MyProject - NYC Property Sales

Anton Prešo

May 2023

Executive Summary

This project is part of the HarvardX PH125.9x Data Science: Capstone online course.
The main aim of the project is use Machine Learning techniques to predict values of the Y variable by showing skills and abilities gained during courses attended. To do this, we need to to have cleaned and prepared dataset. We are using Random Forest (RF) Machine Learning process to develop suitable predicting algorithm by tuning the parameters and evaluating by RMSE.
The most suitable (with the lowest RMSE) RF model that is used on test dataset RF with tuneRF mTry value and chosen trees with RMSE of 0.5146248. Final RMSE with model used on test data is 0.5032866.
We are using R studio software to show different ways how to deal with raw dataset, which methods can be used to clean, edit and prepare downloaded dataset for further data analysis and machine Learning process. Well prepared dataset is one of the fundamentals of good analysis.
To show this we are using Kaggle dataset about NYC property sales over a 12-month period with mostly raw unedited data. After inspecting dataset We are splitting and extracting date, adding borough name instead of numeric value, calculating property age, removing rows containing NAs and dealing with outliers and 0 values. After editing and cleaning dataset we there is 37751 rows containing data (44.7% of default dowloaded dataset).
Second aim of the project is to use cleaned dataset for correlation analysis to find out if there is linear relationship between variables. We are mostly interested in correlation coefficients connected with variable containing sale price of the property. We found out that there is correlation equal to 1 between Residential Units and Total Units and no correlation higher than 0.7 except between land and gross sq feet. Sale price is positively correlated with Total units and Residental units, Land and Gross sq feet, Number of commercial units and Tax class. It is also negatively correlated with Borough and Block and Year Built.
Third aim of the project is to standardize/transform/normalize data. Using standardization we were able to achieve a standardized data format across dataset. Standardized data suits better for Machine Learning techniques. We were able to fulfill all of the aims of the project.

Dataset and Methods

This project uses dataset "NYC Property Sales" by Kaggle and can be downloaded here or here.
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 (September 2016 - August 2017). It contains the location, address, type, sale price, and sale date of building units sold.
It is mostly raw dataset that needs to be cleaned and edited using appropriate methods descripted below.
For computations we are using R version 4.2.2 (The R Project for Statistical Computing) and RStudio Desktop (Open Source Edition AGPL v3) with appropriate packages.

Inspecting dataset

The first step of any data related task is to inspect the data we are dealing with. This is crucial for data wrangling as well, since we need to explore the current structure of the data, in order to identify the required

transformations.
After downloading and creating datasets we need to inspect them. We see main categories. We can see that we need to edit some of the variable types and to deal with NAs. We need to clean and prepare data for further analysis possible. We are mostly interested in Sales Price data.

```
##  [1] "X"                              "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"
```

Before editing, there is 84.548 rows of data.

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

First, we need to edit type of some variables. Then we also need to add new column with borough name instead of numeric value where Manhattan is defined as number 1, Bronx is 2, Brooklyn is 3, Queens is 4, and Staten Island is number 5. For better interpretations we are adding new column with Sale Price in Millions.

Next step is to calculate property age and subtract month from Sale date.

The biggest problem with the dataset is that it contains lot of NAs, 0 values, non-logical values, outliers or missing values.

Problem 1:

NAs in Sale Price (17% of total), Land (31%) and Gross (33%) square feet variables.

```
## [1] 0.1722335
```

```
## [1] 0.3104982
```

```
## [1] 0.3265837
```

Solution 1:

We are removing rows containing NA in SALE.PRICE column.

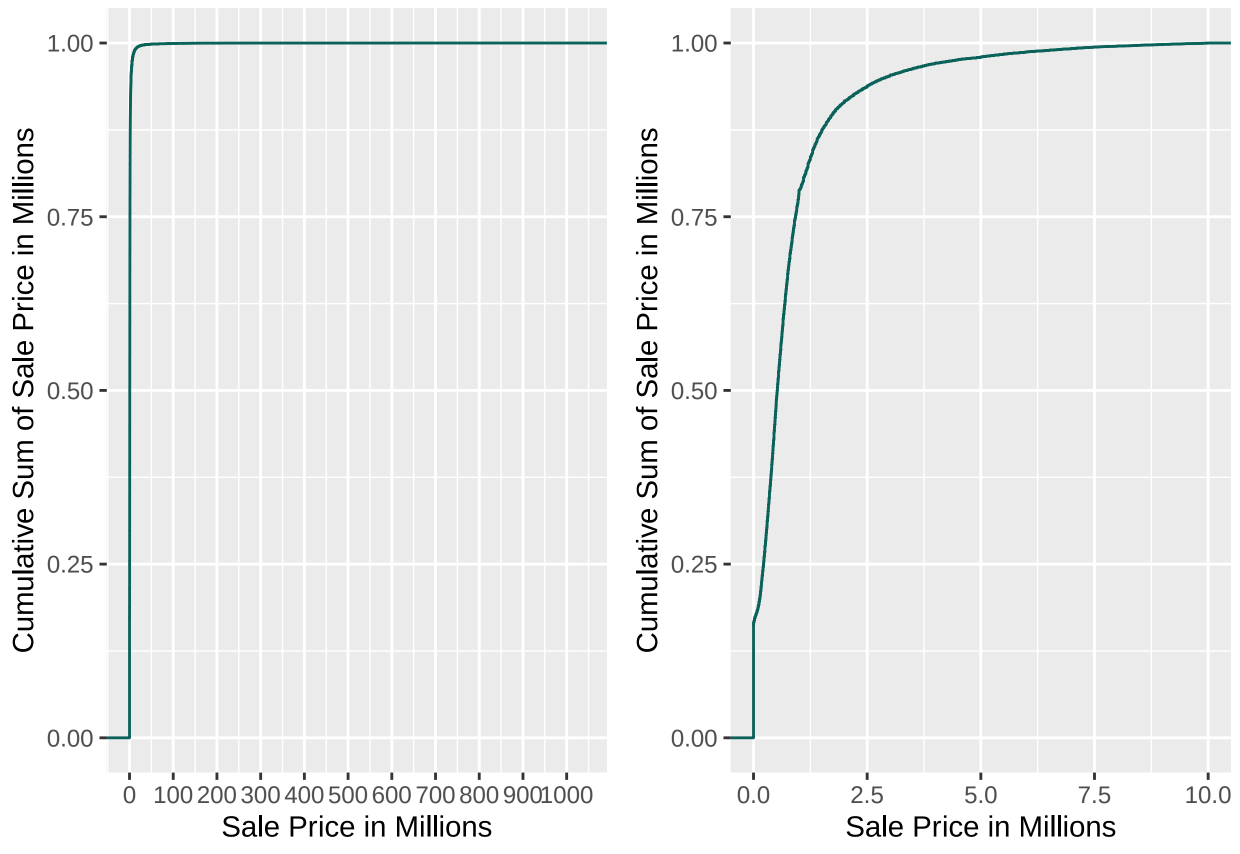
We can check if the problem was solved.

```
## [1] 0
```

```
## [1] 0
```

Problem 2:

Sale Price variable contains outliers and 0 values. Many sales occur with a nonsensically small dollar amount: $0 most commonly. These sales are actually transfers of deeds between parties: for example, parents transferring ownership to their home to a child after moving out for retirement.
Using Cumulative Sum of Sale Price in Millions we can see that around 99% of Sale Price are between 100k and 10 million.

