NYC Property Price Prediction Using Regression

Christopher Pickering

2019 06 13

Introduction

It is without a doubt that many cities have become extremely successful over recent years. Many authors have followed and described this and have demonstrated the clear advantage that cities like New York have over others (Glaeser, 2011). As an answer to decaying urban cores, Richard Florida started to use metrics to compare cities and advocated for changes to attract the so-called *creative class* in order to stimulate innovation and also foster new and vibrant neighbourhoods for these professionals to play and stimulate the local economy (Florida, 2002). Over just 15 years this development has led to an enormous income gap between the *creatives* and the other residents of the cities and home prices have increased such that displacement of people is common (Florida, 2017). Cities like New York, San Francisco and London are so expensive that wealthy buy property simply as a trophy or a sign that they can invest in buildings with the highest value.

As an MSc student in Urban Studies the prediction of house prices is very interesting. Prices are generally higher near important transportation nodes like train stations (Debrezion et al., 2007). However, there is also a psychological and social component that can determine how much someone is willing to pay for a home in a desirable location (Smith, 2011). To explore this I looked for a dataset that I could use to test some of these observations.

Kaggle had a relevant dataset which can be found at https://www.kaggle.com/new-york-city/nyc-property-sales (https://www.kaggle.com/new-york-city/nyc-property-sales). This semi-cleaned dataset includes all properties sold between September 2016 and September 2017 in the 5 boroughs of New York City. For clarification, the corresponding numbers in the dataset are as follows: 1 = Manhattan, 2 = Bronx, 3 = Brooklyn, 4 = Queens and 5 = Staten Island. To focus on actual homes, building classifications were used to select out appropriate data. To summarize these codes, A = single-family, B = two-family, C = walk-up apartments, D = elevator apartments and L = lofts.

Aim

The aim of this project was to develop a machine learning regression model that would predict sale price (outcome) from available building characteristics (5 predictors). The relative success of this model in terms of RMSE could be used to comment on whether home prices are a still a function of lot size, area and location or if they are simply a product of psychology or wealthy investment desires.

Methods

The dataset for this project was Kaggle's NYC Property Sales which can be downloaded at https://www.kaggle.com/new-york-city/nyc-property-sales (https://www.kaggle.com/new-york-city/nyc-property-sales) or as a csv in the GitHub for this project.

Pre-processing and wrangling

```
nycproperties <- read_csv("C:/RCoding/nyc-property-sales/nyc-rolling-sales.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   X1 = col_double(),
##   BOROUGH = col_double(),
##   BLOCK = col_double(),
##   LOT = col_double(),
##   `EASE-MENT` = col_logical(),
##   `ZIP CODE` = col_double(),
##   `RESIDENTIAL UNITS` = col_double(),
##   `COMMERCIAL UNITS` = col_double(),
##   `TOTAL UNITS` = col_double(),
##   `YEAR BUILT` = col_double(),
##   `TAX CLASS AT TIME OF SALE` = col_double(),
##   `SALE DATE` = col_datetime(format = "")
## )
```

```
## See spec(...) for full column specifications.
```

```
colnames(nycproperties)
```

```
##  [1] "X1"                             "BOROUGH"
##  [3] "NEIGHBORHOOD"                   "BUILDING CLASS CATEGORY"
##  [5] "TAX CLASS AT PRESENT"           "BLOCK"
##  [7] "LOT"                            "EASE-MENT"
##  [9] "BUILDING CLASS AT PRESENT"      "ADDRESS"
## [11] "APARTMENT NUMBER"               "ZIP CODE"
## [13] "RESIDENTIAL UNITS"              "COMMERCIAL UNITS"
## [15] "TOTAL UNITS"                    "LAND SQUARE FEET"
## [17] "GROSS SQUARE FEET"              "YEAR BUILT"
## [19] "TAX CLASS AT TIME OF SALE"      "BUILDING CLASS AT TIME OF SALE"
## [21] "SALE PRICE"                     "SALE DATE"
```

```
dim(nycproperties)
```

```
## [1] 84548    22
```

There were 23 variables (columns) of various type, some which were more informative than others. After inspection and based on previous investigation of property prices several variables were selected. The **outcome** was *SALE PRICE* and some candidates for the predictors, *LAND SQUARE FEET*, *GROSS SQUARE*

FEET and *SALE DATE*, were converted to numeric or date. A new field, *BuildingAge*, was calculated from the Year Built data.

```
nycproperties$`SALE PRICE` <- as.numeric(as.character((nycproperties$`SALE PRICE`)))
```

```
## Warning: NAs introduced by coercion
```

```
class(nycproperties$`SALE PRICE`)
```

```
## [1] "numeric"
```

```
nycproperties$`LAND SQUARE FEET` <- as.numeric(as.character((nycproperties$`LAND SQUAR
E FEET`)))
```

```
## Warning: NAs introduced by coercion
```

```
class(nycproperties$`LAND SQUARE FEET`)
```

```
## [1] "numeric"
```

```
nycproperties$`GROSS SQUARE FEET` <- as.numeric(as.character((nycproperties$`GROSS SQU
ARE FEET`)))
```

```
## Warning: NAs introduced by coercion
```

```
class(nycproperties$`GROSS SQUARE FEET`)
```

```
## [1] "numeric"
```

```
nycproperties$`SALE DATE` <- as.Date(as.character((nycproperties$`SALE DATE`)))
class(nycproperties$`SALE DATE`)
```

```
## [1] "Date"
```

```
# Create new column for BuildingAge and fill with data
nycproperties[c("BuildingAge")] <- 2019 - nycproperties$`YEAR BUILT`
```

The dataset had many missing values in most variables and this presented a problem for analysis. One could replace these values with 0 but instead the following code was used to convert NA values to the mean value for the respective column/variable.

```
checknumeric <- sapply(nycproperties, is.numeric)
nycproperties[checknumeric] <- lapply(nycproperties[checknumeric], na.aggregate)
```

Since these data are for the sale of all types of property it presented a challenge for predicting the prices of living spaces specifically. For example, sale prices of 0 were often found in the commercial or government properties. Condominiums (Class R) lacked information on lot size or gross square feet and these missing values complicated the modeling considerably. To build a model using the most predictors, the dataset was reduced (**livingspaces**) to include only building classes with the most complete data on actual home or apartment prices.

```
livingspaces <- nycproperties %>%
  filter(str_detect(`BUILDING CLASS AT TIME OF SALE` ,"A") |
         str_detect(`BUILDING CLASS AT TIME OF SALE` ,"B") |
         str_detect(`BUILDING CLASS AT TIME OF SALE` ,"C") |
         str_detect(`BUILDING CLASS AT TIME OF SALE` ,"D") |
         str_detect(`BUILDING CLASS AT TIME OF SALE` ,"L"))
```

Data were then explored and potential predictor variables contained many extreme outliers or unrealistic values (e.g. sale price = $0). These were excluded using the following code.

```
livingspaces <- livingspaces %>%
  filter((`SALE PRICE` > 20 & `SALE PRICE` < 100000000) &
           BuildingAge > 0 & BuildingAge < 500 &
           `TOTAL UNITS` > 0 & `TOTAL UNITS`< 500 &
           `GROSS SQUARE FEET`> 0 & `LAND SQUARE FEET` > 0 )
```

At this stage the dataset was reduced to 34530 observations from an initial 84548. Excluding over half of the dataset in the pre-processing stage raises many issues which will be addressed later.

