Comparison of Different Y-transformations on House Price Data

DS502 Final Project

Yufei Lin, Jingfeng Xia, Jinhong Yu, Shijing Yang, Yanze Wang

Nov 29 2020

Introduction

Traditional house pricing prediction is by comparing the cost and sale price of that specific house. By doing this, it lacks a proper process and an accepted standard. Therefore, being able to predict the price of a house by applying a machine learning model will help improve the efficiency of the real estate market and tends to be an important skill for both sellers and consumers. For the seller, they could make better sales and consumers could have better understanding when they try to make a purchase. Therefore, in this project, we are planning to make predictions of house price based on the 79 different predictors provided by Kaggle dataset ("Advanced Regression Techniques for House Prices," n.d.) to determine values of residential homes in Ames, Iowa. We are going to build several models based on this data set entirely in R and compare the results to find the model that performs the best in predicting sale price of houses.

Approaches

Model Selection

For our project, we are going to use regression analysis as our major method. Our goal is to predict the price of a certain house based on the location, type, space and other variables that can be considered in the real estate market that might affect the price of the house. There are multiple different methods to solve prediction problems. However, the reason why we chose regression as our method is because it is widely used in prediction and forecasting topics and it performs well with numeric data. Therefore, price prediction, as a numeric variable, would perform as a good fit with regression analysis.

In terms of models, we are building 5 linear regression models.(Linear Regression Model, PCR, Ridge Regression, Lasso Regression and Random Forest). We are going to compare each of these models' performance and select the one that performs the best to be our best-performance model. After doing that, we will apply the model on our test data set to see its final performance.

Y-value Transformation

We have noticed our target label SalePrice has a typical right-skewed distribution, which is against an assumption of linear regression – residuals should follow normal distribution with zero mean and equal variance (homoscedasticity) (James 2013). Since skewness is a serious issue and may be the reason for bad performance of a model, sometimes we can apply some transformations to reduce or remove Skewness. In general, square root transformation and log transformation. Log transformation is one of the most popular one and it does transform our data back to normal distribution. However, square root transformation did not do a good at transforming our data. Therefore we decided to use a new approach, which is using fourth root transformation in order to reduce the skewness. After applying log transformation and fourth-root transformation, the skewness of our data has become 0.07 and 0.49 repectively, which both fall into the category of being symmetric.

Hierachical Cross Validation

We performed hierarchical cross validation on every model we build. Due to lacking data, we split 15% of our data as our testing data and this data is put in a vault. Then we took out 30% of the rest of the data as our testing data to test the performance of each model. Each model is built on a bootstrap sample of the remaining 70% of data with cross validation. After getting the result from the 30%-percent testing data set, we will choose the one that performs the best and apply that method on 15%-percent data set to see its actual performance.

Error Metrics

We chose R square and RMSE as our error metrics to evaluate the performance of each model. Also, we created our own accuracy metric. In real world, customers tend to be satisfied when the house price is greater than its actual value since they can still bargain with the sellers; however they will be less satisfied when the predicted price is lower than the actual price. Therefore, we set our threshold to be between lower than -0.05% and greater than 20%, which means if the predicted price is within this range, we consider the predicted value as accurate.

Additonal Data

In general, all columns provided are about the information and the facts of the house itself. However, considering other factors that might affect house prices (e.g. crime rate, neighborhood information), we added new columns from other data sets to add into this data set. We ended up adding the crime index, medium income, percentage of white collar and percentage of residents with college or higher degrees for each neighborhood.

Data Processing

Description of the Dataset

Our original training data set has 1460 rows and 84 columns without any duplicated rows. There are 43 categorical predictors and 40 numeric predictors. The target label of our data set is column “SalePrice”, which makes our dataset only have one column for target labels. There are some noteworthy predictors include the location classification, utilities, environment of neighborhood, house style and condition, area, year of built, and number of functioning rooms.

Data Exploration

We first explored the distribution of our target label. As we mentioned before, our target values are right skewed, and we can also prove that by looking at Figure 1.