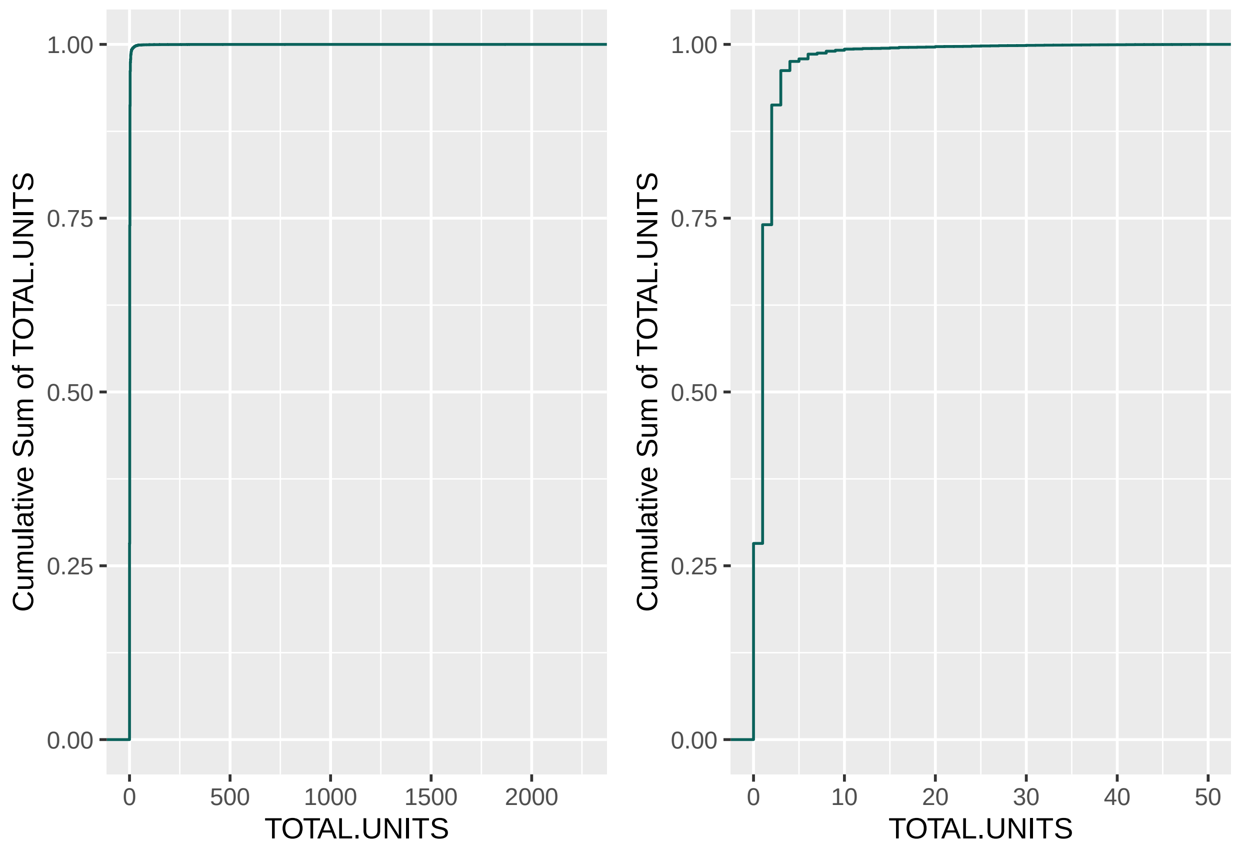


Solution 2:

We can drop values of Sale Price below 100k and over 10M (removing outliers and non-logical values).

Problem 3:

Total Units variable contains outliers and 0 values. Using Cumulative Sum of Total Units we can see that around 99% of Total Units are between 1 and 45.



There are properties that has 0 (MIN) and 2261 (MAX) total units.

```
## [1] 0
```

```
## [1] 2261
```

Solution 3:

We can drop values of total units that is 0 and over 45 (removing outliers and non-logical values).

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

```
## [1] 45
```

Problem 4:

There was 6970 NAs in YEAR.BUILT variable in default dataset.

```
## [1] 6970
```

Solution 4:

We can drop values of YEAR.BUILT that is 0 and check results.

```
## [1] 0
```

Another option to deal with outliers in all variables is to define outliers as a numbers out of boundaries of Mean +- 2 standard deviations (SD). "Mean +- 2SD" should contain around 95% of observations if dataset is large enough.

Characteristics of edited Dataset

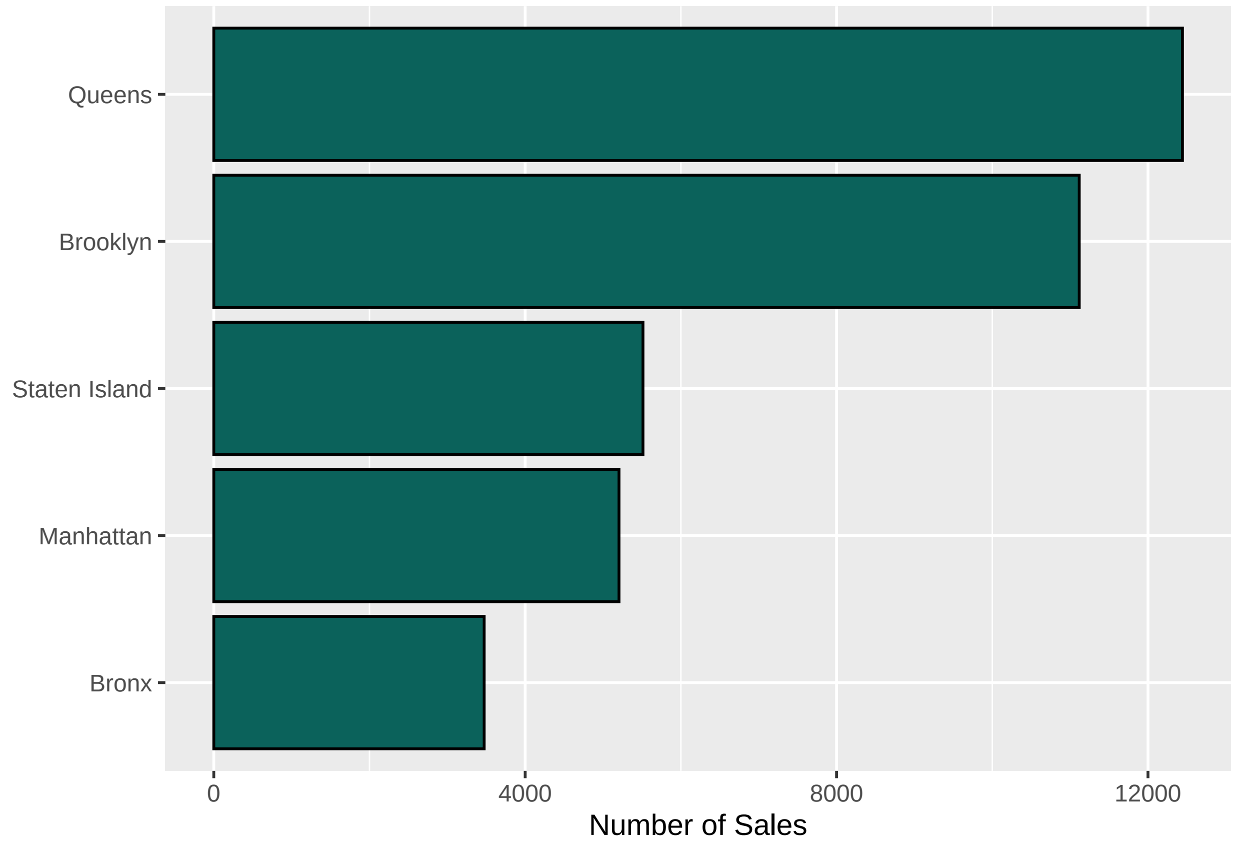
After editing of the dataset there is 37751 rows containing data (44.7% of default dataset). That's lower than half of the dowloaded rows but to further analysis we need correct structure and data.

```
nrow(sales_removed)
```

```
## [1] 37751
```