Data visualization

Complete plots of all potential predictors can be seen by running the entire R code file. For this report key figures were selected.

```
skewness(livingspaces$`SALE PRICE`, type = 1)
```

```
## [1] 16.23248
```

```
skewness(livingspaces$BuildingAge, type = 1)
```

```
## [1] -0.7117104
```

```
skewness(livingspaces$`GROSS SQUARE FEET`, type = 1)
```

```
## [1] 21.92838
```

```
skewness(livingspaces$`LAND SQUARE FEET`, type = 1)
```

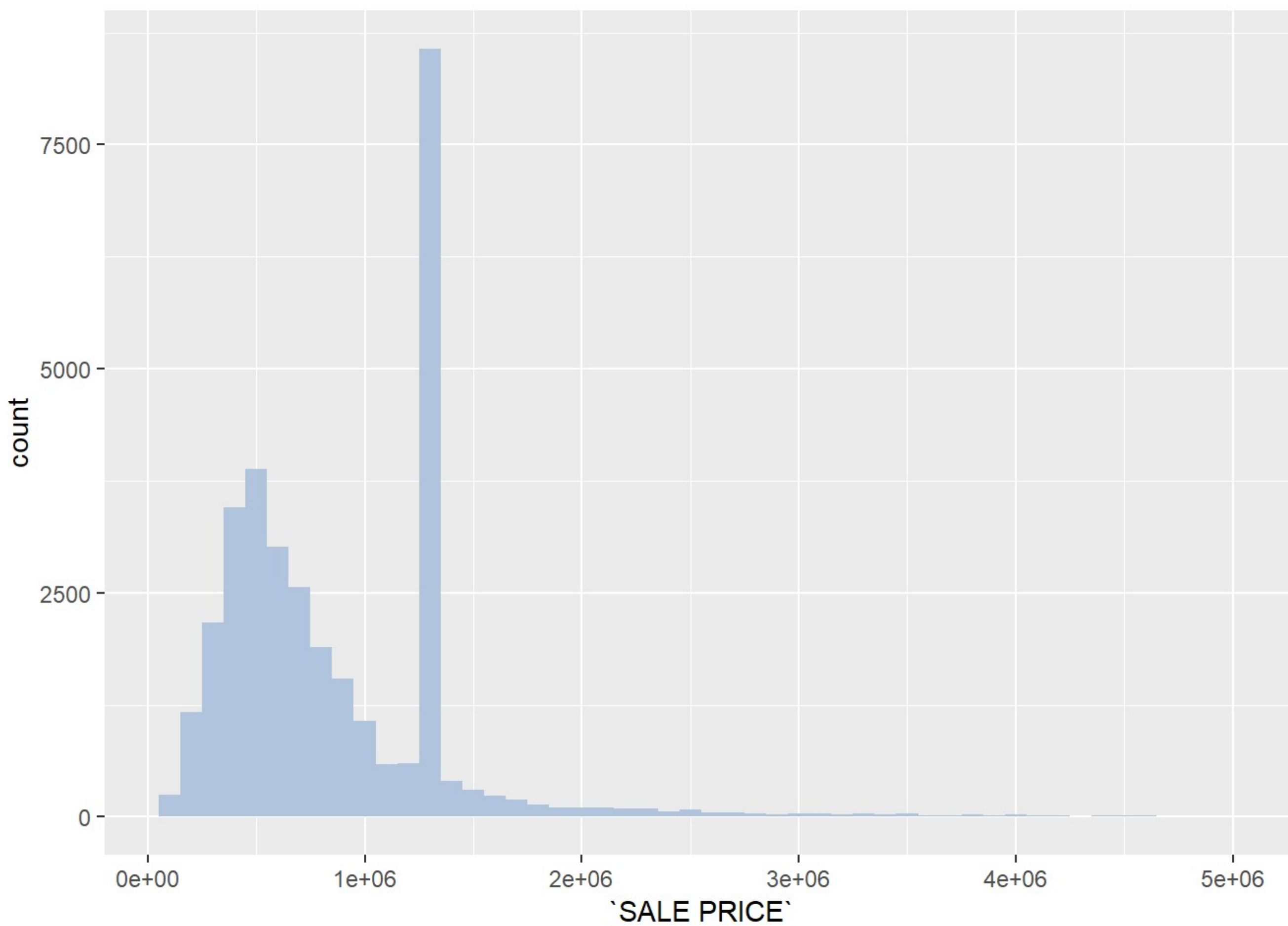
```
## [1] 61.6832
```

```
skewness(livingspaces$`TOTAL UNITS`, type = 1)
```

```
## [1] 18.96063
```

