Predict the Housing Prices in Ames

Problem

You are asked to analyze the housing data collected on residential properties sold in Ames, Iowa between 2006 and 2010. There are 79 explanatory variables describing (almost) every aspect of residential homes and the goal is to predict the final price of a home with those explanatory variables.

Feature Extraction from Data Frame

For feature engineering a helper function was employed. This function was loosely based on the code provided in class[1].

Helper function “helper_fxn.R”: For each of the seventy nine features the number of missing elements was calculated. Features with the most entries missing were dropped from the data frame. The features which we dropped from our analysis were "Alley", "PoolQC", "Fence", "MiscFeature", "FireplaceQU". These features were removed from both the test and train data. The features were then grouped into categorical and numeric variables. For categorical variables missing information was labelled as a new level- NA. For numeric variables missing elements were replaced by median value of that variable-(feature). Median was used instead of mean to prevent bias which arises from data being skewed. Log transform of “SalePrice” was used due to rightward skewness, with some high numerical values. For other features log transform was only performed on features where the skewness ratio was higher than 0.75. All the transformations were performed on both test and train data to maintain data consistency.

Evaluation

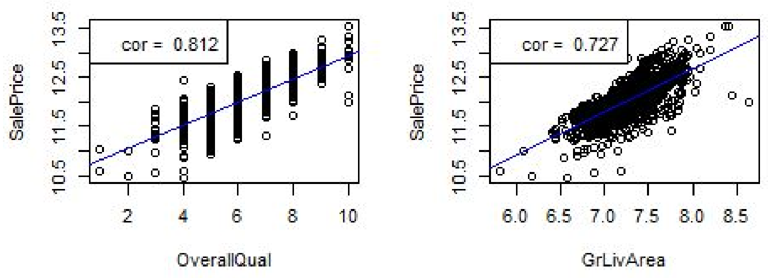
The evaluation was performed using Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price (based on the Kaggle description). For each model, 10-fold cross validation was performed and average run time and RMSE was reported for data which was randomly split ten times using a 10% test - 90% train ratio.

Root Mean Squared Error (RMSE): This function calculates mean squared error for the predicted y-hat versus the test data for each fold, to evaluate the prediction for each fold.

A) Simple Model (script = Simple_Model_ZR.R): Features with correlation coefficients higher than 0.7 were used to perform Multiple Linear Regression (MLR). In order to select high correlation variables, 4-fold cross validation was performed on the data frame such that in each split the original training data available at Kaggle was divided into train data-75% and test data-25%. Coefficients greater than 0.3 were considered for further analysis. To further select features highly correlated with SalePrice, a loop was conducted to calculate RMSE between the test and predicted value obtained using features selected due to having a correlation that exceeded a threshold, using a threshold correlation coefficient ranging from 0.1 to 1 with increments of 0.1. From the loop the optimum value of 0.7 for correlation coefficient was obtained.

There are two features, namely, “OverallQual” and “GrLivArea,” with correlation coefficient (to SalePrice) higher than 0.7. Multiple Linear Regression (MLR) was used to predict the sale price of the test data. For each fold we calculated RMSE between the predicted and test data and then selected the model with the lowest RMSE and reported that. The average run time for the simple model over 10 runs was 0.063 +/- 0.007 seconds. The average RMSE score for the simple model was 0.199 +/-0.0128.

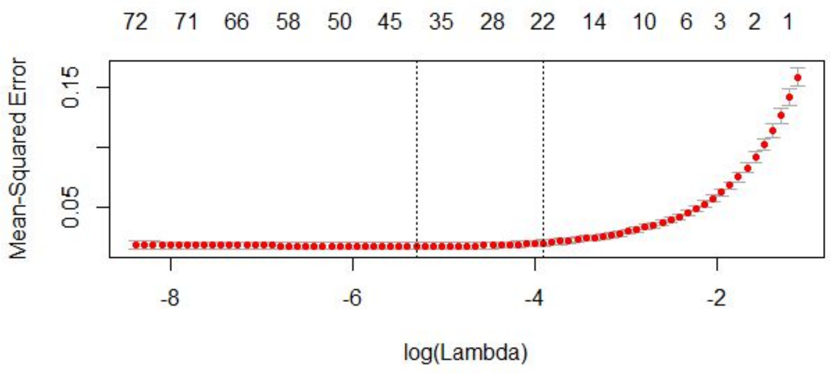
Figure 1: Correlation plots between Sale Price and High Correlation features.



