Predict House Price

Hafiz Khan

2/21/2021

Introduction

This report is the second part of the capstone project for ‘HarvardX: PH125.9x Data Science: Capstone’ course. The goal of this project is to analyse a dataset from Kaggle called ‘House price prediction’ collected from Washington state area. Each row of the dataset contains house price and several other attributes that can affect the price such as number of bedrooms and etc. The final product of the analysis is to predict whether the house price is above the median house price in Washington region or otherwise. The original dataset are splitted into two portion: 90% for training purposes and 10% for testing and validation purposes. The final model will be tested on a independent test dataset and its accuracy will be evaluated for each algorithm performed.

Methods & Analysis

The method and analysis segment is broken into several segments which are data download, data cleaning, data partition, data exploration and data analysis of the ‘House price prediction’ dataset. The detail of each segments will be describe in the report.

Data Download

The original dataset can be found in ‘https://www.kaggle.com/shree1992/housedata (https://www.kaggle.com/shree1992/housedata)’

```
library(tidyverse)
library(caret)
library(data.table)
library(gam)

url   <- "https://raw.githubusercontent.com/apiz8393/CapstoneHarvardX/main/house%20price%20datas
et.csv"
house <- read_csv(url)
```

```
## Parsed with column specification:
## cols(
##   date = col_datetime(format = ""),
##   price = col_double(),
##   bedrooms = col_double(),
##   bathrooms = col_double(),
##   sqft_living = col_double(),
##   sqft_lot = col_double(),
##   floors = col_double(),
##   waterfront = col_double(),
##   view = col_double(),
##   condition = col_double(),
##   sqft_above = col_double(),
##   sqft_basement = col_double(),
##   yr_built = col_double(),
##   yr_renovated = col_double(),
##   street = col_character(),
##   city = col_character(),
##   statezip = col_character(),
##   country = col_character()
## )
```

Data Exploration

The original dataset (edx) contains 4600 rows and 18 columns. Each row contains one record of house price and other important attributes related to it such as number of bedrooms, number of bathrooms, living sq-ft area, lot sq-ft area, number of floors, waterfront, view, condition level, sq-ft above basement level, sq-ft basement level, year built, year renovated, street address, city, state zipcode and country. The following code will show how the information above are produced:

```
# Number of rows and columns
paste('The dataset has',nrow(house),'rows and',ncol(house),'columns.')
```

```
## [1] "The dataset has 4600 rows and 18 columns."
```

```
# Columns name in the dataset
names (house)
```

```
##  [1] "date"          "price"         "bedrooms"      "bathrooms"
##  [5] "sqft_living"   "sqft_lot"      "floors"        "waterfront"
##  [9] "view"          "condition"     "sqft_above"    "sqft_basement"
## [13] "yr_built"      "yr_renovated"  "street"        "city"
## [17] "statezip"      "country"
```

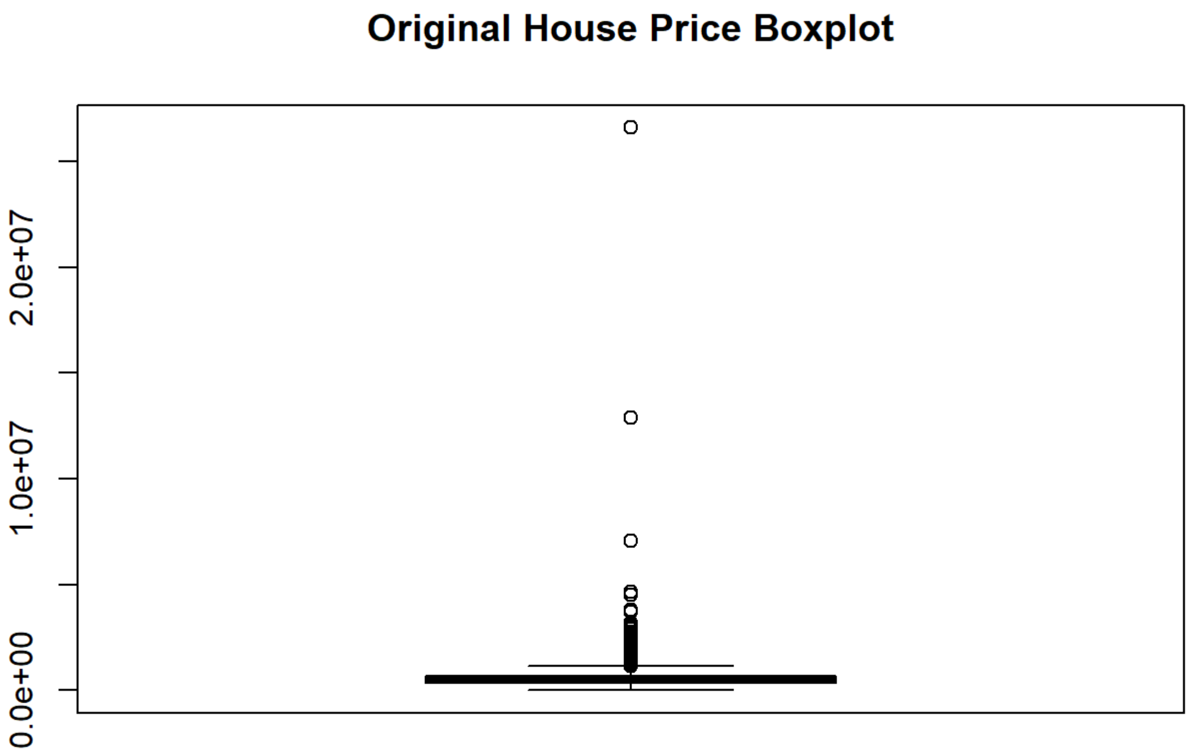
The structure of the dataset is shown below:

```
#Show internal structure of a R object
str(house)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 4600 obs. of  18 variables:
##  $ date         : POSIXct, format: "2014-05-02" "2014-05-02" ...
##  $ price        : num  313000 2384000 342000 420000 550000 ...
##  $ bedrooms     : num  3 5 3 3 4 2 2 4 3 4 ...
##  $ bathrooms    : num  1.5 2.5 2 2.25 2.5 1 2 2.5 2.5 2 ...
##  $ sqft_living  : num  1340 3650 1930 2000 1940 880 1350 2710 2430 1520 ...
##  $ sqft_lot     : num  7912 9050 11947 8030 10500 ...
##  $ floors       : num  1.5 2 1 1 1 1 1 2 1 1.5 ...
##  $ waterfront   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : num  0 4 0 0 0 0 0 0 0 0 ...
##  $ condition    : num  3 5 4 4 4 3 3 3 4 3 ...
##  $ sqft_above   : num  1340 3370 1930 1000 1140 880 1350 2710 1570 1520 ...
##  $ sqft_basement: num  0 280 0 1000 800 0 0 0 860 0 ...
##  $ yr_built     : num  1955 1921 1966 1963 1976 ...
##  $ yr_renovated : num  2005 0 0 0 1992 ...
##  $ street       : chr  "18810 Densmore Ave N" "709 W Blaine St" "26206-26214 143rd Ave SE" "8
57 170th Pl NE" ...
##  $ city         : chr  "Shoreline" "Seattle" "Kent" "Bellevue" ...
##  $ statezip     : chr  "WA 98133" "WA 98119" "WA 98042" "WA 98008" ...
##  $ country      : chr  "USA" "USA" "USA" "USA" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   date = col_datetime(format = ""),
##   ..   price = col_double(),
##   ..   bedrooms = col_double(),
##   ..   bathrooms = col_double(),
##   ..   sqft_living = col_double(),
##   ..   sqft_lot = col_double(),
##   ..   floors = col_double(),
##   ..   waterfront = col_double(),
##   ..   view = col_double(),
##   ..   condition = col_double(),
##   ..   sqft_above = col_double(),
##   ..   sqft_basement = col_double(),
##   ..   yr_built = col_double(),
##   ..   yr_renovated = col_double(),
##   ..   street = col_character(),
##   ..   city = col_character(),
##   ..   statezip = col_character(),
##   ..   country = col_character()
##   .. )
```

