House Prices Prediction in King County

Jiajun Zhang

December 4, 2020

1 Introduction

The goal of this project is to build predictive machine learning models in order to make prediction on house sale prices in King County, USA. Data is found on Kaggle and it can be downloaded through the following link. It includes the information of houses sold between May 2014 and May 2015. The full description of the dataset is available here, and it has the meanings of all the variables. For example, the variable of *view* implies an index from 0 to 4 of how good the view of the property is. Since this is a classical regression problem, there are certainly many approaches we can use to achieve our goal. In this project, we will see some of the methods and their performances respectively. The source code for this project has been posted on RPubs and Github.

2 Data Preparation

The original data contains 21613 observations and 21 features. By looking at the structure of dataset as shown in Figure 1, we can easily remove some of the variables from our analysis since they only provide auxiliary information, such as id, date, etc. Also, we notice that some variables are supposed to be categorical, but stored in numerical types. For example, the variable of *waterfront* indicates whether the house has water view, where 1 suggests yes and 0 otherwise. So, we will need to convert this into a categorical variable instead. Similarly, we can perform the same transformation on other variables as well. Then, it is always a good idea to ensure whether our data have any missing values or outliers.

```
## 'data.frame':    21613 obs. of  21 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : chr  "20141013T000000" "20141209T000000" "20150225T000000" "20141209T000000" ...
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ sqft_basement: int  0 400 0 910 0 1530 0 0 730 0 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

Figure 1: Structure of Housing Dataset

Figure 2 shows the Pearson's correlation between variables. This is worth mentioning because, from the result, we can clearly see that some of the variables are highly correlated with each other such as *sqft_above* and *sqft_living*. In many cases, we do not want our features in a model to have duplicate effects and it can easily cause overfitting. By definition, the variable of *sqft_above* indicates the square feet above ground, and thus it has similar meaning to *sqft_living*. Also, I have manually created a variable *age*, indicating the number of years after houses built or renovated. Finally, I have decided to use explanatory variables of *bedrooms*, *bathrooms*, *sqft_living*, *sqft_lot*, *floors*, *waterfront*, *view*, *condition*, *age*, *lat*, and *long* only.

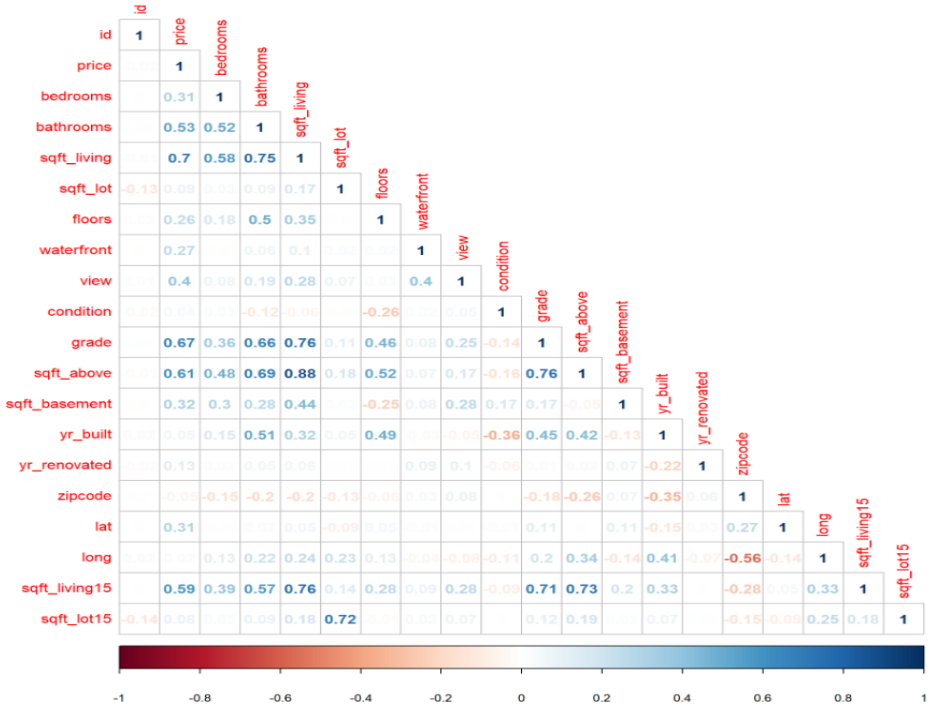

Figure 2: Pearson's Correlation

Data Visualization

After cleaning our data, we can generate some graphs to visualize some patterns between the response and predictors. Ideally, we expect an increasing relationship between *price* and *sqft_living* because a bigger house always implies it is more expensive. Similarly, we can say the relationship between the number of bathrooms and price of the house is also positively correlated. From Figure 3 and Figure 4, we can sort of see the patterns which satisfies our assumptions. There are certainly many other findings we can demonstrate just by exploring the relationships graphically. To keep this report simple, I will not list them all here.