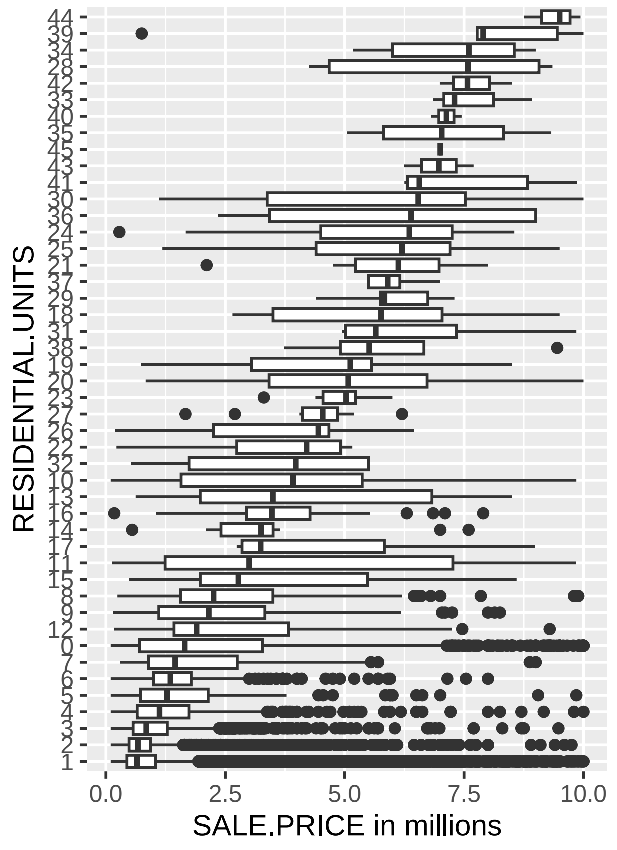
Borough and Neighborhood

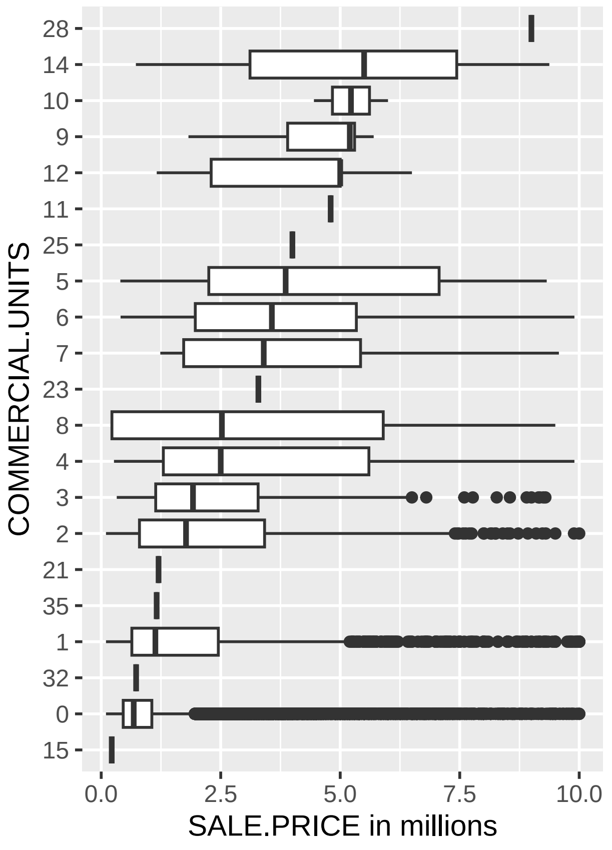
Most sales were made in Queens (12443) borough followed by Brooklyn (11117). Divided into Neighborhoods most properties were sold in Flushing-North (1611 sales, Queens) and Bedford–Stuyvesant (753, Brooklyn).



Number of Residential Units and Building Classification

Residential Units are defined as the number of residential units at the listed property. Most properties sold had 1 Residential Unit, that's 35% of sum of all the Residential Unit.
The Building Classification is used to describe a property's constructive use. R4 Building Class (Condo; Residential Unit In Elevator Bldg.) are most common with 7707 sales.
Highest median price has properties with higher number of residential units, but there is variability seen in median sale price of properties with residential and commercial units.

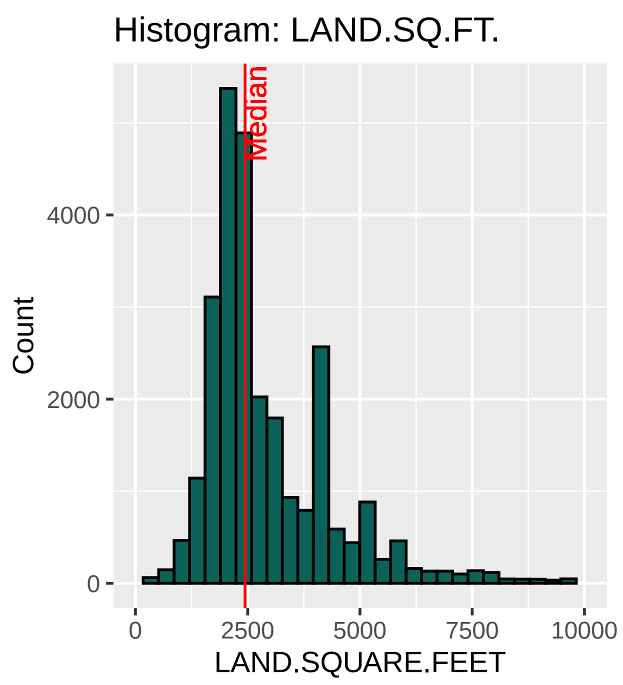


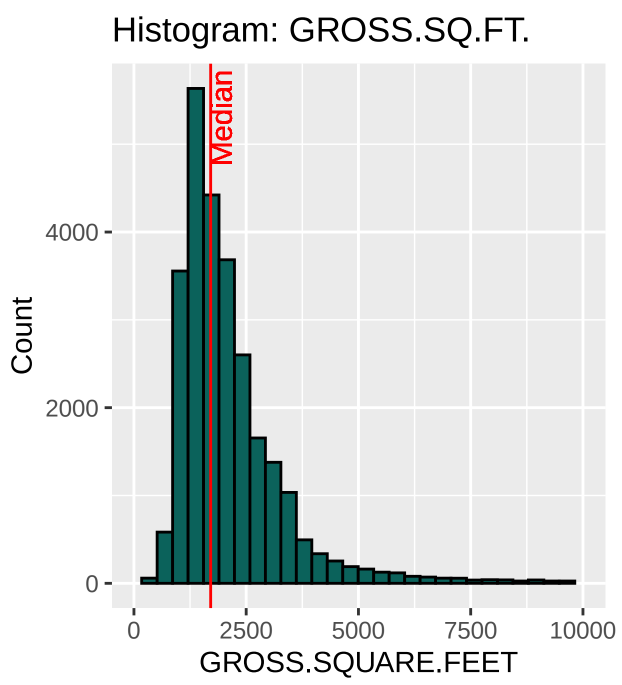


Land and Gross square Feet

Land Square Feet is defined as the land area of the property listed in square feet.

Gross square Feet is the total area of enclosed space measured to the exterior walls of a building. This is an umbrella term that includes everything in a facility, even unusable spaces. Average sold property had Gross Square Feet of 2333 with median value of 1709.



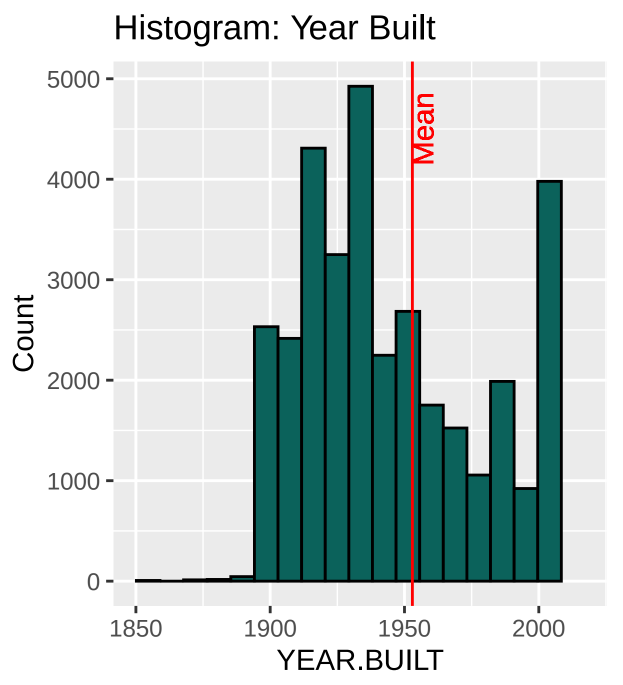


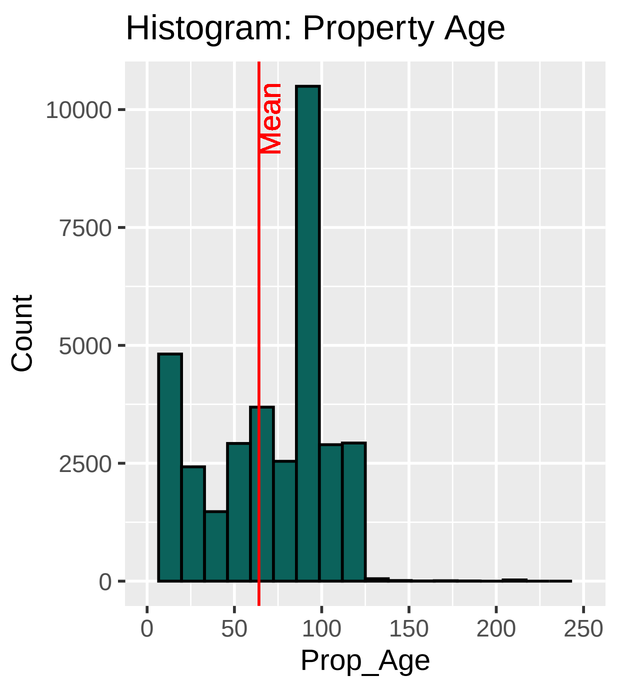
Year Built and Property Age

Average property sold is 64 years old (1952,9 is average year the structure on the property was built). The oldest propert is from 1800.

```
## [1] 64.0662
```

