Prediction of Iowa Housing Data

https://www.kaggle.com/c/house-prices-advanced-regression-techniques

1. Introduction

The house price data we use is downloaded from Kaggle. It contains 1460 recent house sold records and 81 explanatory variables describing (almost) every aspect of these houses. In our data preprocessing procedure, we fill missing values, combine levels of categorical variables and do log transformation of skewed variables. Our first prediction method is linear regression, whose features are selected by AIC and BIC. Then we apply various advanced models to predict house sale prices. After evaluating their performance, we find that XGBoost and ensemble model are robust and well-performed, so we set them as our second and third prediction method.

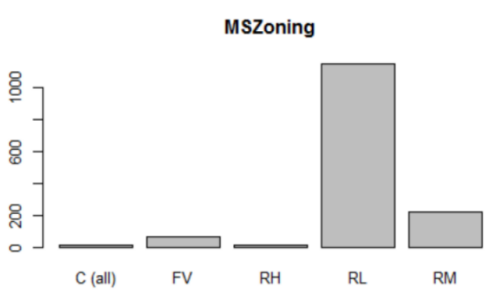
2. Preprocessing

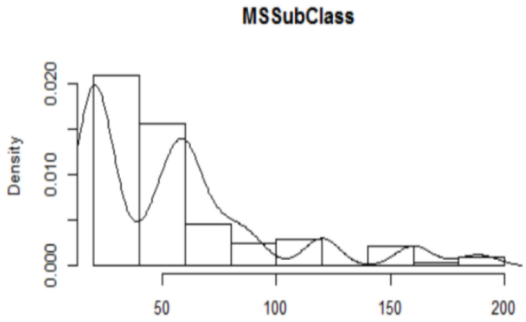
2.1 Filling Missing Values

Missing data is a significant issue in the Iowa Housing Dataset. If we simply exclude variables with missing data from the model, we leave a lot of essential information on the table. Some of the variables have missing data for almost every observation, some have small number of missing values while others have considerable number of missing values. We start by visualizing the missing data using missmap function in the Amelia package. From the map, we find that PoolQC, MiscFeature, Alley, and Fence are missing most (more than 60%) of their data, so we delete these four variables. For other categorical variables with missing values, we construct a new level called "Missing" to replace missing values. For numerical variables, we fill their missing values according to their meaning. For LotFrontage variable, we fill its missing value with its median and for the remaining numerical variables, we fill the missing values with 0.

2.2 Plotting

After filling missing value, we make plots to visualize the distributions of different variables. For categorical variables, we apply bar plots. For numerical variables, we apply histogram and density plots.





2.3 Reduction of Levels for Categorical Variables

Check the summary of each variables. It can be found that a few categorical variables have too many levels. Since about half variables are categorical variables in this dataset, combining some levels will help simplify the models and acquire more accurate predictions. Here, we combine levels with observations fewer than 5% of the total observations and classify them into a new group called "other".

2.4 Log Transformation

Linear models generally work better with data that is not highly skewed. One way to reduce skewness is by a log transform. Then, we may check the skewness of response and numerical predictors and set a threshold of skewness > 0.75. For the numeric variables whose skewness are larger than 0.75, we apply log transformation

to reduce the skewness. For example, for variable X, log_X = log (X + 1).

2.5 Applying 5-fold Cross Validation

We use 5-fold cross validation to evaluate the accuracy of our models. We first randomize the order of observations in training dataset. Then separate the reordered data into 5 equal parts. Each time we use one of the five parts of data as test dataset, with the rest data as training dataset, and get the accuracy.

3. Model Selection

3.1 Multiple Linear Regression

To develop a multiple linear regression model, we first divide the train dataset into 80% training data and 20% testing data, then train the model on training set and evaluate RMSE on testing set and compute mean of RMSE of 5 folds. We use Stepwise AIC and Stepwise BIC, forward and backward to perform variable selection and then use the selected variables to predict. This approach works well reducing the RMSE considerably. Since BIC is more appropriate in prediction, we use BIC to select our final linear model.

3.2 Regularization

3.2.1 Dummy coding

When we want to apply ridge and lasso algorithm to our datasets, we need to use the glmnet function. However, the glmnet cannot take factor directly, then we need to transform factor variables to dummy variables. There is one simple step to get dummy variables using model.matrix.