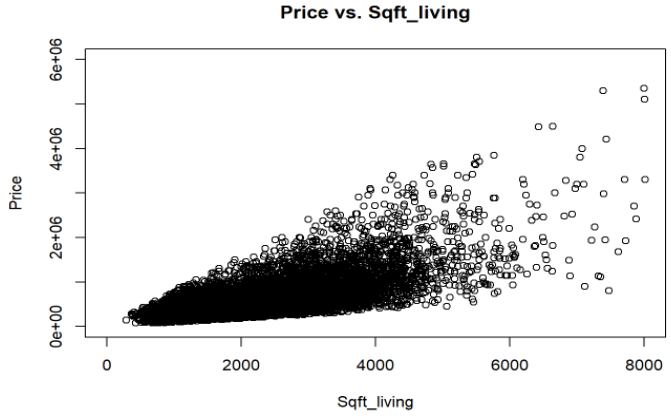

Figure 3: Relationship Between Price and Square Feet Living

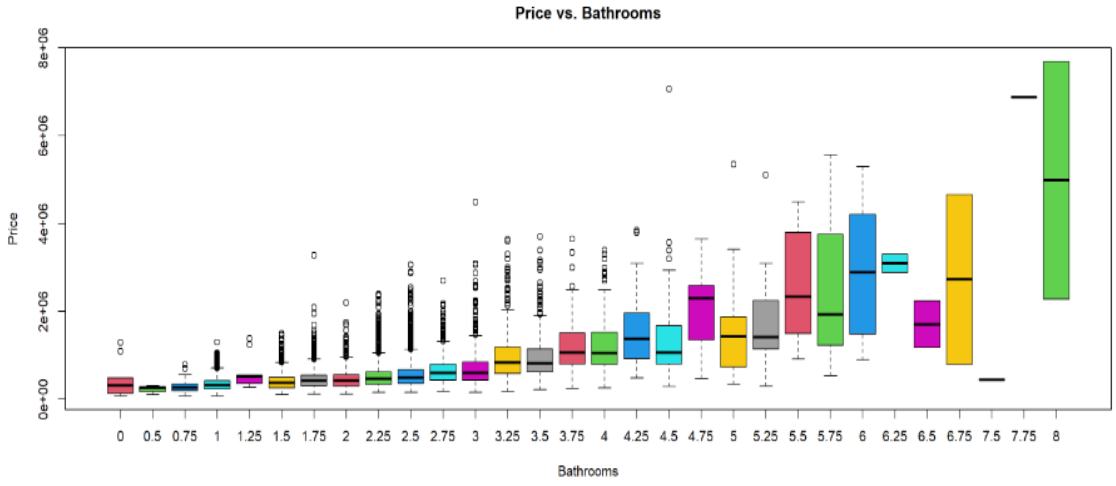

Figure 4: Relationship between Price and Number of Bathrooms

Train-Test Split

Before building our statistical models, we first need to split our data into a training set and a validation set. In this project, I used 80% as training data and we will still have about 4300 observations to test. This step is important because we want to evaluate our models on a local test set before actually making predictions.

3 Data Analysis

Data Preprocessing

First, we have scaled and centered all the numerical explanatory variables around 0. This is important because we can prevent a variable from outweighing another due to the difference in units. For example, the variable of *sqft_living* has a larger range weight than *bathrooms* before scaling.

Linear Regression

After separating our data, we can first attempt the method of linear regression. However, before building our model, we need to make sure whether the assumptions of linear regression are correct such as normality and homoscedasticity. Figure 5 suggests that our data is not normally distributed and the residuals do not seem to have a constant variance. Since the assumptions are violated, we cannot directly apply the method and our response needs some transformation.

