset.

C. Outliers

On the Kaggle website's discussion tab, there is a strong recommendation from an expert to eliminate four observations immediately from the training dataset before preprocessing. Because of the strong correlation between the sale price and the above-grade living area, a scatter plot of these two identified them quickly. Two of them are true outliers (partial sales that likely do not represent actual market values) and the other two are simply unusual sales (very large houses priced relatively appropriately). Therefore, any houses with sizes more than 4000 square feet were filtered from the dataset, which purged these four unusual observations.

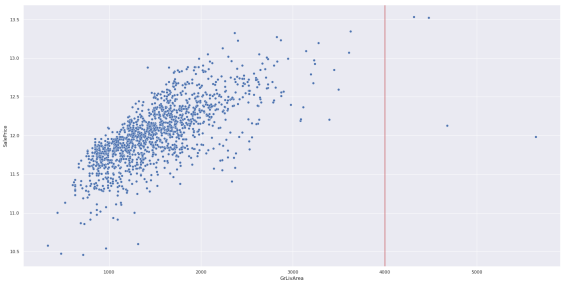

Fig. 2. Four Outliers Eliminated Based on Living Area

One more data point was eliminated from the training data, which appeared as an outlier when plotting the lot frontage of each house grouped by neighborhood. One house in the North Ames neighborhood had an exceptionally large lot frontage area, and the observation was subsequently eliminated from the dataset.

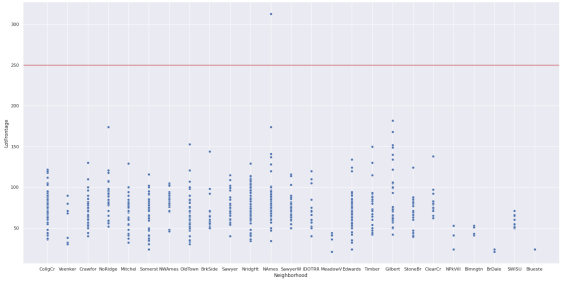

Fig. 3. One Outlier Eliminated Based on Lot Frontage Area

D. Feature Engineering

To both reduce the feature space and make more useful features, new features were generated. Citing a few examples, total square footage, price per lot area, and total number of bathrooms were computed to either operate as a new feature (TotalSF = 1st Floor Area + 2nd Floor Area + Total Basement area) or rid less relevant features while saving important information (TotalBath = FullBath + 0.5 * HalfBath). Curiously enough, this dataset was interesting in that it lacked a feature that indicated whether the houses had a pool or not but had features such as pool area and pool quality. This was surprising, given that an extremely small percentage of houses even had pools; in this case, a single variable (HasPool) replaced the two features (PoolArea and PoolQC).

III. Data Exploration

A. Skewed Variables

Another interesting characteristic of this dataset was the prominence of severe skewness in many of the variables; due to many of the houses lacking a certain feature (e.g., pool area (irrelevant for those without pools or value of miscellaneous items (not many houses have elevators or sheds to account for)), many of the features were right-skewed, with many observations yielding "NaN" for many values as a result.

To begin, the log transformation was applied to the sale price variable, which made it seem much more normally-distributed and symmetrical after application. For the remaining features that exhibited similar skewed behavior, the Yeo-Johnson transformation was applied to numerical variables exceeding a skewness value of 0.5. While most applications of skewed data involve using the Box-Cox transformation instead, due to the presence of many zero values (Box-Cox requires strictly positive values), the Yeo-Johnson was the preferred option.

B. Correlation

Moving on to a bivariate analysis, from the heatmap below, we were able to find some variables such as OverallQual (overall quality of house), TotalSF (total house area in square feet), and GarageArea (garage area in square feet) that were highly correlated with sale price to help us better understand the more important features of the dataset. This knowledge would later help in the modelling process.

