Fall 2018

IST707 Data Mining Final Project

Xinxuan Wei

**House Price Prediction**

**Introduction**

House is a pretty important staff in our life, because it not only offers us place to live in, but the whole environment will affect our daily emotion. In another word, a satisfying house is necessary for daily life. However, good house always means that high price, so I want to find what factors will affect house price mostly. In this project, I present a study for house price prediction based on analytical data from Kaggle. Analysis on house price is valuable because it can help customers get their best buy, and designers to understand what parts of houses are important for people living in.

The dataset of “House Prices: Advanced Regression Techniques” used to be a contest in Kaggle. Because the prediction variable is continuous, I need to use regression way to analyze it, but before that, I add another step that using classification to help simplify the whole task. Thus, classification and regression techniques were applied by R and python.

**Dataset Description**

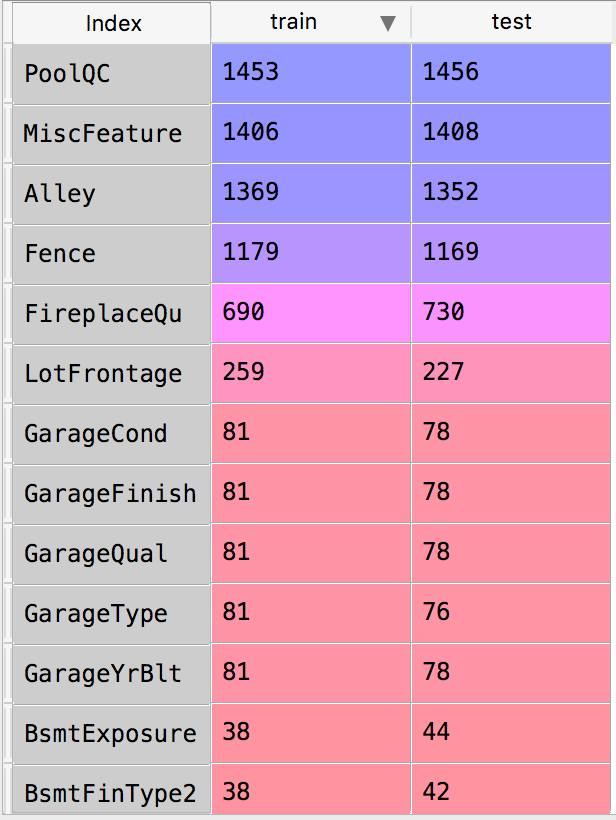
The origin datasets are from Kaggle, and already have been separated into training dataset and testing dataset. There are 80 variables and 1 prediction variable “SalePrice” in training dataset, but only 80 variables in testing data, because SalePrice” is what we need to predict and hand on Kaggle to check whether my model is good. The input variables include attribution of construction of house, outer environment of house and condition of neighbors. There are only 4 numeric variables, others are all categorical variables, including some scores from 1 to 10 to represent how owner satisfy this house. The row number of training data is 1460, and row number of testing data is 1459.

**Dataset Clean**

In Spyder, read training data in python code, and calculate missing values in each colums.