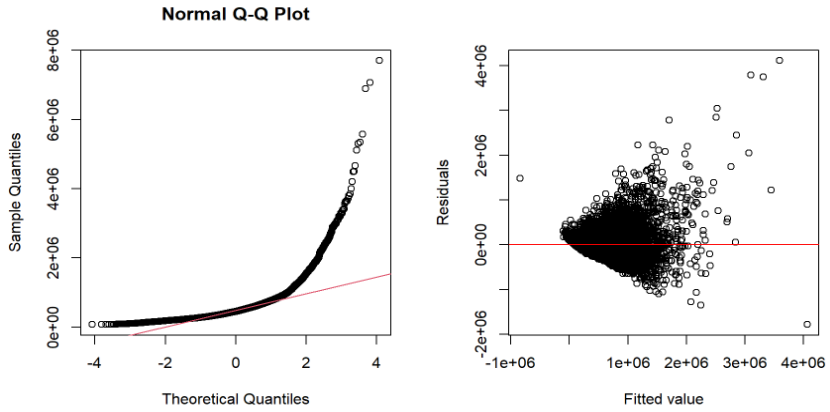

Figure 5: Model Diagnostics

In this particular example, I used the method of Box-Cox and it is basically a power transformation that selecting the optimal λ by maximizing the log-likelihood [1],

$$L(\lambda) = -\frac{n}{2} log(\frac{RSS_\lambda}{n}) + (\lambda - 1) \sum log(y_i).$$

Now, our linear model becomes $y^\lambda = y^{-0.03} = \beta_0 + \beta_i x_i$ where i = 1, ..., 11 since the optimal λ selected is -0.03. Then, by checking the model diagnostics again, we see that the conditions of normality and homoscedasticity have improved a lot as shown in Figure 6. We first try to fit the model with all the explanatory variables and perform a hypothesis test to each of the variable. This suggests that if one of the predictors had a p-value greater than 0.05 from the summary output, then we do not reject the null hypothesis $\beta_j = 0$ and conclude that the variable does not have a significant linear relationship with the response. From the summary output I obtained, only the predictor of *condition2* has a p-value larger than 0.05 and 2 implies the second level because the variable is categorical. This can be removed if we were to perform one-hot encoding on the categorical variable; however, in this case, I do not think whether it is a big deal by simply removing one level from the model.

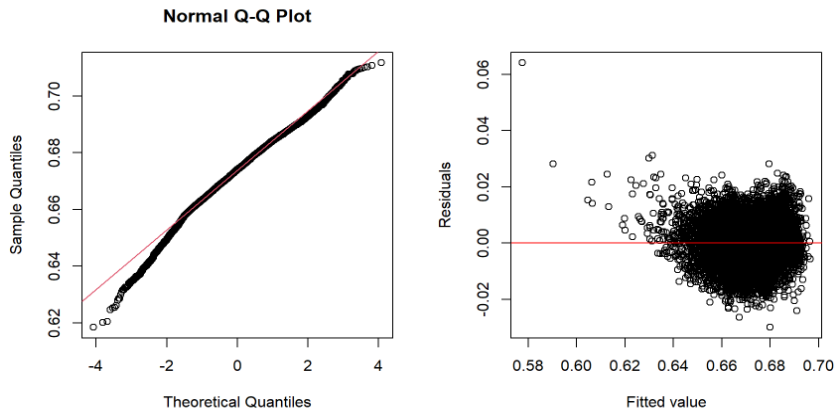

Figure 6: Model Diagnostics After Box-Cox Transformation

Also, the summary output suggests our full model has a R^2 value of 0.7. Then, we can perform a 10-fold cross validation on the training to verify whether our model performance has changed on some new test data. Since the result did not change much, we then use the model to make predictions on the validation set. There are many ways to justify the model accuracy and, in this project, I will use the Root

Mean Square Error(RMSE). So, the linear regression with all features provides a RMSE of 238338.4. In particular, we will use this number to compare with other models later. Moreover, we should always ensure that our regression model does not have the issue of multicollinearity. This can be verified by looking at the VIF values for all predictors. In this example, we do not have such an issue.

Regularization

Since we are able to use a standard linear model, we can then perform some shrinkage or regularization methods to see whether there are improvements to our model. And specifically, regularization prevents our model from overfitting. There are two well-known methods: LASSO and ridge; sometimes, these are referred to L1 and L2 regularization. Particularly, we can use LASSO for feature selection to discard all the unimportant features.

The idea of two methods are similar since they are both trying to find the optimal λ that adjusts the penalty term better in the loss function. LASSO has the loss function written as $RSS + \lambda \sum_{j=1}^{p} |\beta_j|$, where j indicates the columns in our data. On the other hand, ridge regression has the form of $RSS + \lambda \sum_{j=1}^{p} \beta_j^2$[3]. The selection of λ is important here since it affects the values in slopes(β_j). If λ is zero, it is essentially a OLS problem. When λ is large, it makes coefficients zero and hence it will cause the problem of underfitting. As suggested in Figure 7, when we increase the value in λ, we see more coefficients are converging to zero. The lines in different color and the number on the left indicate different coefficients. In this particular example, we find that coefficients 3 and 16, *sqft_living* and *lat*, in green and blue slowest converge to zero which indicates that they are the most significant features in the linear model. So this demonstrates and proves that we are able to perform feature selection using LASSO by tuning the values in λ.

