# House Prices: Advanced Regression

Aditya Lahiri and Ching Yung Chong

## I. Introduction

In this project, we analyzed a dataset provided by Kaggle [1] as a part of a competition. The dataset provided is based off houses in Ames, Iowa. It consists of 79 explanatory variables describing (almost) every aspect of residential homes in Ames and the goal of the project is to predict the final sales prices of residential homes. The dataset is divided into training and testing sets. The training set contains final sale prices for the homes, whereas, the testing set does not contain any information regarding sale prices. The training and testing set contain 1460 and 1461 instances of data points respectively. These datasets contain missing data and various explanatory variables which are non-numeric in nature. Thus, extensive data preprocessing steps are carried out on both the training and testing set to build a viable model to better predict the final house prices. Further information regarding this dataset can be found in the journal paper written by De Cock et.al [2]. The entire analysis of this dataset was done in the R programming language.

## II. Measure of Success

Since this dataset is a part of an ongoing competition hosted by Kaggle the measure of success is defined using the metric Root Mean Squared Logarithmic Error (RMSLE). The RMSLE is defined as follow:

$$\in = \sqrt{\frac{\sum_{i=1}^{n}(\log(\widehat{Y}_i + 1) - \log(Y_i + 1))^2}{n}}$$

In the above equation $\widehat{Y}_i$ represents the predicted sale prices of a houses and $Y_i$ represents the actual sale price of the same house. The total number of data points is represented by n. The lower the value of RMSLE the more accurate is the prediction of the model.

## II. Data Preprocessing

### *A.. Missing Data*

There were several independent features in both the training and testing set where instances of data points were found to be missing. We used the mice [3] package available on R programming to determine the proportion of missing values by category and the results are shown in figure 1 below. Features with more than 15% of missing data are then removed to prevent additional biasness from occurring during the modeling and training process. The mice package is then used to impute the remaining missing data in both the training and test datasets.

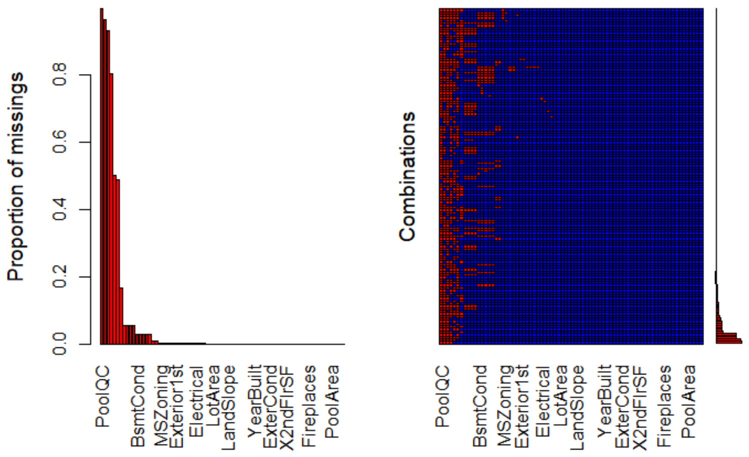

Figure 1: Proportion of missing values within the training and test datasets.

## *B. Correlation Analysis*

A correlation analysis was performed to the dataset to identify features that were highly collinear. The results are shown in the correlation heatmap below. The threshold used to determine high collinearity is 0.75 and the following features are determined to be highly collinear with one another: "GarageYrBlt and Year Built"," TotRmsAbvGrd and GrLivArea", "GarageArea and GarageCars", "X1stFlrSF and TotalBsmtSF". As a result of this analysis, the features "GarageYrBlt","TotRmsAbvGrd",and "GarageCars" are dropped from the training and testing datasets to prevent multi-collinearity from taking effect.

