# NYC Property Sales - Harvard EdX Data Science Capstone

Mary Lampmann

12/29/2020

**Introduction**

The City of New York has made a dataset of all Property Sales transactions in the city for the twelve month period from September 2016 to September 2017 available for data analysis (Source:Kaggle.com).

#Kaggle webpage * [NYC Property Sales]"https://www.kaggle.com/new-york-city/nyc-property-sales"

#Github link for download of Kaggle csv file "https://raw.githubusercontent.com/MLamp20/NYCProps/main/nyc-rolling-sales.csv"

This report documents the application of machine learning (ML) techniques to the variables in that dataset in order to predict the Sale Price of each property as accurately as possible. This report will document my analysis and present my findings with supporting statistics and figures.

The goal of this project is the training of multiple ML algorithms which will use inputs from a subset of the data to predict Sale Price in a separate subset of that same data, determining the algorithm that does so with the lowest residual mean squared error (RMSE). I will be standardizing the data, both to eliminate skew in the Sale Price and to modify the dataset for use with multiple types of machine learning algorithms, so the target number for the RMSE will be the lowest RMSE below the standard deviation of 1.0.

**Project Key steps**

1. Preprocess the dataset, inclusive of:

a. Renaming or creation of variables
b. Class conversions: Character to Numeric, Character to Time
c. Removal or replacement of blanks, NA, and zeroes as appropriate
d. Removal of duplicated lines

2. Conduct exploratory data analysis (EDA) on the training set **nyc** for use in machine learning (ML) model development, as well as additional dataset modification based on EDA learnings, inclusive of:

a. Log transform and scale variables to remove skew identified in EDA
b. Remove extreme outliers from dataset
c. Conversion of variables to factors
d. One-hot conversion of factors and remaining character classes
e. Eliminate variables with lowest correlation to Sale Price to streamline model processing times

3. After EDA, partition the **nyc** training dataset to allow separate training (*train*) and test sets (*test*) for use in evaluation of ML models

4. Iteratively train different machine learning algorithms on the dataset to elicit the lowest RMSE results, identifying the final recommended process

**Methods/Analysis**

**Exploratory Data Analysis** New York City itself is a big place. The dataset, **nyc**, is comprised of **84548** rows and **22** columns, and includes distinct records for **5** boroughs (1-Manhattan, 2-Bronx, 3-Brooklyn, 4-Queens, 5-Staten Island), **254** neighborhoods, **11566** blocks, **47** building class categories, and at the point of sale, **166** building classes. Additional variables in the dataset include the address, apartment number (where applicable), zip code, parcel lot number, residential, commercial and total units in the property, square footage (gross and for the land parcel lot), year built, tax class of the property, sale price, and date of sale.

**Preprocessing** The first step in preprocessing is simplification of 16 of the 22 variable names for ease of reading, both in the removal of spaces and abbreviation of certain terms.

Evaluation of the distinct values for each variable is the next step, which immediately identifies a column **EASEMENT** with no values. We will extract this from the dataset in a future step.

```
  ADDRESS    X1 BLOCK SALE_PX LAND_SF GRSS_SF APT_NO  LOT SALE_DATE
1   67563 26736 11566   10008    6062    5691   3988 2627       364
  NEIGHBORHOOD TTL_UNITS ZIP RESID_UNITS BLDG_CL_0917 BLDG_CL_SOLD BUILT
1          254       192 186         176          166          166   158
  COMM_UNITS BLDG_CL_CATEGORY TAX_CL_0917 BOROUGH TAX_CL_SOLD EASEMENT
1         55               47          10       5           4        0
```

Next up: a review of the percentage of zero values in each vector. Biggest issues exist in the UNITS variables (a notable 94% of the commercial units (COMM_UNITS)), the SF variables (gross square foot (GRSS_SF) and land square foot (LAND_SF)), and in Sale Price (SALE_PX). We will have to remove records with zero Sale Price, but will look to replace other zero values with mean values in future steps.

```
 COMM_UNITS RESID_UNITS   TTL_UNITS     GRSS_SF     LAND_SF     SALE_PX
         94          29           23          14          12          12
      BUILT         ZIP
          8           1
```

Potentially more impactful to our dataset than zeroes are the percentage of NA values. Surprisingly, a review of the NA values identifies only 4 vectors with NA values. We see that 77% of Apartment Number vector (**APT_NO**) values are NAs, suggesting minimal value in using that variable in our model building. We will look to replace NAs in "at present"(**0917**) values of the tax class and building classes of the properties with the values of those classes at the point of sale.

```
    EASEMENT       APT_NO  TAX_CL_0917 BLDG_CL_0917
         100           77            1            1
```

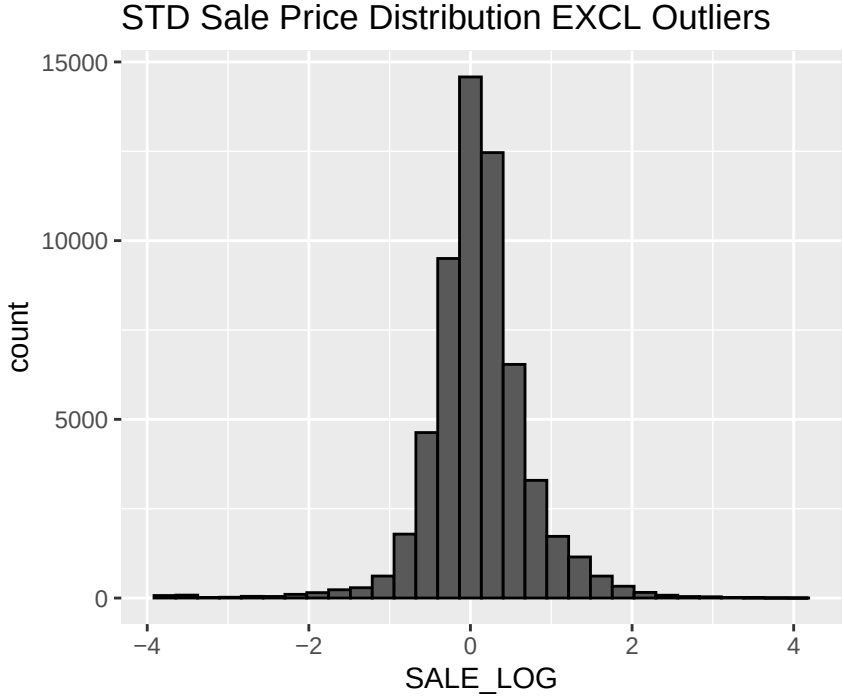
The original dataset consisted of character, numeric, and logical vectors. Next step in our preprocessing is conversion of character vectors that represent numbers to numeric vectors, or, in the case of date of sale (SALE_DATE), to a time vector.

Once we have completed that step, we conduct our replacements of zero and NA values, and remove records where the sale price is NA, blank, or zero. We have also identified duplicated lines in the dataset, so will remove those, the EASEMENT column noted above, and then begin some data visualization.

