NYC Real Estate Predictions

Christina Shao

09/07/2021

Introduction

The NYC Rolling Sales Data Set

The NYC Rolling Sales data set that was examined contains a record of every property sold in New York City over a year derived from the New York City Department of Finance's records and cleaned by Kaggle user Aleksey Bilogur. The link to Bilogur's data on Kaggle can be found here.

Overall, this data set—which will be referred to as the sales dataset from here on out—contains records of 84548 property sales in New York. Below is a sample of five sales from the data set.

borough	neighbourhood	zipCode	landSqft	grossSqft	yearBuilt	saleBuildingClass	salePrice
3	MIDWOOD	11229	2760	2480	1920	C0	1500000
3	BAY RIDGE	11209	0	0	1932	D4	735000
1	UPPER EAST SIDE (79-96)	10028	NA	NA	1920	R1	3300000
1	CHELSEA	10011	3950	7900	2013	K1	10750000
4	SOUTH JAMAICA	11433	3000	1892	2006	B2	400000

Each sale record contains the following relevant variables:

- Borough: A numerical code. Manhattan (1), Bronx (2), Brooklyn (3), Queens (4), and Staten Island (5).
- Neighbourhood: A character string with the name of the neighbourhood the property is located in.
- Building Class At Sale: Defines the usage of the property with a letter and number code.
- ZIP Code: A numerical code identical to the zip code of the property.
- Land Square Feet: A numerical value equal to the land area of the property listed in square feet.
- Gross Square Feet: A numerical value equal to the total area of all the floors of a building/building unit.
- Year Built: A numerical value denoting the year the building containing the property was built.

Each record also includes other information that will not be included in this report for the sake of brevity but information regarding that data can be found here.

Real estate is unique in being both an incredibly lucrative market and essential investment for most people. For some, it will be the largest investment of their lives; for others, working the market may be their main source of income. As with any market, introducing some level of predictability is in the best interest of investors and exactly where the main goal of this project is derived from.

> This project aims to use the data provided in the NYC-Rolling-Sales data set to work towards building a machine learning algorithm that can accurately predict the sale price of a property

given its location, size, usage, and age.

Key Steps

Building this algorithm will require the following tasks:

1. Exploring and formatting the sales data set to identify key predictors
2. Determining the individual and cumulative effects those key predictors have on the sale price via linear regression

- This will involve partitioning the data and performing cross validation.

3. Performing regularization on the combined model to eliminate large effects created by a small number of data points
4. Testing the model on the sales data set

Methods and Analysis

Exploring and formatting the sales data set to identify key predictors

The following predictors were determined to be useful pieces of information to train the sales algorithm on: borough, neighbourhood, zip code, land sqft, gross sqft, and year built.

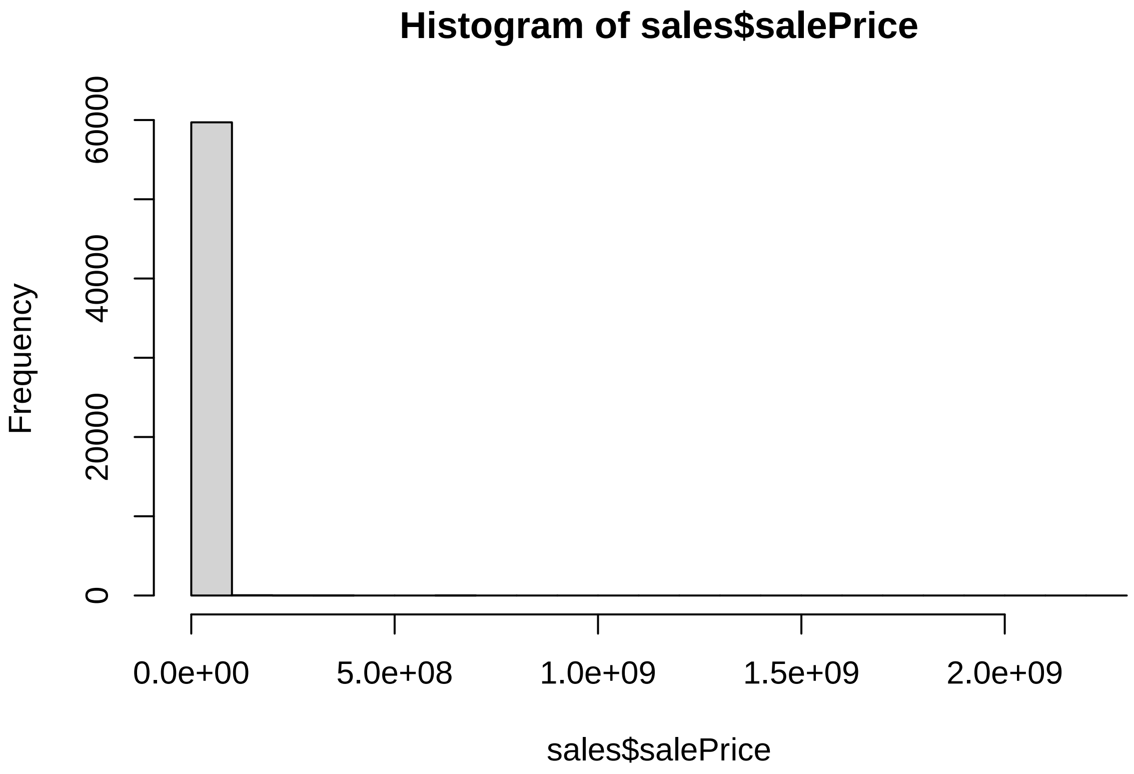
Data Cleaning Before any analysis can be done, the data in these predictors should be converted to appropriate data forms. For example, borough codes and zip codes are continuous numerical values by default but in reality are classification systems that moreso resemble factors. The data contained in the sqft columns and sale price columns are also stored as character strings and should be converted to continuous numerical values. These conversions can be done with the following code:

```r
# convert salePrice to numeric
sales$salePrice <- as.numeric(sales$salePrice)
# convert borough to factor
sales$borough <- as.factor(sales$borough)
# convert zip code to factor
sales$zipCode <- as.factor(sales$zipCode)
# convert landSqft to numeric
sales$landSqft <- as.numeric(sales$landSqft)
# convert grossSqft to numeric
sales$grossSqft <- as.numeric(sales$grossSqft)
```