Target Varaible vs. Predictors

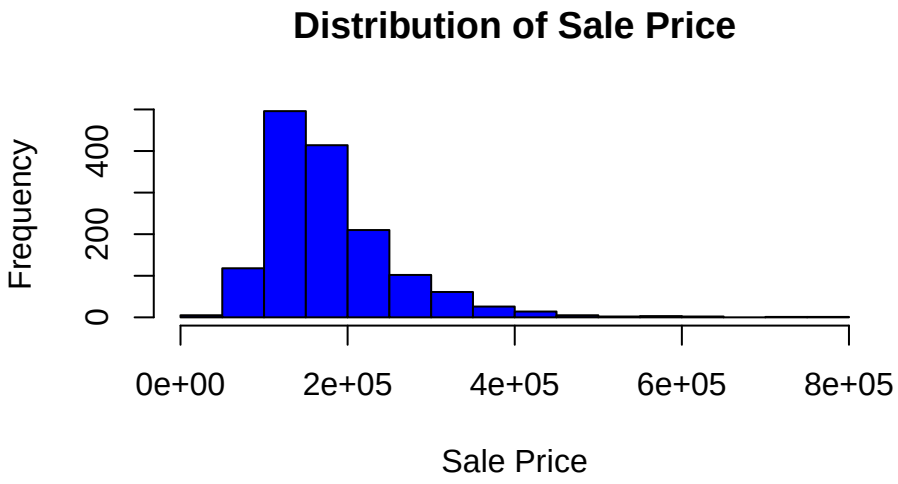

Figure 1: Distribution of Sale Price

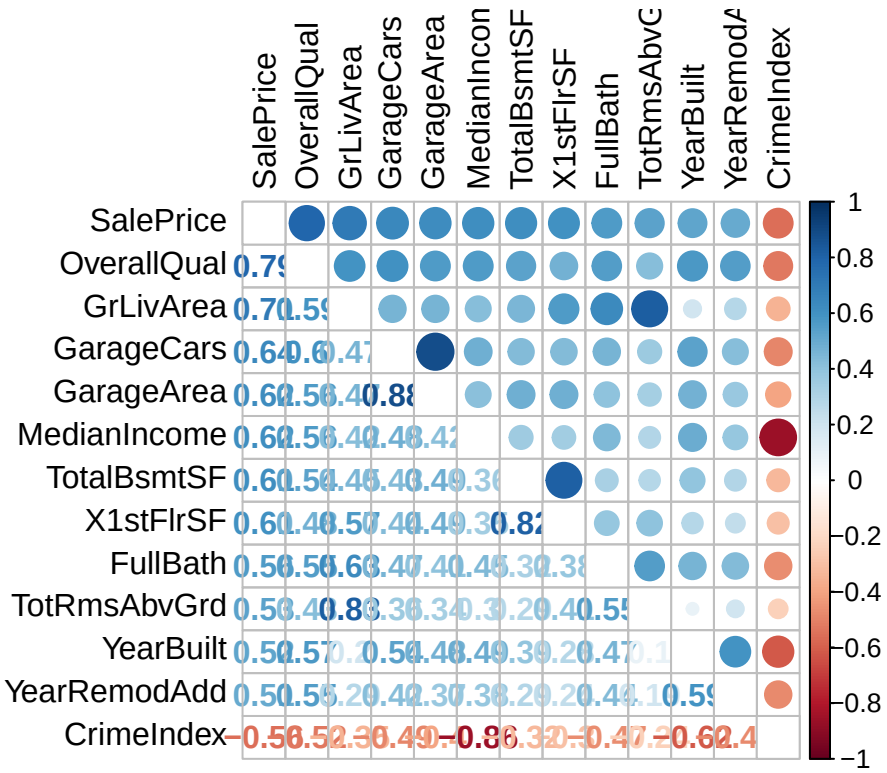

Figure 2: Numerical Data Correlation

The correlation graph above shows that the predictors that have 0.5 or greater correlation with our target variable SalePrice. By looking at the plot above, we can see that 2 out of 4 columns that are newly added seem to have descent correlation with our target variable. According to the plot, neighborhoods with higher median income residents tend to have higher price houses. This scenario totally makes sense since people with higher income tend to be able to afford a more expensive house in general. Also, crime index of a neighborhood seems to play an important role in deciding the house price of that area on average. The

higher crime index an area has, the lower of the house price it tends to have. However, it also becomes clear that there are multicollinearity issues in our data set. For example, predictor GarageCars and GarageArea are strongly correlated (0.89) as well as predictor CrimeIndex and MedianIncome (-0.8), and they are all relatively strongly correlated to the SalePrice predictor.

Overall Quality

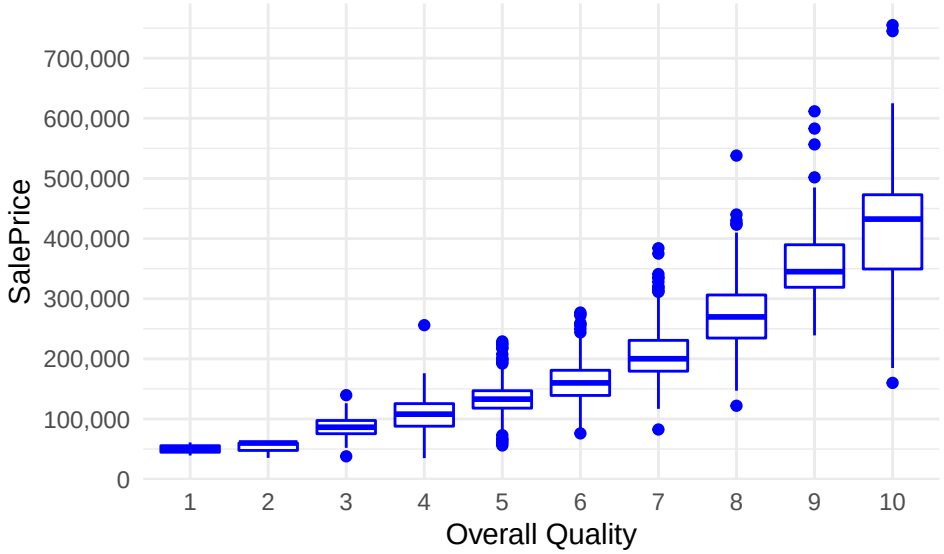

Figure 3: Overall Quality Box Plot

Overall Quality is the predictor that is the most strongly correlated with target SalesPrice. As we can see the graph above, it is clear a upward curve and proves its positive correlation with SalePrice.

