Project 1 Report: Predicting Ames House Prices

CS 598 Practical Statistical Learning

Graham D Chester, 15-Oct-2018

Overall Approach

The overall approach to this project was as follows:-

1) Examine data and look for feature engineering opportunities, then research Kaggle for additional ideas (see Acknowledgements section below).
2) Form a baseline model using Linear Regression on (close to) the raw data i.e. with just the Garage_Yr_Blt NA values set to the year the house was built, and the categorical variables transformed to dummy variables.
3) A selection of other models was also chosen (Ridge, Lasso, Random Forest, Gradient Boosting Regression Trees from scikit-learn, and also XGBoost) with the view to get experience with all and to choose the best two models at the end.
4) These baseline models (with default parameters) were then run on a 70% split and the 10-fold cross-validated results were recorded.
5) Various feature engineering options were then tried on this 70/30 split and all the baseline models run for each option. If a significant number of the models showed improvement then the feature or transformation was kept otherwise it was rejected.
6) Each models parameters were then tuned using a combination of Grid Search for Lasso and Ridge, and hand-tuning for the GBM. Additionally the scikit-optimize library was used to perform an 'intelligent' search of the parameter spaces (Bayesian optimization using Gaussian processes) to check the hand-tuning hadn't found a local minima. Despite 12+ hours of run-time, the hand-tuned / grid search models performed slightly better than the scikit-optimize approach.
7) The various feature engineering options identified in 5 above were then reapplied, checking that they still made a significant improvement in the modelling (they did).
8) The best two models were chosen based on mean performance and also consistency of performance. For a final check of overall performance, these models were also run on the 10 sets of test ids provided and the RMSEs recorded. A simulation was then run to calculate the chance of achieving a mean RMSE on 3 of the 10 sets, of less than 0.12 (shown in Figure 3 in the Appendix below).

The technology used for this project was Python, Jupyter notebooks, with Scikit-learn and XGBoost libraries. The modeling was run on an iMac i7 4.2GHz, 40GB and the runtime of the Python code is 25 seconds. The metric used throughout this report is the 10-fold cross-validated RMSE on the log Sale_Price prediction error.

Feature Engineering

The following Feature Engineering step were found to be beneficial enough to include in the final modelling:

1) Garage_Yr_Blt NA values were updated to the house Year_Built values based on the fact that most garages would be built when the house was built (and also the lack of any other specific information!)
2) Drop the Utilities, and Condition_2 columns as they are almost all a single value.
3) Combine rare values (< 3 occurrences) in categorical columns to a single dummy value.
4) The following variables were changed from numeric to categorical based on the fact there may be a non-linear relationship between the categorical values (Bsmt_Full_Bath, Bsmt_Half_Bath, Full_Bath, Half_Bath, Bedroom_AbvGr, Kitchen_AbvGr, TotRms_AbvGrd, Fireplaces, Garage_Cars)
5) A numeric variable was created for each categorical variable, and it's value was set to the mean sale price for the particular category value.
6) Approximately 20 new variables were created as derived from the existing variables, summing some key values and in other cases multiplying as appropriate to create meaningful measurements that were judged to affect house price (for example, zTotalPlace was the sum of 8 other areas). The initial list was obtained as described in the Acknowledgements section, however it was refined and added to.
7) For columns that are highly skewed (>5), a log transformation was applied to improve normality.

8) Perform PCA on the training data and add the first 5 principal components to the train and test

Results

The results of the above feature engineering steps are shown in Figure 1 below, where almost every model improved at almost every step. The notable exception is that Ridge did not respond well to adding new columns, but responded very well to log transform of high skew variables. The linear regression benchmark is shown only on (almost) raw data, due to collinearity on the feature engineered data causing problems.

