Choose My Own Project: Predict Sale Price - NYC Properties

Virginia Carneiro

January 6th, 2021

Contents

1 Introductory

1.1 Overview

This project will focus to develop a machine learning algorithm to predict the *Sale Price* in a NYC property using the tools learning during the 8 courses included in the *Data Science Professional Certificate*, some recommendations I got from my MovieLens Capstone and, also others resources I will including in the Resource section.

I choose this dataset because it will be my starting point in my real state business.

I downloaded the dataset from Kaggle as Harvard-Edx staff has recommended in this chapter.

The dataset will be storage in github and automatically download from my code.

After that, I will analyze the dataset verifying if all the features are necessary and useful. I will plot some of the features for better understanding.

Then, I'm going to split the dataset in 2 sections one for training and another one for validation. Validation dataset will be a 10% of the original dataset and I will use it just in my final model. Train dataset will be split again in training and test. Test dataset will take 10% of the dataset. These datasets will be used to train and test my algorithms and I'll choose the algorithms that fit better according the RMSE metric.

I choose these percentages to split the dataset 80/10/10 because it a small dataset and I don't want to lose too much training data.

Finally, I will show you the different machine learning models with their respective RMSEs.

These models will be the starting point of my analysis, because I would like study in-depth the ensembling models.

Into the NYC Property Project will be three files:

1. My report in PDF format.
2. My report in RMD format.
3. A script in R format that predict the sale price in a property in NYC and the RMSE score.

These files will be storage into the Edx platform and Github (https://github.com/vircarneiro/NYC-Property-Sales)

2 Download Dataset

- The dataset has been downloaded from https://www.kaggle.com/new-york-city/nyc-property-sales
- The dataset has been saved on https://github.com/vircarneiro/NYC-Property-Sales/blob/main/nyc-rolling-sales.zip?raw=true

3 Description Dataset and Features

3.1 Dataset Description

The dataset includes all the building or building unit sold in New York City over 12-months period between 2016 and 2017. These properties are located in Manhattan, Bronx, Brooklyn, Queens, and Staten Island.

The combination of borough, block, and lot forms a unique key for property in New York City.

Many sales occur with a nonsensically small dollar amount. These sales are actually transfers of deeds between parties. For my analysis, I will remove these cases.

This dataset uses the financial definition of a building/building unit, for tax purposes. In case a single entity owns the building in question, a sale covers the value of the entire building. In case a building is owned piecemeal by its residents (a condominium), a sale refers to a single apartment (or group of apartments) owned by some individual.

3.2 NYC Property - Structure

```
str(NYCProperty, strict.width = "wrap")
```

```
## Classes 'data.table' and 'data.frame':   83981 obs. of  22 variables:
## $ Index : int 4 5 6 7 8 9 10 11 12 13 ...
## $ Borough : int 1 1 1 1 1 1 1 1 1 1 ...
## $ Neighborhood : chr "ALPHABET CITY" "ALPHABET CITY" "ALPHABET CITY" "ALPHABET
##    CITY" ...
## $ BuildingClassCategory : chr "07 RENTALS - WALKUP APARTMENTS" "07 RENTALS -
##    WALKUP APARTMENTS" "07 RENTALS - WALKUP APARTMENTS" "07 RENTALS - WALKUP
##    APARTMENTS" ...
## $ TaxClassPresent : chr "2A" "2" "2" "2B" ...
## $ Block : int 392 399 399 402 404 405 406 407 379 387 ...
## $ Lot : int 6 26 39 21 55 16 32 18 34 153 ...
## $ EASE-MENT : logi NA NA NA NA NA NA ...
## $ BuildingClassCategoryPresent: chr "C2" "C7" "C7" "C4" ...
## $ Address : chr "153 AVENUE B" "234 EAST 4TH STREET" "197 EAST 3RD STREET" "154
##    EAST 7TH STREET" ...
## $ Apartment : chr "" "" "" "" ...
## $ ZipCode : int 10009 10009 10009 10009 10009 10009 10009 10009 10009 10009 ...
## $ ResidentionUnit : int 5 28 16 10 6 20 8 44 15 24 ...
## $ CommercialUnit : int 0 3 1 0 0 0 0 2 0 0 ...
## $ TotalUnit : int 5 31 17 10 6 20 8 46 15 24 ...
## $ LandFeet : chr "1633" "4616" "2212" "2272" ...
## $ GrossFeet : chr "6440" "18690" "7803" "6794" ...
## $ YearBuilt : int 1900 1900 1900 1913 1900 1900 1920 1900 1920 1920 ...
## $ TaxClassSale : int 2 2 2 2 2 2 2 2 2 2 ...
## $ BuildingClassSale : chr "C2" "C7" "C7" "C4" ...
## $ SalePrice : chr "6625000" "-" "-" "3936272" ...
## $ SaleDate : chr "2017-07-19 00:00:00" "2016-12-14 00:00:00" "2016-12-09
##    00:00:00" "2016-09-23 00:00:00" ...
## - attr(*, ".internal.selfref")=<externalptr>
```

3.3 NYC Property - Features Description

- Borough: a digit code for the borough the property is located in:
  - Manhattan (1),
  - Bronx (2),
  - Brooklyn (3),
  - Queens (4) and,
  - Staten Island (5).

- Neighborhood: Department of Finance assessors determine the neighborhood name in the course of valuing properties.
- Building Class Category: identify similar properties by broad usage (e.g. One Family Homes)
- Tax Class: every property in the city is assigned to one of four tax classes:
  - 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),
  - 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.
- Block: is a sub-division of the borough on which real properties are located.
- Lot: is a subdivision of a Block and represents the property unique location.
- Easement: is a right, such as a right of way, which allows an entity to make limited use of another's real property.
- Building Classification at Present: is used to describe a property's constructive use.
  - 1st position, describe a general class of properties (for example "A" signifies one-family homes)
  - 2nd position, , adds more specific information about the property's use or construction style (using our previous examples "A0" is a Cape Cod style one family home). There is more detail about Building Classification in the reference section.
- Address: the street address of the property as listed on the Sales File.
- Zip Code: the property's postal code.
- Residential Units: number of residential units at the listed property.
- Commercial Units: number of commercial units at the listed property.
- Total Units: total number of units at the listed property.
- Land Square Feet: land area of the property listed in square feet.
- Gross Square Feet: 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.
- Building Class at Time of Sale: is used to describe a property's constructive use at the time of the saling.
- Sales Price: price paid for the property. A USD 0 sale indicates that there was a transfer of ownership without a cash consideration. It can be for example a transfer of ownership from parents to children.
- Sale Date: date the property sold.

4 Data Validation and Cleaning - NYCProperty dataset

4.1 Overview Validation

Verify duplicated observations

```
NYCProperty <- unique(NYCProperty)
```

Verify columns with all the values filled as NAs.

```
colnames(NYCProperty)[colSums(is.na(NYCProperty)) > 0]
```

```
## [1] "EASE-MENT"
```

Erase column EASY_MENT.

```
NYCProperty <- NYCProperty[,-c("EASE-MENT")]
```

Erase column Index. Not useful for analysis.

```
NYCProperty <- NYCProperty[,-c("Index")]
```

4.2 Features

4.2.1 SalePrice (target feature)

The feature to predict has values not acceptable such us "-". I'm going to delete them.

```
##       SalePrice
##    1:   6625000
##    2:         -
##    3:   3936272
##    4:   8000000
##    5:   3192840
##   ---
## 9978:    576800
## 9979:    525427
## 9980:    386820
## 9981:    814498
## 9982:    255088
```

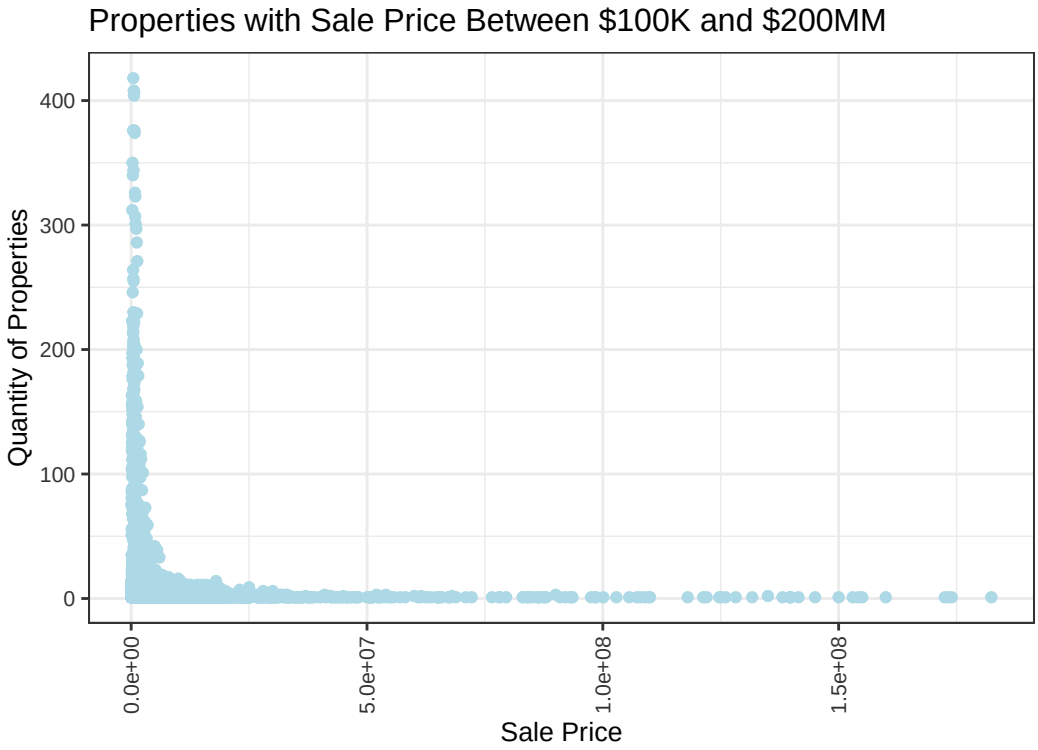
The SalePrice should be a numeric feature not a chr.