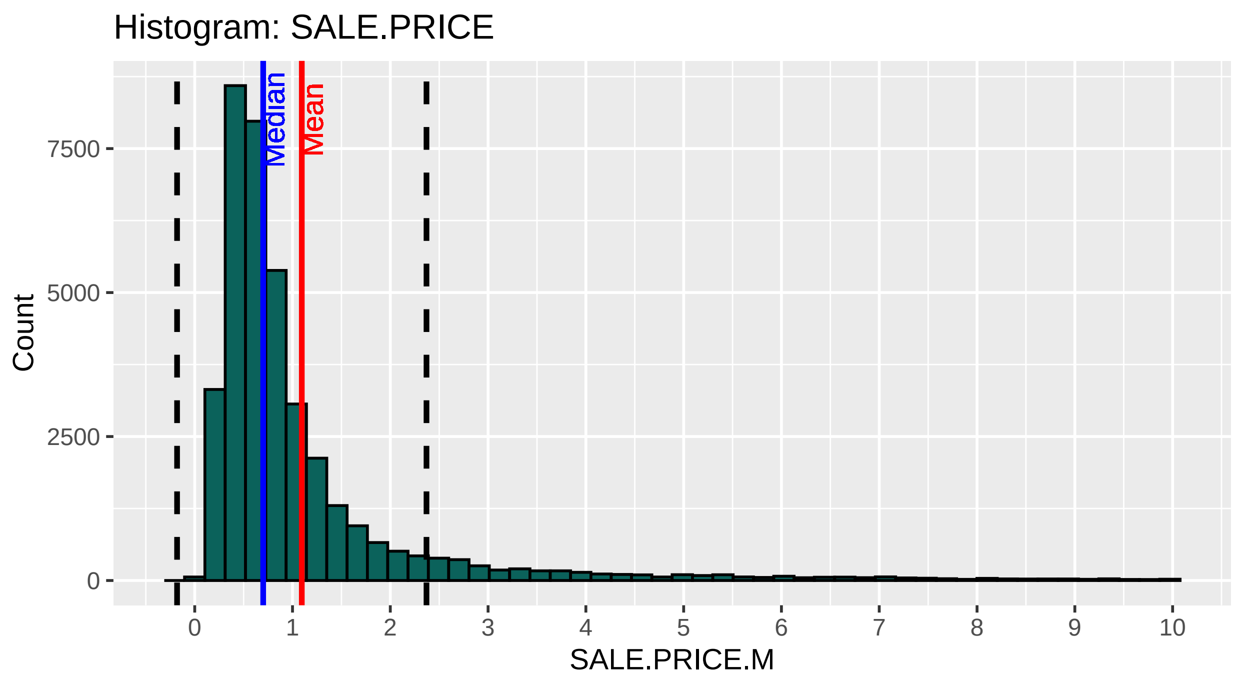
```
## [1] 1952.934
```





Sale Price

For the purpose of this analysis this is the most important variable and is defined as Price paid for the property.

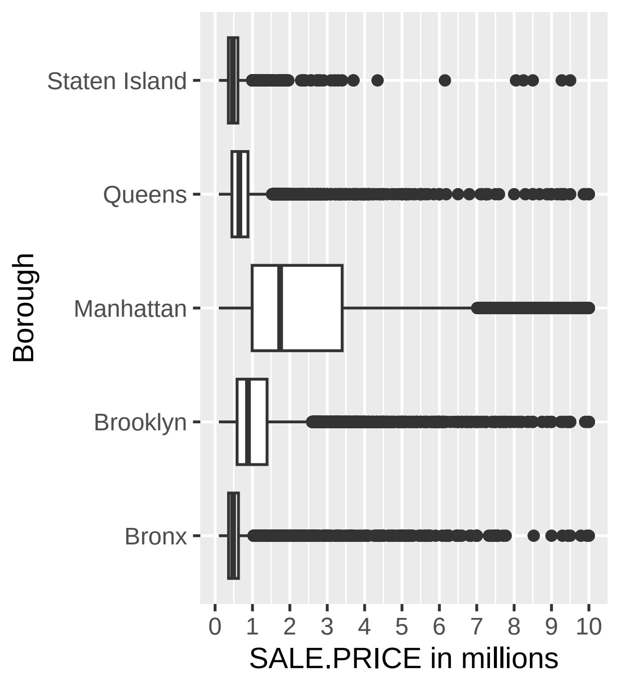


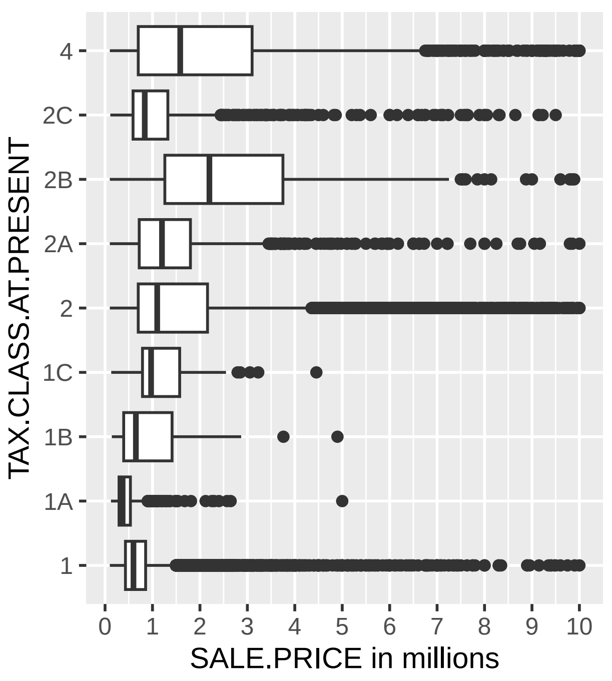
The average price for sold property is just above 1 million USD.

```
## [1] 1094765
```

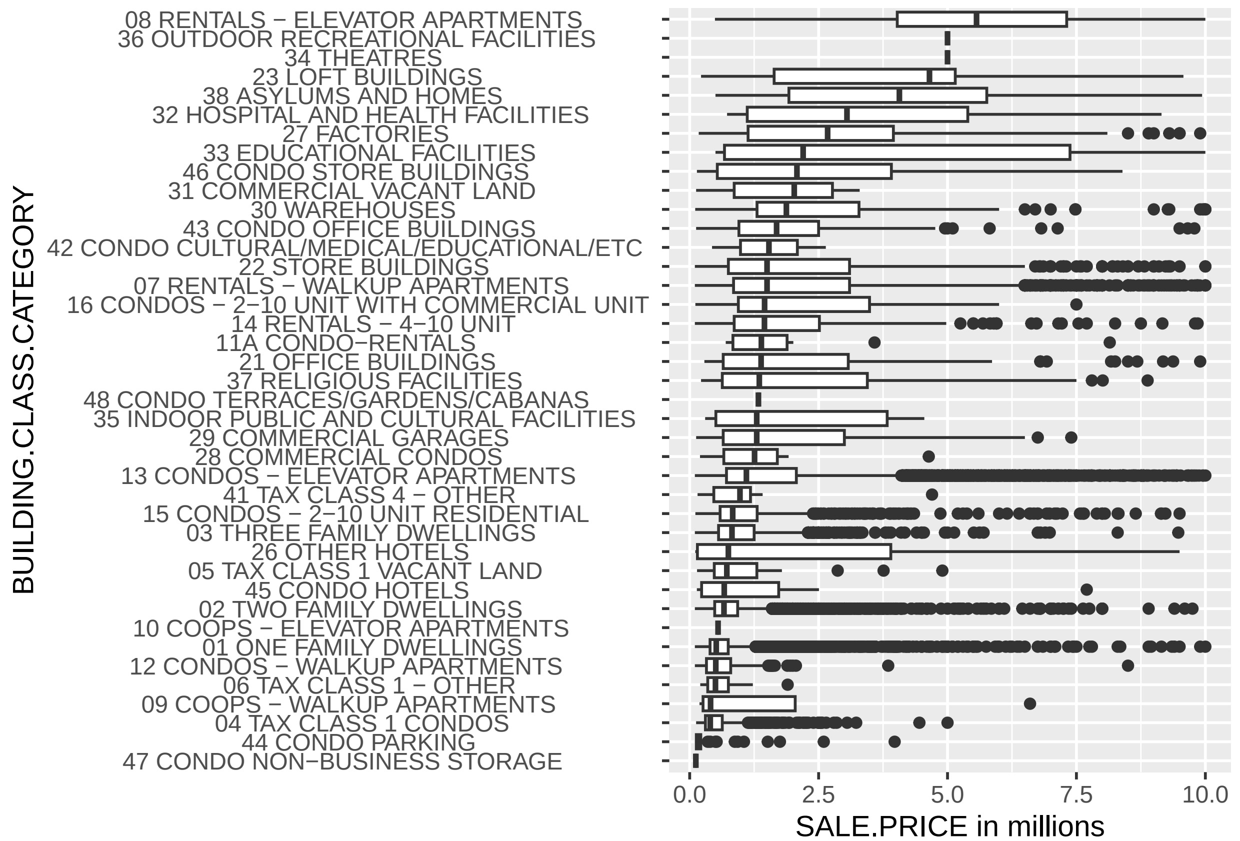
```
## [1] 700000
```

Most expensive Borough is Manhattan followed by Brooklyn. As visible on Boxplot, Manhattan has also a lot of sales with price close to 10 million.
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. Tax class 2B (Includes all other property that is primarily residential, such as cooperatives and condominiums) has highest average sale price.

