HarvardX Data Science Capstone: New York City property price prediction

Zhi Han

12/25/2020

1. Introduction

Predictive modeling is the general concept of building a model that is capable of making predictions. Typically, such a model includes a machine learning algorithm that learns certain properties from a training dataset in order to make those predictions.

In this project we will try to build a predictive model to predict house price. We will use NYC Property Sales dataset from Kaggle (https://www.kaggle.com/new-york-city/nyc-property-sales). This dataset is a record of every building or building unit (apartment, etc.) sold in the New York City property market over a 12-month period.

This dataset contains the location, address, type, sale price, and sale date of building units sold. A reference on the trickier fields:

- BOROUGH: A digit code for the borough the property is located in; in order these are Manhattan (1), Bronx (2), Brooklyn (3), Queens (4), and Staten Island (5).
- TAX CLASS AT PRESENT and TAX CLASS AT TIME OF SALE: Every property in the city is assigned to one of four tax classes (Classes 1, 2, 3, and 4), based on the use of the property.
  - Class 1: Includes most residential property of up to three units (such as one-,two-, and three-family homes and small stores or offices with one or two attached apartments), vacant land that is zoned for residential use, and most condominiums that are not more than three stories.
  - Class 2: Includes all other property that is primarily residential, such as cooperatives and condominiums.
  - Class 3: Includes property with equipment owned by a gas, telephone or electric company.
  - Class 4: Includes all other properties not included in class 1,2, and 3, such as offices, factories, warehouses, garage buildings, etc.
- RESIDENTIAL UNITS:The number of residential units at the listed property.
- COMMERCIAL UNITS: The number of commercial units at the listed property.
- TOTAL UNITS:The total number of units at the listed property.
- LAND SQUARE FEET:The land area of the property listed in square feet.
- GROSS SQUARE FEET:The total area of all the floors of a building as measured from the exterior surfaces of the outside walls of the building, including the land area and space within any building or structure on the property.
- YEAR BUILT:Year the structure on the property was built.

- SALE PRICE:Price paid for the property.
- SALE DATE: Date the property sold.
- $0 Sales Price: A $0 sale or a small value sale such as $10 or $20 indicates that there was a transfer of ownership without a cash consideration. There can be a number of reasons for a $0 sale including transfers of ownership from parents to children

Our objective of this project is to predict the price of the house with the information provided in the dataset. We will build several predictive models with classic machine learning methods and evaluate and compare their performance.

2. Data Analysis and Predictive Methods

2.1 Libraries and data loading

First, we need to load the packages used for this project. The missing packages will be installed automatically.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
library(tidyverse)
library(caret)
library(randomForest)
library(glmnet)
library(data.table)
```

Then we will load the NYC Property Sales data. The data includes only one file "nyc-rolling-sales.csv" and can be downloaded from https://www.kaggle.com/new-york-city/nyc-property-sales or https://github.com/zhan-us/NYC_Sales/raw/main/nyc-rolling-sales.csv .

```
# download the csv data from internet and load the data from csv file
temp<-tempfile()
download.file("https://github.com/zhan-us/NYC_Sales/raw/main/nyc-rolling-sales.csv",temp)
nyc_sales_orginal<-read_csv(temp)
unlink(temp)
```

2.2 Data exploration and data cleaning

First, let us overview the structure of the data.

```
# review the structure of the dataset
summary(nyc_sales_orginal)
```

```
##        X1            BOROUGH      NEIGHBORHOOD       BUILDING CLASS CATEGORY
##  Min.   :    4   Min.   :1.000   Length:84548       Length:84548
##  1st Qu.: 4231   1st Qu.:2.000   Class :character   Class :character
##  Median : 8942   Median :3.000   Mode  :character   Mode  :character
```

```
##  Mean   :10344   Mean   :2.999
##  3rd Qu.:15987   3rd Qu.:4.000
##  Max.   :26739   Max.   :5.000
##  TAX CLASS AT PRESENT     BLOCK            LOT         EASE-MENT
##  Length:84548         Min.   :    1   Min.   :   1.0   Mode:logical
##  Class :character     1st Qu.: 1323   1st Qu.:  22.0   NA's:84548
##  Mode  :character     Median : 3311   Median :  50.0
##                       Mean   : 4237   Mean   : 376.2
##                       3rd Qu.: 6281   3rd Qu.:1001.0
##                       Max.   :16322   Max.   :9106.0
##  BUILDING CLASS AT PRESENT   ADDRESS           APARTMENT NUMBER
##  Length:84548              Length:84548       Length:84548
##  Class :character          Class :character   Class :character
##  Mode  :character          Mode  :character   Mode  :character
##
##
##
##     ZIP CODE     RESIDENTIAL UNITS  COMMERCIAL UNITS     TOTAL UNITS
##  Min.   :    0   Min.   :   0.000   Min.   :   0.0000   Min.   :   0.000
##  1st Qu.:10305   1st Qu.:   0.000   1st Qu.:   0.0000   1st Qu.:   1.000
##  Median :11209   Median :   1.000   Median :   0.0000   Median :   1.000
##  Mean   :10732   Mean   :   2.025   Mean   :   0.1936   Mean   :   2.249
##  3rd Qu.:11357   3rd Qu.:   2.000   3rd Qu.:   0.0000   3rd Qu.:   2.000
##  Max.   :11694   Max.   :1844.000   Max.   :2261.0000   Max.   :2261.000
##  LAND SQUARE FEET   GROSS SQUARE FEET    YEAR BUILT   TAX CLASS AT TIME OF SALE
##  Length:84548       Length:84548       Min.   :   0   Min.   :1.000
##  Class :character   Class :character   1st Qu.:1920   1st Qu.:1.000
##  Mode  :character   Mode  :character   Median :1940   Median :2.000
##                                        Mean   :1789   Mean   :1.657
##                                        3rd Qu.:1965   3rd Qu.:2.000
##                                        Max.   :2017   Max.   :4.000
##  BUILDING CLASS AT TIME OF SALE  SALE PRICE
##  Length:84548                   Length:84548
##  Class :character               Class :character
##  Mode  :character               Mode  :character
##
##
##
##    SALE DATE
##  Min.   :2016-09-01 00:00:00
##  1st Qu.:2016-11-29 00:00:00
##  Median :2017-02-28 00:00:00
##  Mean   :2017-02-26 10:03:23
##  3rd Qu.:2017-05-26 00:00:00
##  Max.   :2017-08-31 00:00:00
```

Then we checked and counted the missing value for each varibles

```
nyc_sales<-nyc_sales_orginal

#check the missing value for each varibles
colSums(is.na(nyc_sales))
```

```
##                            X1                          BOROUGH
```

```
##                            0                         0
##                 NEIGHBORHOOD   BUILDING CLASS CATEGORY
##                            0                         0
##         TAX CLASS AT PRESENT                     BLOCK
##                          738                         0
##                          LOT                 EASE-MENT
##                            0                     84548
##     BUILDING CLASS AT PRESENT                   ADDRESS
##                          738                         0
##             APARTMENT NUMBER                  ZIP CODE
##                        65496                         0
##            RESIDENTIAL UNITS          COMMERCIAL UNITS
##                            0                         0
##                  TOTAL UNITS          LAND SQUARE FEET
##                            0                         0
##            GROSS SQUARE FEET                YEAR BUILT
##                            0                         0
##     TAX CLASS AT TIME OF SALE BUILDING CLASS AT TIME OF SALE
##                            0                         0
##                   SALE PRICE                 SALE DATE
##                            0                         0
```

According to the structure and the missing information of the data, first we dropped some varables which has lot of missing value and also are unneccessary for our analysis.

```
#drop the unneccesary varibles with lot of missing value
#drop EASE-MENT since all the value are missing
nyc_sales$`EASE-MENT`<-NULL

#drop "APARTMENT NUMBER" since it is unnecessary variable and most value are missing
nyc_sales$`APARTMENT NUMBER`<-NULL

#drop unneccesary numeric varibles BLOCK,LOT,ZIP CODE
nyc_sales$BLOCK<-NULL
nyc_sales$LOT<-NULL
nyc_sales$`ZIP CODE`<-NULL
```

Then we used the name convention to rename some variables which name has space.

