House Price Prediction: A Comparison of Several Models

Chen Jieteng

May 24, 2019

Abstract

The problem I am trying to solve in this report is the prediction of house price based on its characteristics and features. The traditional house pricing methods are based on the discount of cash flow and comparison method. However, the approach I adopt to solve the problem is regression models. The description of dataset and the graphs can help you to have a better overview of this problem. Then I use several methods to construct the models, *Linear Regression*, *Ridge Regression*, *Lasso* and *Elastic Net*. By comparing the prediction *mean squared error* , we find that the performance of *Elastic Net* is better than others, because it works well in the situation with colinearity. Finally, I analyze and explain the feature of these models in detail.

Keywords: Linear Regression, Ridge Regression, Lasso,Elastic Net, Colinearity

Contents

1 Introduction

The real estate industry has an important impact on the development of economy. House pricing become a significant event for the national economy and the people's livelihood. However, ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. And what are the factors that effect the price of house, how to predict the price of house more accurately?

The mainstream and traditional house pricing methods are based on the discount of cash flow theoretically and comparison with the similar ones traded in our active market in practice. In this computer age, statistical machine learning has a wide range applications in many fields. If we can use statistical learning methods for predictive estiamation of house prices, people can better value the house price easily.

Now we have a data set that contains the transaction price of houses and some features of houses, such as general living areas, lot areas, year bulit and so on. We will bulid a pridiction models based on this problem. In order to get more accurate results, I tried a variety of methods, such as *linear regression*, *ridge regression*, *Lasso*, and *Elastic Net*. By analyzing and comparing the results of each model, I found that *Elastic Net* model has the smallest prediction *mean squared error*. If we were to predict house prices in practice, the *Elastic Net* model would lead to better results.

2 Description of Data Set

2.1 Variables Description

We have a data set that describes the sale of individual residential property in Ames, Iowa, America from 2006 to 2010. The data set contains 1460 observations and several explanatory variables (8 continuous,4 discrete). Most of those variables are exactly the type of information that a typical home buyer would want to know about a potential property. (e.g.When was it bulit? How big is the area? How many squares feet of living space is in the dwelling?)

The eight continuous variables related to various area dimensions are as follows:

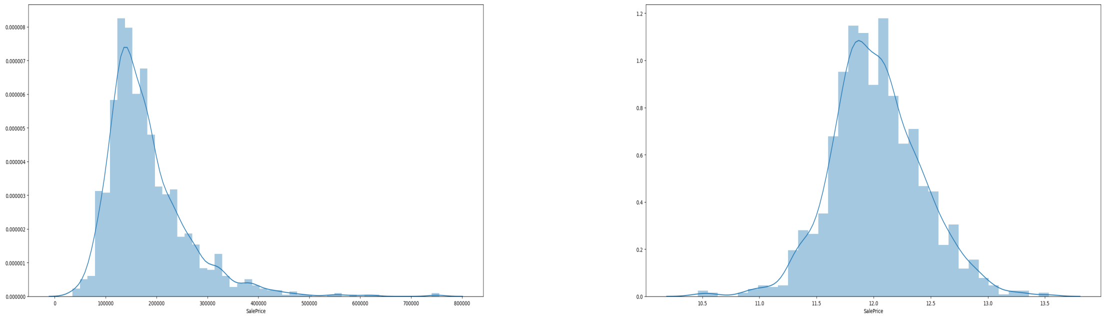
- $LotFrontage$: Straight distance from the street
- $LotArea$: Lot size in square feet
- $TotalBsmtSF$: Total square feet of basement area
- $1stFlrSF$: First floor square feet
- $2ndFlrSF$: Second floor square feet
- $GrLivArea$: Above ground living area square feet
- $GarageArea$: Size of garage in square feet
- $PoolArea$: Pool area in square feet

The four discrete variables measuring the quality are as follwos:

- $YearBuilt$: Original construction date
- $TotRmsAbvGrd$: Total rooms above ground (does not include bathrooms)
- $OverallQual$: Rates the overall material and finish of the house
- $OverallCond$: Rates the overall condition of the house

2.2 Target Variable: $SalePrice$

Our aim is to predict the $SalePrice$. After taking natural logarithm, its the distribution is more closed to normal distribution visually. We can observe it through the figures below. And we take natural logarithm of $SalePrice$ when buliding models. Such a transformation resuls in a greater amount of shrinkage of the the larger response, leading to a reduction in heteroscedasticity, which can bring better performance for our prediction models.