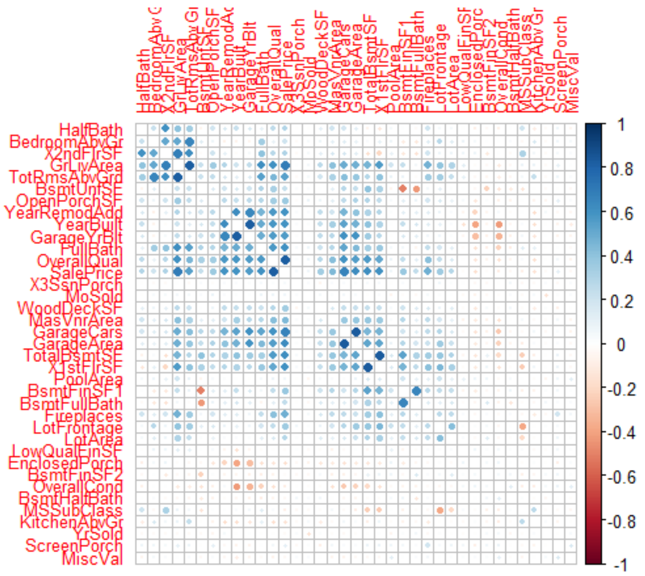

Figure 2: Correlation heatmap of the features in the dataset

## *C. Feature Engineering*

The correlation analysis revealed that the feature "First Floor Surface Area" and "Total Basement Surface Area" were highly correlated. Using domain knowledge of house prices, we know that the total square footage or area of houses is an important factor in its pricing. So, we created a feature called the "Total Surface Area" by adding the features “First Floor Surface Area”, “Total Basement Surface Area”, “Second Floor Surface Area”, and “Total Ground Floor Living Surface Area”. Since the feature “Total Surface Area” is a deterministic function of the rest of the four features, those features were removed to avoid the effects of multi-collinearity in the final predictions.

### *D. Skewed Feature Analysis*

In figure 3 below we plot the sale prices in the training set (left) and observe that it resembles a normal distribution skewed to the left. Furthermore, we use the RMSLE metric to evaluate the models so we log transform the training sale prices to center them. With log transformation the sale prices appear to be more centered (right).

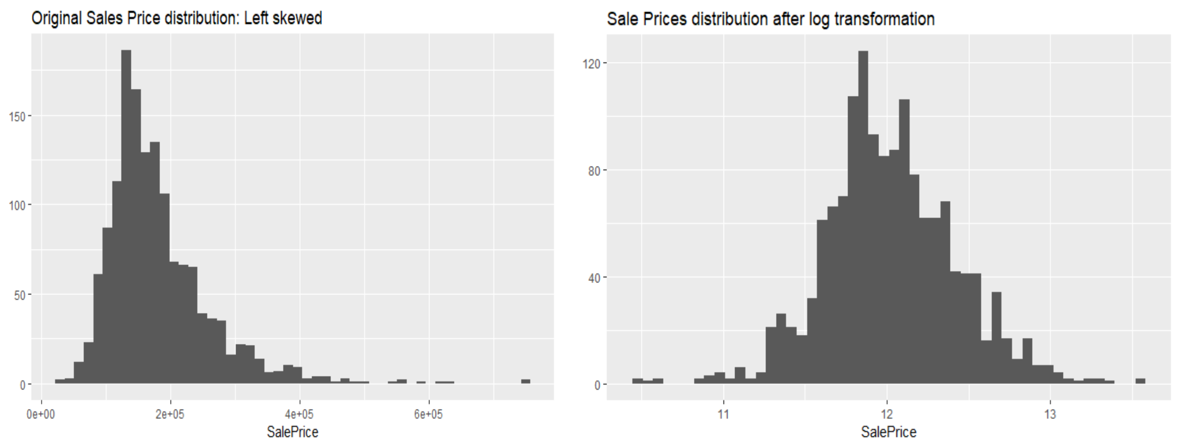

Figure 3: Log transformation of the SalePrice feature before and after.

The independent features in the dataset were adjusted for skew using the Box Cox transformation. If any of the independent features had a skewness larger than 0.75 they were selected for transformation. The Box Cox transformation is a more generalized version of the log transform and is given by:

$$T(X) = \frac{X^{\lambda} - 1}{X}$$

In the equation above X represents the independent feature being transformed, T(X) represents the transformed feature, and $\lambda$ represents the transform parameter. The value of $\lambda$ was set to 0.15 for our analysis.