```
#use the name convention to rename the columns wich name has space
nyc_sales<-nyc_sales%>%rename(
  id = X1,
  BUILDING_CLASS_CATEGORY = `BUILDING CLASS CATEGORY`,
  TAX_CLASS_AT_PRESENT = `TAX CLASS AT PRESENT`,
  BUILDING_CALSS_AT_PRESENT = `BUILDING CLASS AT PRESENT`,
  RESIDENTIAL_UNITS = `RESIDENTIAL UNITS`,
  COMMERCIAL_UNITS = `COMMERCIAL UNITS`,
  TOTAL_UNITS = `TOTAL UNITS`,
  LAND_SQUARE_FEET = `LAND SQUARE FEET`,
  GROSS_SQUARE_FEET = `GROSS SQUARE FEET`,
  YEAR_BUILT = `YEAR BUILT`,
  BUILDING_CLASS_AT_TIME_OF_SALE = `BUILDING CLASS AT TIME OF SALE`,
  SALE_PRICE = `SALE PRICE`,
```

```
  SALE_DATE = ‘SALE DATE‘,
  TAX_CLASS_AT_TIME_OF_SALE= ‘TAX CLASS AT TIME OF SALE‘

)
```

According to the structure of the data, we found the class of LAND_SQUARE_FEET, GROSS_SQUARE_FEET, SALE_PRICE are character. We need to convert them to numeric.

```
#change the variable LAND_SQUARE_FEET, GROSS_SQUARE_FEET, SALE_PRICE to numeric class
nyc_sales$LAND_SQUARE_FEET<-as.numeric(nyc_sales$LAND_SQUARE_FEET)
nyc_sales$GROSS_SQUARE_FEET<-as.numeric(nyc_sales$GROSS_SQUARE_FEET)
nyc_sales$SALE_PRICE<-as.numeric(nyc_sales$SALE_PRICE)
```

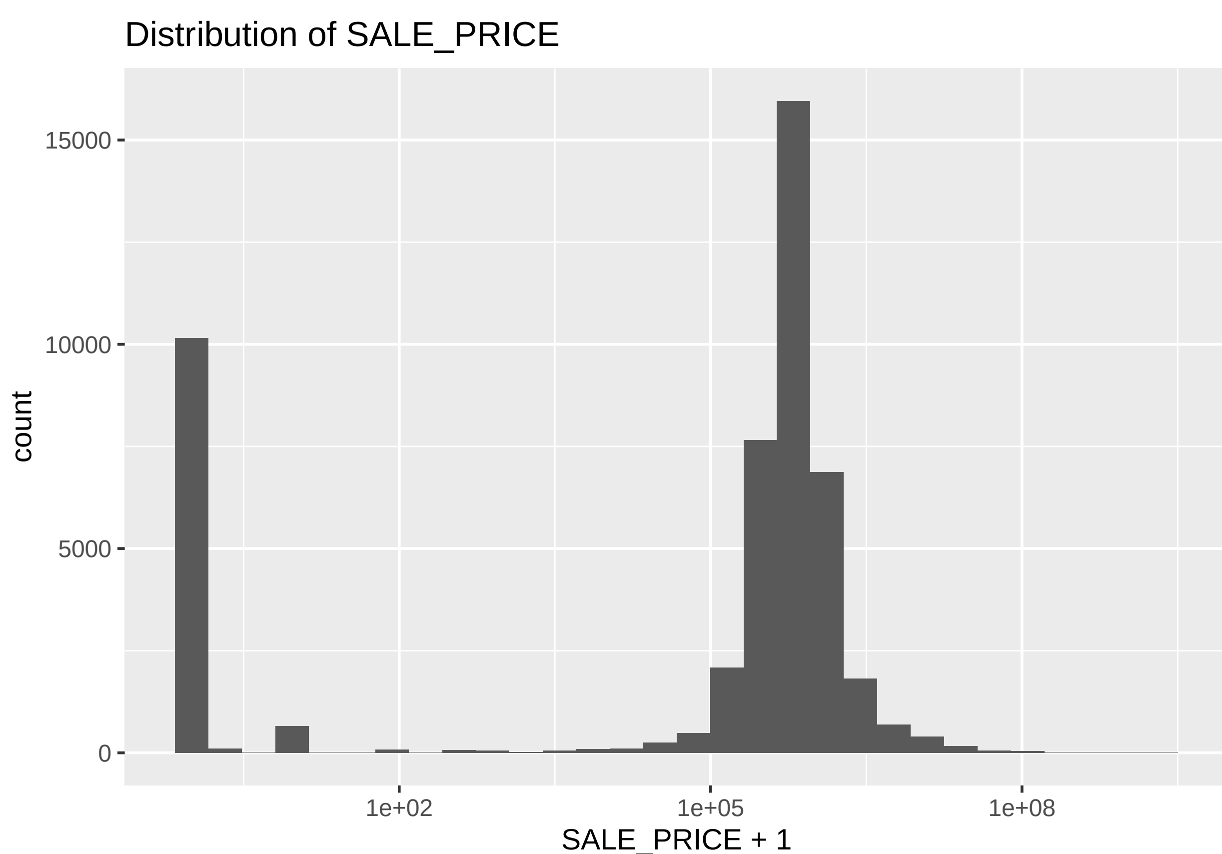
Then we drop all the rows containing the missing value

```
#drop all the rows containing missing value
nyc_sales<-drop_na(nyc_sales)
```

Now, we checked the distribution of the variable SALE_PRICE:

```
# check the distribution of sale price
nyc_sales%>%ggplot(aes(SALE_PRICE+1))+
  geom_histogram()+scale_x_continuous(trans='log10')+
  ggtitle("Distribution of SALE_PRICE")
```

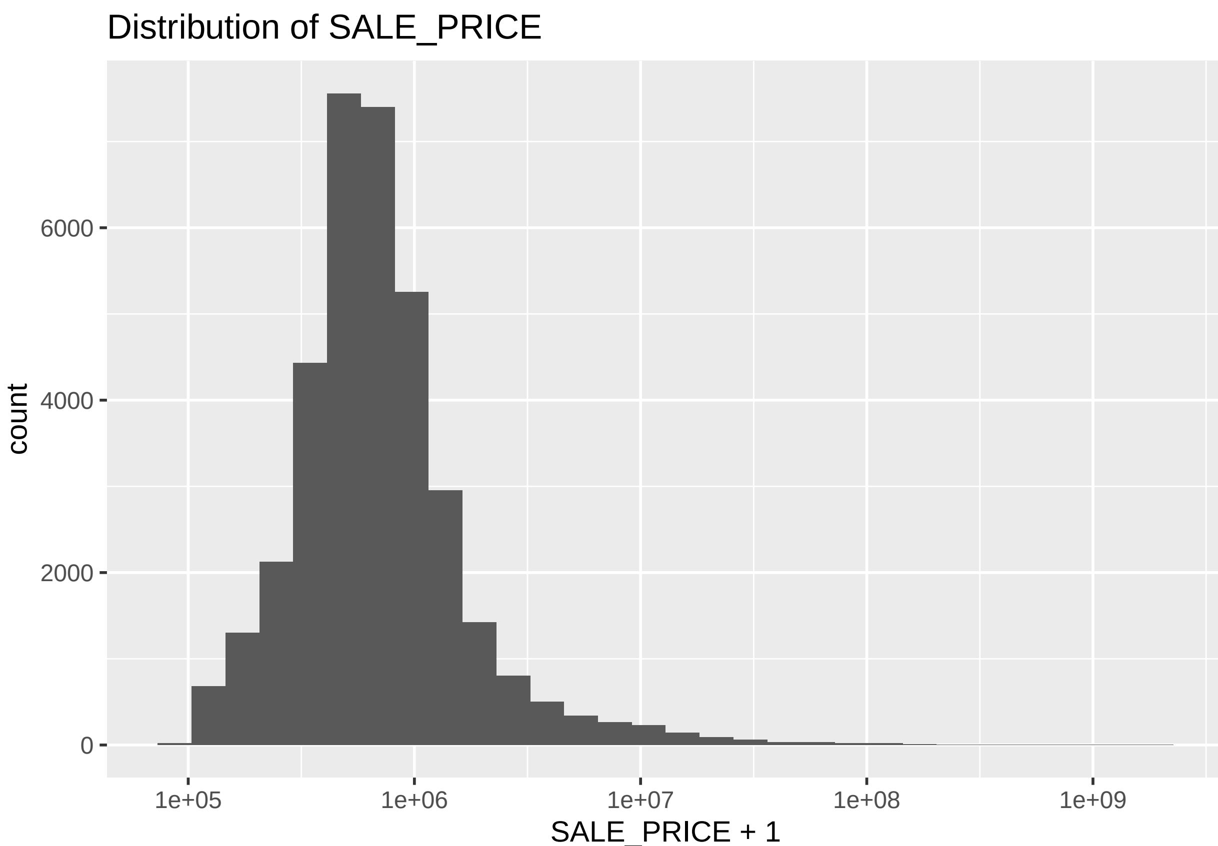
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



We found a lot of houses were sold by unreal prices, this means that the houses where not sold, they were transfer between owners. Here we will only use the houses with prices over $100,000 to build our model, since it is a relistic price that won't mess the models.

