# Variables Screening for House Pricing

AMS561 Project

By

Kai Song, JianJie Zheng, Yujia Zhou

Spring 2017

## 1. Background

Our data sources were obtained from Kaggle -- House Prices: Advanced Regression Techniques. The test data has 79 explanatory variables describing almost aspect of residential homes in Ames, Iowa. We divide these explanatory variables into three types:

1.Numerical variables, like *'GrLivArea'* (ground living area square feet) and *'TotalBsmtSF'* (total square feet of basement area), which could be easily analyzed.

2.Ordinal categorical variables, A few features such as *'OverallQual'* and *'YearBuilt'* are belong to categorical variables but can be ordered by their levels. In the case, *'OverallQual'* is numerically labeled by 1 to 10 representing the 10 levels of overall quality from "very poor" to "very excellent".

3.Nominal categorical variables.The other type of features are also categorical but cannot be ordered since they have no such levels. Such as *'Neighborhood',* we can't order physical locations by levels or say, no order among locations.

## 2. Objective

Our initial objective was conducting a fitting model based on model selection and, we even wanted to predict the house price of train data through fitting model. But after we stucked in feature selection, we realized that it's not easy to analyze categorical variables. After researching, we found some machine learning methods, such as Lasso and Ridge, are simplified and useful for linear regression model with nominal variables. Since all group members knew little about machine learning, so we spent time on learning Lasso and Ridge and decided to focus on feature selection. Therefore, our objective is narrow to three parts: 1.Data processing 2. Multivariate study-- parameters selection (three approaches) 3.Compare with selections and our expectation.

## 3. Processing Data

### 3.1 Missing Value

First, we checked all variables with missing values. We directly deleted 6 variables with missing values taking part more than 15% of total. In the remaining cases, all of 5 features in 'Garage' family have 81

missing values, so we back to the data set in Excel and found all of these missing values from the same observations. We also noticed that these 81 missing values in 'Garage' family corresponding to zeros in GarageCar and GarageArea. So we supposed these NAs means no Garage and converted NAs into the string 'NoGar'. Similar with the case of 'Basement' and 'MasVnr' families. There are 38 observations with no basement. So we converted these NAs into 'NoBsm'. After that we only left one observation in Electronic, we directly delete this observation.

```
In [2]: missing_data.head(20)
Out[2]:
              Total   Percent
PoolQC         1453  0.995205
MiscFeature    1406  0.963014
Alley          1369  0.937671
Fence          1179  0.807534
FireplaceQu     690  0.472603
LotFrontage     259  0.177397
GarageCond       81  0.055479
GarageType       81  0.055479
GarageYrBlt      81  0.055479
GarageFinish     81  0.055479
GarageQual       81  0.055479
BsmtExposure     38  0.026027
BsmtFinType2     38  0.026027
BsmtFinType1     37  0.025342
BsmtCond         37  0.025342
BsmtQual         37  0.025342
MasVnrArea        8  0.005479
MasVnrType        8  0.005479
Electrical        1  0.000685
Utilities         0  0.000000
```

```
   ...: missing_data[missing_data.sum(axis=1) > (1460*0.15)]
Out[1]:
              Total   Percent
PoolQC         1453  0.995205
MiscFeature    1406  0.963014
Alley          1369  0.937671
Fence          1179  0.807534
FireplaceQu     690  0.472603
LotFrontage     259  0.177397
```

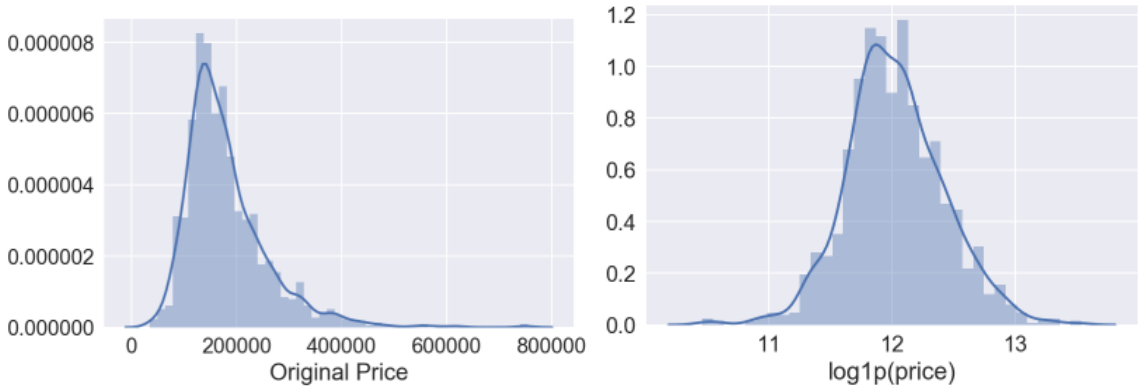
### 3.2 Check Normality

Since our project is a multivariate analysis, so we need to ensure assumptions underlying multivariate analysis are satisfied. Multivariate normality is one of assumptions that we should check before further analysis, which means not only guarantee a univariate normality (Y is normal) also ensure all numeric features normally distributed. Box-Cox (module: scipy) is useful in checking normality. It tells us which transformation will make model normal.