The data will also have to be cleaned. As the model aims to predict sale price, all observations with a sale price of NA or 0—which represent private transfers of property without the exchange of funds and are therefore irrelevant to market trends—must be removed with the following code:

```r
# remove observations with no salePrice data
sales <- sales[!is.na(sales$salePrice), ]
# remove observations with salePrice of 0
sales <- sales %>% filter(salePrice != 0)
```

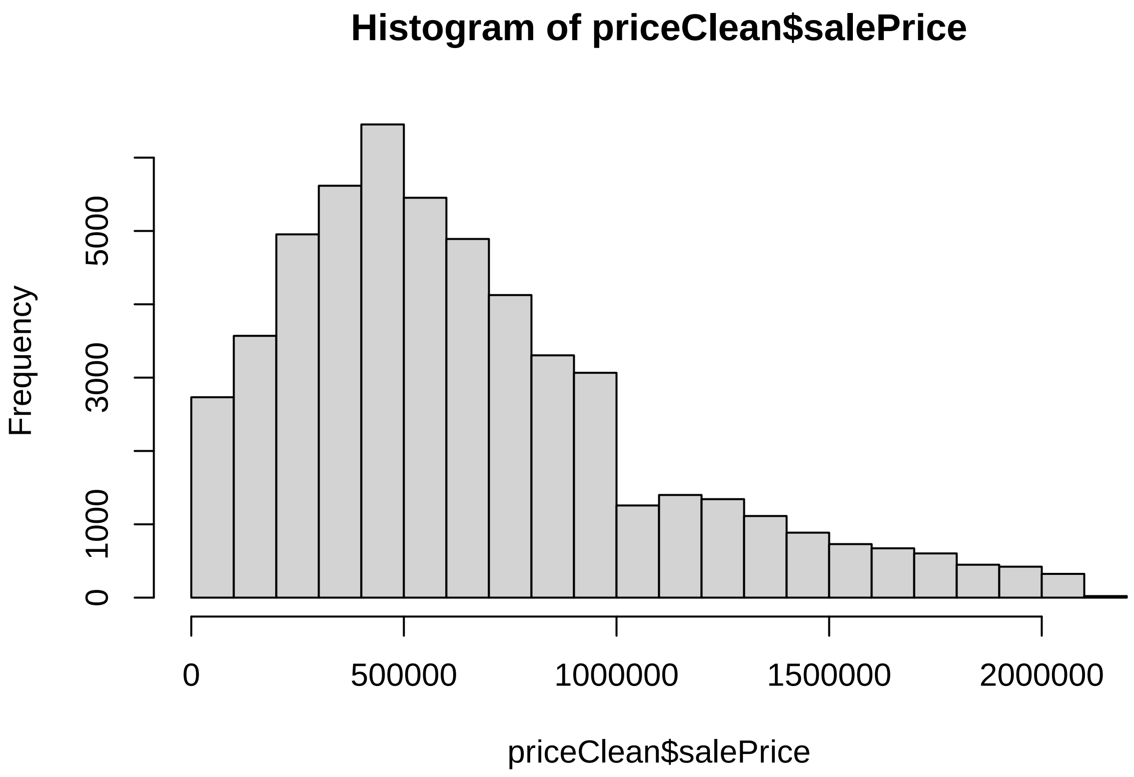
Sale Prices Taking a look into the distribution of sale prices in the sales data set returns this very uninformative histogram:



There seems to be a small number of extremely expensive properties that is making it more difficult to examine the distribution. The outliers (very expensive and very cheap) can be removed by only observing data points that fall inside the interquartile space of the data. A function to remove outliers can be defined as follows:

```r
removeOutliers <- function (i, df){
  Q <- quantile(df[i], probs=c(.25, .75), na.rm = TRUE)
  # find difference between 75th and 25th quartiles
  iqr <- IQR(unlist(df[i]), na.rm = TRUE)
  # find upper and lower range of interquartile space
  up <-  Q[2]+1.5*iqr # Upper Range
  low<- Q[1]-1.5*iqr # Lower Range
  # remove outliers, return cleaned data
  subset(df, df[i] > low & df[i] < up)
}
```

With the outliers removed, the distribution of sale prices now looks like this:

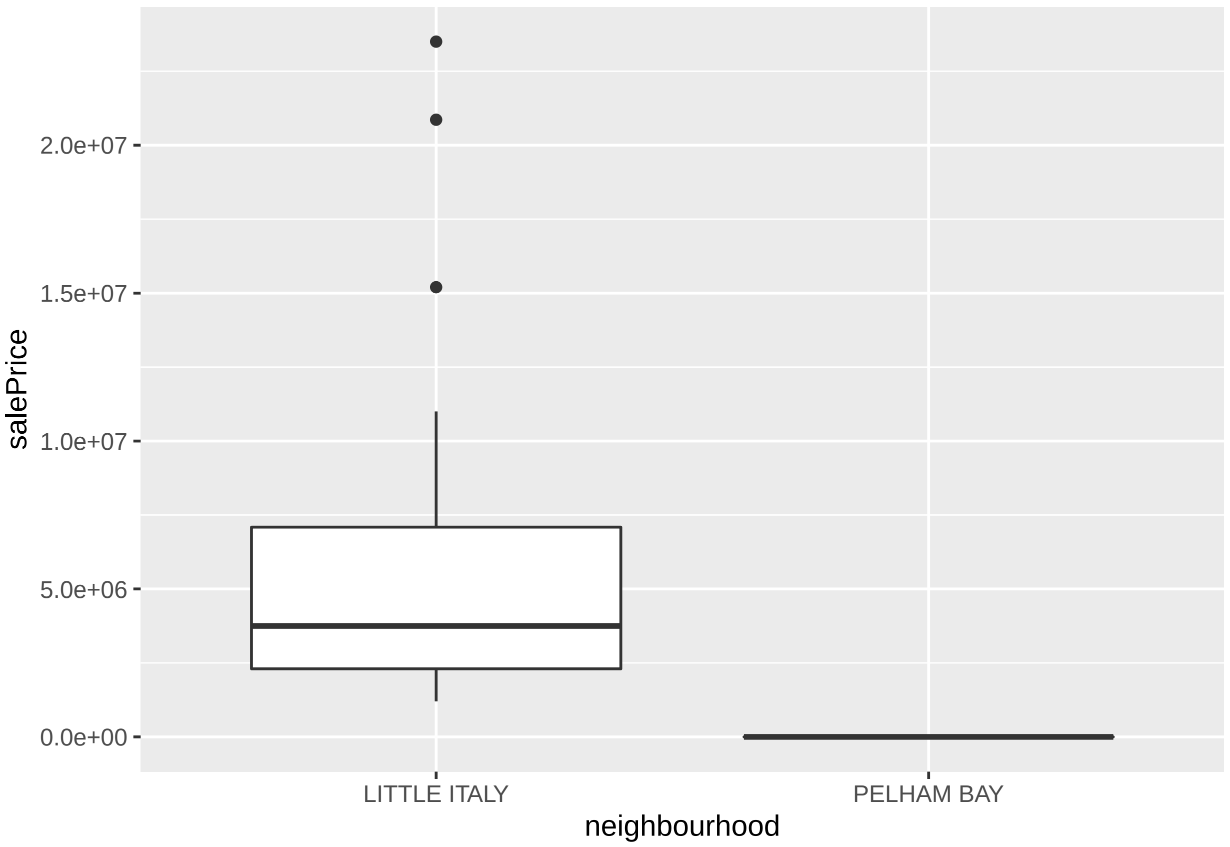


This is much more interpretable. It seems like below a price point of $1,000,000 , the sale price is approximately normally distributed around $50,000. Clearly, this model will have to take outliers and their effects on its predictions into account.

Borough It seems that some boroughs are much more expensive than others, with boroughs 1 and 3 (Manhattan and Brooklyn) selling over $1,000,000 on average. Below are the average sale prices by borough:

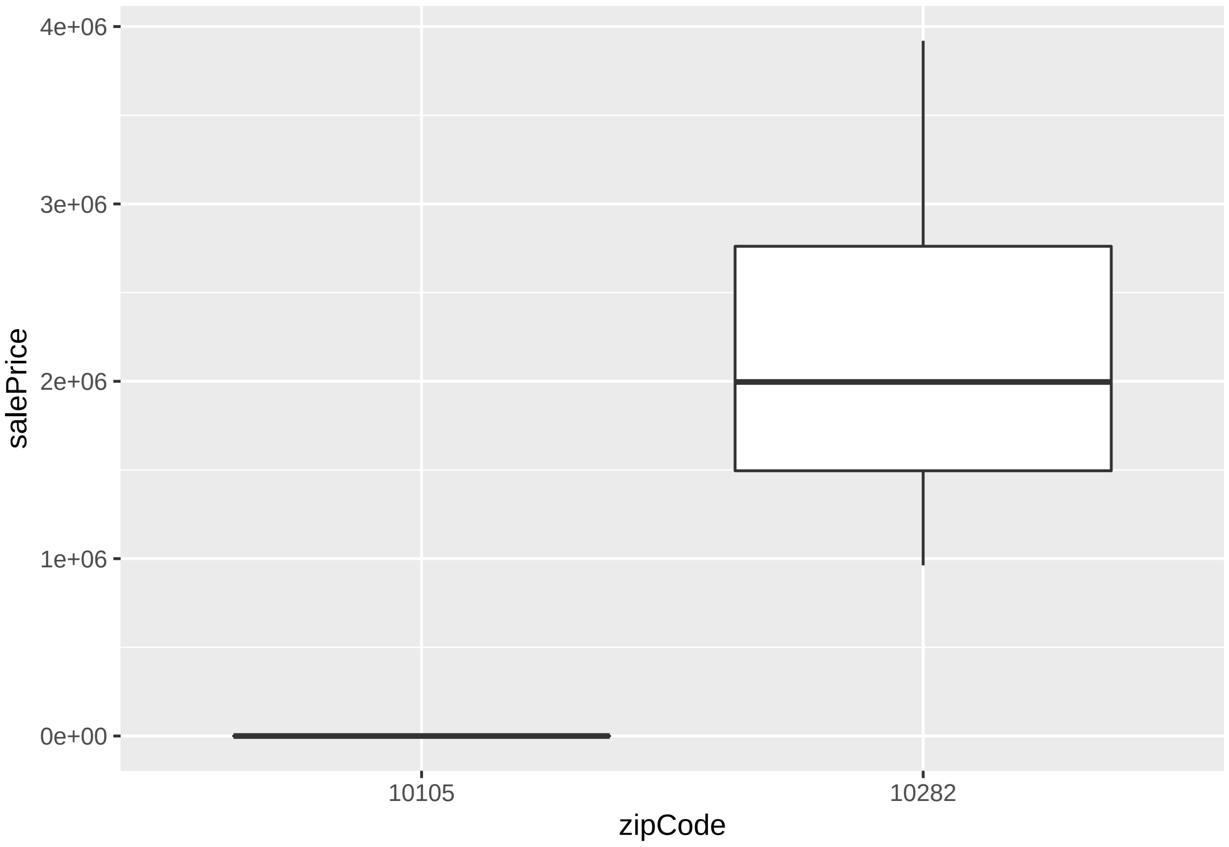
borough	mean
1	3337951.3
2	803452.1
3	1278963.8
4	739908.6
5	543472.1

Neighbourhood Given the sheer amount of neighbourhoods in NYC, it would not be informative to create a box plot of their prices Instead, comparing the most expensive and the cheapest neighbourhoods—Little Italy and Pelham Bay, respectively—yields the following box plots:



Note that the average sale price in Pelham Bay is \$10 despite looking like \$0 on the boxplot.