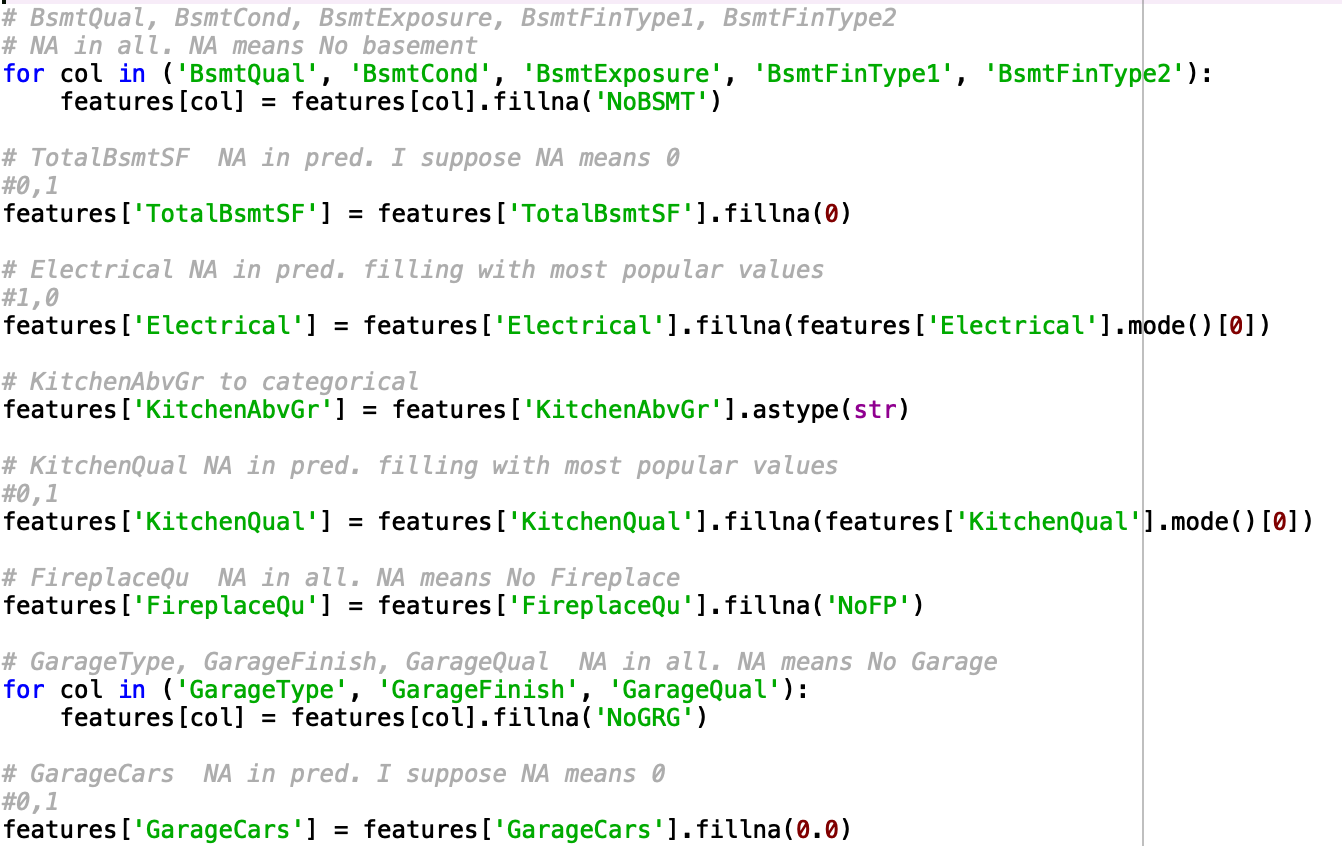


Then we can get a dataframe:



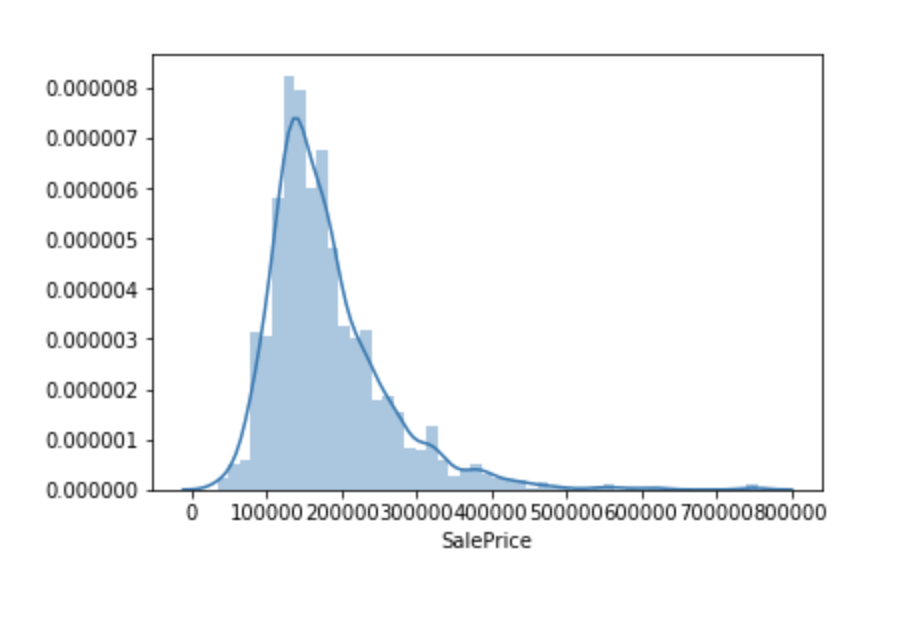
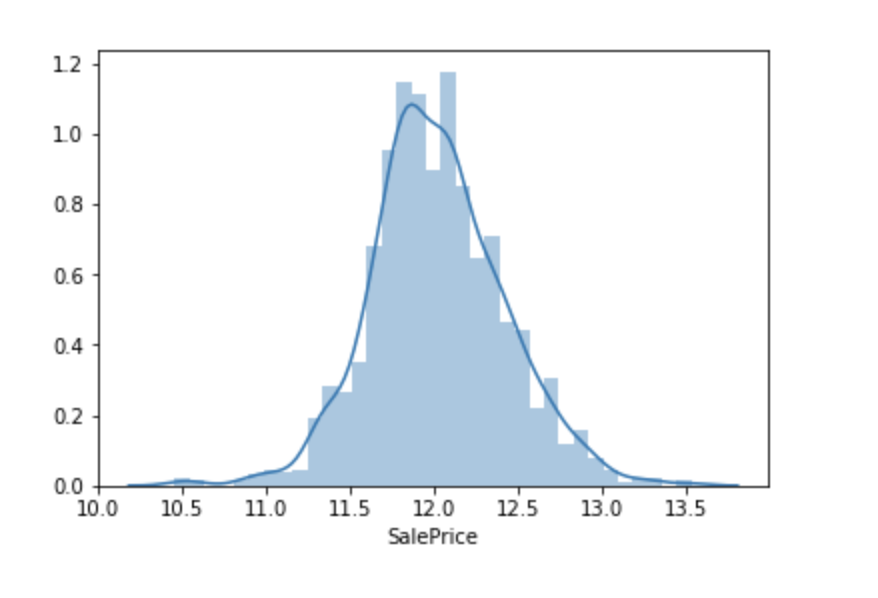
In order to not influence the result, I dropped features that have more than half of missing information or do not correlate to “SalePrice”. And I notice that there are some Nas representing 0, so I remain these kinds of Nas and set them as 0. Finally I droped these variables: ﻿'Utilities', 'RoofMatl', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'Heating', 'LowQualFinSF', 'BsmtFullBath', 'BsmtHalfBath', 'Functional', 'GarageYrBlt', 'GarageArea', 'GarageCond', 'WoodDeckSF','OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC', 'Fence', 'MiscFeature', 'MiscVal'.

For the rest Nas, I Fill them with mode or mean data of columns.