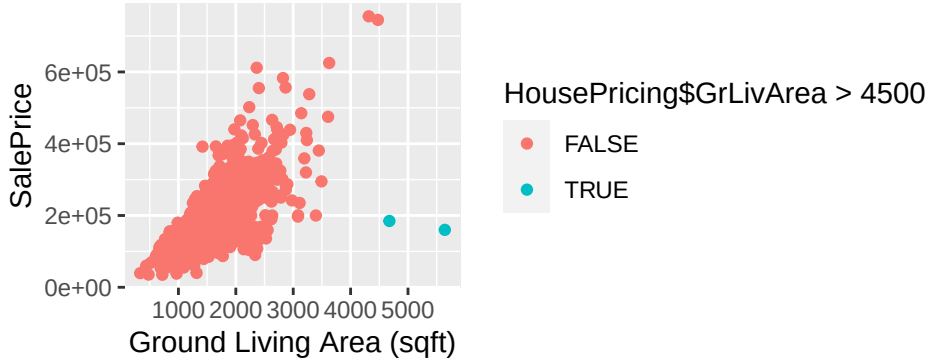

Figure 4: Living Area Scatter Plot

Table 1: Summary of Living Area

	GrLivArea	OverallQual	SalePrice
524	4676	10	184750
1299	5642	10	160000

Predictor "Ground Living Area" is the second most important numeric feature for SalePrice. We plotted a scatter plot for this predictor versus sale price. As we can see from the plot, there are two points that appear to be outliers since the living area of both houses are big but having relatively low prices. However, sometimes houses with poor quality can also lead to low prices. In order to further explore these two houses, we also plotted the overall quality of these two houses. According to the table, they are both listed as 10 in terms of quality. So we can basically make an assumption that these two points are outliers and it is relatively safe to remove these points from the data set.

Feature Engineering

Missing values

The very first thing to handle is the missing values of this data set. Overall, there are 35 columns with missing values. Usually, the rule of thumb to delete a column where there are more than 30% missing values. We have created a rule to handle numeric missing values and categorical missing values separately. For numeric missing values, we generally apply the median value of the corresponding column; for categorical missing values, we generally apply the most common category of that specific column. Also, for values that can be interpreted as none (e.g. no swimming pool), we replace those missing values to "None".

Numeric Predictors

Data normalization

After handling all missing values for numeric data, we simply apply a normalization function on our numeric columns to scale the numeric predictors, since it will help to generate a model that provides better accuracy on average.

Factorization

Some columns with numeric values are actually categorical data. We apply the corresponding explanation to replace the actual number then change them to factors.

Categorical Predictors

Ordinal Data

For categorical predictors, we apply two different ways to handle different types of them. For columns that can be interpreted as ordinal data, we scale them from 0-5.

Factorization

For other categorical predictors, we simply apply a factor function to make them become a factor within the data frame.

New Predictors

We also manually added some new predictors. For example, we combine the area of the basement and ground area of the house to become a new predictor called "TotalSqft". Figure 6 shows the relationship between new added features with our target label "SalePrice". Moreover, we use the predictor "YearSold" minus "YearBuilt" to create a predictor "Age".According to figure 6, it is obvious that the sale price of a house decreases as the age of the house increases. Also, houses that have been remodeled tend to have a higher price than those that have not, and it is the same for houses that are newer than older houses. Based on Figure 7, the new added variables such as "Age" and "TotaSqft" which have higher than 0.5 correlation with target variable "SalePrice".

Dropping columns

After doing all steps above, we then delete columns that fall into multicollinearity issue category, columns that contain over 95% same values as well as those outliers.