(a) $SalePrice$.

(b) $\log(SalePrice)$.

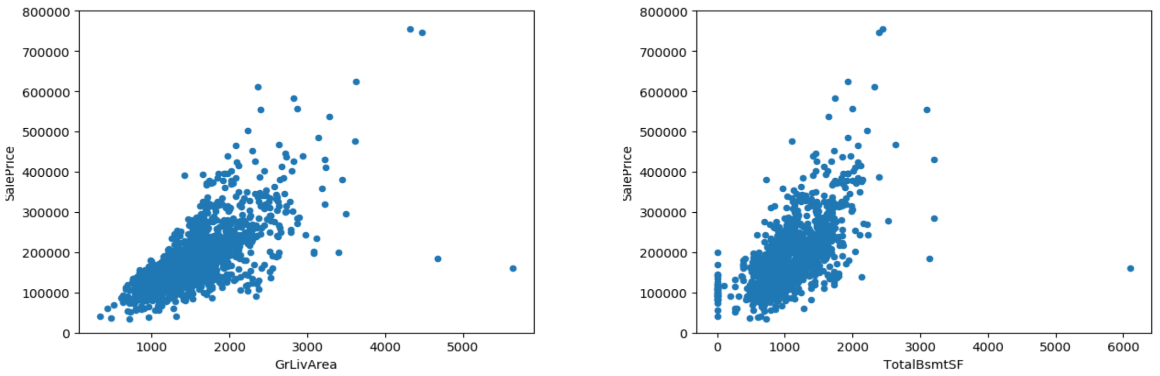
Figure 1: The frequency map of $SalePrice$.

2.3 Relationship with Continuous Variables

Through the previous exploration, I found that two numerical variables may have a greater relationship to the target variable, which are $GrLivArea$ and $TotalBsmtSF$

In this part, we plot those two numerical varibales with sale price. Through the picture below we can infer that there should be underlying relationship between $SalePrice$ and those two numerical variables. We will analyze that in the following sections.

$GrLivArea$ and $TotalBsmtSF$ seem to be linearly related with $SalePrice$. Both relationships are positive, which means that as one variable increases, the other also increases. In the case of $TotalBsmtSF$, we can see that the slope of the linear relationship is particularly high.



(a) Scatter plot of *GrLivArea* and *SalePrice*

(b) Scatter plot of *TotalBsmtSF* and *SalePrice*

Figure 2: Scatter Plot of Continuous Variables

2.4 Relationship with Discrete Variables

Through the early observation of the data, I find the two discrete variables have significant effect on sale price, which are *OverallQual* and *YearBulit*. *OverallQual* measures the quality of material of the house. From 1 to 10, it represents ”Very Poor” to ”Very Excellent”.

