STA141C Final Project
Predict House Prices with Advanced Regression Techniques

Ying-Chen Chou (ycchou@ucdavis.edu)
Wenqian Li (wqvli@ucdavis.edu)

June 12nd 2017

Introduction

This housing dataset includes 79 explanatory variables describing various aspects of residential homes in Ames, Iowa. We aim to predict the final price of each home with advanced regression methods.

We will first perform exploratory data analysis to check whether the data needs further cleaning and transformation. During exploratory data analysis, we may find more possible features by methods of feature engineering. Then we will adopt model fitting, model selection and advanced regression techniques, such as random forest and gradient boosting, to build reasonable model for predicting house price. At last, we will apply our selected model on testing data to test the model accuracy.

Data Description and Exploratory Analysis

In this project, we would like to fitting models to predict the housing price in Ames, Iowa. There are 1460 samples in the training dataset and 1459 observations in the testing data. The data includes 79 predictor variables describing several aspect of residential home. 37 variables among them are continuous while the other 42 are categorical. Our response variable **Salesprice** is a continuous variable which implies that we might need to fit regression models to find a better prediction of it. No missing values found in response variables but for predictor variables, there are several missing values. We will deal with it in the later part.

Before we do the forward analysis, considering the normality assumption of the residuals, we have to ensure our response variables are approximately symmetric. Figure 1 showed the distribution of the original Saleprice and the transformed ones. From Figure 1, it's clear that the original Saleprice is right-skewed, a transformation is needed. Here, we choose to do three type of the most common transformation method including logarithm, square root, and inverse. Among them, the log-transformation one is the most symmetric. Thus, we choose to do the log-transformation for the response variable.

