Predict the Housing Prices in Ames

Delong Meng

Oct 16th, 2021

Introduction

Statistical learning and machine learning strategies have been widely used in predicting categories (classification) and continuous variables (regression). In this project, I aimed to train regression models to predict house prices. Specifically, I utilized the Ames housing dataset, including 2930 rows (houses) and 83 columns (features). One of these features, the price of the house, serves as the response variable that we would like to predict using other features. Here, I explored both linear regression models and tree-based regression models to achieve the prediction goal.

Methods

Train/test splitting of the data and performance evaluation

To train and evaluate prediction models, the dataset was randomly split into train set (70%) and test set (30%). To make the most use of the data and get different replicates, 10 different sets of the train/test split were generated. For each set, models were first trained based on the 2051 train samples (70%), and then applied to the other 879 test samples (30%) to test their prediction performance. The performance was evaluated by the test root mean square errors (RMSEs) based on the logarithm of the true and predicted house prices.

Pre-processing of the data

Data pre-processing is usually a necessary step before we feed data into models. Here, logarithm transformation was performed for the response data (price) for both train and test dataset. In addition, missing data might cause problem with modeling and prediction. There are some missing data for the year information of when the garage was built due to lack of garage in some houses, so those missing entries were converted to 0.

Furthermore, winsorization is another data transformation technique that was used for this project. It aims to limit the extreme values in the data. Here, 95% percentile was artificially chosen to cut off the signal for a subset of numerical variables regarding area of lot or rooms: 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. Data greater than the 95% percentile for a given variable in the dataset were cut down to the 95% percentile value.

Lastly, categorical variables were manually converted into dummy variables (i.e., one-hot encoding) for all the levels seen in the train set for the convenience of modeling.

Note that both train and test set were processed the same way, and the test data were processed according to the same parameters used in the train set, because in reality, one can have only one single piece of test data to make prediction on. For example, for winsorization, the threshold to cut off the high signal for the test set was pre-determined by the train set and thus a fixed set of values. Similarly, for the categorical variables, test data were one-hot encoded according to the possible levels seen in the train data. New unseen values in the test data were automatically ignored.

Model 1: A linear regression model

Linear regression models with regularization were explored for this dataset. L1-norm penalty (Lasso), L2-norm penalty (Ridge) or a combination of L1-norm and L2-norm penalty (Elastic Net) were tested. Elastic Net was chosen for the final model and the parameter alpha was set to be 0.5. Modeling and prediction were performed using the R Glmnet package. The alpha parameter determines the percentage of L1-norm penalty (versus L2-norm penalty), so alpha = 1 is equivalent to Lasso, alpha = 0 is equivalent to Ridge model, and alpha values between 0 and 1 correspond to Elastic Net models. Several alpha values were compared, and no big difference was observed in this project.

The lambda parameter (for penalty strength) was chosen using cross-validation based on the train set. Specifically, the lambda value for the minimum cross-validation error ("lambda.min") was chosen for each train set and applied on the test set.

Model 2: A tree-based regression model

Tree-based regression forest models were also explored for this dataset. The boosting-based regression forest was utilized for the final model (the R XGBoost package). The learning rate parameter eta was set to be 0.05 (typically 0.01-0.2), the tree depth parameter max_depth was set to be 6 (default), the subsampling rate parameter was set to be 0.5 (50% of training data to sample for each tree; typically 0.5-1), and n_rounds, i.e., the number of decision trees in the final model, was set to be 1000.

Results

Performance of the two models on the test data set

As shown below, the test RMSEs of the two models for the 10 Train/Test splits were summarized in Table 1.

Table 1. Test RMSEs of the two models

Train/Test split	RMSE (Model 1)	RMSE (Model 2)
1	0.124	0.112
2	0.119	0.119
3	0.122	0.113
4	0.120	0.116
5	0.114	0.111
6	0.134	0.129
7	0.129	0.132
8	0.121	0.129
9	0.131	0.130
10	0.125	0.127

Running time

Total running time of 10 splits were around 6 minutes. Note that the computer system used in this project was: MacBook pro, 2.2 GHz, 16 GB memory. The breakdown of the running time by processes (pre-processing of the train or test data sets, or the training and prediction time for the two models) and the Train/Test splits are shown in Table 2.

Table 2. Running time of processes (in seconds)

Train/Test split	Pre-processing (train set)	Pre-processing (test set)	Training & Prediction (Model 1)	Training & Prediction (Model 2)
1	0.074	0.033	2.085	28.113
2	0.103	0.038	2.293	34.224
3	0.061	0.026	3.446	39.622
4	0.073	0.038	2.675	35.475
5	0.086	0.035	2.976	37.525
6	0.062	0.03	2.694	35.354
7	0.081	0.029	2.436	32.858
8	0.062	0.031	2.769	32.634
9	0.081	0.03	2.364	33.499
10	0.051	0.028	2.429	32.984

Discussion

The overall performance of the two models used in this project were better than the benchmark and were comparable (Table 1). For linear regression models, it is interesting that the Elastic Net outperformed both Lasso and Ridge models (data not shown here) in the preliminary exploration. The parameters (an alpha value of 0.5 and minimum lambda value determined by cross-validation) are both commonly used in the field and appear to be good choices in this project. This particular dataset of house prices has 81 predicting features. Some of them are internally related (such as the area of rooms, the number of rooms, and the area of the lot), and some of the features may not be significantly contributing to the response. Elastic Net method combines the advantage of the Lasso (feature selection) and Ridge (coefficient shrinkage) models and can handle features with internal correlations. For decision tree-based regression models, the Gradient Boosting Machine (GBM) models outperform random forest models (data not shown here) in the preliminary exploration. Parameters used in this project are within common ranges. The n_rounds (number of trees in the final model) were found to be determining the speed of the modeling process. In our preliminary exploration, n_rounds values of 100 to 10000 were tested, and 500-1000 trees were found to be good enough to achieve the optimal performance.

Several data pre-processing techniques have been used in this project. It is worth mentioning that Winsorization of some numerical variables improved the performance of the linear regression model, but did not influence the GBM model (I still included this processing step for the GBM model). This might reflect the fact that there is some non-linear relationship between those predictors and the response, and clipping the signal eliminated the negative effects of some extreme values on the model.

For the runtime analysis (Table 2), the preprocessing of the train or test datasets can even be ignored (within 0.2 second), the Elastic Net model was pretty fast (2-4 seconds), while the GBM model took significantly longer time (around 30-40 seconds) to build. As mentioned above, this is highly correlated with the number of trees (n_rounds). Further optimization could be done to reduce the runtime. From the perspective of speed, the Elastic Net model seems to be a better option over the GBM model for this project.

In the future, to improve the performance of the prediction model, the parameters for the models could be more systematically tuned using a grid search. In addition, other machine learning models or data processing techniques could also be considered.