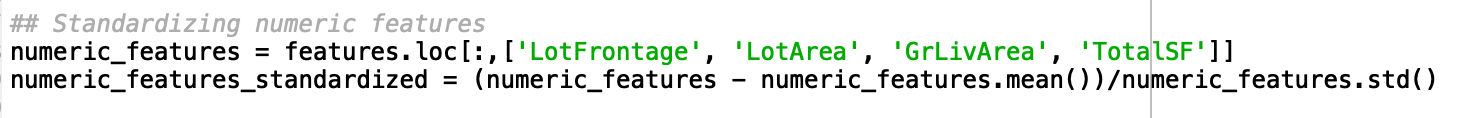


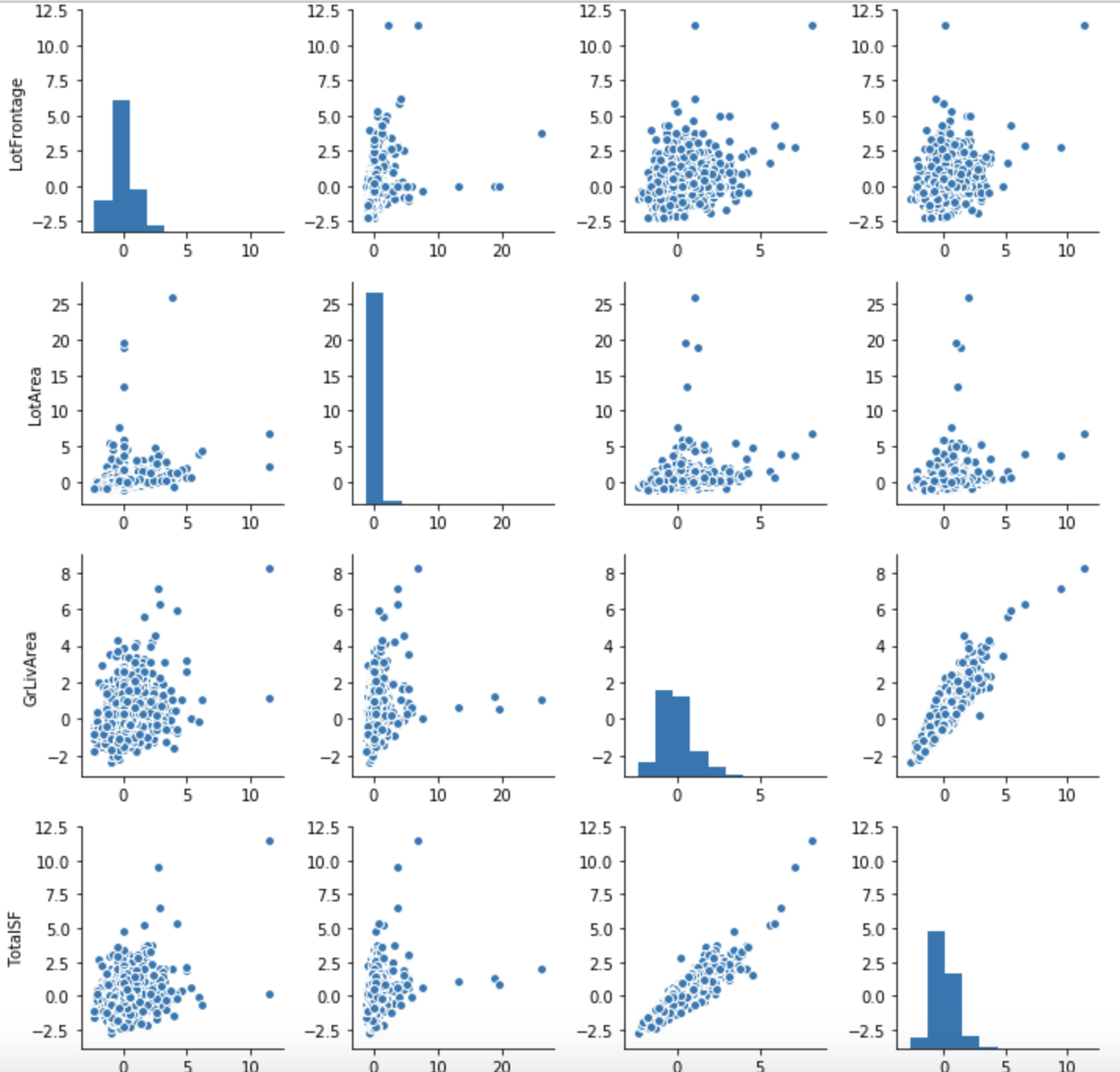
**Data Standardizing**

For variable “SalePrice”, its original distribution is as left below, which is left skewed. Thus, to make it more balanced distribution, I set ln(SalePrice) as new prediction variable, the result is as right below.



As for 4 numeric variables, ﻿'LotFrontage', 'LotArea', 'GrLivArea', 'TotalSF' , I standardize them with normal distribution:

Then, draw a pair plot to see relationships between them.