### *E. Categorical Feature Analysis*

The dataset had multiple features which were non numeric categorical in nature. Some of these variables were nominal and the rest were ordinal. These features were converted to numeric values using Label Encoder and then converted into dummy variables using One-Hot Encoder. Label Encoder and One-Hot Encoder are commonly used categorical data preprocessing techniques. We used them through the CARET package [4] in the R programming language.

*F. Dataset Partition*

The testing set was preprocessed the same way as mention in sub-sections a-e and then set aside. After the training set was preprocessed as outlined above, it was divided into two parts. 25% of the training data was designated as validation dataset and the rest 75% of the training set constituted the training set used for model building. In the sections below we refer to this 75% subsection of training dataset as the training set.

## III. Model Selection & Parameter Tuning

The dependent variable being predicted in this model is the sale price which can be modeled as a continuous random variable. Therefore, the output of the designed model should also be continuous, this why our models are designed as regressors. We selected linear regressors and ensemble learners for the design of the regressor. This was done to ensure models are interpretable and didn't take too much time to train. We selected Lasso, Ridge, Linear SVM, Decision Trees, Random Forest, and XG Boost as our models to predict the sale prices. We also stacked the Lasso and XG Boost regressor to investigate the effects of combining models. Each of the models were fined tuned and cross validated on the validation set. For each of the regressors a grid of parameters was designed over which grid search was ran to reduce the RMSLE. The models with the tuned parameters were then finally cross validated against the validation set. The model with lowest validation was selected as the benchmark model to predict the final house prices on the testing set.

## IV. Results

After each model was trained, it was tuned and validation RMSLE was recorded for each of the tuned modeled. Due to availability of multiple submissions on Kaggle testing RMSLE was obtained for each of the models as well. The results are summarized in Table 1 below.

| Models | Validation RMSLE | Testing RMSLE |
|---|---|---|
| Lasso | 0.1074482 | 0.12634 |
| Ridge | 0.1161661 | 0.13197 |
| Linear SVM | 0.1119615 | 0.13133 |
| Decision Tree | 0.2092982 | 0.22924 |
| Random Forest | 0.1350584 | 0.14275 |
| XG Boost | 0.1177752 | 0.12877 |
| Stacked model: XG Boost + Lasso | 0.1098929 | 0.12405 |

Table 1: Validation RMSLE and Testing RMSLE

Clearly by inspecting the validation RMSLE scores the Lasso model was established as the benchmark model for this project. The Lasso model produced the best testing score when each model was used to predict the sale prices against the test data set. However, by stacking the Lasso and XGBoost model we notice a slight decrease in the Test RMSE. The stacked model was created by just averaging out the sale prices predicted by the individual Lasso and XGBoost models. From this observation we can conclude that model stacking reduced the testing RMSLE even when it was not evident from validation RMSLE. Therefore, we believe the stacked model was able to gain further insight into the intricacies of the dataset and make better predictions. The stacked model placed us in the top 32% (4404 total teams) in the Kaggle competition.

## V. Cross Validation & Tuning Timing

The models were trained on a 64-bit windows machine with intel I5 processor. The version of R used is 3.4.3. No parallelization was applied. The timing for cross validation and tuning was tested using the microbenchmark package in R [5]. The timing results are summarized in table 2 below.

| Models | Timing for tuning and cross validation in seconds |
|---|---|
| Lasso | 1.839367 |
| Ridge | 2.112812 |
| Linear SVM | 194.934937 |
| Decision Tree | 4.156299 |
| Random Forest | 229.381291 |
| XG Boost | 74.62788 |

Table 2: Parameter tuning and cross-validation timing

We found the Random Forest took the longest time to tune and validate whereas Lasso took the shortest time. Also, the number parameters and the grids for each parameter over which tuning is performed varies from model to model which accounts for the varying time across each of the models.

