Predicting House Price with Multiple Linear Regression

*yung chou*

*November, 2018*

House Price Prediction

This project was to develop a Machine Learning model for predicting a house price. Despite there were a number of tree-based algorithms relevant to this application, the project was to examine linear regression and focused on specifically four models: Linear Regression, Ridge Regression, Lasso Regression and Elastic Net.

* [Overview](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#Overview)
* [Data Analysis](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#data)
* [Feature Selection](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#features)
* [Data Visualization](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#visualization)
* [Model Development](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#model)
* [Model Comparison](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#comparison), [3D Views](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#3d)
* [Closing Thoughts](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#closing)

Overview

[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

In this article, “variable”" as a general programming term and “feature” denoting a predictor employed in a Machine Learning model are used interchangeably. The following outlines my approach and highlights the logical steps which I followed for developing a Machine Learning models. *The development process was highly iterative and the presented steps were not necessarily the exact order. Nevertheless, these steps correctly depict the thought process and overall strategies for developing a Machine Learning model.*

* **Data Set** The data set was downloaded from [Kaggle House Prices:](https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data) Advanced Regression Techniques. There were two files: train.csv with 1460 observations and 81 variables, while test.csv with ???? observations of the label, SalePrice.
* **Missingness** There were a few variables with considerable amount of missing values, essentially unusable and removed from subsequent process. Those missing at random were later imputed with values.
* **Character variables** Factor variables were read in as character ones. Some character variables with several unique values. They were converted into ordinal and minimized into two or three levels for later imputing missing values and selecting features programmatically.
* **Numeric Variables** There were some extreme values among numeric variables in the train data set due to the way the values were captures. For instance, those measures such as deck, porch or pool ranges from 0 when not applicable to hundreds in squared footage. When modeling these variables as predictors, those with large values might overwhelm and skew the model. These variables were minimized to just a few levels and converted to numbers better reflect real-world scenarios. Above all, the strategies to select what and determine how to convert a variable have much to do with the composition and distribution of the data. Often, the values of a variable are not as significant as the variance of those values.
* **Extreme Values and Outliers** Not all variables with values larger than 1.5 IQR were removed from the train data set. Some of these extreme values appeared characteristic and influential to some model configurations. In a few test runs, removing outliers or those observations resulting residuals with much leverage actually decreased Rsquared values. For a data set, like the Kaggle House Price, the interactions among variables can be intricate since there are many variables. Making one change at a time, documenting the changes well, and backing up the settings often are the lessons I have learned well from handling outliers and extreme values of this project.
* **Imputation of Missing Values** Used the package, Multivariate Imputation by Chained Equations [(mice)](https://cran.r-project.org/web/packages/mice/index.html), for imputing values programmatically throughout the development.
* **Feature Selection** Used the package, [Boruta: Wrapper Algorithm for All Relevant Feature Selection](https://cran.r-project.org/web/packages/Boruta/index.html), to initially selecting features. Subsequently, removed insignificant features from the model based on the significance level of test runs. This process was iterative and carried out along with model development. As test statistics confirming the impact or importance of a feature, it was restored or removed accordingly. [An example of running Boruta is available.](https://yungchou.wordpress.com/2018/11/19/feature-selection-with-help-from-boruta/)
* **Near Zero-Variance Variables** A variable with little variance behaves like and is essential a constant with values distributed near its mean. A constant-like or near zero-variance variable contributes little to a Machine Learning model since little correlation with an outcome, namely a prediction, of applying changes to the model. With the package, Classification and Regression Training [(caret)](https://cran.r-project.org/web/packages/caret/index.html), once can identify and process a near zero-variance variable programmatically.
* **Partitioning Data** Partitioned the train data set into 70/30 where 70% for training and 30% for evaluating the model.
* **Cross-Validation** Used 10-fold cross-validation in all training and with 5 repetitions.
* **Linear Model** Overall, simply including all variables in a linear model without interaction between variables could achieve Rsquared value above 80%, while the model remained unstable. Adding relevant interaction variables improved the model noticeably with stability. However, the model seemed reach its limitation in current configuration when Rsquared near 91%.
* **Ridge, Lasso and Elastic Net** Tried various combinations and ranges of lambda and alpha values to find sets of tuning parameters. This process was in some way experimental due to the results were based on the combined effect of the seed value for randomness, the starting and the end points of lambda and alpha, and the step size. In Elastic Net, although various settings resulted in various sets of turning parameters, the overall Rsquared values of the elastic model remained stable.
* **Model Comparison** Comparing the four models: Linear, Ridge, Lasso, and Elastic Net showed Lasso was influential and largely adopted by Elastic Net in the developed model.
* **Predictions** Although the main objective of the project was to examine and analyze linear regression and not necessarily engineer for a high Kaggle score. Submissions made resulted to .014 range with predictions made by the Elastic Net model.

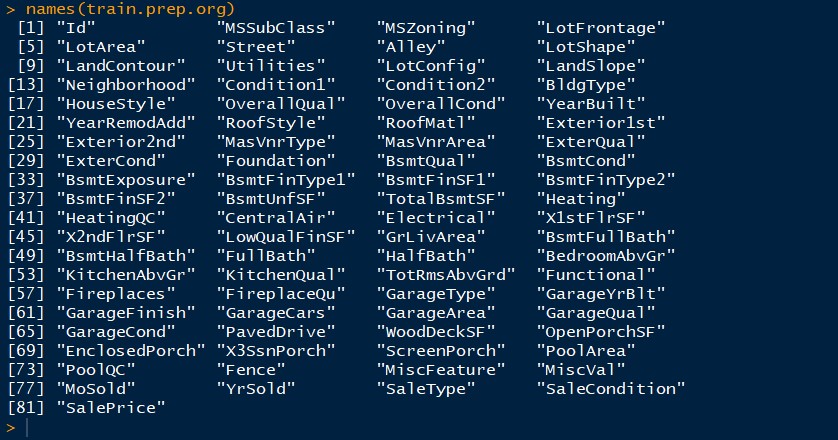
Data Analysis

[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

Kaggle House Price Dataset

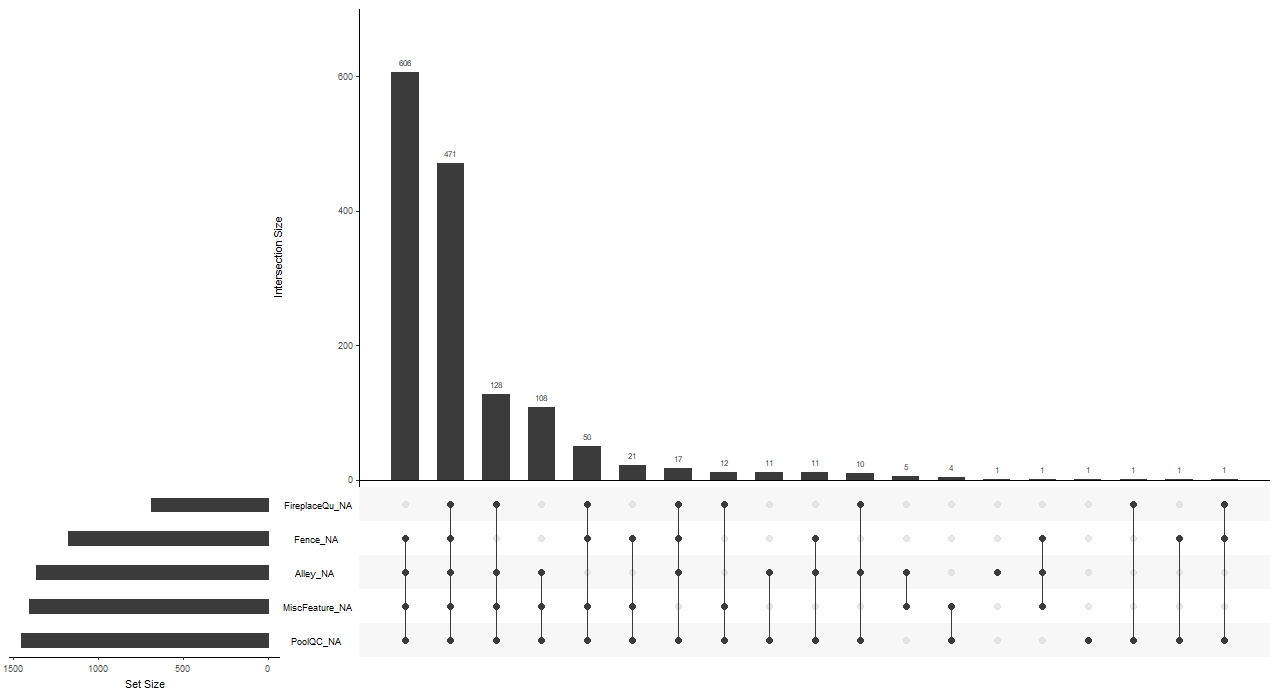
Downloaded and imported the train data set. Here’s some information by examining the structure and the summary.