B) Lasso Model (script = Lasso_DZ.R): All 75 features obtained from the data cleaning (i.e., missing value replacement and dropping features whose values are almost all missing values) were used for LASSO regularization. The R function, cv.glmnet, is used with a 10-fold cross-validation on the training data. The cross validation error is shown in Figure 1. The largest lambda such that the error is within 1 standard error of the minimum is picked for the final LASSO model for prediction. The following variables are selected by the final model:
"Condition1" , "Neighborhood", "MSZoning", "MSSubClass", "LandSlope","YearRemodAdd", "OverallQual", "LotFrontage" ,"LandContour","RoofMatl", "Condition2", "Utilities", "HouseStyle" ,"LotArea" ,"Street" ,"RoofStyle","LotConfig","OverallCond" ,"YearBuilt" ,"BldgType" ,"LotShape"

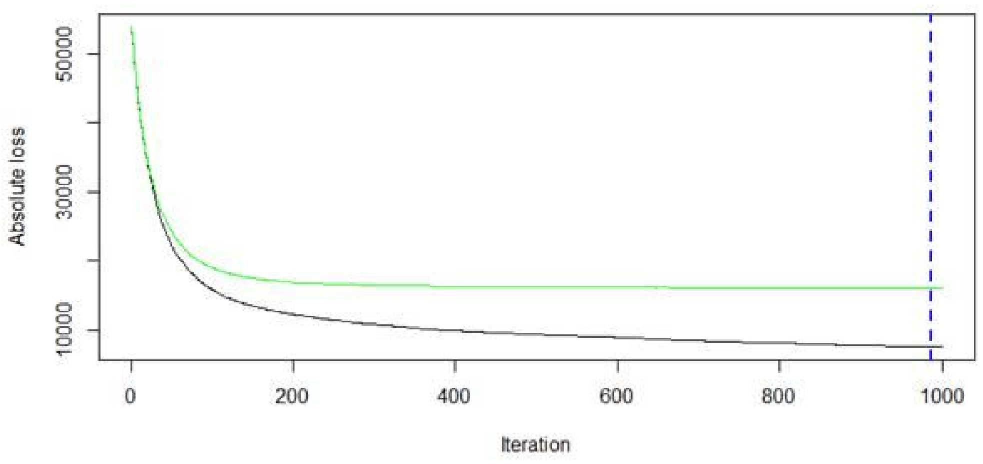
The average RMSE score for the LASSO model over 10 runs is 0.145 +/- 0.016, which is an improvement over the simple model. The average running time is 0.542 +/- 0.033 seconds.

Figure 2: Cross validation error versus the regularization factor lambda



C) Gradient Boosting Machine (GBM) (script = GBM_JL.R). GBM was performed using the gbm function present in the GBM-R package. In order to choose the number of trees 10-fold cross validation was performed where the number of trees tested ranged from 100 to 1000 with interval of 100 trees. The shrinkage, bag.fraction and cv.folds constants were selected based on the recommendations in reference [2]. For the regression problem, distribution = "laplace" is used [2]. The function "ntrees<-gbm.perf(current_model)" guides how many trees should be used. The black line (Figure 3) is the performance on the training data and the green line is out of sample performance on a held-out subset of the data. As the number of trees goes up, training performance continues to improve and the error rate continues to go down with an increasing number of trees. However,at a certain point, there is not much further improvement in performance on the held out data with an increase in the number of trees. The GBM function, which seems sensitive to even small improvements on the test data set, has indicated that the optimal number of trees to use is 1000, as indicated by the dotted blue line.

Figure 3: Absolute loss as a function of number of trees used for GBM function.



The function "summary(current_model)" showed variables sorted in order of relative importance which reveals that "Neighborhood" was the feature most useful to the model. This information can be used for later variable pruning and reruns. The average run time for this algorithm is 124.496 +/- 1.960 over 5 runs and the average Root-Mean-Squared-Error (RMSE) for the predicted versus test data is 0.128 +/- 0.034, which is an improvement over the previous two models.

Acknowledgments

[1]https://piazza-resources.s3.amazonaws.com/iwgts4iuaq41h1/iyf5s5pqnnpae/Iowa_housing_Jan27.html

[2] Kaggle by Мёнин "Prediction of House Prices (3 methods)" https://www.kaggle.com/myonin/house-prices-advanced-regression-techniques/prediction-of-house-prices-3-methods