**Result**

Using Box-Cox to test normality of dependent variable 'SalePrice'. The score is -0.0769 near zero, so we do log transformation on Y. From distplot (module: Seaborn) of Y, you can see Y looks normal after the

transformation. Then we log transform skewed numeric features. Here we used log1p instead of log and log(1+x) because it is more accurate for small values of x.




## 4. Multivariate study-- parameters selection

**Approach 1 Correlation Matrix**

Correlation matrix tells us how strong relationship among variables. A high correlation between a feature and 'SalePrice' means this feature has a significant effect on 'SalePrice'. Moreover, we should pay attention to the high correlation among dependent variables because it will result in a situation of multicollinearity. To deal with multicollinearity, we may select one of variable and ignore others which have strong correlation. Heatmap (module: Seaborn) is an essential tool to detect this kind of situations and in problems dominated by feature selection.

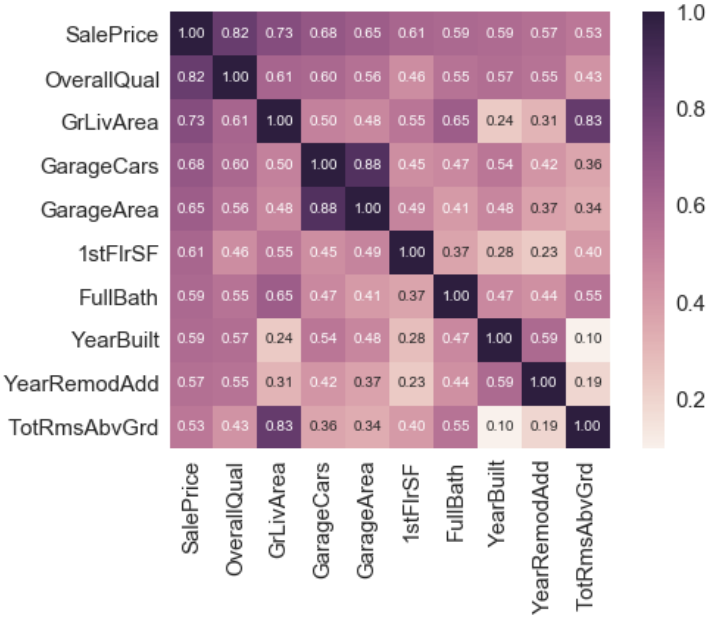
**Result**

First we checked all of correlations among features, and found there exists some strong correlations that we should pay attention. Furthermore, we used heatmap shows 9 explanatory variables with highest correlation to 'SalePrice'. We noticed that some of variables having strong correlation, so we just picked one of them, such as GarageCars and GarageArea. We selected GarageCars as it has stronger effect on SalePrice that GarageArea.

We listed all of 9 variables and compare ones have strong correlation, pick one of variable marked by(*):

OverallQual(*)
GrLiveArea(*) vs TotRmsAbvGrd vs FullBath

GarageCars(*) vs GarageArea
1stFlrSF(*)
YearBuilt(*) vs YearRemodAdd




**Approach 2 Ridge(module:scikit learn)**

Assume model is $y = f(\mathbf{z}) + \varepsilon, \quad \varepsilon \sim (0, \sigma^2)$, and the estimator can be written as $\hat{f}(\mathbf{z}) = \mathbf{z}^\top \hat{\beta}$. Least Square is the most basic method for obtaining parameters. Ridge and Lasso are based on least square method, adding one constraint that help to avoid the model overfitting. This constraint is called penalty term. We can write the ridge constraint as the following penalized residual sum of squares (PRSS):

$$\begin{aligned} PRSS(\beta)_{\ell_2} &= \sum_{i=1}^{n} (y_i - \mathbf{z}_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \\ &= (\mathbf{y} - \mathbf{Z}\beta)^\top (\mathbf{y} - \mathbf{Z}\beta) + \lambda ||\beta||_2^2 \end{aligned}$$

In PRSS equation, Z is assumed to be standardized (mean 0, unit variance) and y is assumed to be centered. This is the reason that we should normalized Y and X before regression analysis.