## [1] "Imported train data set: 1460 obs. of 81 variables"



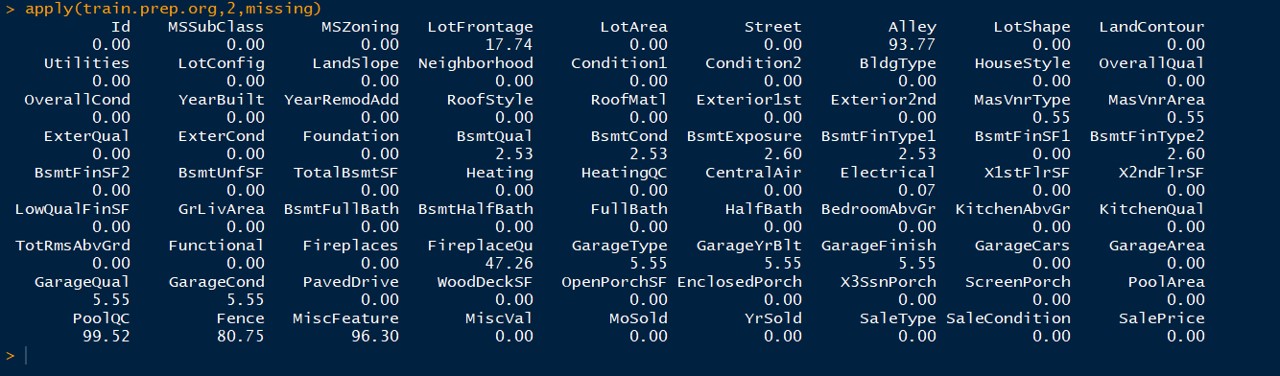
Missingness

Next examined the distribution of missingness and the percentage of missing values. There were a few variables with most observations missing, which made these variables not usable and they were consequently removed. Here’s a visualization of missingness of the train data set.



Percentage of Missing Values

Further examination of the percentage of missing values of each variable revealed:



Feature Selection

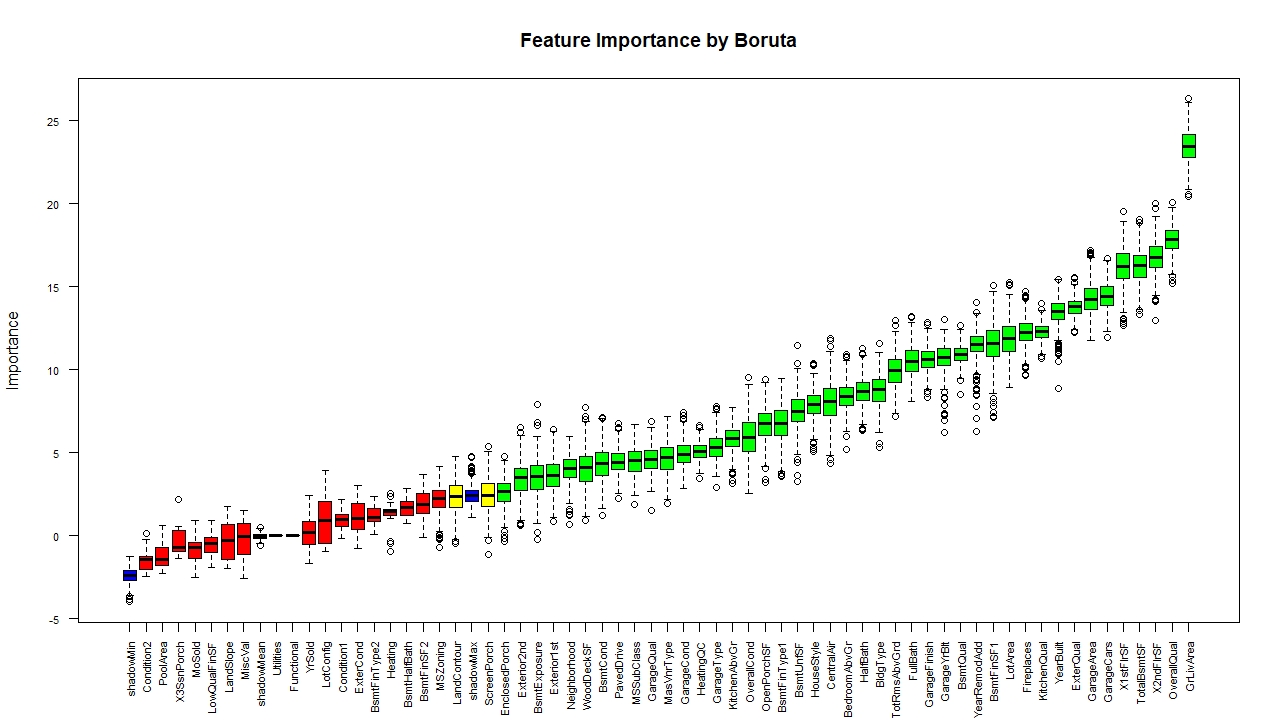
[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

Removed a set of variables at this time based on:

* a large percentage of missing values which made a variable not usable
* feature importance confirmed by Boruta
* a consistent insignificant level of p-value as a predictor in test runs

[Boruta](https://yungchou.wordpress.com/2018/11/19/feature-selection-with-help-from-boruta/)

After having converted all variables in train dataset to integer or numeric fields, programmatically imputed the data for missing values, I ran Boruta to initially analyze the importance of variables. And it took about 40 minutes in the context to iterate 500 times and produced something like the following results where those in green were with confirmed importance, while red rejected, i.e. not important features. The yellow ones were tentative which were not yet resolved before reaching the set number of iterations.

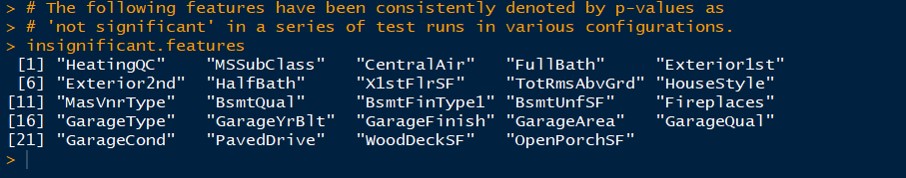


Stored the list of features confirmed by Boruta and subsequently removed these features not included in this list from the train dataset.



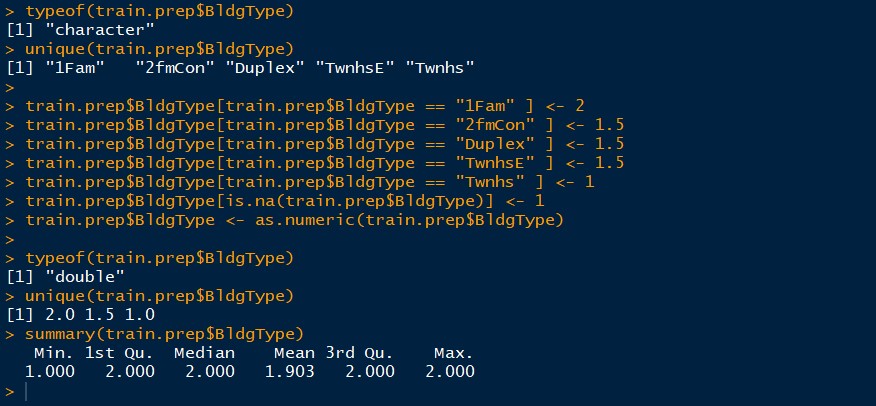
Features with Insignificant P-Values

While iteratively developing, fitting, and tuning the model, I documented a list of features consistently with insignificant p-values, i.e. greater than 0.05, in test runs. Below is a snapshot of these features to be removed form train dataset prior to executing a test run. Notice these features were not a unique set and various development paths and configurations could and would result a different set of features.



Character Variables

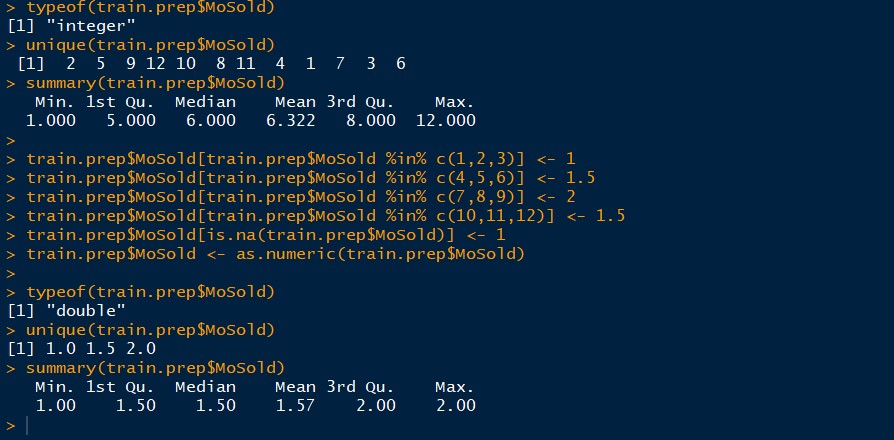
Factor variables were read in as character ones. Rather than converting into factor variables, they were converted into integer or numeric fields for later imputing data as well as deriving feature importance programmatically.



The above, for example, showed the variable, BldgType, was a character variable with five unique levels. It was converted into an ordinal one with values between 1 and 2. Notice that the process was iterative during data preparation and feature engineering. Both converting and combining variables were considered. Domain knowledge, subjectivity, and common sense were all relevant to the what and how to convert a variable, as applicable. The technique and strategies can and will vary from person to person and model to model.

Numeric Variables

For numeric variables, their values can produce unintended effects. For instance, assume modeling a house price having a linear relationship with the month a house is sold. In such case, a generalization is essentially inherited into the model, that a house sold in December with a value of 12 would contribute 12 times more to a response variable than one sold in January with a value of 1. This configuration fundamentally does not correctly reflect the seasonality, nor the degree of impact on a house price based on the month a house is sold.



One alternative way of modeling seasonality is, as shown above, to convert the variable values to a scale between 0 and 1 where in the summer, i.e. July to September, with the most weight contributing to the market house price, the response variable, and in the winter time the least weight to signify the slow period.