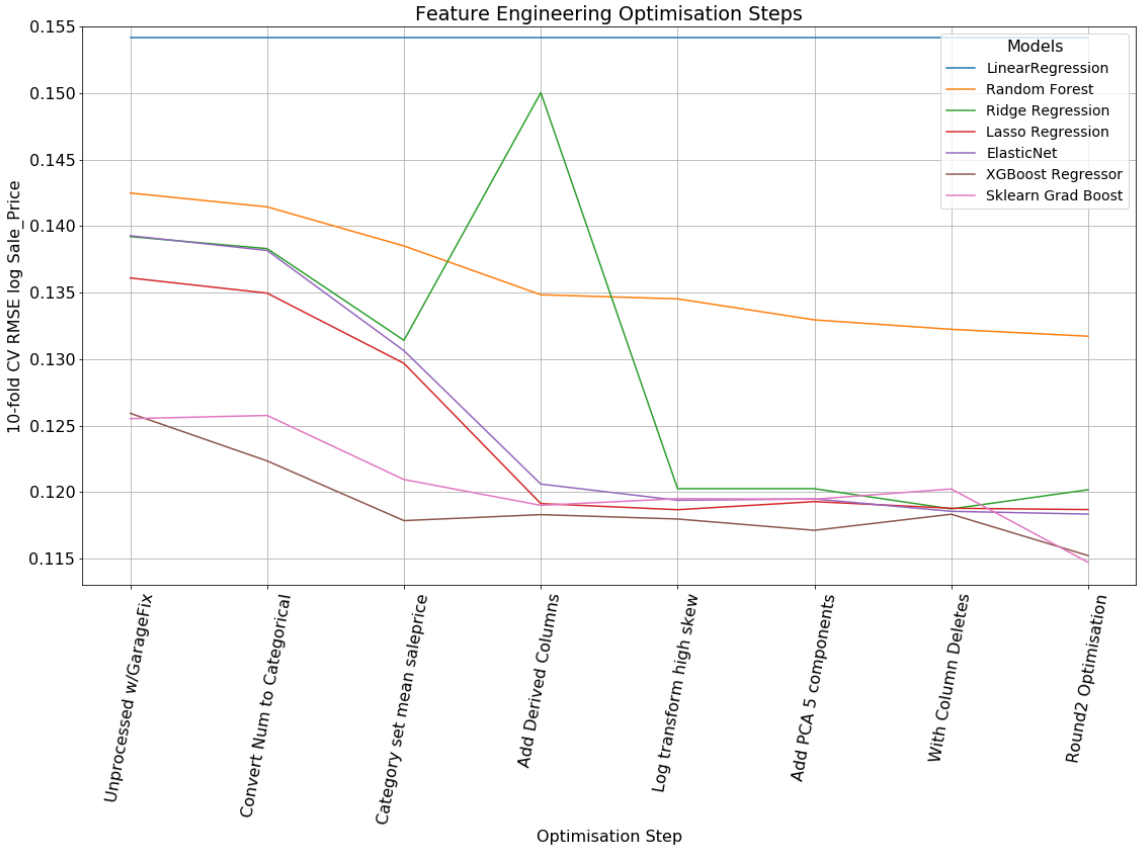

Figure1: RMSE optimization with Feature Engineering and Model Tuning

A comparison of the final 10-fold CV RMSE performance for all models are in Figure 2 below.