```
# filter the houses with price lower than $100000
nyc_sales<- nyc_sales%>%filter(SALE_PRICE>100000)
nyc_sales%>%ggplot(aes(SALE_PRICE+1))+
  geom_histogram()+scale_x_continuous(trans='log10')+
  ggtitle("Distribution of SALE_PRICE")
```

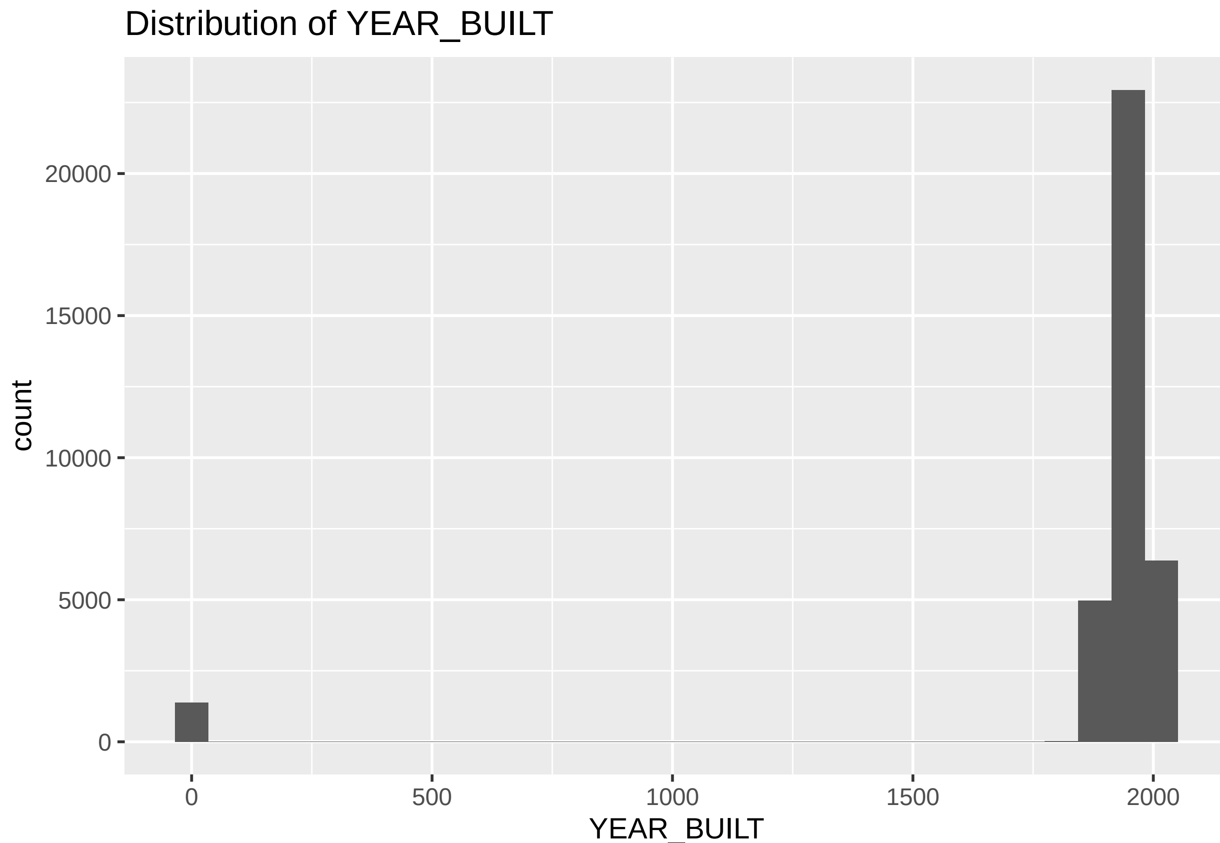
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Similiarly, we checked the distribution of YEAR_BUILT

```
# check the distribution of year built
nyc_sales%>%ggplot(aes(YEAR_BUILT))+
  geom_histogram()+
  ggtitle("Distribution of YEAR_BUILT")
```

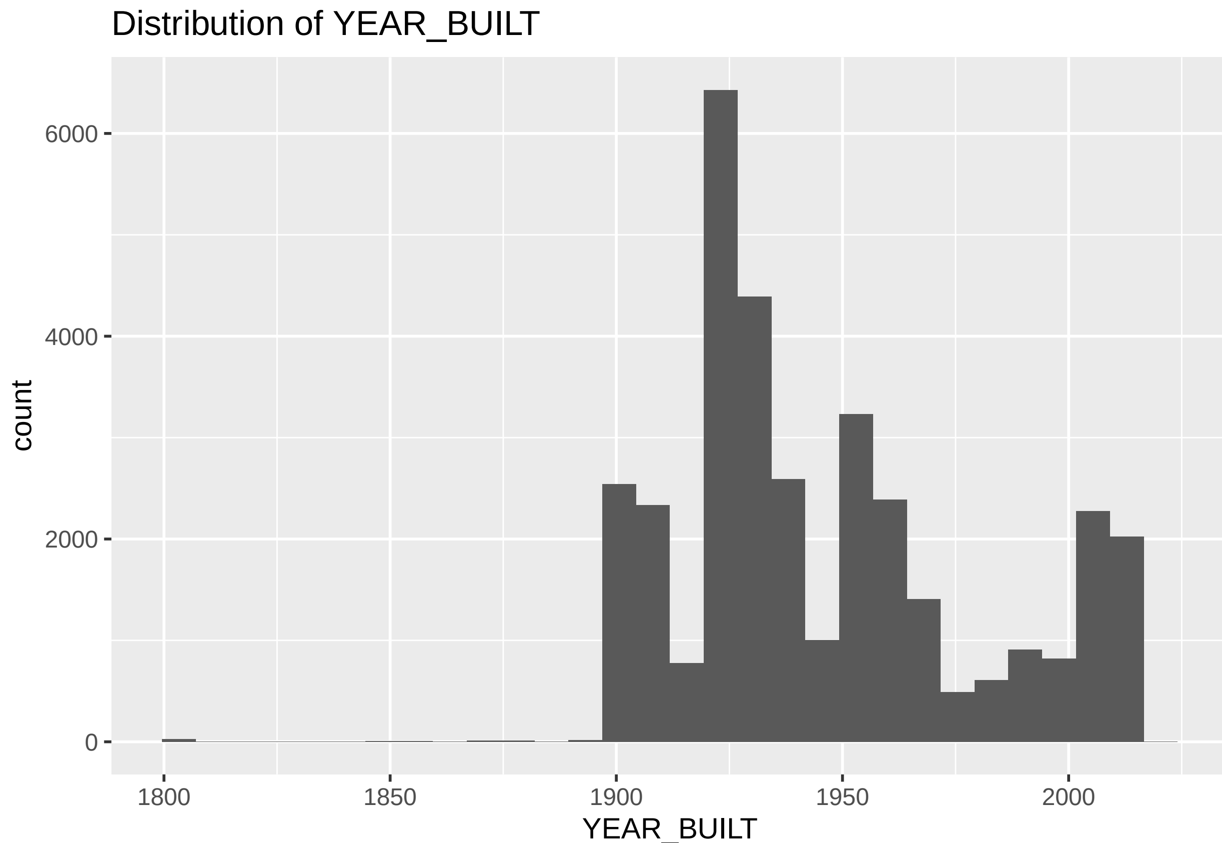
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We found the YEAR_BUILT of some houses are 0 and then we need to delete these rows.