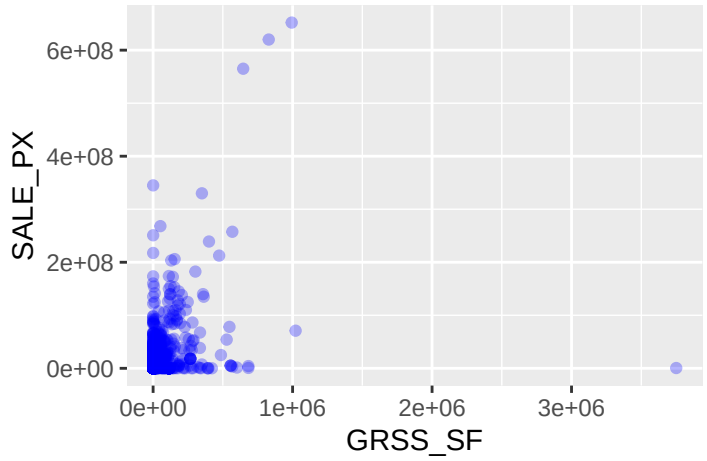
With completion of the above, our revised dataset is now **59592 rows** and **21 columns** in dimension.

**Data Visualization** We start with a review of the distribution of our Sale Prices (SALE_PX). We have a pretty wide range of Sale Prices, starting at $1 and maxing out at $2.21 Billion, so we will first review this via a log 10 transformation of the plot.


The data is clearly skewed with a long tail. Next step: log transform and then standardize the sale price. Once complete, I elected to remove the records that have an absolute scaled Sale Log value of >4, which leaves us with the following plot and the variable SALE LOG, which represents the log transformed and scaled Sale Prices:

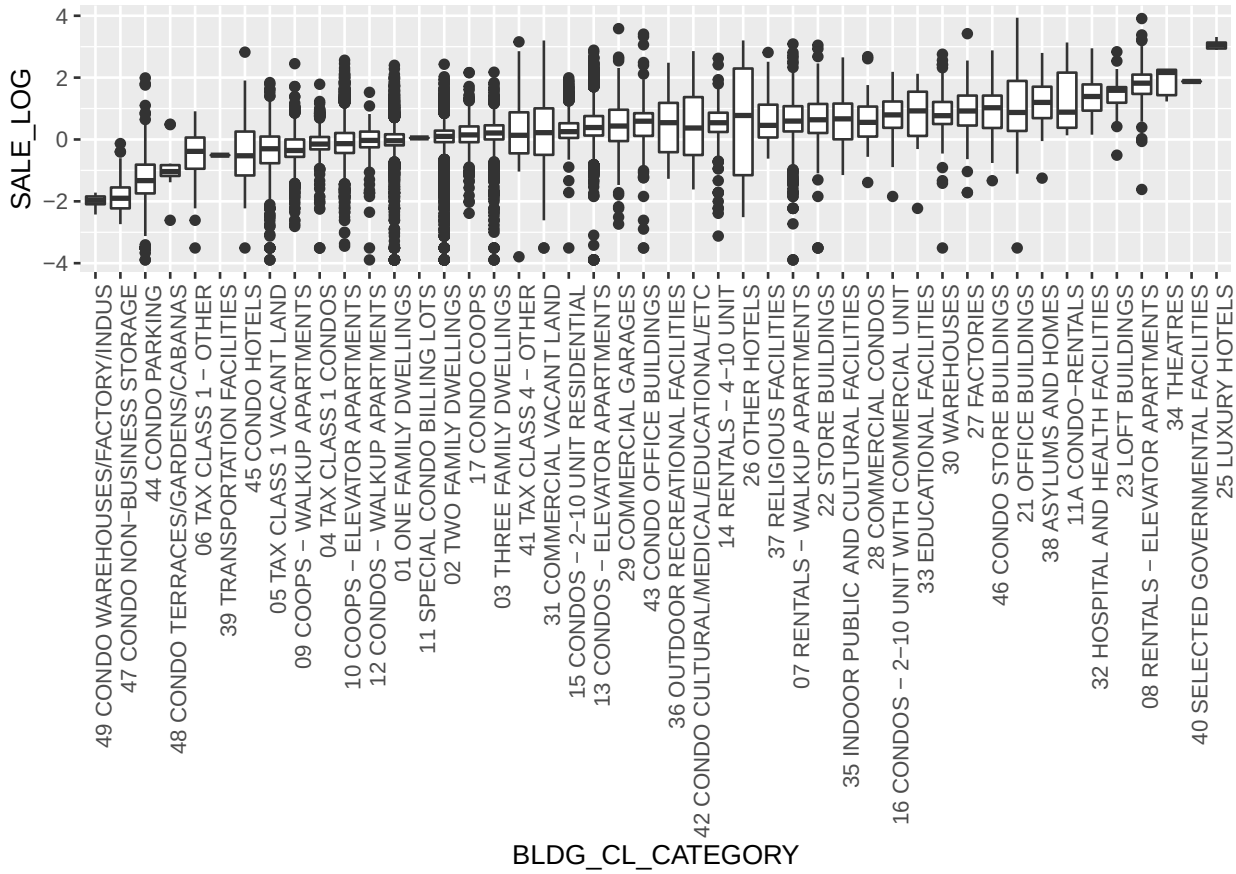


Square Footage is a metric that is often discussed in the context of sale prices, here's the comparison of the gross square footage (GRSS_SF) of our data set with the Sale Price:



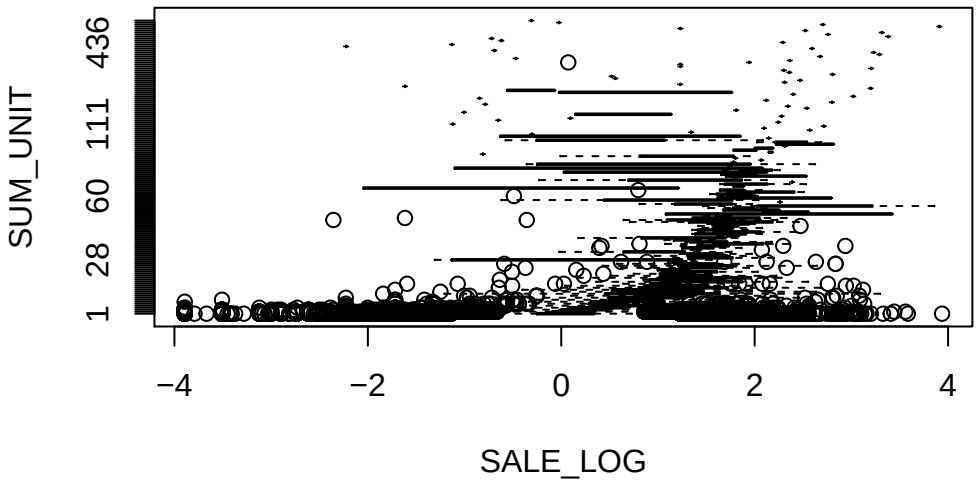

The values are tightly clustered, and we will remove that GRSS_SF extreme outlier on the far right in a future step.