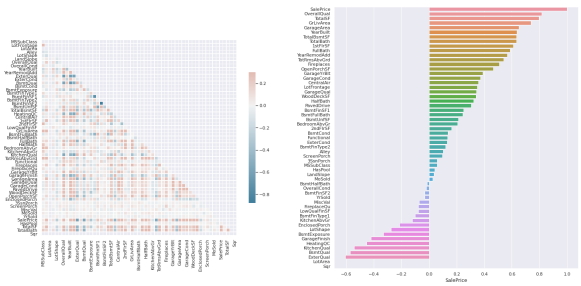

Fig. 4. Correlation Heatmap of Variables against Sale Price

IV. Modelling

A. Ensembles

Given the complexity of the dataset and potential for outliers, an ensemble approach seemed like the optimal way to yield the best performance. This was later verified through online examples that employed similar techniques; winners of machine learning competitions largely attribute their success to ensemble models. In the implementation for this project, a total of four models were employed: three of them being ensemble models (XGBoost, Gradient Boosting, Random Forest) and one being a linear regression model (Ridge Regression).

Ensemble methods are commonly used to boost predictive accuracy by combining the predictions of multiple machine learning models. This speaks of the traditional wisdom that seeks to combine so-called "weak" learners to arrive at a much more robust final model. They also offer significant advantages by way of performance and robustness; they often make better predictions than any single contributing model and reduce the spread of predictions and model performance [2].

B. Model Stacking

However, a more modern approach is to create an ensemble of a well-chosen collection of strong yet diverse models. This method is referred to as model stacking and is an efficient ensemble method in which the predictions, generated by using various machine learning algorithms, are used as inputs in a second-layer learning algorithm. This second-layer algorithm is trained to optimally combine the model predictions to form a new set of predictions. It has also been shown in literature, that model stacking works particularly well for small or medium size datasets [3].

In the case of this project, four different regression methods were used to create predictions (XGBoost, Gradient Boosting, Ridge Regression, and Random Forest). The hope was that the individual weaknesses and biases of each model would be offset by the strengths of the other models.

However, to balance the results by incorporating the individual strengths of each model, a weighted average of the predictions was taken based on the performance of each machine learning model.

C. Hyperparameter Tuning

Each of the regression techniques incorporated required some measure of hyperparameter tuning. To do this, a combination of cross-validation and grid/random search was utilized to find the best parameters yielding the lowest RMSE scores.

Grid Search is most basic and straightforward search technique that has been widely used in many machine learning applications for hyperparameter optimization. Basically, a list of candidate values for each hyperparameter is defined and evaluated. The name "grid" comes to the fact that all possible candidates within all needed hyperparameters are combined in a sort of grid. The one downside of this method is its tendency to be extremely time-consuming when the search space is big. Therefore, as an alternative to grid search, one can instead rely on randomness through the Random Search technique. Grid search was adequate for training the Ridge Regression model since the only parameter that needs to be tuned is alpha, which regulates the L2 penalty. However, for models such as Random Forest and Gradient Boosting, it was necessary to run the Random Search algorithm to avoid spending countless hours of trying out the thousands of different parameter combinations.

V. Results

A. Individual Models

To calculate the results of each of the individual models, a 5-fold cross-validation method was used to arrive at the mean RMSE value for each model (after optimal hyperparameter tuning). Surprisingly, the Ridge Regression model yielded the lowest overall RMSE (0.1148) and Random Forest with the highest overall RMSE (0.1370). XGBoost and Gradient Boosting lay between those two extremes with RMSEs of 0.1226 and 0.1240, respectively. It is important to note here that these values were not set in stone as the hyperparameter tuning algorithms for each model were run multiple times in order to get a good mix of final outputs to submit to Kaggle. But the RMSE values were mirrored across many iterations without much deviation in the bias.

