Modeling 2016-2017 Apartment Housing Data Selling Prices in Queens, NY

Final project for Math 342W Data Science at Queens College
May 25, 2021

By Hubert Majewski

In collaboration with:
Marin Azhar
Sara Raizel Cinamon
Enoch Kim
Kennly Weerasinghe

Abstract

This report consists of several mathematical models to predict the sale price of an apartment located in Queens, New York. The models provided will utilize a dataset between February 2016 and February 2017 for a maximum sale price that is up to 1 Million dollars. The models will be a result of a Regression Tree, Ordinary Least Squares (OLS), and a Random Forest algorithm. All models provided will consist of the most influential features given the dataset which have different predictive powers. The Random Forest model has the most predictive power out of the other two models generated.

1. Introduction

New York City (NYC) is one of the densest places in the world. Real estate services such as Zillow attempt to provide an estimation for the sale price in dollars of houses and apartments for anywhere in NYC. However, such estimations, known as “zestimates” are often lackluster; Zillow sale price estimations do not apply well to the Queens borough. By utilizing data harvested with Amazon’s Mechanical Turk (MTurk) from the Multiple Listing Service of Long Island (MLSLI), this report will provide several mathematical models that can predict the sale price of a Queens apartment.

A model is understood as a reflection of reality. Mathematical models are a kind of model that is expressed as a function of quantitative data. The quantitative dataset is used to explain a phenomenon. Because models are only a reflection of reality, they will consist of error. Most commonly, error due to ignorance of the true causal features that influence the phenomenon. Such errors can be minimized by processing the dataset ($\mathbb{D}$) through an algorithm (A). Therefore, even with precisely defined features, there needs to be another parameter, a hypothesis set ($\mathcal{H}$), that will allow A to a create model for prediction. Through a Regression Tree, Ordinary Least Squares (OLS), and a Random Forest algorithm, this report will create three models for predicting the sale price of an apartment in Queens, NY.

2. The Dataset

The dataset used in this report was obtained through Amazon’s MTurk from MLSLI. This is a raw dataset from the system that consists of 2230 tuples and 55 attributes. The attributes are either of the following data types: character, logical, or numerical. The population of interest that this dataset is relevant to are any sellers of a co-op or condo in Queens, New York; the sellers set the sale price of an apartment. Raw datasets are flawed by nature and therefore consist of missing data in specific cells that can be difficult to use when predicting. There are several reasons for such missing data. This can be due to the population of interest not specifying specific attributes intentionally to raise sale price (NMAR), or missing data can occur at random (MAR). It is not fully representative of the raw dataset as many attributes, including those we want to model, are missing from the dataset. Such a dubious population can also provide outlier tuples. Examples of these tuples include typos in the dataset, different yet synonymous values, and often incorrect/inaccurate values such as a representation for yes to be “y”. Because the population of interest is dynamic, the scope that the attributes define changes. Therefore, extrapolation becomes very risky for a dataset that does not follow strict rules and can change dynamically especially when there is an economy involved; an economic crisis such as the events that occurred in 2008 will have a severe impact on the sale prices of the apartments that violate extrapolation in the future.

2.2. Featurization

Trivial attributes such as columns created during the process of creating the dataset from MLSLI are removed. As Amazon’s MTurk provided a raw dataset from their system, it consists of various cells that are not defined for many tuples. All cells that do not have a value defined will be imputed through Miss Forest; any tuple that does not have a corresponding prediction variable will be removed as it does not provide any additional information about the corresponding sale price. The finalized, not imputed, cleaned dataset consists of 528 tuples and 30 features. All features have a respective dummy column associated with corresponding missingness. This allows for tracking which features have been imputed after the miss forest algorithm and additionally models if the removal of the features has an influence on the sale price.

The 16 selected features that are going to be used to model the outcome variable (y), sale_price, are cats_allowed, dogs_allowed, garage_exists, coop_condo, fule_type, dining_room_type, kitchen_type, walk_score, zip_codes, num_bedrooms, num_full_bathrooms, num_half_bathrooms, num_total_rooms, sq_footage, approx_year_built, and total_cost. The first 3 features are logical; 1 describes that the name of the attribute is present. For example, if the cats_allowed has a 1 in its tuple, then cats are allowed in the apartment. The same is applied for dogs which describes if dogs are allowed in the apartment. The garage_exists feature describes if the apartment comes with a garage. The other features are categorical with the exception of total_cost.

The categorical features have been factorized as well. The coop_condo feature consists of two (binary) categories, one is a "co-op" and the other is a "condo". The fule_type feature consists of 4 levels, they are "gas", "oil", "other", or "electric". The dining_room_type feature has categories consisting of "combo", "formal", "other", and "dining area". The kitchen_type feature has the following categories: "eat-in", "efficiency", or "combo". The walk_score was originally a numeric value between 0 and 100. However, by using the walk score definitions provided by Redfin, it is converted into the following categorical variables from largest to smallest "Walker's Paradise", "Very Walkable", "Somewhat Walkable", "Car-Mostly-Dependent", "Car-Dependent" (Walk Score, n.d.). The zip codes were grouped by districts as "Northeast Queens", "North Queens", "Central Queens", "Jamacia", "Northwest Queens", "West Central Queens", "Southeast Queens", "Southwest Queens" and "West Queens".