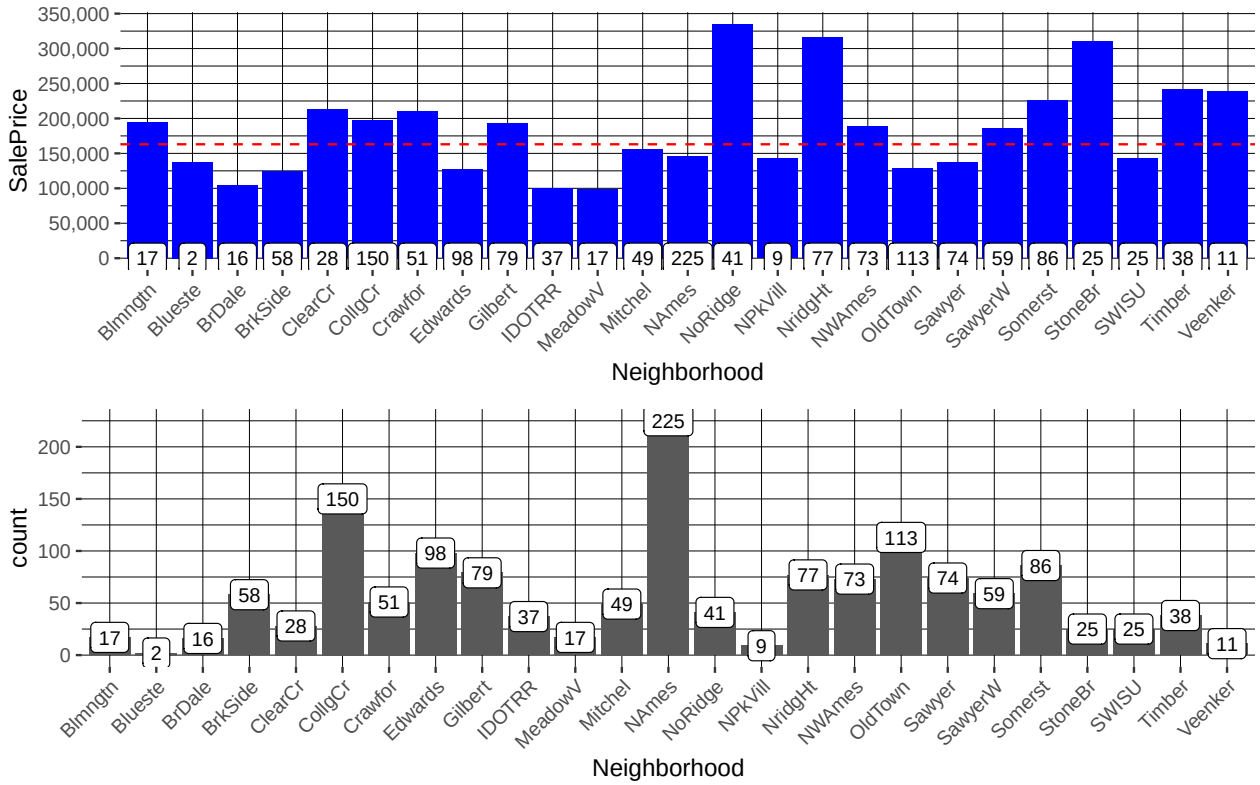

Figure 5: Neighborhood Analysis

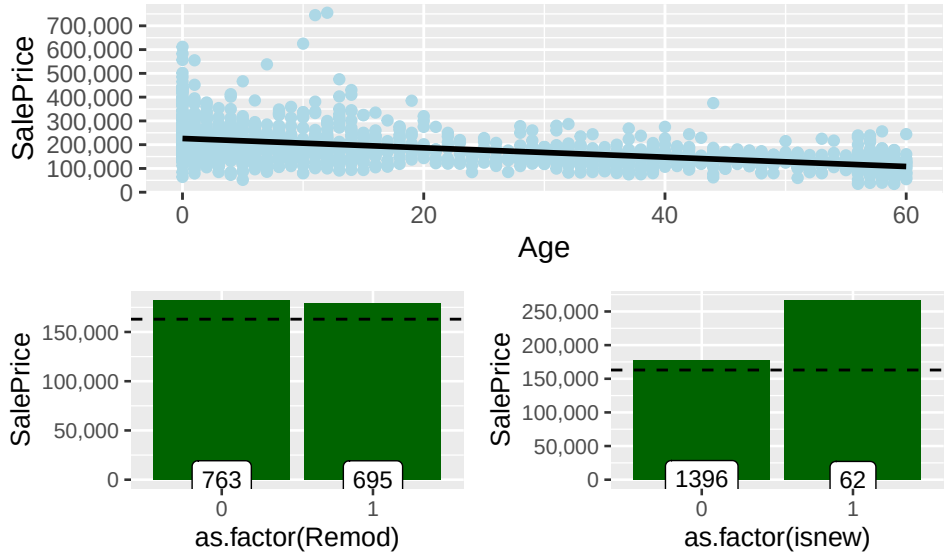

Figure 6: new predictors

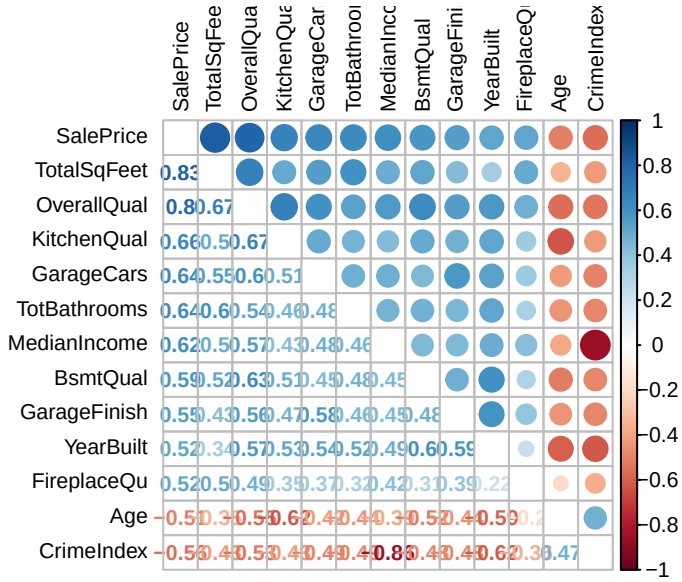

Figure 7: Correlation of numeric predictors

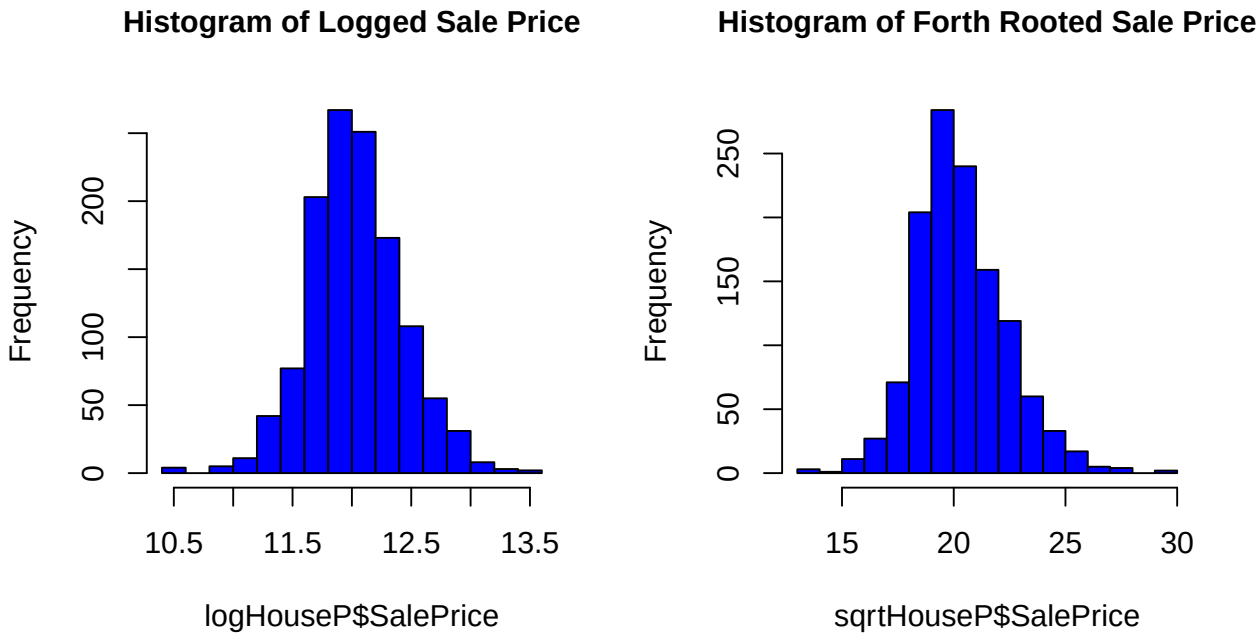

Figure 8: Histogram Comparison

Regression Methods

We choose to use linear regression, random forest, Principle Components Regression (PCR), Lasso and Ridge regression to look at how each model would be suitable for our regression analysis.

1. Linear Regression

A simple linear regression model is the first model we build. The reason why we chose this model is because we want to understand how each numeric variable is linear related to our House Price prediction. We applied

linear regression model on original y value, log's transformation y as well as fourth-root y transformation to test its performance one by one. Then we tested the model on our test data set and got the result.

2. Random Forest

Explanation

We have chosen this model because random forest is based on a collection of decision trees that could help us get better understanding of which tree and division contribute to which section such that we could have a better picture of the overall importance of each different factor in the prediction.

Prepare Model

We have 199 independent variables in the data set, therefore we have set mtry(Number of randomly selected variables for each split) to be the square root of that number for maximum performance of the model.

The following is the result from Random Forest algorithm:

1) Original Data Prediction Model

Call: randomForest(formula = SalePrice ~ ., data = train_ori, mtry = sqrt(totalIV), importance = TRUE)
Type of random forest: regression Number of trees: 500 No. of variables tried at each split: 15

Mean of squared residuals: 962687895 % Var explained: 84.01

2) Log Transformed Data Prediction Model

Call: randomForest(formula = SalePrice ~ ., data = train_log, mtry = sqrt(totalIV), importance = TRUE)
Type of random forest: regression Number of trees: 500 No. of variables tried at each split: 15

Mean of squared residuals: 0.02245887 % Var explained: 85.27

3) Forth Root Transformed Data Prediction Model

Call: randomForest(formula = SalePrice ~ ., data = train_sqrt, mtry = sqrt(totalIV), importance = TRUE)
Type of random forest: regression Number of trees: 500 No. of variables tried at each split: 15

Mean of squared residuals: 0.560824 % Var explained: 86

Variable Importance

Here we are going to show the top 10 most important variables in predicting sale price of a house.