	Models	RMSE
0	Ridge	0.114790
1	XGBoost	0.122587
2	Random Forest	0.137031
3	Gradient Boosting	0.124028

Fig. 5. Mean RMSE Values of Individual Models

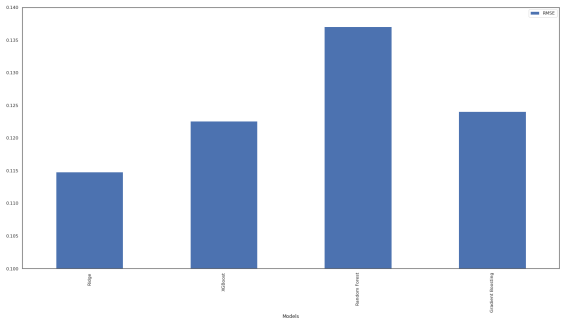

Fig. 6. Mean RMSE Values of Individual Models

Given these outputs, it made sense for the weighted predictions for the submissions to favor Ridge Regression and the stacked regressor. Thereby, it was set that 60% of the final predictions would come from the two strongest models (Ridge and Stacked) and the remaining three models (XGBoost, Gradient Boosting, Random Forest) sharing the remaining 40%.

B. Best Kaggle Submission

After hypertuning the model parameters differently several times and trying out a variety of weights for the blended

predictions, my best predictions submission for the test dataset on Kaggle resulted in a RMSE value of 0.12439, which ranked in the top 19.4% of all teams (1141 out of 5874).

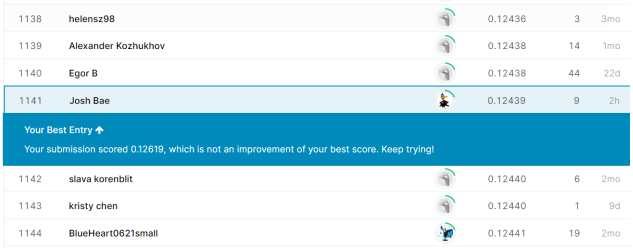

Fig. 7. Kaggle Leaderboard Ranking

VI. Conclusion

Overall, this project was considered a success given the time and resources allotted. While there existed some challenges of missing data points and ill-distributed features, the combination of deliberate data engineering techniques and careful model selection resulted in a highly competitive ranking among data scientist from around the world.

For this particular project, It was evident that by putting time and effort into managing and preparing the dataset and learning the business domain helped achieve a decent score on the Kaggle leaderboards. By saving almost all of the data through coordinated imputation techniques, the models were able to develop into robust ones with solid prediction accuracy.

One surprising takeaway from the modelling section was the success of Ridge Regression for the problem at hand. While it was believed that non-ensemble models would pale in comparison to the robustness of the ensembles ones, the penalized linear regression model took home the crown for best RMSE. After a more thorough analysis, it could be reasonably concluded that this may have resulted from the multicollinearity present in the dataset, which is when Ridge particularly excels. Given the limitations of a house's build (basement area limited by the 1st floor area) and the seemingly repetitive nature of some of the features (GarageCars, the size of garage in car capacity; GarageArea, the size of garage in square feet), its reason for success could be well understood.

One method that could have been attempted to produce even better results had time permitted was running multiple model stacks on top of each other and trying out a variety of different weight combinations for the final predictions. With each iteration, the model would have iteratively gotten better to arrive at a lower RMSE value. Despite this knowledge, it was decided that the marginal returns to accuracy with stacking would not have been worth it at the end. This sequential process would have been much too computationally intensive, with the marginal benefit from obtaining better results outweighed by the sheer cost of running the stacking procedure, both in computation time and power.

References

[1] "House Prices - Advanced Regression Techniques — Kaggle." Kaggle.com, 2016, www.kaggle.com/c/house-prices-advanced-regression-techniques/overview. Accessed 16 Dec. 2020.

[2] Valentini, Giorgio Masulli, Francesco. (2002). Ensembles of Learning Machines. Neural Nets WIRN Vietri-2002, Series Lecture Notes in Computer Sciences. 2486. 3-22. 10.1007/3-540-45808-5_1.

[3] Funda Güneş. "Why Do Stacked Ensemble Models Win Data Science Competitions?" The SAS Data Science Blog, 18 May 2017, blogs.sas.com/content/subconsciousmusings/2017/05/18/stacked-ensemble-models-win-data-science-competitions/. Accessed 16 Dec. 2020.