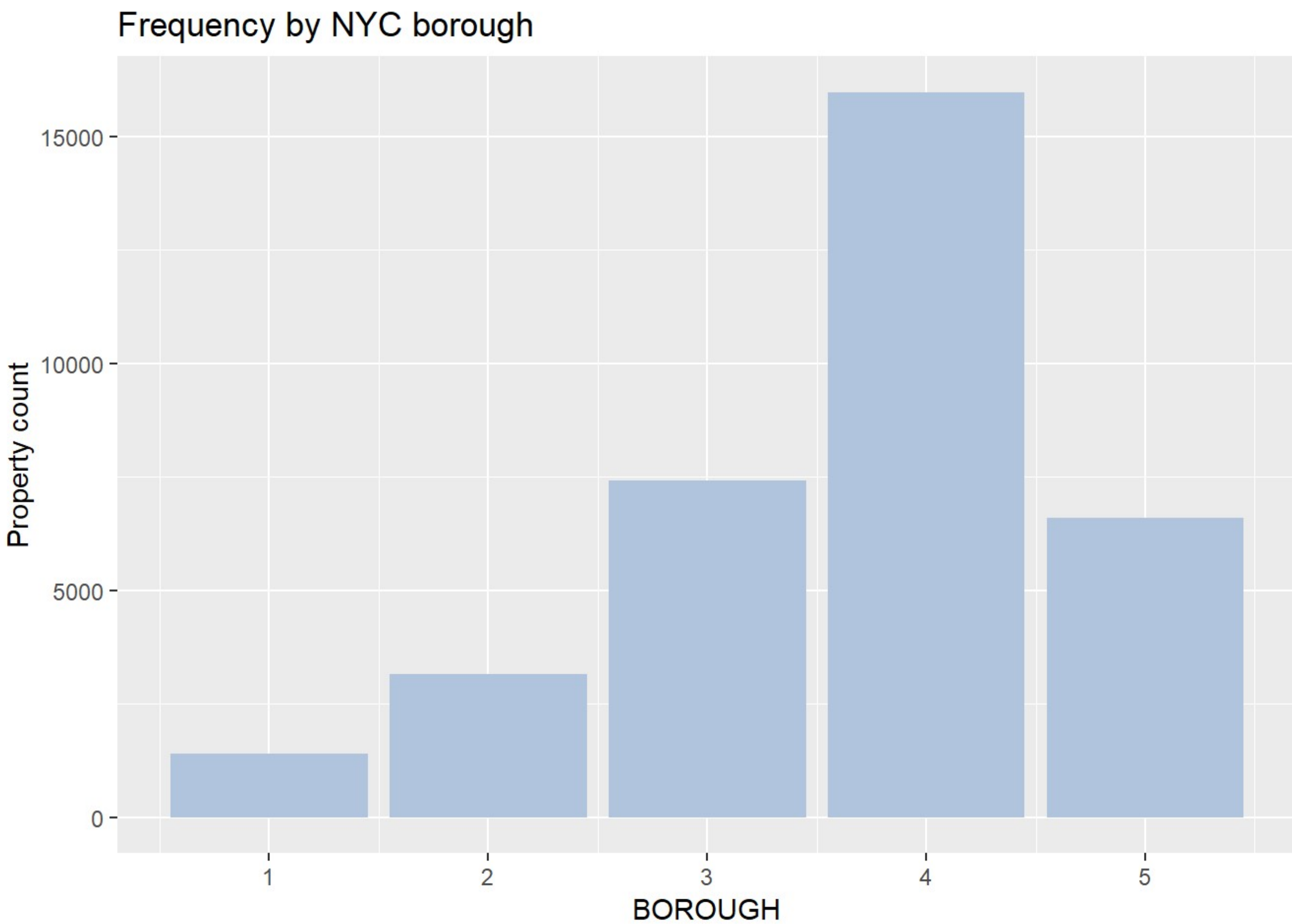
All data except for BuildingAge were skewed and the range was very large. During the development and testing stages a log transformation of these predictors was made and this helped in RMSE calculations. However, many regression models are not valid on log transformed data and produce a model that cannot be easily used with actual sale price numbers. So despite problems with the distribution, raw data was used for the remainder of the project.

```
livingspaces %>%
  filter(`SALE PRICE` > 100000 & `SALE PRICE` < 5000000) %>%
  ggplot(aes(`SALE PRICE`)) +
  geom_histogram(binwidth = 100000, fill = "lightsteelblue")
```



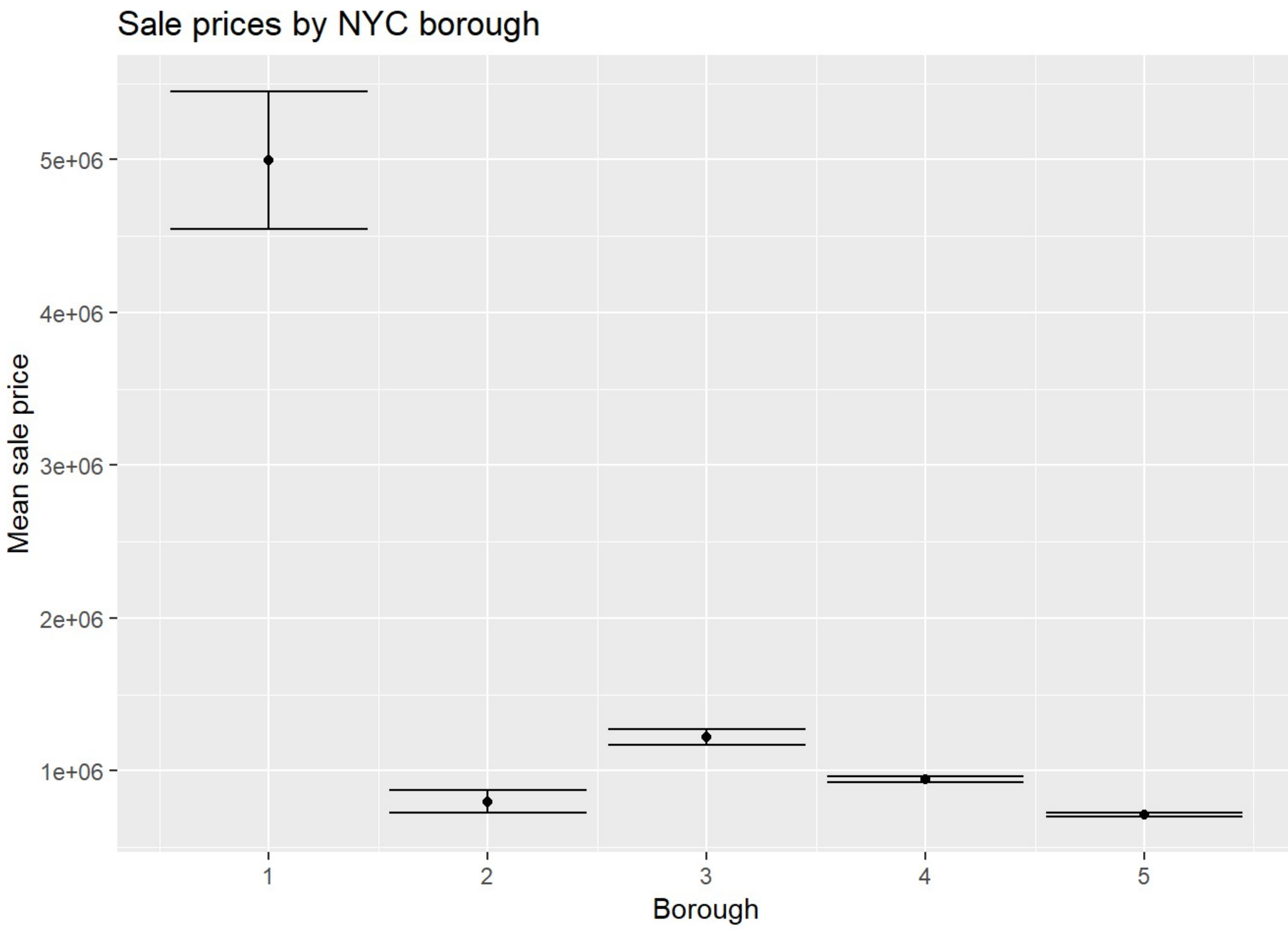
The left shift of the *Sale Price* data is clearly visible in this graph. Note also the peak which is a result of replacing NA values with the mean value for the Sale Price group.

```
livingspaces %>%
  ggplot(aes(BOROUGH)) +
  geom_bar(fill = "lightsteelblue") +
  ylab("Property count") +
  ggtitle("Frequency by NYC borough")
```