I'm going to filter all properties with a sale price <= USD 100.000. Properties below this price are not valid for my analysis.

Plotting the outliers

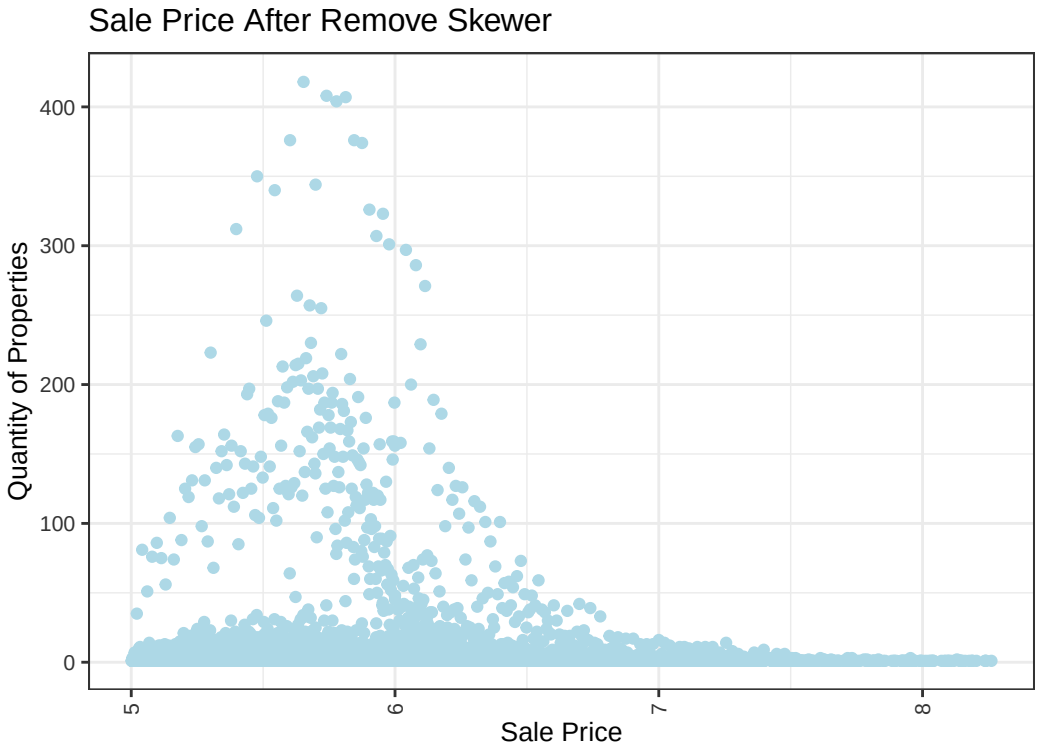
The boxplot shows us the outliers values when the SalePrice is > USD 200.000.000 aprox.. Then I'm going to remove these values.

We can see the distribution becomes skewed to the left with a long tail on the right.



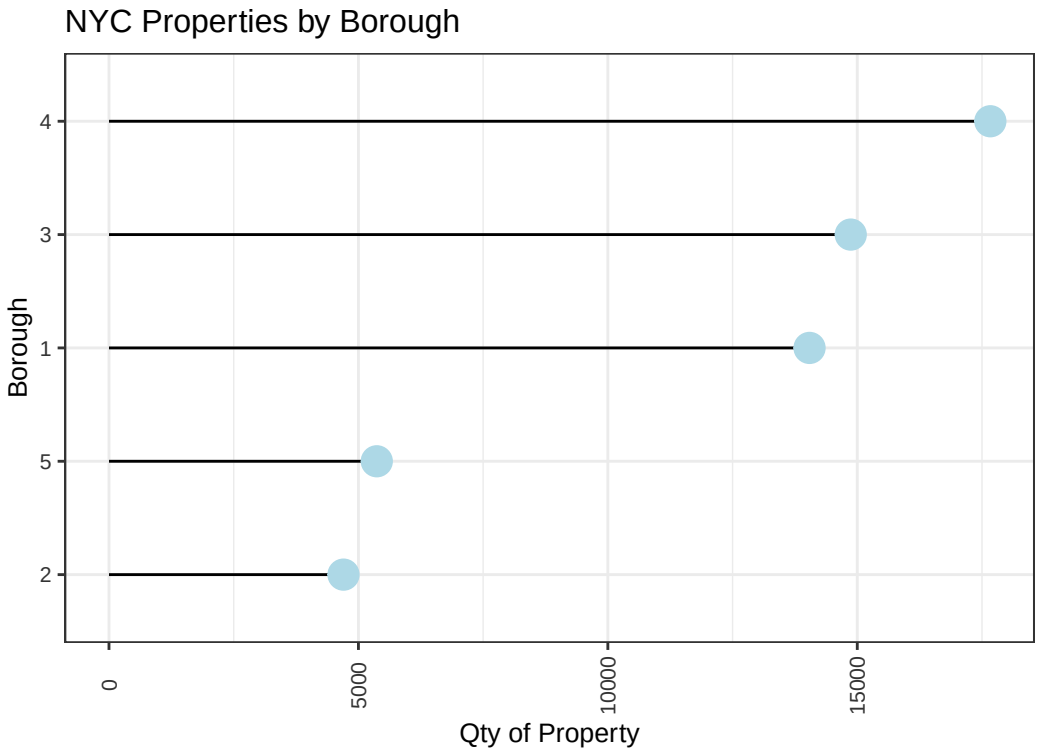
To reduce the skew I'm going to apply the logarithm.

This is the plot after apply the logarithm.



4.2.2 Borough

There are no incorrect values.



In this plot we can see that most of the properties sold are in the area of Queens (4), Brooklyn (3) and, Manhattan (1).

4.2.3 Neighborhood

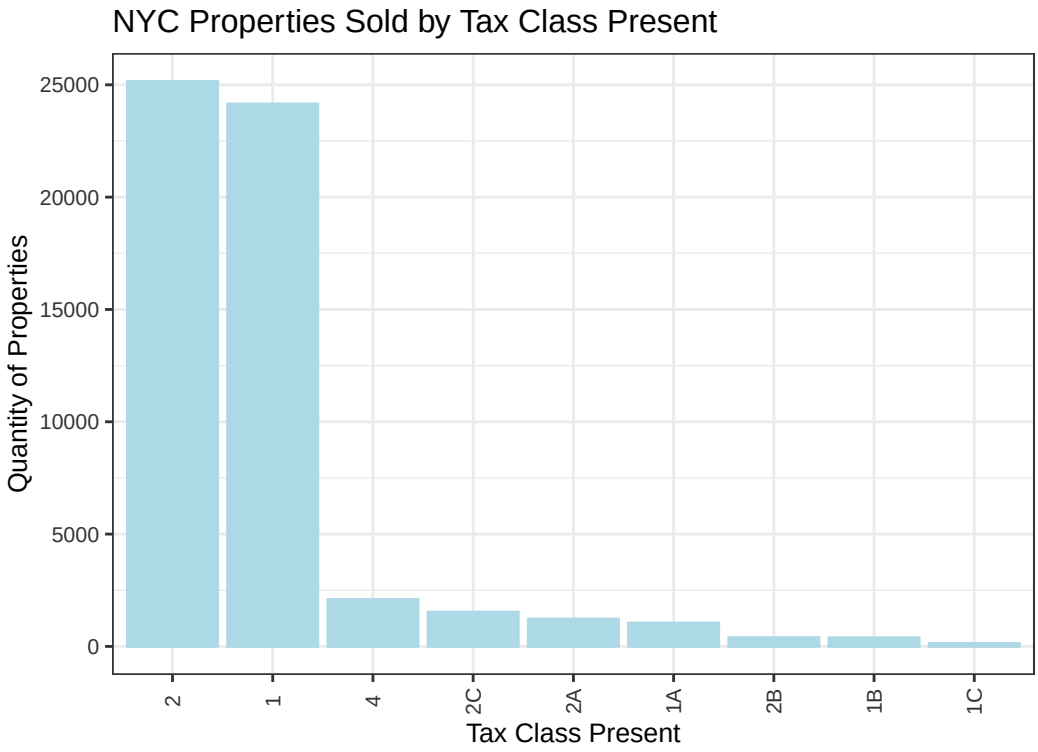
There are no incorrect values. It has to be numeric and categorical.

4.2.4 BuildingClassCategory

There are no incorrect values. It has to be numeric and categorical.