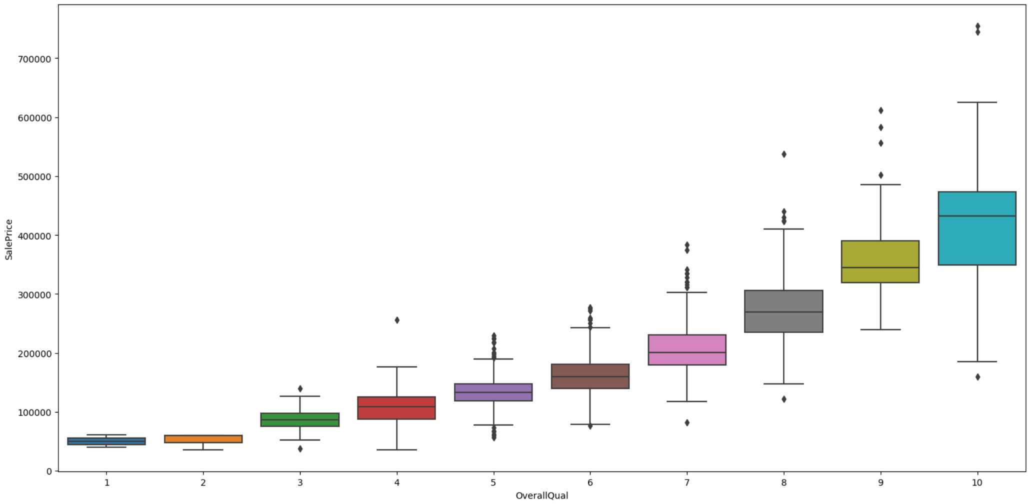


Figure 3: Boxplot with Overall Quality with Price

From left to right in next graph, the x-axis represents the year the house was built from 1872 to 2010. And the y-axis is the boxplot of price in every individual year. Althougt there is no a strong and obvious tendency, this factor still has the potential to affect the sale prices.

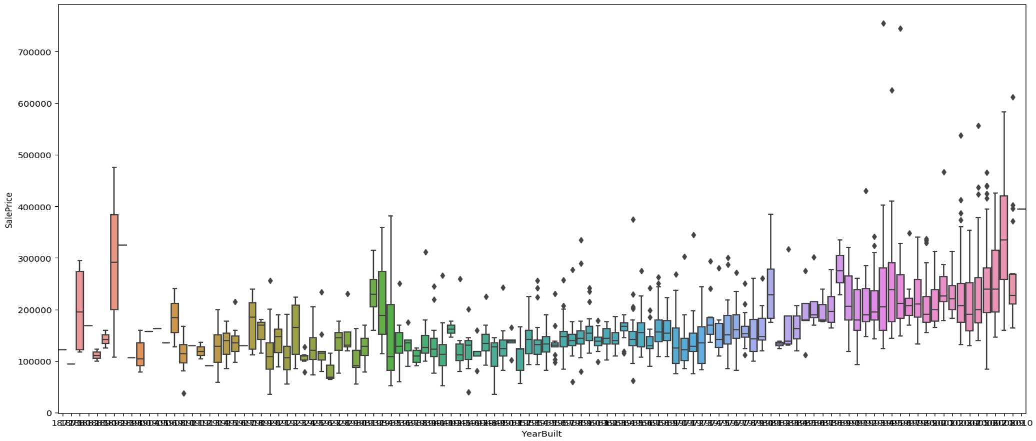


Figure 4: Boxplot of Year Bulit with Price

$OverallQual$ and $YearBuilt$ seem to be related with $SalePrice$. The relationship seems to be stronger in the case of $OverallQual$, where the box plot shows how sales prices increases with the overall quality.

2.5 Overview of Data Set

To explore the universe, we will start with some practical recipes: Correlation matrix, which is the best way to get a quick overview of the correlation between variables.

At first sight, there are two light-colored squares that get my attention. The first one refers to the ”$TotalBsmtSF$” and ”$1stFlrSF$” variables, and the second one refers to the ”$GrLivArea$” and ”$TotRmsAbvGrd$” variables. Both cases show how significant the correlation is between these variables. Actually, those correlation is so strong that it can indicate a situation of collinearity, which refers to the situation in which two or more predictor variables are closely related to one another. Thinking about these variables, we can conclude that they may give almost the same information so multicollinearity really occurs. As for the problem of collinearity, we will discuss it in detail in the next section when modelling.