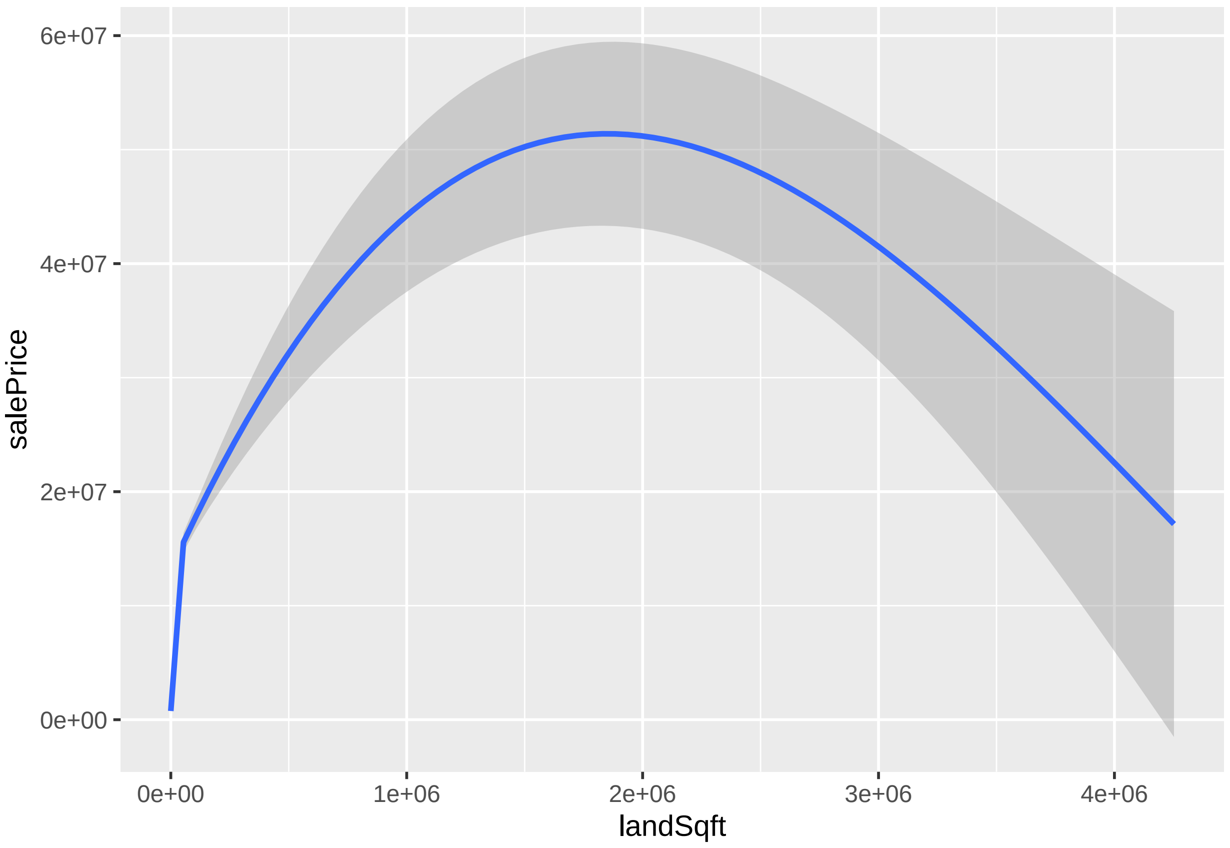
ZIP Code Again, finding the most and least expensive ZIP codes—10282 and 10105, respectively—yields the following:



Note that, again, the average sale price of zip code 10105 is $10 despite looking like $0 on the boxplot.

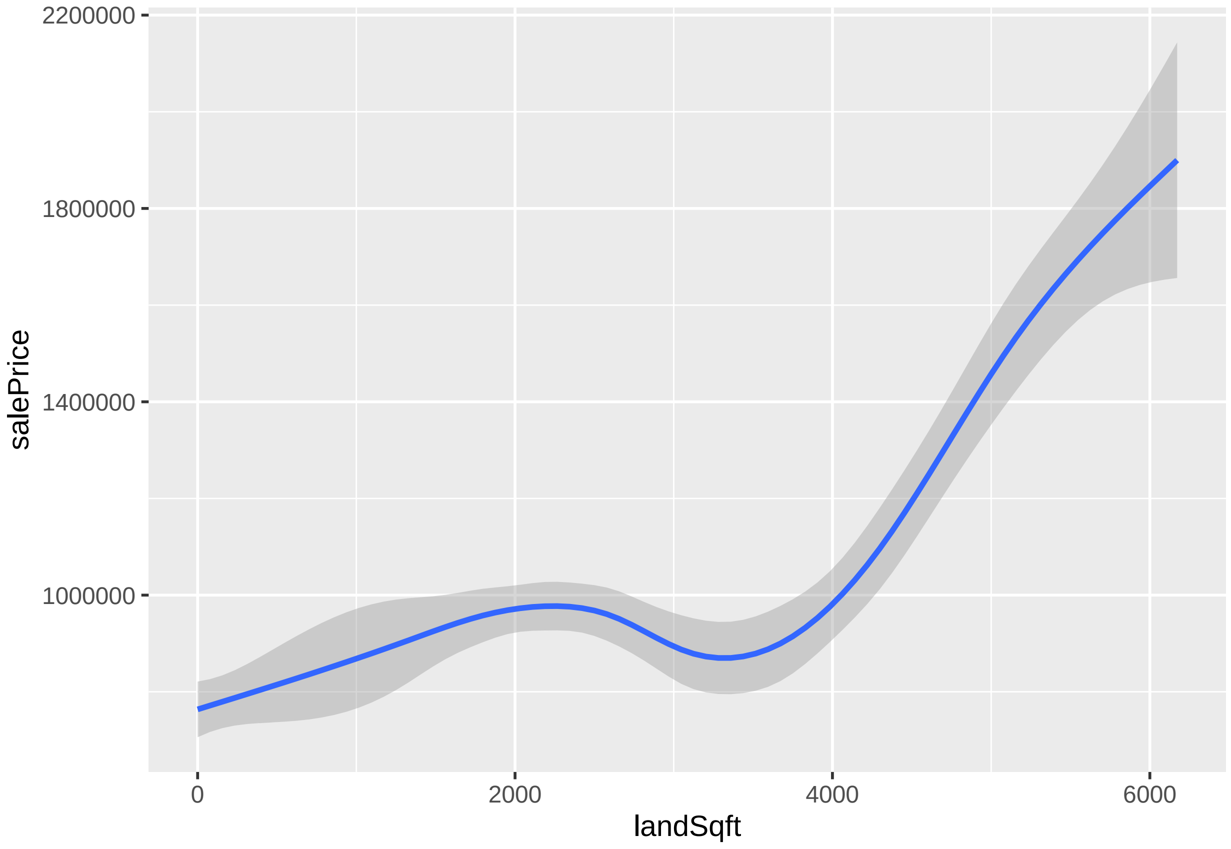
By examining borough, neighbourhood, and zip code, it becomes obvious that location has a very notable effect on the sale price of a property.