**Sales Price Boxplot** Evaluating sales prices by category, we see Condo Warehouses/Factory with the lowest average Sales Price, and Luxury Hotels with prices at the top of the group.

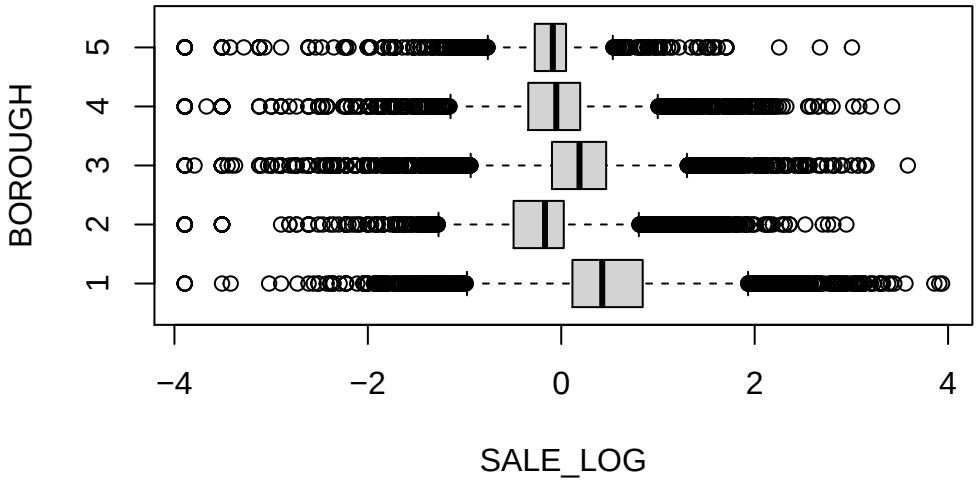



Although there appears to be some linearity in the 18-50 SUM UNIT counts, the sale price averages for the

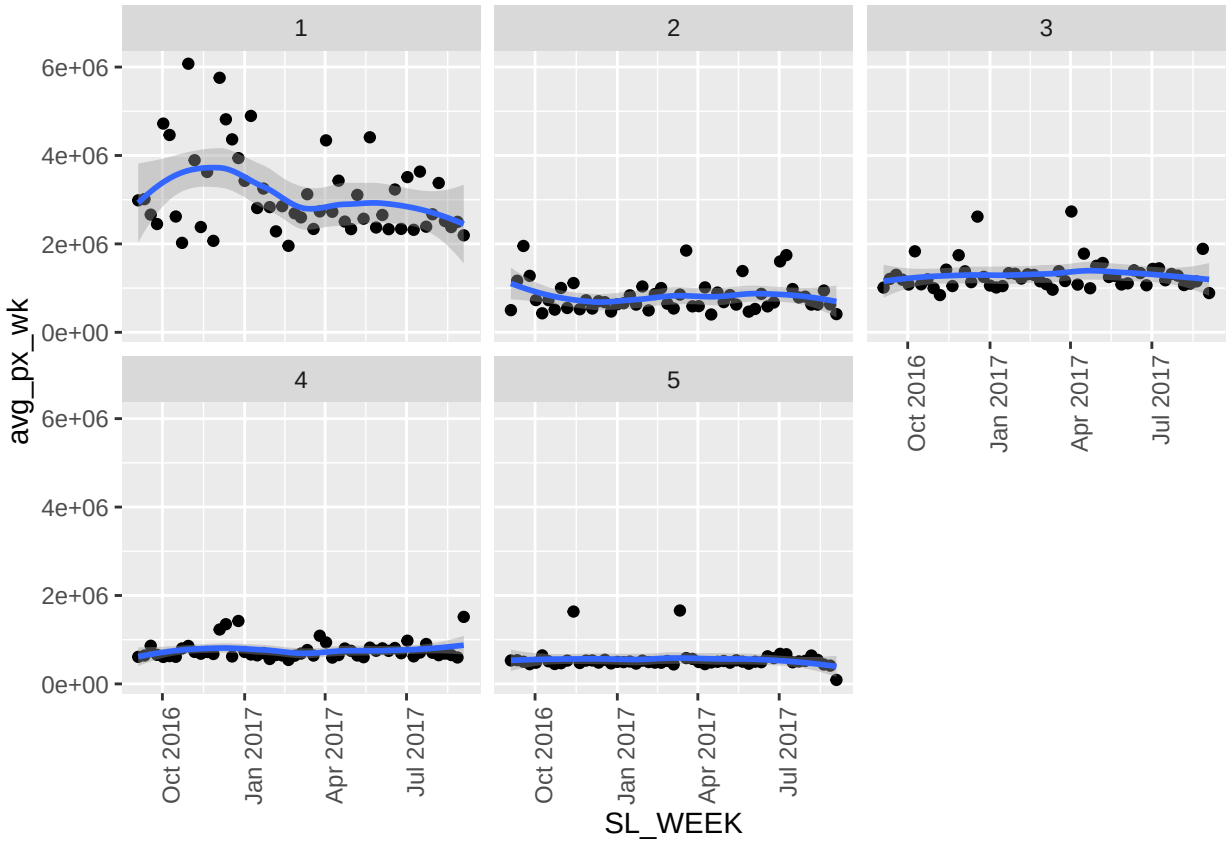
sum of units are notably variable in the 50-100 unit range.




In sale price by Borough, we see that Borough 1, Manhattan has both the highest average price and the largest interquartile range (IQR). Borough 2, the Bronx has the lowest average, and Borough 5, Staten Island has the smallest interquartile range.




**Time Effect on Sale Price by Borough** The 5 boroughs were faceted for analysis as to whether time played a significant role in Sale Price. Although there was some time effect in Borough #1, Manhattan, most notable in the October-January time period, the other 4 boroughs average sale prices were remarkably stable across the weeks of the dataset. Considering this, time will not be incorporated into the ML model.