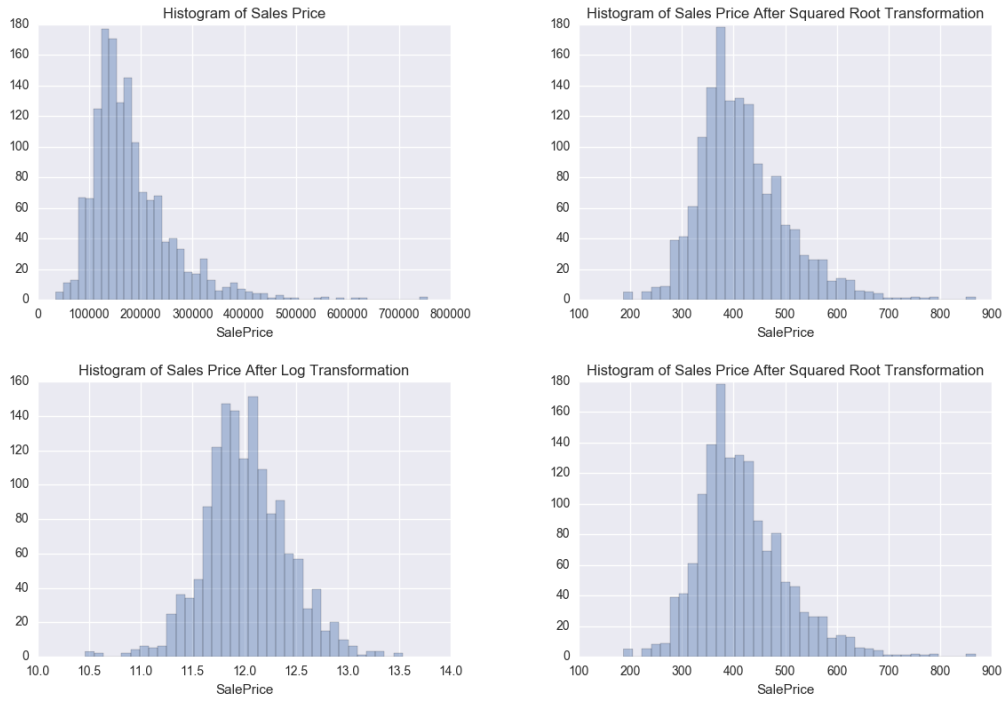

Figure 1: Histogram of Response Variable

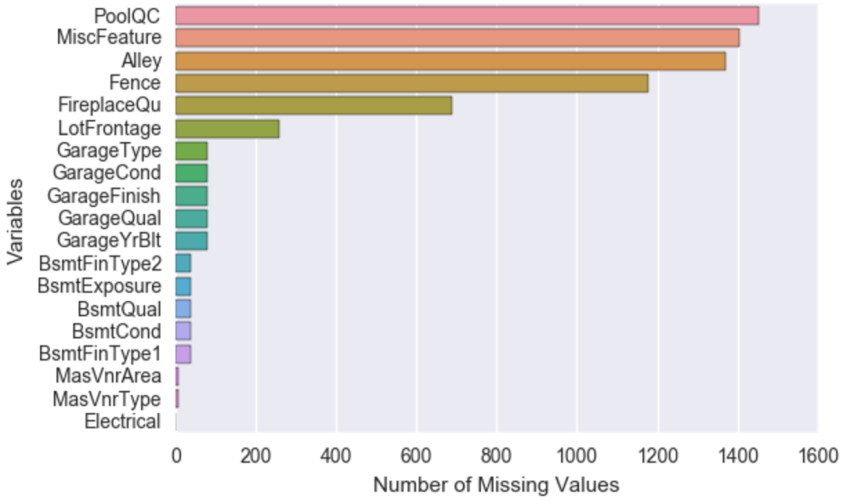

Figure 2: Barchart of Missing Values For Predictor Variables

For the predictor variables, our dataset is filled with many missing values (Figure 2). Therefore, before we can build any predictive model, it's necessary for us to fill in the missing values with appropriate values. For some categorical variables including MiscFeature, Alley, Fence, and FireplaceQu, in the data description file provided by Kaggle, the missing value represent None, which is also a category. Therefore, for these features, we use "No" to fill in the missing values. For the feature PoolQC, there are more than 99 percent of them are missing values. It's a reasonable assumption that the houses with NA's for this variables most likely do not have a pool. To confirm this assertion, we check the value of PoolArea (PoolArea: Pool area in square feet) to see if any of the houses that have a NA for PoolQC recorded a PoolArea $\neq 0$. Since all corresponding PoolArea are all zero, we can fill in the value of missing PoolQC with "No" based on our assumptions. For variable LotFrontage, this is a continuous variables, Here we will use the mean of the other values of LotFrontage to fill in the NA's. In addition, the variables relate to the information of Garage also contain lots of NA's. We take a look at all the Garage related variables, when missing

values present in variables including GarageType, GarageQual, GarageCond, GarageYrBlt and GarageFinish, the corresponding values of GarageCars and GarageArea are all zero. This implies that there is no garage in the house. Thus, for the categorical features, we use "No" to fill in the value and use 0 to fill in the NA's for continuous featrues. Also for the feature related to the basement, we also use the similar method as Garage features. Lastly, for the categorcial variable Electrical, there is no practically meaning for the missing values, therefore, we use mode which is SBrkr to replace it.