```
# drop the house with the YEAR_BUILT 0.
nyc_sales<- nyc_sales%>%filter(YEAR_BUILT>0)
# check the distribution of year built
nyc_sales%>%ggplot(aes(YEAR_BUILT))+
  geom_histogram()+
  ggtitle("Distribution of YEAR_BUILT")
```

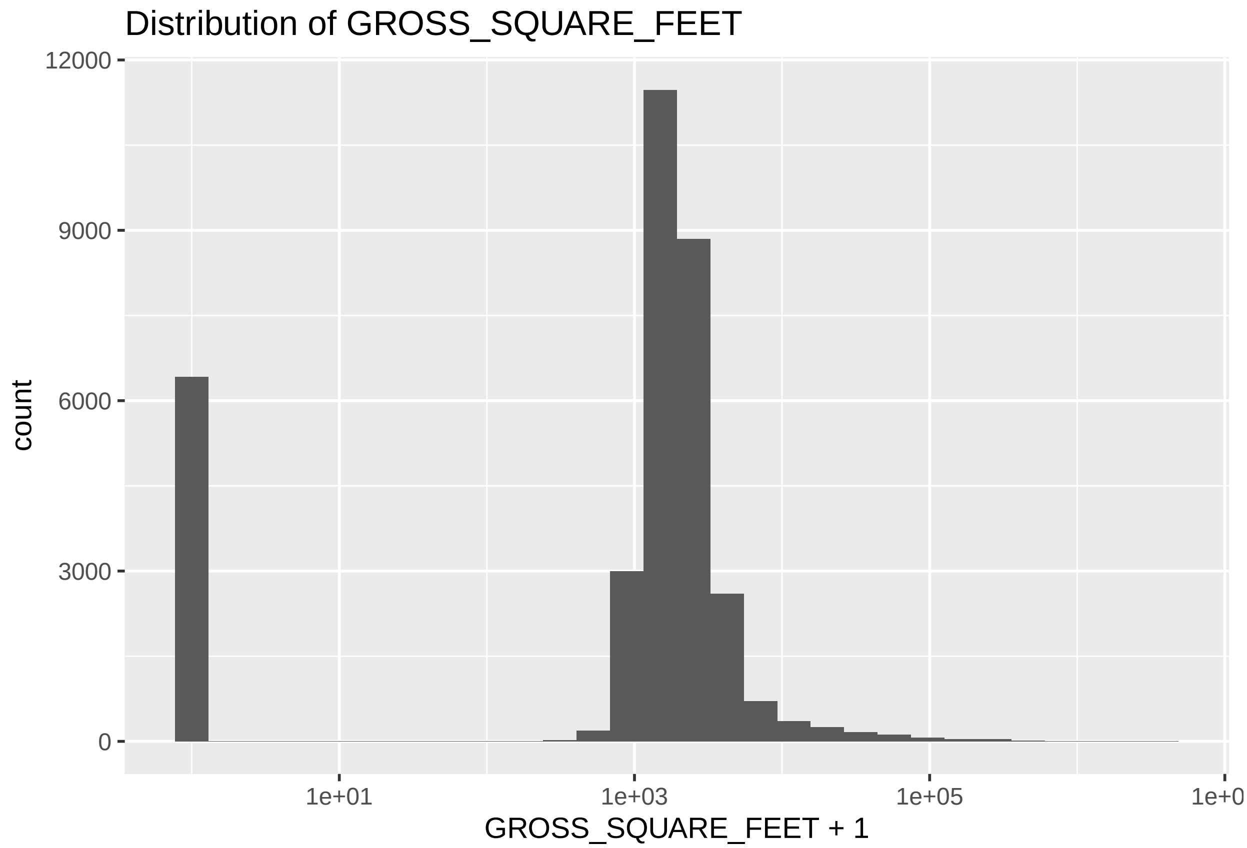
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Following is the distribution of GROSS_SQUARE_FEET

```
#check the distribution of GROSS_SQUARE_FEET
nyc_sales%>%ggplot(aes(GROSS_SQUARE_FEET+1))+
  geom_histogram()+scale_x_continuous(trans = 'log10')+
  ggtitle("Distribution of GROSS_SQUARE_FEET")
```

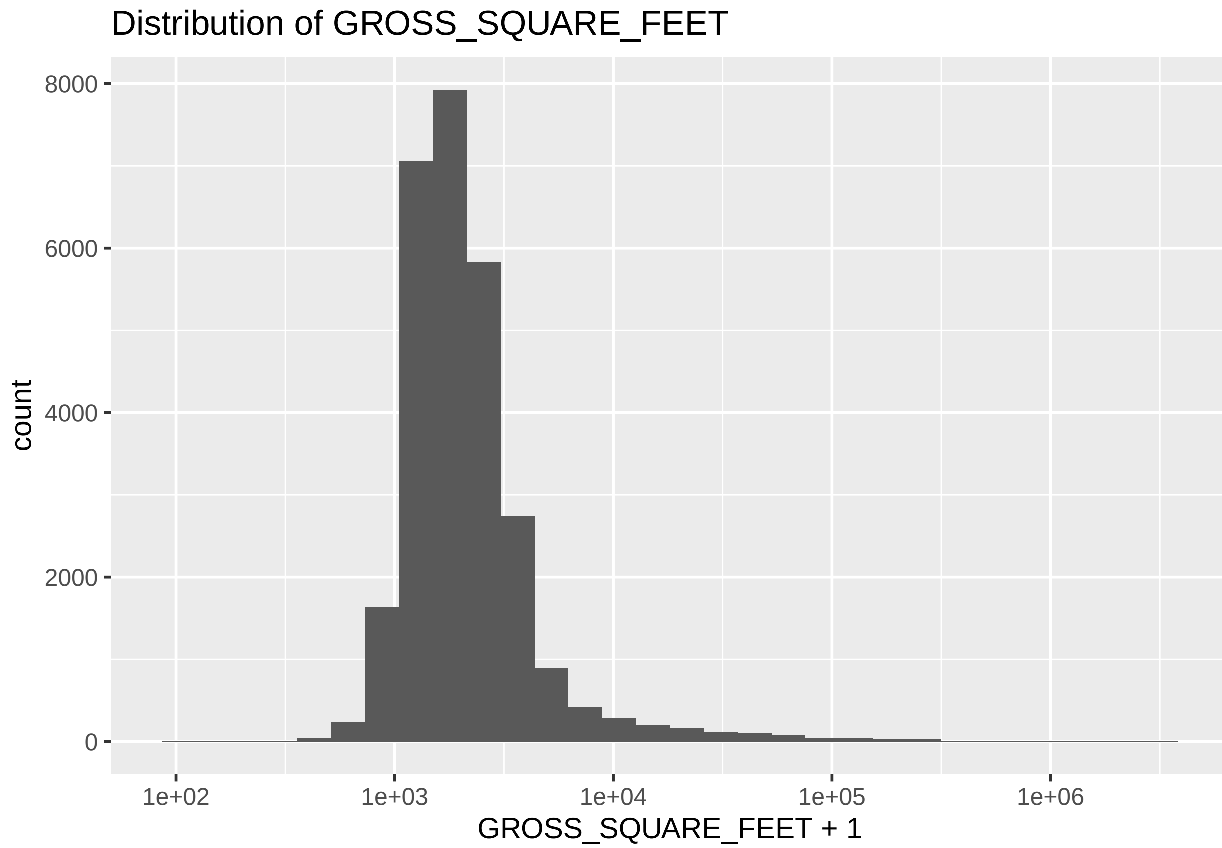
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We found that lot of houses has 0 GROSS_SQUARE_FEET and we need to delete these houses for building our predictive models.