## VI. Feature Importance

Though the stacked model performed better we were not able to select it as our benchmark model, because we used the validation RMSLE to select the benchmark model, which is the Lasso regressor. In general, the testing RMSLE is not available so benchmarking is done based on a validation metric. Based on the Lasso model, we were able to obtain the features that were most important in predicting sale prices, these features are shown in figure 4 below with their importance score. Higher the importance scores the more valuable is the feature in predicting the house prices. We notice that the feature we engineered "Total Surface Area (TotalSF)" was the most important. This is a consistent observation that house prices depend on the size of the houses [7].

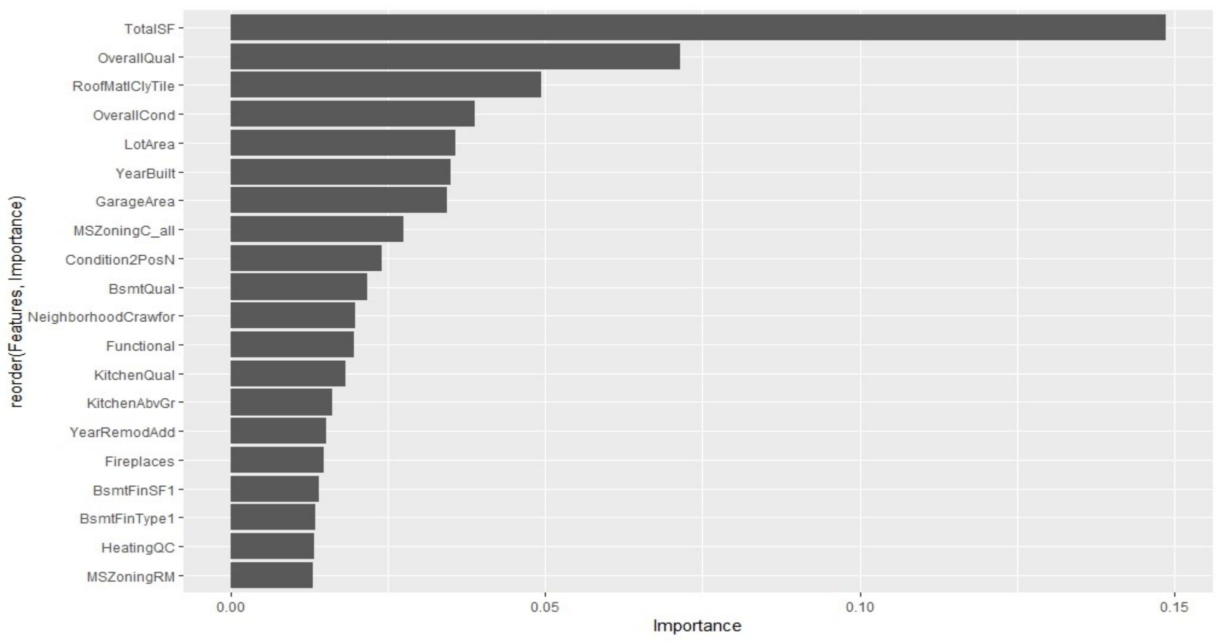

Figure 4: Feature Importance Plot for Lasso Model.

## VII. Availability of data and script

The data used for this project is available on the Kaggle website [1] and the scripts for running our model are publicly available on Github [6]. The script has been written to accommodate different data sets pertaining to house prices from different cities with same features. The script can also work with larger datasets.

## References

[1] Kaggle.com. (2019). House Prices: Advanced Regression Techniques | Kaggle. [online] Available at: https://www.kaggle.com/c/house-prices-advanced-regression-techniques [Accessed 27 Apr. 2019].

[2] De Cock, D. (2011). Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education, 19(3).

[3] Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software, 45(3).

[4] Kuhn, M. (2008). Caret package. Journal of Statistical Software, 28(5)

[5] O. Mersmann, C. Beleites, R. Hurling, A. Friedman , JM. Ulrich. microbenchmark: Accurate Timing Functions. https://cran.r-project.org/web/packages/microbenchmark/index.html

[6] A. Lahiri, CY Chong. STAT689. https://github.com/adilahiri/STAT689

[7] Anon, (2019). [online] Available at: https://www.homeadvisor.com/cost/architects-and-engineers/build-a-house/ [Accessed 28 Apr. 2019].