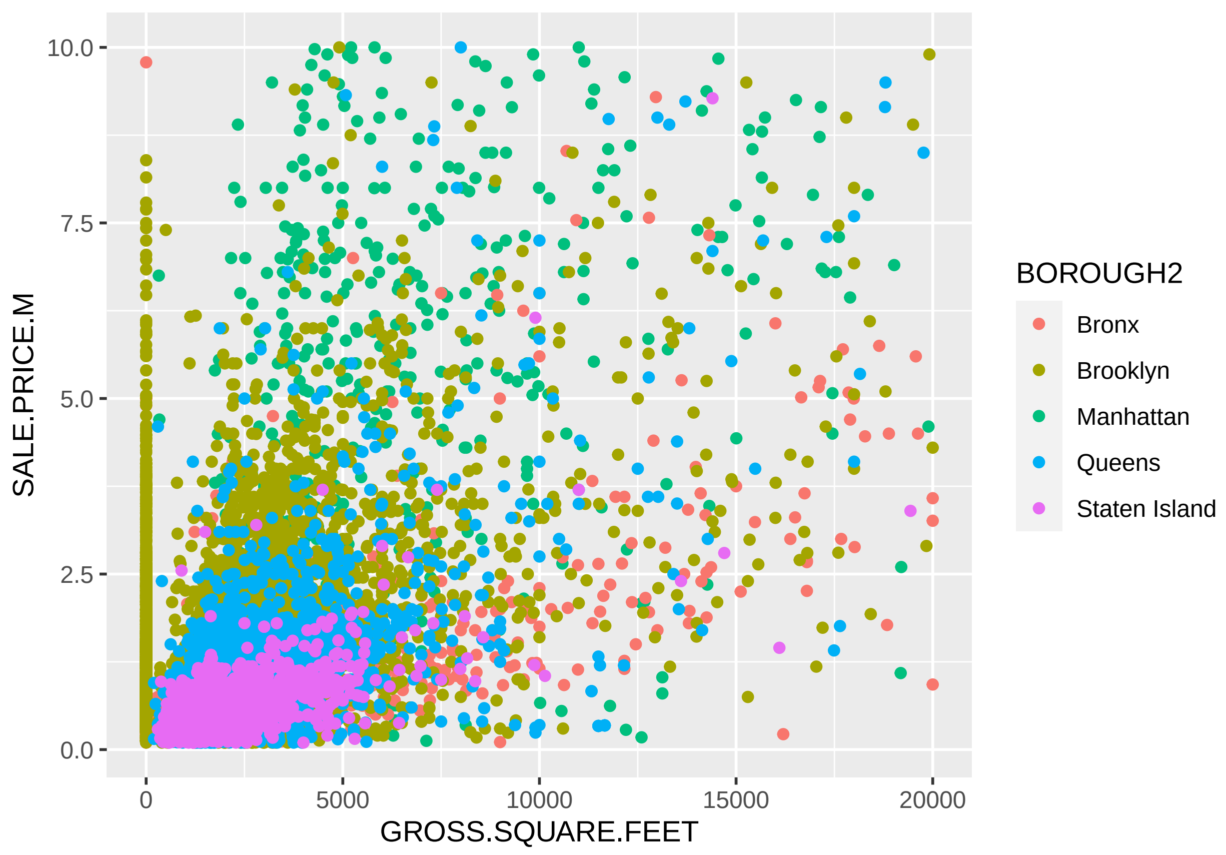




In more detailed form, Rentals-Elevator Apartments are the most expensive.



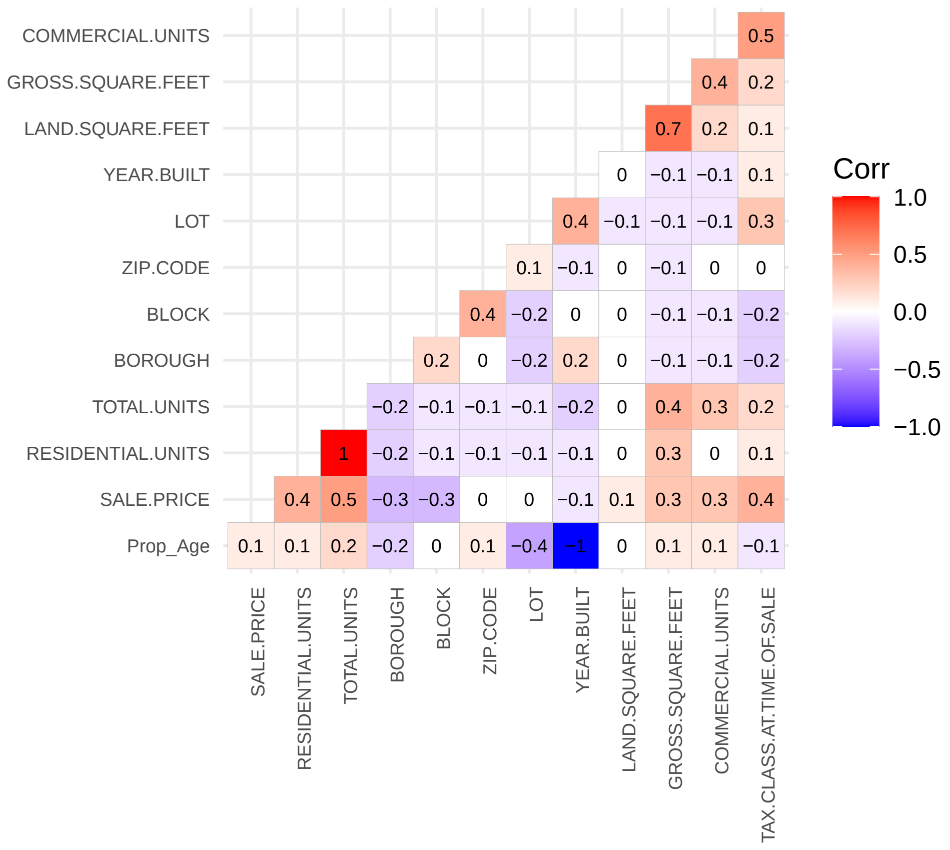
State Island (pink) has the lowest average sale prices of properties, on the other hand Manhattan has the highest (green).