```
#drop the houses with 0 GROSS_SQUARE_FEET
nyc_sales<- nyc_sales%>%filter(GROSS_SQUARE_FEET>0)
nyc_sales%>%ggplot(aes(GROSS_SQUARE_FEET+1))+
  geom_histogram()+scale_x_continuous(trans = 'log10')+
  ggtitle("Distribution of GROSS_SQUARE_FEET")
```

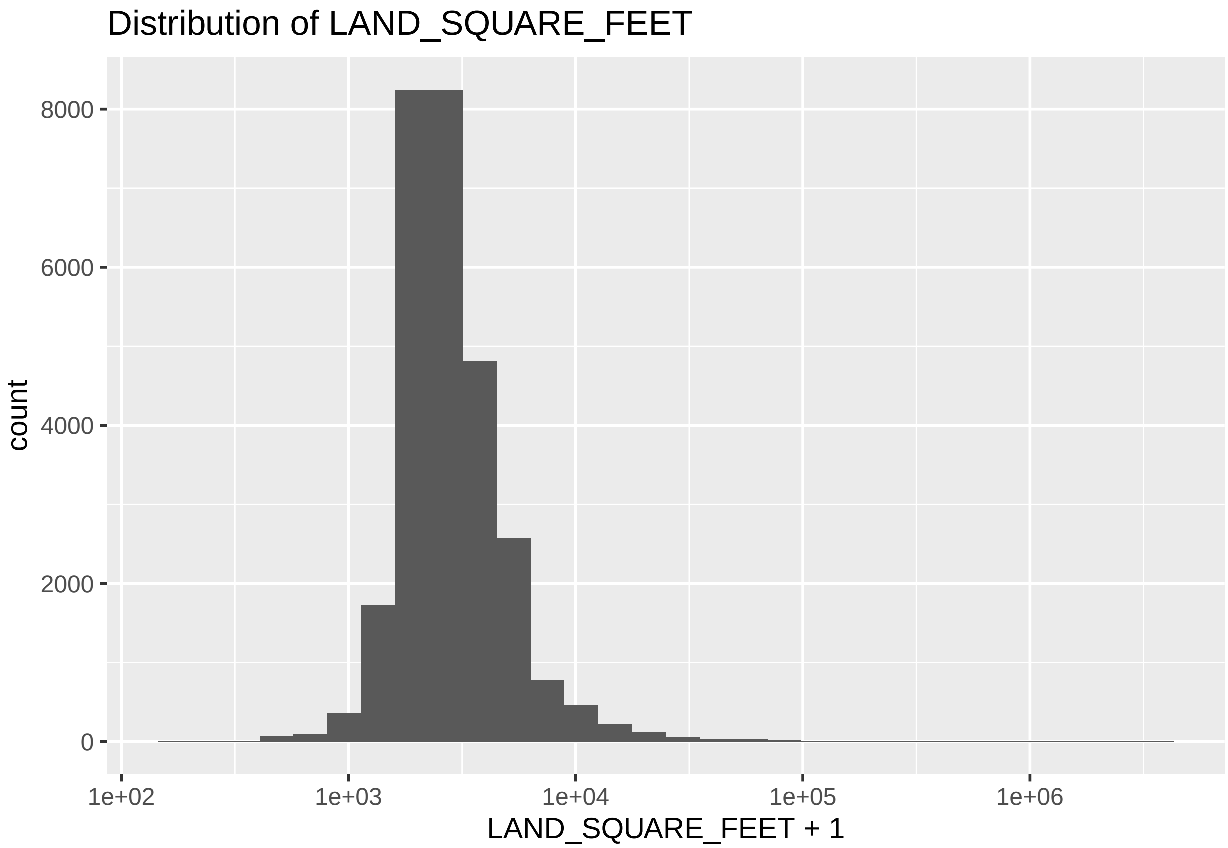
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Similarly, we plot the distribution of LAND_SQUARE_FEET

```
# check the distribution of LAND_SQUARE_FEET
nyc_sales%>%ggplot(aes(LAND_SQUARE_FEET+1))+
  geom_histogram()+scale_x_continuous(trans = 'log10')+
  ggtitle("Distribution of LAND_SQUARE_FEET")
```

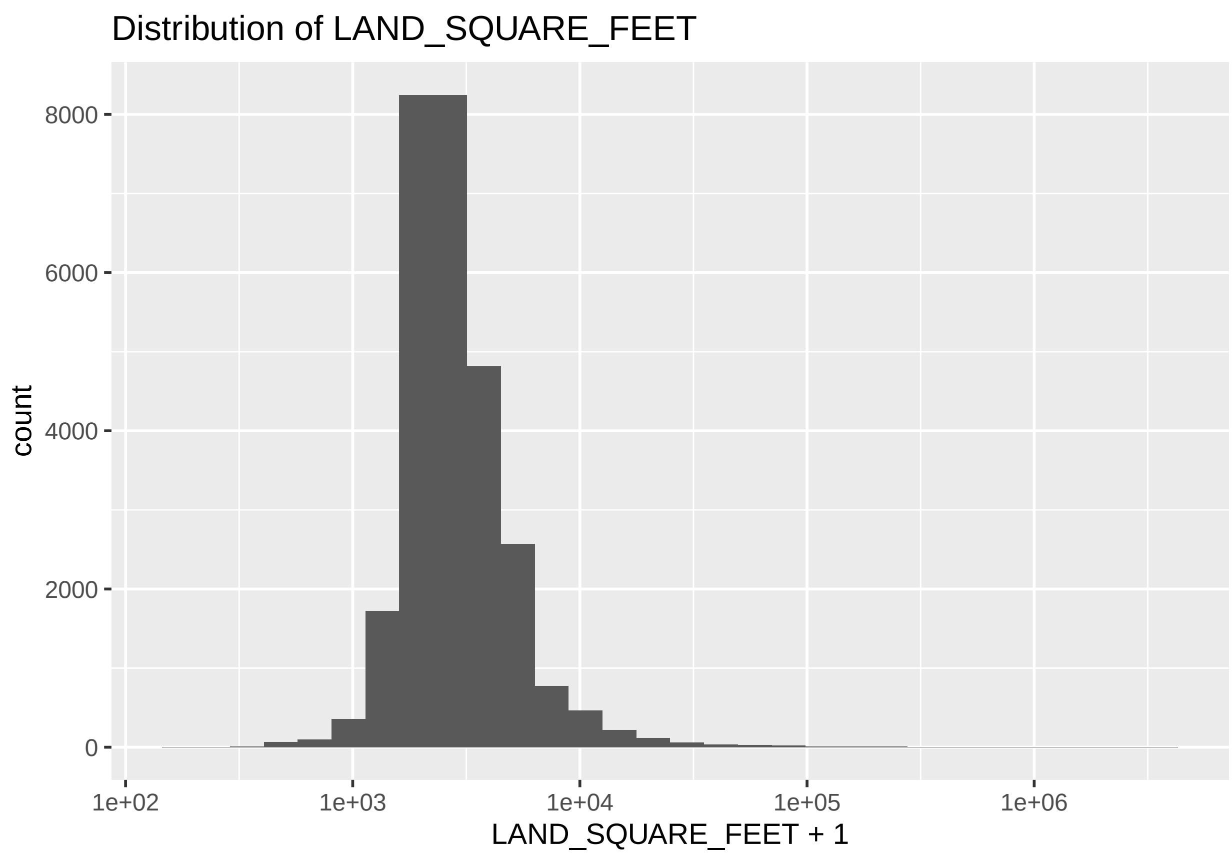
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



we also deleted the rows with 0 LAND_SQURE_FEET