Land Square Feet As land area is a continuous value, sales prices in relation to it can be visualized with a smooth line graph:



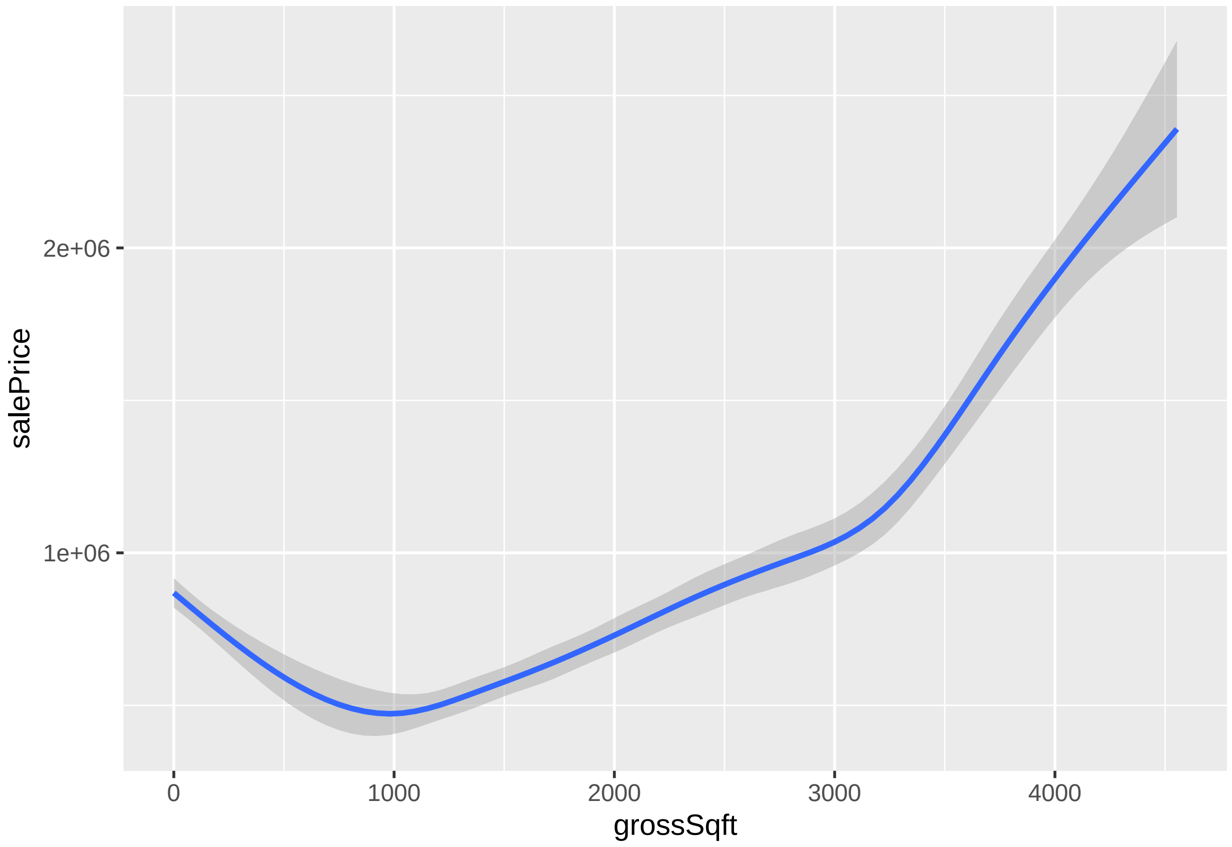
This graph is clearly unintelligible, again due to a small number of properties with massive land area stretching the scale of it.

Removing the outliers provides a much better visualization:



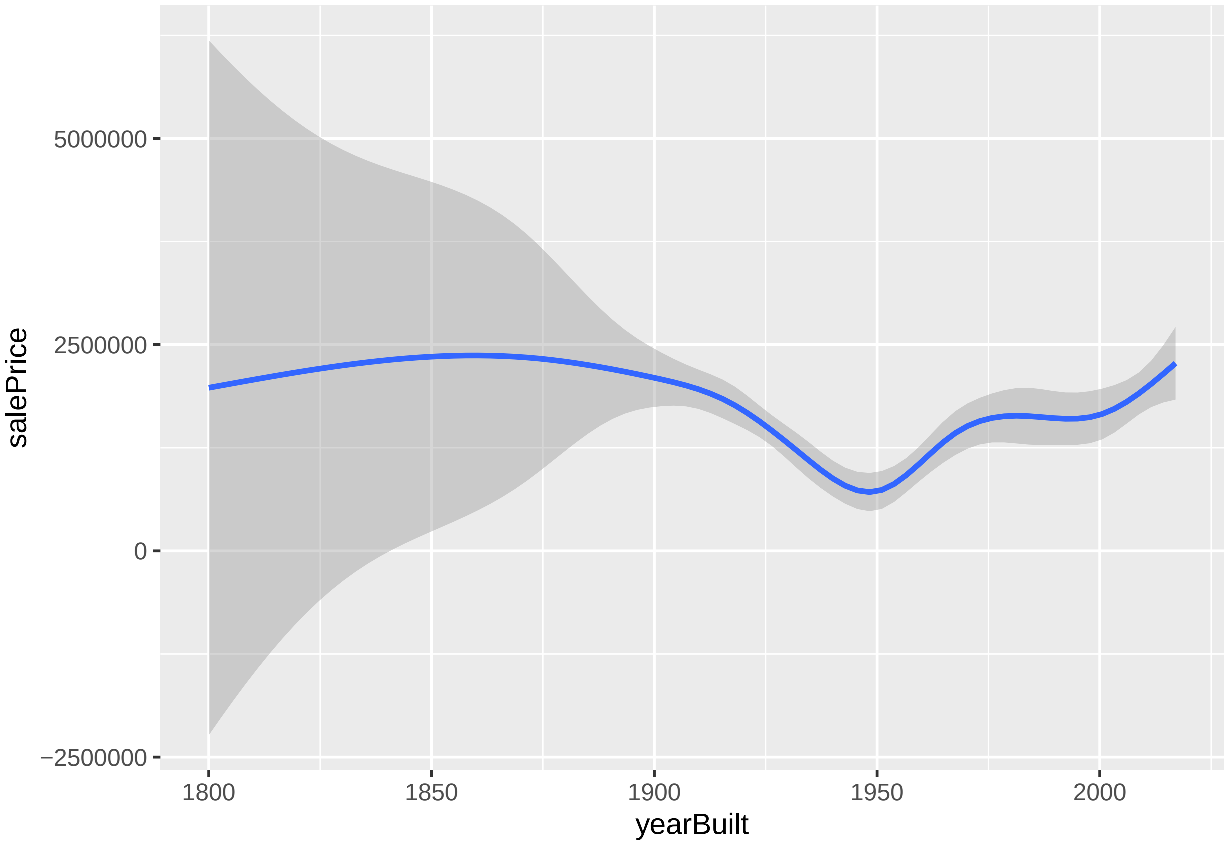
Although a bit bumpy, there is a clear overall positive relationship between land area and sale price.

Gross Square Feet As with land square feet, gross sqaure feet can be visualized with a smooth line graph after removing outliers:



A clearer upwards trend can be seen when using gross square foot as a predictor. This makes sense when considering that most real estate transactions in New York likely involve units in larger multilevel buildings, having less to do with the physical land that building sits on and more to do with the floor space.

Year Built Again, this continuous data can be visualized with a smooth line graph. There is no need to remove outliers here.



The trend here is rougher but it can be vaguely identified—the newer the building, the more expensive with the exception of really old, heritage buildings.

Determining individual and cumulative effects of predictors

The data set must first be partitioned into a set with which to train the model and one with which to datermine its accuracy on. Without much else of a better guideline, the training and testing sets will be split 80-20, respectively, in accordance with the Pareto principle.

```r
# validation set will be 20% of the data set according to Pareto's principle
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = sales$salePrice, times = 1, p = 0.2, list = FALSE)
train <- sales[-test_index,]
validation <- sales[test_index,]
```

As using the validation set to determine which model is the most effective must be avoided, a further testing and training set must also be created by subsetting the train data set with the following function:

```r
createSet <- function (df, n){
  samp <- sample_n(df, n)
  test_index <- createDataPartition(y = samp$salePrice, times = 1, p = 0.2,
                                    list = FALSE)
  train_set <- samp[-test_index,]
  test_set <- samp[test_index,]
  return(list(train_set, test_set))
}

# create test and training set
set <- createSet(train, nrow(train))
train_set <- as.data.frame(set[1])
test_set <- as.data.frame(set[2])
```

With these data sets, the individual predictors can then be used to create models with linear regression.

Borough The effects of the borough each property is located in can be determined with linear regression. It would be easy to simply do so with

```r
lm(salePrice ~ borough, data = train_set)
```

but that would likely crash R given how massive the sales data set and the subsetted test data sets are. As such, an estimation can be made by looking at the equation that is being built for the predictions.

Taking borough bias into account, the equation looks as follows:

$$Y_{\mu,b} = \mu + b_b + \varepsilon_{\mu,b}$$

Where Y = sale price, μ = average sale price, b = bias/effect of the predictor of interest, and ε = error.

Rearranging this equation yields an approximation for b_b, or the borough bias:

$$b_b = Y_{\mu,b} - \mu$$

Disregarding error($\varepsilon_{\mu,i}$), the following code will produce the borough bias (b_b):

```r
borough_avgs <- train_set %>%
  group_by(borough) %>%
  summarize(b_b = mean(salePrice - mu))
# NAs to 0
borough_avgs$b_b[is.na(borough_avgs$b_b)] <- 0
#add borough biases to train_set for future use
train_set <- train_set %>% left_join(borough_avgs, by='borough')
```

As there are some entries for which there is no available borough data, the borough effect will be set to 0 so the model will predict the next best guess—in this case, the overall average sale price.

Neighbourhood This process can be repeated for the neighbourhood bias. The equation for sale price now becomes:

$$Y_{\mu,b,n} = \mu + b_b + b_n + \varepsilon_{\mu,b,n}$$

This makes it so that the user effect (b_n) becomes:

$$b_n = Y_{\mu,b,n} - \mu - b_b$$

This can be applied on the data set with the following code:

```r
neighbourhood_avgs <- train_set %>%
  group_by(neighbourhood) %>%
  summarize(b_n = mean(salePrice - mu - b_b))
# NAs to 0
neighbourhood_avgs$b_n[is.na(neighbourhood_avgs$b_n)] <- 0
#add neighbourhood biases to train_set for future use
train_set <- train_set %>% left_join(neighbourhood_avgs, by='neighbourhood')
```

