House Price Analysis and Prediction for Investing in King County

Michael Zhuang

May 11, 2020

ABSTRACT

This paper is going to analyze the house prices in King County, Seattle according to various attributes of the house units, and try to find the best way to predict the house prices. I will briefly explain 10 different model and compare their performance using mean squared error in order to select the best model for prediction. The observed result based on my study indicates that the boosting model after parameter tuning performs better than all the other models I build in predicting the house prices.

INTRODUCTION

Nowadays, more and more people would like to invest in real estate, which is expected to be safer and more profitable. King County as one of the most populous county in Washington with over 2.2 million populations in 2018[1] attracts hundreds of investors with no doubt. One of the most important thing that these investors are focusing on is the trend of house sale prices in King County and what factors may have significant effect on the prices. To predict future house sale prices is difficult but a thorough statistical analysis could help people to have some ideas about the relationship between prices and properties of the houses.

This paper uses the King County housing data from Kaggle, where can be found in *https://www.kaggle.com/harlfoxem/housesalesprediction*. This dataset contains house sale prices for King County in Seattle including 21613 house units sold between May 2014 and May 2015 with their sale prices, the number of bedrooms, the number of bathrooms, the area of living space, and other seventeen properties of the houses.

In this paper, I build ten different models using 80 percent of the data as training set and test the models on the remaining 20 percent. To be more precise, all models use the exact same training and test set. According to O'Farrell's 2018 paper about comparison of data mining model for prediction using the same data[2], he shows that linear regression has a better performance than KNN and boosting model in this dataset. However, I also select seventeen independent variables for model construction but with a little difference, for example, I use latitude and longitude instead of zipcode. Not only because of this, I also introduce some extra steps to enhance the performance of some models, which leads to a different conclusion than O'Farrell's. For my linear regression model, I first build 5 models using different independent variables, and choose the model that has the highest R-squared as my best LR model. Also, I include a LR model with all the seventeen independent variables for comparison. For all the other models: KNN, Ridge, Decision Tree, Bagging, Random Forest, Boosting, XGboost, and Neural Networks, I use the same seventeen independent variables except for Lasso model which performs variable selection automatically. The best model of this paper is the model with the smallest mean squared error in test set, and I will show the importance matrix of the best model for further analysis and discussion.

ANALYSIS

Summary Statistics

First from the summary statistics of the variables, the extreme values are very far from the first and third quantile for house price, and so do the other variables. It is possible for the existence of outliers in this situation. Also, the median and mean are significantly different from each other for sqft_lot, which is the area of the lot. It shows that the distribution of this variable may be extremely right-skewed.

Specifically, the distribution of price is right-skewed, which means most prices are concentrated on about 200 thousand dollars to 1.2 million dollars. Also, the distribution of number of bedrooms is right-skewed, which means most houses have only about 2 to 5 bedrooms. This is reasonable because most housing units in this area are apartments and houses. The distribution of number of bathrooms is a little left-skewed, which means most house units have more than 2 bathrooms. The distribution of area of living space is right-skewed as well, which means most area of living space is concentrated on about 800 square feet to 4000 square feet. The distribution of area of lot is very right-skewed, which means most houses have area of lot less than 20 thousand square feet.

For correlation between the dependent variable and the independent variable, the box-plot shows that correlation between the number of bedrooms and the house price is first positive and then becomes negative. Also, for the house with 5 or 6 bedrooms, they have larger range of the price. For houses with 8 or 9 bedrooms, they have greater interquartile range. The reason behind this is that initially the larger house with more bedrooms would have higher sale price. While, the demand starts to decrease when there are too many bedrooms, for example 8 bedrooms, so the sale price becomes more flexible. The correlation between the number of bathrooms and the

house price is overall positive but becomes noise after 4.25. This probably because there are less samples with bathrooms greater than 4. Moreover, there are more samples that have higher sale price than the mean price, which is indicated as red point, of houses with the same number of bathrooms for houses with less than 4 bathrooms, and it becomes fewer for houses with greater than 4 bathrooms. Also, the price range is obviously larger for houses with larger number of bathrooms. The correlation between the area of living space and the house price is overall positive, which means the larger area of living space, the higher house price. Also, it is noticeable that most data are concentrated with the area of living space below 5000 square feet. While, there is no clear relationship between the house price and the area of the lot, so it is better to try to use log of price instead of price for our Y value.