The number of properties on Manhattan (Borough = 1) dropped significantly after selection by property type. This could also reflect a lack of houses or rentals on Manhattan.

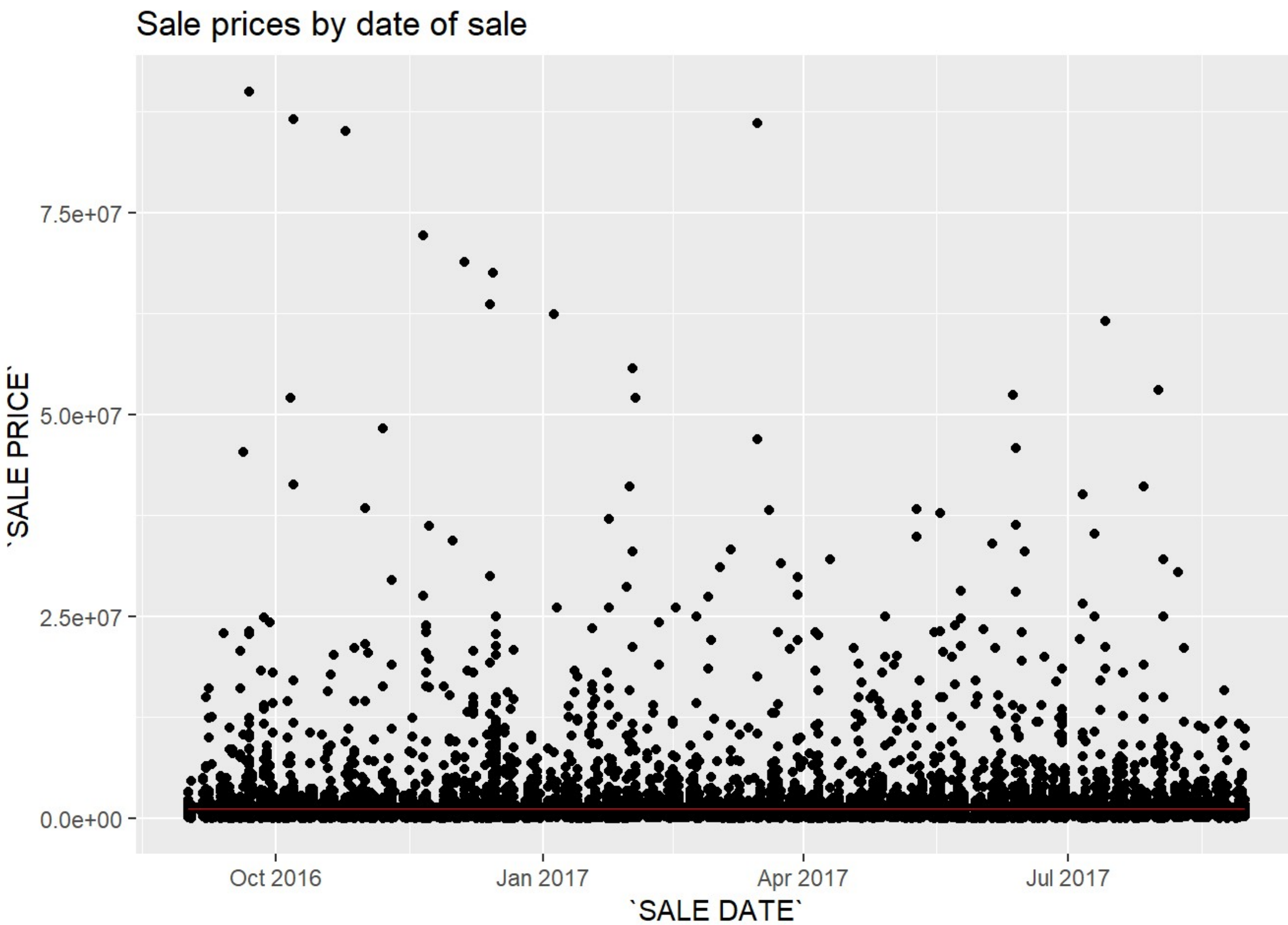
```
livingspaces %>%
  group_by(BOROUGH) %>%
  summarise(n = n(), avg = mean(`SALE PRICE`), se = sd(`SALE PRICE`)/sqrt(n())) %>%
  ggplot(aes(x = BOROUGH, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  xlab("Borough") +
  ylab("Mean sale price") +
  ggtitle("Sale prices by NYC borough")
```



Another problem is that prices in Manhattan were much higher than the other 4 boroughs. This creates an immediate outling group which may more may not affect the regression modeling. This could be a reason to exclude the Manhattan data altogether.