As can be seen from the results below, there might be potential error on the recorded data. It is common to see this problem and we need to identified them and remove it manually.

```
# Distribution of house price to see any potential error
boxplot(house$price, main="Original House Price Boxplot")
```



From the boxplot we can see that there are 2 data points that have significantly high house price compared to others. The following code will be used to identified if it is an error when recording the data.

```
# Remove the 2 highest house price dataset since they does not seems to have correct house price
with the criteria (error)
house[which.max(house$price),]
```

```
## # A tibble: 1 x 18
##   date                  price bedrooms bathrooms sqft_living sqft_lot floors
##   <dttm>                <dbl>    <dbl>     <dbl>       <dbl>    <dbl>  <dbl>
## 1 2014-07-03 00:00:00 2.66e7        3         2        1180     7793      1
## # ... with 11 more variables: waterfront <dbl>, view <dbl>, condition <dbl>,
## #   sqft_above <dbl>, sqft_basement <dbl>, yr_built <dbl>, yr_renovated <dbl>,
## #   street <chr>, city <chr>, statezip <chr>, country <chr>
```

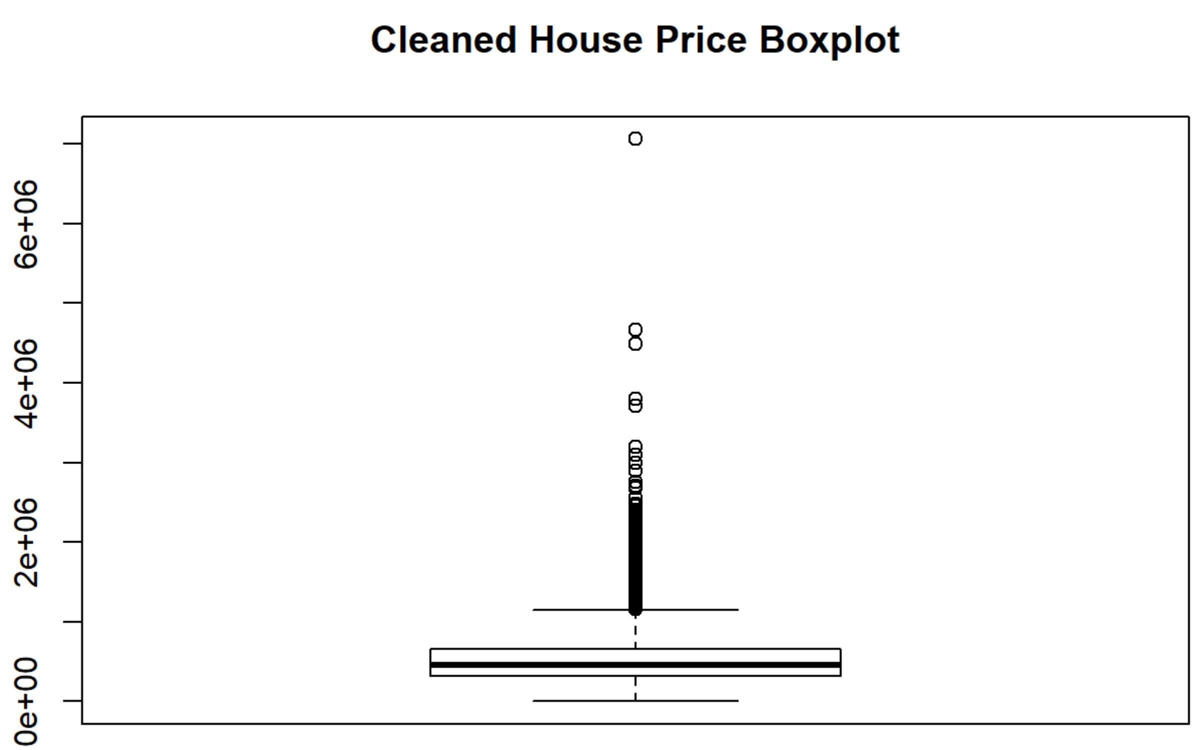
```
house <- house %>% filter(price<max(price))
house[which.max(house$price),]
```

```
## # A tibble: 1 x 18
##   date                  price bedrooms bathrooms sqft_living sqft_lot floors
##   <dttm>                <dbl>    <dbl>     <dbl>       <dbl>    <dbl>  <dbl>
## 1 2014-06-23 00:00:00 1.29e7        3       2.5        2190    11394      1
## # ... with 11 more variables: waterfront <dbl>, view <dbl>, condition <dbl>,
## #   sqft_above <dbl>, sqft_basement <dbl>, yr_built <dbl>, yr_renovated <dbl>,
## #   street <chr>, city <chr>, statezip <chr>, country <chr>
```

```
house <- house %>% filter(price<max(price))
```