Later, this feature was removed from the final model due to insignificance consistently denoted by p-values in a series of test runs. Still, it was necessary to make the effort to prepare the data and convert this variable, from a January-to-December as 1-to-12 scale to a more meaningful and realistic one for describing real-world scenarios. With a proper scale of this and other similar variables, packages like mice could calculate meaningful values for imputation and Boruta for deriving feature importance.

Above all, the strategies to determine what and how to convert a variable have much to do with an examiner’s domain knowledge, subjectivity, and common sense in addition to reviewing the composition and distribution of the data.

*And the values of a variable sometimes do not tell the whole story. It may not be the values of a variable, but the variance of those values plays a more influential role for making predictions.*

Data Visualization

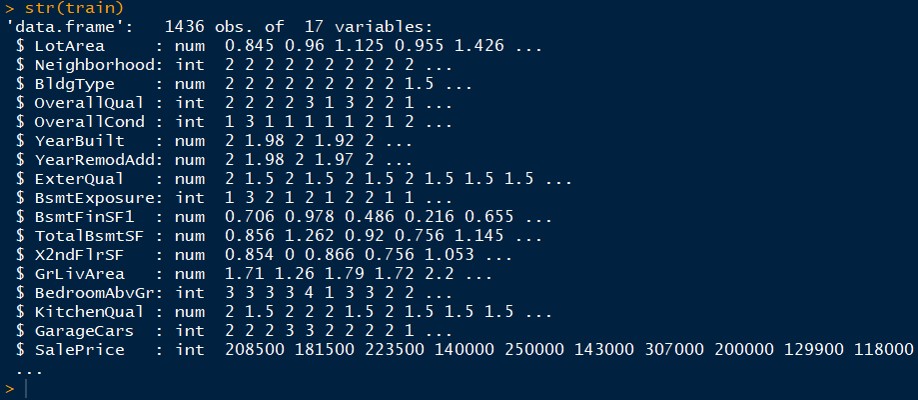
[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

Up to this time, I had an initial set of features to start working on developing a model. Throughout the development, I would make changes of the feature set and observations based on diagnostics of the test results. The presented series of plots were generated along the development process.

Along the development, I produced multiple versions and configurations of the following plots. The set presented here is just one of the many.

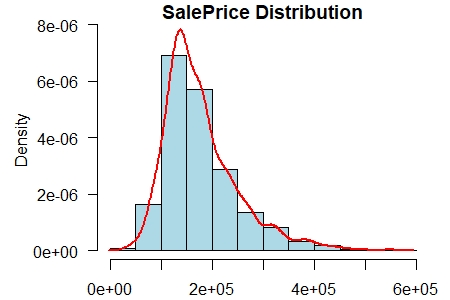
Prepared Train Dataset

Here’s a snapshot of the prepared data set ready for Machine Learning development.



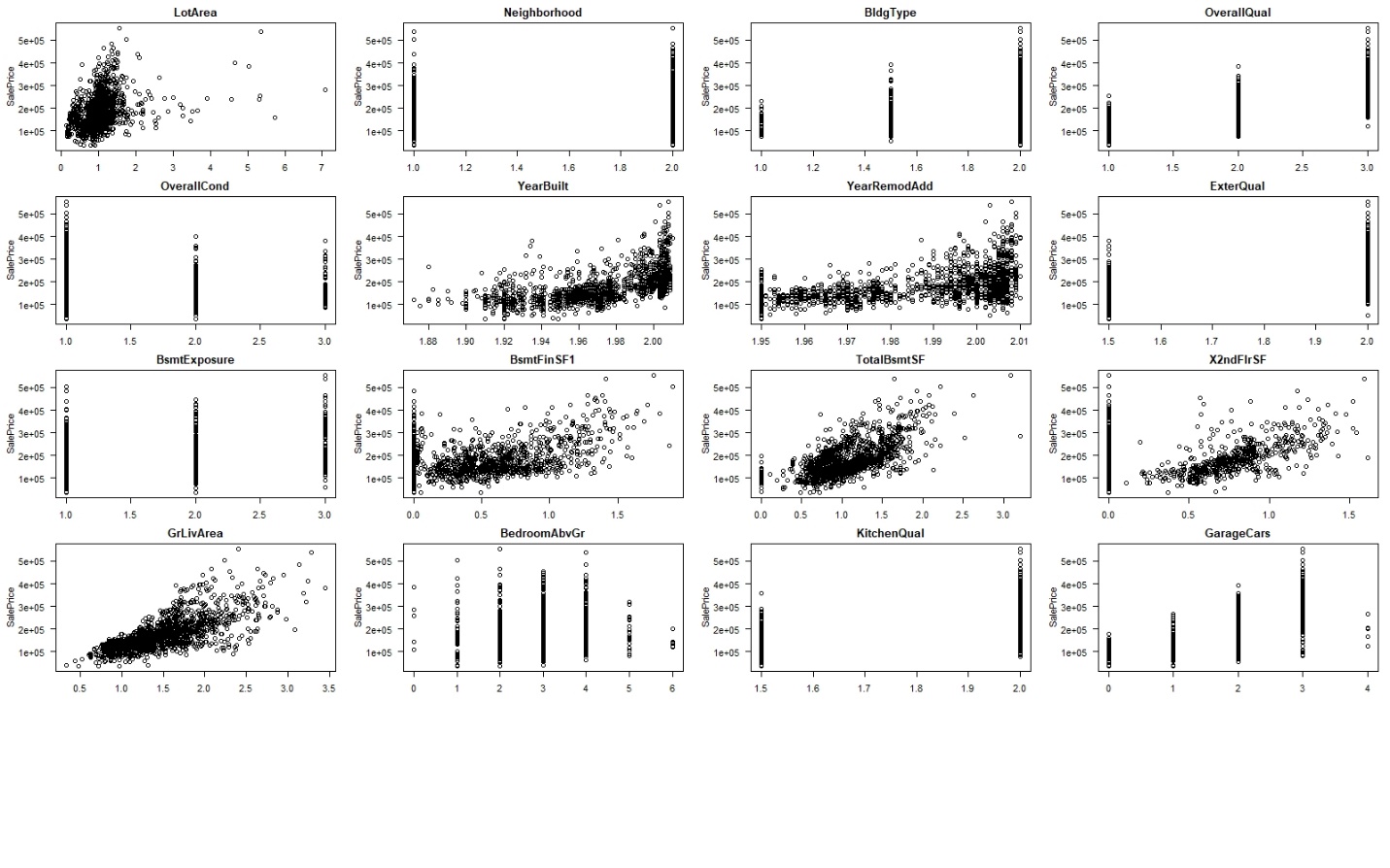
Distribution of the Label

The label, i.e. response variable, was SalePrice. Here it was plotted without logarithm.



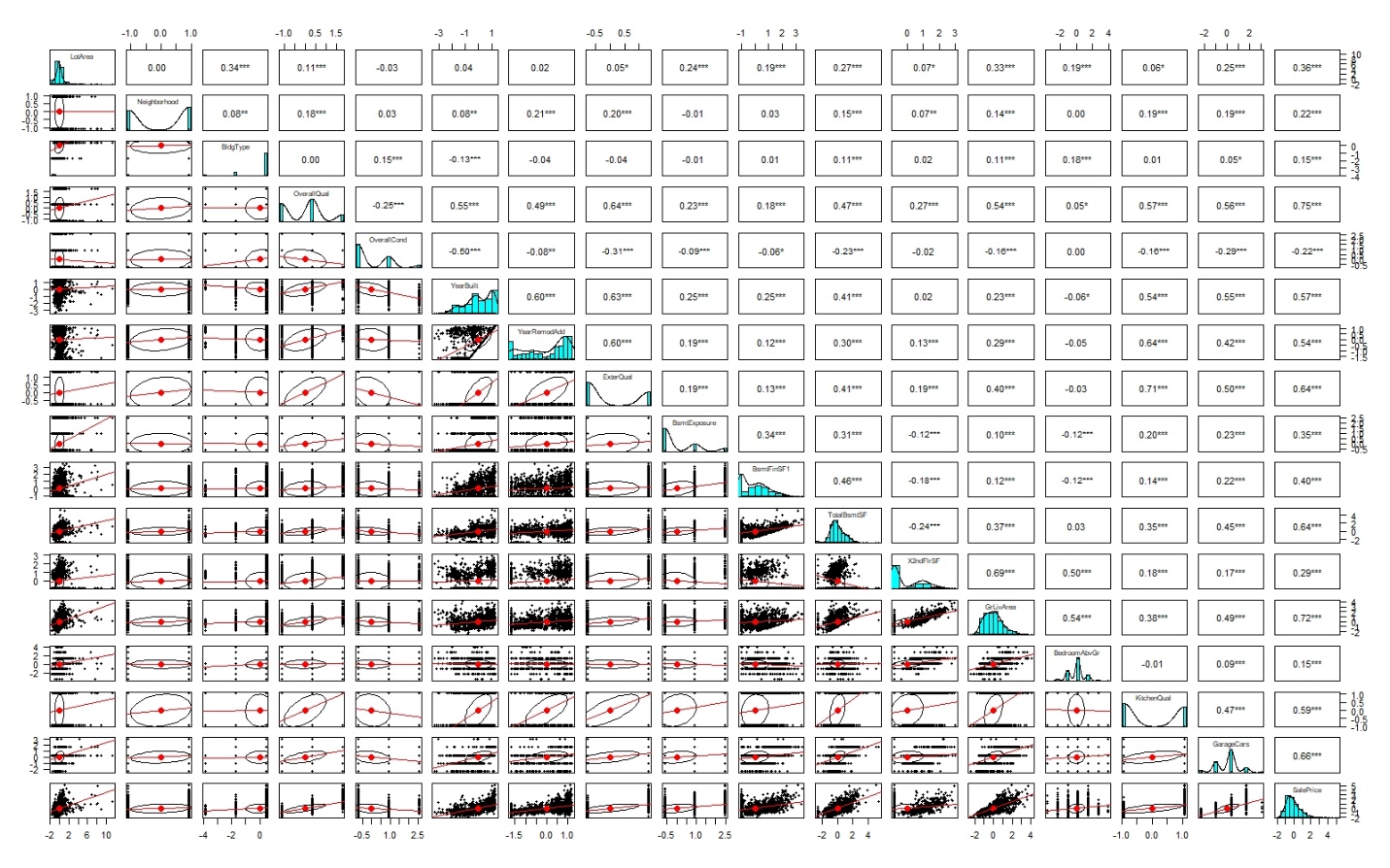
Label vs.Feature

To examine a feature relevant to the label, SalePrice, plotted each pair individually. The linearity among variables was obvious.