4.2.5 TaxClassPresent

There are null values to remove. It has to be numeric and categorical.



In this plot we can see the majorities of the properties sold have a Tax Class 2 (cooperatives and condominiums) and 1 (residential properties).

4.2.6 Block

There are no incorrect values.

4.2.7 Lot

There are no incorrect values.

4.2.8 BuildingClassCategoryPresent

It has to numeric and categorical.

4.2.9 Address

This feature is not important for my analysis. It contains a lot of detail that does not add value to my analysis.

Before making the decision to delete it, I normalized it by giving all the observations the same format. It didn't add any value to my analysis and my models, just add a lot of code making confusion on it.

4.2.10 Apartment

This feature has most of values invalid and does not add value to my analysis. It will be deleted.

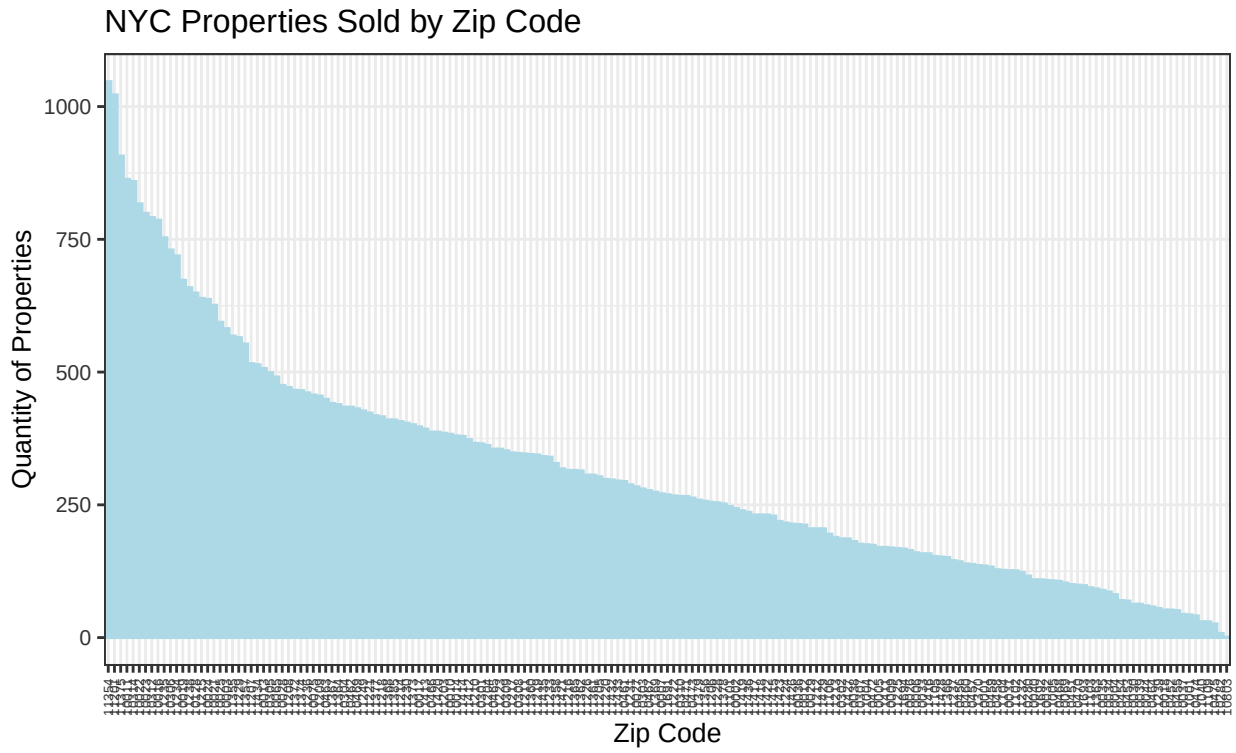
```
## [1] 43146
```

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

4.2.11 Zip Code

There are null values to remove. It has to be numeric.

```
## [1] 219
```



The top 5 zip codes of sold properties are 11354 (Queens, NYC), 11201(Brooklyn, NYC), 11375 (Queens, NYC), 10011 (Manhattan, NYC), 10314 (Staten Island, NYC)

4.2.12 ResidentionUnit

It has a lot of null values, but I'm not going to remove them because it's related with TotalUnit. I'm leave it as it for analysis.

4.2.13 CommercialUnit

It has a lot of null values, but I'm not going to remove them because it's related with TotalUnit. I'm leave it as it for analysis.

4.2.14 TotalUnit

This feature is composed by the sum of CommercialUnit + ResidentionUnit. There are highest correlated. The observations that are null will be deleted.

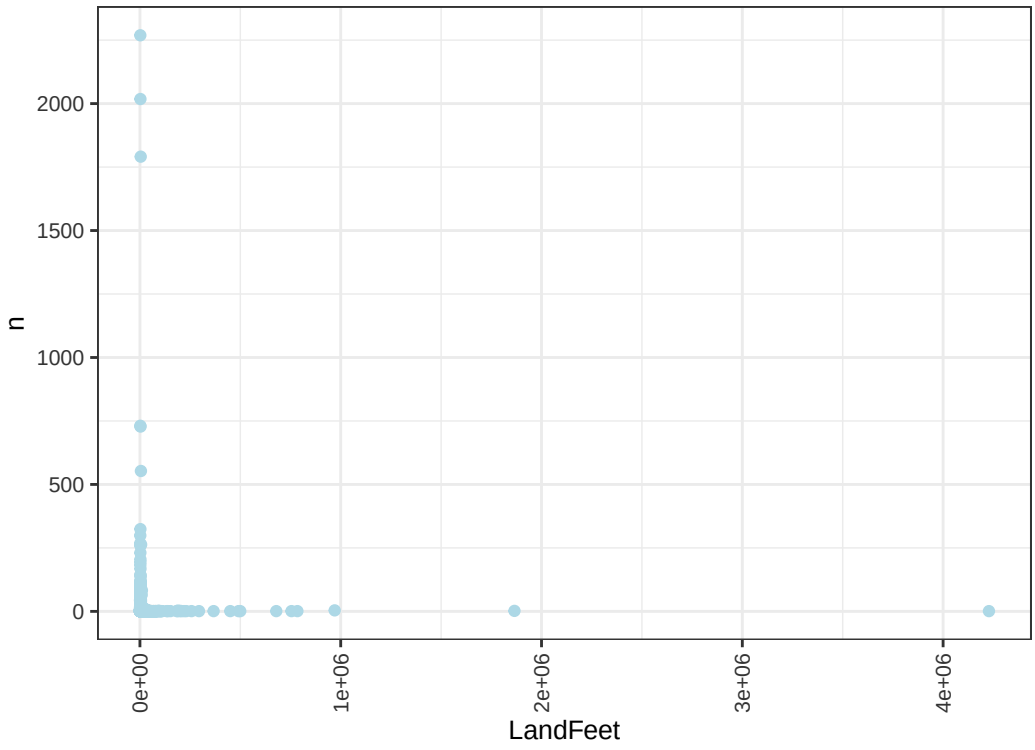
4.2.15 LandFeet

It has invalid values "0" and "-" to be removed. It has to be a numeric feature.

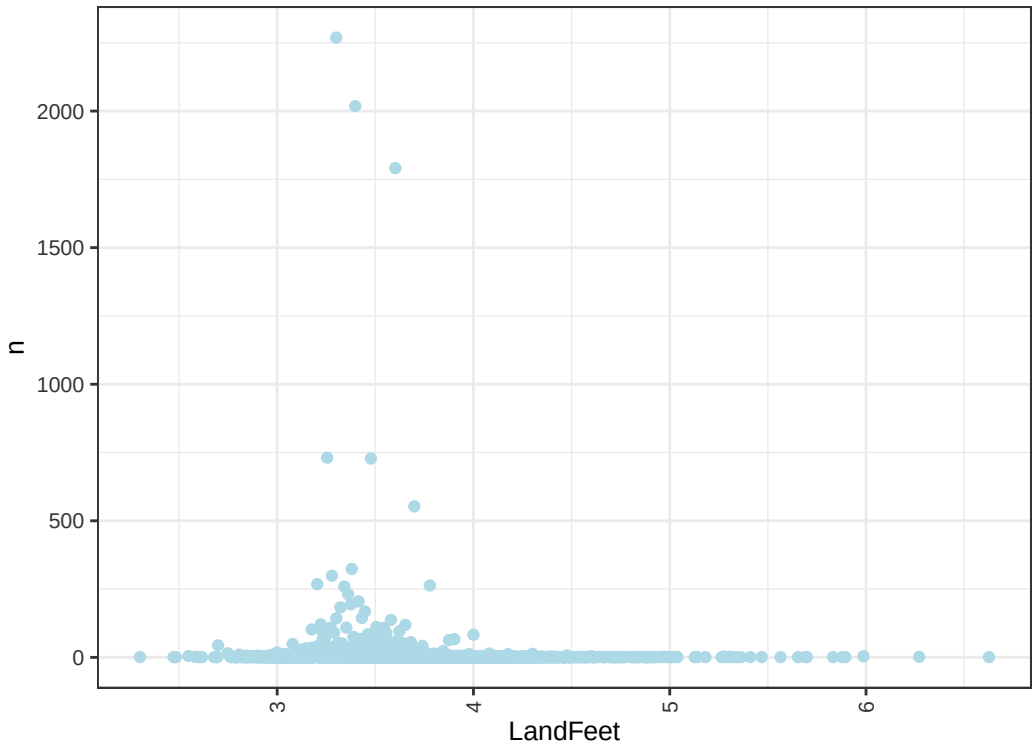
```
## [1] 4227
```

```
## [1] 8773
```

This is an important feature for my analysis, Here is a plot to show us the distribution



I'm going to apply the logarithm to reduce the skew.