Correlation Analysis: matrix and visualization

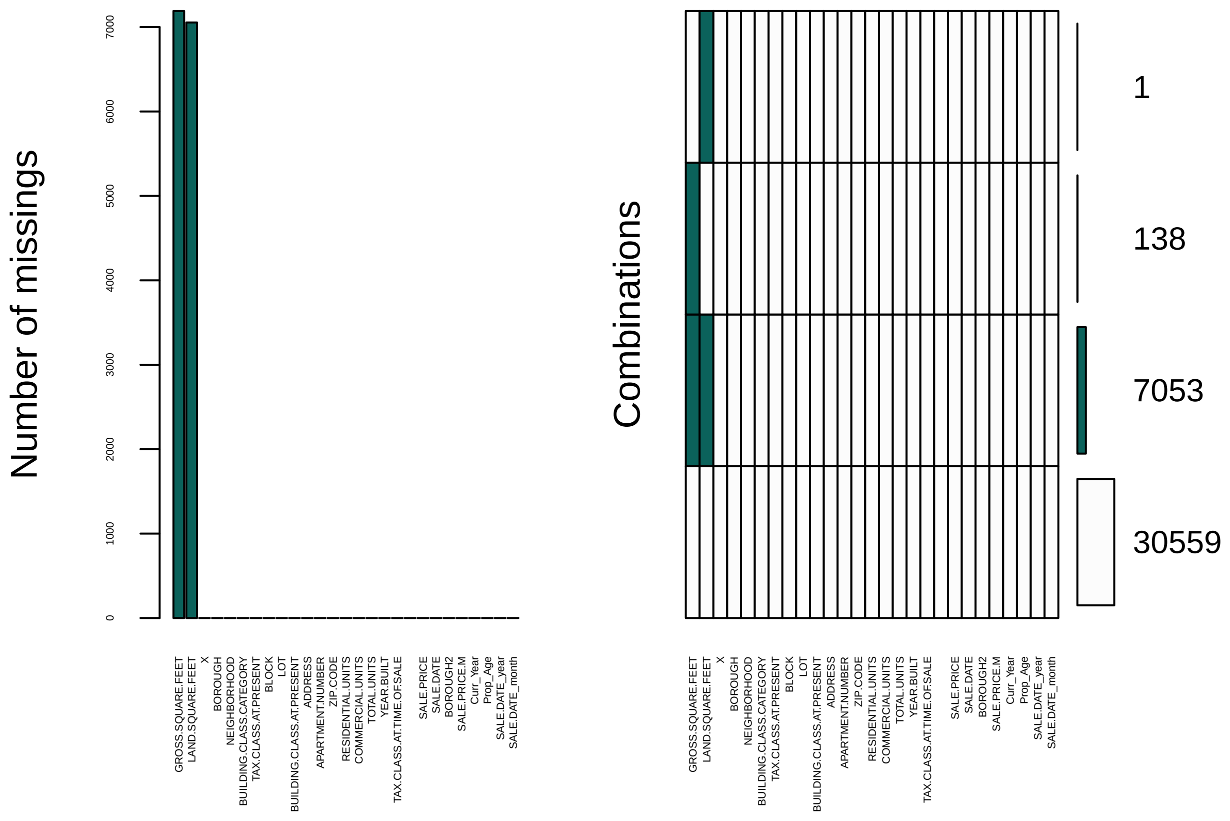
The correlation coefficient is a statistical measure of the strength of a linear relationship between two variables. Its values can range from -1 to 1. A correlation coefficient of -1 describes a perfect negative, or inverse, correlation, with values in one series rising as those in the other decline, and vice versa. A coefficient of 1 shows a perfect positive correlation, or a direct relationship. A correlation coefficient of 0 means there is no linear relationship.
We are computing a correlation matrix using variables in edited dataset. After that, we are computing a matrix of correlation p-values.

Using package "ggcorrplot" we are showing correlation coefficients in pleasing plot. Correlation coefficients vary from -1 (dark blue) to 1 (red). There is correlation that is equal to 1 between Residential Units and Total Units and no correlation higher than 0.7 except between land and gross sq feet, which is not problem because these two variables are connected.
We are mostly interested in correlation coefficients connected with variable SALE.PRICE. We can see that sale price is positively correlated (0.5) with Total units and Residental units (0.4), land and gross sq feet, number of commercial units and tax class. It is also negatively correlated with Borough (Manhattan is defined as number 1) and Block (-0.3) and Year Built.



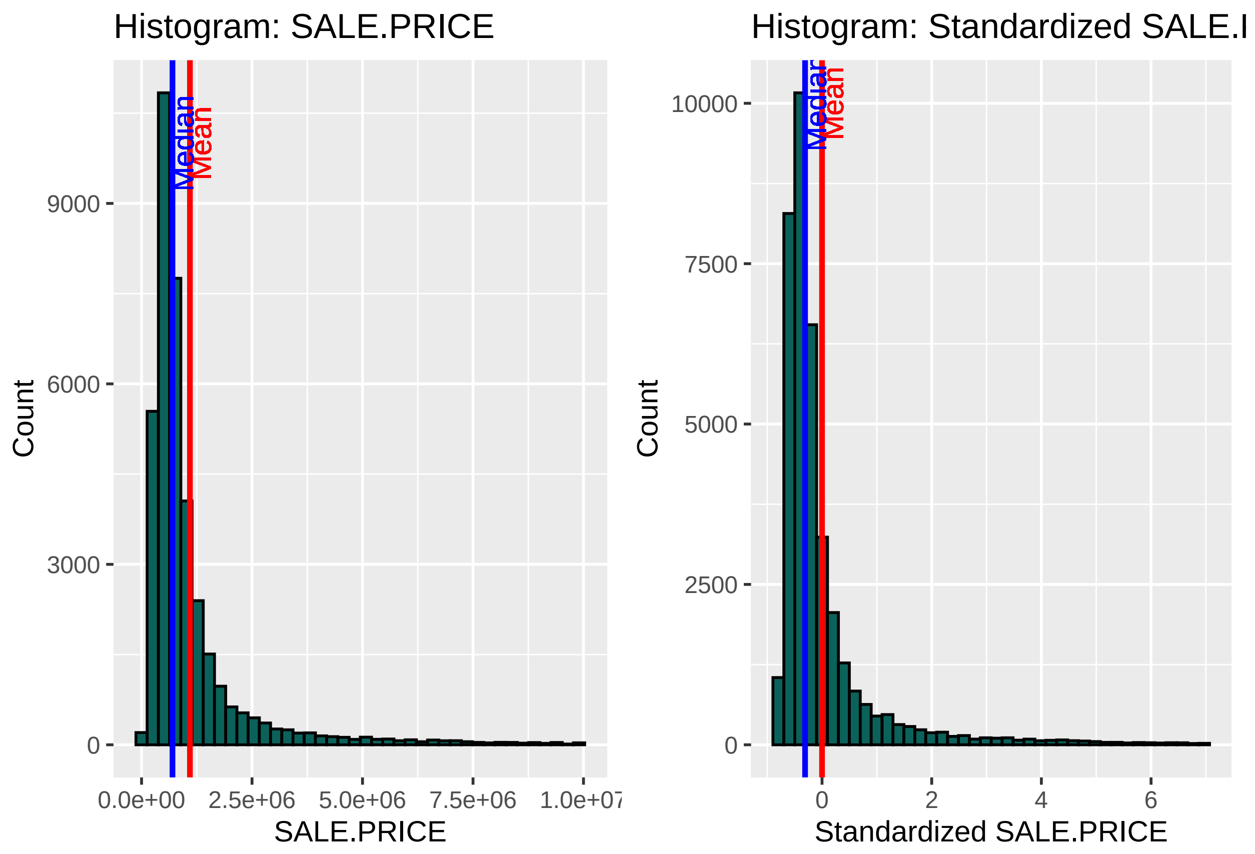
Transforming/Normalizing/Standardizing data

Given that the data have different units, structure and we need normalized data for modelling we are going to transform/regularize/normalize them. In Standard scaling, also known as Standardization of values, we scale the data values such that the overall statistical summary of every variable has a mean value of zero and an unit variance value. The main goal of data normalization is to achieve a standardized data format across dataset. Many models might perform poorly if the numeric features do not more or less follow a standard Gaussian (normal) distribution. First, we need to check for missing and NA values in variables. Most of the missing values are in variable Basement which is not that important for us for purposes of the analysis.
For variable LAND.SQUARE.FEET we have 7054 NAs and for variable GROSS.SQUARE.FEET we have 7191 NAs. We are using a linear models to predict these values.



```
##
##  Variables sorted by number of missings:
##                       Variable Count
##              GROSS.SQUARE.FEET  7191
##               LAND.SQUARE.FEET  7054
##                              X     0
##                        BOROUGH     0
##                   NEIGHBORHOOD     0
##        BUILDING.CLASS.CATEGORY     0
##           TAX.CLASS.AT.PRESENT     0
##                          BLOCK     0
##                            LOT     0
##      BUILDING.CLASS.AT.PRESENT     0
##                        ADDRESS     0
##               APARTMENT.NUMBER     0
##                       ZIP.CODE     0
##              RESIDENTIAL.UNITS     0
##               COMMERCIAL.UNITS     0
##                    TOTAL.UNITS     0
##                     YEAR.BUILT     0
##      TAX.CLASS.AT.TIME.OF.SALE     0
##  BUILDING.CLASS.AT.TIME.OF.SALE     0
##                     SALE.PRICE     0
##                      SALE.DATE     0
##                       BOROUGH2     0
##                   SALE.PRICE.M     0
##                          Curr_Year     0
##                           Prop_Age     0
##                      SALE.DATE_year     0
##                     SALE.DATE_month     0
```

Before standardization the data were one side skewed. After normalization data are no more skewed and centralized.