Another thing that got my attention was the "*SalePrice*" correlations. We can see our well-known "*GrLivArea*", "*TotalBsmtSF*", and "*OverallQual*" indicate stronger relationship with the target variable. However other variables also should be taken into consideration.

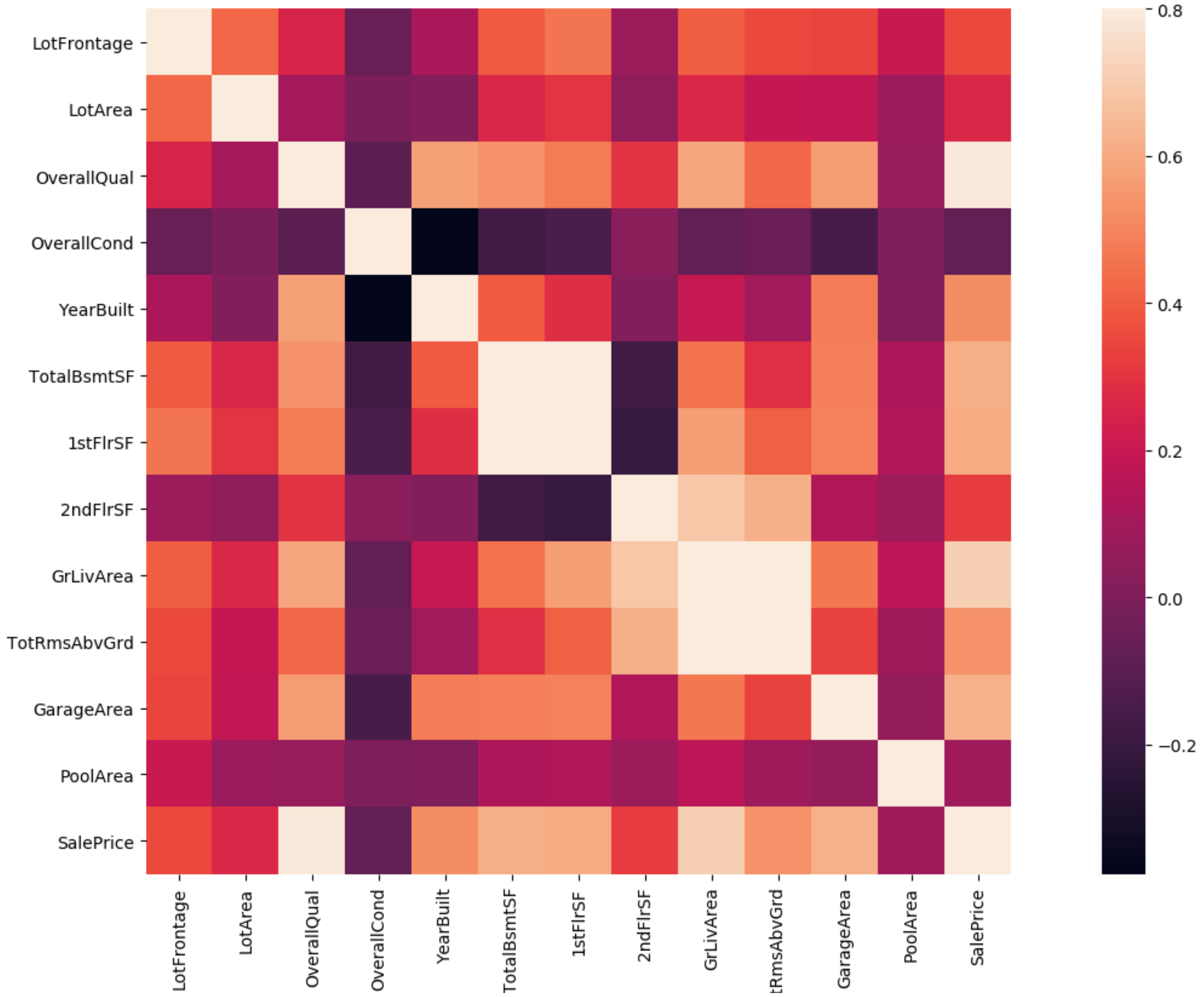


Figure 5: Heatmap of Correlation Matrix

3 Model Construction

3.1 Notation

Here, Y is the target variable $SalePrice$, and $X_1, X_2, \ldots, X_8$ represent continuous variables: $LotFrontage$, $LotArea$, $TotalBsmtFS$, $1st\ FlrSF$, $2nd\ FlrSF$, $GrLivArea$, $GarageArea$, and $PoolArea.$ $X_9, \ldots, X_{12}$ represent discrete variables : $YearBuilt$, $TotRmsAbvGrd$, $OverallQual$ and $OverallCond.$

Typically, we have data set $(x_1, y_1), \ldots, (x_N, y_N)$. Each $x_i = (x_{i1}, \ldots, x_{i12})^T$ is a vector of feature measurements in ith case.

We split the data set into two parts. One is the training set, which will be used to train models and estimate parameters. The other part is the test set, in which we can evaluate model according to its performance.

Denote by $\mathbf{X}$ the $N \times (p + 1)$ matrix with each row an input vector(with 1 in the first postion), and let $\mathbf{y}$ be the N-vector of output in the training set.

3.2 Linear Regression

A linear regression model assumes that the regression function $E(Y|X)$ is linear in the input $X_1, X_2, \ldots, X_p$. We have an input vector $X^T = (X_1, X_2, \ldots, X_{12})$ and want to predict a real-valued result. The linear regression model has the form:

$$f(X) = \beta_0 + \sum_{j=1}^{12} X_j \beta_j \tag{1}$$

We use the least squares method to estiamation, in which we pick the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_{12})^T$ to minimize the residual sum of squares

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 \tag{2}$$

Then write the residual sum-of-squares as

$$RSS(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \tag{3}$$

to obtain the solution:

$$\hat{\beta} = (\mathbf{X^T X})^{-1}\mathbf{X^T y} \tag{4}$$

The fitted values are:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T y} \tag{5}$$