```
# drop the houses with 0 LAND_SQUARE_FEET
nyc_sales<- nyc_sales%>%filter(LAND_SQUARE_FEET>0)
nyc_sales%>%ggplot(aes(LAND_SQUARE_FEET+1))+
  geom_histogram()+scale_x_continuous(trans = 'log10')+
  ggtitle("Distribution of LAND_SQUARE_FEET")
```

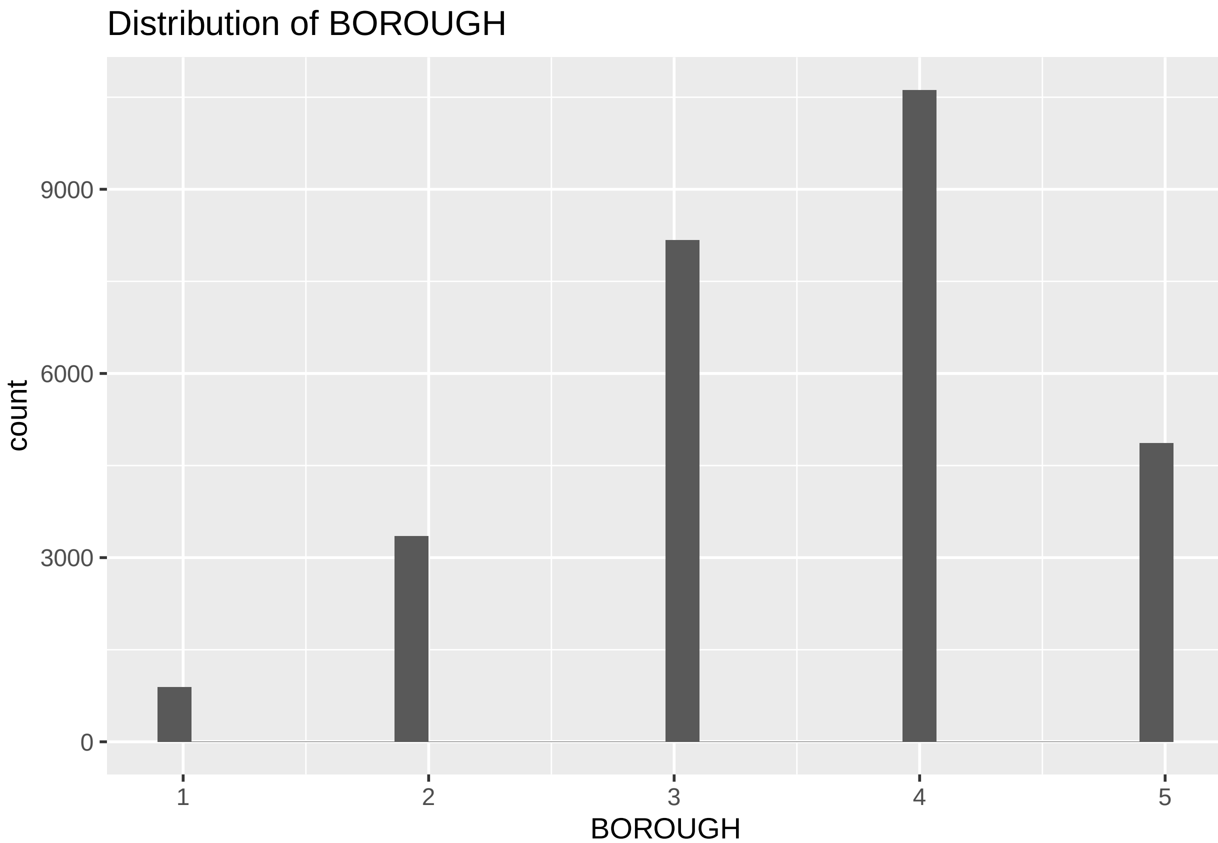
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The varaible BOROUGH is a digit code for the borough the property is located in. For the convinience for the further analysis, we convert it to 5 independent numeric variables: Manhattan,Bronx, Brooklyn, Queens, and State_Island.

```
#plot the histogram of variable BOROUGH
nyc_sales%>%ggplot(aes(BOROUGH))+
  geom_histogram()+
  ggtitle("Distribution of BOROUGH")
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



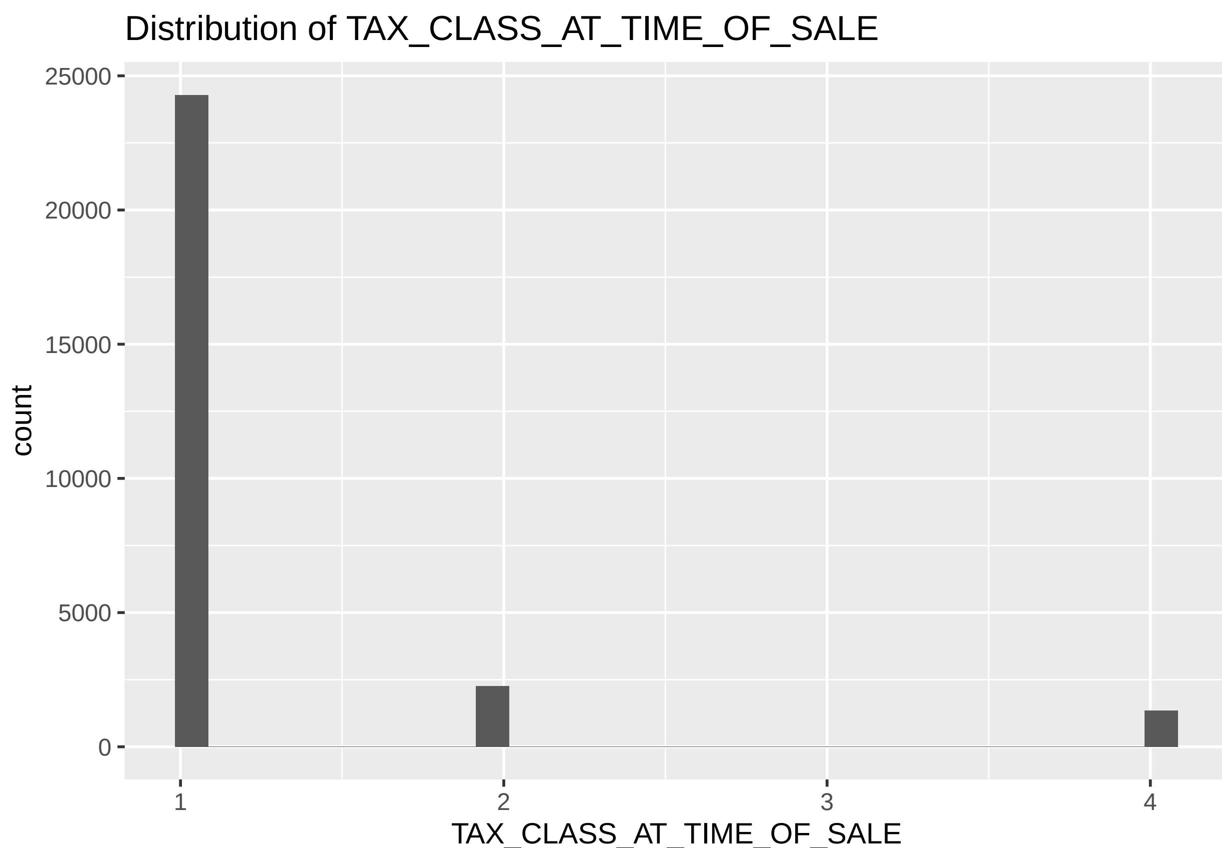

```
#convert categorical variable BOROUGH into 5 numeric variables for the convinient of further analysis
nyc_sales<-nyc_sales%>%mutate(Manhattan = ifelse(BOROUGH==1,1,0))
nyc_sales<-nyc_sales%>%mutate(Bronx = ifelse(BOROUGH==2,1,0))
nyc_sales<-nyc_sales%>%mutate(Brooklyn = ifelse(BOROUGH==3,1,0))
nyc_sales<-nyc_sales%>%mutate(Queens = ifelse(BOROUGH==4,1,0))
nyc_sales<-nyc_sales%>%mutate(State_Island = ifelse(BOROUGH==5,1,0))

#delete the BOROUGH variable from the dataset
nyc_sales$BOROUGH<-NULL
```