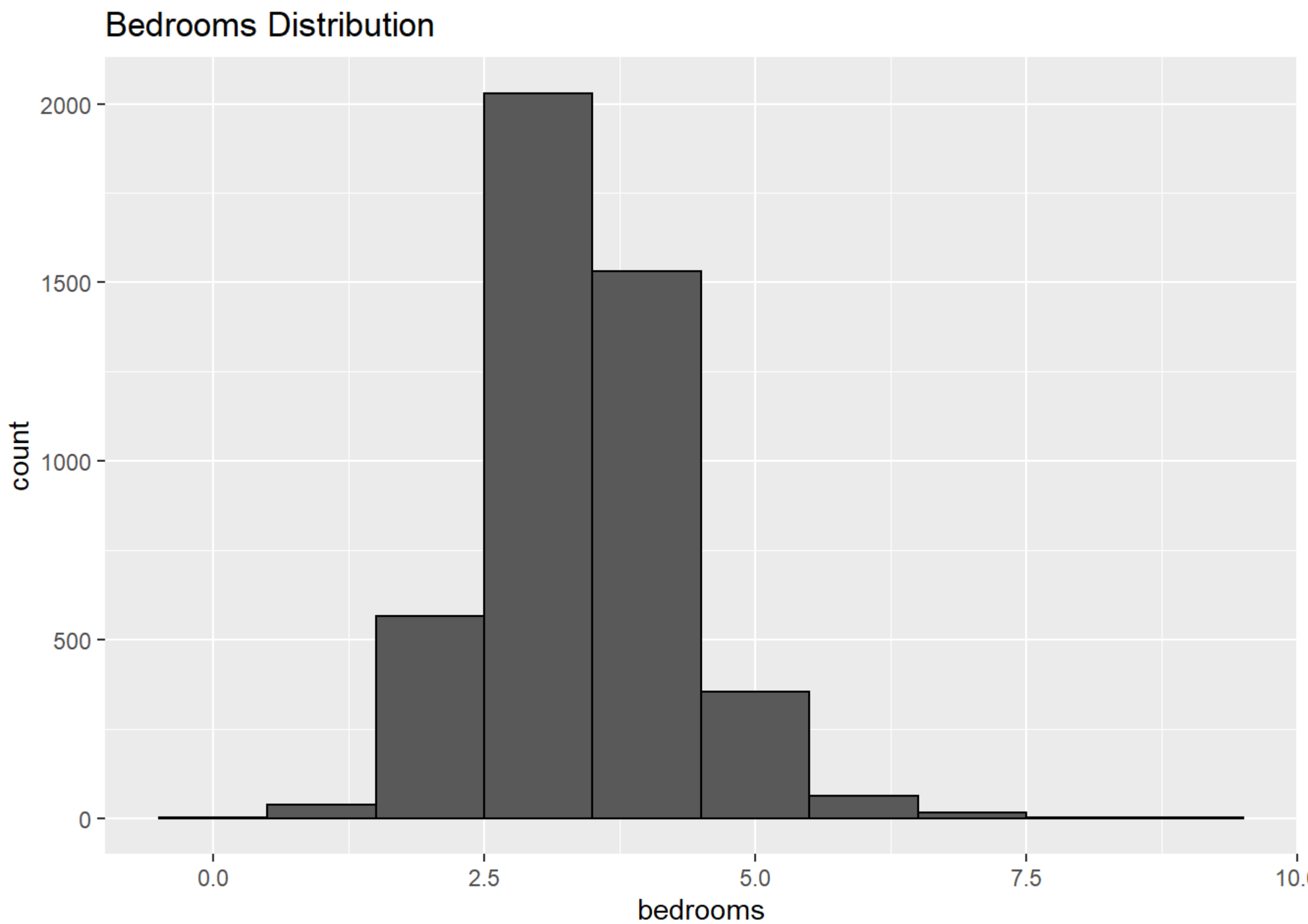
The maximum house price is $26.5 million but the attributes such as sq-ft area, view and built year does not represent the price stated, thus we decided to remove the data. Similarly, repeated exercise was performed on the second highest house price. After the house price being cleaned up, we can plot the distribution of the house price again using a boxplot.

```
# Distribution of house price to see any potential error
boxplot(house$price, main="Cleaned House Price Boxplot")
```



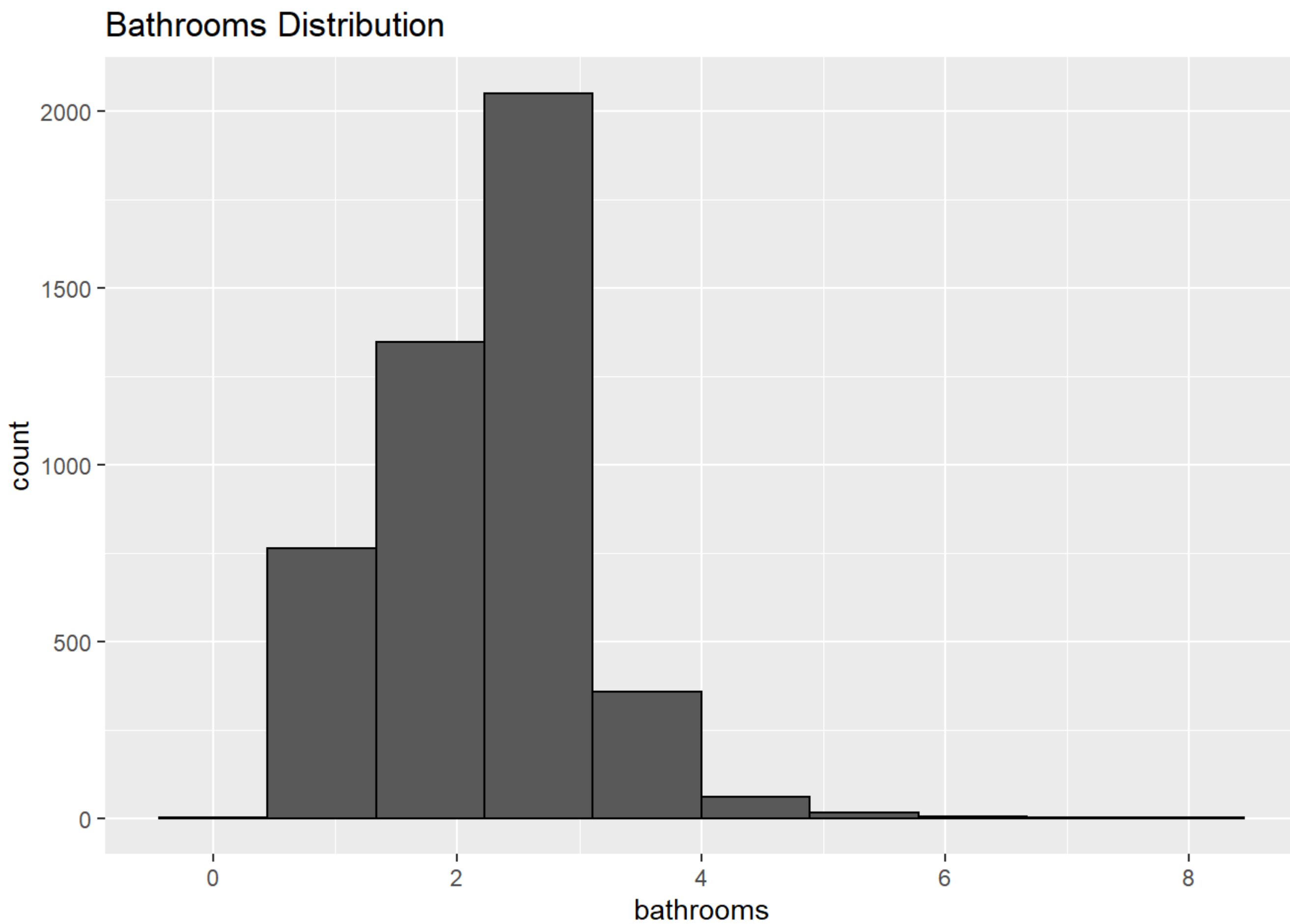
Next, we can check the distribution of several key variables to understand the distribution of the attributes. First, the number of bedrooms distribution shows that it ranges between 1 to 5.

```
house %>% ggplot(aes(bedrooms))  + geom_histogram(bins = 10, color = "black") + ggtitle("Bedroom
s Distribution")
```