3.3 Ridge Regression

Ridge Regression works better in the situations where the least squares estimates have high variance. Similar to least squares, the ridge regression coefficients estimates $\hat{\beta}^R$ are the values that minimize a penalized residual sum of square:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{12}\beta_i x_{ij})^2 + \lambda\sum_{j=1}^{12}\beta_j^2 = RSS + \lambda\sum_{j=1}^{12}\beta_j^2 \tag{6}$$

λ is the tuning parameter in *Ridge Regression*. The greater the value of λ, the greater the amount of shrinkage, then the coefficients are shrunk toward to zero. $Cross-Validation$ is a good way to select the tuning parameter. I choose a grid of λ, and compute the cross mean squared error for each value of λ. Then select the tuning parameter value for which the $Cross-Validation$ error is smallest.

To obtain the solution:

$$\hat{\beta}^R = (\mathbf{X^T X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{7}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix. And the fitted values are:

$$\hat{\mathbf{y}}^R = \mathbf{X}\hat{\beta}^R = \mathbf{X}(\mathbf{X^T X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{8}$$

3.4 Lasso

Lasso can generally select models that involve just a subset of the variables. The Lasso coefficients, $\hat{\beta}^L$, minimize the quantity:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{12}\beta_i x_{ij})^2 + \lambda\sum_{j=1}^{12}|\beta_j| = RSS + \lambda\sum_{j=1}^{12}|\beta_j| \tag{9}$$

Here I still use the Cross-Validation to set the tuning parameter λ. Some of the coefficients estimates will be exactly equal to zero if the tuning parameter λ is sufficiently large. However there is no exact form expression in Lasso. Fortunately, computer algorithm can help us to calculate the solution. I also show the code in appendix.

3.5 Elastic Net

Through the exploration and observation of the dataset previously, we found that there are strong correlation, or colinearity, among the variables. So we are trying to solve colinearity with *Elastic Net* model, which has good performance in resolving colinearity.