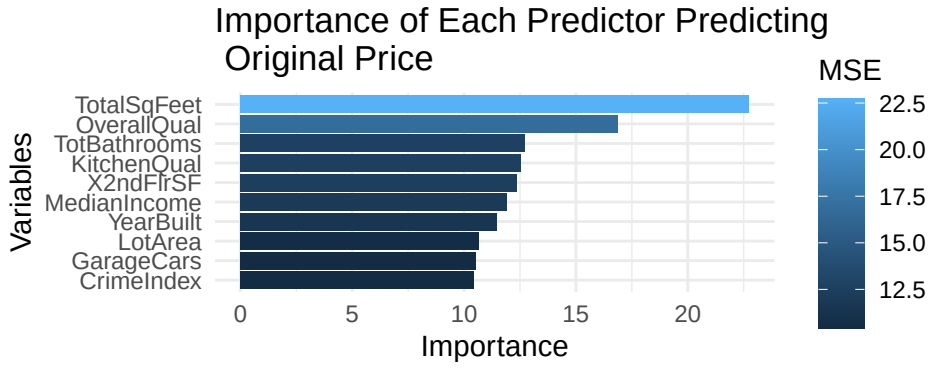

Figure 9: Original Price Importance

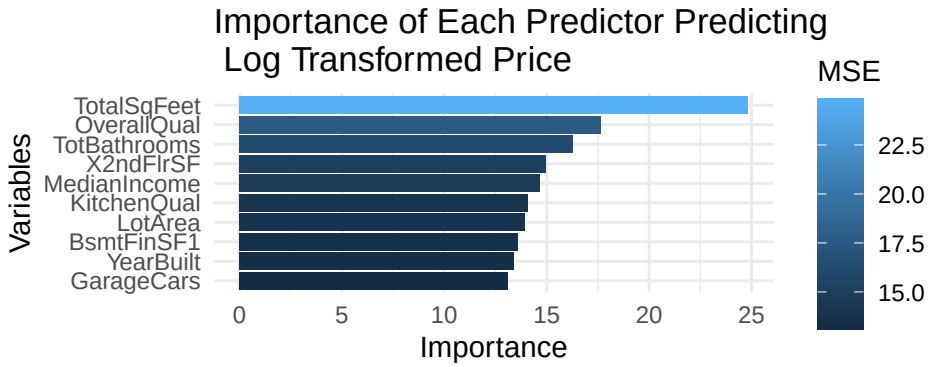

Figure 10: Log Transformed Price Importance

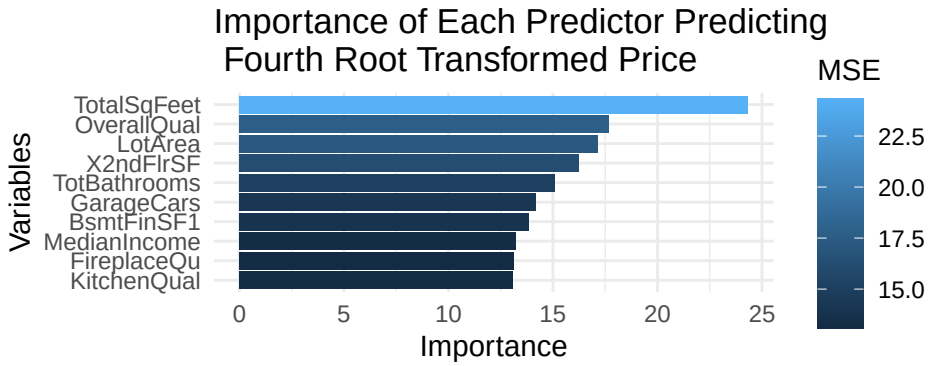

Figure 11: Fourth Root Transformed Price Importance

From the above importance analysis, we could see the group of top ten most important predictors are the same for all three types of y-values. However, their importance varies a lot, but from the graph we could be assured that the top two most important predictors are the following:

1. TotalSqFeet (Total Area)
2. OverallQual (Overall Quality of the building)

3. Principle Components Analysis (PCA) and Principle Components Regression (PCR)

i. PCA

So far, our data totally has 216 columns of features (29 dense columns of numeric values and 187 sparse columns of one hot encoding values). In order to reduce the dimension of a mostly sparse feature matrix effectively, we applied the PCA method. Fortunately, we successfully reduced the dimension from 216 to no more than 32. 32 PCs have totally 99.997% of variance proportion. 17 PCs are enough for totally getting 85.068% of variance proportion. It is interesting that the number of PCs, 32, which is more than the number of dense columns, 29, shows that PCA has indeed extracted meaningful information from 187 sparse columns of one hot encoding and has successfully cut down the number of dimensions.

According to figure 10, we have found that all variance proportions behind 32th PC are 0, which means 32 PCs are totally enough important to describe all features of the original data. In fact, we can take less than 32 PCs and find the best number of PCs with cross validation in training process.