The next step was to check whether there is a relationship between *Sale Price* and the potential predictors.

```
livingspaces %>%
  ggplot(aes(`SALE DATE`,`SALE PRICE`)) +
  geom_point() +
  geom_smooth(na.rm = TRUE, color = "red", size = 0.1, method = lm) +
  ggtitle("Sale prices by date of sale")
```

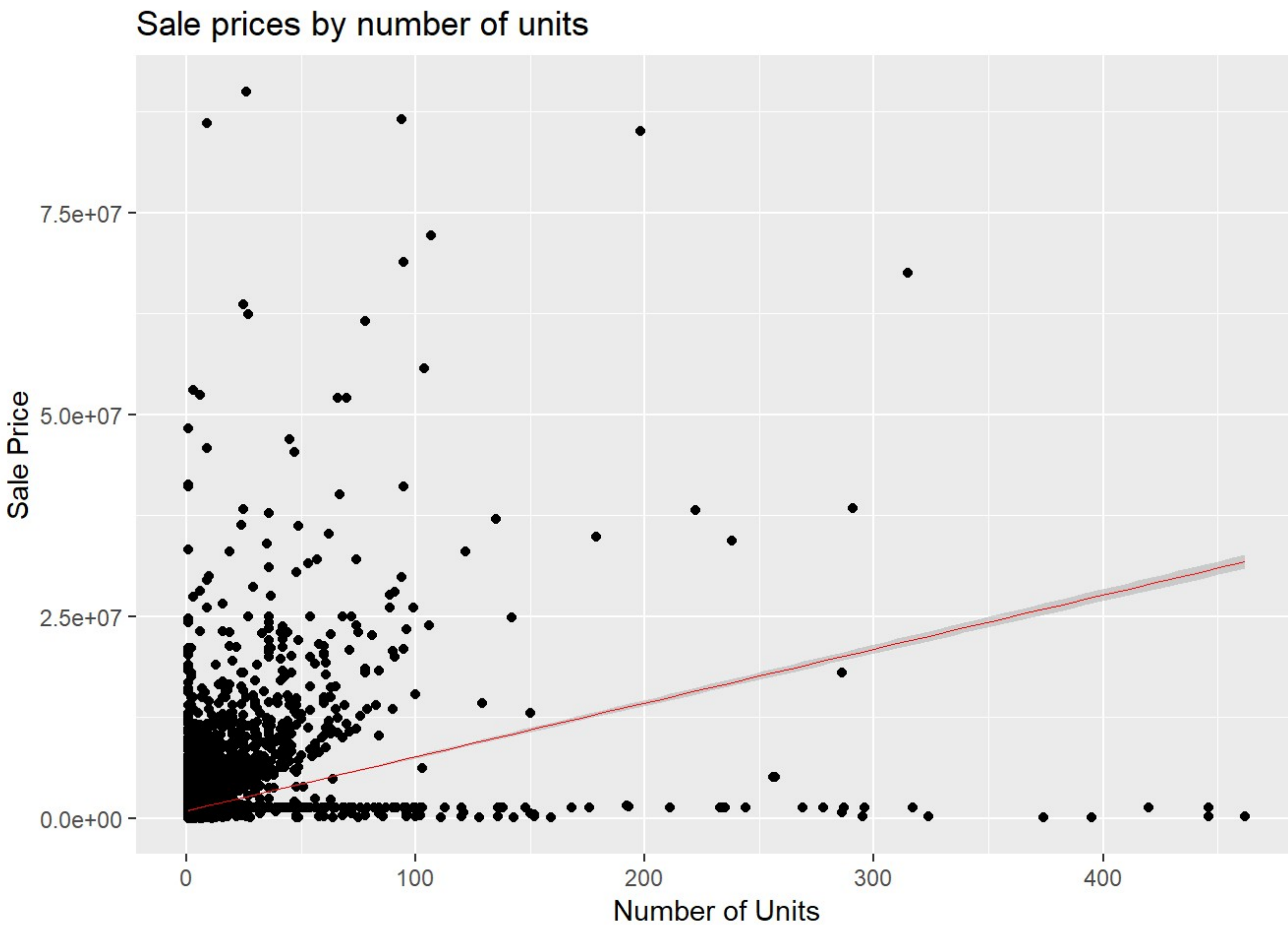


Sale Price did not seem to vary by *Sale Date* suggesting that overall prices were fairly stable over the 1-year data period. Because of this *Sale Date* was excluded from the predictor list.

```
livingspaces %>%
  ggplot(aes(BuildingAge,`SALE PRICE`)) +
  geom_point() +
  geom_smooth(color = "red", size = 0.1, method = lm) +
  xlab("Building Age (years)") +
  ylab("Sale Price") +
  ggtitle("Sale prices by age of building")
```

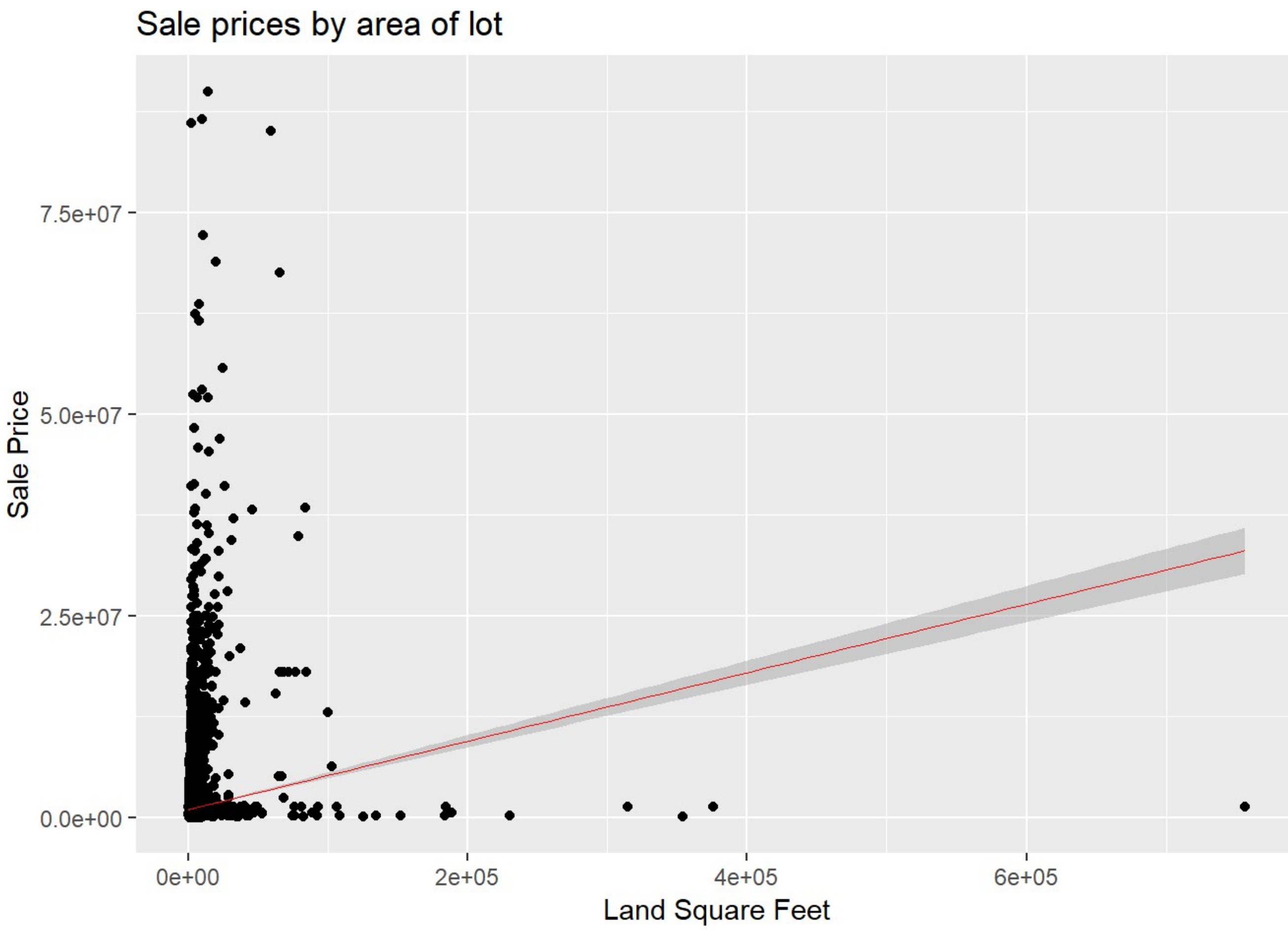
Sale Price did vary by *BuildingAge* so this chosen as a predictor. Note that buildings around 100 years old had the highest price.

```
livingspaces %>%
  ggplot(aes(`TOTAL UNITS`,`SALE PRICE`)) +
  geom_point() +
  geom_smooth(color = "red", size = 0.1, method = lm) +
  xlab("Number of Units") +
  ylab("Sale Price") +
  ggtitle("Sale prices by number of units")
```



