Ames Housing Price Modeling

Marcus Tan

May 13, 2021

1 Pre-processing

This project was implemented with the Python libraries `Pandas`, `numpy`, `xgboost`, `sklearn` and `matplotlib`. A MacBook Pro with 2.2 GHz Quad-Core Intel Core i7 CPU and 16 GB 1600 MHz DDR3 memory was used to run the code.

First, the input data was checked for missing or "nan" values, which were then replaced with 0. The only predictor with such invalid values was `Garage_Yr_Blt`. `Sale_Price` (the response variable) did not contain any invalid value.

Next, categorical variables were converted to binary variables using one-hot-encoding, which was achieved with the built in method `get_dummies` from `Pandas`. The number of predictors increased greatly from 81 to 353 after one-hot encoding.

Standardization (subtraction of the mean followed by standard deviation) was applied to the predictor variables. `Sale_Price` was transformed to ln(`Sale_Price`) and standardized.

As suggested by the instructors, Winsorization of the following predictor variables were explored: `Lot_Frontage`, `Lot_Area`, `Mas_Vnr_Area`, `BsmtFin_SF_2`, `Bsmt_Unf_SF`, `Total_Bsmt_SF`, `Second_Flr_SF`, `First_Flr_SF`, `Gr_Liv_Area`, `Garage_Area`, `Wood_Deck_SF`, `Open_Porch_SF`, `Enclosed_Porch`, `Three_season_porch`, `Screen_Porch`, `Misc_Val`.

2 Feature selection

Correlation between the predictor variables and the response variable was calculated with the `Pandas` method `corr_with`. Predictor variables (including all their respective one-hot-encoded variables) with correlation ≤ 0.1 were removed. Those variables were `smtFin_SF_2`, `Bsmt_Half_Bath`, `Condition_2`, `Land_Slope`, `Low_Qual_Fin_SF`, `Misc_Feature`, `Mo_Sold`, `Pool_Area`, `Pool_QC`, `Roof_Matl`, `Screen_Porch`, `Street`, `Utilities`, `Year_Sold`. After one-hot-encoding, there were 311 remaining predictor variables.

3 Cross-validation (CV)

Two models were studied in this project: ElasticNet (a linear regression model) and Xgboost (a boosting tree model). Accuracies of those models using all predictor variables, low correlation variables removed and winsorization were calculated by applying cross-validation on the ten training-test sets provided in `project1_testIDs.dat`. (The portion of the code that does CV has been commented out in the submitted code). The accuracy was characterized by the root mean square error, which was defined as RMSE = sqrt(mean((ln(predicted sale price) - ln(true sale price))^2)). An example of the cross-validation result for ElasticNet is shown in Fig. 1, where the minimum of the maximum of RMSE for all sets occurs around a penalization parameter of $\alpha = 0.008$. For the rest of this report, such a minimum is called the best parameter. When comparing different choices of feature engineering, the best parameter for each choice was used. Fig. 2 compares the different feature choices. As alluded by this figure, ElasticNet requires significantly more effort in feature engineering to improve its accuracy. Eliminating low correlation variables only improves the accuracy slightly. Winsorization with cutoff quantile 92% is crucial for reducing the RMSE below the required level of 0.125 for the first 5 sets and 0.135 for the last 5 sets. Other cutoff quantiles (90%, 91%, 92,%, 93%, 94%, 95% etc) for Winsorization were explored but not shown here. 92%

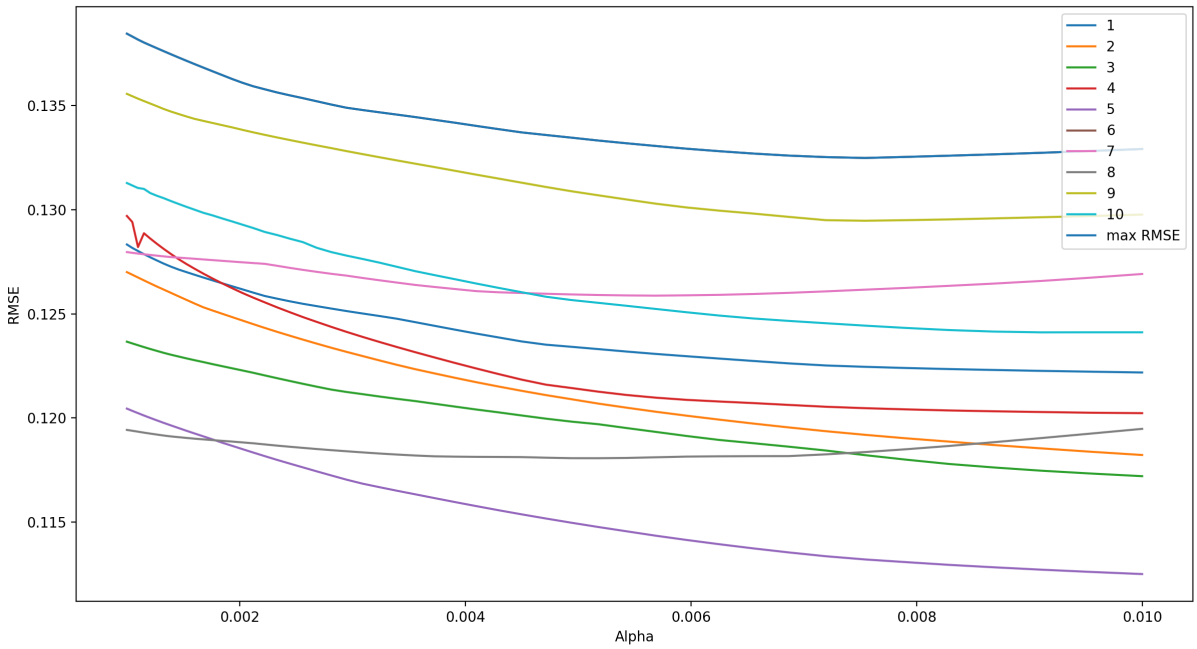

Figure 1: Optimization of the penalization parameter α for the ElasticNet model using CV on 10 training-test sets. Each curve corresponds to a set and "max RMSE" refers to the maximum of all sets at each parameter value.