ii. PCR

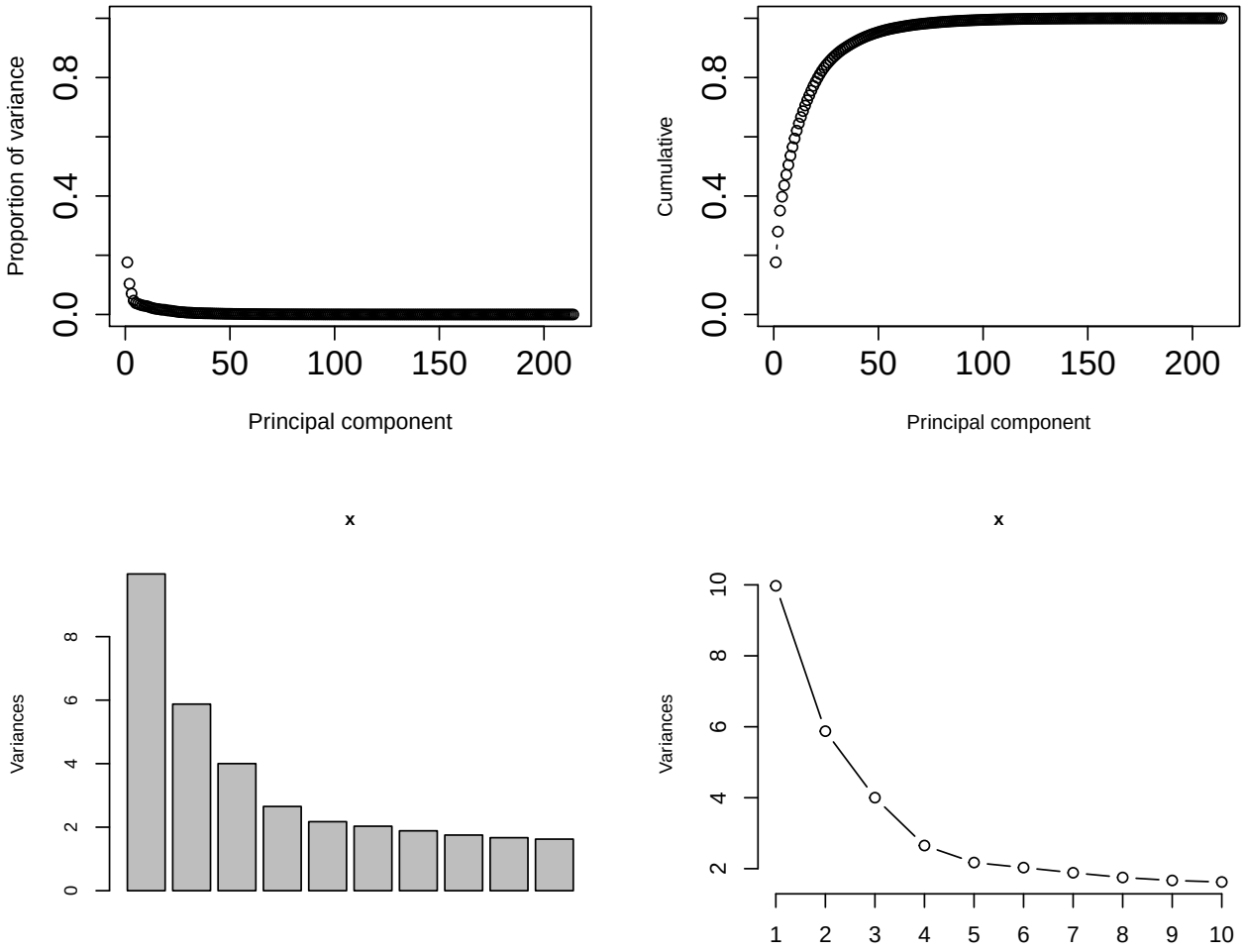

Figure 12: Principle Components Analysis

Since we wanted to obtain as much variance as we could, but also trying to limit the number of predictors, we calculated the cross validation MSE for each number components in a pcr model to get a relatively ideal number of components. As we can see from the graph below, the cross validation MSE is the lowest when there are 29 components in the PCR model. Therefore we applied 29 components to test the result of our testing data set.

4. Ridge Regression

Explanation

In our data, the number of the data is relatively small compared to the number of the predictors, and we think there are some Multicollinearites within our predictors. All of these would cause overfitting. Under this circumstance, we think maybe linear regression would not perform pretty well. To prevent the overfitting and increase the explanatory power of the model, we will try to use some Shrinkage methods. In this case, we would use Ridge regression and Lasso regression.

Prepare Model

First, we set initial alpha to 0 to fit the ridge regression,and set the values of initial lambda ranging from 10^{-2} to 10^{10}, essentially covering the full range of scenarios from the null model containing only the intercept,

to the least squares fit.

This is the optimal lambda computed by cross validation during the model training for Ridge Regression. as the following:

Table 2: Lambda for Each Prediction

y	lambda
Original	6483
Log	0.05837
Fourth Rooted	0.2601

5. Lasso Regression

Explanation

Lasso is very similar to Ridge regression in that a penalty term is added to the regression optimization function(OLS) to prevent overfitting and reduce the effect of Multicollinearity.

But compare to ridge, Lasso use the L1-regulartion, which will shrink some parameters to 0, this could use for parameter selection and make the result more interpretable, could help us find out which parameter is what Lasso thinks is important.

Prepare Model

Set the initial alpha is equal to 1 (Ridge regression is 0), and also use the same initial lambda, then try to use cross validation to choose the optimal lambda for Lasso after using different type of SalePrice

According to different type of SalePrice, we conducted the model training for five times to each, and see how the model performened on average and whether the transformation in SalePrice will make the prediction better.

This is the optimal lambda computed by cross validation during the model training for Lasso Regression. as the following:

Table 3: Lambda for Each Prediction

y	lambda
Original	516.9
Log	0.01
Fourth Rooted	0.01078

Coefficient From Lasso Regression

Here we are going to show the predictors lasso chose

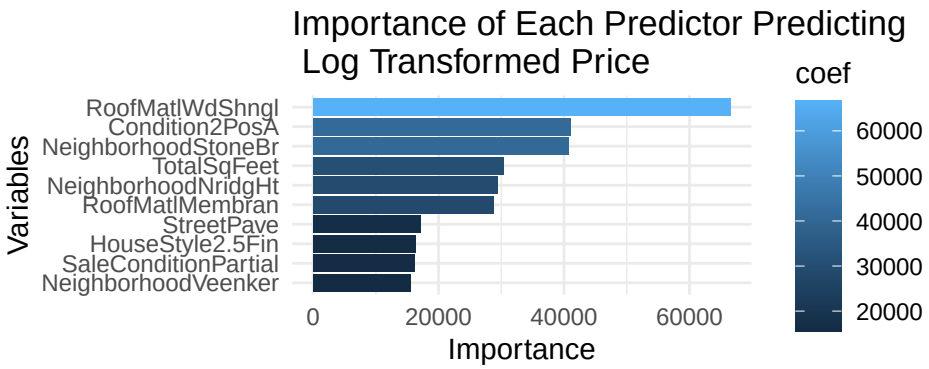

Figure 13: Original Price Importance

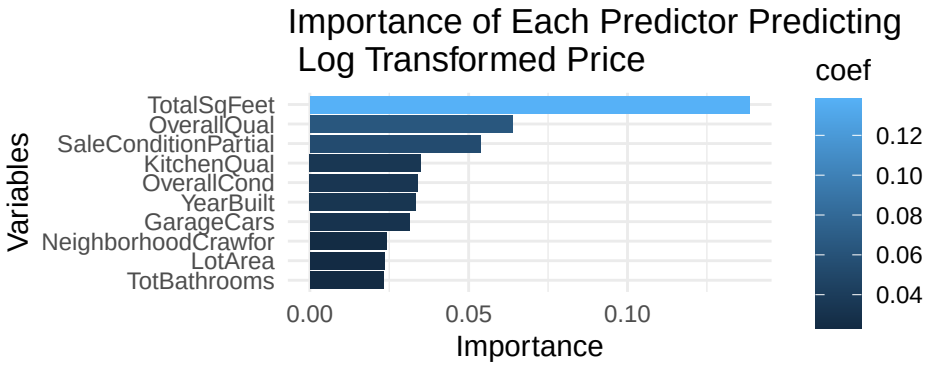

Figure 14: Log Transformed Price Importance

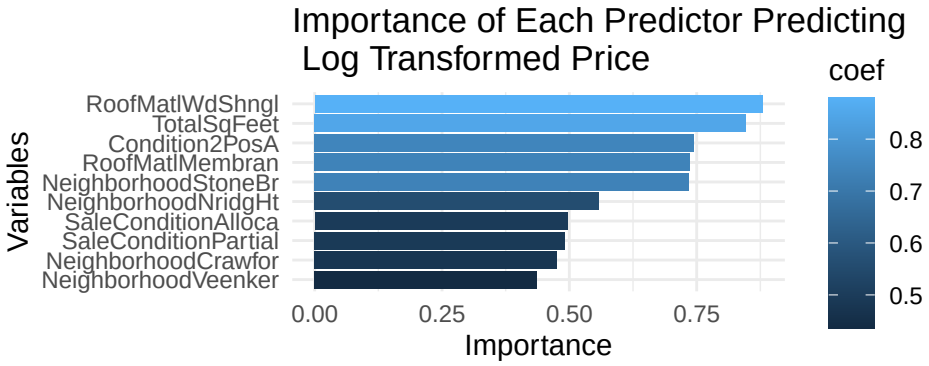

Figure 15: Fourth Root Transformed Price Importance

Evaluation of different models

We then need to check accuracy, as assumed before, we would look at whether the difference between predicted SalePrice and true SalePrice is within the range we define as accurate prediction. And compare the three reuslts computed by different type of SalePrice. The following is the result. We will select the model with least R^2 as a version of best model.

Table 4: R squared Error Metric

y	Linear.Regression	Random.Forest	PCR	Ridge	Lasso
Original	0.862	0.9152	0.8263	0.8929	0.8929
Log	0.7259	0.9033	0.878	0.9235	0.9196
Fourth Root	0.8557	0.9102	0.8409	0.9278	0.9344

Table 5: RMSE Error Metric

y	Linear.Regression	Random.Forest	PCR	Ridge	Lasso
Original	28688	27454	28717	25868	24575
Log	9014392	30873	33091	22042	23063
Fourth Root	34638	29718	29246	20449	21689

Table 6: Accuracy Error Metric

y	Linear.Regression	Random.Forest	PCR	Ridge	Lasso
Original	0.5893	0.6156	0.5211	0.6483	0.5931
Log	0.6674	0.6613	0.6398	0.6621	0.6268
Fourth Root	0.692	0.6613	0.6092	0.6659	0.6628

Conclusion

From our observation in the error metric tables, we realize that RMSE works best in describing which ones of our models works the best in predicting the y-values. It is because it helps to penalize larger errors, while R^2 is not very decisive, and it could imply overfitting. Accuracy is an indicator that we created just to have an additional insight of how well the models perform. Thus, we take a closer look at RMSE for each model:

1. Lasso works the best for Original y-value
2. Ridge works the best for both Log and Fourth Root transformed y-value

We then, could say that Ridge is the one we would like to use for our final validation and testing of prediction of House Price at North Ames, Iowa. From the above three error metric table, we could see that accuracy for each model has actually improved a little for each converted y-value. However, it is worth notice that most of the models are having trouble improving their R-squared or RMSE with a processed y-value, therefore showing a risk of overfitting original data with a converted y-value.

At the end, we pulled our test data out from the vault and performed Ridge Regression over fourth-root transformation, the final result is 0.93 of R-square, 20449.24 of RMSE and approximately 60% of accuracy.

Table 7: Final Result

Error.Metrics	Value
R2	0.9278
RMSE	20449
Accuracy	0.6183

Discussion & Future Development

The challenges we face in this data set are having too few rows of data (only 1468 rows) while having a lot of predictors (both numeric and categorical), with a simple regression model might not be helpful to have a very accurate prediction. We can consider using ensemble method and other dimension reduction methods to increase its accuracy. Also, we did not take months into account to build our model, however months might affect the price of houses simply because people tend to buy houses in some seasons. We would try to use new methods and create more features to try to build a more robust and accurate model in the future.

Reference

"Advanced Regression Techniques for House Prices." n.d. https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data?select=test.csv.

James, Gareth. 2013. *An Introduction to Statistical Learning with Applications in R*. Springer. https://doi.org/10.1007/978-1-4614-7138-7.