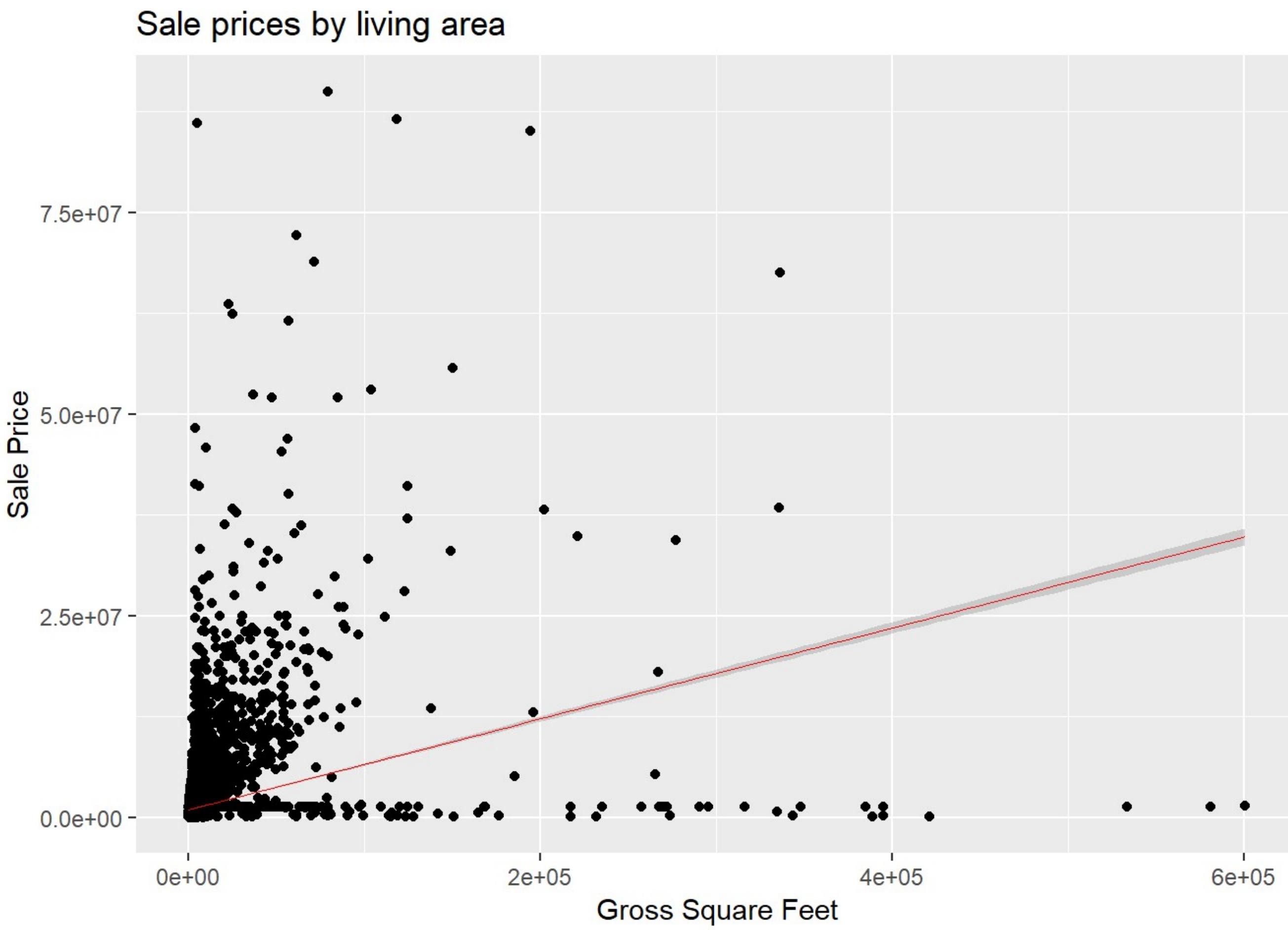
Sale Price also varied by *Total Units* so this was a predictor.

```
livingspaces %>%
  ggplot(aes(`LAND SQUARE FEET`,`SALE PRICE`)) +
  geom_point() +
  geom_smooth(color = "red", size = 0.1, method = lm) +
  xlab("Land Square Feet") +
  ylab("Sale Price") +
  ggtitle("Sale prices by area of lot")
```



Sale Price varied according to *Land Square Feet*, which makes sense if the land value is determining the overall price.

```
livingspaces %>%
  ggplot(aes(`GROSS SQUARE FEET`,`SALE PRICE`)) +
  geom_point() +
  geom_smooth(color = "red", size = 0.1, method = lm) +
  xlab("Gross Square Feet") +
  ylab("Sale Price") +
  ggtitle("Sale prices by living area")
```



Similarly, the *Sale Price* varied with *Gross Square Feet*, indicating that one can sell a building for a higher price if it has more living space in it. Outliers were still observed in all of the above graphs and perhaps more data could be excluded. But given the loss of half the dataset already it was decided to proceed with the remaining despite potential risks.