We've reviewed visualizations on sale price vs. some of the variables, but now let's evaluate the count of sales transactions ($n$), neighborhoods ($nbrhd$), blocks ($blk$), building class categories ($categ$), and average sale price per gross square footage ($avg\_pxgsf$). Note that $avg\_px\_K$ is an expresssion of the Sale Price in Thousands.

| | BOROUGH | n | nbrhd | categ | blk | avg_px_K | avg_pxgsf |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 14243 | 39 | 39 | 1286 | 3147 | 1332 |
| 2 | 2 | 5037 | 38 | 34 | 1732 | 826 | 214 |
| 3 | 3 | 15349 | 60 | 43 | 4714 | 1307 | 607 |
| 4 | 4 | 18115 | 59 | 39 | 6590 | 751 | 395 |
| 5 | 5 | 5868 | 58 | 27 | 2238 | 555 | 273 |

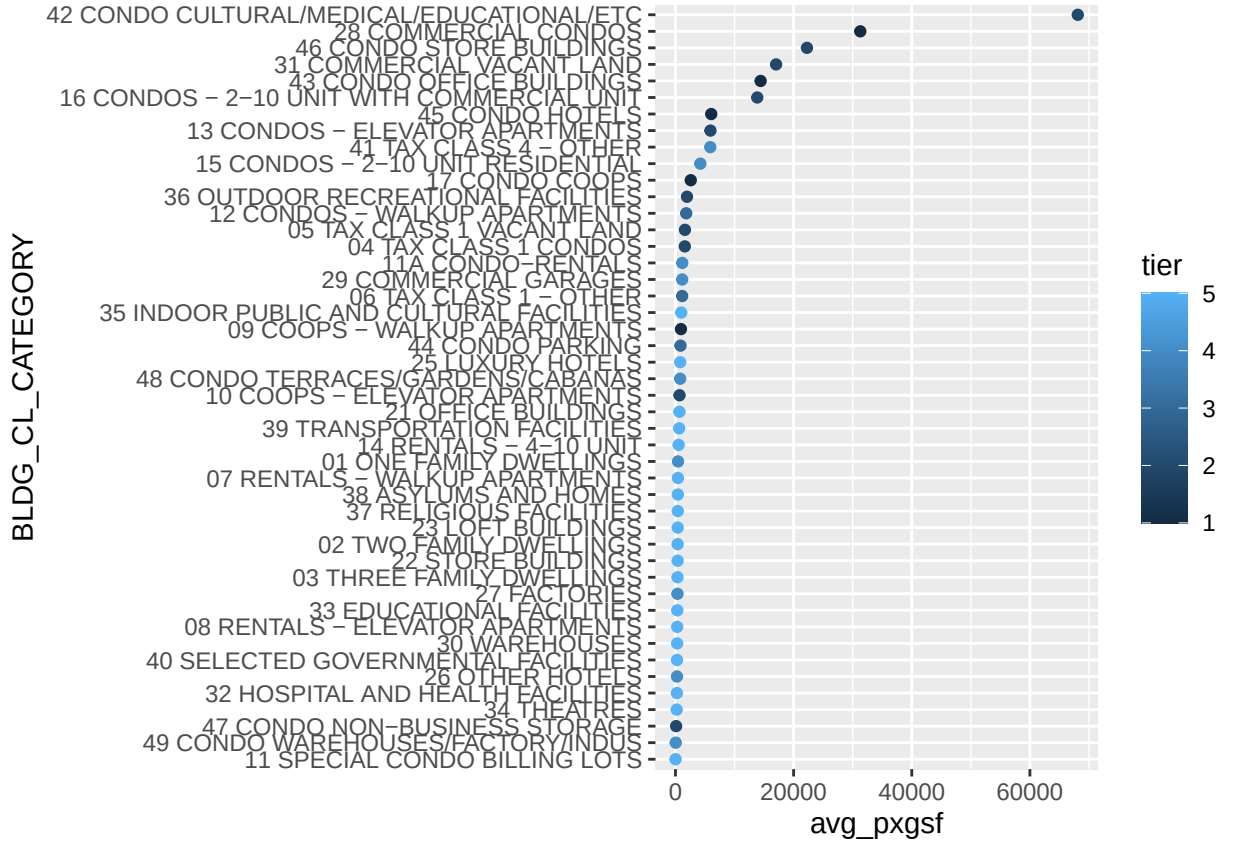
The greatest number of sales transactions came from Boroughs 4,3, and 1. The number of transactions in Borough 4 and 3 are the largest, but both Boroughs rank at the top in number of blocks, and theoretically, have the largest inventory of properties. Borough #1, Manhattan, had the highest average price per gross square footage of property sold at $1332.

**Building Class Categories** Slicing the information by building class category, we see that the top 10 categories by number of transactions ($n$) are comprised of family dwellings, apartments, and condos. These categories represent 92% of the transactions in our entire dataset. The categories in lines 1-8 are present in each of the 5 boroughs.

| | BLDG_CL_CATEGORY | n | boro | nbrhd | blk | avg_px_K | avg_pxgsf |
|---|---|---|---|---|---|---|---|
| 1 | 01 ONE FAMILY DWELLINGS | 12701 | 5 | 219 | 6317 | 670 | 434 |
| 2 | 10 COOPS - ELEVATOR APARTMENTS | 11513 | 5 | 141 | 1426 | 801 | 690 |
| 3 | 13 CONDOS - ELEVATOR APARTMENTS | 10262 | 5 | 129 | 1117 | 2100 | 5922 |
| 4 | 02 TWO FAMILY DWELLINGS | 9883 | 5 | 215 | 5474 | 802 | 375 |
| 5 | 09 COOPS - WALKUP APARTMENTS | 2503 | 5 | 114 | 699 | 494 | 931 |
| 6 | 03 THREE FAMILY DWELLINGS | 2322 | 5 | 164 | 1715 | 1008 | 349 |
| 7 | 07 RENTALS - WALKUP APARTMENTS | 1742 | 5 | 171 | 1290 | 3444 | 413 |
| 8 | 04 TAX CLASS 1 CONDOS | 1248 | 5 | 111 | 401 | 561 | 1582 |
| 9 | 17 CONDO COOPS | 1111 | 4 | 47 | 175 | 995 | 2592 |
| 10 | 15 CONDOS - 2-10 UNIT RESIDENTIAL | 1033 | 3 | 72 | 431 | 1495 | 4216 |

We do see some notable swings in average sale price per square foot here, but in further analysis, note that most of the condo categories had large percentages of NAs for the gross square footage that we use in this calculation, and it is likely that in using the average SF for all categories, our assigned values are too low for the condo categories specifically.

The categories have been assigned tiers corresponding to the percentage of GRSS_SF records that were NA, with tier 1 representing 75+% NA, tier 2 50-74%, tier 3 25-49%, tier 4 1-24%, and tier 5 with no NA in GRSS_SF.