Pairs.Panels

Here’s a pairs.panels plot with all features and the label. This plot gives an overview of the linearity between variables and the variance of individual variables.



Correlation Matrix

These three plots: correlation matrix, label vs. feature, and pairs.panels were my main references for developing an initial model.



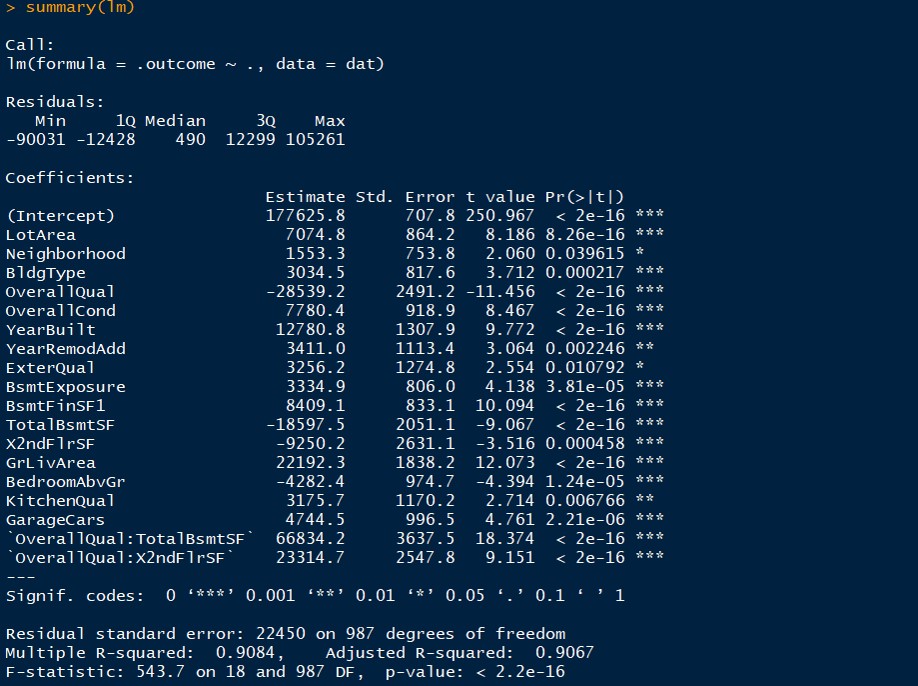
Partitioning Data

I partitioned the train dataset into a 70-30 split where 70% for training and 30% for testing. Here is a set of plots produced by fitting the four regression models: Linear, Ridge, Lasso, and Elastic Net.

1. Linear Model

[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

Here’s a summary of lm for one of the runs. The adjusted R-squared was 0.9067 with insignificant features removed.

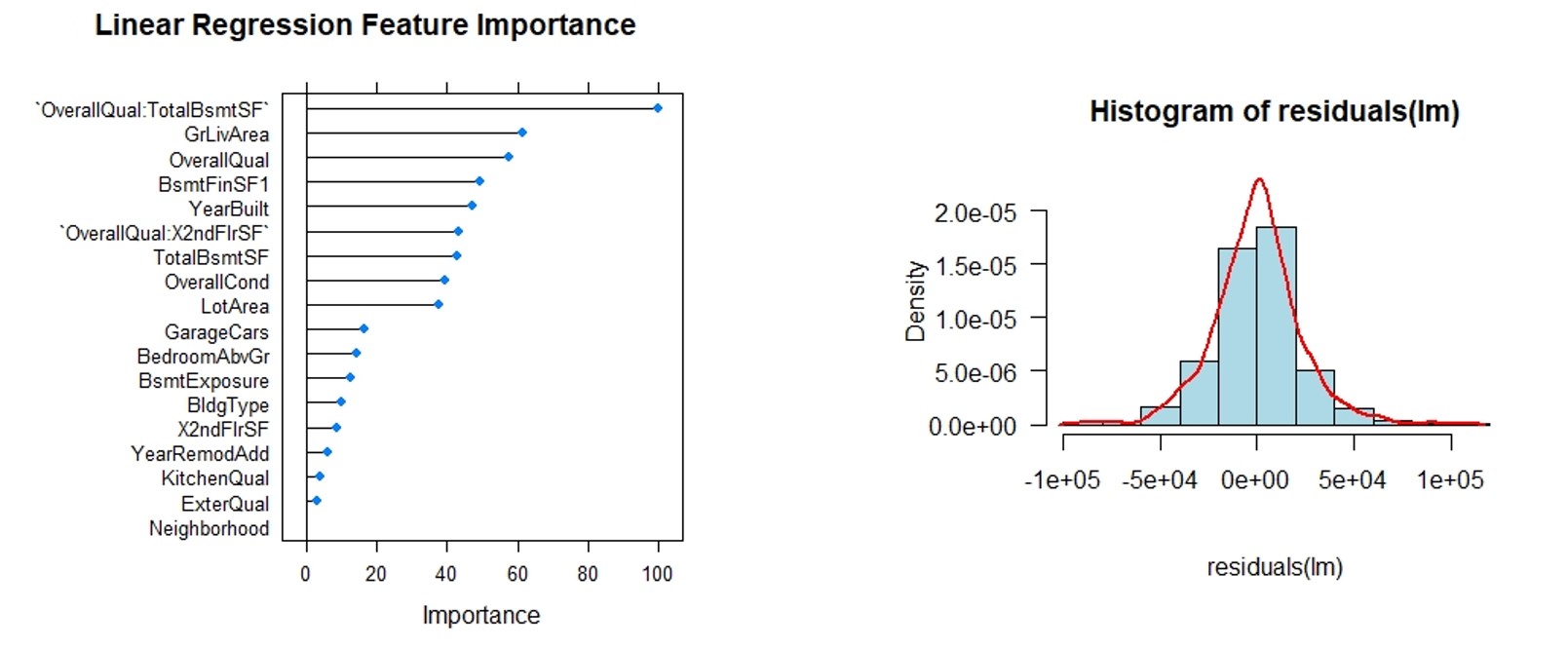


1.1 Diagnostic Plots

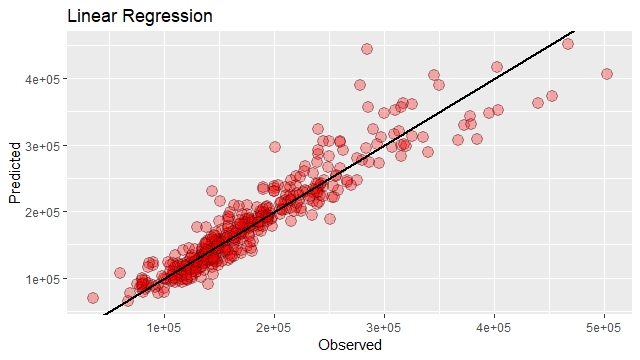
The diagnostic plots played an important role in the initial development. Form the Residuals vs. Fitted plot, there seemed some nonlinearity. Many changes and adjustment made were based on examining and interpreting these plots. In each iteration, I reviewed the plots and changed the composition of features and interactions, removed outliers or added back observations, etc. followed by more test runs. The process was highly iterative and the productivity relied much on well documentation to facilitate the analysis and restore a configuration when needed.



