PREDICT THE HOUSING PRICES IN AMES REPORT

Atom Group: Jifu Zhao (jzhao59), Jinsheng Wang (jwang278)

Nuclear, Plasma, and Radiological Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801, USA

1. INTRODUCTION

In this project, our goal is to predict the final price of a home in Ames, Iowa. The data are collected in Ames between 2006 and 2010, which contains 79 explanatory variables about the local homes. The given training dataset has 1460 records in total. And there are 43 categorical variables and 36 numerical variables.

In this project, we first explore the given training data set. Through some pre-processing methods such as one-hot-encoding and log-transformation, we finally get 287 features. Finally, we applied different models as simple linear regression, Lasso regression and xgboost on the new feature space and predict the house prices. More details are described in the following sections.

2. PRE-PROCESSING

After exploring the training data set, the first thing we noticed is that, there are some missing values for some features, a summary of missing values is shown in Table 1.

From Table 1 we can see that, there are 6 features whose missing value is more than 15% of the training set. So, our of first step of pre-processing is to drop those 6 features: LotFrontage, Alley, FireplaceQu, PoolQC, Fence and MiscFeature.

After dropping those 6 features, for other feature that has missing values, we do the following processing: for numerical value, we replace the missing value with the median of training set, which is the same for the test set. For the categorical data which has missing value, we add a new level of NA for each categorical feature that has missing features. Then, for those categorical feature, it is not a good idea to directly transform them into numerical variables. A better idea is to transform them into vectors using one-hot-encoding methods.

After finishing above feature processing, our feature space expands into 287 features in total. With all of these 287 features, we can apply a lot of different models. In this project, we have tried linear regression, Ridge regression, Lasso regression, xgboost model, random forest model and GBM model. After comparing the performance and running time of each model through cross-validation, we finally choose three models: simple linear regression, Lasso regression and xgboost model.

3. METHODS

3.1. Linear Regression

With linear regression, we fit a simple linear model that includes all the variables. During our experiment, we find that, linear regression model is easy and fast to implement. But the result is not as good as other advanced models.

Table 1. Summary of Missing Values

Feature Name	Data Type	# of Missing
LotFrontage	integer	259
Alley	factor	1369
MasVnrType	factor	8
MasVnrArea	integer	8
BsmtQual	factor	37
BsmtCond	factor	37
BsmtExposure	factor	38
BsmtFinType1	factor	37
BsmtFinType2	factor	38
Electrical	factor	1
FireplaceQu	factor	690
GarageType	factor	81
GarageYrBlt	integer	81
GarageFinish	factor	81
GarageQual	factor	81
GarageCond	factor	81
PoolQC	factor	1453
Fence	factor	1179
MiscFeature	factor	1406

3.2. Lasso Regression

In Lasso regression, the best choice of lambda is determined through 10-folder cross-validation and choose the lambda that minimize the cross-validation error. During our experiment, we noticed that, Lasso model perform very good.

3.3. Random Forest Model

In random forest model, we use 500 trees and the build model work fun.

4. CODE DESCRIPTION

All of our code is contained in the file named mymain.R. There are basically three parts. On the first part, the code will check whether or not the required packages are already installed. The used package includes dummies, DAAG, randomForest, glmnet, moments. On the second part of the code, we first read the training and test data sets, then drop the useless variables and process the features to form the new feature space. On the third part, we mainly build our three model: linear regression, Lasso regression and random forest model. The built model will make predictions and the results are saved into local file system.

5. RESULTS

To test our model, we randomly choose 75% of data in train.csv as the training set, then use the rest 25% of data as the test set. This process is repeated 5 times to calculate the average running time, mean-square-error and standard deviation of MSE (MSE is calculated based on log scale). The result of 5 running is shown in Figure 1.

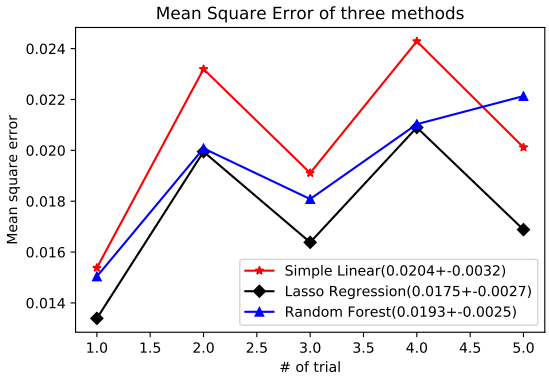

Fig. 1. Results of different models

The average running time, MSE and standard deviation is shown in Table 2.

Table 2. Summary of Models

Model	Time(s)	MSE	std
Linear Regression	1.3126	0.0204	0.0032
Lasso Regression	1.2413	0.0175	0.0027
Random Forest	25.6746	0.0193	0.0025

From Table 2, we can find that, Lasso model performs best, and linear regression model performs worst. Lasso and linear model run very fast while random forest runs very slow.

Acknowledgement

The authors would like to thank Xichen Huang for his tutorial notebook in Piazza.