Data partitioning

```
livingspaces <- livingspaces[, !(colnames(livingspaces) %in% c("X1", "BLOCK", "NEIGHBO
RHOOD", "BUILDING CLASS CATEGORY",
                                                                "TAX CLASS AT PRESENT",
"LOT", "EASE-MENT", "BUILDING CLASS AT PRESENT",
                                                                "ADDRESS", "APARTMENT N
UMBER", "ZIP CODE", "RESIDENTIAL UNITS",
                                                                "COMMERCIAL UNITS", "TA
X CLASS AT TIME OF SALE", "YEAR BUILT",
                                                                "BUILDING CLASS AT TIME
OF SALE", "SALE DATE"))]
```

Unnecessary columns were removed and the final model consisted of 6 variables. The outcome was *Sale Price* and the 5 predictors were *Borough*, *Total Units*, *Land Square Feet*, *Gross Square Feet* and *BuildingAge*.

Data were partitioned as follows:

```
set.seed(1)
test_index <- createDataPartition(y = livingspaces$`SALE PRICE`, times = 1, p = 0.1, l
ist = FALSE)
livingtrain <- livingspaces[-test_index,]
livingtest <- livingspaces[test_index,]
rm(test_index)
```

This produced a test set (livingtest) that was 10% of the total livingspaces dataset.

Modeling and Results

To predict the continuous variable *Sale Price* a regression approach was used. When using this errors should be reported using the Root Mean Squared Error, or RMSE. This function was defined according to the following code:

```
RMSE <- function(true_price, predicted_price){
  sqrt(mean((true_price - predicted_price)^2))
}
```

As discussed previously, these data should have been log transformed and doing so would have produced the type of RMSE more usual in machine learning courses. However, earlier testing in this project found that log transformation meant that only linear regression gave any valid results. Since the point of this capstone was to also move beyond linear models, it was decided to use *Sale Prices* in their raw form and note the change in RMSE over models, not the actual RMSE value itself.

Model 1

```
mu_hat <- mean(livingtrain$`SALE PRICE`)
meanRMSE <- RMSE(livingtest$`SALE PRICE`, mu_hat)

rmse_results <- data_frame(Model = "Mean sale price", RMSE = meanRMSE)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
rmse_results
```

```
## # A tibble: 1 x 2
##   Model                RMSE
##   <chr>               <dbl>
## 1 Mean sale price 2220327.
```

The first model simply predicts that the *Sale Price* would be the mean sale price for the group. It also creates the rmse_results summary data frame to present progress. The final table will be presented at the end.

Model 2

```
lmGross <- train(`SALE PRICE` ~ `GROSS SQUARE FEET`, data=livingtrain, method = "lm")
lmGross
```

```
## Linear Regression
##
## 31076 samples
##     1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results:
##
##   RMSE     Rsquared    MAE
##   2337111  0.09265058  678161.3
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
predictedmodel2 <- predict(lmGross, livingtest)
lmGrossRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel2)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Linreg Gross Area",
                                     RMSE = lmGrossRMSE ))
```

The second model was a linear regression including only *Gross Square Feet* as a predictor. This variable was most correlated with *Sale Price* and reduced RMSE significantly compared to Model 1.

Model 3

```
lmAll <- train(`SALE PRICE` ~ ., data=livingtrain, method = "lm")
lmAll
```

```
## Linear Regression
##
## 31076 samples
##     5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   2195851  0.1372467  716439.1
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
predictedmodel3 <- predict(lmAll, livingtest)
lmAllRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel3)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Linreg All Predictors",
                                     RMSE = lmAllRMSE ))
```

The third model used linear regression and all of the predictors. This reduced the RMSE value again.

Model 4

```
lmnoBorough <- train(`SALE PRICE` ~ BuildingAge + `TOTAL UNITS` + `LAND SQUARE FEET` +
`GROSS SQUARE FEET`, data=livingtrain, method = "lm")
lmnoBorough
```

```
## Linear Regression
##
## 31076 samples
##     4 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   2270276  0.1098717  676356.3
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
predictedmodel4 <- predict(lmnoBorough, livingtest)
lmnoBoroughRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel4)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Linreg without Borough",
                                     RMSE = lmnoBoroughRMSE ))
```

Since the *Borough* data was so skewed in that Manhattan prices were much higher than the other 4 boroughs, a linear regression without this predictor was made in Model 4. However, this increased the RMSE.

Model 5

```
fitLasso <- train(`SALE PRICE` ~ ., data=livingtrain, method = "lasso",
                  tuneGrid = data.frame(fraction = seq(0.1, 1, 0.1)))
fitLasso
```

```
## The lasso
##
## 31076 samples
##     5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results across tuning parameters:
##
##   fraction  RMSE     Rsquared   MAE
##   0.1       2337304  0.1194287  691797.5
##   0.2       2301298  0.1217045  685575.5
##   0.3       2275162  0.1294420  682668.6
##   0.4       2254585  0.1348499  684093.2
##   0.5       2240451  0.1366571  689356.6
##   0.6       2232588  0.1366287  696193.2
##   0.7       2229768  0.1362709  703332.8
##   0.8       2230188  0.1360708  709782.2
##   0.9       2231343  0.1358146  714900.3
##   1.0       2233387  0.1354633  718938.3
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was fraction = 0.7.
```

```
predictedmodel5 <- predict(fitLasso, livingtest)
lassoRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel5)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Lasso regression",
                                     RMSE = lassoRMSE ))
```

The next model tried Lasso regression as a was to predict the outcome *Sale Price*. The model was tuned according to *fraction*.

Model 6

```
fitPCR <- train(`SALE PRICE` ~ ., data=livingtrain, method = "pcr",
                tuneGrid = data.frame(ncomp = seq(1, 5, 0.5)))
fitPCR
```

```
## Principal Component Analysis
##
## 31076 samples
##     5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results across tuning parameters:
##
##   ncomp  RMSE     Rsquared    MAE
##   1.0    2267271  0.08443628  671500.2
##   1.5    2267271  0.08443628  671500.2
##   2.0    2263584  0.08915539  671655.7
##   2.5    2263584  0.08915539  671655.7
##   3.0    2259706  0.09163704  672125.2
##   3.5    2259706  0.09163704  672125.2
##   4.0    2222119  0.11801661  671612.6
##   4.5    2222119  0.11801661  671612.6
##   5.0    2202391  0.13341357  713806.9
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was ncomp = 5.
```

```
predictedmodel6 <- predict(fitPCR, livingtest)
PCRRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel6)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Principal component analysis",
                                     RMSE = PCRRMSE ))
```

Principal component analysis was tested next, and tuned according to *ncomp*.