If we rework our sort and examine the highest average price per gross SF (avg_pxgsf), we clearly see that those same condos where we have replaced 50%+ of GRSS_SF have extremely high avg_pxgsf averages. In a future iteration of this analysis, we may want to replace the average square footage assignments we have made for NAs to the average for each specific building class category.

| | BLDG_CL_CATEGORY | n | avg_pxgsf | na_pct | tier |
|---|---|---|---|---|---|
| 1 | 42 CONDO CULTURAL/MEDICAL/EDUCATIONAL/ETC | 4 | 68108 | 0.60 | 2 |
| 2 | 28 COMMERCIAL CONDOS | 14 | 31292 | 0.79 | 1 |
| 3 | 46 CONDO STORE BUILDINGS | 74 | 22256 | 0.70 | 2 |
| 4 | 31 COMMERCIAL VACANT LAND | 187 | 17039 | 0.54 | 2 |
| 5 | 43 CONDO OFFICE BUILDINGS | 241 | 14419 | 0.87 | 1 |
| 6 | 16 CONDOS - 2-10 UNIT WITH COMMERCIAL UNIT | 38 | 13847 | 0.61 | 2 |
| 7 | 45 CONDO HOTELS | 76 | 6058 | 0.98 | 1 |
| 8 | 13 CONDOS - ELEVATOR APARTMENTS | 10262 | 5922 | 0.74 | 2 |
| 9 | 41 TAX CLASS 4 - OTHER | 55 | 5893 | 0.23 | 4 |
| 10 | 15 CONDOS - 2-10 UNIT RESIDENTIAL | 1033 | 4216 | 0.25 | 4 |

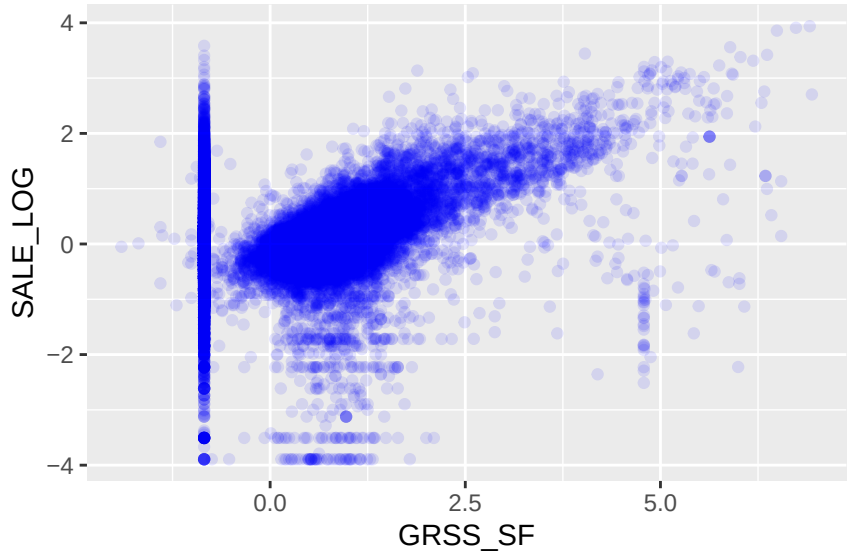
Resorting the categories by total US dollars sold, the top 10 categories here represent almost 81% of the total dollar sales in our dataset. The distortion on the average price per square foot as a result of the assigned GRSS_SF value for condo categories continues to be notable on this chart.

| | BLDG_CL_CATEGORY | TTL_SLS_K | avg_pxgsf |
|---|---|---|---|
| 1 | 13 CONDOS - ELEVATOR APARTMENTS | 21552283 | 5923 |
| 2 | 10 COOPS - ELEVATOR APARTMENTS | 9226653 | 690 |
| 3 | 01 ONE FAMILY DWELLINGS | 8505244 | 434 |
| 4 | 02 TWO FAMILY DWELLINGS | 7925747 | 375 |
| 5 | 07 RENTALS - WALKUP APARTMENTS | 5999432 | 413 |
| 6 | 21 OFFICE BUILDINGS | 5464142 | 672 |
| 7 | 08 RENTALS - ELEVATOR APARTMENTS | 4910787 | 294 |
| 8 | 03 THREE FAMILY DWELLINGS | 2340114 | 349 |
| 9 | 22 STORE BUILDINGS | 1975404 | 374 |
| 10 | 15 CONDOS - 2-10 UNIT RESIDENTIAL | 1544434 | 4216 |

Categories that make up the bottom of the list generally appear to be non “living” categories - warehouses, transportation hubs, parking, and storage.

| | BLDG_CL_CATEGORY | TTL_SLS_K | avg_pxgsf |
|---|---|---|---|
| 1 | 49 CONDO WAREHOUSES/FACTORY/INDUS | 164 | 46 |
| 2 | 39 TRANSPORTATION FACILITIES | 220 | 620 |
| 3 | 11 SPECIAL CONDO BILLING LOTS | 600 | 20 |
| 4 | 47 CONDO NON-BUSINESS STORAGE | 2730 | 110 |
| 5 | 48 CONDO TERRACES/GARDENS/CABANAS | 3414 | 802 |
| 6 | 40 SELECTED GOVERNMENTAL FACILITIES | 16000 | 261 |
| 7 | 06 TAX CLASS 1 - OTHER | 25116 | 1135 |
| 8 | 36 OUTDOOR RECREATIONAL FACILITIES | 56946 | 1952 |
| 9 | 44 CONDO PARKING | 83483 | 856 |
| 10 | 42 CONDO CULTURAL/MEDICAL/EDUCATIONAL/ETC | 96603 | 68107 |

Given our learnings thus far, we are ready to move on to log transformation and standardization of the square footage and units vectors. One more outlier trim on gross square footage, and here is our revised plot of that vs. Sale Price, demonstrating a linear relationship:




Last steps before moving onto partitioning will be conversion of Borough and Tax Class Sold to factors, and then One Hot Encoding of both vectors as well as all the remaining character vectors we want to consider in our model.

Here are the final list of variables that we will use in our model, and their correlation with the Sale Price (logged and scaled, SALE_LOG). Note that GRSS_SF with 19.3% correlation to SALE LOG is not the highest correlated variable. The highest correlation instead is whether the property is in Borough.1, Manhattan.

| | term | SALE_LOG |
|---|---|---|
| 1 | BOROUGH.1 | 0.366 |
| 2 | BOROUGH.2 | -0.162 |
| 3 | BOROUGH.4 | -0.201 |
| 4 | BOROUGH.5 | -0.135 |
| 5 | NEIGHBORHOODCIVIC.CENTER | 0.126 |
| 6 | NEIGHBORHOODTRIBECA | 0.120 |
| 7 | BLDG_CL_CATEGORY01.ONE.FAMILY.DWELLINGS | -0.118 |
| 8 | BLDG_CL_CATEGORY07.RENTALS...WALKUP.APARTMENTS | 0.154 |
| 9 | BLDG_CL_CATEGORY08.RENTALS...ELEVATOR.APARTMENTS | 0.161 |
| 10 | BLDG_CL_CATEGORY09.COOPS...WALKUP.APARTMENTS | -0.132 |
| 11 | BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS | -0.153 |
| 12 | BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS | 0.258 |
| 13 | BLDG_CL_CATEGORY44.CONDO.PARKING | -0.169 |
| 14 | BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE | -0.110 |
| 15 | GRSS_SF | 0.193 |
| 16 | TAX_CL_SOLD.1 | -0.143 |
| 17 | TAX_CL_SOLD.2 | 0.113 |
| 18 | BLDG_CL_SOLDC1 | 0.123 |
| 19 | BLDG_CL_SOLDC6 | -0.132 |
| 20 | BLDG_CL_SOLDC7 | 0.105 |
| 21 | BLDG_CL_SOLDD1 | 0.106 |
| 22 | BLDG_CL_SOLDD4 | -0.161 |
| 23 | BLDG_CL_SOLDRG | -0.149 |
| 24 | SUM_UNIT | 0.234 |

### Model Construction Methods

**Split nyc into train and test sets** The next step in the machine learning algorithm creation was the split of the **nyc** data set into training and testing subsets, *train* and *test* respectively.