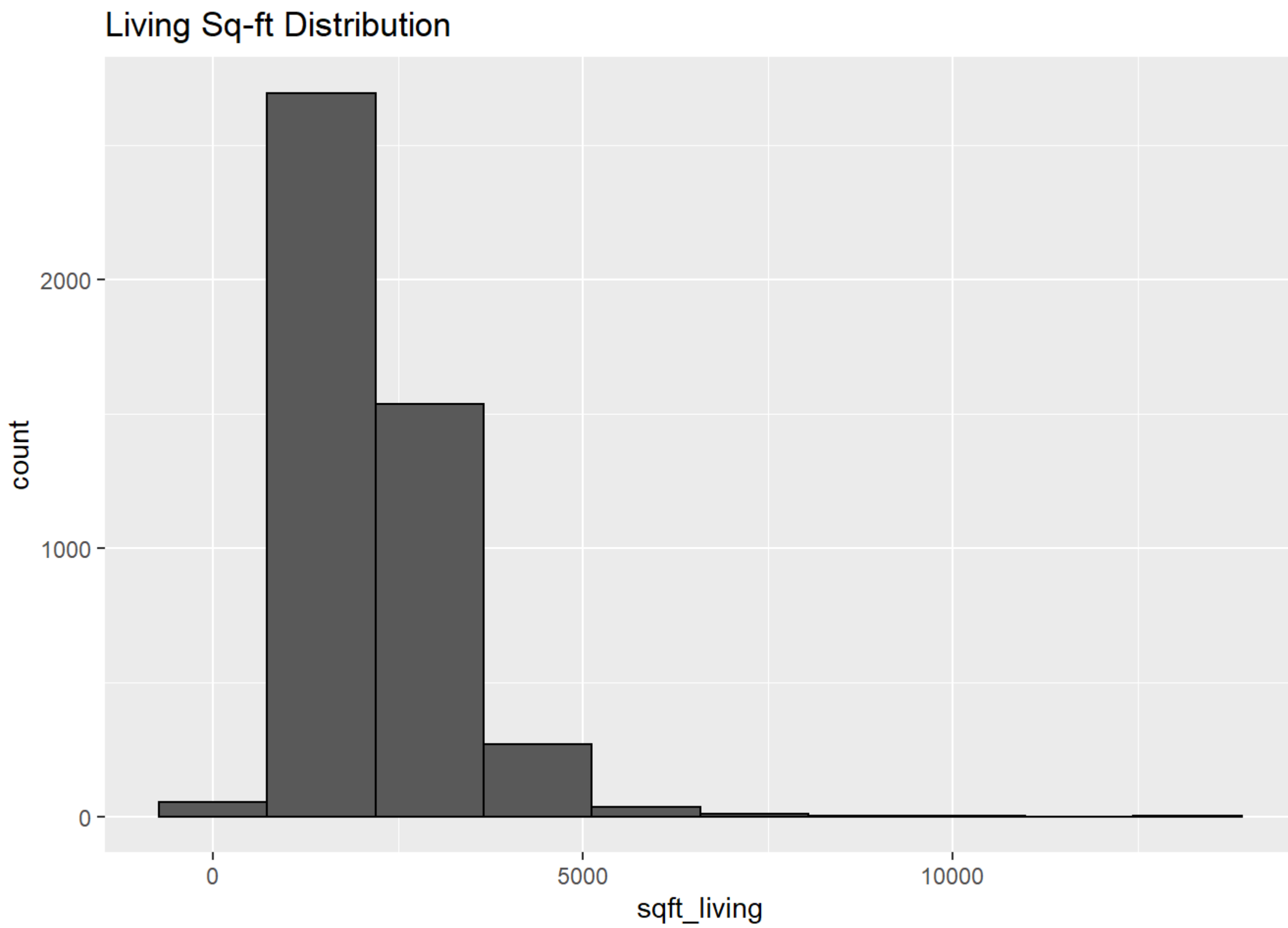


Similarly, we can plot the distribution for other attributes such as number of bathrooms and living sq-ft area.

```
house %>% ggplot(aes(bathrooms)) + geom_histogram(bins = 10, color = "black") + ggtitle("Bathroo
ms Distribution")
```