ZIP Code The process is then repeated with the following equation to include ZIP code:

$$Y_{\mu,b,n,z} = \mu + b_b + b_n + b_z + \varepsilon_{\mu,b,n,z}$$

ZIP code effect (b_z) then becomes:

$$b_z = Y_{\mu,b,n,z} - \mu - b_b - b_n$$

Which is implemented as such:

```r
zip_avgs <- train_set %>%
  group_by(zipCode) %>%
  summarize(b_z = mean(salePrice - mu - b_b - b_n))
# NAs to 0
zip_avgs$b_z[is.na(zip_avgs$b_z)] <- 0
#add zip code biases to train_set for future use
train_set <- train_set %>% left_join(zip_avgs, by='zipCode')
```

Land Square Foot With the next few predictors, the data becomes continuous instead of discrete/catagorical. Regardless, the implementation of regression does not change. To include land sqft, the equation becomes:

$$Y_{\mu,b,n,z,l} = \mu + b_b + b_n + b_z + b_l + \varepsilon_{\mu,b,n,z,l}$$

Land square foot then has an effect of:

$$b_l = Y_{\mu,b,n,z,l} - \mu - b_b - b_n - bz$$

Which is implemented as follows:

```r
land_avgs <- train_set %>%
  group_by(landSqft) %>%
  summarize(b_l = mean(salePrice - mu - b_b - b_n - b_z))
# NAs to 0
land_avgs$b_l[is.na(land_avgs$b_l)] <- 0
#add land sqft biases to train_set for future use
train_set <- train_set %>% left_join(land_avgs, by='landSqft')
```

Gross Square Foot To include gross sqft, the equation becomes:

$$Y_{\mu,b,n,z,l,g} = \mu + b_b + b_n + b_z + b_l + b_g + \varepsilon_{\mu,b,n,z,l,g}$$

Gross square foot effect subsequently becomes:

$$b_g = Y_{\mu,b,n,z,l,g} - \mu - b_b - b_n - bz - b_l$$

Which is implemented with the following code:

```
gross_avgs <- train_set %>%
  group_by(grossSqft) %>%
  summarize(b_g = mean(salePrice - mu - b_b - b_n - b_z - b_l))
# NAs to 0
gross_avgs$b_g[is.na(gross_avgs$b_g)] <- 0
#add gross sqft biases to train_set for future use
train_set <- train_set %>% left_join(gross_avgs, by='grossSqft')
```

Year Built

The equation now becomes:

$$Y_{\mu,b,n,z,l,g,y} = \mu + b_b + b_n + b_z + b_l + b_g + b_y + \varepsilon_{\mu,b,n,z,l,g,y}$$

Year built effect (which is essentially the building's age) becomes:

$$b_y = Y_{\mu,b,n,z,l,g} - \mu - b_b - b_n - bz - b_l - b_g$$

Which is then added to the train set data frame with the following code:

```
year_avgs <- train_set %>%
  group_by(yearBuilt) %>%
  summarize(b_y = mean(salePrice - mu - b_b - b_n - b_z - b_l - b_g))
# NAs to 0
year_avgs$b_y[is.na(year_avgs$b_y)] <- 0
#add year biases to train_set for future use
train_set <- train_set %>% left_join(year_avgs, by='yearBuilt')
```

Regularization At this point, there is already a complete model that has been trained and is ready for implementation. However, as mentioned in the data analysis section, it would seem like there are lots of outliers of extremely expensive, extremely big properties that are vastly skewing the accuracy of the model, especially as Y is being calculated by taking the mean price, not the median. As such, another model that will implement regularization will also be created and tested. This will theoretically reduce the disproportionately large effects those relatively few outliers will have on the overall model.

This model will be very similar to the pure linear regression model. However, now, instead of taking the mean as a value for Y, Y will instead equal:

$$Y = \frac{1}{\lambda + n} \sum_{u=1}^{n} (Y_{b,n,z,l,g,y} - \hat{b_i}^2)$$

Where

$$b_{b,n,z,l,g,y}$$

represents all calculated effects added together.

Implementing the above equation to the model built in previous sections effectively minimizes an equation that includes a penalty for small sample sizes. That equation looks like this:

$$\sum (Y + \mu + b_{b,n,z,l,g,y})^2 + \lambda \sum (b_{b,n,z,l,g,y})$$

The equation $Y = \frac{1}{\lambda+n}\sum_{u=1}^{n}(Y_{b,n,z,l,g,y} - \hat{b_i}^2)$ can be implemented in each predictor's code (seen in previous sections) by simply replacing the mean(salePrice) function with

```
sum(salePrice - mu - b)/(n() + l)
```

where b is the sum of all appropriate effects. To demonstrate, here is how this change would be implemented on the neighbourhood bias code:

```
neighbourhood_avgs <- train_set %>%
    group_by(neighbourhood) %>%
    summarize(b_n = sum(salePrice - mu - b_b)/(n() + l))
```

Where l = λ.

Now, all that must be done is to determine the lambda (λ) value that will minimize the root mean square error of the model. This can be done by encompassing the modified code that has been built so far with the following code:

```
lambdas <- seq(0, 10, 0.25)

regularization <- function(l, train_set, test_set){
  # borough bias code
  # neighbourhood bias code
  # ZIP code bias code
  # land sqft bias code
  # gross sqft bias code
  # year built bias code
}
rmses <- sapply(lambdas, regularization, train_set=train_set, test_set=test_set)
lambda <- lambdas[which.min(rmses)]
```

This code applies the model built so far on a vector of possible lambda values before determining which one results in the lowest root mean square error.

Results

Linear Regression Results

The naive root mean square error of this model was **6161066** without regularization—an utterly abysmal RMSE but to be expected with just how wildly the prices have been show to fluctuate.

Applying borough bias with the equation $Y_{\mu,b} = \mu + b_b + \varepsilon_{\mu,b}$ resulted in an RMSE of **6100210** without regularization—slightly better but still unacceptable.

Applying neighbourhood bias with the equation $Y_{\mu,b,n} = \mu + b_b + b_n + \varepsilon_{\mu,b,n}$ resulted in an RMSE of **6099938** without regularization.

Applying zip code bias with the equation $Y_{\mu,b,n,z} = \mu + b_b + b_n + b_z + \varepsilon_{\mu,b,n,z}$ resulted in an RMSE of **6082665** without regularization.