Similarly, we also convert the variable TAX_CLASS_AT_TIME_OF_SALE into 3 seperate numeric variables:Taxclass1,Taxclass2,Taxclass4.

```
#plot the histogram of variable TAX_CLASS_AT_TIME_OF_SALE
nyc_sales%>%ggplot(aes(TAX_CLASS_AT_TIME_OF_SALE))+
  geom_histogram()+
  ggtitle("Distribution of TAX_CLASS_AT_TIME_OF_SALE")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#convert categorical variable TAX_CLASS_AT_TIME_OF_SALE" into 3 numeric variables
nyc_sales<-nyc_sales%>%mutate(Taxclass1 = ifelse(TAX_CLASS_AT_TIME_OF_SALE==1,1,0))
nyc_sales<-nyc_sales%>%mutate(Taxclass2 = ifelse(TAX_CLASS_AT_TIME_OF_SALE==2,1,0))
nyc_sales<-nyc_sales%>%mutate(Taxclass4 = ifelse(TAX_CLASS_AT_TIME_OF_SALE==4,1,0))

#delete the variable TAX_CLASS_AT_TIME_OF_SALE from the dataset
nyc_sales$TAX_CLASS_AT_TIME_OF_SALE <-NULL
```

From above distribution figures, we found the variable SALE_PRICE,GROSS_SQUARE_FEET and LAND_SQUARE_FEET are highly right skewed and then we will use logarithmic transformationd to tranform them into ones that are more approximatedly normal variables.

```
# Use logarithmic transformations to transforming highly skewed variables
# into ones that is more approximately normal.
nyc_sales$SALE_PRICE<-log(nyc_sales$SALE_PRICE)
nyc_sales$GROSS_SQUARE_FEET<-log(nyc_sales$GROSS_SQUARE_FEET)
nyc_sales$LAND_SQUARE_FEET<-log(nyc_sales$LAND_SQUARE_FEET)
```

```
# check the dimension of the data after data cleaning
dim(nyc_sales)
```

```
## [1] 27901    23
```

Now we have a clean dataset which have 27,901 rows and 23 variables. Before we begin to build our predictive models, we need to split the dataset into 2 part: training set and test set. Tranining set is used to build the model and test set is used to evaluate the model with RMSE metric.Considering the size of the dataset, we randomly select 80% of data as training data and 20% as test data

```
# split the data into training data and test data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = nyc_sales$SALE_PRICE, times = 1, p = 0.2, list = FALSE)
nyc_training <- nyc_sales[-test_index,]
nyc_test <- nyc_sales[test_index,]
```

2.3 Building predictive models

In this project, we used the typical error loss, the residual mean squared error (RMSE),to evaluate the methods. Following is a function that computes the RMSE for vectors of ratings and their corresponding predictors:

To evaluate the dependent variables that are most important in predicting SALE_PRICE, we first calculate the correlation between SALE_RPICE and all other numeric variables in the traning dataset.

```
#get the index of numeric column
num_vars <- which(sapply(nyc_training, is.numeric))
#get the name of numeric column
num_vars_colnames <- data.table(names(num_vars))

#get the table with all numeric variabls
nyc_training_num <- nyc_training[, num_vars]
nyc_test_num<-nyc_test[,num_vars]

#do the correlations of all numeric variables in pairwise
cor_num_vars <- cor(nyc_training_num, use="pairwise.complete.obs")

#sort on decreasing correlations with SalePrice
cor_sorted <- as.matrix(sort(cor_num_vars[,"SALE_PRICE"], decreasing = TRUE))
cor_sorted
```

```
##                          [,1]
## SALE_PRICE         1.00000000
## GROSS_SQUARE_FEET  0.68767700
## Manhattan          0.47700871
## Taxclass2          0.38295682
## Taxclass4          0.33929651
## LAND_SQUARE_FEET   0.29142930
## TOTAL_UNITS        0.21016152
## Brooklyn           0.19575538
## RESIDENTIAL_UNITS  0.18655587
## COMMERCIAL_UNITS   0.13669412
## id                -0.02763828
## Queens            -0.11384207
## Bronx             -0.11401774
## YEAR_BUILT        -0.12420381
## State_Island      -0.21831516
## Taxclass1         -0.52836709
```

2.3.1 Model 1: Linear regression model with one variable

According to above correlation result, we found GROSS_SQUARE_FEET has the highest correlation with SALE_PRICE. So we will use GROSS_SQUAR_FEET as depent variable to build our fist linear regression model to predict the SALE_PRICE.

```
#model 1, linear regession model with one virable
set.seed(1, sample.kind="Rounding")
model_1 <- lm(SALE_PRICE ~ GROSS_SQUARE_FEET, data = nyc_training_num)
summary(model_1)
```

```
##
## Call:
## lm(formula = SALE_PRICE ~ GROSS_SQUARE_FEET, data = nyc_training_num)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4211 -0.3335 -0.0074  0.3357  4.9476
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       7.279141   0.044187   164.7   <2e-16 ***
## GROSS_SQUARE_FEET 0.813423   0.005749   141.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.626 on 22317 degrees of freedom
## Multiple R-squared:  0.4729, Adjusted R-squared:  0.4729
## F-statistic: 2.002e+04 on 1 and 22317 DF,  p-value: < 2.2e-16
```

```
# use the model to predict on test data
prediction <- predict(model_1, nyc_test, type="response")

model_1_rmse <- RMSE(prediction, nyc_test$SALE_PRICE)
model_1_rmse
```

```
## [1] 0.605191
```

```
RMSE_table <- data_frame(Method = "Linear regression model with only gross square feet effect",RMSE = mo
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

The RMSE of this model is 0.605191, which is a good start. We will try to improve it further with other methods.

2.3.2 Model 2: Linear regression model with multiple variables

Next, we will use all the numeric varibles as depent varibles to build the linear regression model.

```r
#model 2, linear regression model with multiple varibles
set.seed(1, sample.kind="Rounding")
model_2 <- lm(SALE_PRICE ~RESIDENTIAL_UNITS+COMMERCIAL_UNITS+TOTAL_UNITS+LAND_SQUARE_FEET+GROSS_SQUARE_I
summary(model_2)
```

```
##
## Call:
## lm(formula = SALE_PRICE ~ RESIDENTIAL_UNITS + COMMERCIAL_UNITS +
##     TOTAL_UNITS + LAND_SQUARE_FEET + GROSS_SQUARE_FEET + YEAR_BUILT +
##     Manhattan + Bronx + Brooklyn + Queens + State_Island + Taxclass1 +
##     Taxclass2 + Taxclass4, data = nyc_training_num)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9465 -0.2624  0.0445  0.3002  3.7485
##
## Coefficients: (2 not defined because of singularities)
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.5249336  0.2813996  30.295   <2e-16 ***
## RESIDENTIAL_UNITS -0.0602351  0.0544338  -1.107   0.2685
## COMMERCIAL_UNITS  -0.0578179  0.0544315  -1.062   0.2881
## TOTAL_UNITS        0.0575433  0.0544364   1.057   0.2905
## LAND_SQUARE_FEET   0.1336097  0.0082382  16.218   <2e-16 ***
## GROSS_SQUARE_FEET  0.5930682  0.0082636  71.768   <2e-16 ***
## YEAR_BUILT        -0.0002760  0.0001393  -1.981   0.0476 *
## Manhattan          1.4717861  0.0271794  54.151   <2e-16 ***
## Bronx             -0.1386409  0.0151829  -9.131   <2e-16 ***
## Brooklyn           0.4551607  0.0136096  33.444   <2e-16 ***
## Queens             0.2313737  0.0115305  20.066   <2e-16 ***
## State_Island              NA         NA      NA       NA
## Taxclass1         -0.3618049  0.0198984 -18.183   <2e-16 ***
## Taxclass2         -0.2135990  0.0222838  -9.585   <2e-16 ***
## Taxclass4                 NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5512 on 22306 degrees of freedom
## Multiple R-squared:  0.5916, Adjusted R-squared:  0.5913
## F-statistic:  2692 on 12 and 22306 DF,  p-value: < 2.2e-16
```

```r
# use the model to predict on test data
prediction <- predict(model_2, nyc_test, type="response")

model_2_rmse <- RMSE(prediction, nyc_test$SALE_PRICE)
model_2_rmse
```

```
## [1] 0.5378123
```

```r
RMSE_table <- rbind(RMSE_table,
                    data_frame(Method = "Linear regression model with multiple effects",
                               RMSE = model_2_rmse))
```

We can see the RMSE of this model has been improved to 0.5378123.

2.3.3 Model 3: Ridge regression model

Ridge regression is an extension of linear regression where the loss function is modified to minimize the complexity of the model. This modification is done by adding a penalty parameter that is equivalent to the square of the magnitude of the coefficients. Here we will build a ridge regression model to do the prediction.