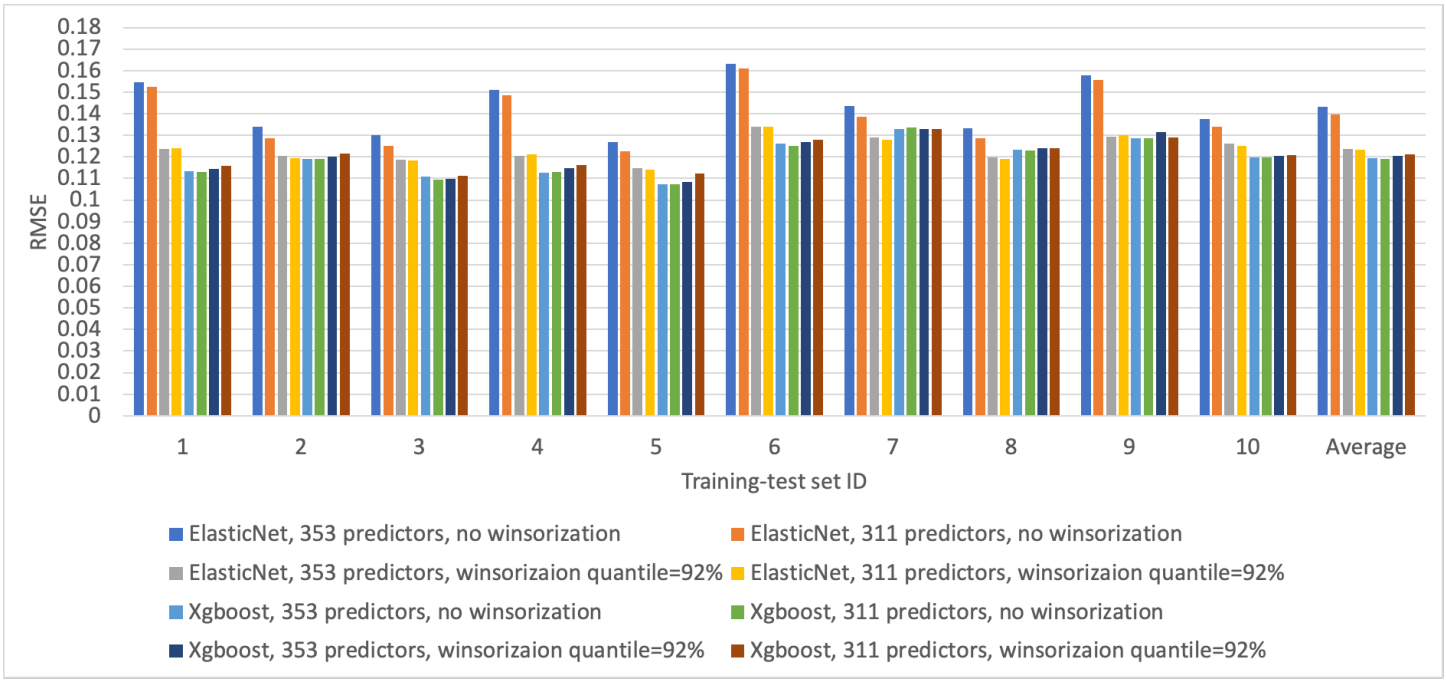

Figure 2: Impact of different feature engineering choices on the accuracy of the ElasticNet and Xgboost models. The original number of predictors after one-hot-encoding is 353. 311 predictors results from eliminating variables with correlation with the response variable ≤ 0.1.

was the best among those that were explored. Removing low correlation variables and Winsorization with 92% quantile cutoff improved accuracy the most for ElasticNet.

It is clear from Fig. 2 that Xgboost does not require much feature engineering to obtain high accuracy. Eliminating low correlation variables helps slightly but Winsorization results in some degradation. On average,

Xgboost is more accurate compared with ElasticNet. However, even with 4 threads, running time was much longer at about 50s compared with 0.12s for ElasticNet. Only two learning rates (0.025 and 0.05) were explored with CV due to the long running time. But the smaller learning rate resulted in higher accuracy.

4 Results

Table 1: Test RMSE for the ElasticNet and Xgboost models trained with the sets given in "project1_testIDs.dat".

Test/training set	1	2	3	4	5	6	7	8	9	10	Mean
ElasticNet RMSE	0.1239	0.1193	0.1183	0.1211	0.1140	0.1338	0.1281	0.1192	0.1300	0.1251	0.1233
Xgboost RMSE	0.1129	0.1189	0.1096	0.1132	0.1075	0.1252	0.1337	0.1228	0.1286	0.1197	0.1192

Table 1 shows the RMSE for the ElasticNet model with $\alpha = 0.0082864$, low correlation variables removed and Winsorization with 92% quantile. The RMSE of the Xgboost model with learning rate 0.025 and with low correlation variables removed can also be found in the table. Both models satisfy the requirement that the RMSE be less than 0.125 for the first 5 sets and 0.135 for the last 5 sets.

5 Conclusion

One linear model (ElasticNet) and one boosting tree model (Xgboost) were explored in this project with various choices of feature engineering. It is clear that Xgboost has less need for feature engineering to achieve high accuracy but with a large price to pay on computational cost. ElasticNet can achieve similar level of accuracy with a judicious choice of features and a fraction of the computational cost.