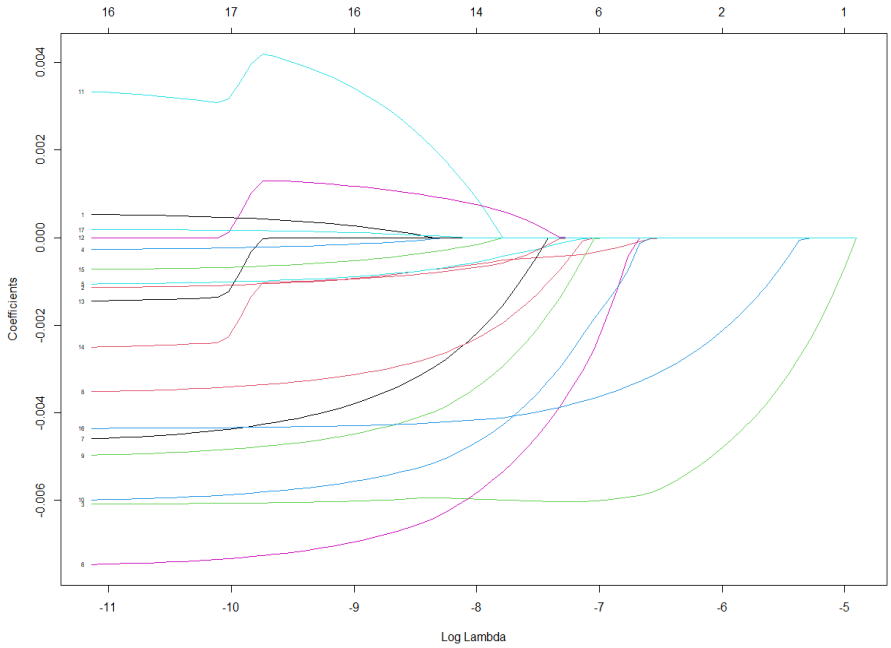

Figure 7: LASSO Convergence

Ridge regression has a similar meaning in terms of the cost function and selection of λ. However, in this case, we are not able to force the coefficients to be zero. When λ gets larger, the effect of the shrinkage penalty grows, and the coefficients can only get closer to zero but not exactly zero[2]. Apparently, the convergences of ridge regression coefficients are slower as shown in Figure 8 and they do not shrink all the way to zero. So, ridge regression penalizes the variables with minor contribution to the model by having their coefficients close to zero.

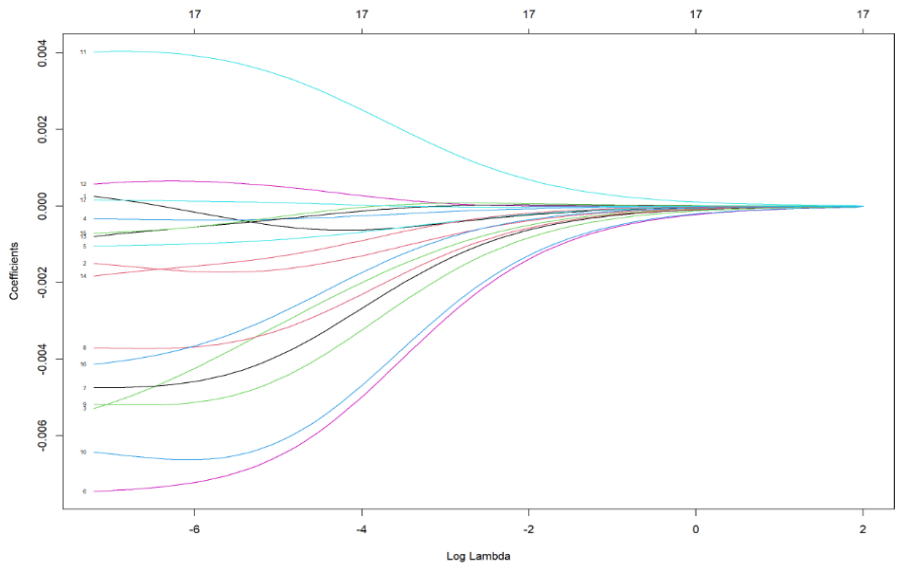

Figure 8: Ridge Convergence

By tuning different values in λ, we will certainly get different results in coefficients and model performances. As shown in Figure 9, the idea is to find the optimal λ to minimize the test error using cross validation on the training. In this case, if we were to increase the values, the MSE would get larger. So, the optimal value is around e^{-11}.