The *Elastic Net* penalty makes a compromise between the ridge and the lasso penalties and has the form

$$\sum_{j=1}^{p}\left(\alpha\,|\beta_j| + (1-\alpha)\beta_j^2\right) \tag{10}$$

The *Elastic Net* coefficients, minimize :

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{12}\beta_i x_{ij})^2 + \lambda\sum_{j=1}^{12}(\alpha|\beta|_j + (1-\alpha)\beta_j^2) = RSS + \lambda\sum_{j=1}^{12}(\alpha|\beta|_j + (1-\alpha)\beta_j^2) \tag{11}$$

The *Elastic Net* has an additional tuning parameter $\alpha \in [0, 1]$, that has to be determined. In practice, it can be set on subjective grounds. And we can include a grid of values of α in a $Cross - Validation$ approach when determining the tuning parameters. There are a variety of differenet computer algorithms can be used to solve this optimization problem. I also show the code in appendix.

3.6 Assessing Model Accuracy

In order to evaluate the performance of a statistical learning model on this given data set, we need to some methods to measure how well its prediction actually match the observed data. Here, we use the most commonly-used measure *mean squared error*(MSE), which is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{12}$$

The MSE will be small if the predicted reponses are pretty close to the true responses and will be large if for some of the observations, the predicted and the true responses differ substaintially.

In practice, we don't really care how well a model working on training set. Rather, we are more interested in the accuarcy of the predictions that we obatin when we apply our model to previously unseen test data. In other words, we choose the method that gives the lowest test MSE, as opposed to the lowest training MSE.

4 Results

We divide the total data into training set and test set randomly. Train the model by training set and then calculate the accuarcy on test set. Repeat this process 10000 times. And calculate the MSE of the four models every time, then use boxplot to show the MSE of those models. With this diagram, we find that the mean values MSE(green line) of four models are similar. But the upper bound of the *Elastic Net* is lower than other models, which indicates that *Elastic Net* is the good choice in this prediction problem.