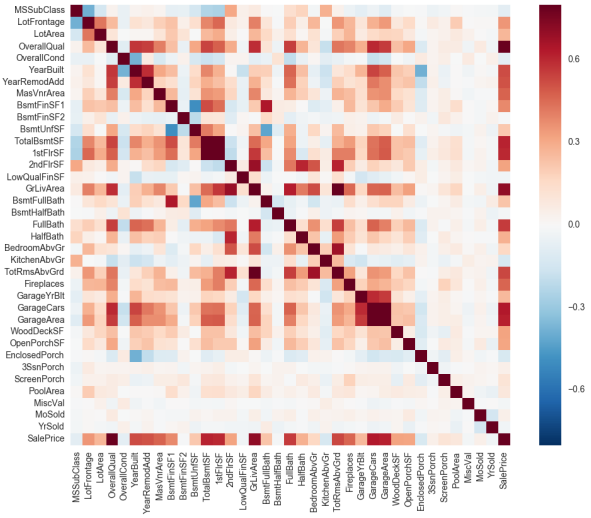

Figure 3: Heatmap of Continuous Predicor Variables

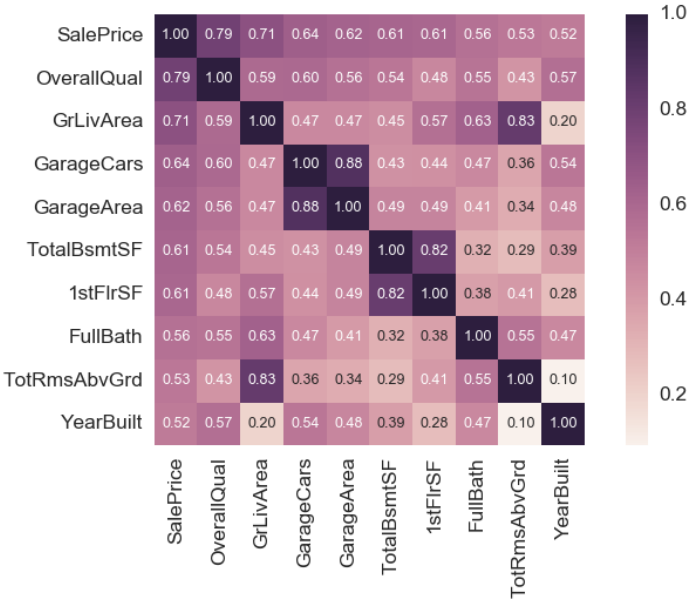

Figure 4: Zoomed Heatmap of Continuous Predicor Variables

After dealing with the missing values, we did different method for categorical variables and numerical ones separately. For categorical variables, if there are K class in the categorical variable, we will create K-1 dummy variables. For numerical variables, we next use correlation to check the relationship between them. Based on Figure 3, SalePrice is highly correlated to some features. To be more specific, Figure 4 showed the correlation between SalePrice and the top 10 X's which has higher correlation value with our response variable. From these two plots, we

can have a rough idea that SalePrice is highly correlated with OverallQual, GrLivArea, GarageCars, GarageArea, TotalBsmlSF, 1stFirSF, FullBath, TotRmsAbvGrd, and YearBuilt. Also, some features such as GarageCars and GarageArea are also highly correlated. Multicollinearity might be severe. Therefore, although some features are highly or moderate correlated to SalePrice, it might end up not including in the model because the other variables already have similar infomation provided. In the model building part, we will use some method to select the features.

Model Fitting

For predicting the SalePrice, since it's a numerical variable, it's more appropriate for us to use regression models. Here, we take models including linear regression model, ridge regression model, lasso regression model, decision tree regressor, random forest regressor, and xgboost into consideration. We will also use cross validation to prevent overfitting.

We use the sklearn module for model fitting. First of all, we fit the linear regression model and simply use all features as predictor variables and the log(SalePrice) as response variable. We do the leave-one-out cross validation and the MSE we got was 0.0397. Then we add the regularized term into the model. We try both l_1 (Lasso) and l_2 (Ridge) regularization. For these regularized linear regression model, we use leave-one-out cross validation to tune the parameters. For ridge, the candidate parameter λ includes 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1, 10, 100, 1000, and 10000. The cross validation score is shown in Figure 5. When λ is 10, we will get the MSE equal to 0.021038992601428608.

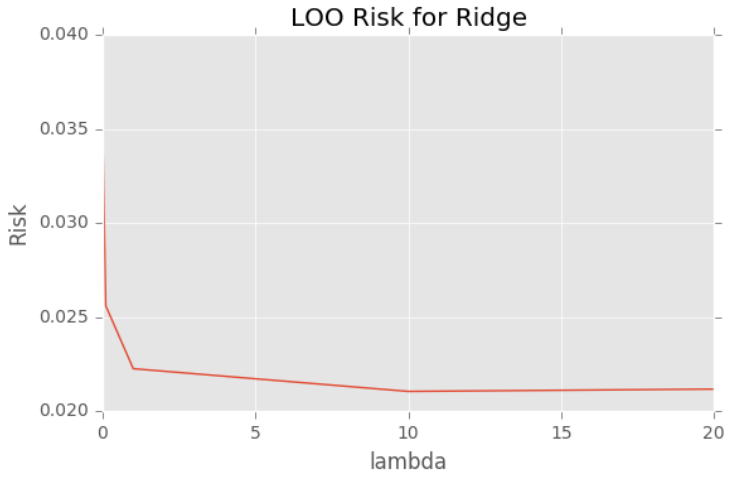

Figure 5: MSE For Different α in Ridge Regression

For Lasso, we use the LassoCV() in sklearn module, which will automatically tune the model and report the best parameters when using leave-one-out cross validation. The best parameter for lasso is 1.03 and the MSE we get is 0.039885723197330143. There is one thing about lasso is that lasso will do feature selection for us. It will set coefficients of feature which lasso deems unimportant to be zero. Therefore, after doing Lasso, we now have the subset of all features. We tried to use the feature Lasso selected to fit the linear regression with and without ridge penalty again and use the same candidate parameter. The MSE for linear regression is 0.0361 and that for model with ridge penalty is also 0.03613. The penalty term here is not so much importance and the MSE we got are almost the same.

After fitting the regularized linear model, we next fit models with regression trees including decision tree regressor and random forest regressor. For decision tree regressor, we do 10-Fold cross validation to tune the parameter and the optimal max_depth for decision tree is 10. The corresponding MSE is 0.03986. When fitting the decision tree regression model, the model will report us with the importance of each feature. In sklearn, we can easily use .feature_importance_ to find out the value of it. The higher the value is, the more important the feature are.