Modelling

We are using Random Forest (RF) Machine Learning process to develop suitable predicting algorithm by tuning the parameters and evaluating by The root mean square error (RMSE). First, we are splitting data into training (90%) and testing (10%) set and removing unnecessary variables for the purpose of the analysis. We are training models on training dataset and the best performing model is used to predict on test dataset.

Our dependent variable is standardized sale price of the property in millions (s_SALE.PRICE.M). We want to see how we can describe variability in dependent variable and predict using independent variables:

```
##  [1] "BOROUGH"                  "BLOCK"
##  [3] "LOT"                      "ZIP.CODE"
##  [5] "YEAR.BUILT"               "TAX.CLASS.AT.TIME.OF.SALE"
##  [7] "SALE.DATE_year"           "SALE.DATE_month"
##  [9] "s_LAND.SQUARE.FEET"       "s_GROSS.SQUARE.FEET"
## [11] "s_RESIDENTIAL.UNITS"      "s_COMMERCIAL.UNITS"
## [13] "s_TOTAL.UNITS"            "s_SALE.PRICE"
```

To evaluate the quality of the models we are using RMSE. RMSE allows us to measure how far predicted values are from observed values in a regression analysis. The larger the difference indicates a larger gap between the predicted and observed values, which means poor regression model fit. In the same way, the smaller RMSE that indicates the better the model. Based on RMSE we can compare the two different models with each other and be able to identify which model fits the data better.

```
## # A tibble: 0 x 0
```

Model: using only average (mean) of values

Simplest possible prediction and model is using mean of the dependent variable. We predict the same sale price (mean) for all properties sold.

```
## # A tibble: 1 x 2
##   Method       RMSE
##   <chr>       <dbl>
## 1 Model: Mean  1.00
```

Model: Random Forest with default settings

Random Forest in R Programming is an ensemble of decision trees. It builds and combines multiple decision trees to get more accurate predictions. It's a non-linear classification algorithm. Random Forest takes random samples from the observations, random initial variables(columns) and tries to build a model. We are applying it on training set.

```
# Mean of squared residuals
rf_model_1_MSE
```

```
## [1] 0.2703717
```

```
# % Var explained
rf_model_1_var_expl
```

```
## [1] 73.05
```

The mean of squared residuals and % variance explained indicate how well the model fits the data. Residuals are a difference between prediction and the actual value. We can increase or decrease the number of trees (ntree) or the number of variables tried at each split (mtry) and see whether the residuals or % variance change. Default settings are: ntree is set to 500 and default mtry is set to 4.

```
rf_model_1_RMSE_default
```

```
## [1] 0.5199728
```

As first improvement, We are choosing number of trees with lowest mean squared error (MSE). In this case its 485 trees with MSE of 0.2703717. The RMSE of default model with chosen trees is:

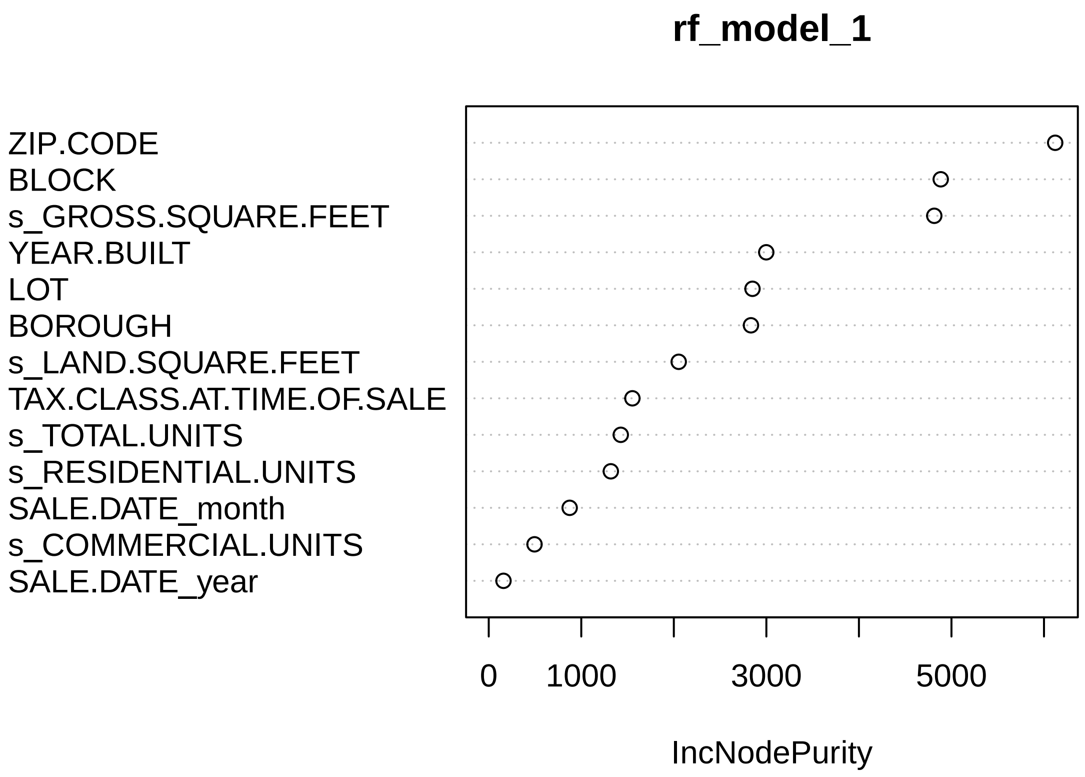
```
rf_model_1_RMSE_chosen_trees
```

```
## [1] 0.5198547
```

Comparison table:

```
## # A tibble: 3 x 4
##   Method                                    RMSE Trees  mTry
##   <chr>                                    <dbl> <dbl> <dbl>
## 1 Model: Mean                              1.00     NA    NA
## 2 Model 1: RF with default settings        0.520   500     4
## 3 Model 1: RF with chosen trees (lowest MSE) 0.520   485     4
```

Here we can see our variables by importance (using default RF model). It looks like ZIP.CODE and BLOCK (connected) with GROSS.SQUARE.FEET are the most important variables in our model followed by YEAR.BUILT and LOT. Node purity is measured by Gini Index which is the the difference between RSS before and after the split on that variable.

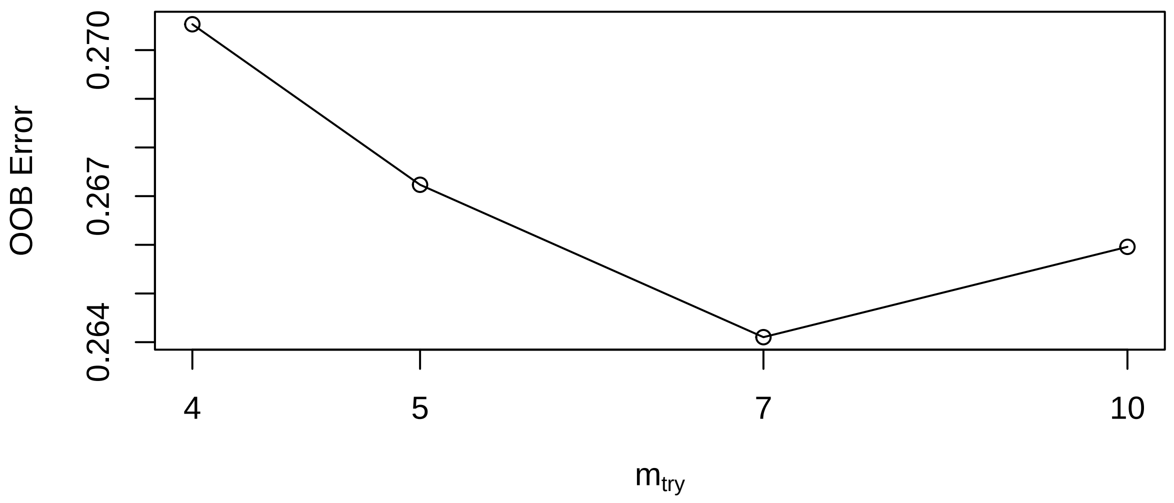


```
##                          IncNodePurity
## BOROUGH                      2833.3202
## BLOCK                        4884.2378
## LOT                          2849.8935
## ZIP.CODE                     6121.0065
## YEAR.BUILT                   2998.7080
## TAX.CLASS.AT.TIME.OF.SALE     1552.1277
## SALE.DATE_year                 159.3443
## SALE.DATE_month                874.5138
## s_LAND.SQUARE.FEET            2052.8795
## s_GROSS.SQUARE.FEET           4814.1872
## s_RESIDENTIAL.UNITS           1319.0245
## s_COMMERCIAL.UNITS             494.8146
## s_TOTAL.UNITS                 1426.7173
```