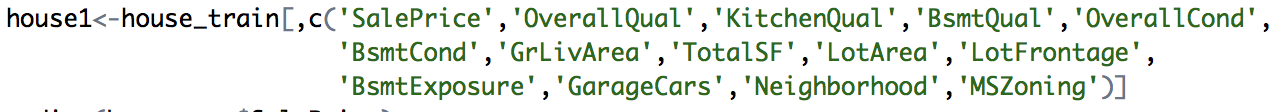


**Analysis Experiments: Classification**

Because of high ratio categorical variables over numeric variables, using classification will be easier to get result. In this way, I need to change prediction variable from continuous to discrete, with median value to separate sale price to 2 parts: “normal” and “high”.

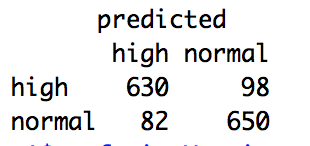
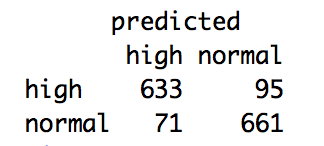
**Decision Tree:**

I use the filtered variables to build Decision Tree model in both R, and cross-validation in RWeka to check model’s results. The filtered variables are as follows:



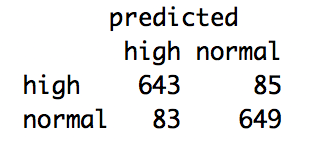
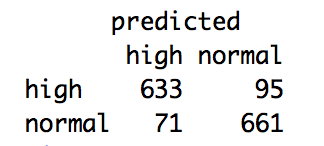
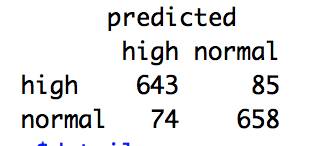
And I use 2 ways, non-cutting leaves and cutting leaves, to build decision tree models, then there are different performances:





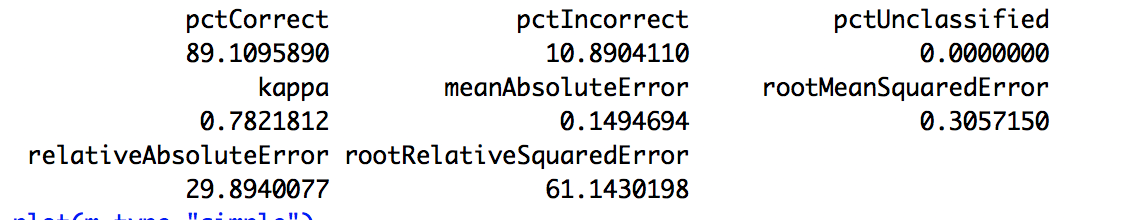
From the results, we can see cutting leaves model is better than no-cutting, which means that it is no need to include all factors in one variable. In another word, in one input variable, there are some factors much more important than others.

What’s more, when building model with deleting variable 'BsmtQual', and building model with deleting 'BsmtQual', 'LotArea', the model become better and better.



Filtered variables. Deleting 'BsmtQual'. Deleting 'BsmtQual', 'LotArea'

Through decreasing and increasing input variables, I get best decision tree model with result as follows:

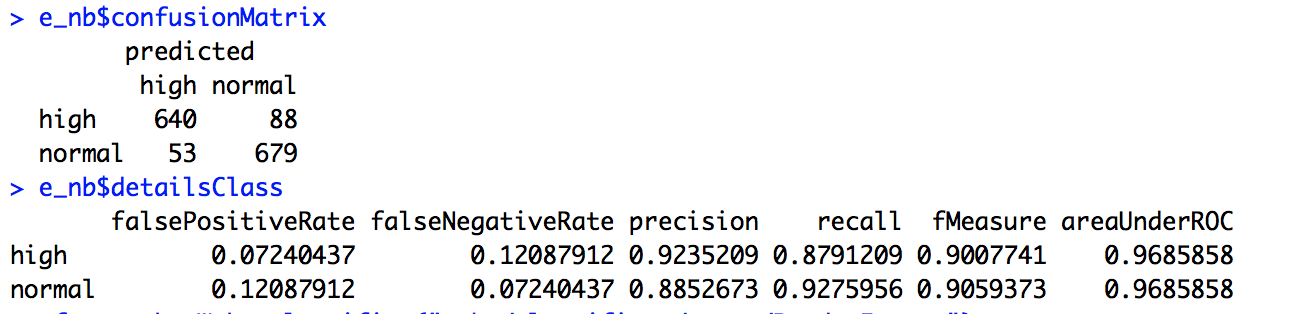


**Naive Bayes:**

The standard Naïve Bayes classifier computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule. Based on input variables filtered by decision tree, I use standard Naïve Bayes model to train the same data.