Through these values, we selected most importance 20 features and fit the ridge regression model again with the selected variables (Figure 6. With leave-one-out cross validation to selecting the parameter α, the optimal value of α is 10 and the corresponding MSE value is 0.02677. The result is also shown in Figure 7

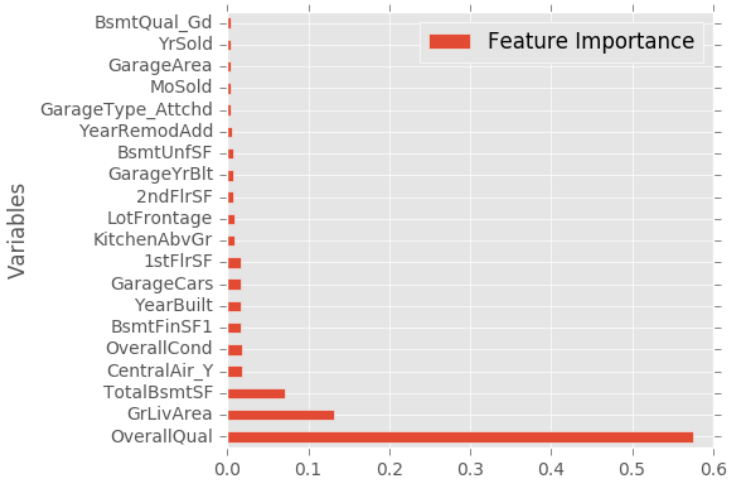

Figure 6: The Most Important Features Selected By Decision Tree Regressor

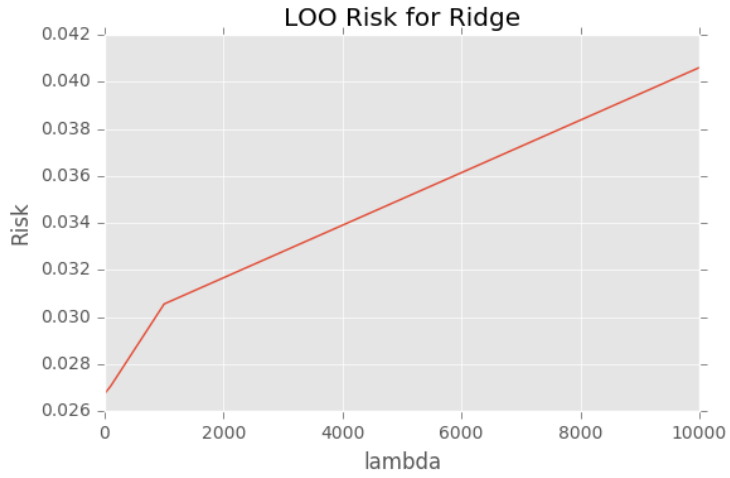

Figure 7: MSE For Different α in Ridge Regression With Features Selected From Decision Tree

For random forest regressor, we used 10-fold cross validation to tune the number of trees and max_depth. We used the *RandomForestRegressor* class from *sklearn.model_selection*. To find the best parameters, we used the method of *StratifiedKFold* and *GridSearchCV*. The best *n_estimators* is 60, and *max_depth* is 9. With these parameters, we got *MSE* in the training dataset as 0.0049060439.

One advantage of random forest method is that it can return the importance of features as *feature_importances_*. After sorting the features by importances, we plotted the top 20 features in the figure below.