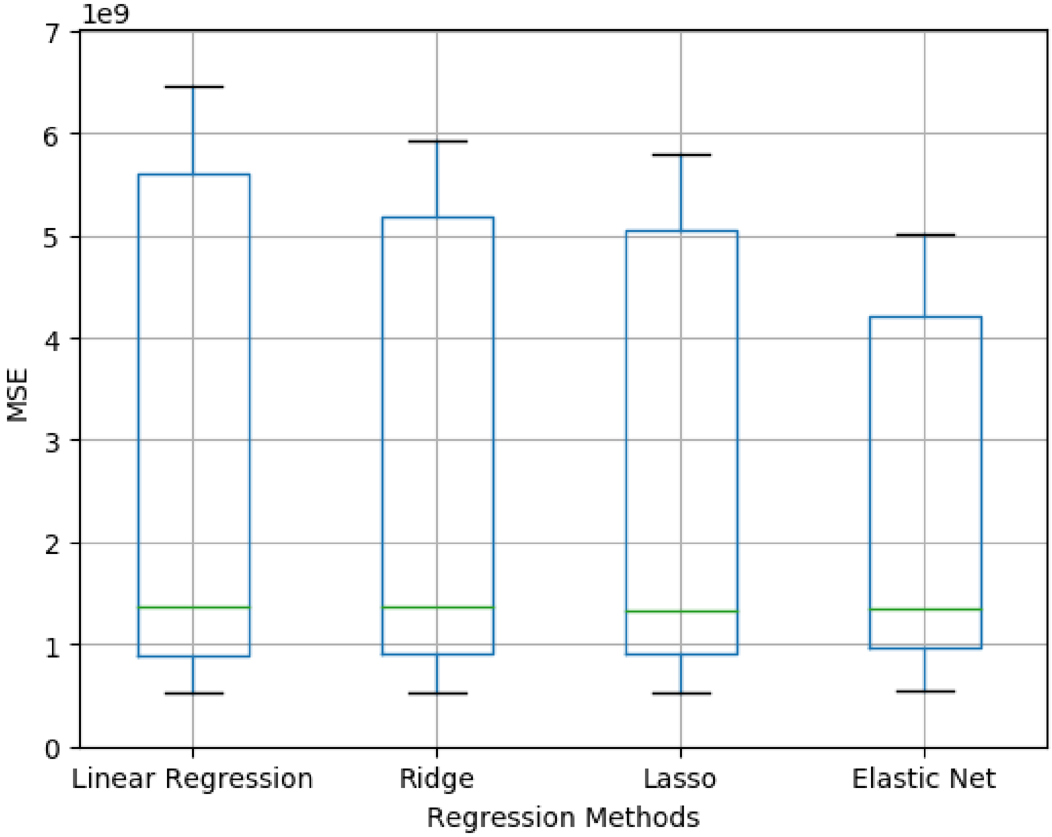


Figure 6: Boxplot of MSE of Four Models

5 Conclusion

In fact, the performance of these regression models is good. The R^2 statistics of these models on test set are all above 0.85, which can indicate a good prediction result of these models. However, the test MSE of those models are not as close as R^2. We have known that *Elastic Net* has the lowest MSE compared with other models.

Compared with *Linear Regression*, *Ridge Regression* works better in the situations where the least squares estimates have high variance. It is obvious that *Lasso* has a major advantage over ridge regression for it can produce simpler and more interpretable models that just involve a subset of the variables. But it doesn't mean *Lasso* can have a better prediction over *Ridge Regression*. In fact, neither *Ridge Regression* nor *Lasso* wil universally dominate the other. On this given dataset, they have similar result , no matter MSE or R^2.

In section two, we have found that there exists strong colinearity in this data set. However, *Linear Regression* doesn't consider this problem at all. The *Lasso* and *Ridge Regression* also can't handle the highly correlated variables very well. While the lasso penalty is indifferenet to the choice among a set of strong correlated variables. The ridge penalty tends to shrink the coefficients of correlated variables toward each other.[1] For examples, if we augment our data set with an identical copy variable $X_1 = X_1'$, they can share a coefficient in infinitely many ways, $\hat{\beta}_1 + \hat{\beta}_1' = \hat{\beta}_1$. The lasso penalty is indifferenet when choosing those two coefficients. While the ridge penalty will divide $\hat{\beta}_1$ equally between these two identical variable. Although it is impossible to meet identical variables in practice, colinearity is a very common situation.

Thinking back to *Elastic Net* penalty, $\sum_{j=1}^{p}\left(\alpha\left|\beta_j\right| + (1-\alpha)\beta_j^2\right)$, a compromise between the ridge and the lasso penalties The second term encourages highly correlated features to be averaged, while the first term encourages a sparse solution in the coefficients of these averaged features.[2] This feature allows the *Elastic Net* model to perform better under

[1] Trevor Hastie, Robert Tibshirani, Marth Wainwright. Statistical Learning with Sparsity: the Lasso and Generalizations. 2016

[2] Trevor Hastie, Robert Tibshirani, Jerome Frideman. The Elements of Statistical Learning. 2008

certain conditions, such as in some situations existing colinearity, compared with *Lasso* and *Ridge Regression*.

Back to situation of prediction of house prices, there must be colinearity within the variables. If we remove some variables directly, we will lose a part of information of the data set. That loss may lead to a higher prediction error. Thas is the reason why I don't throw away some predictors directly. Fortunately, *Elastic Net* can deal with this problem well.

In practice, we will face a more complex situation with more variables, more serious colinearity. It is often necessary to choose the right method based on the actual problem.