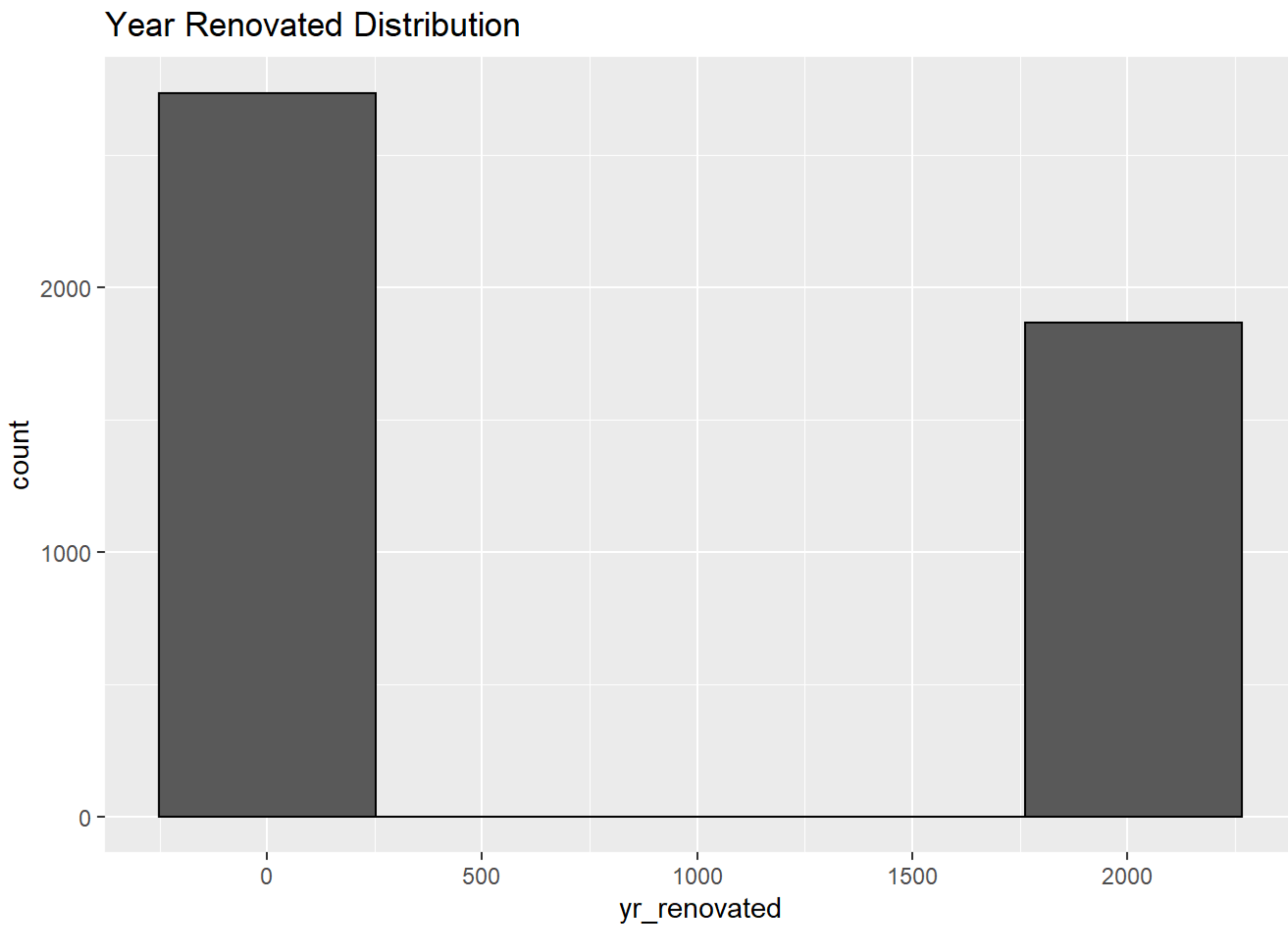



```
house %>% ggplot(aes(sqft_living)) + geom_histogram(bins = 10, color = "black") + ggtitle("Livin
g Sq-ft Distribution")
```



There are some attributes that can be dropped from the dataset. For example, we can drop the year renovated variable because not all houses are renovated and thus it has large number of 0s. This is not helpful for the model to predict house price.

```
house %>% ggplot(aes(yr_renovated)) + geom_histogram(bins = 5, color = "black") + ggtitle("Year Renovated Distribution")
```



Data Cleaning

Since we are interested to predict whether the house price is above the median value, we need to see the summary of the dataset.

```
# House price summary
summary(house$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0  322625  460443  543615  653750 7062500
```

```
# Calculate median house price of the dataset
median_house <- as.numeric(summary(house$price)[3])
```

As shown in the results above, the median house price is around $460,000. Thus, we can create a new variable called ‘price_fac’ that has 1 if house price is greateer than median and 0 otherwise.

```
# Create 'price_fac' variable that has 1 if house price greater than median and 0 otherwise
house$price_fac = as.factor(ifelse(house$price>median_house,1,0))
```

There are also several cleaning steps performed to produce the final usable data for the rest of the analysis. For example, we can drop country variable because all the houses in the dataset are located in US. Thus, there is no variability in the variable. Some other attributes that can be drop are like street address, date recorded and state zipcode.

```
# Remove unnecessary/high corelation/low variability variable to improve processing time
house <- subset(house,select=-c(price,country,yr_renovated,street,date,statezip))
```

Data Partition

The original dataset are splitted to training set and validation set. The proportion of split is 90-10 for training and validation respectively. The training set will be used to build model and the validation sets will be used to evaluate the accuracy of the model at the end of the analysis.

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index  <- createDataPartition(y = house$price_fac, times = 1, p = 0.1, list = FALSE)
train_house <- house[-test_index,]
test_house  <- house[test_index,]
```

Model Development

The model development section in this report are broken to 5 parts. The first part will be building model using generalized linear model (GLM). Next, the same dataset will be tested using k-nearest neighbor algorithm. Third step will be Gam LOESS model and followed by Classification and Regreesion Trees (CART) model. The final piece of the model development is to ensemble top 3 out of 4 best models to produce the highest accuracy in predicting house price.

Model 1: GLM Method

First, we will apply the GLM method on train dataset. The accuracy is computed againts the validation set and the number is 0.8391304.