Model: Random Forest with tuneRF tuning

We want enough trees (ntree) to stabalize the error but using too many trees is unncessarily inefficient, especially when using large data sets. Mtry is the number of variables selected at each split is denoted by mtry in randomforest function. We are finding the optimal mtry value and selecting mtry value with minimum out of bag(OOB) error. Tuning starts with mtry = 5 and increasing by a factor of 1.5 until the OOB error stops improving by 1%.
Mtry with lowest OOBError of 0.2640916 is 7.

```
## mtry = 5  OOB error = 0.2672336
## Searching left ...
## mtry = 4     OOB error = 0.2705308
## -0.01233829 0.01
## Searching right ...
## mtry = 7     OOB error = 0.2641028
## 0.0117155 0.01
## mtry = 10    OOB error = 0.2659582
## -0.007025362 0.01
```



```
##    mtry  OOBError
## 4     4 0.2705308
## 5      5 0.2672336
## 7      7 0.2641028
## 10    10 0.2659582
```

After finding the best performing value of mtry=7 we are applying it on default RF model with ntree=500 and mtry=7.

```
# Mean of squared residuals
rf_model_2_MSE
```

```
## [1] 0.2651559
```

```
# % Var explained
rf_model_2_var_expl
```

```
## [1] 73.57
```

The RMSE of RF model with tuneRF mtry of 7 and ntre=500:

```
rf_model_2_RMSE_tuneRF
```

```
## [1] 0.5149329
```

Next, we are applying tuneRF mtry=7 and chosen ntree=485 from default model with lowest MSE to finding out if it can improve our RMSE even more.

```
# Mean of squared residuals
rf_model_2_chosen_trees_MSE
```

```
## [1] 0.2648387
```

```
# % Var explained
rf_model_2_chosen_trees_var_expl
```

```
## [1] 73.61
```

The RMSE of RF model with tuneRF mtry=7 and ntre=485:

```
rf_model_2_chosen_trees_RMSE_tuneRF
```

```
## [1] 0.5146248
```

Updated comparison table:

```
## # A tibble: 5 x 4
##   Method                                       RMSE Trees  mTry
##   <chr>                                       <dbl> <dbl> <dbl>
## 1 Model: Mean                                 1.00     NA    NA
## 2 Model 1: RF with default settings           0.520   500     4
## 3 Model 1: RF with chosen trees (lowest MSE)  0.520   485     4
## 4 Model 2: RF with tuneRF mTry                0.515   500     7
## 5 Model 2: RF with tuneRF mTry and chosen trees 0.515   485     7
```

Model: Random Forest with Grid Search tuning

Another method is to define a grid of algorithm parameters to try (search). Each axis of the grid is an algorithm parameter, and points in the grid are specific combinations of parameters. Because we are only tuning one parameter, the grid search is a linear search through a vector of candidate values. We are using XGboost Regression tree model (package).
RMSE is being used to select the optimal model using the smallest value.

The values used for the optimal model were nrounds = 100, max_depth = 7, eta = 0.3, gamma = 0, colsample_bytree = 0.6, min_child_weight = 1 and subsample = 1. We are using values nrounds = 100 and mtry=7 in the following model.

```
# Mean of squared residuals
rf_model_3_grid_MSE
```

```
## [1] 0.2703607
```

```
# % Var explained
rf_model_3_grid_var_expl
```

```
## [1] 73.06
```

The RMSE of RF model with Grid Search tuning (XGboost) with values of mtry=7 and ntre=100:

```
rf_model_3_chosen_trees_RMSE_grid
```

```
## [1] 0.5199622
```

```
## # A tibble: 6 x 4
##   Method                                      RMSE Trees  mTry
##   <chr>                                      <dbl> <dbl> <dbl>
## 1 Model: Mean                                1.00     NA    NA
## 2 Model 1: RF with default settings          0.520   500     4
## 3 Model 1: RF with chosen trees (lowest MSE)  0.520   485     4
## 4 Model 2: RF with tuneRF mTry               0.515   500     7
## 5 Model 2: RF with tuneRF mTry and chosen trees 0.515   485     7
## 6 Model 3: RF with grid mTry and chosen trees   0.520   100     7
```

Final model and predicting

After tuning and comparing all models based on their RMSE we are choosing RF model with tuneRF chosen mtry=7 and number of trees ntree=485 with lowest MSE and applying it on final test data.

```
# RF with default settings
rf_model_1_RMSE_default
```

```
## [1] 0.5199728
```

```
# RF with chosen trees (lowest MSE)
rf_model_1_RMSE_chosen_trees
```

```
## [1] 0.5198547
```

```
# RF mtry value by tuneRF
rf_model_2_RMSE_tuneRF
```

```
## [1] 0.5149329
```

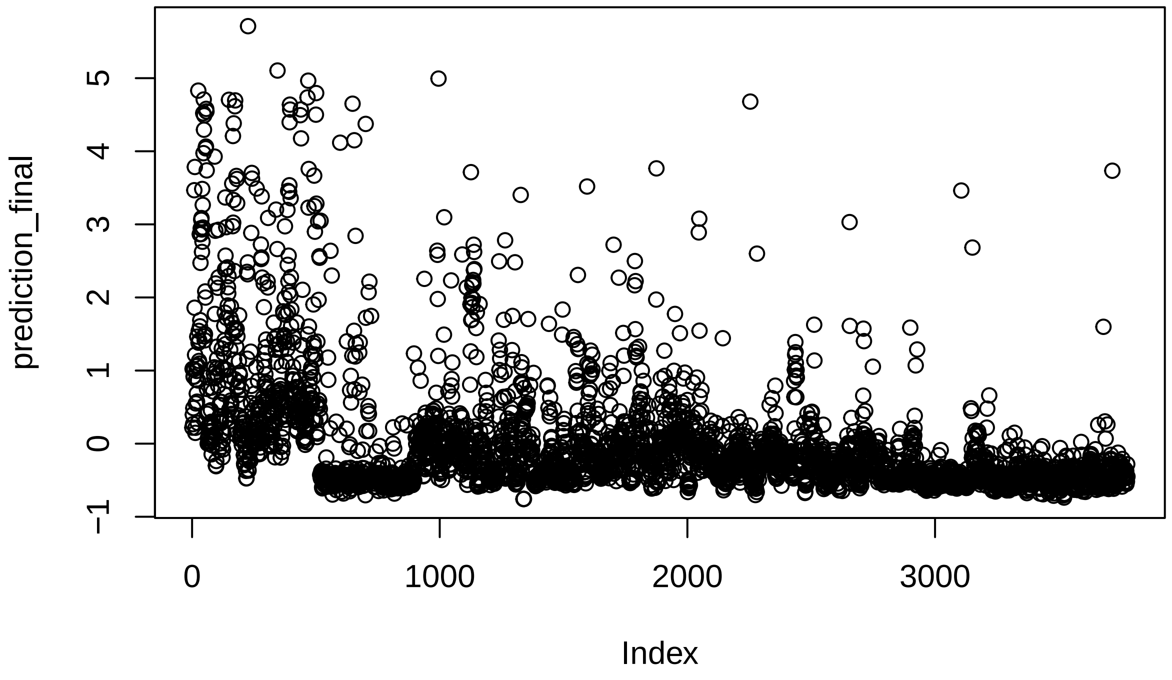
```
# RF mtry value by tuneRF and chosen trees
rf_model_2_chosen_trees_RMSE_tuneRF
```

```
## [1] 0.5146248
```

```
# RF tuned by grid search
rf_model_3_chosen_trees_RMSE_grid
```

```
## [1] 0.5199622
```

We are using it to predict sale price (standardized) of the property on a test dataset and showing it using plot.



RMSE is a measure of how spread out residuals are. In other words, it tells us how concentrated the data is around the line of best fit. Final RMSE of best performing model:

```
## [1] 0.5032866
```

Possible next work

The analysis can be extended by for example creating loop for most suitable value of mtry (fine tuning parameters of Random Forest model), Full grid search with ranger package, tuning more parameters or choosing new or deleting current predictors. All calculations and tunings require a lot of time to compute.

References

- Irizarry RA. Introduction to Data Science : Data Analysis and Prediction Algorithms with R. BC Press. Leanpub; 2019.
- Kaggle. "NYC Property Sales" dataset. Available at https://www.kaggle.com/datasets/new-york-city/nyc-property-sales/download?datasetVersionNumber=1
- JASON FERNANDO. Investopedia. The Correlation Coefficient: What It Is, What It Tells Investors. Available at https://www.investopedia.com/terms/c/correlationcoefficient.asp