Project 1 Report – Predict the Housing Prices in Ames

Team MAK Members

Mriganka Sarma (ms76), Ajay Menon (kamenon2), Kai Pak (kaipak2)

Contributions:

Each team member has attempted to develop the models independently to achieve the performance benchmark. The final submitted version was developed by Kai and was reviewed and modified by the other team members using GitLab. The final report is written collaboratively by all the members using SharePoint.

1. Objective

In this project, we use supervised machine learning techniques to predict the sale price of homes in the Ames Housing Dataset using numerical (e.g. square footage, number of bathrooms, etc.) and qualitative (e.g. neighborhood, home style, etc.) characteristics of the home as inputs. The log sale price is the target variable while the other features make up the predictors constituting the design matrix. For our loss function, we seek to minimize the root mean square log loss error (RMSLE):

$$RMSLE = \sqrt{\frac{1}{n}\sum_{1}^{n} \ln\left(\hat{y}_i\right) - \ln(y_i)}$$

The criteria goal is ***RMSLE* ≤ *0.125*** on test sets 1:5, and ***RMSLE* ≤ *0.135*** on test sets 6:10 which was successfully achieved.

2. Data Pre-Processing

The original data set contained 2930 observations and 83 columns. Using the provided Test IDs, 10 sets were generated. During exploratory data analysis (EDA), attribute selection was done based on the correlation coefficient we had calculated for each variable against the Sale Price. Coeff closer to -1 and 1 were considered and closer to 0 were ignored and all variables within a range -0.25 to 0.25 coefficient's seemed to be good candidates for imputation.

2.1. Cleaning

- *Garage_Yr_Blt* NA values replaced with 0. The reason for NA is that the there was no garage built and Garage Type = ***No_Garage***, ***Detached*** etc. Typically, we would consider mean for such imputations, but in this case 0 was appropriate.
- *Garage_Yr_Blt* > 2010, since this is the last year of the dataset. We discovered a value of 2207 which was corrected to 2007 since this observation has a remodel year of this value.
- All other columns appeared to have sane values and required no further cleansing.

2.2. Transformations

- Applied logarithm transformation to *Sale_Price*.
- We used the caret package to apply one-hot encoding to the categorical variables. Initially, ***fullRank*** option was set to TRUE to avoid perfect collinearity for binary variables, but we found performance was slightly better for ElasticNet when this was set to FALSE. It had no discernable effect on the tree-based methods we tried.
- The following numerical variables contained outliers or produced skews. So Winsorization was applied to reduce this since such skews can hurt model performance due to high leverage.
  - *Lot_Frontage*, *Lot_Area*, *Mas_Vnr_Area*, *BsmtFin_SF_2*, *Bsmt_Unf_SF, Second_Flr_SF, First_Flr_SF*, *Gr_Liv_Area*, *Garage_Area*, *Wood_Deck_SF, Open_Porch_SF*, *Open_Porch_SF*, *Enclosed_Porch, Three_season_porch, Screen_Porch, Misc_Val.*
- *Overall_Qual* and *Overall_Cond* were converted to ordinal values of 1-10.

Project 1 Report – Predict the Housing Prices in Ames

2.3. Variable/Predictor Removal

The following variables were removed because they were dominated by certain class factors (like Roof_mtl) and would provide little value as predictors since they were nearly homogenous, or they provided questionable predictive capability (latitude/longitude).

- *PID, Street, Utilities, Condition_2, Roof_Matl, Heating, Pool_QC, Misc_Feature, Low_Qual_Fin_SF, Pool_Area, Longitude, Latitude*

2.4. Feature Engineering

Following additional features were introduced to the design matrix.

- *Bath_to_Bedroom*: Ratio of 'Sum of all bathrooms' to 'Bedrooms above grade'
- *Bedroom_to_Gr_Liv_Area*: Ratio of 'Bedrooms above grade' to 'Above grade living area'
- *Has_Basement*: Binary indicating presence of basement

3. Model Selection and Performance

3.1. Model Selection:

3.1.1. Linear Model:

We selected ElasticNet as the linear model with an alpha value = 0.5 to get it enabled and thereby get both Lasso and Ridge penalties mixed. Instead of having two different lambda values (to control lasso and ridge penalties) we have an alpha component to control the regression penalties and by using cv.glmnet() we got the 10-fold cross-validation based optimum lambda value.