I chose to use an 80/20 split on the partitioning based on the Pareto principle, and because I wanted to leave a testing set of sufficient size for the successful testing of a variety of machine learning algorithms.

I used R's RMSE function to calculate the RMSE, as well as a data frame that would detail the RMSE values associated with different machine learning algorithms.

```
#Test set will be 20% of data as set up for multiple algorithm uses later
set.seed(1,sample.kind = "Rounding")
test_index<-createDataPartition(y=nyc_pre_split$SALE_LOG,times=1, p=0.2, list=FALSE)
train<-nyc_pre_split[-test_index,]
test<-nyc_pre_split[test_index,]
```

**The model fittings machine learning algorithms trained:** Next steps were the selection of different machine learning algorithms for model fittings.

**Linear Model** The plot we viewed earlier with gross square footage (GRSS_SF) and Sale Price (SALE_LOG) indicated a strong linear relationship, so the logical place to start our machine learning exercise is with simple linear regression.

```
model<-lm(SALE_LOG ~ ., data = train)
print(tidy(model),n=Inf)
```

```
# A tibble: 25 x 5
   term                                    estimate std.error statistic   p.value
   <chr>                                      <dbl>     <dbl>     <dbl>     <dbl>
 1 (Intercept)                                0.386    0.0126     30.6   2.22e-203
 2 BOROUGH.1                                  0.422    0.00680    62.1   0.
 3 BOROUGH.2                                 -0.414    0.00852   -48.5   0.
 4 BOROUGH.4                                 -0.172    0.00586   -29.4   3.32e-188
 5 BOROUGH.5                                 -0.299    0.00852   -35.0   1.95e-265
 6 NEIGHBORHOODCIVIC.CENTER                   0.578    0.0309     18.7   1.41e- 77
 7 NEIGHBORHOODTRIBECA                        0.231    0.0232      9.95  2.59e- 23
 8 BLDG_CL_CATEGORY01.ONE.FAMILY.DWELLIN~    -0.0588   0.00897    -6.56  5.51e- 11
 9 BLDG_CL_CATEGORY07.RENTALS...WALKUP.A~    -0.0510   0.0207     -2.47  1.36e-  2
10 BLDG_CL_CATEGORY08.RENTALS...ELEVATOR~     0.713    0.0573     12.4   1.61e- 35
11 BLDG_CL_CATEGORY09.COOPS...WALKUP.APA~     0.0412   0.190       0.217 8.28e-  1
12 BLDG_CL_CATEGORY10.COOPS...ELEVATOR.A~     0.129    0.0377      3.42  6.35e-  4
13 BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.~     0.205    0.0107     19.2   6.28e- 82
14 BLDG_CL_CATEGORY44.CONDO.PARKING          -1.76     0.0746    -23.6   3.62e-122
15 BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS~    -2.23     0.0641    -34.8   1.10e-262
16 GRSS_SF                                    0.258    0.00540    47.8   0.
17 TAX_CL_SOLD.1                             -0.328    0.0128    -25.5   1.09e-142
18 TAX_CL_SOLD.2                             -0.166    0.0162    -10.2   1.54e- 24
19 BLDG_CL_SOLDC1                             0.445    0.0310     14.4   1.14e- 46
20 BLDG_CL_SOLDC6                            -0.383    0.190      -2.02  4.38e-  2
21 BLDG_CL_SOLDC7                             0.492    0.0412     11.9   9.03e- 33
22 BLDG_CL_SOLDD1                             0.0937   0.0716      1.31  1.90e-  1
```

```
23 BLDG_CL_SOLDD4                          -0.362    0.0368     -9.82  9.39e- 23
24 BLDG_CL_SOLDRG                           0.323    0.0809      3.99  6.67e-  5
25 SUM_UNIT                                -0.0510   0.00572    -8.91  5.12e- 19
```

```
as.data.frame(varImp(model))%>%arrange(desc(Overall))
```

```
                                                Overall
BOROUGH.1                                        62.055
BOROUGH.2                                        48.529
GRSS_SF                                          47.841
BOROUGH.5                                        35.033
BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE    34.847
BOROUGH.4                                        29.395
TAX_CL_SOLD.1                                    25.521
BLDG_CL_CATEGORY44.CONDO.PARKING                 23.575
BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS  19.210
NEIGHBORHOODCIVIC.CENTER                         18.679
BLDG_CL_SOLDC1                                   14.361
BLDG_CL_CATEGORY08.RENTALS...ELEVATOR.APARTMENTS 12.449
BLDG_CL_SOLDC7                                   11.932
TAX_CL_SOLD.2                                    10.230
NEIGHBORHOODTRIBECA                               9.953
BLDG_CL_SOLDD4                                    9.823
SUM_UNIT                                          8.914
BLDG_CL_CATEGORY01.ONE.FAMILY.DWELLINGS           6.558
BLDG_CL_SOLDRG                                    3.988
BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS    3.416
BLDG_CL_CATEGORY07.RENTALS...WALKUP.APARTMENTS    2.467
BLDG_CL_SOLDC6                                    2.016
BLDG_CL_SOLDD1                                    1.309
BLDG_CL_CATEGORY09.COOPS...WALKUP.APARTMENTS      0.217
```

```
rmse_train_lm<-summary(model)$sigma

y_hat_lm<-predict.lm(model,newdata = test)
rmse_lm<-RMSE(y_hat_lm,test$SALE_LOG)

rmse_result <- data_frame(Method = "Linear Regression Predictor",RMSEtrain = rmse_train_lm,
                          RMSEtest = rmse_lm)
rmse_result%>%knitr::kable()
```

| Method | RMSEtrain | RMSEtest |
|---|---|---|
| Linear Regression Predictor | 0.464 | 0.475 |

The linear model generated an RMSE on the training set of .464 and on the test set of .475. The top 5 variables on this model in importance were BOROUGH.1 (Manhattan), BOROUGH.2 (Bronx), GRSS_SF, BOROUGH.5 (Staten Island), and BLDG_CL_CATEGORY47.CONDO.NON-BUSINESS.STORAGE.

**rpart Model** Moving onto a more complex algorithm, we next evaluate decision trees. Although the normalization/scaling that we completed on the dataset is not required for a decision tree, that step should

not have any negative impact on the *slower time to train* associated with this type of algorithm.

Typically with rpart, we have the opportunity to review the decision tree that produces the lowest RMSE results. In the case of our dataset, there are simply too many nodes and leaves to provide visual clarity, so we will instead provide the data on the points of distinction.