After using log transformation, it looks that most observations are concentrated in the range of 0 to 250000 square feet for area of the lot, so it may be helpful to zoom in to this range to see if there is any relationship. Even though by narrowing the range for square footage of the lot, it's still hard to find any significant relationship with the house price. Log(price) seems to have a cumulative sum limit of 13.5. Also, as the area of the lot becomes larger, the variance of percent change in price becomes smaller, which means the volatility of price is high for smaller housing units. However, it still possible that this variable can jointly correlated to house price with the other variables together.

It is noticeable that the correlation between the number of bedrooms and average house price becomes negative after the number of bedrooms is over 8, and the correlation between the number of bathrooms and average house price becomes noise after the number of bathrooms is greater 6.25. This may because there are few observations with bathrooms more than 6, and we could treat these observations as outliers if needed. Also, the correlation matrix indicates that the

area of living space has relatively larger effect on the house price, and the area of lot singly has almost no effect on the house price.

Linear Regression

The linear regression is using ordinary least squares for estimation of parameters in model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + e$, where $e \sim N(0, \sigma^2)$. By minimizing the sum squared error, one can find the best estimator of betas.

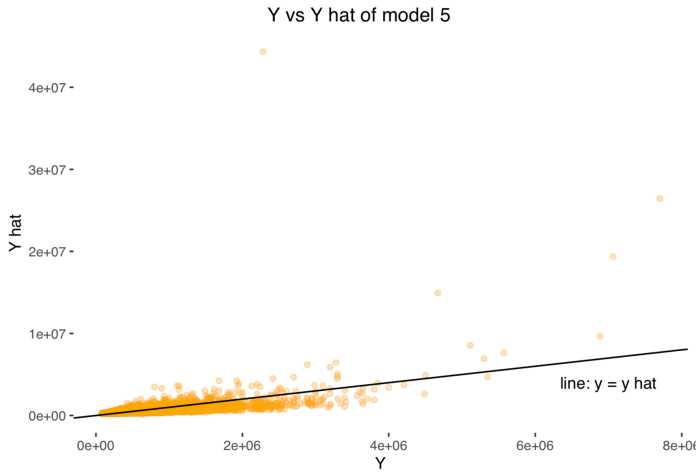
Among the five models, I consider the fifth model as my best model since both R-squared and adjusted R-squared of that model are highest.

```
model5 = lm(log(price) ~ bedrooms + sqft_living + condition + factor(waterfront) + floors, data = house
summary(model5)

##
## Call:
## lm(formula = log(price) ~ bedrooms + sqft_living + condition +
##     factor(waterfront) + floors, data = house)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.96771 -0.27322  0.01435  0.25363  1.79880
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          1.191e+01  1.845e-02  645.66   <2e-16 ***
## bedrooms            -4.651e-02  3.311e-03  -14.05   <2e-16 ***
## sqft_living          4.030e-04  3.539e-06  113.87   <2e-16 ***
## condition            8.792e-02  3.995e-03   22.01   <2e-16 ***
## factor(waterfront)1  5.896e-01  2.916e-02   20.22   <2e-16 ***
## floors               1.001e-01  5.126e-03   19.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3674 on 21607 degrees of freedom
## Multiple R-squared:  0.5135, Adjusted R-squared:  0.5134
## F-statistic:  4561 on 5 and 21607 DF,  p-value: < 2.2e-16
```

The coefficients of the fifth model show that for every unit increase of the number of bedrooms, the house sale price will decrease by about 4.54% (exp(-0.04651)-1); for every unit increase of square footage of the living space, the house sale price will increase by about 0.04% (exp(0.000403)-1); for every unit increase of house condition rank, the house sale price will increase by about 9.19% (exp(0.08792)-1); for every unit increase of the number of floors, the house sale price will increase by about 10.53% (exp(0.1001)-1); moreover, when the other conditions are the same, the house sale price of house unit on the waterfront is expected to be about 80.33% (exp(0.5896)-1) higher than the price of house unit that is not on the waterfront.

The coefficients are all significantly different from zero because the p-values of their t-test are much smaller than 0.01. Also, the F-statistic is 4561 on 5 and 21607 degrees of freedom and is significant in this model, which means all variable in this model are jointly significant.