Applying land sqft bias with the equation $Y_{\mu,b,n,z,l} = \mu + b_b + b_n + b_z + b_l + \varepsilon_{\mu,b,n,z,l}$ resulted in an RMSE of **3762000** without regularization. Note that this is the most significant improvement in accuracy that any of the predictors gave in this model. This makes sense given the relative scarcity of land that plagues New York coupled with its high population density. Land is probably the single most valuable commodity that comes with real estate and one of—if not the—driving factor in determining property value.

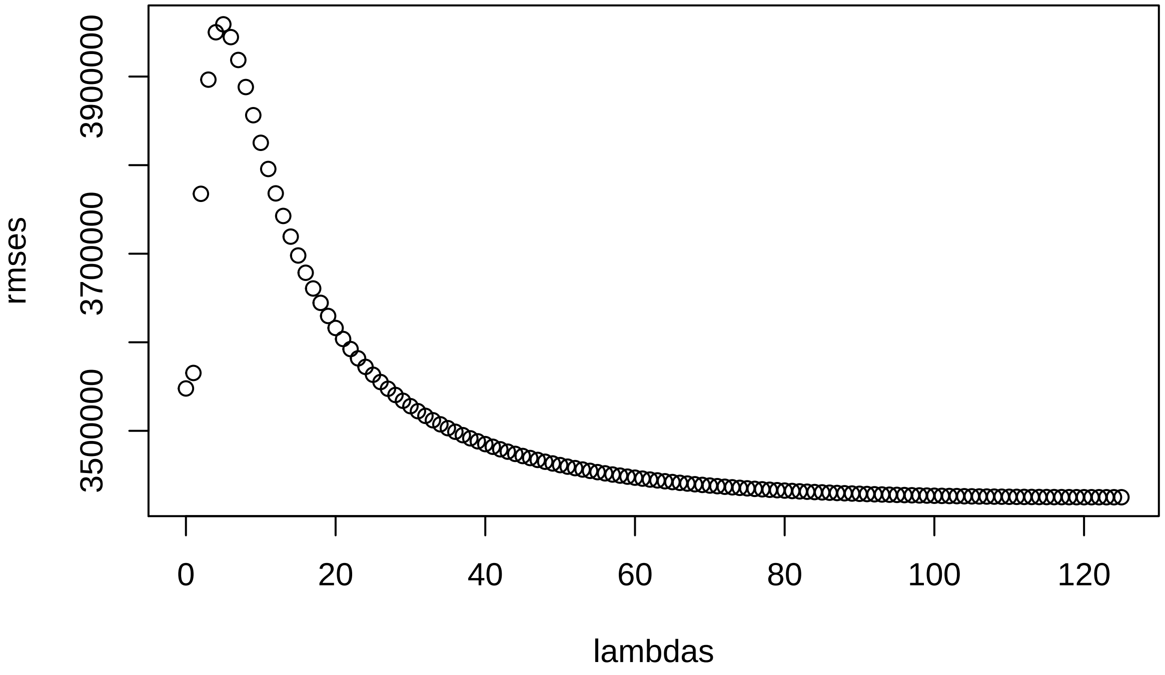
Applying gross sqft bias with the equation $Y_{\mu,b,n,z,l,g} = \mu + b_b + b_n + b_z + b_l + b_g + \varepsilon_{\mu,b,n,z,l,g}$ resulted in an RMSE of **3543360** without regularization.

Applying year built bias with the equation $Y_{\mu,b,n,z,l,g,y} = \mu + b_b + b_n + b_z + b_l + b_g + b_y + \varepsilon_{\mu,b,n,z,l,g,y}$ resulted in an RMSE of **3516079** without regularization.

While the RMSE of 3516079 that the pure regression model provides is a vast improvement over the initial naive RMSE of 6161066, it is still a very low degree of accuracy. As such, the regularized model must be tested to determine how it performs.

Regression Results

Substituting the appropriate code segments as determined above into the code template outlined in the Regularization section of Methods results in an ideal lambda value of **124** via cross validation as seen in the figure below:



Note that, although it may appear that the RMSE continues decreasing as the λ value increases, it actually reaches a global minimum at $\lambda = 124$.

> Applying this lambda value to train the finalized model on the complete train dataset and running it on the validation dataset resulted in an RMSE of **2708405**.

Although this is still not ideal, it is a drastic improvement over the naive RMSE of 6161066 and even over the pure regression accuracy of 3516079.

Conclusion

Through the methods outlined in the above sections, a model was built via pure linear regression using the predictors of movie reviewed, user, release year of the movie, and the combination of genres the movie was classified under. Another model was built that implemented regularization, using a lambda value of 124 to train the model on the train data set before testing on the validation data set. This resulted in an RMSE improvement from 6161066 (naive) to 3516079 (pure regression), to 2708405 (regularized).

Impact/Applications As mentioned in the introduction, real estate serves two vital purposes: a market for capital investment and simply a market for people to purchase a place of residence or business. While the two overlap, they both represent incredibly important and monetarily heavy investments. Successfully delving into the real estate market involves a great deal of timing and can be intimidating for the uninitiated. For both investing professionals and prospective homeowners, an algorithm like the one that has begun to be developed in this report can introduce a degree of predictability and stability in this volatile market. This may (and likely already has) allowed users to make smarter investments with the hundreds of thousands of dollars that go into a piece of property.

Limitations As this project was done with significant time and—more importantly—computing restraints, the final model's accuracy is no where near adequate enough for real world use. I do not have access to a more powerful computer nor do I have access to remote servers and therefore can only run relatively simple code.

In the future, and with a more powerful computer and time to allow complicated code to run, the model could have grown to include the other factors included in the NYC rolling sales dataset like tax class, date of sale, bulding class, address, unit number, number of units in building, etc. to improve accuracy. Given more time, it would have also been valuable to run cross validation on this model, partitioning the train_set and test_set data sets multiple times while training the model to ensure maximum accuracy.

Overall, this model is complicated and the data that is being predicted is dependent on a massive number of variables, many of which are not recorded in the provided data set. For real world use, it would need much more development but this report has presented a start.