3.2.2 Ridge regression

For ridge regression, we set alpha = 0 in glmnet function and use the lambda sequence provided automatically by glmnet function to train the model. Then, we use cv.glmnet function to get the two lambda values – lambda.min and lambda.1se. Lambda.min is the value of lambda which has minimum cross validation error, lambda.1se is the largest lambda value whose cross validation error is within one standard error of CV error of lambda.min. Then we can use predict function to predict the response value of the test dataset. In the predict function, we apply ridge model and use the lambda.min to get the smallest rmse.

3.2.3 Lasso regression

For lasso regression, we set alpha = 1 in glmnet function and use the lambda sequence provided automatically by glmnet function to train the model. Similarly, we can get lambda.min and lambda.1se by cv.glmnet function. Then apply lasso model in the predict function to get the predicted values of response.

3.3 Random Forest

Using default settings for parameters, fit a random forest model to the training set.

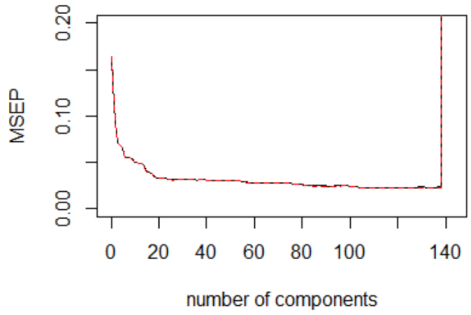
3.4 Generalized Boosted Regression

The GBM function in R has many parameters. Specify the distribution to be “gaussian”. To lessen to problem of overfitting, we assign 0.05 for “shrinkage” and 0.7 for “bag.fraction”. These values are chosen according to the performance of prediction.

3.5 Principle Component Regression

Before using PCA, we first apply dummy coding for categorical variables. Then, we fit PCR to the data and plot the CV error with the number of components. We repeat the process for 5 times with different training

and testing datasets, and find the graphs are quite similar. The following is one of the graphs. Thus, we choose 120 as the number of components for the PCR model.



3.6 XGBoost

Since XGBoost only takes numerical features, we have to code categorical variables into numeric variables. Thus, we create dummy variables for the categorical features. In order to find the optimal number of round in XGBoost, we apply 'xg.cv' to repeat cross-validation for nround times. We firstly set nround equals to 500, and find that when nround equals to 300, cross-validation test error tend to be stable, so in our final XGBoost model, we set nround as 300. As for other parameters in XGBoost function, we refer to other people's codes on Kaggle to find suitable sets of parameters and then use grid search to find the optimal parameters. The optimal choice for those parameters is max.depth equals to 10, eta equals to 0.03, nround equals to 300,gamma equals to 0.1, min_child_weight equals to 1, and objective equals to "reg:linear".

3.7 Ensemble Model

Since different models enjoy different advantages during prediction, for example, linear regression results are more smooth while random forest results are more sensitive, we consider an ensemble model which is a linear combination of different models' prediction results.

4. Result

Model \ rmse	**Round1**	**Round2**	**Round3**	**Round4**	**Round5**	**Average**	**std**
rmse_lm	0.1381	0.1267	0.1455	0.1300	0.1704	**0.1421**	**0.0156**
rmse_ridge	0.1342	0.1239	0.1490	0.1292	0.1789	**0.1430**	**0.0198**
rmse_lasso	0.1328	0.1266	0.1456	0.1298	0.1756	**0.1421**	**0.0180**
rmse_rf	0.1303	0.1246	0.1329	0.1367	0.1590	**0.1367**	**0.0118**
rmse_gbm	0.1226	0.1259	0.1371	0.1212	0.1479	**0.1309**	**0.0102**
rmse_PCA	0.1332	0.1274	0.1508	0.1301	0.1822	**0.1448**	**0.0204**
rmse_xgboost	0.1222	0.1266	0.1378	0.1318	0.1445	**0.1326**	**0.0079**
rmse_ensemble	0.1202	0.1146	0.1318	0.1181	0.1535	**0.1276**	**0.0142**
Time (s)	28.76	25.33	23.08	23.83	23.87	**24.97**	**2.0286**

From the results above, we finally choose linear, xgboost and ensemble models since they have better performance than other models.