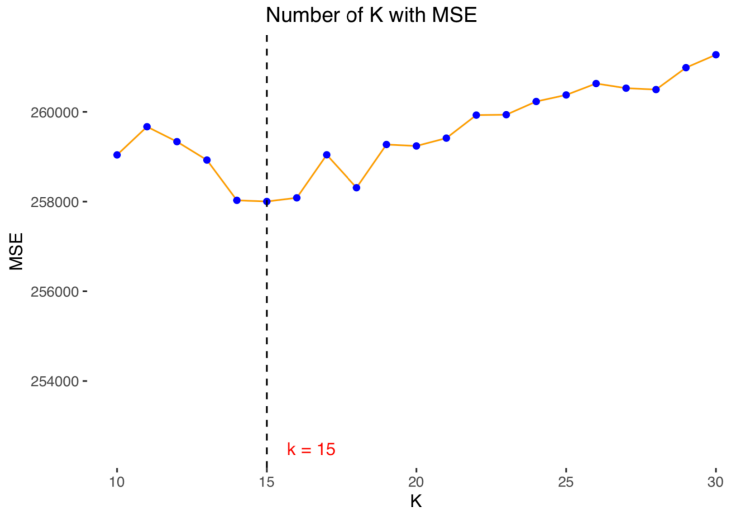
According to the plot of "Y vs Y hat", I find that the overall trend is correct for the outcomes of my model, but the variance of my model is increasing as Y increases. Also, when Y becomes bigger, my model tends to underestimate most of the data. This indicates that there is heteroscedasticity with my regression model. Also notice that, there is an outlier where actual Y price is about 2.5 million dollars.

I will also apply the linear regression model (OLS model) using all independent variables for comparison in the end.

K Nearest-Neighbors (KNN)

KNN is a method that to predict a point according to its K nearest neighbors. It is a non-parametric method which we do not assume the functional form for the relation between Y and X. Here I use KNN Regression model since my dependent variable is continuous, and the functional form looks like this: $\hat{y}(x_0) = \hat{f}(x_0) = \frac{1}{K}\sum_{x_i \in N_0} y_i$.

By comparison, the KNN model with the lowest MSE for the testing data is when k equals 15.



Here, the KNN model is constructed using the same 5 predictors as the linear regression above, and I will apply the KNN model using all the independent variables for comparison in the end.

Ridge Regression

Ridge regression adds a shrinkage method to the regular regression and try to shrink the parameters to constrain their variance. In a ridge regression, we want to minimize the function: $\mathcal{L} = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2$, where λ is a hyper parameter which determines how strict the constraint is.

By tuning the model with λ, the best λ is the λ within 1 standard error of the minimum λ, which is 0.059 here. Then, I can compute the parameters with this λ.

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)   -5.390964e+01
## bedrooms      -6.647239e-03
## bathrooms      6.478295e-02
## sqft_living    7.808436e-05
## sqft_lot       3.425349e-07
## floors         6.256187e-02
## waterfront     3.433283e-01
## view           6.083482e-02
## condition      6.244753e-02
## grade          1.314735e-01
## sqft_above     7.151501e-05
## sqft_basement  7.981716e-05
## yr_built      -2.568574e-03
## yr_renovated   4.474867e-05
## lat            1.247386e+00
## long          -8.765360e-02
## sqft_living15  1.101827e-04
## sqft_lot15    -1.989293e-07
```

Notice that *waterfront* and *lat* have relatively higher coefficients than the others, which means these two variables are more important when predicting sale price of the house in a ridge model.

Lasso Regression

Lasso regression is similar to ridge regression but uses a different shrinkage method. In a lasso regression, we want to minimize the function: $\mathcal{L} = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j|$.

The best λ is the λ within 1 standard error of the minimum λ, which is 0.0068 in this lasso regression. Then, I can compute the parameters with this λ.

