Project 1 Predict the Housing Prices in Ames

Wenjing Du (wenjing4) *Xinyi Song (xinyis8)*

The aim of this project is to predict the sale price of residential properties by analyzing the housing data collected in Ames, Iowa between 2006 and 2010. The housing data contains 2930 observations and 81 explanatory variables describing features of residential homes. This report is divided into three parts. The first part is pre-processing the training data. In the second part, we provide technical detail of the three models we used. Finally, we evaluate the performance of the three models.

Pre-processing

Firstly, we check whether there are missing values in this data set. After checking the whole dataset, we found that all missing values in this dataset occur in the attribute "Garage_Yr_built", and there are 159 missing values in total. and we thought up with two ways to deal with it: (1) we could replace the missing values with other values such as 0. (2) we could directly remove the observations with missing values. Considering that the size of this dataset is not very big, and the observations containing missing value account for about 5% of total, we decided to replace the missing values with 0. The reason we use 0 is because when missing exactly corresponds to a level of another categorical variable (in this case, missing of "Garage_Yr_Blt" corresponds to "No_Garage" in a categorical variable "Garage_Cond"), it doesn't matter which value is used to replace the missing value.

Next, we find that some of the categorical variables only have two or three levels, and nearly all of observations fall into a specific level. In Figure 1, the bar plots show some examples of this kind of variables. These categorical variables provide little information when fitting linear models. Therefore, in order to prevent overfitting and other problems, we choose to remove these variables in our linear model.

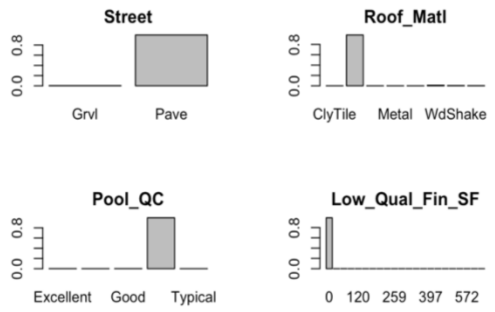

Figure 1. Bar plots of some categorical variables which should be removed

Furthermore, for categorical data in this dataset, we factorize them and then translated them into binary vector, which could be much more convenient for our analysis since it will fit more algorithms and enhance efficiency.

In addition, for linear model, we also should take outliers into consideration which could greatly affect the estimation. In Figure 2, we can see that for some numerical variables, there are observations with extreme values. Therefore, we do a 95% upper winsorization to numeric variables, that is, for each variable, we replace all values that are bigger than the 95th percentile with the 95th percentile.

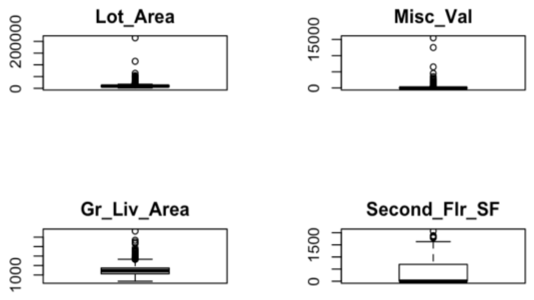

Figure 2. Boxplots of some numeric variables with outliers

Last but not the least, we decide to use logarithm of “Sale_Price” as the response variable when fitting model, and then transform it back when predicting.

Implementation

Linear Model

As for the linear model, we chose to use an elastic net model. The reason is as follows: Lasso regression can help us do model selection by shrinking coefficients to zero. However, it has some limitations. For example, when two variables are highly correlated, Lasso tends to select one variable and ignore the other. In this situation, ridge regression could be more effective by systematically reducing correlated variables together. Therefore, by applying elastic net model, we could obtain both variable selection features of lasso penalty and effective regularization characteristics of ridge regression.

In R, we apply the elastics model by using same method as applying ridge and lasso model, except setting different values to alpha. When alpha=0, we perform the ridge regression, and when alpha=1, we perform the lasso regression. When alpha takes value from 0 to 0.5, it will have a heavier ridge penalty applied, and alpha from 0.5 to 1 will apply a heavier lasso penalty. Here, in this project, we set alpha=0.5 to perform an equal combination of penalty of lasso and ridge.

Tree Model

As for the tree model, we chose to use boosting tree to analyze our data and do prediction. For boosting tree, trees are grown sequentially: each tree is grown using information from previously grown trees and each tree is fitted on a modified version of the original data set. For better performance, we select the XGBoost algorithm. This algorithm uses a more regularized model formalization to control over-fitting. We set the max number of iterations as 100, eta as 0.1, which controls the learning rate.

General Additive Model (GAM)

Generalized Additive Model (GAM) allow us to fit a non-linear function to each variable, so that we can explore information, especially non-linear relationships, that standard linear regression will miss. In this project, the number of predictors is large, and GAM has the advantage of modeling highly complex nonlinear relationships. Similar as the linear model, we first exclude some useless variables. Then we divide the remaining variables into different groups. Categorical variables remain the same, while some of the numeric variable are selected and converted into binary vectors for nonlinear transformation.

However, as we can see, the fitting of generalized additive model (GAM) is much more complex than that of net elastic model. Besides, its result is sensitive to the class of additive models available for selection and it is hard for us well-define it since data are being used to choose among a wide range of alternatives.

Performance Evaluation

Below is the table containing the root mean square error (RMSE) for each model, and the total running time for each data split. The computer system we use is MacBook Pro, 2.6 GHz, 8 GB memory.

Table I RMSE and Running time of Linear Model, Tree Model and Generalized Additive Model

Split	Liner Regression	Tree Model	GAM	Running time (mins)
1	0.1227	0.1218	0.1232	1.1492
2	0.1175	0.1326	0.1267	1.3542
3	0.1214	0.1211	0.1107	0.7447
4	0.1203	0.1227	0.1449	0.6666
5	0.1119	0.1151	0.1150	1.3484
6	0.1328	0.1362	0.1300	1.1461
7	0.1259	0.1369	0.1421	0.8702
8	0.1206	0.1346	0.1252	0.9388
9	0.1300	0.1364	0.1559	0.7901
10	0.1235	0.1331	0.1261	3.4305

Based on the table above, overall, elastic net model performs the best among these three models in terms of RMSE. We think of two possible reasons to explain this situation. For the tree model, we did not exclude variables with imbalanced classes. For GAM model, its result is sensitive to the class of additive models available for selection and to the selection method and it is hard for us do well-define the class of it available for selection, so the it might not perform as well as we expect. During the running process, we found that GAM model was quite computationally intensive and covered most of the running time.