1.2 Variable Importance and Distribution of Residuals



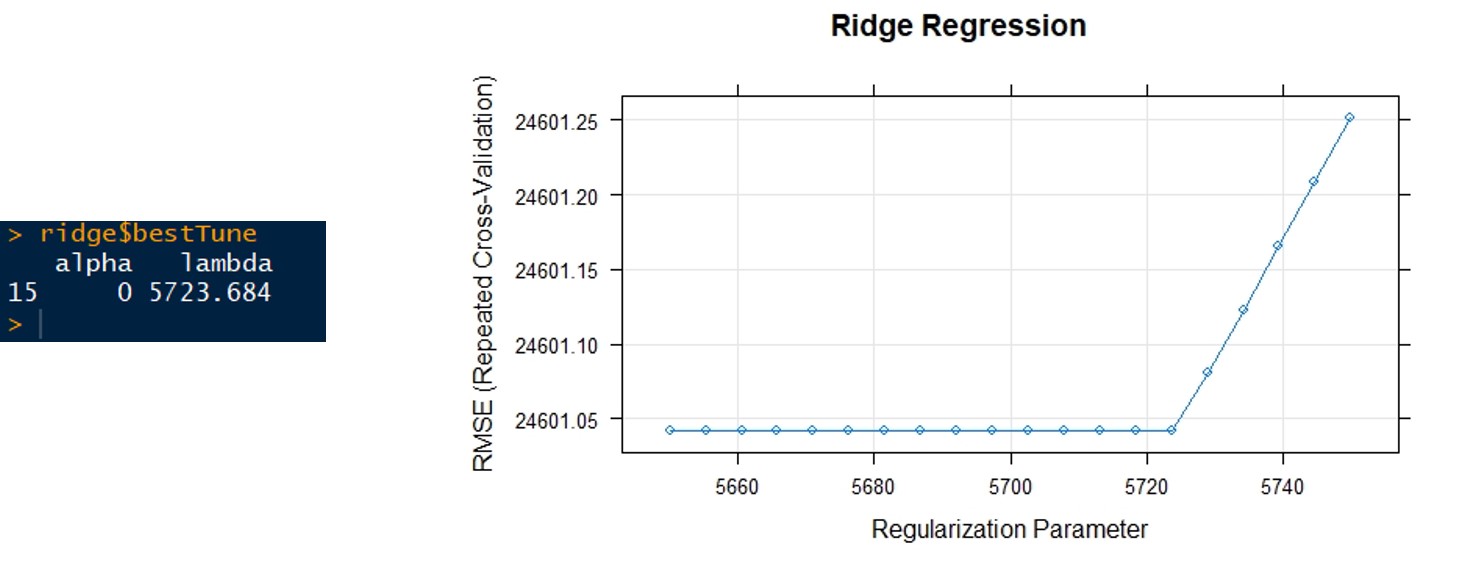
1.3 Predicted vs. Observed

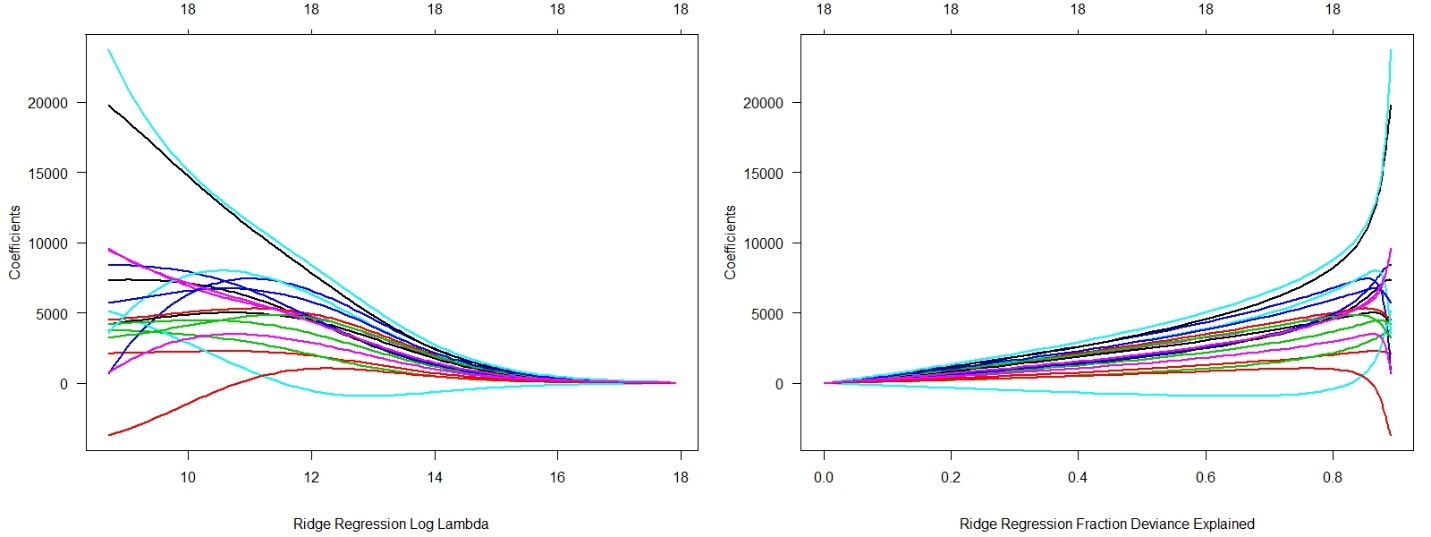


2. Ridge Regression

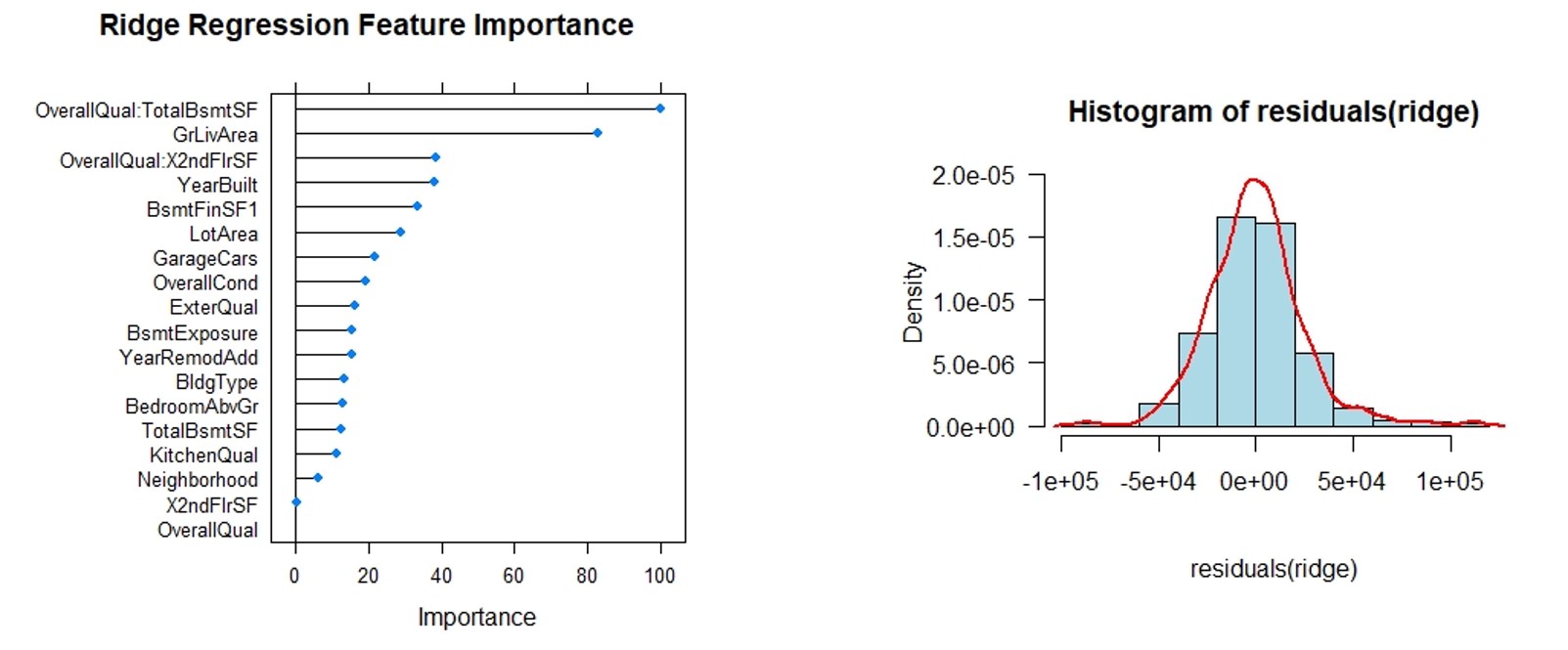
Set alpha=0 and a sequence for tuning Lambda. I started from a wide range like 0.001 to 100 and gradually reduced the range to find a good window. The size of a step sometimes had a noticeable effect on the outcome. Many experimentation and repetitions happened here.