```
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
# train glm model
train_glm             <- train(price_fac ~ ., method="glm", data=train_house)
pred_glm              <- predict(train_glm, test_house, "raw")
```

```
# calculate accuracy using glm method and store in data frame
accuracy_glm          <- confusionMatrix(pred_glm,factor(test_house$price_fac))$overall[["Accura
cy"]]
accuracy_results      <- data_frame(Method = "GLM Method", Accuracy = accuracy_glm)
accuracy_glm          <- data_frame(Method = "GLM Method", Accuracy = accuracy_glm)
accuracy_glm %>% knitr::kable()
```

Method	Accuracy
GLM Method	0.8391304

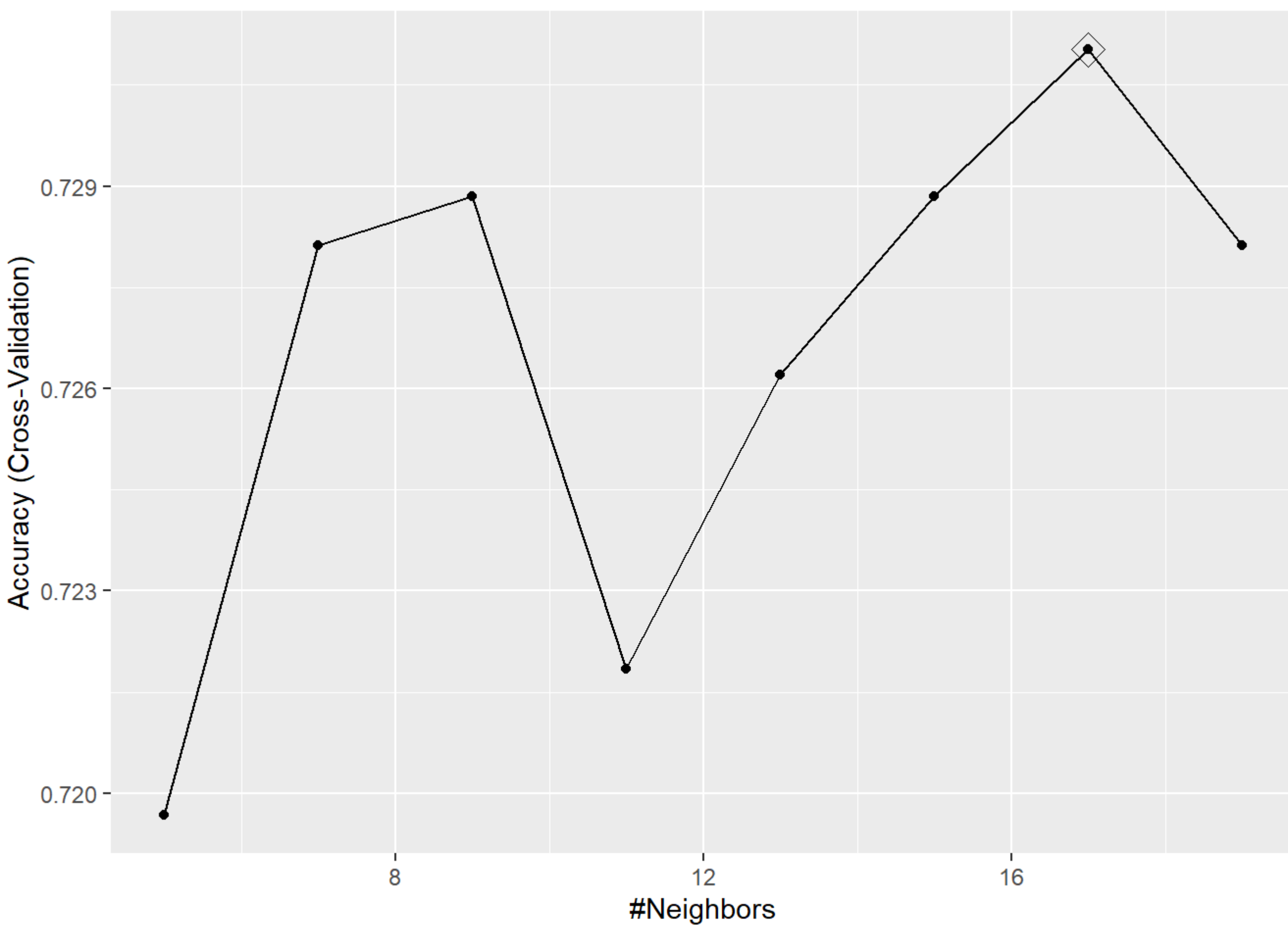
Model 2: KNN Method

The second method that we are presenting here is the KNN method. Instead of using the default tune parameter, we will try using odd number from 5 to 19. The best tuning parameter is 17 as can be identified from the code below. The accuracy for this method is 0.7521739 and it is lower than the GLM method from the first model.

```
modelLookup("knn")
```

```
##   model parameter      label forReg forClass probModel
## 1   knn         k #Neighbors   TRUE     TRUE      TRUE
```

```
# 5 fold cross-validation tuning
control                <- trainControl(method = "cv", number = 5, p = .9)
# train knn model by using tune grid from 5 to 31 (odd number)
train_knn              <- train(price_fac ~ ., method = "knn", data = train_house, tuneGrid = dat
a.frame(k = seq(5,19,2)), trControl = control)
# plot parameter that provides the best accuracy
ggplot(train_knn, highlight = TRUE)
```