```
## 18 x 1 sparse Matrix of class "dgCMatrix"
##                             1
## (Intercept)   -4.669582e+01
## bedrooms       .
## bathrooms      5.995450e-02
## sqft_living    1.394518e-04
## sqft_lot       9.066175e-08
## floors         4.503209e-02
## waterfront     3.158965e-01
## view           6.134603e-02
## condition      5.551492e-02
## grade          1.597310e-01
## sqft_above     .
## sqft_basement  .
## yr_built      -2.927642e-03
## yr_renovated   2.858018e-05
## lat            1.324615e+00
## long          -3.420569e-03
## sqft_living15  9.149763e-05
```

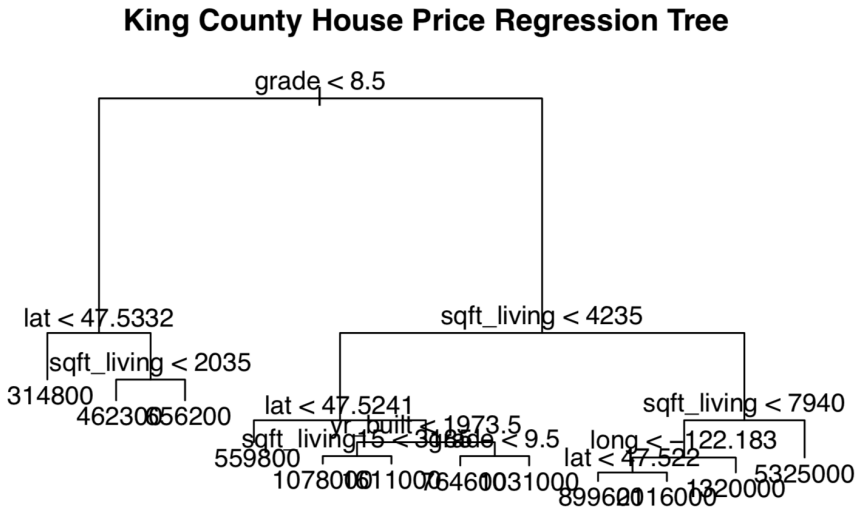
Lasso regression is different from ridge regression that it automatically performs variable selection by setting some variables exactly to zero so that they can be omitted from the model.

Decision Tree

In decision regression tree, it first divide the predictor space into N mutually exclusive regions: R_1, …, R_N, and then predict the outcome for all the observations that fall into R_N by computing the mean of the outcome for the training data points located in that region. To pruning a tree, we find a T that minimize the following RSS for a non-negative value of α:

$\sum_{m=1}^{|T|} \sum_{i \in R_j} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$, which α is a tuning parameter for complexity.

Using cross-validation for pruning, the best tree I get is a 12-node tree.



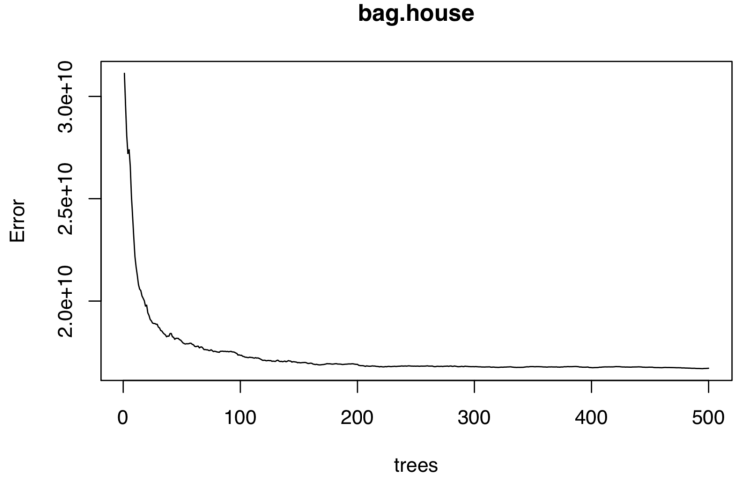
In this tree model, it first uses grade for first split with threshold of 8.5. For houses with grade smaller than 8.5, it split again in lat with threshold of 47.5 and then split in *sqft_living* with threshold of 2035 for houses with lat greater than 47.5. For houses with grade higher than 8.5, most notes are also split using lat or *sqft_living*, except one using long for split.

In order to improve the performance of my tree model, I decide to use bootstrap to build a new tree model. Bootstrap is a resampling method that helps to enlarge the dataset and to reduce the bias. When using bootstrap and fit 100 different trees, I use best of 9 for pruning, since 12-node tree may bigger than some tree size, and the plot in previous part indicates that the difference of cross-validation error for 9-node tree and 12-node tree is not significant. By averaging the 100 trees, the result shows that the tree model using bootstrap performs better than the single decision tree.

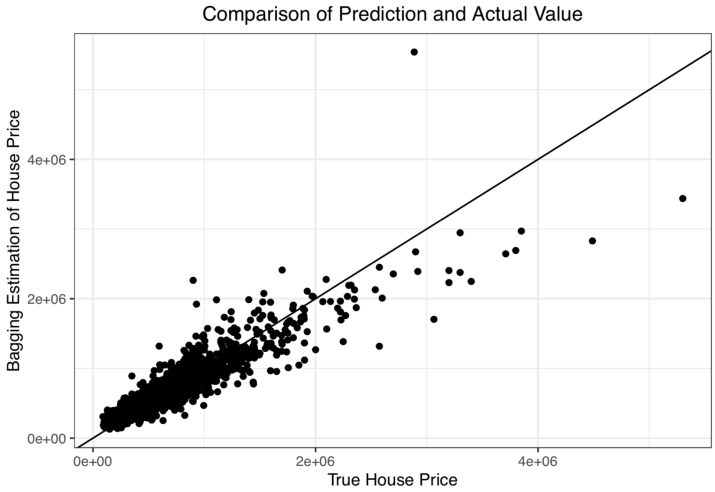
Bagging

Bootstrap aggregation or bagging is the average of multiple trees generated from bootstrap samples, using the following functional form: $\hat{f}_{bag} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(X)$. Moreover, instead of using cross-validation, we use out-of-bag error to measure test error in bagging.