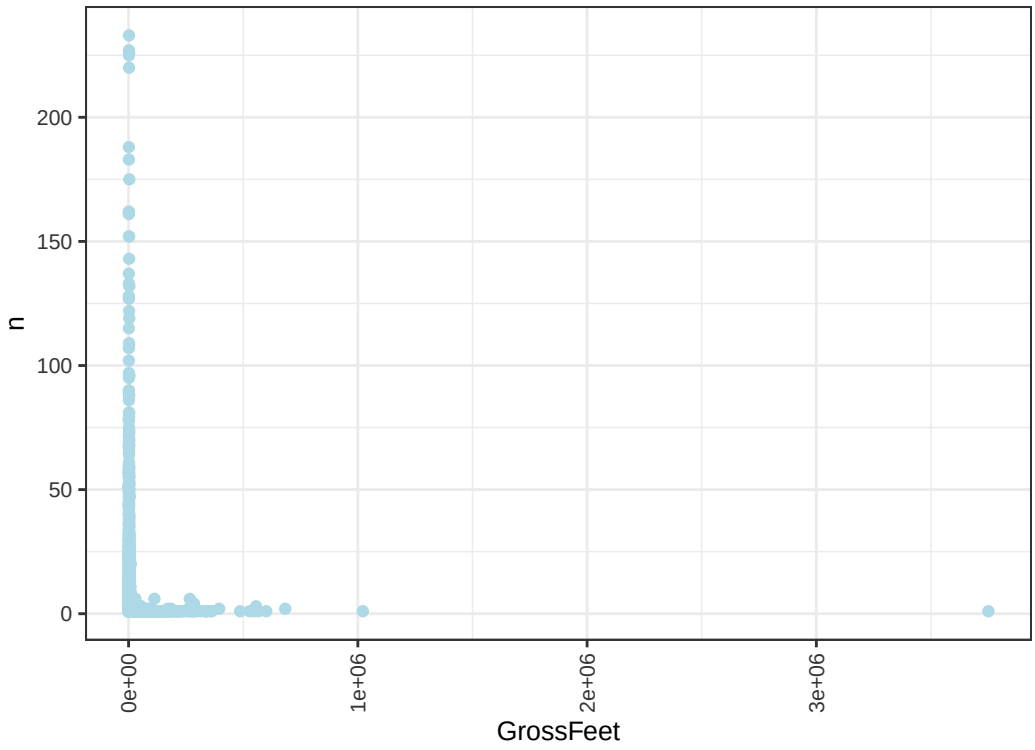
4.2.16 GrossFeet

It has invalid values “0” and “-” to be removed. It has to be a numeric feature.

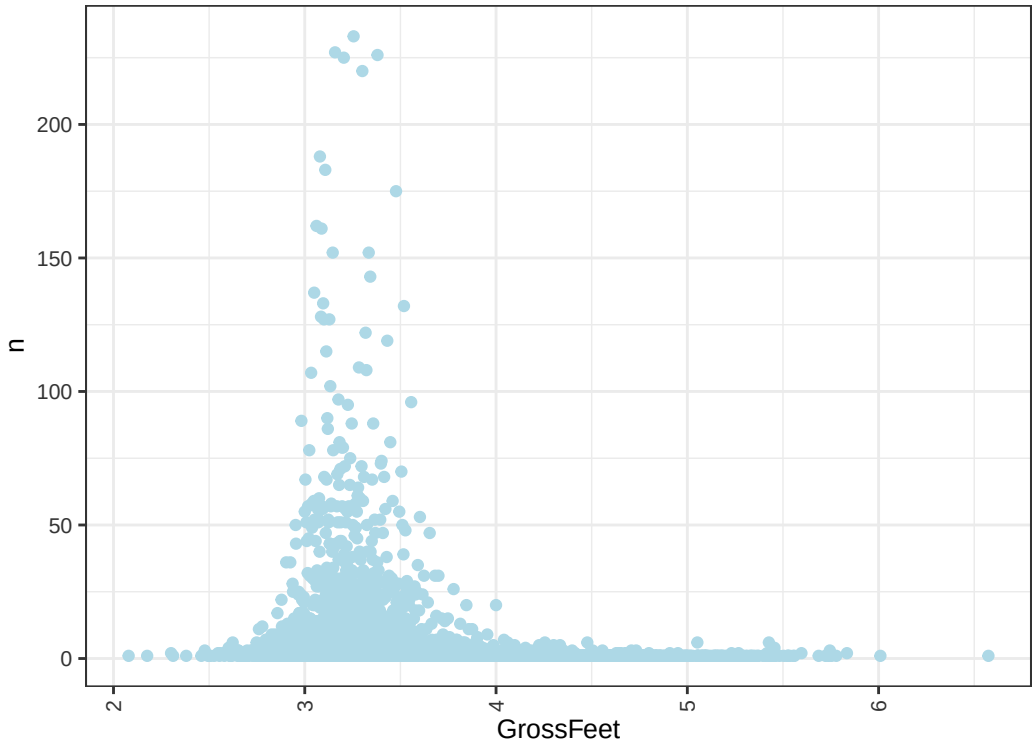
```
## [1] 50
```

```
## [1] 137
```

This is an important feature for my analysis, Here is a plot to show us the distribution.



I'm going to apply the logarithm to reduce the skew.



4.2.17 YearBuilt

It has some invalid values and it has to be numeric.

```
## [1] 8
```

4.2.18 TaxClassSale

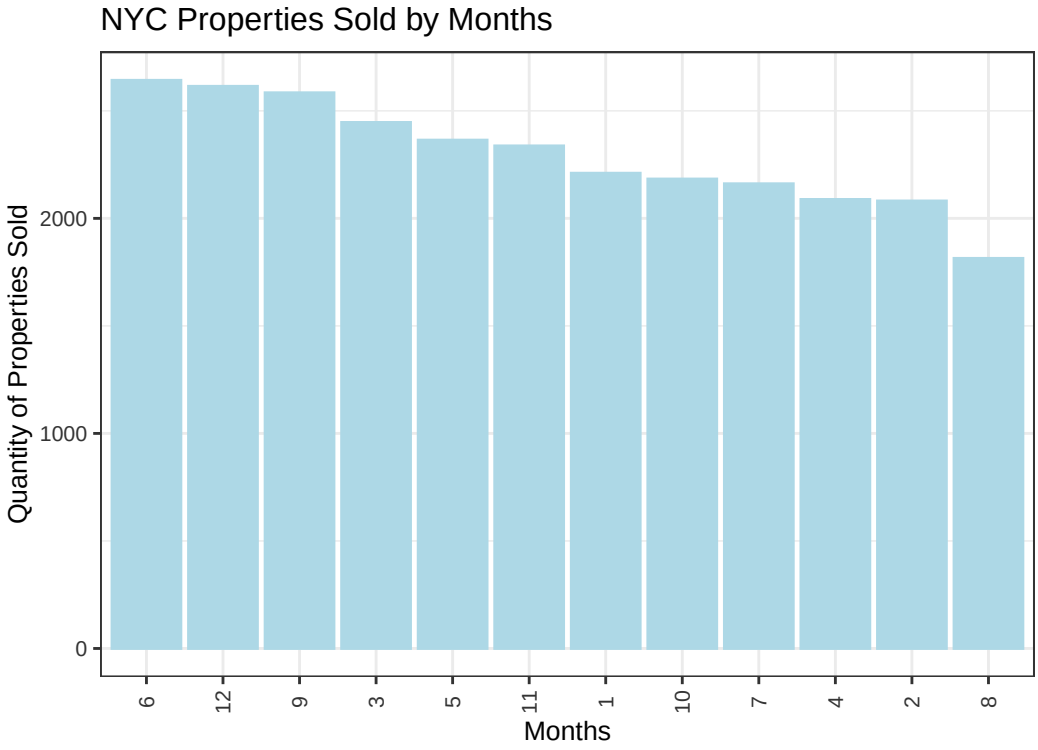
It has to be numeric and categorical.

4.2.19 BuldingClassSale

It has to be numeric and categorical.

4.2.20 SaleDate

This features has to be set as date and I'm going to split it in 3 different features as YearSale, MonthSale and, DaySale and take off SaleDate. These 3 new features will be numeric.



This graph shows us that June, December, and September are the months when the majority of property sales occur in New York. On the other hand, August is the least amount of properties sales, therefore if I have to buy a property in NYC, August would be a starting point since there would not be as much demand for properties as in the rest of the months.