```
train_knn$bestTune
```

```
##    k
## 7 17
```

```
pred_knn             <- predict(train_knn, test_house, type="raw")
accuracy_knn         <- confusionMatrix(pred_knn, factor(test_house$price_fac))$overall[["Accur
acy"]]
accuracy_results     <- bind_rows(accuracy_results,data_frame(Method="KNN Method", Accuracy = a
ccuracy_knn))
accuracy_knn         <- data_frame(Method = "KNN Method", Accuracy = accuracy_knn)
accuracy_knn %>% knitr::kable()
```

Method	Accuracy
KNN Method	0.7521739

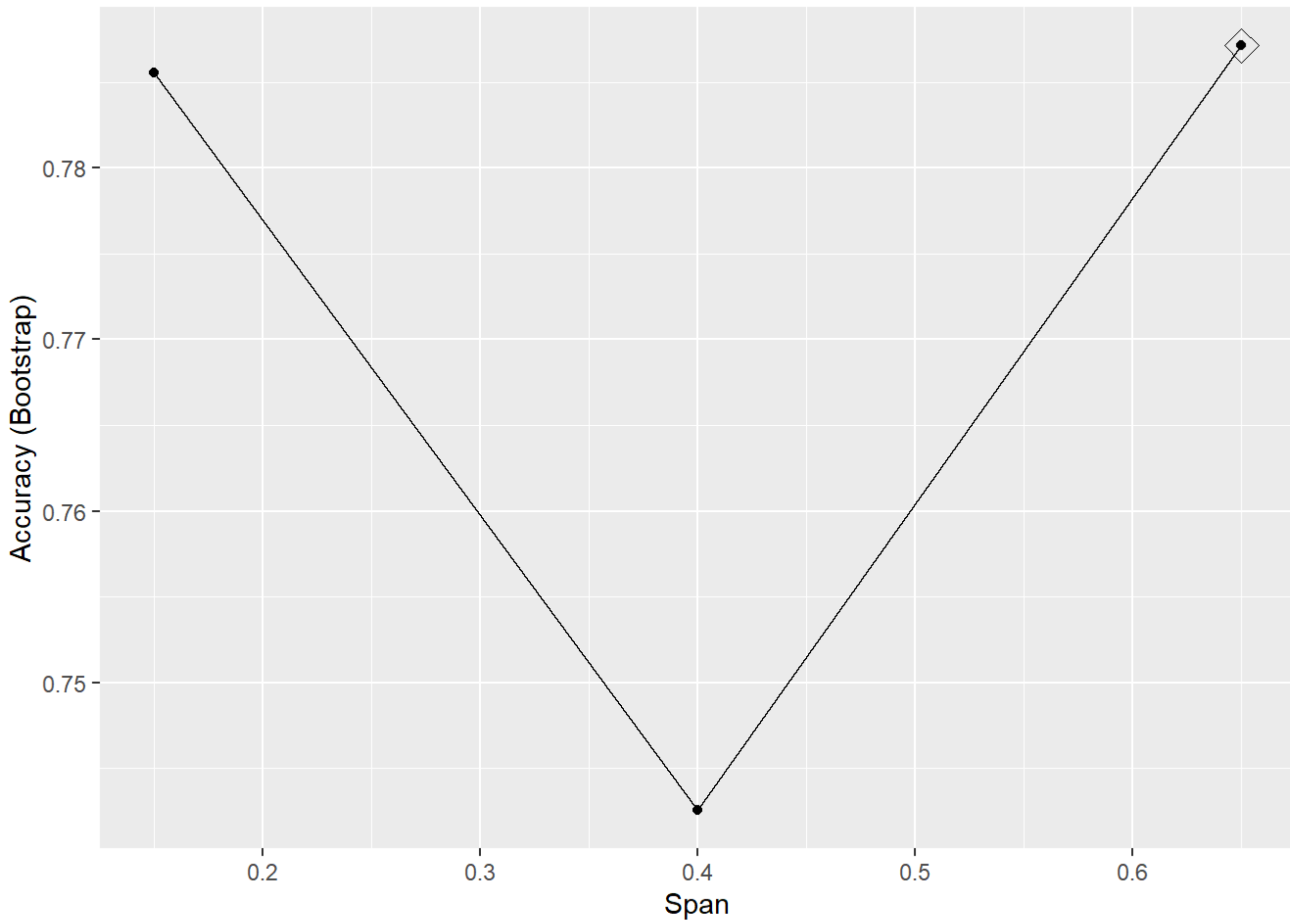
Model 3: GamLoess Method

The third model that will be tested for this dataset is the LOESS method. The grid defined is between 0.15 and 0.65. The setup is limited to 3 grids due to long computational time in running the algorithm. Even by only using 3 values, we managed to achieve a reasonable accuracy of 0.7826087. The accuracy is better than KNN method but still not as good as the GLM method. The following code will walk through the steps in setting up gamLoess function in testing the house price dataset.

```
modelLookup("gamLoess")
```

```
##      model parameter  label forReg forClass probModel
## 1 gamLoess      span   Span   TRUE     TRUE      TRUE
## 2 gamLoess    degree Degree   TRUE     TRUE      TRUE
```

```
# Define grid. Using length of 3 due to limit capacity of my laptop
grid                 <- expand.grid(span = seq(0.15, 0.65, len = 3), degree = 1)
train_loess          <- train(price_fac ~ ., method = "gamLoess", tuneGrid=grid,data = train_ho
use)
# plot parameter that provides best accuracy
ggplot(train_loess, highlight = TRUE)
```




```
pred_gamLoess        <- predict(train_loess, test_house)
```

```
# calculate accuracy using gamLoess method and store in data frame
accuracy_gamLoess     <- confusionMatrix(data = predict(train_loess, test_house), reference = te
st_house$price_fac)$overall["Accuracy"]
accuracy_results      <- bind_rows(accuracy_results,data_frame(Method="GamLoess Method", Accurac
y = accuracy_gamLoess))
accuracy_gamLoess     <- data_frame(Method = "GamLoess Method", Accuracy = accuracy_gamLoess)
accuracy_gamLoess %>% knitr::kable()
```

Method	Accuracy
GamLoess Method	0.7826087

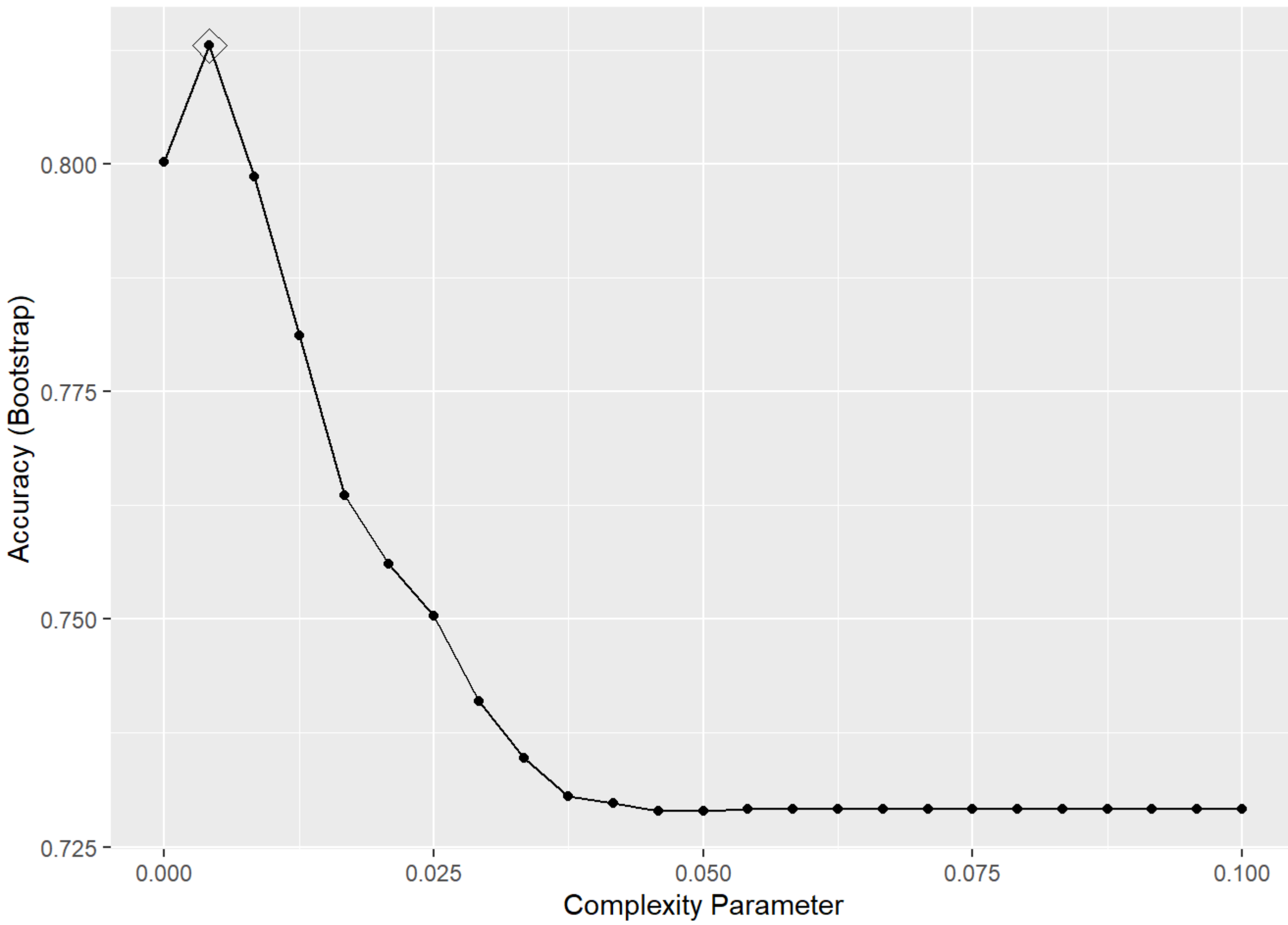
Model 4: Classification and Regression Trees (CART) Method

The final model that will be tested before ensemble process is the CART method. The tuning parameters used for this analysis is by using 25 equally spaced number between 0 and 0.1. As can be seen from the graph below, the maximum accuracy can be achieved when complexity parameter equal to 0.004166667. This method yield second best accuracy among all the models that have been investigated, 0.823913.