3.1.2. Tree-based Model:

We initially tested extensively with Random Forest model but were not able to consistently beat the benchmarks. So we selected XGBoost instead with the following parameters:

Parameter	Value	Meaning
max_depth	6	Maximum depth of a tree; we're using the default value 6.
eta	0.05	Learning rate in range (0, 1) that controls the contribution of each tree.
subsample	0.7	Ratio of the subset of train data used to fit a regression tree; range (0, 1).
nrounds	5000	Maximum number of boosting iterations. Large for small eta.

Both models required minimal hyperparameter tuning to achieve decent results. Benchmarks were beaten for every test set.

3.2. Model Performance:

We have achieved the following results using the above models:

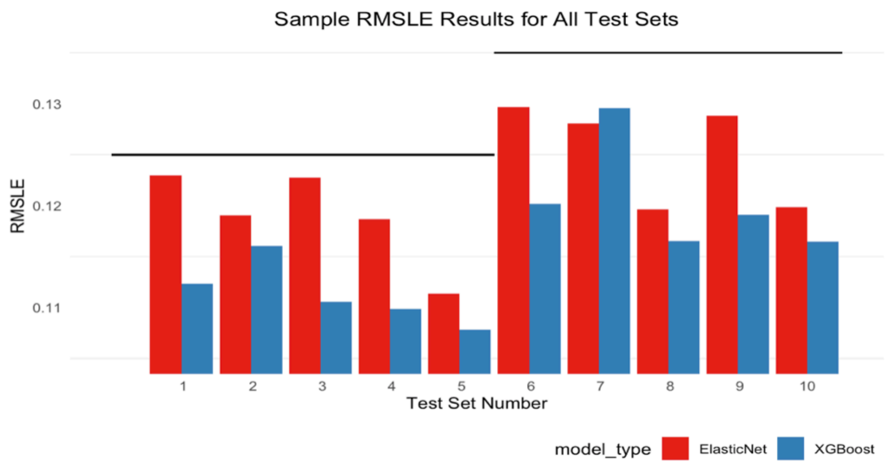

Figure 1: Test RMSLE results. Black horizonal lines indicate benchmark thresholds.

Test	ElasticNet RMSE	XGBoost RMSE
1	0.12296	0.11236
2	0.11907	0.11606
3	0.12275	0.11058
4	0.11869	0.10987
5	0.11136	0.10783
6	0.12970	0.12018
7	0.12804	0.12959
8	0.11963	0.11652
9	0.12880	0.11912
10	0.11985	0.11648

3.3. Model Runtime:
The models were trained and tested using the following systems:

- MacBook Pro (15-inch, 2017) with Processor 2.8 GHz Quad-Core Intel Core i7 and Memory 16 GB
- MacBook Pro (13-inch, 2016) with Processor 3.3 GHz Dual-Core Intel Core i7 and Memory 16 GB

The runtimes were better with the newer (2017) MacBook Pro and those runtimes are listed below:

Test	ElasticNet Runtime (s)		XGBoost Runtime (s)	
	Train	Test	Train	Test
1	2.335	0.027	136.614	0.084
2	1.947	0.010	127.665	0.081
3	1.720	0.008	124.771	0.082
4	2.056	0.008	116.533	0.069
5	1.934	0.025	106.207	0.089
6	1.844	0.009	130.011	0.082
7	1.908	0.009	127.545	0.096
8	1.687	0.008	125.657	0.085
9	1.708	0.008	120.808	0.088
10	1.989	0.009	116.436	0.081

For ElasticNet, training time took an average of 1.913s with a standard deviation of 0.195s while XGBoost required a mean of 123.225s with a standard deviation of 8.553s.

4. Discussion and Conclusion

Model selection and tuning clearly has an important effect on prediction performance, although, we found that data cleansing and feature engineering aspects netted the largest gains for all models compared to fine tuning hyperparameters. They are all important aspects of the machine learning pipeline, however this underscores the criticality of good data processing flowing from thoughtful analysis to achieve best model performance.

Attribute selection was done based on the correlation plot created during EDA and highly correlated variables to Sale Price were considered and the ones having coefficients closer to 0 were ignored. On the original dataset containing 83 variables, once the dummy variables were created for the categorical data, we ended up with 307 variables. Based on the correlation coefficients calculated, we were able to retrieve 59 variables which seemed highly correlated to Sale price.

Inclusion of latitude and longitude in the design matrix provided for some discussion within the team. Although initially removed as predictors, correlation analysis revealed some potentially useful patterns. We tried some methods including converting to factor variables using cut, but we did not observe measurable improvements. However, cluster analysis revealed some inherent structure in these variables that might potentially be used in an ensemble method in a later study. For these results, they remained excluded.