Prediction on Iowa Housing Data

Han Huang, Yiming Tang, Zixuan Yang, Zhuangfuli Shen

Data cleaning

After observing train data for Iowa housing from Kaggle, we first check the possible unique values for continuous variables. We convert those with less than 15 unique values into factors. In addition, we convert the two variables *OverallCond* and *OverallQual* into ordered factor variables to describe the overall house condition and quality since they are scaled from 1 to 10.

Next, we remove variables with more than 1000 missing values. For continuous variables, we replace them with the corresponding median and we add a new level *NA* to store the missing values in categorical variables. Then we look at the levels of each factor variable, and drop those with a single level contains more than 1300 numbers (out of 1460 observations).

For variable *FireplaceQu*, we renamed the *NA* level into *Missing* for future convenience. Next, we use the *DataExplorer* and *data.table* packages to reconstruct the levels for factor variables. To use this package, we need to first convert data structure from data.frame to data.table, then combine all levels with frequency less than 10% (after trial and error) together as a new level, which is *other*, for every factor variable.

After we are done with reconstructing levels, we convert data structure back to data.frame since the *DataExplorer* package convert factor variables into character variables. We then convert the data type back to factors. For the continuous variables, we standardized them by the scale function in order to avoid the situation that regression coefficients are very small due to the large scale of variables. Lastly, we applied log-transformation on response variable *SalesPrice* due to its large scale.

Method I Linear Regression

After finishing the pre-processing procedure, we check the correlation of the response with the numerical features. We select the variables with relatively large correlations (larger than 0.5), which are listed as following:

[1] YearBuilt YearRemodAdd TotalBsmtSF X1stFlrSF GrLivArea GarageArea

We then fit a simple linear model with these six variables. Since we randomly split our dataset into train data and test data, the coefficients for the predictors can be different. One of our final model can be expressed as:

SalePrice = 181677 + 14946**YearBuilt* + 11126**YearRemodAdd* +14752**TotalBsmtSF*
+ 1362**X1stFlrSF* + 35641**GrLivArea* + 12408**GarageArea*

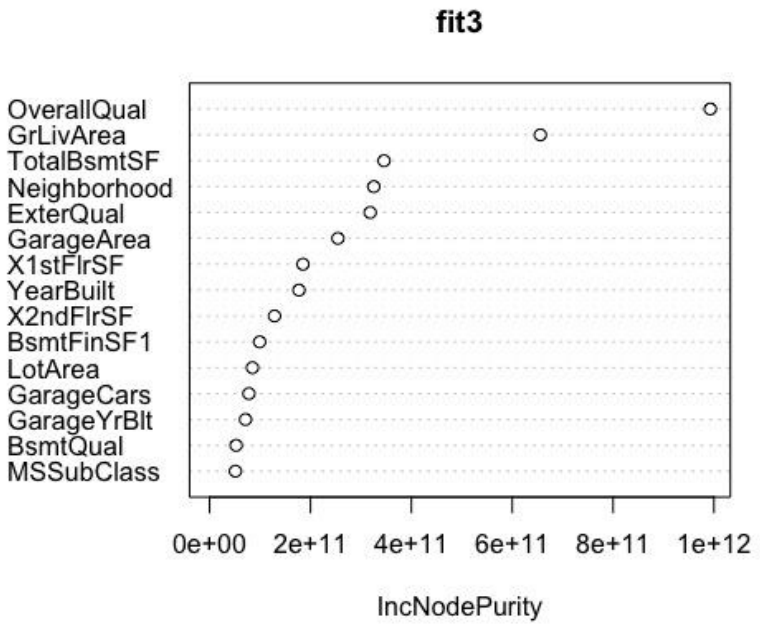
The RMSE is around 0.18. Besides, from the model, we can find that there is a positive relationship between the housing price and these features, meaning that the larger these features are, the higher price the house will be.

Method Ⅱ Random Forest

Our second approach to predict *SalePrice* is by utilizing random forest, which fits many classification regression tree models to random subsets of the input data and uses the combined result (the forest) for prediction. Besides, it also can rank the importance of variables in a natural way.

In our case, as the response *SalePrice* is a numerical variable, our model should handle with the regression problem. So we use ANOVA method and set *nodesize* as 5 in the function. When deciding number of variables randomly sampled as candidates at each split, we use p/3(round down), where p is the number of variables in the data. In function, argument *mtry* equals 20. As for tree size, we set *ntree* as 300 because this way we save more computing time and the performance of this model is not worse than models having more trees.

Inputting test data in random forest model, we find the RMSE is around 0.14, which means the prediction is pretty good. *IncNodePurity* relates to the loss function which by best splits are chosen. More useful variables achieve higher increases in node purities. The plot below shows the top 15 important variables and its corresponding values in *IncNodePurity*.



Method Ⅲ Lasso

Since the types of variables in the function *cv.glmnet* is required to be numerical, we code all categorical variables as dummy ones. After this change, the data set has 151 columns and all variables are numerical. Then, we use these variables to build a Ridge and Lasso model.

There are three ways in R to build a Ridge or Lasso model. The first one is to use function *glmnet* from package *glmnet.* The second one is to use function *cv.glmnet* from package *glmnet* and the third one is to use the train function from package caret. Since the function *cv.glmnet* runs k-fold cross-validation for *glmnet* several times which can give us the value for particular lambda such as lambda.min(value of lambda that gives minimum *cvm*) and lambda.1se(largest value of lambda such that error is within 1 standard error of the minimum). Thus, we choose to use *cv.glmnet* function. We set alpha=0 and 1 respectively which represent Ridge and Lasso. We find that the RMSE of model using Ridge method is bigger than that using Lasso, so we choose to use Lasso.

The final model has 63 predictors with different levels of categorical variables among which 8 have very small coefficients. Based on these predictors we predict the sale prices on test data and the RMSE is about 0.16.

Accuracy

We write a function to calculate the accuracy and running time of three prediction models. After running the evaluation for 5 times on different random splitting of the data, we calculate the standard deviation of five corresponding standard error and the average value of running time which are shown as follow.

Table1: Mean of RMSE for Three Models

RMSE(Linear)	RMSE(RandomForest)	RMSE(Lasso)	Running Time
0.1791868	0.1448983	0.1585580	4.7654246

Table2: Standard Deviation of Five Errors

SD(Linear)	SD(RandomForest)	RMSE(Lasso)
0.02014817	0.01736413	0.01707110