Model 7

```
fitEnet <- train(`SALE PRICE` ~ ., data=livingtrain, method = "enet")
fitEnet
```

```
## Elasticnet
##
## 31076 samples
##     5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results across tuning parameters:
##
##   lambda  fraction  RMSE     Rsquared   MAE
##   0e+00   0.050     2356108  0.1119415  694668.5
##   0e+00   0.525     2236487  0.1293714  684657.5
##   0e+00   1.000     2229600  0.1318997  712843.3
##   1e-04   0.050     2356117  0.1119426  694670.3
##   1e-04   0.525     2236497  0.1293661  684642.8
##   1e-04   1.000     2229598  0.1319020  712843.1
##   1e-01   0.050     2358488  0.1124835  695162.1
##   1e-01   0.525     2245057  0.1267937  680761.3
##   1e-01   1.000     2233282  0.1308203  714489.8
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were fraction = 1 and lambda = 1e-04.
```

```
predictedmodel7 <- predict(fitEnet, livingtest)
EnetRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel7)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Elasticnet regression",
                                     RMSE = EnetRMSE ))
```

Model 7 was an Elasticnet regression. Default tune parameters gave the best RMSE.

Model 8

```
fitRidge <- train(`SALE PRICE` ~ ., data=livingtrain, method = "ridge",
                  tuneGrid = data.frame(lambda = seq(0, 0.1, 0.01)))
fitRidge
```

```
## Ridge Regression
##
## 31076 samples
##     5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results across tuning parameters:
##
##   lambda  RMSE     Rsquared   MAE
##   0.00    2214198  0.1294590  714101.1
##   0.01    2214029  0.1293739  714212.9
##   0.02    2214093  0.1292289  714358.5
##   0.03    2214298  0.1290562  714517.9
##   0.04    2214592  0.1288724  714687.6
##   0.05    2214944  0.1286862  714865.4
##   0.06    2215337  0.1285020  715050.2
##   0.07    2215757  0.1283223  715239.6
##   0.08    2216198  0.1281481  715434.5
##   0.09    2216655  0.1279801  715633.3
##   0.10    2217123  0.1278182  715834.5
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was lambda = 0.01.
```

```
predictedmodel8 <- predict(fitRidge, livingtest)
RidgeRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel8)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="Ridge regression",
                                     RMSE = RidgeRMSE ))
```

Ridge regression was tuned according to *lambda* values.

Model 9

```
fitICR <- train(`SALE PRICE` ~ ., data=livingtrain, method = "icr",
                tuneGrid = data.frame(n.comp = seq(1, 5, 0.5)))
fitICR
```

```
## Independent Component Regression
##
## 31076 samples
##     5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 31076, 31076, 31076, 31076, 31076, 31076, ...
## Resampling results across tuning parameters:
##
##   n.comp  RMSE     Rsquared    MAE
##   1.0     2287342  0.09959456  672222.7
##   1.5     2287342  0.09959456  672222.7
##   2.0     2265351  0.11423610  714748.8
##   2.5     2265351  0.11423610  714748.8
##   3.0     2244793  0.13137931  721850.7
##   3.5     2244793  0.13137931  721850.7
##   4.0     2243951  0.13318278  715591.3
##   4.5     2243951  0.13318278  715591.3
##   5.0     2237591  0.13894100  714515.6
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was n.comp = 5.
```

```
predictedmodel9 <- predict(fitICR, livingtest)
ICRRMSE <- RMSE(livingtest$`SALE PRICE`, predictedmodel9)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model="ICR regression",
                                     RMSE = ICRRMSE ))
```

The Independent Component Regression was tuned using *n.comp*.

Model 10

The final model was used was a Bayesian ridge regression. The code for this can be seen in the R file. It was excluded here for visual purposes due to the long running time and its output of t and m values. RMSE value for this was 1956773.

Final comparison of RMSE

The final list of RMSEs sorted by increasing value can be seen here.

```
rmse_results %>%
  arrange(RMSE) %>%
  knitr::kable()
```

Model	RMSE
Ridge regression	1956204

Model	RMSE
Lasso regression	1957978
Elasticnet regression	1958617
Linreg All Predictors	1958646
Principal component analysis	1958646
ICR regression	1958646
Linreg without Borough	1977331
Linreg Gross Area	1990666
Mean sale price	2220327

Note again that Model 10 Bayesian ridge regression is not in this list as running the code would have cluttered this report. RMSE for this was 1956773.

Conclusion

With this approach it was possible to predict the sale price of living spaces in New York City using 5 predictor variables available in the Kaggle dataset. However, the considerable amount of missing or nonsensical data meant that over half of the dataset was excluded from the start. The remaining data was skewed further which resulted in high RMSE values. Nonetheless, **ridge regression** provided the best model for these data according to RMSE. It would be interesting to validate this using housing price datasets from other areas which are larger and cleaner in terms of realistic values.

Excluding data for Manhattan may have improved these numbers given that they were outliers. However, this also illustrates that property prices for the most desirable areas of large cities is no longer determined by structural properties like size or land area. These may reflect proximity to transport or major landmarks or sites of business (Debrezion et al., 2007). It could also reflect the subjective or psychological component of prices in that people are willing to pay more for the most desirable locations (Smith, 2011). In conclusion, home prices for boroughs outside of Manhattan may still be predictable according to lot size and area. But prices in Manhattan seem to be much more complicated and more difficult to predict with this type of dataset and regression approach.

References

Debrezion, G., Pels, E. Rietvald, P. (2007), The impact of railway stations on residential and commercial property value, Journal of Real Estate Finance and Economics, 35:161-180.

Florida, R. (2002). The Rise of the Creative Class. New York: Basic Books.

Florida, R. (2017). The New Urban Crisis. New York: Basic Books.

Glaeser, E. (2011). Triumph of the city: How our greatest invention makes us richer, smarter, greener, healthier and happier. Pan Macmillan.

Smith, S. J. (2011) Home Price Dynamics: a Behavioural Economy? Housing, Theory and Society, 28:3, 236-261.

Code and dataset are included in my GitHub at https://github.com/nordicbychris
/NYCPropertyMLEdxProject20190613.git (https://github.com/nordicbychris
/NYCPropertyMLEdxProject20190613.git). This project is for the Capstone course of the HarvardX Data Science professional certification program.