The bagging model considers all 17 predictor variables on each split node of the tree. I also set the model to generate 500 trees and trace the out-of-bag error for every 100 trees.



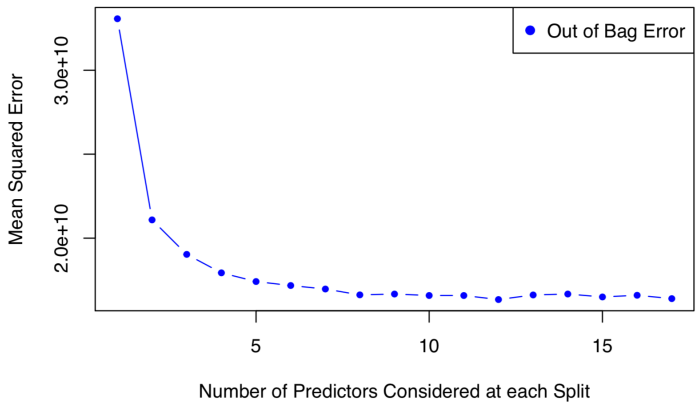
The plot shows the trend of out-of-bag error with increasing number of trees. It looks like the error drops most sharply when the number of trees is about 20 and then drop slowly after the number of trees is greater than 100.



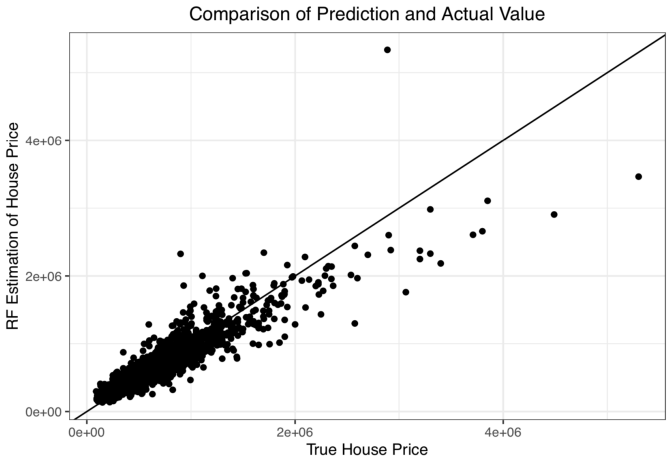
According to the plot of prediction vs actual values, it shows that the bagging model tends to underestimate the house price when actual house price is larger than 2 million dollars.

Random Forest

Doesn't like bagging, random forest model considers a random subset of predictors instead of all predictors when splitting a tree, which prevents some very strong predictors domain the prediction.



Based on the out-of-bag error trend plot, the MSE drops dramatically after the model chooses 2 predictors at each split. Notice that the change of MSE becomes non-significant after considering 9 predictors at each split, and hits the minimum at about 12.

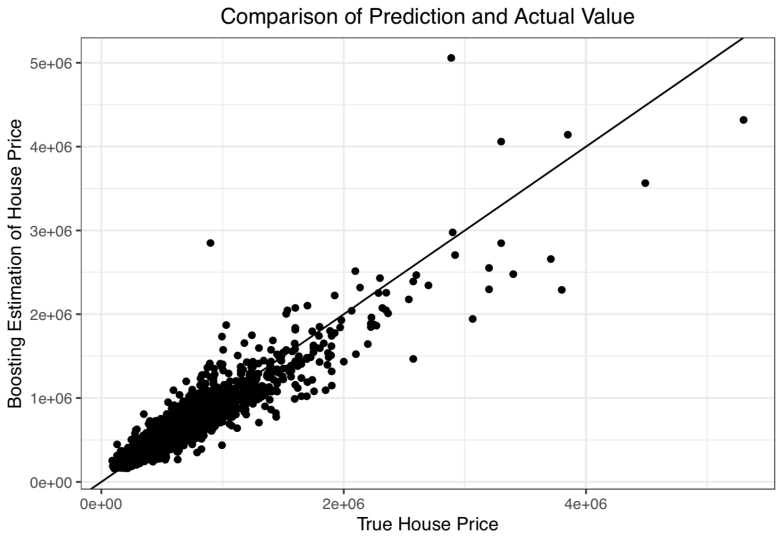


According to the plot of prediction vs actual values, it shows that the random forest model also tends to underestimate the house price when actual house price is larger than 2 million dollars.

Boosting

Boosting is a general method and this paper uses regression tree as its method. For boosting trees, each tree uses the information from the previous tree to grow sequentially. It does not involve bootstrap sampling, but each tree is fitted on a modified version of the original data, which is the residuals from the last period. The functional form looks like this: $Y = \lambda\hat{f}_1(x) + \widehat{\lambda f_2}(x) + \lambda\hat{f}_3(x) + \cdots + \lambda\hat{f}_B(x) + r_B$, where λ is the shrinkage parameter.

By default, my boosting model uses shrinkage of 0.01, depth of 4, and number of trees of 5000.

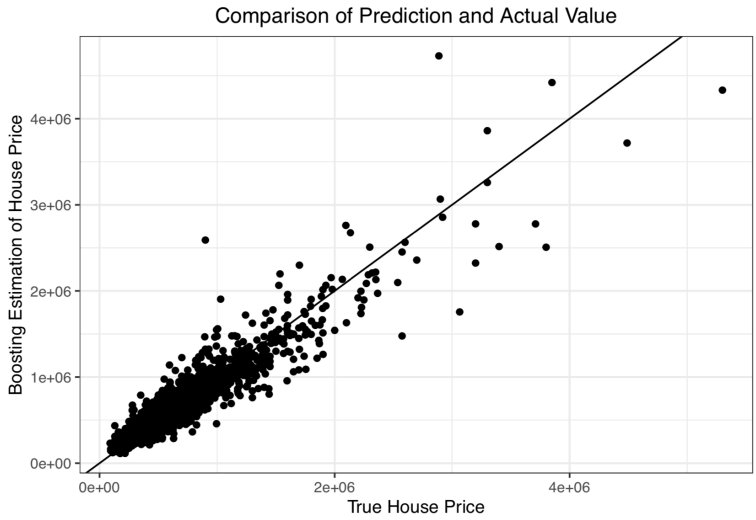


The prediction vs actual value plot of boosting model shows similar pattern as the bagging and random forest model when true house price is smaller than 2 million dollars, but its predictions are closer to the true value even when true house price is greater than 2 million dollars.

To further improve the performance of boosting model, tuning its parameters is required. By using grid search method, I find the best combination of its parameters.