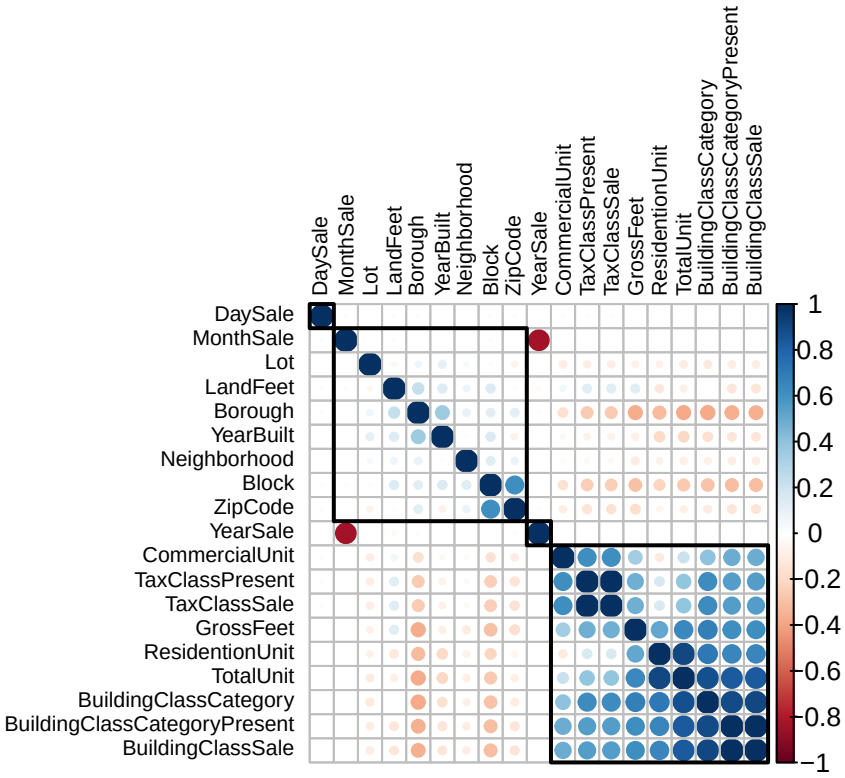
```
##        n
## 1: 27462
```

Once I have the dataset clean, I'm going to take a dataset portion to validate my final model.

5 Split the dataset - Create NYCProperties dataset, Validation dataset.

I'm going to take just 10% of the observations because the final dataset is small and I don't want to lose data to train my models. Then Validation dataset will be used just for my final model.

6 Correlatives Features

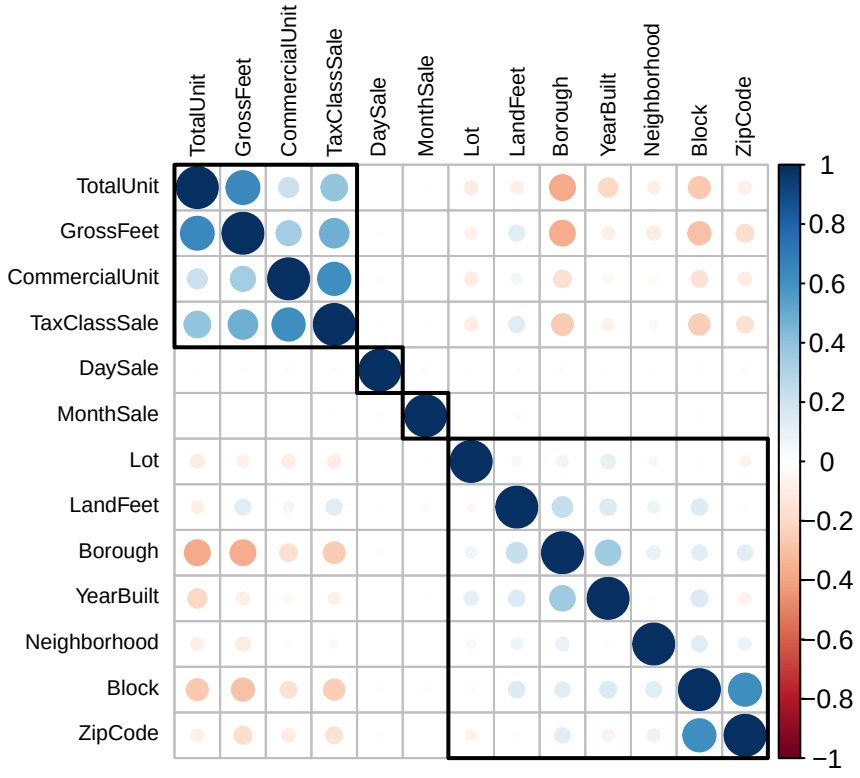


Here we can see:

- YearSale is highly uncorrelated with MonthYear.
- BuildingClassCategoryPresent, BuildingClassSale, BuildingClassCategory and, ResidentionUnit are highly correlated with TotalUnit.
- TaxClassPresent is highly correlated with TaxClassSale.

  I'm going to remove these features because they bring the same information without any added value. These features will be removed in validation dataset as well to keep the consistence.

  After remove these features there are not correlatives features. You can see it clearly in the below plot.



Reviewing the dataset structure.

```
str(NYCProperties, strict.width = "wrap")
```

```
## Classes 'data.table' and 'data.frame':   25095 obs. of  14 variables:
## $ Borough : int 1 1 1 1 1 1 1 1 1 1 ...
## $ Block : int 392 402 404 406 400 376 391 393 394 390 ...
## $ Lot : int 6 21 55 32 21 14 19 4 5 34 ...
## $ Neighborhood : num 2 2 2 2 2 2 2 2 2 2 ...
## $ ZipCode : num 10009 10009 10009 10009 10009 ...
## $ CommercialUnit: int 0 0 0 0 0 0 1 1 1 1 ...
## $ TotalUnit : int 5 10 6 8 10 24 4 5 6 1 ...
## $ LandFeet : num 3.21 3.36 3.37 3.24 3.57 ...
## $ GrossFeet : num 3.81 3.83 3.66 3.63 4.09 ...
## $ YearBuilt : num 1900 1913 1900 1920 2009 ...
## $ TaxClassSale : num 2 2 2 2 2 2 2 2 2 3 ...
## $ MonthSale : num 7 9 11 9 10 6 11 1 4 9 ...
## $ DaySale : num 19 23 17 23 17 21 15 30 3 28 ...
## $ SalePrice : num 6.82 6.6 6.9 6.5 7.01 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

There is a lot of disparity between the values of each feature. To get a standard scale of measure I'm going to convert each value of features to a range between the max and min of each one. It will influence the prediction accuracy.

Now we can see a standard scale of measures.

```r
summary(NYCProperties, strict.width = "wrap")
```

```
##     Borough           Block             Lot            Neighborhood
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:0.5000   1st Qu.:0.1704   1st Qu.:0.004853   1st Qu.:0.2320
##  Median :0.7500   Median :0.3062   Median :0.009976   Median :0.4720
##  Mean   :0.6326   Mean   :0.3459   Mean   :0.016404   Mean   :0.4872
##  3rd Qu.:0.7500   3rd Qu.:0.4855   3rd Qu.:0.016986   3rd Qu.:0.7600
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.0000
##     ZipCode       CommercialUnit        TotalUnit           LandFeet
##  Min.   :0.0000   Min.   :0.0000000   Min.   :0.0000000   Min.   :0.0000
##  1st Qu.:0.2741   1st Qu.:0.0000000   1st Qu.:0.0000000   1st Qu.:0.2312
##  Median :0.7212   Median :0.0000000   Median :0.0004425   Median :0.2536
##  Mean   :0.5963   Mean   :0.0001411   Mean   :0.0009964   Mean   :0.2655
##  3rd Qu.:0.8116   3rd Qu.:0.0000000   3rd Qu.:0.0004425   3rd Qu.:0.3008
##  Max.   :1.0000   Max.   :1.0000000   Max.   :1.0000000   Max.   :1.0000
##    GrossFeet        YearBuilt       TaxClassSale       MonthSale
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.2346   1st Qu.:0.5530   1st Qu.:0.00000   1st Qu.:0.2727
##  Median :0.2652   Median :0.5991   Median :0.00000   Median :0.4545
##  Mean   :0.2763   Mean   :0.6461   Mean   :0.08797   Mean   :0.5098
##  3rd Qu.:0.2993   3rd Qu.:0.7143   3rd Qu.:0.00000   3rd Qu.:0.8182
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##     DaySale         SalePrice
##  Min.   :0.0000   Min.   :5.003
##  1st Qu.:0.2667   1st Qu.:5.652
##  Median :0.5333   Median :5.810
##  Mean   :0.5213   Mean   :5.865
##  3rd Qu.:0.7667   3rd Qu.:5.991
##  Max.   :1.0000   Max.   :8.261
```

```r
summary(validation, strict.width = "wrap")
```

```
##     Borough           Block             Lot            Neighborhood
##  Min.   :0.000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.500   1st Qu.:0.1644   1st Qu.:0.01498   1st Qu.:0.2249
##  Median :0.750   Median :0.2847   Median :0.02918   Median :0.4578
##  Mean   :0.627   Mean   :0.3178   Mean   :0.04342   Mean   :0.4811
##  3rd Qu.:0.750   3rd Qu.:0.4335   3rd Qu.:0.04890   3rd Qu.:0.7590
##  Max.   :1.000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##     ZipCode       CommercialUnit     TotalUnit           LandFeet
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:0.2717   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.2765
##  Median :0.7177   Median :0.0000   Median :0.006711   Median :0.3088
##  Mean   :0.5763   Mean   :0.0116   Mean   :0.011761   Mean   :0.3227
##  3rd Qu.:0.8057   3rd Qu.:0.0000   3rd Qu.:0.006711   3rd Qu.:0.3695
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.000000   Max.   :1.0000
##    GrossFeet        YearBuilt       TaxClassSale       MonthSale
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.2293   1st Qu.:0.4696   1st Qu.:0.0000   1st Qu.:0.1818
##  Median :0.2749   Median :0.5249   Median :0.0000   Median :0.4545
##  Mean   :0.2904   Mean   :0.5861   Mean   :0.0864   Mean   :0.4978
```