2.1 Regularization

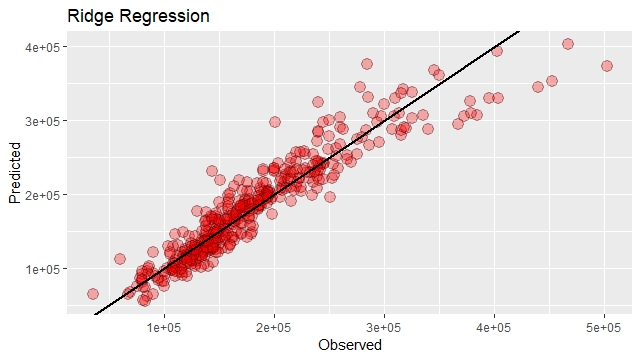




2.2 Variable Importance and Distribution of Residuals



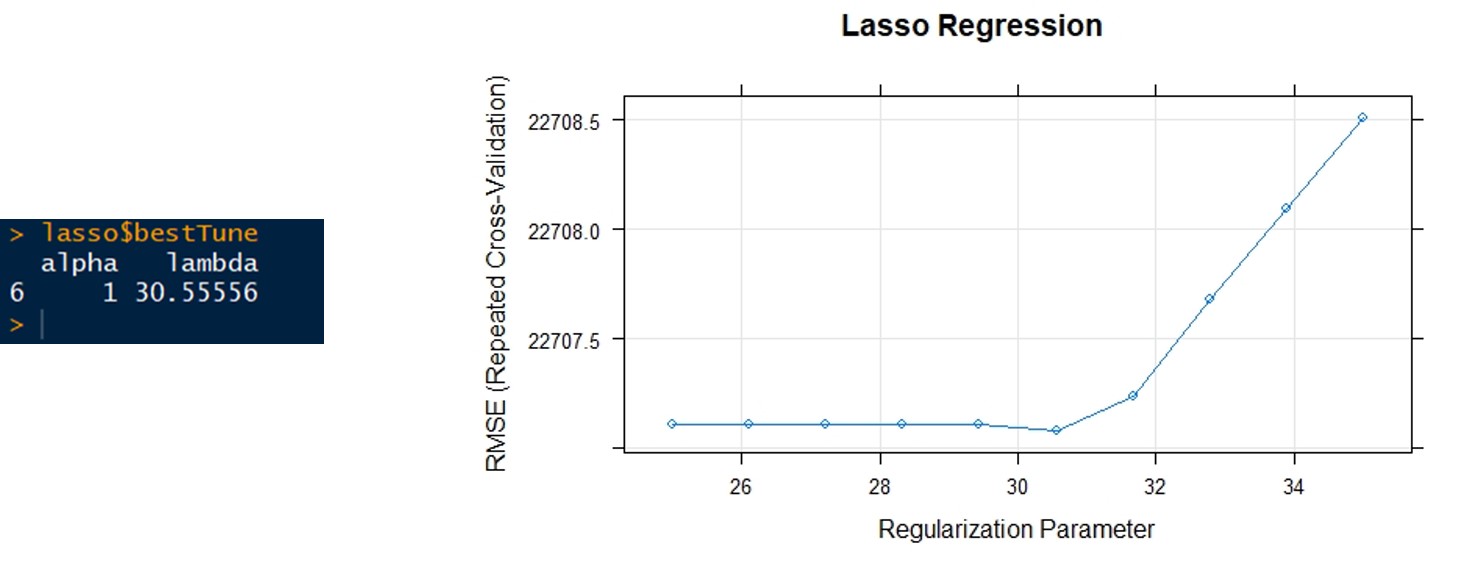
2.3 Predicted vs. Observed

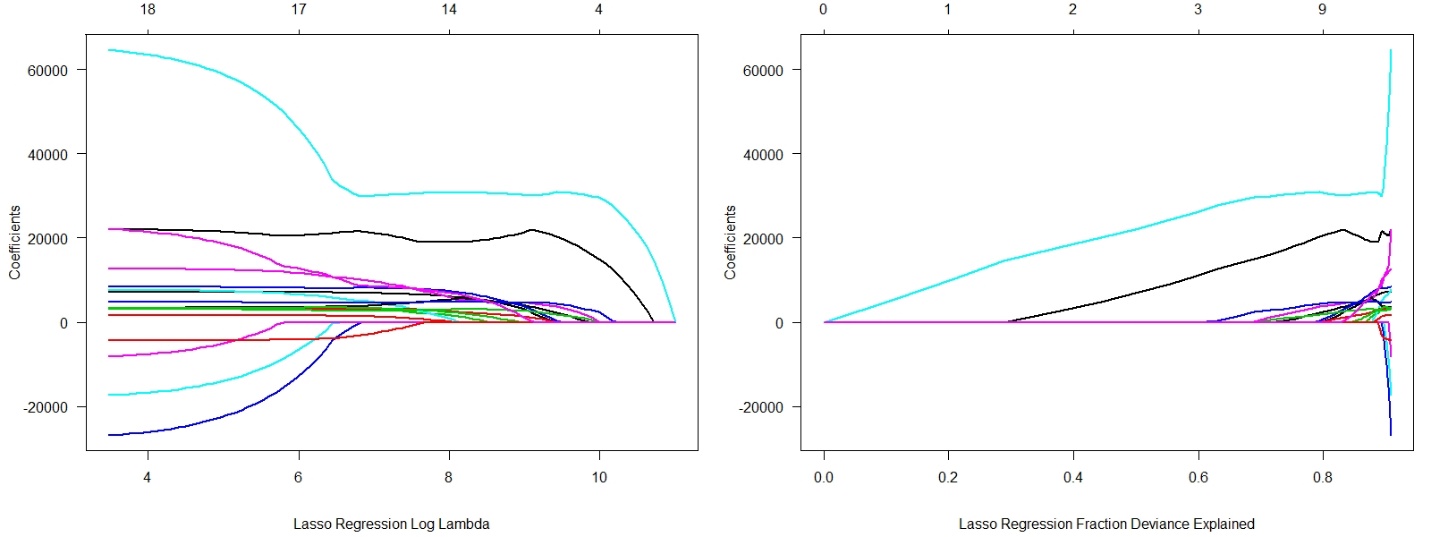


3. Lasso Regression

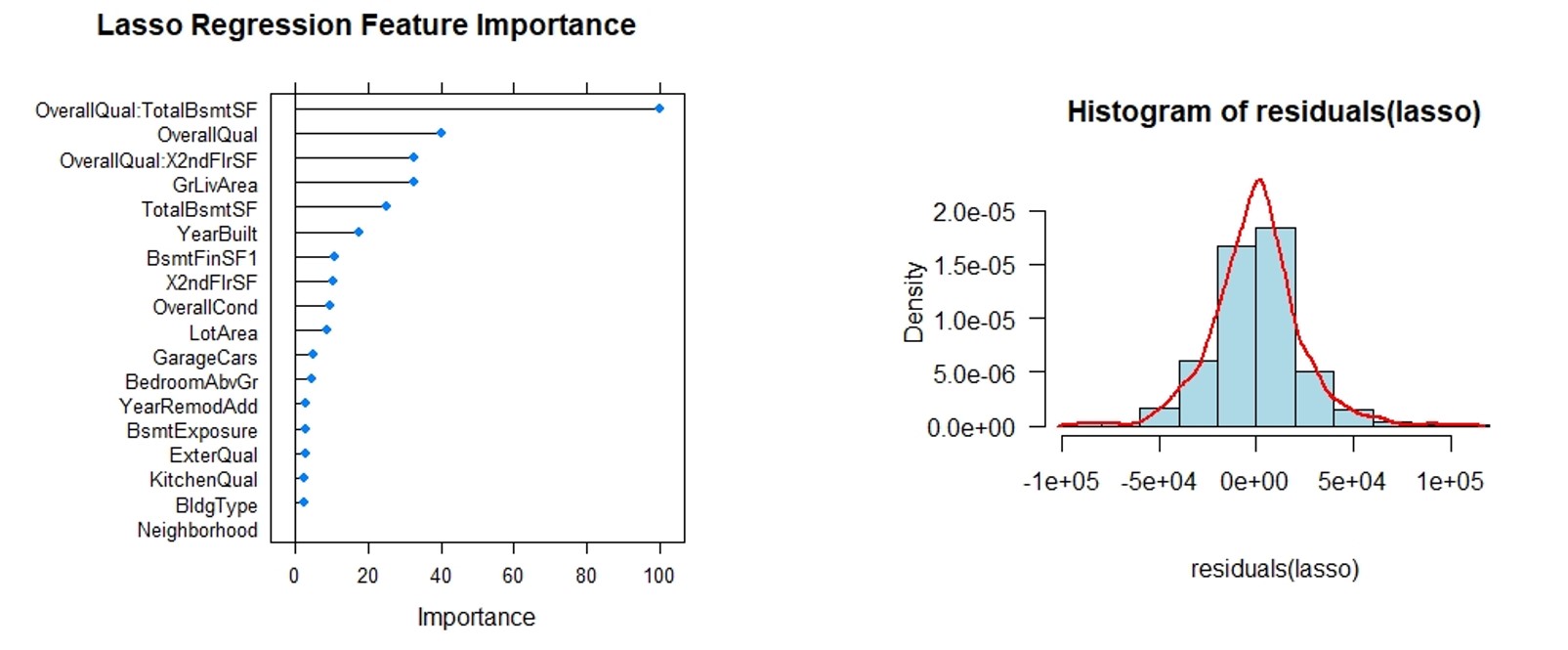
Set alpha=1 and a sequence for tuning Lambda. Like what I did in Ridge Regression, I started from a wide range and gradually reduced to and identified a good range and step to scan.