As usual, we need to minimize residual to get the optimal estimated parameters. So, taking derivatives, we get: $\hat{\beta}_{\lambda}^{ridge} = (\mathbf{Z}^{\top}\mathbf{Z} + \lambda \mathbf{I}_p)^{-1}\mathbf{Z}^{\top}\mathbf{y}$ .Then let it equal to zero, we obtained the parameters under Ridge method: $\frac{\partial PRSS(\beta)_{\ell_2}}{\partial \beta} = -2\mathbf{Z}^{\top}(y - \mathbf{Z}\beta) + 2\lambda\beta$ , that is related to the tuning parameter λ. The tuning parameter has following functions for selecting coefficients.

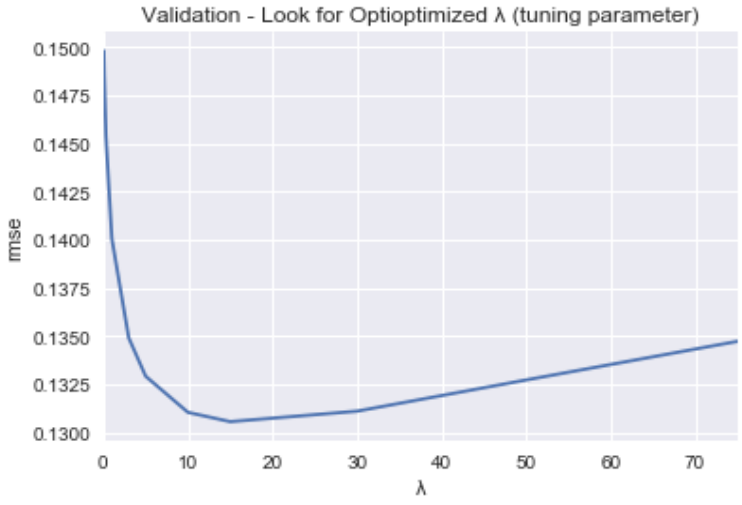
1) λ controls the size of the coefficients
2) λ controls amount of regularization
3) As $\lambda \downarrow 0$, we obtain the least squares solutions
4) As $\lambda \uparrow \infty$, we have $\hat{\beta}_{\lambda}^{ridge} = \infty = 0$ (intercept-only model)

Therefore, we can adjust the value of λ to control the number of parameters in the model.

In our code, we define a function that returns the cross-validation rmse (root mean square error) in order to compare the models with 11 different tuning parameters λ. As the properties of λ shown above, when λ increases, the number of parameters will decrease, but the model cannot capture all complexities in the data if we dismiss too many parameter. Contrarily, the model will select too many parameters that results in overfitting if λ decreases. Therefore, we tried different λ, and used rmse to represent the cross validation for each λ model and Y.

**Result**

We got the plot of validation corresponding to the value of rmse with λ increasing. The minimized rmse is 0.13055335835063617 at λ around 14.

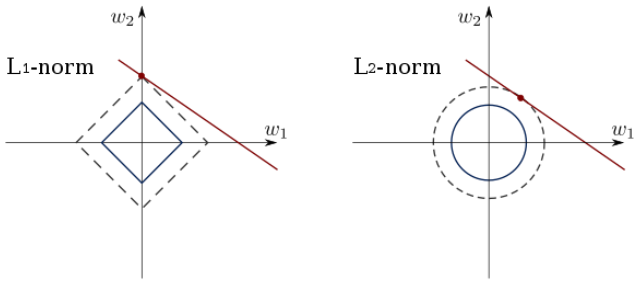


**Approach 3 : lasso(module:scikit learn)**

In lasso (least absolute shrinkage and selection operator) model, the regulation term is equivalent to **absolute value of the magnitude** of coefficients while λ is the tuning parameter that control the amount of regulation. λ works similar to that of ridge and provides a trade-off between balancing RSS and magnitude of coefficients.

$$PRSS(\beta)_{\ell_1} = \sum_{i=1}^{n}(y_i - \mathbf{z}_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$
$$= (\mathbf{y} - \mathbf{Z}\beta)^\top (\mathbf{y} - \mathbf{Z}\beta) + \lambda ||\beta||_1$$

The following picture shows the forms of the constraint regions for lasso and ridge regression.