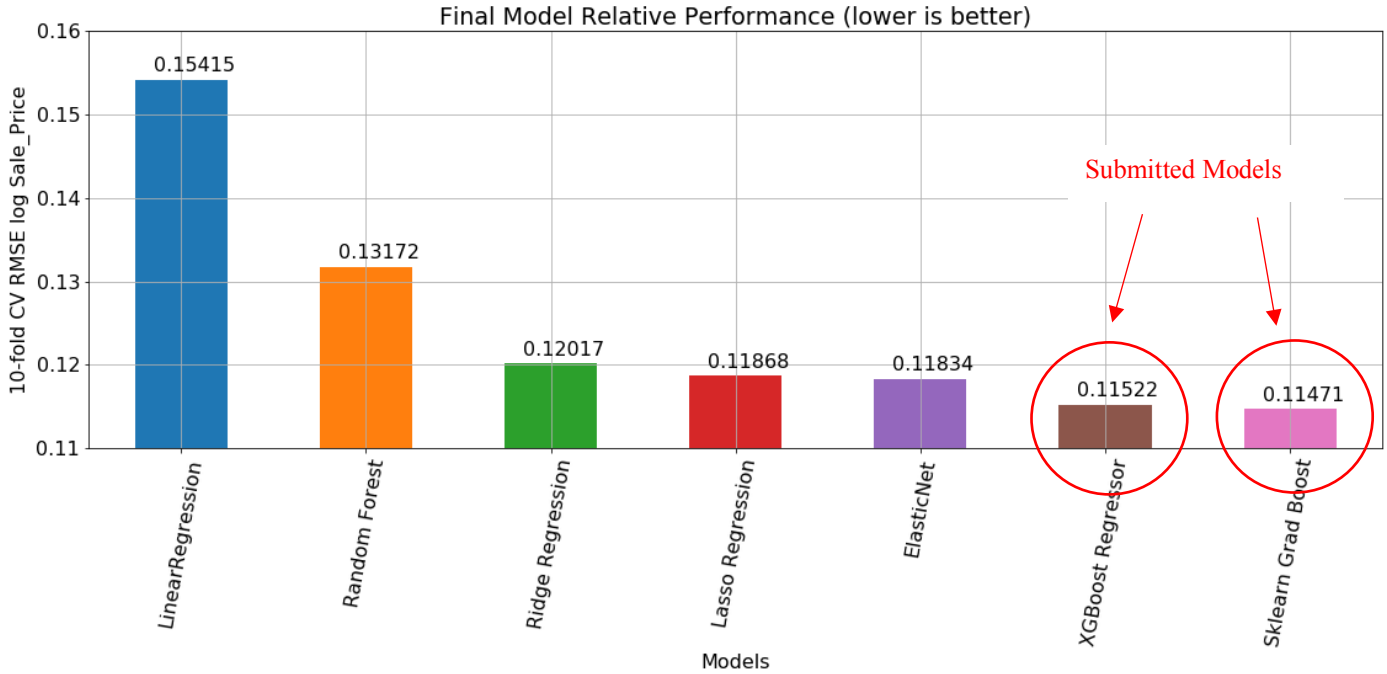

Figure2: Final Relative Model Performance

Conclusion

On this dataset, gradient boosted models required far less feature engineering than the linear models, so their performance improved more with tuning than it did with feature engineering, and the tuning was time consuming. Conversely the linear models performance improved more with feature engineering than with tuning, however the tuning was much simpler than with the gradient boosted models.

The final results were that the linear regression benchmark on the raw data was significantly outperformed by all other models. Random forest also performed relatively poorly. However Ridge, Lasso and ElasticNet all performed quite well, and were only outperformed by scikit-learn's gradient boosted regressor and XGBoost. Somewhat surprisingly, XGBoost was not quite the best performer as may expected from its reputation, however perhaps this was due to parameter tuning.

The two models chosen for submission were scikit-learn Gradient Boosted Regressor (GBR) and XGBoost. These two models performed better than the linear models, though ElasticNet was not far behind and would have been a reasonable choice if the model families were required to be different.

Both the GBR and XGBoost models achieved 10-fold cross-validated RMSE of less than 0.120 (as shown in Figure 2 above). And on the 10 data splits provided for this project, GBR obtained an average RMSE of 0.11338, and XGBoost obtained and average RMSE of 0.11486 (as shown in the appendix) – also both less than the 0.120 required.

Acknowledgements

The initial list of derived variables in the Feature Engineering was obtained from a Kaggle kernel by Alexande Papiu at https://www.kaggle.com/apapiu/regularized-linear-models.

Various Kaggle kernels, discussions, along with UIUC lectures and internet searches also contributed to the ideas utilized in this project.

Appendix A – Performance Proof

The following screen print shows the performance of the 2 best models for each of the 10 test id splits provided in the assignment instructions. Performance worse than the 0.12 cutoff has been circled. Also the chance of a mean RMSE less than 0.12 is calculated by simulation (by choosing 3 splits at random from the 10 splits, repeat 50,000 times, and test the mean of the 3 splits to the 0.12 cutoff).

```
XGBRegressor : Processing dataset 0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,
XGBRegressor Avg RMSE= 0.11486 [0.11339 0.11935 0.13066 0.11923 0.09979 0.11665 0.10965 0.10628 0.119   0.11458]
Chance of mean less than 0.12= 0.8833

GradientBoostingRegressor : Processing dataset 0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,
GradientBoostingRegressor Avg RMSE= 0.11338 [0.11126 0.12026 0.12803 0.11831 0.0964  0.11834 0.11023 0.10483 0.11683
0.10935]
Chance of mean less than 0.12= 0.9494
```

Figure3: Performance Proof

Appendix B – Python Environment Requirements

Python 3.6: Pandas version=0.22, Numpy version=1.15, Scipy=1.1, Scikit-learn=0.12, xgboost=0.8
The results in Appendix A using these libraries are very slightly better than with the older version of xgboost used in the stat542 build environment.