The test method we choose is the same as previous one -- cross validation and the number of folds is 10, and random seed is 1. The confusion matrix performance comparing with decision tree is slightly better, which is accuracy 90.44%, and decision tree is 89.11%. Thus, Naive Bayes model is slightly better than decision tree working on this input dataset.

The performance of confusion matrix is as follows:

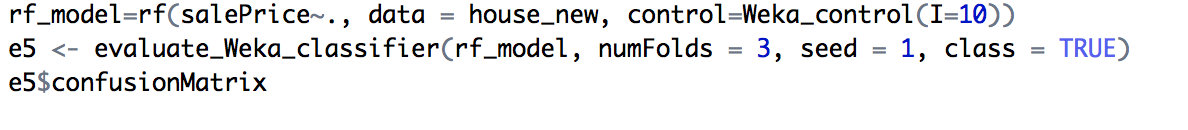


**Random Forest and SMO**

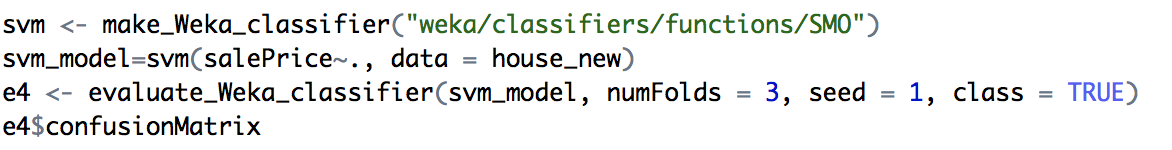
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

SMO implements John Platt's sequential minimal optimization algorithm for training a support vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. (In that case the coefficients in the output are based on the normalized data, not the original data --- this is important for interpreting the classifier.)

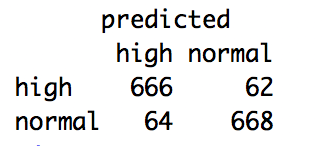
Thus, Random Forest is a bagging way based on decision tree, and SMO is a boosting way based on SVM. I both started training using the default settings in algorithm parameters. The running code is as follows.

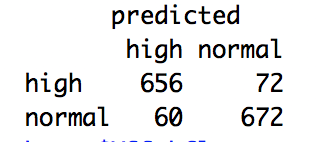


Code of random forest model



Code of SMO model

The final result is as follows:

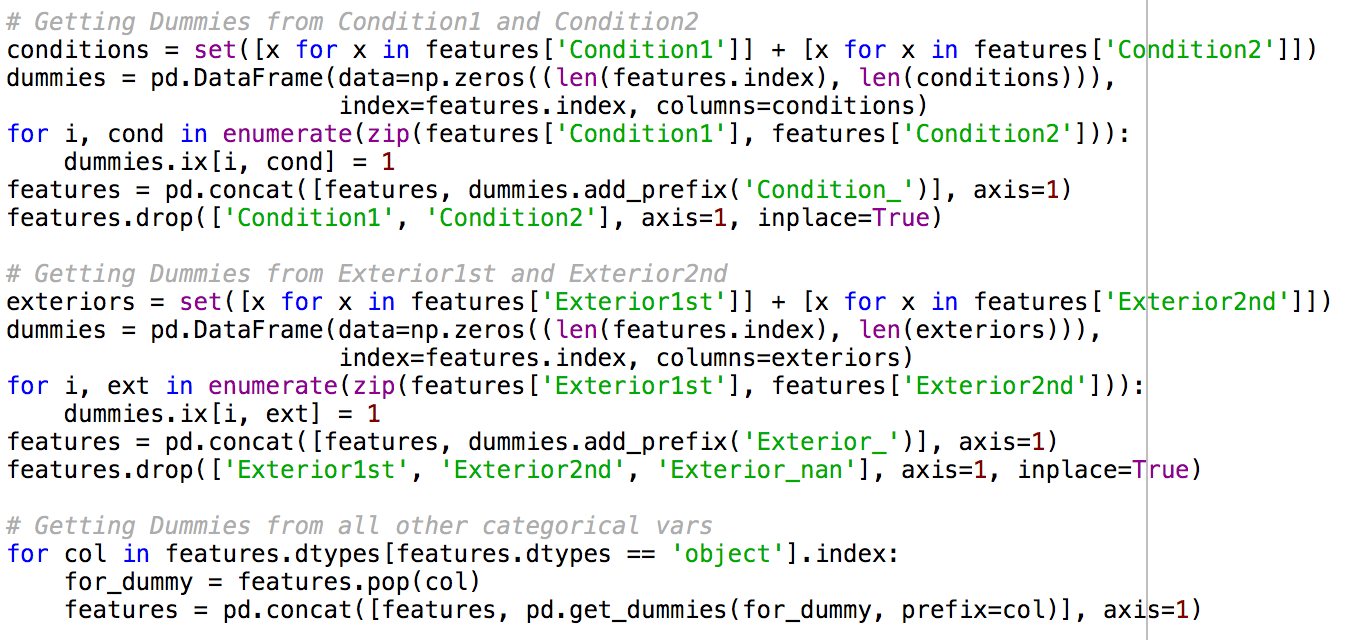


Confusion matrix of random forest Confusion matrix of SMO

**Linear Multiple Regression Model with Elastic Net**

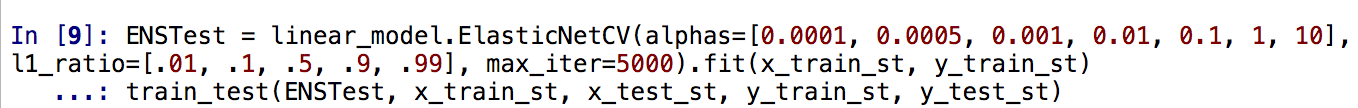
In this part, to deal with price prediction precisely, I use linear regression to combine all input variables to predict exactly price. In order to get rid of unrelated variables, I try elastic net to compress variables. In statistics and, in particular, in the fitting of linear or logistic regression models, the elastic net is a regularized regression method that linearly combines the L1 and L2 penalties of the lasso and ridge methods. Because in data processing, there will be lots of dummy variables, which means that the total input matrix is a sparse matrix. In this way, elastic net to compress unrelated variables is necessary.

**Switch categorical variables to dummy variables:**



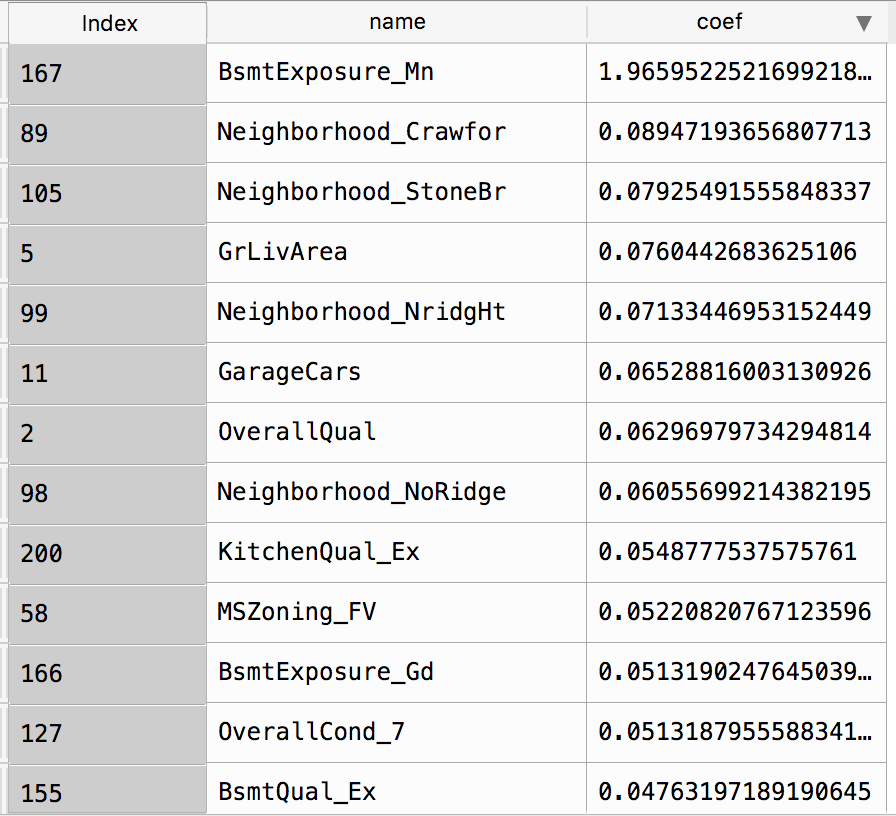
Because there are some variables such as “condition”, separating to two parts, condition1 and condition2, which means that if there is not only 1 condition, condition2 will exist. Thus, these dummy variables are a little different