```
##  3rd Qu.:0.3254   3rd Qu.:0.6906   3rd Qu.:0.0000   3rd Qu.:0.8182
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##     DaySale         SalePrice
##  Min.   :0.0000   Min.   :5.021
##  1st Qu.:0.2667   1st Qu.:5.648
##  Median :0.5333   Median :5.810
##  Mean   :0.5187   Mean   :5.859
##  3rd Qu.:0.7667   3rd Qu.:5.989
##  Max.   :1.0000   Max.   :8.108
```

Some algorithms that I'm going to use,such as KNN classifies objects by finding similarities between them using distance functions. These algorithms are receptive to the size of the variables. If we don't scale, the feature with a higher magnitude will have more influence than the feature with a lower magnitude, and this leads to bias in data, and it also affects the accuracy. For more detail see the References section "Data Normalization".

7 Predicting the Sale Price of a NYC Property.

The dataset that I chose will be evaluated according to the RMSE metric.

The RMSE() function available in the package in R is used to calculate the root mean square error between actual values and predicted values.

The lower values of RMSE indicate a better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criteria to fit the main purpose of the model is prediction.

The first step is to split my dataset for training and test. This split will use to measure the performance of the models.

The Validation dataset will be just used for prediction with the best performance model. To follow the same criteria, like validation, the test will be 10 % of the data.

Once the training and test data sets are split, I'm going to start with Linear Regression. LR will be my baseline approach. After that, I'm going to work with Support Vector Regression. Then, I'm going to implement KNN to have a different model. Finally, I'm going to work with Decision Tree models, Decision Tree, Random Forest and, Gradient Boosting Machines at the end. Showing a different option as well.

I choose four different kinds of models because I would like to expand my research with ensembling modeling. The ensemble method combines multiple models allowing to produce better predictions compared to a single model.

I'll set up a seed in each model to be sure everyone can reproduce the same RMSEs and, I'll show the RMSE values in a table to make easier the comparison between them.

7.1 Create train and test sets from NYCProperties dataset.

7.1.1 Train Dataset (train_NYCProperties)

```
str(train_NYCProperties, strict.width = "wrap")
```

```
## Classes 'data.table' and 'data.frame':   22584 obs. of  14 variables:
## $ Borough : num 0 0 0 0 0 0 0 0 0 0 ...
## $ Block : num 0.0237 0.0243 0.0245 0.0242 0.0227 ...
```

```
## $ Lot : num 0.00135 0.00539 0.01456 0.00539 0.0035 ...
## $ Neighborhood : num 0.004 0.004 0.004 0.004 0.004 0.004 0.004 0.156 0.156
##    0.156 ...
## $ ZipCode : num 0.00473 0.00473 0.00473 0.00473 0.00473 ...
## $ CommercialUnit: num 0 0 0 0 0 ...
## $ TotalUnit : num 0.00177 0.00398 0.00221 0.00398 0.01018 ...
## $ LandFeet : num 0.211 0.244 0.248 0.293 0.304 ...
## $ GrossFeet : num 0.385 0.39 0.353 0.448 0.477 ...
## $ YearBuilt : num 0.461 0.521 0.461 0.963 0.59 ...
## $ TaxClassSale : num 0.5 0.5 0.5 0.5 0.5 0.5 1 0 0 0 ...
## $ MonthSale : num 0.545 0.727 0.909 0.818 0.455 ...
## $ DaySale : num 0.6 0.733 0.533 0.533 0.667 ...
## $ SalePrice : num 6.82 6.6 6.9 7.01 7.08 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

7.1.2 Test Dataset (test_NYCProperties)

```
str(test_NYCProperties, strict.width = "wrap")
```

```
## Classes 'data.table' and 'data.frame':   2511 obs. of  14 variables:
## $ Borough : num 0 0 0 0 0 0 0 0 0 0 ...
## $ Block : num 0.0246 0.0237 0.0238 0.0481 0.0488 ...
## $ Lot : num 0.00836 0.00485 0.00108 0.00944 0.00485 ...
## $ Neighborhood : num 0.004 0.004 0.004 0.156 0.156 0.172 0.176 0.176 0.176
##    0.284 ...
## $ ZipCode : num 0.00473 0.00473 0.00473 0.00591 0 ...
## $ CommercialUnit: num 0 0.000442 0.000442 0.000885 0.008403 ...
## $ TotalUnit : num 0.0031 0.00133 0.00221 0.00133 0.00796 ...
## $ LandFeet : num 0.218 0.204 0.219 0.208 0.322 ...
## $ GrossFeet : num 0.344 0.322 0.332 0.345 0.574 ...
## $ YearBuilt : num 0.553 0.507 0.507 0.553 0.558 ...
## $ TaxClassSale : num 0.5 0.5 0.5 1 1 0.5 0.5 0.5 0.5 0.5 ...
## $ MonthSale : num 0.7273 0.9091 0.2727 0.2727 0.0909 ...
## $ DaySale : num 0.7333 0.4667 0.0667 0.5667 0 ...
## $ SalePrice : num 6.5 6.52 6.68 7.03 7.52 ...
## - attr(*, ".internal.selfref")=<externalptr>
```

7.2 Data Preparation

```
y_train <- train_NYCProperties$SalePrice
x_train <- train_NYCProperties[,!("SalePrice")]

y_test <- test_NYCProperties$SalePrice
x_test <- test_NYCProperties[,!("SalePrice")]

y_val <- validation$SalePrice
x_val <- validation[,!("SalePrice")]
```

During the implementation of each model you will see:

- A table with a **train row** showing the result of the model using the **training data** and, a **test row** showing the result of the model using **unseen data**.
  This table shows us how well the model is generalizing and how it potentially will behave with future information.
- A plot showing us the **difference** between using the training data and the model using unseen data (sale price prediction). The y-axis shows us this difference in *sale price* applying in each model.

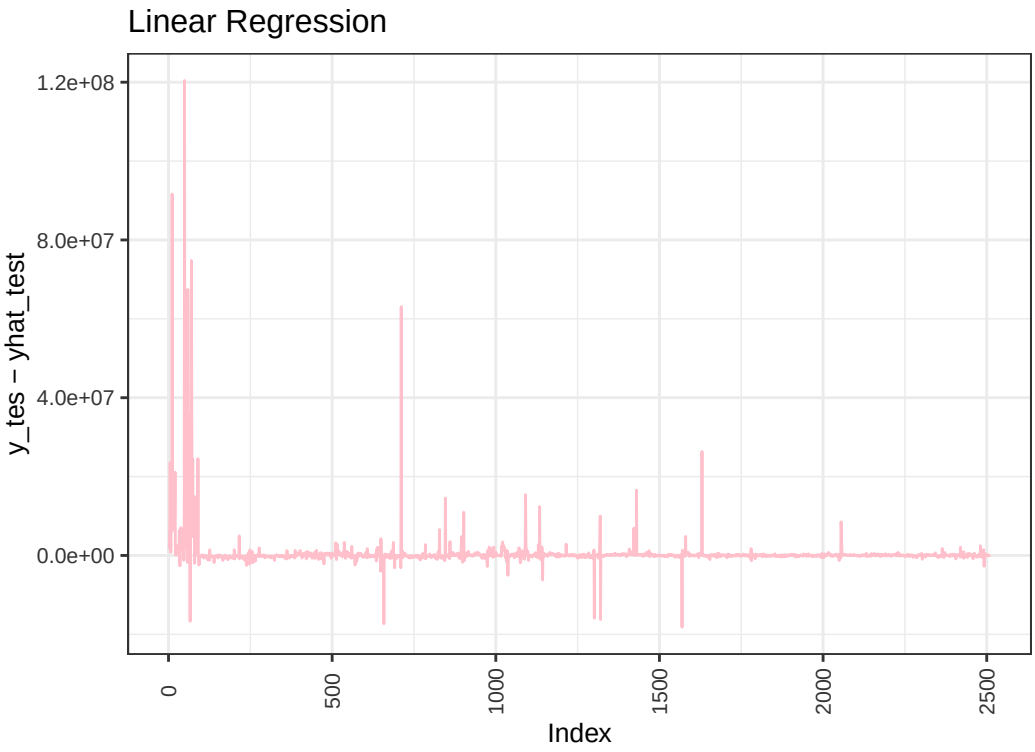
8 Model #1 - Linear Regression - Baseline Approach

The first model is the baseline approach.

```
set.seed(1, sample.kind = "Rounding")

lr.fit <- lm(
  formula = SalePrice ~.,
  data = train_NYCProperties)

lr.yhat_train <- lr.fit %>% predict(as.data.frame(x_train))
lr.yhat_test <- lr.fit %>% predict(as.data.frame(x_test))

lr.train_rmse <- RMSE(lr.yhat_train, y_train)
lr.test_rmse <- RMSE(lr.yhat_test, y_test)
```

Table 1: RMSEs

Method	RMSE
Linear Regression - Train	0.2485105
Linear Regression - Test	0.2435840



9 Model #2 - Support Vector Regression

This is the second model applying Support Vector Regression with default values.

```
library(e1071)
set.seed(1, sample.kind = "Rounding")

svr.fit <- svm(
  formula = SalePrice ~ .,
  data=train_NYCProperties)

svr.yhat_train <- svr.fit %>% predict(as.data.frame(x_train))
svr.yhat_test <- svr.fit %>% predict(as.data.frame(x_test))

svr.train_rmse <- RMSE(svr.yhat_train, y_train)
svr.test_rmse <- RMSE(svr.yhat_test, y_test)
```

Table 2: RMSEs

Method	RMSE
Support Vector Regression - Train	0.1945428
Support Vector Regression - Test	0.1959461

10 Model #3 - KNN

This is the third model applying k-nearest Neighbors algorithm. In this case I've be working tuning the model, looking for the better K (number of neighbors).