```
#model 3, ridge regression model
x = model.matrix(SALE_PRICE~., nyc_training_num)[,-1] # trim off the first column,leaving only the pred
y<-nyc_training_num$SALE_PRICE

x_test = model.matrix(SALE_PRICE~., nyc_test_num)[,-1]
y_test <-nyc_test_num$SALE_PRICE

#train the model
lambdas <- 10^seq(2, -3, by = -.1)
model_3 <- cv.glmnet(x, y, alpha = 0, lambda = lambdas)
summary(model_3)
```

```
##            Length Class  Mode
## lambda     51     -none- numeric
## cvm        51     -none- numeric
## cvsd       51     -none- numeric
## cvup       51     -none- numeric
## cvlo       51     -none- numeric
## nzero      51     -none- numeric
## call        5     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
```

```
optimal_lambda <- model_3$lambda.min
optimal_lambda
```

```
## [1] 0.001995262
```

```
# use the model and optimal lambda  to predict on test data
prediction<- predict(model_3, s = optimal_lambda, newx = x_test)
model_3_rmse<-RMSE(prediction,nyc_test$SALE_PRICE)
model_3_rmse
```

```
## [1] 0.5344493
```

```
RMSE_table <- rbind(RMSE_table,
                    data_frame(Method = "Ridge regression model",
                               RMSE = model_3_rmse))
```

The RMSE of ridge regression model is 0.53344493, which is better than the above 2 linear regression models.

2.3.4 Model 4: Lasso regession model

Lasso regression, is also a modification of linear regression. In lasso, the loss function is modified to minimize the complexity of the model by limiting the sum of the absolute values of the model coefficients (also called the l1-norm). Here, we will build the lasso regression model to predict the house price.

```
#model 4, lasso regression
set.seed(1, sample.kind="Rounding")
#train the model
lasso_control <-trainControl(method="cv", number=5)
lassoGrid <- expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.0005))
model_4 <- train(SALE_PRICE ~ ., data = nyc_training_num, method='glmnet',
                     trControl= lasso_control, tuneGrid=lassoGrid)
summary(model_4)
```

```
##             Length Class      Mode
## a0            68   -none-     numeric
## beta        1020   dgCMatrix  S4
## df            68   -none-     numeric
## dim            2   -none-     numeric
## lambda        68   -none-     numeric
## dev.ratio     68   -none-     numeric
## nulldev        1   -none-     numeric
## npasses        1   -none-     numeric
## jerr           1   -none-     numeric
## offset         1   -none-     logical
## call           5   -none-     call
## nobs           1   -none-     numeric
## lambdaOpt      1   -none-     numeric
## xNames        15   -none-     character
## problemType    1   -none-     character
## tuneValue      2   data.frame list
## obsLevels      1   -none-     logical
## param          0   -none-     list
```

```
#use the model to predict on test data
prediction <- predict(model_4, nyc_test)
model_4_rmse <- RMSE(prediction, nyc_test$SALE_PRICE)
model_4_rmse
```

```
## [1] 0.5345257
```

```
RMSE_table <- rbind(RMSE_table,
                    data_frame(Method = "Lasso regression model",
                               RMSE = model_4_rmse))
```

The RMSE of Lasso regression model on test data set is 0.5345257.

2.3.5 Model 5: Elastic net regression model

Elastic net regression combines the properties of ridge and lasso regression. It works by penalizing the model using both the 1l2-norm1 and the 1l1-norm1. The model can be easily built using the caret package, which automatically selects the optimal value of parameters alpha and lambda. Here, we also build a elastic net regression model to predict the house price.

```
#model 5, Elastic Net Regression
set.seed(1, sample.kind="Rounding")
# Set training control
elastic_control <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = TRUE)
# Train the model
model_5 <- train(SALE_PRICE ~ ., data = nyc_training_num, method = "glmnet",
        preProcess = c("center", "scale"), tuneLength = 10,trControl = elastic_control)
```

```
summary(model_5)
```

```
##             Length Class      Mode
## a0            69   -none-     numeric
## beta        1035   dgCMatrix  S4
## df            69   -none-     numeric
## dim            2   -none-     numeric
## lambda        69   -none-     numeric
## dev.ratio     69   -none-     numeric
## nulldev        1   -none-     numeric
## npasses        1   -none-     numeric
## jerr           1   -none-     numeric
## offset         1   -none-     logical
## call           5   -none-     call
## nobs           1   -none-     numeric
## lambdaOpt      1   -none-     numeric
## xNames        15   -none-     character
## problemType    1   -none-     character
## tuneValue      2   data.frame list
## obsLevels      1   -none-     logical
## param          0   -none-     list
```

```
# Best tuning parameter
model_5$bestTune
```

```
##       alpha      lambda
## 4 0.3721239 0.004793309
```

```
# use the model to make predictions on test set
prediction <- predict(model_5, nyc_test)
model_5_rmse <- RMSE(prediction, nyc_test$SALE_PRICE)
model_5_rmse
```

```
## [1] 0.5346134
```

```r
RMSE_table <- rbind(RMSE_table,
                    data_frame(Method = "Elastic net regression model",
                               RMSE = model_5_rmse))
```

The RMSE of elastic net regression model is 0.5346134.

2.3.6 Model 6: Random forest regression model

Random Forest is a popular machine learning model that is commonly used for both classification and regression.A Random Forest's nonlinear nature can give it a leg up over linear algorithms, making it a great option.

```r
#model 6,random forest regression model
set.seed(1, sample.kind="Rounding")
model_6 <- randomForest(SALE_PRICE ~., data = nyc_training_num)
summary(model_6)
```

```
##                 Length Class  Mode
## call                3  -none- call
## type                1  -none- character
## predicted       22319  -none- numeric
## mse               500  -none- numeric
## rsq               500  -none- numeric
## oob.times       22319  -none- numeric
## importance         15  -none- numeric
## importanceSD        0  -none- NULL
## localImportance     0  -none- NULL
## proximity           0  -none- NULL
## ntree               1  -none- numeric
## mtry                1  -none- numeric
## forest             11  -none- list
## coefs               0  -none- NULL
## y               22319  -none- numeric
## test                0  -none- NULL
## inbag               0  -none- NULL
## terms               3  terms  call
```

```r
#use the model to make prediction on test data
prediction <- predict(model_6, nyc_test)
model_6_rmse <- RMSE(prediction, nyc_test$SALE_PRICE)
model_6_rmse
```

```
## [1] 0.4441618
```

```r
RMSE_table <- rbind(RMSE_table,
                    data_frame(Method = "Random forest regression model",
                               RMSE = model_6_rmse))
```

The RMSE of random forest regression model is 0.4441618.

3. Results

We built 6 regression models here to predict house prices and the RMSE of 6 methods are as following table

```r
#results for all the models
RMSE_table %>% knitr::kable(caption = "RMSE of predictive models ")
```

Table 1: RMSE of predictive models

Method	RMSE
Linear regression model with only gross square feet effect	0.6051910
Linear regression model with multiple effects	0.5378123
Ridge regression model	0.5344493
Lasso regression model	0.5345257
Elastic net regression model	0.5346134
Random forest regression model	0.4441618

4. Conclusion

In this project, we built 6 regression models to predict the house price on NYC property sales dataset. Overall, all the models are performing well with stable RMSE values. Among the 6 predictive models, fandom forest regression model got the best performance, which RMSE is 0.4441618. All the data and code and report can be downloaded from https://github.com/zhan-us/NYC_Sales.

References

1. https://rafalab.github.io/dsbook/
2. https://www.kaggle.com/new-york-city/nyc-property-sales
3. https://github.com/zhan-us/NYC_Sales