**Elastic Net regression model:**



In this model, alphas mean the length of steps we choose, and l1\_ratio is punishment arguments. These two parameters are created by myself, and the model will try each of them to find which parameters have the least number of iteration times.

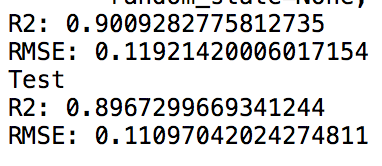
After training, we get coefficients of each input variables, and I sort them from large absolute value to small absolute value. The coefficients are as follows:



Coefficients which are larger than 0.05

From the result, we can see that there are some coefficients are compressed really small, which means that they are not important in improving the whole model. Three variables 'BsmtCond’, 'LotArea’ and 'LotFrontage’ are included in this part, which is the same conclusions as before filtering variables.

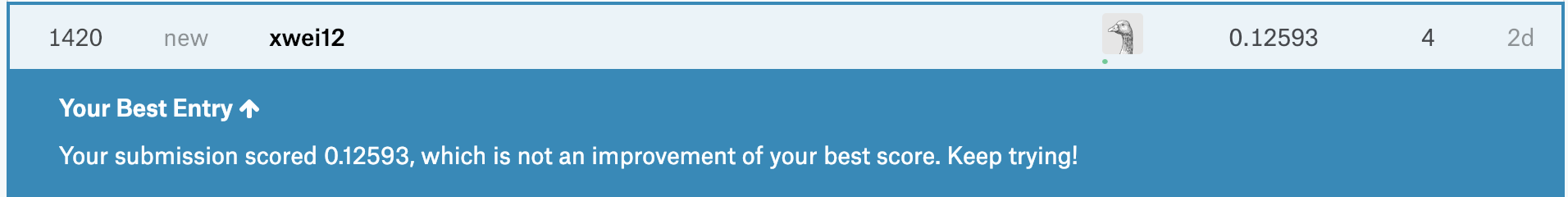
The performance on training data and validation data:



From RMSE, we can see that bias and various of this model are both good.

**Handing on Kaggle:**

Based on Elastic Net regression model, I predict house price from testing data, saving the results as csv file and making submissions on Kaggle, the final result is as follows:

****

0.12593 is public score, which represents RMSE. The final model ranks in top 30%.

**Conclusion**

According to regression results, variables 'SalePrice', 'OverallQual', 'KitchenQual', 'BsmtQual', 'OverallCond', 'BsmtCond', 'GrLivArea',' TotalSF', 'BsmtExposure', 'GarageCars', 'Neighborhood', 'MSZoning' are the most important factors influencing house price, in which, “Neighborhood” affects mostly. It means that no matter what kind of house it is, where the house is decides mostly the whole price. This conclusion makes sense because the value of districts decide the whole price level of houses in.

Another interesting conclusion is that lot area does not affect price, or in other word, there is no linear relationship between lot area and house price. I think reasons may be area including two dimensions, wide and long. People may like house with larger lot width but do not care much about lot long, because larger width of their house means more personal space between owner and his or her neighbors.