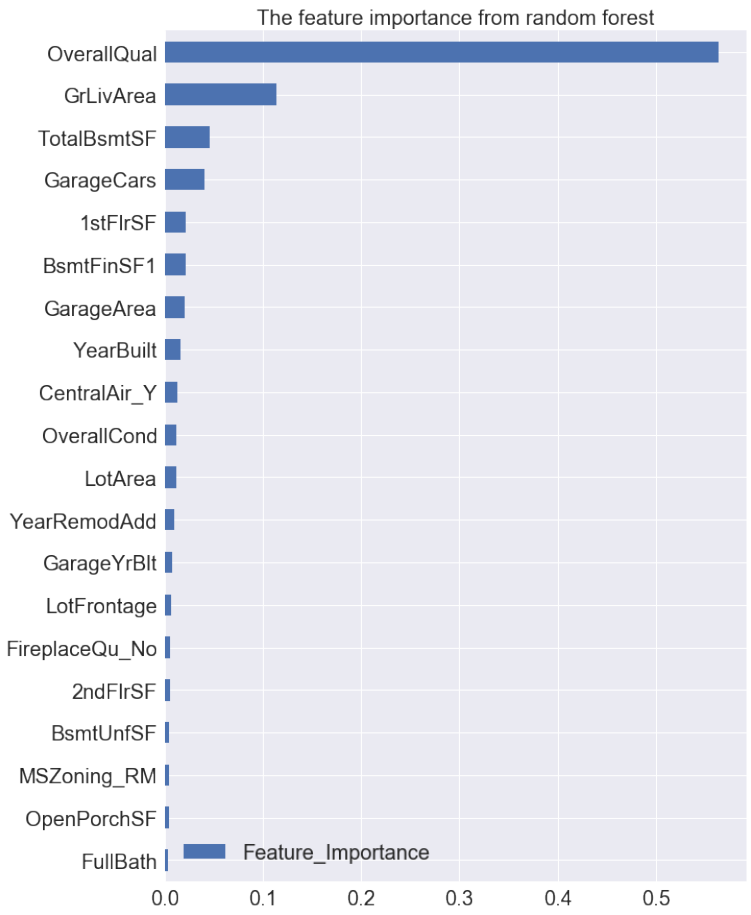

Figure 8: The most important features returned by random forest method

From Figure 8, we can see that, out of about 260 features, only 10 or so features dominate. We sub-selected 20 features and re-tuned RidgeCV parameters. The best λ is 1.0 and the corresponding *MSE* is 0.02512578761371. The leave-one-out risk is plotted as below.

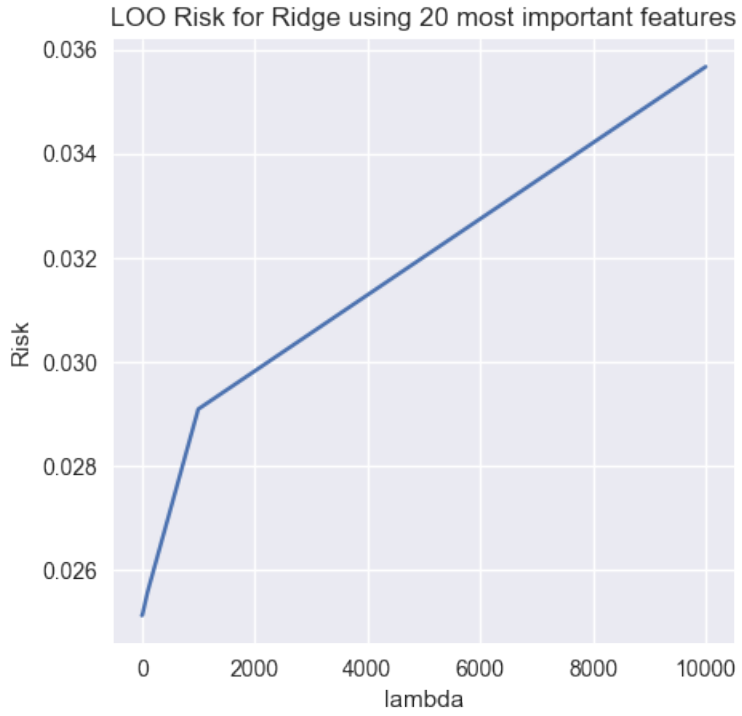

Figure 9: LOO Risk for Ridge using 20 most important features

In addition, we also adopted Xgboost Regressor to build model for predicting sale price. *XGBRegressor* class from *Xgboost* module was used. Similar as Random Forest Regressor, 10-fold cross-validation was used to find the best *n_estimators*, 90, and *max_depth*, 5. With these setup, the *MSE* of training dataset is 0.003811732139.

Model Selection and Final Model

We summarized the *MSE* of training datasets using above methods mentioned in Model Fitting section (Table 1). Random Forest and XGBoost both gave significant lower *MSE* compared with other methods.

Table 1: MSE Value For All Candidate Models

Model	MSE
Linear	0.0396
Ridge	0.02103
Lasso	0.03988
Ridge With Features Selected By Lasso	0.03613
Decision Tree	0.03986
Ridge With Top Feature Selected Through Decision Tree	0.02676
Random Forest	0.0049
Ridge With Top Feature Selected Through Random Forest	0.02512
XGBoost	0.00381

Conclusion

Considering the small difference between the *MSE* using Random Forest and XGBoost, we predicted the *log(SalePrice)* using both methods and submitted the prediction to Kaggle competition. Interestingly, Random Forest even has a lower *MSE* of the test data, which is 0.18056, while the *MSE* using XGBoost is 0.22581. Therefore, we concluded that Random Forest Regression is the best model to predict the sales price for the Ames Housing dataset.

Reference

- Data resource: Kaggle (https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data)
- Data resource: Skleran (http://scikit-learn.org/)