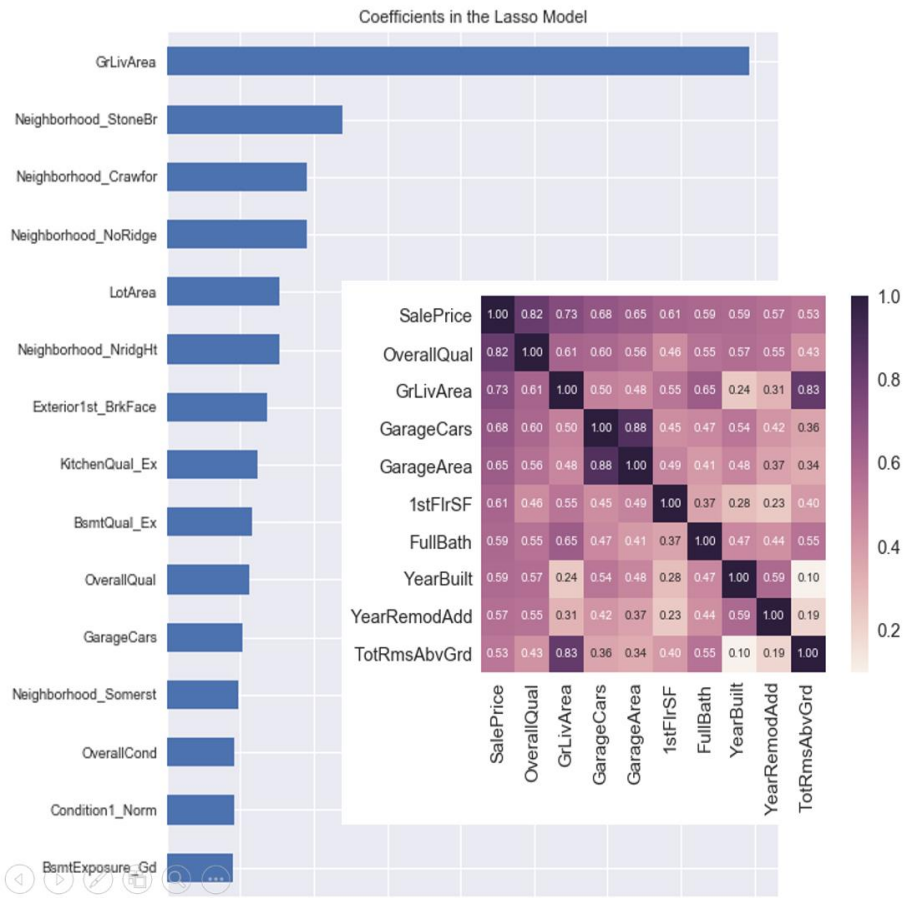



For the same values of λ, the coefficients of lasso regression are much smaller as compared to that of ridge regression. For the same λ, lasso has higher RSS (poorer fit) as compared to ridge regression Many of the coefficients are zero even for very small values of λ.

**Result**

Lasso picked 101 variables and eliminated the other 238 variables, and its rmse is 0.12746820777373982 smaller than that for Ridge. Thus, we chose Lasso as our tool for parameter selection.

As the most 15 important coefficients selected in Lasso Model shown below, Compare with the result of using correlation matrix, the big difference is Lasso selection including some categorical variables, such as Neighborhood and some ‘Quality’ features. In correlation selection, the most three significant features ‘OverallQual’, ‘GrLivArea’, ‘GarageCars’ also appear in Lasso selection. However, note that ‘OverallQual’ in Lasso method is not as same important as that in correlation selection.




## 5. Conclusion and What we learned from this project...

The three methods we used, Correlation Matrix, Lasso and Ridge, are approaches to selecting a subset of effects for a linear regression model. Correlation Matrix is very intuitively gives us the correlation between variables. Lasso and Ridge are efficient to deal with situations in which independent variables are highly correlated or their number is large relative to the sample size. We can compare Lasso and Ridge models by rmse. In our project, Lasso model provides lower rmse. Therefore, we concluded that Lasso is the best among three methods.

Comparing with the results from Lasso and Correlation Matrix. Lasso includes some nominal variables, such as ‘Neighborhood’, ‘Condition’. Additionally, the most important four features in Correlation Matrix also appear in Lasso result. However ‘OverallQual’ is not important in Lasso than that in Matrix. One thing interesting here is ‘GarageCars’ is selected by both methods, which beyond our expectation.

There are some other methods for features selection, such as stepwise, forward, backward also enables to explore all possible models and provides an optimal one. Understanding each parameter selection method helps us in model fitting.

The other thing we learned from this project is that we should understand our data before doing software analysis. Especially at the beginning when we doing data processing, we read description of features carefully and thought over which are necessary which are meaningless for analysis. Regarding dealing with missing data, we back to the original data in Excel to guess why these data missing. This guides us to dealing with missing data reasonably.

Reference:
https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python/notebook/notebook
https://www.kaggle.com/apapiu/regularized-linear-models

Theory of Lasso and Ridge: http://statweb.stanford.edu/~tibs/sta305files/Rudyregularization.pdf
Application of Lasso and Ridge:
https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/