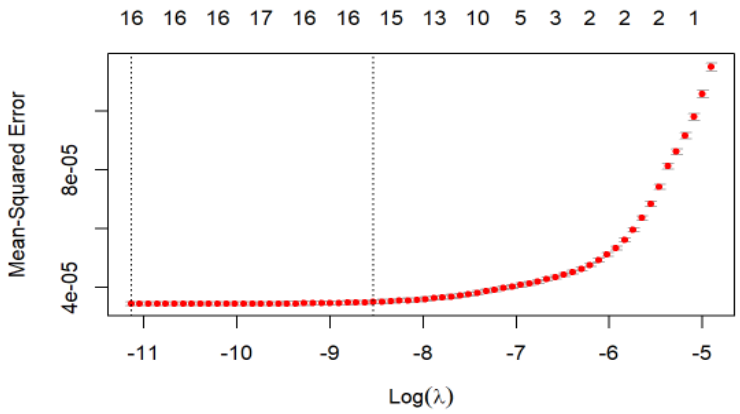

Figure 9: Parameter Tuning

Since the obtained R^2 values are all about 0.70, we may think that a linear relationship does not explain the data well enough. Personally, I am not a fan of polynomial regression, even though it might work well in this particular example. The method itself is a bit difficult to interpret and we need to pay careful attention to the choice of degree, since it can easily cause overfitting. On the other hand, we can attempt some tree-based models. Specifically, I tried to use regression tree and XGBoost tree which is also a type of ensemble methods.

XGBoost Tree

The method of regression tree provides a best result of 224030.2 in RMSE and 0.627 in R^2. I have gave up the method of random forest given that it is really time consuming on my computer. Then, since XGBoosting requires numerical data input only, we first need to perform one-hot encoding on the categorical variables. Due to the page limits of this report, we will not see and explain the whole procedure of XGBoosting. I would basically say it is a more enhanced approach than bagging. Also, unlike linear regression, it is also hard to interpret; however, the trees built in XGBoosting algorithm are not as complicated as random forest.

Tuning the hyperparameters in XGBoosting is a big problem since there are quite many. In this case, I have used the default hyperparameters and a 10-fold CV training. The result shown in Figure 10 has significantly improved than the results from linear regressions with regularization. This amazed me a lot since I have not previously worked with XGBoosting method. The result could potentially be better if we were to spend some time on tuning the hyperparameters. Since I have no experience with it yet, I think by following the procedures in this tutorial, we can improve our model better.

```
## eXtreme Gradient Boosting
##
## 17290 samples
##    11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 15562, 15560, 15561, 15560, 15562, 15560, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   140488.3  0.8544111  74309.33
##
```

Figure 10: Result of XGBoosting on Training

At the end, we make our predictions using XGBoosting and obtain a R^2 value of 0.894 and a RMSE of 119420. This result may need further investigation and it can potentially be overfitting, given that both values have significant improvements than the results from training in Figure 10.

4 Model Comparisons

So far, we have seen several different approaches and their performances are shown in Table 1.

5 Conclusion

Although there are many other methods we could have used in this project, I think this housing dataset is especially good for building different algorithms in regression. Previously, I have focused too much on finding a linear relationship and reg-

Model Performances		
Methods/Models	Rsquared	RMSE
Linear Regression	0.702	238338.4
Linear Regression with L1 Regularized (LASSO Regression)	0.705	236769.2
Linear Regression with L2 Regularized (Ridge Regression)	0.702	221165
Regression Tree	0.627	224030.2
XGBoosting	0.894	119420

Table 1: Model Performances

ularization. In this particular project, we have seen that linear regression with L2 regularization slightly improved the linear model in terms of RMSE but not as strong as XGBoosting. In nowadays machine learning project, ensemble methods are often being used and one of the most famous boosting methods is XGBoosting. I believe that starting from this project, I will need to begin applying the method better and experiment with it more in the future.

References

[1] David Dalpiaz. *Applied Statistics with R Chapter 14 Transformations*, 30 Oct. 2020. https://daviddalpiaz.github.io/appliedstats/transformations.html.

[2] Kassambara. *Penalized Regression Essentials: Ridge, Lasso Elastic Net*, 03 Nov. 2018. http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/153-penalized-regression-essentials-ridge-lasso-elastic-net/.

[3] Anuja Nagpal. *L1 and L2 Regularization Methods*, 14 Oct. 2017. https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c.