```
modelLookup("rpart")
```

```
##   model parameter                label forReg forClass probModel
## 1 rpart        cp Complexity Parameter   TRUE     TRUE      TRUE
```

```
# train rpart model with selected parameters
train_rpart           <- train(price_fac ~ ., data = train_house, method = "rpart",tuneGrid = da
ta.frame(cp = seq(0.0, 0.1, len = 25)))
# plot parameter that provides the best accuracy
ggplot(train_rpart, highlight = TRUE)
```




```
train_rpart$bestTune
```

```
##            cp
## 2 0.004166667
```

```
pred_rpart            <- predict(train_rpart, test_house, type="raw")
accuracy_rpart        <- confusionMatrix(pred_rpart, factor(test_house$price_fac))$overall[["Acc
uracy"]]
accuracy_results      <- bind_rows(accuracy_results,data_frame(Method="CART Method", Accuracy =
 accuracy_rpart))
accuracy_rpart        <- data_frame(Method = "CART Method", Accuracy = accuracy_rpart)
accuracy_rpart %>% knitr::kable()
```

Method	Accuracy
CART Method	0.823913

Model 5: Ensemble

The final steps of the model development is to combine several best model results in a process called ensemble. The 3 best models that produced highest accuracy will be used in this process. If 2 out of the 3 best models predicted the house price to be higher than median, it will be assigned as 1. The accuracy produced using this method is 0.8478261.

```
# Pick top 3 method that provides highest accuracy and ensemble them together
combine               <- as.numeric(as.character(pred_rpart)) + as.numeric(as.character(pred_gam
Loess)) +  as.numeric(as.character(pred_glm))
# If there are 2 out of the 3 top models predict 1, it will assign 1 to it
pred_ensemble         <- factor(ifelse(combine>=2,1,0))
accuracy_ensemble     <- confusionMatrix(pred_ensemble, factor(test_house$price_fac))$overall[[
"Accuracy"]]
accuracy_results      <- bind_rows(accuracy_results,data_frame(Method="Ensemble Method", Accurac
y = accuracy_ensemble))
accuracy_ensemble     <- data_frame(Method = "Ensemble Method", Accuracy = accuracy_ensemble)
accuracy_ensemble %>% knitr::kable()
```

Method	Accuracy
Ensemble Method	0.8478261

Result

The summary below shows that ensemble method not only produces high accuracy, but also high sensitivity and high specificity. This is important because we want a model that is good in prediciting proportion of actual positive and negative outcomes correctly.

```
summary_ensemble     <- confusionMatrix(pred_ensemble, factor(test_house$price_fac))
summary_ensemble
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 195  35
##          1  35 195
##                                           
##                Accuracy : 0.8478          
##                  95% CI : (0.8117, 0.8794)
##     No Information Rate : 0.5             
##     P-Value [Acc > NIR] : <2e-16          
##                                           
##                   Kappa : 0.6957          
##                                           
##  Mcnemar's Test P-Value : 1               
##                                           
##             Sensitivity : 0.8478          
##             Specificity : 0.8478          
##          Pos Pred Value : 0.8478          
##          Neg Pred Value : 0.8478          
##              Prevalence : 0.5000          
##          Detection Rate : 0.4239          
##    Detection Prevalence : 0.5000          
##       Balanced Accuracy : 0.8478          
##                                           
##        'Positive' Class : 0               
## 
```

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
GLM Method	0.8391304
KNN Method	0.7521739
GamLoess Method	0.7826087
CART Method	0.8239130
Ensemble Method	0.8478261

The summary below shows that ensemble method in the final steps of model development shows the highest accuracy.

```
accuracy_results %>% knitr::kable()
```

Method	Accuracy
GLM Method	0.8391304
KNN Method	0.7521739

Method	Accuracy
GamLoess Method	0.7826087
CART Method	0.8239130
Ensemble Method	0.8478261

Conclusion

Overall, all 5 models produce remarkably high accuracy between 75% to 85%. Although some method yield higher accuracy, we cannot conclude that any method is better than the other. It is depending on the tuning parameter that we set in the algorithm and also the nature of the dataset. The tuning parameter that we used in computing the results might also significantly affect the computational time. Thus, there is a tradeoff between getting a good accuracy and the complexity of the model. Ensembling several best method in pretting house price also can drastically improve the accuracy of the model. There are multiple ways in ensembling the method (i.e. choosing only top 2 models in predicting the house price) and each method might yield different outcomes. Finally, we should also make sure the model is not bias in predicting positive or negative outcome by making sure it has high sensitivity and specificity.