Ames Housing Dataset Sales Predictions

PSL Fall 2020 - Project 1

Project Members

- Derek Chapman (derek4)
- Zeed Jarrah (zjarrah2)

Overview

This project asked students to use the popular Ames Housing Dataset to make predictions on the sale prices of the included homes. The Ames dataset is a feature rich dataset. There are approximately 3,000 observations with over 80 different aspects coded in for each house. The data is split into 10 separate chunks and students are challenged to beat a benchmark score.

Preprocessing

We determined that 12 features would not be considered when building our models. We decided to completely drop Garage_YR_Blt as it contained NAs and was not an important feature. Next we dropped several features that were better explained by other predictors and we felt were unnecessary such as Pool_Area and Longitude/Latitude. For example the size of a pool isn't nearly as important for a middle class home as whether the home has a pool or not. Others were dropped since they would add many levels and increase the complexity of the model without adding much value since the predictor was overwhelmingly centered on a single value. For example Heating has 6 levels but over 98% of the homes have the same coding of GasA. We take the log of the sale price of a home in order to have a more normalized distribution.

Next, we winsorized several of the predictors at the suggestion of the professor. This is done to reduce the impact of some of the more 'outrageous' values in a few of the predictors. For example Lot_Area has a largest value that is 25x bigger than the median value. A bigger yard will increase the sale price of the house but has diminishing returns. This closely follows the classic advice of: "you never want the nicest houses in the neighborhood". Or more accurately in our case that you don't want the most expensive home since prices typically veer towards the middle. We decided to keep the commonly used value of 95% after exploring the impact of the quantile cut-off value (more below).

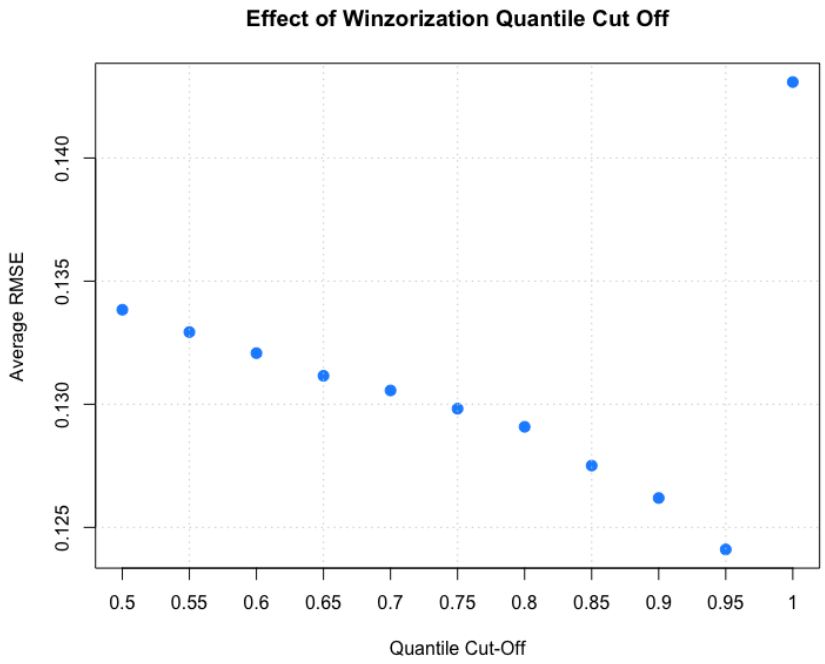
Finally, we manually create the dummy variables for each remaining predictor. This turns the remaining 69 predictors (after immediately dropping 12 and removing the PID and Sale Price) into 303 predictors and dummy variables.

Interesting Findings

In the context of utilizing tree models, it was determined that randomForest was unable to meet the target Root Mean Square Error for any of the training/test splits. However, the relatively few parameters in randomForest was helpful with establishing a baseline for

further improvement. The additional parameters in XGBoost provided the appropriate level of complexity. It was interesting to tune a boosting tree, namely working with the shrinkage parameter and number of rounds. When tuning these parameters it is interesting to note that as you make the shrinkage parameter smaller, one should increase the number of rounds.

We explored the effect of the quantile used for winsorization on The Lasso and Ridge as winsorization seemed to have a much larger effect on that type of model than tree-based methods. We plot the average RMSE value for each quantile cut-off across the ten splits. Surprisingly even setting a quantile that cuts off the top *half* of 'extreme' values does an overall better job than keeping in all original values for all predictors. Interestingly 95% seems to be the magic number which validates the intuition that we only need to cap the *most* extreme values in a given category and leave the rest alone.



Model 1 Parameters

Our first model uses The Lasso for parameter selection and then an ElasticNet for final model creation. After some analysis we choose 0.8 for the alpha value during parameter selection with a lambda value using the one standard error above minimum (..$lambda.1se) value stored by the cv.glmnet function. The combination of these two seemed to give the best results. Setting alpha to a pure Lasso of 1 and then using the minimum score resulted in far too few parameters which led to overfitting and very poor RMSE on the test data. For some of the splits it choose less than 50 of the dummy variables created earlier in the process. And many of these were highly correlated since they were created from the same original predictor. For the model creation we kept the common alpha value for ElasticNet of 0.2 as it did well 'out of the box' and further adjustments did not result in improvements. The model was trained on the variables selected by The Lasso. We did not perform any further refinement and used all of the parameters that had a non-zero coefficient.

Model 2 Parameters

XGBoost with a shrinkage parameter of 0.05, with a max tree depth of 6, sampling on 50% of the observations after 5000 rounds. The shrinkage and subsampling parameters

were both major contributors to the accuracy of the model. It is reasonable to conclude that boosting trees are most impacted by the quality of the regression trees as well as the criteria for pruning the candidate trees.

Scores

The target score for the first five splits was 0.125 and for the last five splits was 0.135. The last five splits in general are more difficult and contain more outliers. Overall the XGBoost model performed the best across all of the training/test splits and was robust enough to accurately predict on the more difficult training/test splits. The Lasso/ElasticNet model also did well but was significantly faster to compute.

	Training/Test Split									
Model	1	2	3	4	5	6	7	8	9	10
XGBoost	0.11316	0.11938	0.11337	0.11643	0.11140	0.12930	0.13163	0.12601	0.12973	0.12603
Lasso/ ElasticNet	0.12454	0.11775	0.12471	0.12453	0.11311	0.13323	0.12670	0.12075	0.13182	0.12391

Runtime and System Info

Windows 10, 3.6GHz, 16 GB ram
XGBoost: approx 3 minutes and 31 seconds
Lasso-ElasticNet: 16.4489 secs or just under 16.5 seconds

Scores for other models we explored

Green shaded models are the two submitted models.
The other 4 red shaded models are ones we explored but did not submit.
The dashed line is the 0.125 benchmark.