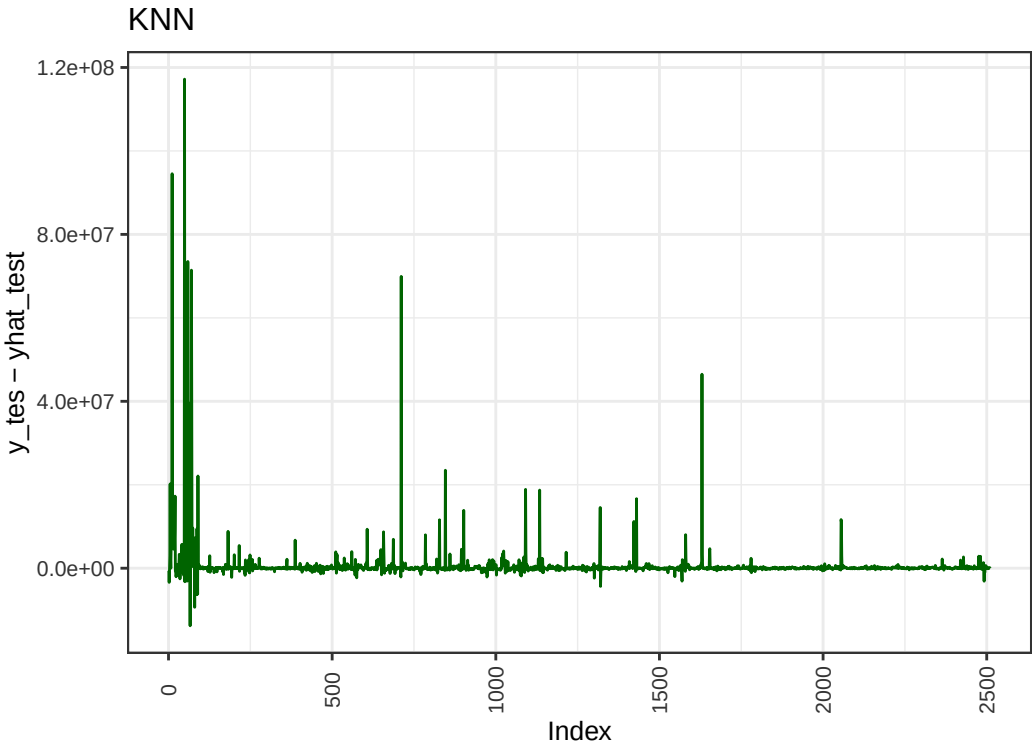
```
set.seed(1, sample.kind = "Rounding")
knn.fit <- train(
  form = SalePrice ~ .,
  method = "knn",
  data = train_NYCProperties,
  tuneGrid = data.frame(#k = seq(15, 35, 2)
                        k = 15)) #bestTune
knn.fit$bestTune
```

```
##    k
## 1 15
```

```
knn.yhat_train <- knn.fit %>% predict(as.data.frame(x_train))
knn.yhat_test <- knn.fit %>% predict(as.data.frame(x_test))

knn.train_rmse <- RMSE(knn.yhat_train, y_train)
knn.test_rmse <- RMSE(knn.yhat_test, y_test)
```

Table 3: RMSEs

Method	RMSE
KNN - Train	0.2127271
KNN - Test	0.2253979



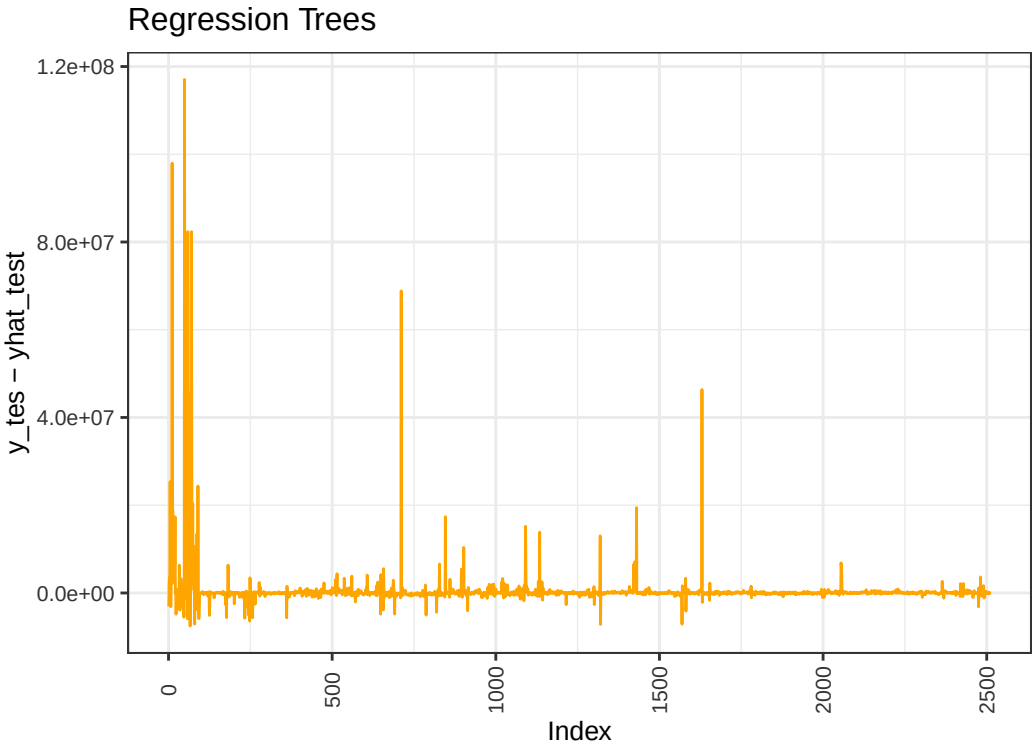
11 Model #4 - Regression Trees

This is the fourth model applying Regression Trees algorithm. In this models just apply method= "anova" for a regression tree.

```
library(rpart)
set.seed(1, sample.kind = "Rounding")

rt.fit <- rpart(
  formula = SalePrice ~ .,
  method="anova",
  data=train_NYCProperties)

rt.yhat_train <- rt.fit %>% predict(as.data.frame(x_train))
rt.yhat_test <- rt.fit %>% predict(as.data.frame(x_test))

rt.train_rmse <- RMSE(rt.yhat_train, y_train)
rt.test_rmse <- RMSE(rt.yhat_test, y_test)
```