```
##   shrinkage interaction.depth optimal_trees     min_MSE
## 1      0.05                 4          4570 15319072248
## 2      0.05                 5          3286 15530029704
## 3      0.10                 3          3040 15669533476
## 4      0.10                 5          2262 15828232272
## 5      0.10                 4          4987 15984720224
```

The best boosting model has learning rate of 0.05, maximum depth of 4, and optimal trees of 4570.

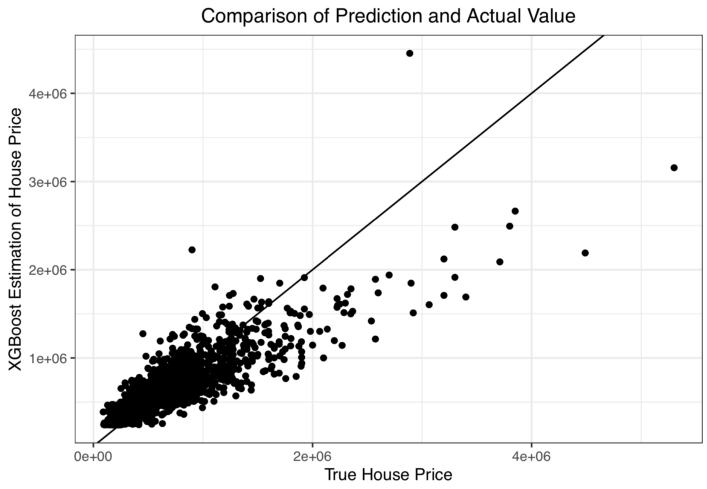


It is not very clear, but more predictions are closer to the true house sale price using this model. I will also compare its MSE with all the other models in the end to see its performance.

XGboost

As a modification version of gradient boosting, eXtreme Gtadient Boosting (XGboost) includes a regularization parameter and implements parallel processing. Its functional form becomes: $obj = \sum_{i=1}^{n} l\left(y_i, y_i^{t-1} + f_t(x_t)\right) + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$, where T is the number of leaves, and ω is the vector of scores on leaves.

My default XGboost model uses max depth of tree of 2, learning rate (eta) of 0.1, max number of trees of 40, and shrinkage of 0.



The XGboost model shows a similar pattern as the bagging and random forest model, but does not like boosting model, XGboost model with default parameters would still underestimate the house sale price when true value is greater than 2 million dollars.

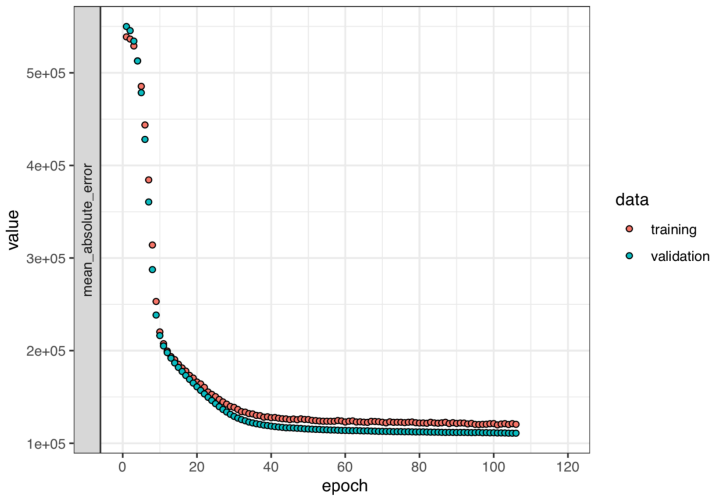
Neural Networks

Neural Networks is constructed by multiple hidden layers which are not directly observed, and we try to find $Y = f(g(\sigma(...(X))))$. For regression problem, each layer has a functional form: $f_k(X) = g_k(\beta_{0k} + \sum_{m=1}^{M} \beta_{mk} Z_m)$, k = 1,…K, where $Z_m = \sigma(\alpha_{0k} + \sum_{i=1}^{P} \alpha_{im} X_i)$, m = 1,…M. For $\sigma(.)$ in this paper, I use Rectified Linear Units ReLU functions which $\sigma(v) = \max(0, v)$. Neural Networks also use gradient decent and back-propagation to find the optimal value of the weight parameters.

In this paper, I build Neural Networks using two hidden layers with 64 nodes in each layer. There are 1152 parameters associated with the first hidden layer, and 4160 parameters associated with the second hidden layer. The number of total parameters is 5377.

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## dense (Dense)                (None, 64)                1152
## _________________________________________________________________________
## dropout (Dropout)            (None, 64)                0
## _________________________________________________________________________
## dense_1 (Dense)              (None, 64)                4160
## _________________________________________________________________________
## dense_2 (Dense)              (None, 1)                 65
## =========================================================================
## Total params: 5,377
## Trainable params: 5,377
## Non-trainable params: 0
## _________________________________________________________________________
```