3.1 Regularization

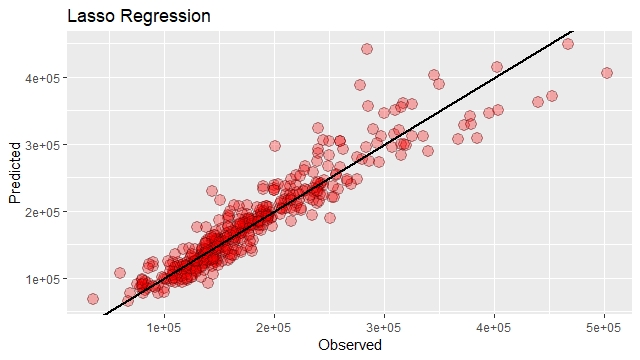




3.2 Variable Importance and Distribution of Residuals



3.3 Predicted vs. Observed



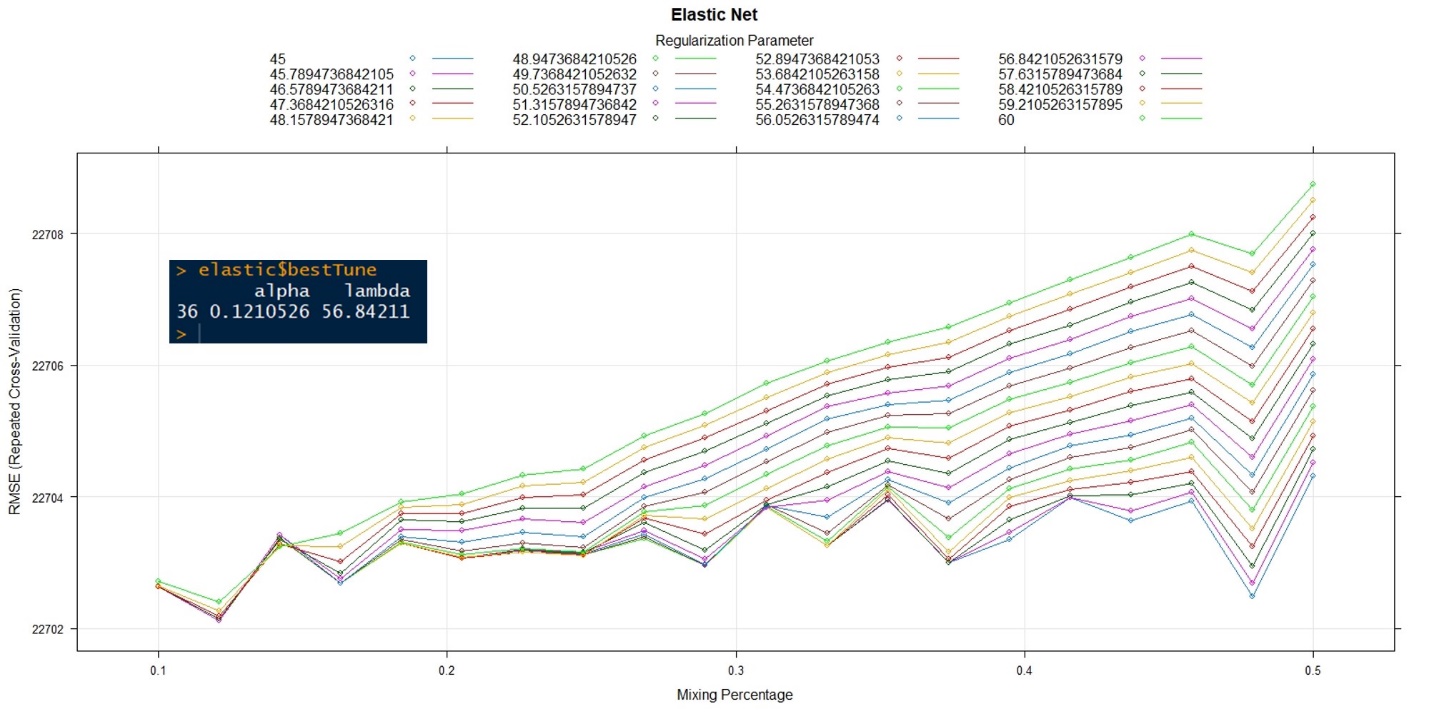
4. Elastic Net

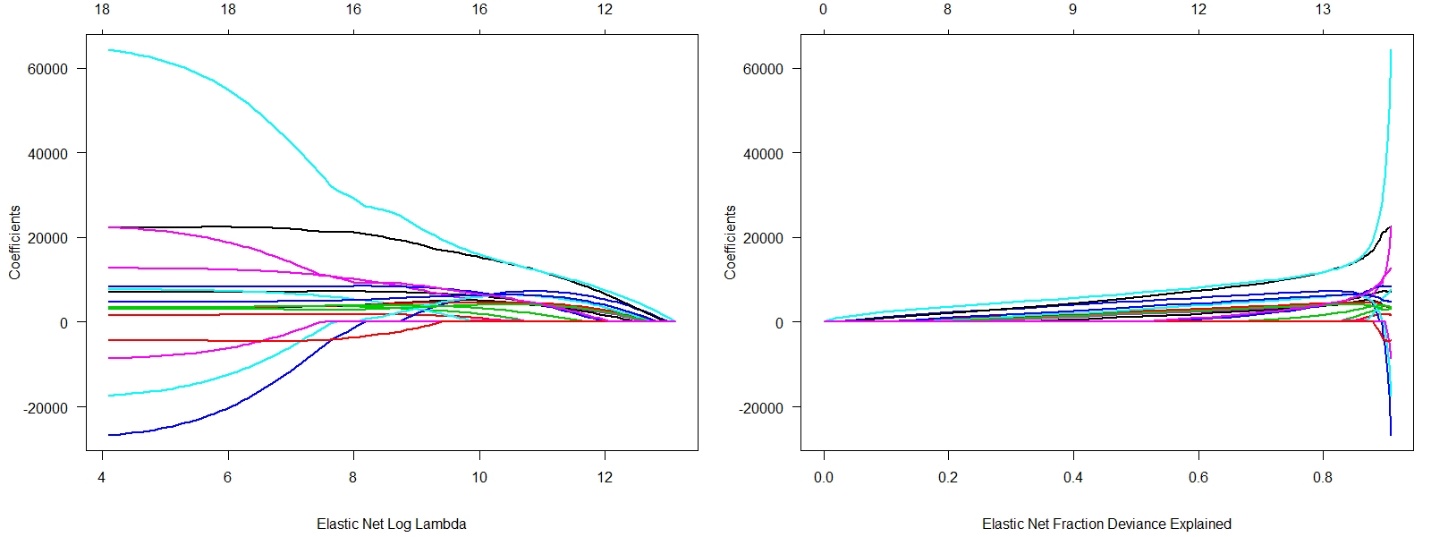
Initially I set one sequence for tuning both alpha and lambda. This turned out not productive for me. Since in a configuration the two values were far apart from each other, the range for scanning would become relatively extensive with a small step sometimes necessary to initially locate the values. A few times my laptop would run out of resources and simply not responding later in a run.

Setting an individual sequence for alpha and Lambda was a more productive approach for me. Nevertheless, the increased combinations and with 10-fold cross validation, it took longer and a few iterations to narrow the ranges and locate the best set of alpha and lambda.

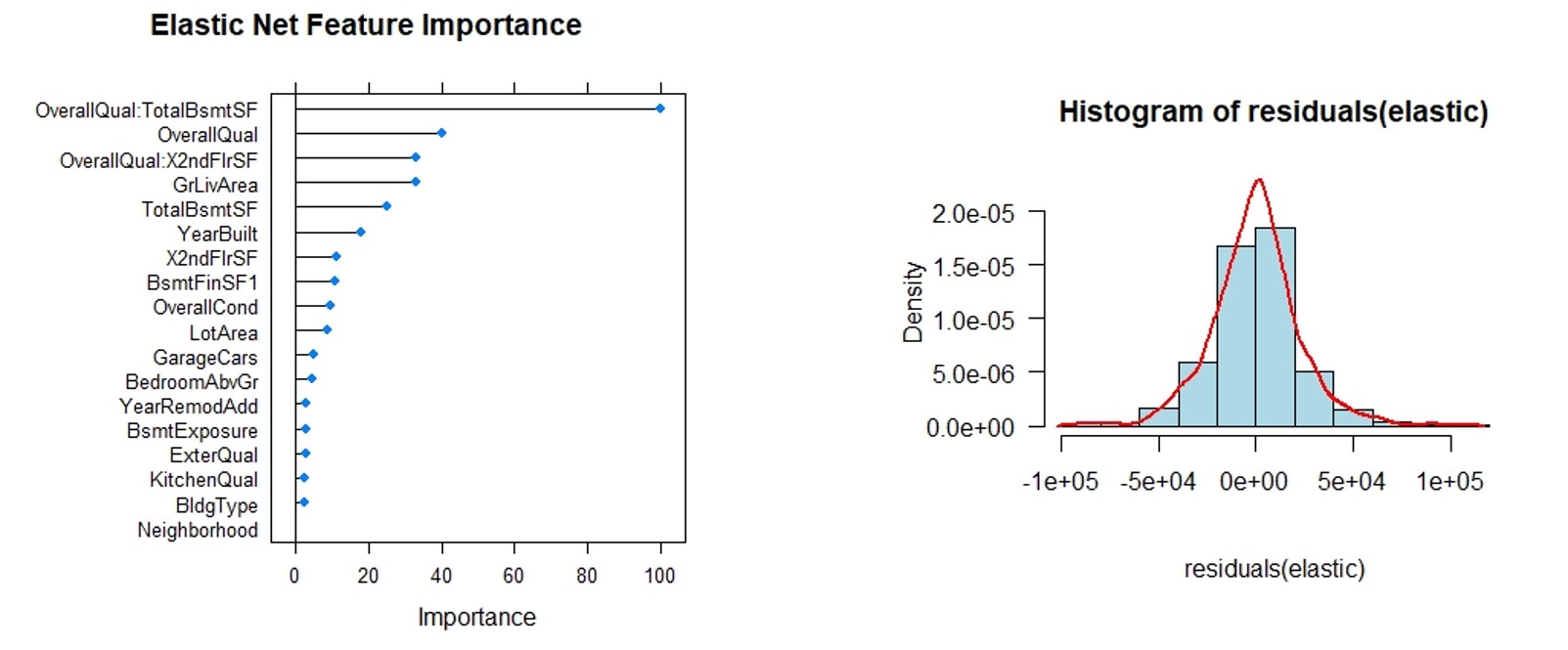
4.1 Regularization

With these many features, overfitting would be likely as these plots revealed.

