STAT 542 Project 1 Report

Deli Yang (deliy2)

1 INTRODUCTION

The goal of the project is to predict the price of a home sold in Ames, Iowa between 2006 and 2010 with explanatory variables such as the height of the basement ceiling or the proximity to an east-west railroad. The data set contains 2930 observations and a large number of explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values (De Cock, 2011). We built two prediction models selected from the following two categories: one based on linear regression models with Lasso, and the other based on tree models, such as boosting tree.

Submissions are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.

2 PRE-PROCESSING

2.1 Missing Values

Originally, we planned to identify NA values and replace them with 0. However, later we found out that there aren't many missing values in the Ames data: 159 missing values, all from the "Garage_Yr_Blt". Further investigation reveals that those 159 houses exactly correspond to category "No_Garage" from a categorical feature Garage_Cond. In another related categorical feature Garage_Type, 157 out of the 159 houses do not have garages and 2 of the 159 houses have detached garages. In short, the 159 houses miss Garage_Yr_Blt because no garages have ever been built for these houses. I would just replace the missing values with zero.

In fact, when missing exactly corresponds to a level of another categorical variable (in this case, missing of Garage_Yr_Blt corresponds to level "No_Garage" in another categorical variable Garage_Cond), it doesn't matter which value you'll use to replace the missing and the resulting LS model stays the same.

2.2 Remove Imbalanced Categorical Variables

We decide to remove imbalanced categorical variables due to the concern that an indicator variable for just a small set of samples tends to overfit those samples. For instance, we decided to remove "Longitude" and "Latitude" since we do not think they are interpretable predictors.

We also removed extremely imbalanced categorical variables with most samples belonging to one category. Furthermore, we dropped some variables with negative effects on the performance.

2.3 Winsorization

Winsorization began as a way to "robustify" the sample mean, which is sensitive to extreme values. As the sample size increases, the impact of outliers on probability of Type I errors decreases, and a relatively larger percentile (90^{th} or 95^{th}) of Winsorization is sufficient to accommodate the effects of outliers to achieve an acceptable Type I error rate (Liao, 2016). In our case, winsorizing at the 95^{th} percentile shows a good control of Type I error rates across all sample sizes.

In this case, we apply winsorization on some numerical variables: compute the upper 95% quantile of that variable based on the train data, denoted by M; then replace all values in the train and also test that are bigger than M by M. For example, the contribution of the size of a house to the selling price can be well approximated by a linear relationship only within a certain range of the square footage; such contribution usually becomes flat; this is why we winsorize those features.

2.4 Remove Categorical Features

The Ames House data contains both numerical and categorical information. In particular, generate K binary dummy variables for any categorical feature with K levels (K>2). With Lasso, we would like to pick a reference level that can lead to a sparse coefficient vector. The solution here is not to pre-specify any reference level; instead, code all K levels as K binary indicator variables.

3 MODEL PREDICTION ACCURACY

First, we use Lasso with lamda.min to select variables and then fit a Ridge regression model on the selected variables using lambda.min and using an elastic net with alpha = 0.2. Second, we use boosting tree. Before calling XGBoost, I need to change the X matrix to a numerical matrix (no factors). For tree models, we need to generate K binary categorical variables when K>2.

The table below reports the accuracy of the test data. The minimum RMSEs from both prediction models are all below the benchmarks (0.125 for the first 5 datasets; 0.135 for the rest of the datasets). For the first 5 datasets, compared to LASSO, Boosting Tree has better performance overall. However, for the last 5 datasets, the 2 algorithms have similar performance.

Dataset No.	LASSO	Boosting Tree
1	0.1226248	0.1216553
2	0.1179168	0.117769
3	0.1204814	0.1154785
4	0.1198121	0.1152072
5	0.111469	0.1106916
6	0.1336539	0.1324905
7	0.1266296	0.1301144
8	0.1208138	0.1281927
9	0.1307505	0.1288689
10	0.123968	0.1210084

4 MODEL RUNNING TIME

We run the models on a MacBook Pro laptop:

> macOS Monterey (version 12.1), MacBook Pro (M1, 2020), Apple M1 Chip, Total number of cores: 8 (4 performance and 4 efficiency), Memory 16GB.

The table below reports the running time:

Dataset No.	LASSO/s	Boosting Tree/s
1	0.972	26.189
2	0.668	26.067
3	0.801	26.223
4	0.727	27.179
5	0.703	27.349
6	0.701	26.882
7	0.689	25.788
8	0.713	26.478
9	0.704	25.779
10	0.725	25.993

LASSO's running time is much shorter than Boosting Tree's. This is because LASSO uses a global regression model instead of calculating parameters for small intervals between nodes in the dataset. Due to the fact that 10,000 trees had to be built for each dataset and predictions had to be made based on all of these trees, boosting trees took such a long time.