We have used a tuning option here on the complexity parameter (cp), the minimum benefit a split must add to the tree, and arrived at the cp of .002 as the ideal.

```
set.seed(1991,sample.kind="Rounding")

train_rpart <- train(SALE_LOG ~ ., method = "rpart",
                     tuneGrid=data.frame(cp=seq(0,0.05,0.002)),data = train)
train_rpart$bestTune
```

```
     cp
2 0.002
```

```
train_rpart$finalModel
```

```
n= 46887

node), split, n, deviance, yval
      * denotes terminal node

   1) root 46887 17600.00  0.10300
     2) BOROUGH.1< 0.5 35474 10800.00 -0.02490
       4) GRSS_SF< 0.974 28438  6970.00 -0.11400
         8) BLDG_CL_SOLDD4>=0.5 4949   617.00 -0.39700
          16) BOROUGH.2>=0.5 765   107.00 -0.60800 *
          17) BOROUGH.2< 0.5 4184   470.00 -0.35800 *
         9) BLDG_CL_SOLDD4< 0.5 23489  5880.00 -0.05450
          18) BLDG_CL_CATEGORY44.CONDO.PARKING>=0.5 217   163.00 -1.41000 *
          19) BLDG_CL_CATEGORY44.CONDO.PARKING< 0.5 23272  5310.00 -0.04190
            38) BLDG_CL_CATEGORY09.COOPS...WALKUP.APARTMENTS>=0.5 1387   225.00 -0.39800
              76) BOROUGH.4>=0.5 928    40.90 -0.50400 *
              77) BOROUGH.4< 0.5 459   153.00 -0.18400
               154) BOROUGH.2>=0.5 40    10.80 -1.15000 *
               155) BOROUGH.2< 0.5 419   101.00 -0.09110 *
            39) BLDG_CL_CATEGORY09.COOPS...WALKUP.APARTMENTS< 0.5 21885  4900.00 -0.01930
              78) BOROUGH.2>=0.5 1953   539.00 -0.27800
               156) TAX_CL_SOLD.2>=0.5 262    80.90 -0.64800 *
               157) TAX_CL_SOLD.2< 0.5 1691   417.00 -0.22100 *
              79) BOROUGH.2< 0.5 19932  4220.00  0.00601
               158) BOROUGH.5>=0.5 3908   670.00 -0.18500 *
               159) BOROUGH.5< 0.5 16024  3370.00  0.05270
                 318) BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS< 0.5 12729  2800.00  0.00715
                   636) BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE>=0.5 25     4.19 -2.07000 *
                   637) BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE< 0.5 12704  2690.00  0.01120
                    1274) GRSS_SF< 0.67 8886  1910.00 -0.03150
                      2548) BOROUGH.4>=0.5 5465  1020.00 -0.08530 *
                      2549) BOROUGH.4< 0.5 3421   853.00  0.05450 *
                    1275) GRSS_SF>=0.67 3818   721.00  0.11100 *
                 319) BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS>=0.5 3295   443.00  0.22900
```

```
                 638) BOROUGH.4>=0.5 1366   116.00  0.06970 *
                 639) BOROUGH.4< 0.5 1929   269.00  0.34100 *
        5) GRSS_SF>=0.974 7036  2650.00  0.33500
         10) GRSS_SF< 1.91 6151  1600.00  0.23100
           20) BOROUGH.2>=0.5 1036   222.00 -0.06290 *
           21) BOROUGH.2< 0.5 5115  1270.00  0.29000
             42) GRSS_SF< 1.34 3565   760.00  0.23100 *
             43) GRSS_SF>=1.34 1550   467.00  0.42700 *
         11) GRSS_SF>=1.91 885   516.00  1.06000
           22) GRSS_SF< 2.98 537   204.00  0.81900 *
           23) GRSS_SF>=2.98 348   231.00  1.44000
             46) BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS>=0.5 20     9.39 -0.40200 *
             47) BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS< 0.5 328   150.00  1.55000 *
      3) BOROUGH.1>=0.5 11413  4430.00  0.50100
        6) GRSS_SF< 1.4 10808  3430.00  0.44800
         12) BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS< 0.5 6199  1830.00  0.27600
           24) BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE>=0.5 29     7.62 -1.57000 *
           25) BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE< 0.5 6170  1720.00  0.28500
             50) BLDG_CL_SOLDC6>=0.5 594   120.00 -0.00338 *
             51) BLDG_CL_SOLDC6< 0.5 5576  1550.00  0.31600
              102) BLDG_CL_SOLDD4>=0.5 4118   905.00  0.26300 *
              103) BLDG_CL_SOLDD4< 0.5 1458   598.00  0.46300 *
         13) BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS>=0.5 4609  1170.00  0.68000
           26) NEIGHBORHOODCIVIC.CENTER< 0.5 4415  1070.00  0.65300 *
           27) NEIGHBORHOODCIVIC.CENTER>=0.5 194    28.10  1.28000 *
        7) GRSS_SF>=1.4 605   429.00  1.44000
         14) BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS>=0.5 7     4.95 -0.68400 *
         15) BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS< 0.5 598   392.00  1.47000
           30) GRSS_SF< 4.79 556   310.00  1.39000
             60) GRSS_SF>=4.76 16     2.50 -1.21000 *
             61) GRSS_SF< 4.76 540   196.00  1.46000 *
           31) GRSS_SF>=4.79 42    29.50  2.55000 *
```

```
varImp(train_rpart)
```

```
rpart variable importance

  only 20 most important variables shown (out of 24)

                                                Overall
GRSS_SF                                          100.00
BLDG_CL_SOLDD4                                    52.38
BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS    51.23
SUM_UNIT                                          39.68
BOROUGH.2                                         37.17
BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS   35.14
BOROUGH.4                                         26.80
BLDG_CL_CATEGORY08.RENTALS...ELEVATOR.APARTMENTS  20.56
TAX_CL_SOLD.1                                     19.25
TAX_CL_SOLD.2                                     16.29
BOROUGH.5                                         13.10
BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE     11.40
BLDG_CL_SOLDC6                                    10.71
BLDG_CL_CATEGORY44.CONDO.PARKING                  10.56
```

```
BLDG_CL_CATEGORY09.COOPS...WALKUP.APARTMENTS       10.55
BOROUGH.1                                          8.99
BLDG_CL_SOLDRG                                     6.00
NEIGHBORHOODTRIBECA                                4.26
NEIGHBORHOODCIVIC.CENTER                           4.17
BLDG_CL_SOLDC1                                     2.62
```

```
rmse_train_rpart<-train_rpart$results$RMSE[which.min(train_rpart$results$RMSE)]

y_hat_rpart <- predict(train_rpart, test, type = "raw")
rmse_rpart<-RMSE(y_hat_rpart,test$SALE_LOG)

rmse_result <- bind_rows(rmse_result,
                         data_frame(Method="rpart cp = .002",
                                    RMSEtrain = rmse_train_rpart, RMSEtest = rmse_rpart))

rmse_result %>% knitr::kable()
```

| Method | RMSEtrain | RMSEtest |
|---|---|---|
| Linear Regression Predictor | 0.464 | 0.475 |
| rpart cp = .002 | 0.464 | 0.461 |

The rpart model generated an RMSE on the training set of .464, and on the test set of .461. The top 5 variables in this model, in order of importance, were GRSS_SF, BLDG_CL_SOLDD4, BLDG_CL_CATEGORY10.COOPS...ELEVATOR APARTMENTS, SUM_UNIT, and BOROUGH.2 (Bronx).