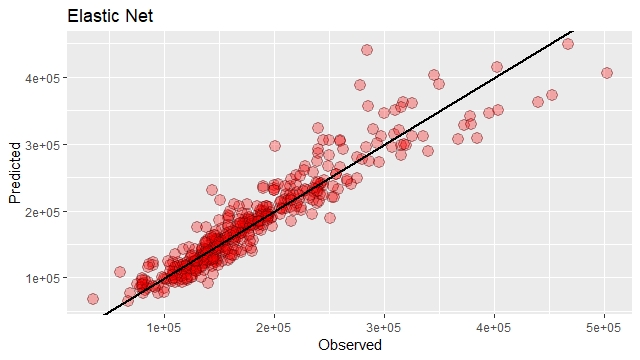




4.2 Variable Importance and Distribution of Residuals



4.3 Predicted vs. Observed

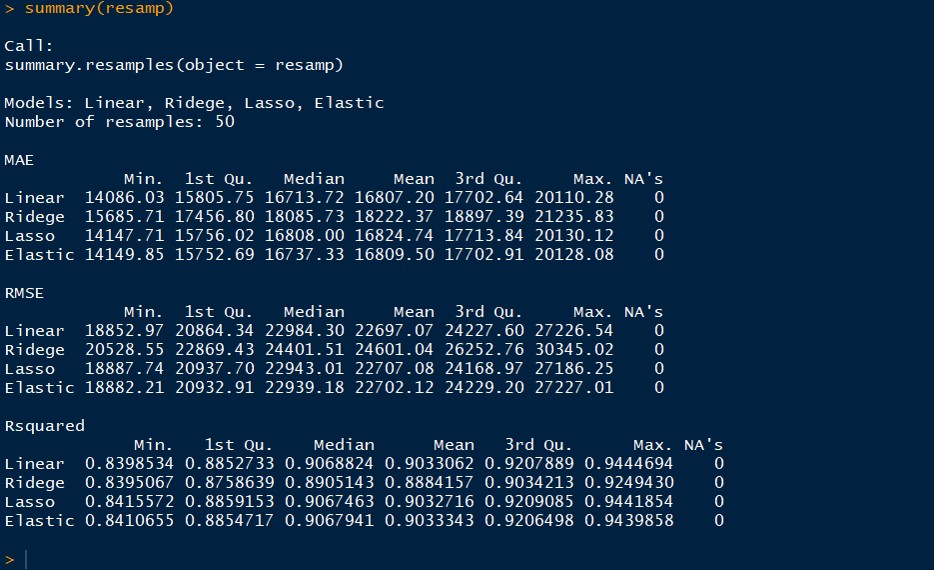


Model Comparisons

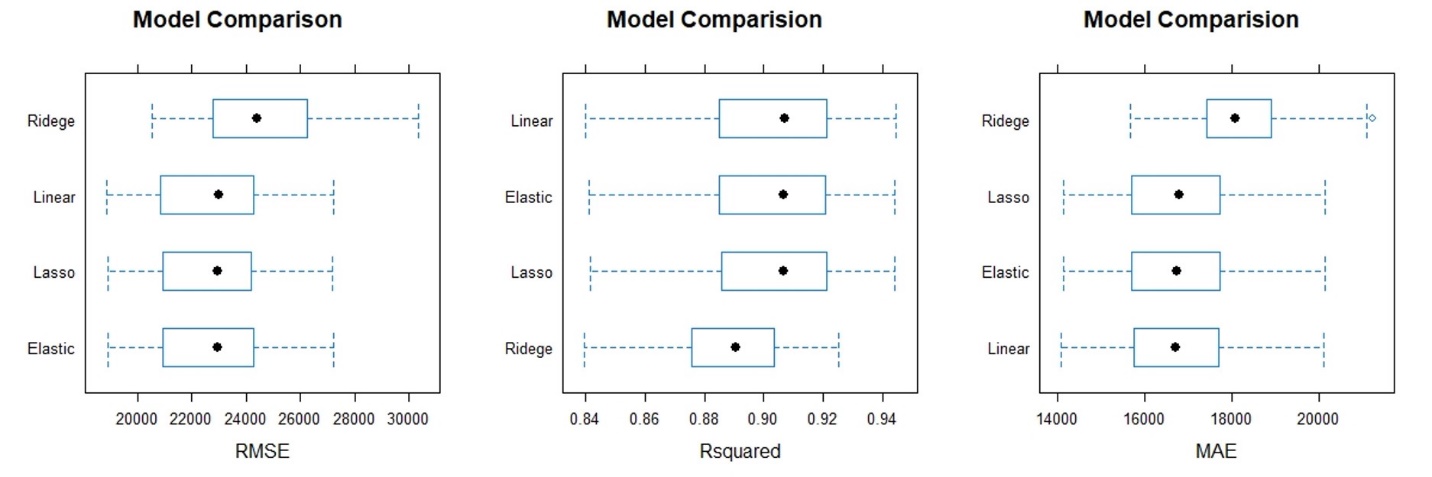
[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

Other than Ridge Regression, the rest three performed very much at the same level.

Summary of Models

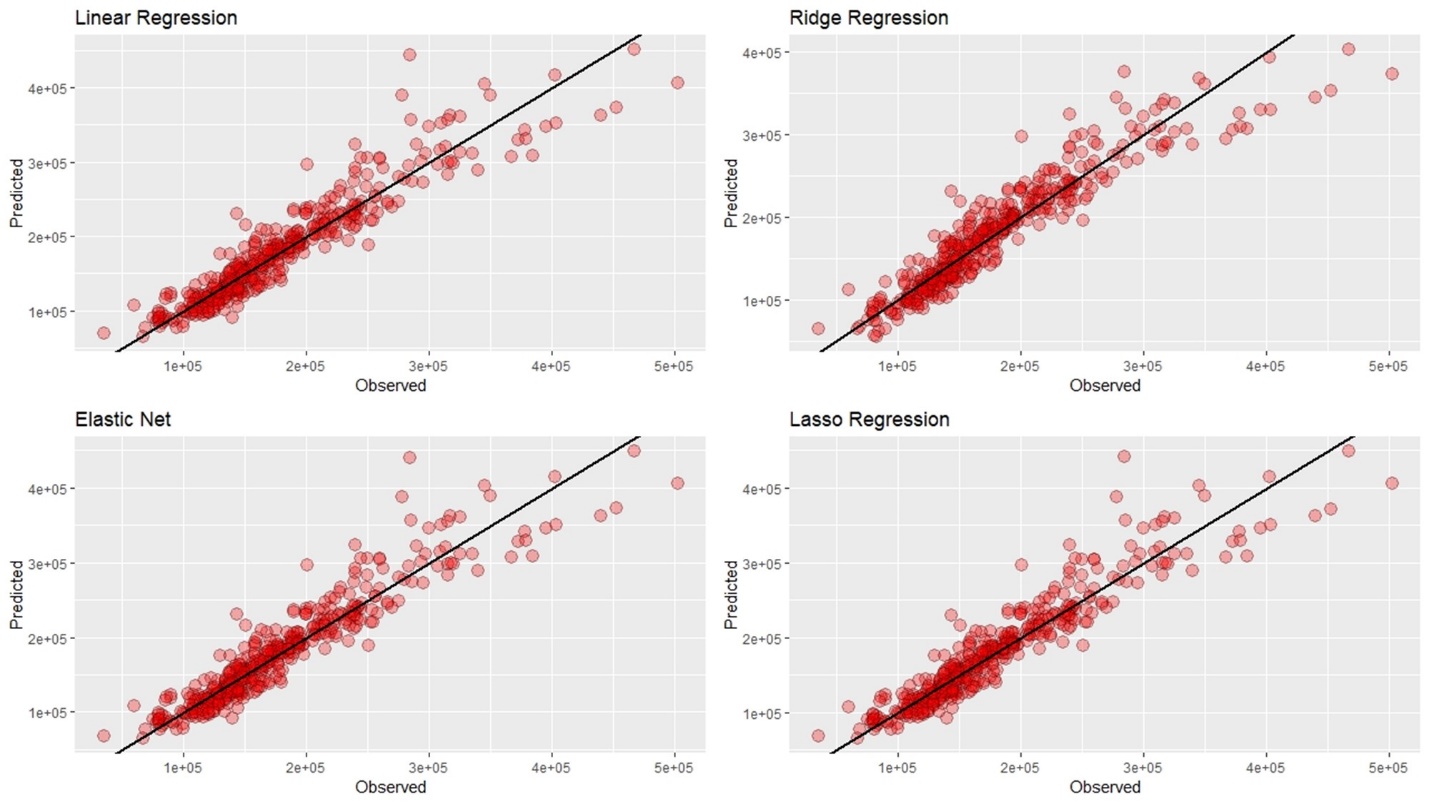


Boxplots



Predicted vs. Observed

Placing all four models together, Elastic Net apparently favorited Lasso Regression and the pattern are almost identical. While Linear, Lasso, and Elastic Net all have a very similar pattern, the color nevertheless shows there were subtle differences in density.



3D Scatter Plots

[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

No Machine Learning article should finish without a 3D plot. Here are three.

Model Comparison (RMSE)

Model Comparison (Rsquared)

Model Comparison (MAE)

Next

With a baseline model in place, the fun has just got started. Using test.csv, the submission file provided by Kaggle, start fine-tuning and improving the model, submit and score.

Closing Thoughts

[(back)](file:///N:\..ML.lab\ML.lab\project\yung.chou\yc.housing.rtm.html#back)

Considering this model employed just multiple linear regression, I was surprised that the scores turned out to be higher than expected, based on a few submissions I have done. Linear regression is conceptually simple and relevant to many activities happening in our daily life. We all do linear regression in our mind when making a purchase. Is this expensive or cheap? Every time, we ponder that thought, we are doing linear regression in some shape and form.

We must however not mistakenly and carelessly assume linear regression is as simple as it appears, as I have learned from my own mistake. There is much to investigate and learn from linear regression. Ordinary Least Square (OLS) which linear regression is built upon is too fundamental to overlook. The simplicity of OLS offers a clear strategy and enables Machine Learning algorithms to describe the combining effects of a set of predictors based on the distance. The concept of residuals is simple, approach straightforward, and objective clear. Ultimately, we want to minimize the distance of what is observed and what is predicted. This distance is our cost or error function.

There are a few options to continue the development. Tree-based models, ensemble learning, further refining and optimizing the data, more feature engineering, etc. are all applicable. With these many variables, a tree-based model should have a good story to tell. Which is what I plan to try next.

<[Read more Yung’s articles.](http://yungchou.wordpress.com)>