Table 4: RMSEs

Method	RMSE
Regression Trees - Train	0.2362479
Regression Trees - Test	0.2325676



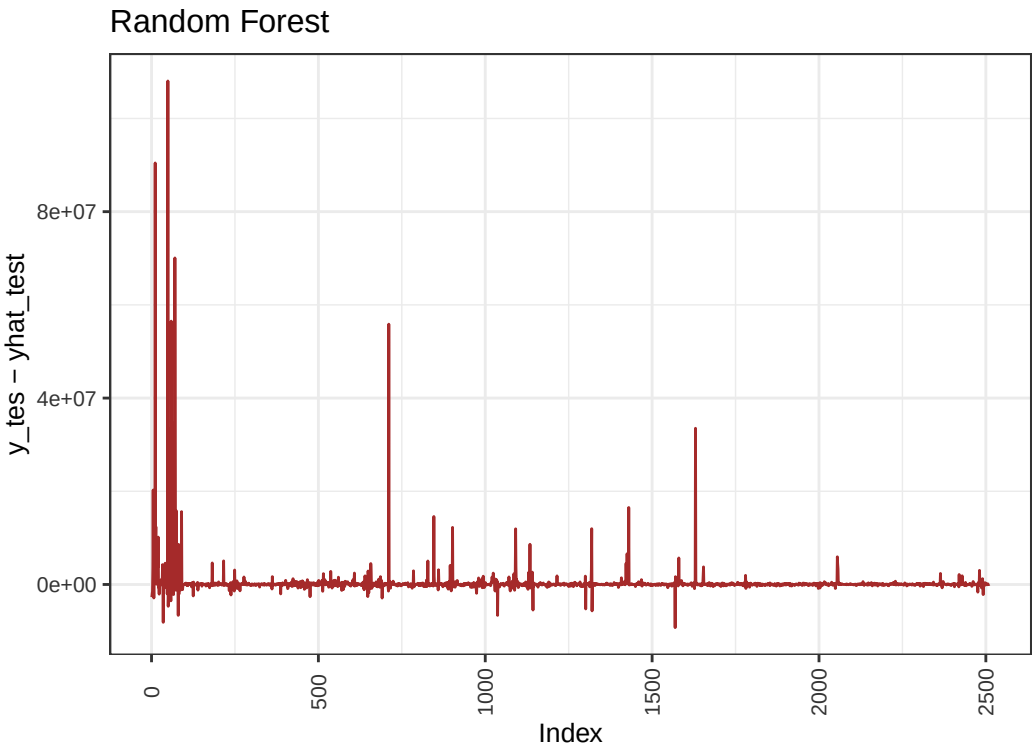
12 Model #5 - Random Forest

This is the fifth model applying Random Forest algorithm. This model has been evaluated with default values. It showed difference between train-RMSE VS test-RMSE, which means overfitting .
Overfitting is characteristic of RF models. To avoid it, I added the parameter nodesize = 50 (min.obs.in a terminal node). This parameter is by default 5.

```
library(randomForest)
set.seed(1, sample.kind = "Rounding")

rf.fit <- randomForest(
  formula = SalePrice ~ .,
  data = train_NYCProperties,
  nodesize = 50              # min. #observations in a terminal node
  )

rf.yhat_train <- rf.fit %>% predict(as.data.frame(x_train))
rf.yhat_test <- rf.fit %>% predict(as.data.frame(x_test))

rf.train_rmse <- RMSE(rf.yhat_train, y_train)
rf.test_rmse <- RMSE(rf.yhat_test, y_test)
```

Table 5: RMSEs

Method	RMSE
Random Forest - Train	0.1513032
Random Forest - Test	0.1741518



13 Model #6 - Gradient Boosting Machines

This is the sixth model applying Gradient Boosting Machines Forest algorithm. In this model, I've been working on tuning some parameters. Below are the best tune values for this dataset.

```
library(gbm)
set.seed(1, sample.kind = "Rounding")

gbm.grid <- expand.grid(#interaction.depth = seq(12, 30, 2), # max_depth
                        interaction.depth = 30,              #bestTune
                        n.trees = 1000,                      # nround / # of Iterations
                        shrinkage = 0.01,                    # ETA / Learning Rate (Default Value)
                        n.minobsinnode = 10)                 # Default Value

gbm.ctrl <- trainControl(method="cv",                        # Cross Validation
                         number = 5,
                         verboseIter = FALSE)
```

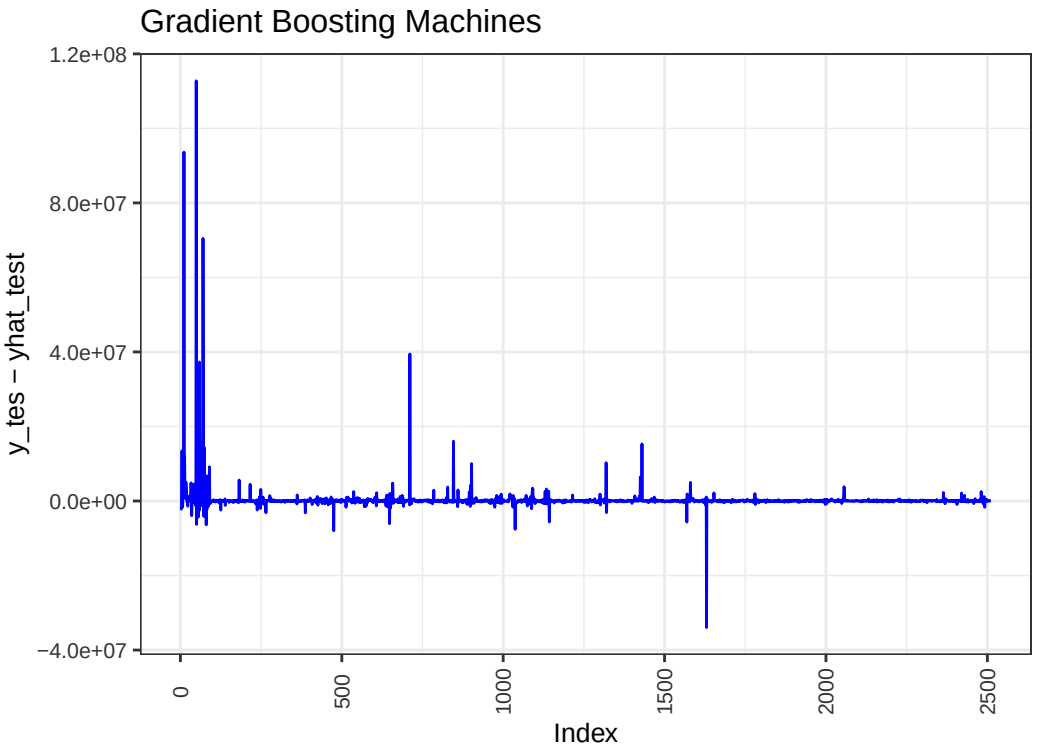
```
gbm.fit <- train(
  form = SalePrice ~ .,
  method = "gbm",
  data = train_NYCProperties,
  tuneGrid = gbm.grid,
  trControl = gbm.ctrl,
  verbose = FALSE)

gbm.fit$bestTune
```

```
##   n.trees interaction.depth shrinkage n.minobsinnode
## 1    1000                30      0.01             10
```

```
gbm.yhat_train <- gbm.fit %>% predict(as.data.frame(x_train))
gbm.yhat_test <- gbm.fit %>% predict(as.data.frame(x_test))

gbm.train_rmse <- RMSE(gbm.yhat_train, y_train)
gbm.test_rmse <- RMSE(gbm.yhat_test, y_test)
```

Table 6: RMSEs

Method	RMSE
Gradient Boosting Machines - Train	0.1583152
Gradient Boosting Machines - Test	0.1692144



14 Summary: RMSEs in Test dataset

Table 7: Final RMSE

Method	RMSE
Linear Regression - Test	0.2435840
Support Vector Regression - Test	0.1959461
KNN - Test	0.2253979
Regression Trees - Test	0.2325676
Random Forest - Test	0.1741518
Gradient Boosting Machines - Test	0.1692144

15 The best RMSE running in Validation Dataset.

The model that shows us the best predictions in the sale price of a property in NYC is *Gradient Boosting*. Now we are going to implement this model in the validation data set.

```
best.yhat <- gbm.fit %>% predict(validation)
best.val_rmse = RMSE(best.yhat, y_val)
```

Table 8: Final RMSE

Method	RMSE
Linear Regression - Test	0.2435840
Support Vector Regression - Test	0.1959461
KNN - Test	0.2253979
Regression Trees - Test	0.2325676
Random Forest - Test	0.1741518
Gradient Boosting Machines - Test	0.1692144
Gradient Boosting Machines - Validation	0.2511636

16 Conclusion

After the dataset has been cleaning and normalize to get consistent data for the goal of my project. I have been working with different models (Linear Regression, Neighbors, Vector Regression and, Decision Tree algorithms) to predict the sale price of a property in NYC. Some models worked better than others. With some of those I have been implemented tuning strategies to get a better RMSE and with other models just left them as the default setup. To see the performance in each model I used the training dataset comparing with the test dataset. The difference between them has been plotted for a better understanding. Finally, I took the model that better fit (smaller RMSE) predicting the sale price in the validation partition with unseen information. I am satisfied with the results I obtained, analysis of the dataset, and the implemented models. This project will be the starting point to continue with the study of the ensemble models to obtain a much better prediction to apply in my real estate business.

16.1 Potential Impact

This project can help the real estate investor to have an idea about a property sale price according to the market.

16.2 Limitations

The limitation of this project is the size of the dataset. This limitation can bring the model to not have a very accurate prediction. The model's result may be improved by gathering more updated information.

16.3 Future work

As a future work with this project, we can make a deeper study of ensemble models that can lead to improving the prediction and robustness of the sale price of the properties. Ensemble methods are meta-algorithms that combine several machine learning models into one predictive model.

17 References

- Data set
  - https://www.kaggle.com/annavictoria/ml-friendly-public-datasets?utm_medium=email&utm_source=intercom&utm_campaign=data+projects+onboarding
  - https://www.kaggle.com/new-york-city/nyc-property-sales
  - https://www1.nyc.gov/assets/finance/downloads/pdf/07pdf/glossary_rsf071607.pdf (Features Definitions)
  - https://www1.nyc.gov/assets/finance/jump/hlpbldgcode.html (Building Classification)
  - https://www.unitedstateszipcodes.org/ (NYC Zip Codes)
- General Information
  - https://rafalab.github.io/dsbook/index.html
- General Information in R
  - https://cran.r-project.org/
  - https://stackoverflow.com/
  - https://jkzorz.github.io/2019/06/11/Correlation-heatmaps.html
- Data Normalization
  - https://medium.com/swlh/data-normalisation-with-r-6ef1d1947970
- Decision Tree - Random Forest
  - https://towardsdatascience.com/understanding-random-forest-58381e0602d2
  - https://cran.r-project.org/web/packages/randomForest/randomForest.pdf