I first set the model with 300 epochs, and tuning it with an early stop if there is no significant change in loss for 10 observations.



The plot shows that the tuning process stops when epoch is around 108, and I will use this model for my prediction.

Comparing the Results

I collect the MSE of all models in this paper to compare their performance using the same test set.

```
##                       MSE
## bestboost_mse 13448310571
## boost_mse     15539227261
## rf_mse        17119931365
## bag_mse       17384117713
## xgb_mse       32276171271
## nn_mse        35327628220
## boot_mse      39074829188
## lasso_mse     40525344017
## ridge_mse     40767907005
## ols_mse       41835648587
## tree_mse      45481239311
## knn_mse       68519428620
```

According to the comparison table, the best boosting model (boosting model after tuning parameters with grid search) performs the best with the lowest MSE. The default boosting model also performs well with the second lowest MSE. This indicates that the boosting model is a good method in predicting observations in this dataset. The KNN model performs the worst probably because there are too many independent variables in this dataset, which reduces the ability of finding the best neighbors, or sometimes called the "curse of dimension." However, it is also the fact that some models can increase their performance by tuning the parameters properly or using bootstrap, for example the MSE of Decision tree model decreases a lot after performing bootstrap (from tree_mse to boot_mse). While, as Gan states in his paper about data mining analysis using similar data of house prices in King County but for year 2012 to 2013 that "Neural Networks algorithm produced lower mean errors than Decision Trees,"[3] I also come up with the same result even after using bootstrap. In short, there is still probability that some models can perform better than the best boosting model after tuning perfectly.

```
##                          var    rel.inf
## sqft_living     sqft_living 32.3172438
## grade                 grade 19.5816489
## lat                     lat 14.6358833
## long                   long  7.7160688
## sqft_living15 sqft_living15  4.1756791
## yr_built           yr_built  3.5857504
## waterfront       waterfront  3.4664874
## sqft_above       sqft_above  3.3498005
## view                   view  3.0308952
## bathrooms         bathrooms  2.2937347
## sqft_basement sqft_basement  1.5386850
## sqft_lot           sqft_lot  1.4794385
## sqft_lot15       sqft_lot15  1.4005695
## condition         condition  0.4917389
## yr_renovated   yr_renovated  0.4827264
## bedrooms           bedrooms  0.2466872
## floors               floors  0.2069623
```