**randomForest Model** Random Forest as an algorithm gives us the ability to have R process many alternative combinations of variables in order to determine the combination that delivers the lowest RMSE. This algorithm is more robust than the single decision tree rpart algorithm we just fit.

For this particular model, and in the interest of processing time, we chose 64 trees (*informed by How Many Trees in a Random Forest? Authors Thais Mayumi Oshiro,Pedro Santoro Perez,José Augusto Baranauskas*), and tuned the mtry, the number of variables randomly sampled in each tree, which produced a bestTune mtry metric of 11. The higher mtry reduces the correlation between trees in the forest, thus improving prediction accuracy, but the tuning process and the *slower time to fit* associated with random Forest make this segment the longest to run of the three models we fit. Although mtry values of 1-14 were tested, we have abbreviated the code to 9-12 in this model to minimize run time.

```
set.seed(14,sample.kind = "Rounding")

fit_rf <- train(SALE_LOG ~ ., method = "rf",
                tuneGrid = data.frame(mtry = 9:12),ntree=64, data = train)
fit_rf$bestTune
```

```
  mtry
3   11
```

```
fit_rf
```

```
Random Forest

46887 samples
   24 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 46887, 46887, 46887, 46887, 46887, 46887, ...
Resampling results across tuning parameters:

  mtry  RMSE   Rsquared  MAE
   9    0.451  0.459     0.289
  10    0.451  0.460     0.289
  11    0.450  0.460     0.289
  12    0.451  0.459     0.289

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 11.
```

```
varImp(fit_rf)
```

```
rf variable importance

  only 20 most important variables shown (out of 24)

                                                Overall
GRSS_SF                                          100.00
BOROUGH.1                                         73.54
BLDG_CL_CATEGORY13.CONDOS...ELEVATOR.APARTMENTS   21.80
SUM_UNIT                                          19.57
BLDG_CL_SOLDD4                                    15.97
BOROUGH.2                                         13.75
BLDG_CL_CATEGORY10.COOPS...ELEVATOR.APARTMENTS    11.69
BLDG_CL_CATEGORY44.CONDO.PARKING                  11.68
BLDG_CL_CATEGORY47.CONDO.NON.BUSINESS.STORAGE      7.98
BOROUGH.4                                          7.26
BOROUGH.5                                          6.54
BLDG_CL_CATEGORY09.COOPS...WALKUP.APARTMENTS       5.57
BLDG_CL_SOLDC6                                     5.33
TAX_CL_SOLD.1                                      4.63
TAX_CL_SOLD.2                                      3.96
BLDG_CL_SOLDRG                                     3.46
NEIGHBORHOODCIVIC.CENTER                           2.33
BLDG_CL_CATEGORY08.RENTALS...ELEVATOR.APARTMENTS   1.97
BLDG_CL_CATEGORY07.RENTALS...WALKUP.APARTMENTS     1.63
NEIGHBORHOODTRIBECA                                1.12
```

```
rmse_train_rf<-fit_rf$results$RMSE[which.min(fit_rf$results$RMSE)]
y_hat_rf<-predict(fit_rf, test, type = "raw")
rmse_rf<-RMSE(y_hat_rf,test$SALE_LOG)
rmse_result <- bind_rows(rmse_result,
                         data_frame(Method="randomForest ntree=64, mtry=11",
                                    RMSEtrain = rmse_train_rf,
```

```
                          RMSEtest = rmse_rf))
rmse_result %>% knitr::kable()
```

| Method | RMSEtrain | RMSEtest |
|---|---|---|
| Linear Regression Predictor | 0.464 | 0.475 |
| rpart cp = .002 | 0.464 | 0.461 |
| randomForest ntree=64, mtry=11 | 0.450 | 0.452 |

The random Forest model generated an RMSE of .450 on the training set, and .452 on the test set.

Reviewing the variable importance, we note that gross square footage of the property (*GRSS_SF*) was a variable present in all of the decision trees in the ensemble, which is interesting considering that it was not, as discussed earlier, the variable with the highest correlation to SALE LOG. Rounding out the top 5 variables, in order of importance: BOROUGH.1 (Manhattan), BLDG_CL_CATEGORY13.CONDOS... ELEVATOR.APARTMENTS, SUM_UNIT, and BLDG_CL_SOLDD4.

### Final Model and Results

**Final Model -** The machine learning model trained that resulted in the *lowest RMSE of .452* was the randomForest algorithm with mtry=11 as noted above. Of the 24 predictors that were used in the model, only the top 8 were included in more than 10% of the trees in the ensemble, and as noted above, gross square footage (**GRSS_SF**) was present in each of the models.

### Conclusion

The RMSE results of all three models fit ranged from .475 to .452, supporting our original premise that we could build a machine learning algorithm that could predict Sale Price with relative accuracy based on an evaluation of other variables in the dataset such as Borough, Square Footage, Building Class, and Building Class Category. The random Forest algorithm specifically should be the one for future application given the RMSE results.

These models should now be used to evaluate future 12 month period of New York City Property Sales. While it would be my assumption that there may be macro level trends that alter the effectiveness of this model for future periods, it would be useful to understand whether the key predictors of this dataset remain the key predictors of future time periods, even if the degree of correlation varies. Conceivably, this model, after evaluation of additional years of data, could be a key tool for real estate developers to use in predicting the return on investment over time on new property development.

### Limitations of the model, next steps

### Limitations of the model

The RMSE results here are tied to this model's fit to the specific subset of data that was partitioned in this process into the **train** set(training set) and **test** set (test set), and might not be replicated as favorably should the same model be used on future data partitions of the same **nyc** dataset.

Other limitations of the model are that it is trained on a relatively small set of 24 variables, and that the period of data is only 12 months. It serves well on this dataset in isolation. Additionally, as is true of any model, the quality of the model is contingent on the consistency and accuracy of the data therein.

**Future Work**

Interesting future work on this dataset could include:

- revising the data partitioning to allow for k fold cross validation (60% training, 20% cross validation, 20% testing hold out)
- putting the original dataset without log transformation or standardization through the model fitting process to understand the benefit of that original process in either processing time or RMSE output
- constructing an ensemble of these machine learning algorithms
- converting the dataset to a matrix focused on the combination of borough, block, and lot(commonly called a BBL), which forms a unique key for property in New York City. Block and Lot were removed from the original dataset due to low direct correlation with Sale Price, but building a separate model against BBL could include stratification by groupings of block number, and would be another intriguing analysis on this same dataset