Appendix A. Algorithm

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pprint import pprint
from scipy.stats import skew
from sklearn.preprocessing import StandardScaler,RobustScaler,LabelEncoder
from sklearn.linear_model import RidgeCV,LinearRegression,ElasticNetCV,
                                       LassoCV
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import mean_squared_error
data=pd.read_csv("DATA.csv")
data=data.dropna()# remove the observations containing missing sort_values
# Part  1
print(data.shape)
# observe the distribution of the target variables "SalePrice"
sns.distplot(data["SalePrice"])
plt.show()
#take nutural logrithmn of "SalePrice"
sns.distplot(np.log1p(data["SalePrice"]))
plt.show()
# Relationship between continuous variables snd SalePrice
data.plot.scatter(x="GrLivArea",y="SalePrice",ylim=(0,800000))
plt.show()
data.plot.scatter(x="TotalBsmtSF",y="SalePrice",ylim=(0,800000))
plt.show()
#Relationship between discrete variables and SalePrice
sns.boxplot(x="OverallQual",y="SalePrice",data=data)
plt.show()
sns.boxplot(x="YearBuilt",y="SalePrice",data=data)
plt.show()
```

```
# the heatmap of the correlation Coefficients
corrmat=data.corr()
sns.heatmap(corrmat,vmax=0.8,square=True)
plt.show()

# Part 2
data['SalePrice']=np.log(data['SalePrice'])
#tranform into exponential, it seem more likely to normal distribution
y=data['SalePrice']
x=data.drop(['SalePrice'], axis=1)
scaler = StandardScaler()
mse_elastic=[]
mse_lr=[]
mse_lasso=[]
mse_ridge=[]
for i in range(1,1000):
    X_train,X_test,Y_train,Y_test=train_test_split(x,y,random_state=i)
    scaler.fit(X_train)  # Don't cheat - fit only on training data
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
    Y_test=np.exp(Y_test)

    lasso=LassoCV(cv=5)
    lasso.fit(X_train,Y_train)
    ridge=RidgeCV(cv=5)
    ridge.fit(X_train,Y_train)
    lr=LinearRegression().fit(X_train,Y_train)
    ElasticNet= ElasticNetCV(cv=5, random_state=0)
    ElasticNet.fit(X_train,Y_train)

    Y_pred_lasso=lasso.predict(X_test)
    Y_pred_ridge=ridge.predict(X_test)
    Y_pred_lr=lr.predict(X_test)
    Y_pred_Elastic=ElasticNet.predict(X_test)

    Y_pred_lasso=np.exp(Y_pred_lasso)
    mse_lasso.append(mean_squared_error(y_true=Y_test,y_pred=Y_pred_lasso))

    Y_pred_ridge=np.exp(Y_pred_ridge)
    mse_ridge.append(mean_squared_error(y_true=Y_test,y_pred=Y_pred_ridge))

    Y_pred_lr=np.exp(Y_pred_lr)
    mse_lr.append(mean_squared_error(y_true=Y_test,y_pred=Y_pred_lr))

    Y_pred_Elastic=np.exp(Y_pred_Elastic)
    mse_elastic.append(mean_squared_error(y_true=Y_test,y_pred=Y_pred_Elastic))

s1=pd.Series(np.array(mse_lasso))
s2=pd.Series(np.array(mse_ridge))
s3=pd.Series(np.array(mse_lr))
s4=pd.Series(np.array(mse_elastic))
data_mse=pd.DataFrame({"Linear Regression":s3,"Ridge":s2,"Lasso":s1,"Elastic Net":s4})
data_mse.boxplot()
plt.xlabel("Regression Methods")
plt.ylabel("MSE")
plt.show()
```

Appendix B. Data Source

This data has be posted in my github repositories , which can be downloaded freely.

Reference

1. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning. 2017

2. Trevor Hastie, Robert Tibshirani, Jerome Frideman. The Elements of Statistical Learning. 2008

3. Trevor Hastie, Robert Tibshirani, Marth Wainwright. Statistical Learning with Sparsity the Lasso and Generalizations. 2016