Also, the important matrix of the best boosting model indicates that the *sqft_living* and *grade* are the most influential variables. This is consistent with Mayer's project of introducing flashlight model using the same data of house prices in King County [4] that *sqft_living* is the most important variable when excluding *zipcode*.

CONCLUSION

Generally, from results of the basic statistical summary, the houses with larger number of bedrooms tend to have higher sale price, but with the number of bedrooms greater than 6, the sale price starts to fall. The number of bathrooms also has a positive relationship with the house price, which means the more bathrooms, the higher sale price; however, the relationship starts to fluctuate after the number of bathrooms greater than 6, and I need more data or more variables together to explain it. Area of living space is overall positively correlated with the house sale price, which the larger living space generally leads to higher sale price. While for the area of the

lot as a single predictor, I cannot find any significant relationship to the house sale price, but I don't exclude the possibility that it can be jointly significant with the other variables together.

Results of the linear regression and KNN model shows that the linear regression model with 5 independent variables is considered better than the other models with fewer independent variables, since it has higher R-squared and adjusted R-squared value which show that this model can explain about 51.35% variation of the house sale price. A KNN regression model with k equals 15 is also considered as the best model among the KNN models.

Furthermore, by comparing the mean squared error of different models, the lasso regression model has smaller MSE than ridge model, OLS model with all covariates, KNN model, and single tree model. Interestingly, when we apply KNN model to the entire dataset, it now has the highest MSE among all models. This reflects that KNN model does not perform well when data dimension is large, which is called the "curse of dimensionality". The tree model training with boot-strap shows better performance than the single tree model. This proves that boot-strap can help improve the performance of my model in this dataset, and I should consider using boot-strap in future experiment.

By generating more models, I notice that either bagging, random forest, boosting, or XGboost model has a better performance than ridge and lasso model. Bagging and random forest model give me similar prediction results and both of them tend to underestimate the true house sale price after the true house price is greater than 2 million dollars. Also, their important matrix both agree that latitude and longitude of the house unit are the most influential variables. However, the importance matrix of boosting and XGboost model show that the most effective variables are sqft_living and grade. I more agree with the result of boosting and XGboost model since boosting model gives the smallest test MSE among all models, and does not like bagging and

random forest model, boosting model well predicts the house price even when the true value is greater than 2 million dollars. One important thing is that the XGboost model with default parameters given by the XGboost package performs even worse than bagging and random forest model. Besides, by tuning the parameters of boosting model using grid search, I find the best boosting model containing parameters that learning rate of 0.05, maximum depth of 4, and optimal trees of 4570. This best model also successfully reduces test MSE of the original boosting model by 13%, and this proves that tuning parameters is such an important process when building a complex model.

Last but not least, the Neural Networks model with two hidden layers and 64 nodes in each layer has the best number of epoch around 108. While after comparing its performance with all the other models, it does not have the lowest MSE. In fact, the best model with the lowest MSE is the boosting model after tuning, and the worst model with the highest MSE is the KNN model. On the other hand, some model can be improved after properly tuning or applying boot-strap. This reminds me that what I can add to my future study, and let me notice that there are still much I can explore in tuning the models. While, so far the best boosting model is the most suitable model for predicting the house sale price in King County and area of living space seems like the most important factor that affects the house price.

REFERENCES

1. "U.S. Census Bureau QuickFacts: King County, Washington." Census Bureau QuickFacts, *www.census.gov/quickfacts/kingcountywashington*.
2. O'Farrell, Stephen (2018). Comparison of Data Mining Models to Predict House Prices. Academia.edu, *https://www.academia.edu/38358601/House_Price_Prediction*.
3. Gan, Victor, Agarwal, Vaishali, &Kim, Ben (2015). DATA MINING ANALYSIS AND PREDICTIONS OF REAL ESTATE PRICES. Iacis.org. *https://iacis.org/iis/2015/4_iis_2015_30-36.pdf*.
4. Mayer, Michael (2020). Using flashlight. cran.r-project.org. *https://cran.r-project.org/web/packages/flashlight/vignettes/flashlight.html*.