The final features are numerical they describe the apartment directly which by correlation also describes the sale price. Num_bedrooms describes the number of bedrooms within an apartment. Num_full_bathrooms describes the number of complete bathrooms which is a bathroom that consists of a sink, bathtub, shower, and a toilet. Num_half_bathrooms describes the total number of half-bathrooms that only consist of a sink and a toilet. If there are any undefined values for a half bathroom, it will be assumed that an apartment will consist of 0 half bathrooms (Evans, 2017). Num_total_rooms describes the number of rooms in the apartment. Sq_footage describes the total square footage of all rooms and bathrooms that the apartment consists of. Approx_year_build describes the approximate year that the building finished construction. The total cost is the overall monthly cost that occurs for the apartment.

Cost features are very important to look at as they behave differently for a type of apartment. Firstly, for a condo, the condoCharges attribute was computing by summating the monthly charges that occur. Any maintenance_cost that was not defined was replaced with a value of 0 and then is computed in the common_charges attribute for an apartment. As total_taxes are provided yearly, they are divided by 12 and summed with the common_charges that take place. The total_cost is computed using the sum of the maintenance_cost and the condoCharges attributes for each apartment building. The total_cost feature reflects the monthly individual costs for both a co-op and a condo apartment (Crook, 2019).

```
-- Data Summary ------------------------
                           Values
Name                       housing2
Number of rows             528
Number of columns          30
_______________________
Column type frequency:
  factor                   9
  numeric                  21
________________________
Group variables            None

-- Variable type: factor ------------------------------------------------------------------------------------
# A tibble: 9 x 6
  skim_variable    n_missing complete_rate ordered n_unique top_counts
* <chr>                <int>         <dbl> <lgl>      <int> <chr>
1 cats_allowed             0         1     FALSE          2 0: 285, 1: 243
2 coop_condo               0         1     FALSE          2 co-: 399, con: 129
3 dining_room_type       120         0.773 FALSE          4 com: 241, for: 116, oth: 49, din: 2
4 dogs_allowed             0         1     FALSE          2 0: 381, 1: 147
5 fuel_type               24         0.955 FALSE          4 gas: 301, oil: 180, oth: 12, ele: 11
6 garage_exists            0         1     FALSE          2 0: 434, 1: 94
7 kitchen_type             7         0.987 FALSE          3 eff: 231, eat: 209, com: 81
8 walk_score               0         1     TRUE           5 Ver: 237, Wal: 219, Som: 61, Car: 9
9 zip_codes                0         1     FALSE          9 Nor: 113, Wes: 93, Nor: 72, Wes: 69

-- Variable type: numeric -----------------------------------------------------------------------------------
# A tibble: 21 x 11
   skim_variable              n_missing complete_rate       mean        sd    p0    p25    p50    p75   p100 hist
 * <chr>                          <int>         <dbl>      <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <chr>
 1 approx_year_built_missing          0         1       0.0114     0.106       0      0      0      0      1 ▇▁▁▁▁
 2 cats_allowed_missing               0         1       0          0           0      0      0      0      0 ▁▁▇▁▁
 3 dining_room_type_missing           0         1       0.227      0.419       0      0      0      0      1 ▇▁▁▁▂
 4 fuel_type_missing                  0         1       0.0455     0.208       0      0      0      0      1 ▇▁▁▁▁
 5 kitchen_type_missing               0         1       0.0133     0.114       0      0      0      0      1 ▇▁▁▁▁
 6 maintenance_cost_missing           0         1       0.0398     0.196       0      0      0      0      1 ▇▁▁▁▁
 7 num_bedrooms_missing               0         1       0          0           0      0      0      0      0 ▁▁▇▁▁
 8 num_total_rooms_missing            0         1       0          0           0      0      0      0      0 ▁▁▇▁▁
 9 sale_price_missing                 0         1       0          0           0      0      0      0      0 ▁▁▇▁▁
10 sq_footage_missing                 0         1       0.597      0.491       0      0      1      1      1 ▆▁▁▁▇
11 zip_codes_missing                  0         1       0          0           0      0      0      0      0 ▁▁▇▁▁
12 condoCharges_missing               0         1       0.0189     0.136       0      0      0      0      1 ▇▁▁▁▁
13 approx_year_built                  6         0.989   1962.       20.6    1915   1950   1957   1968   2016 ▁▇▆▁▁
14 maintenance_cost                  21         0.960    626.      482.        0    387    670    827   4659 ▇▂▁▁▁
15 num_bedrooms                       0         1          1.54      0.748     0      1      1      2      3 ▁▇▁▇▁
16 num_full_bathrooms                 0         1          1.20      0.422     1      1      1      1      3 ▇▁▂▁▁
17 num_half_bathrooms                 0         1          0.0587    0.243     0      0      0      0      2 ▇▁▁▁▁
18 num_total_rooms                    0         1          4.02      1.20      1      3      4      5      8 ▁▅▇▂▁
19 sale_price                         0         1     314957.    179527.   55000 171500 259500 428875 999999 ▇▅▃▁▁
20 sq_footage                       315         0.403    965.      490.      375    750    874   1010   6215 ▇▁▁▁▁
21 condoCharges                      10         0.981    135.      284.        0      0      0      0  1501. ▇▁▁▁▁
```

Figure 1: Nominal and Ordinal features with Statistical Descriptions

The skimr library provides useful insight into the processed dataset. The nominal variables represented as factors in this dataset, consist of different categories that are used to represent the feature. Figure 1 summarizes the following for each factor feature, n_missing is the number of tuples that consist of an undefined value for that feature. Complete_rate describes the percentage of the tuples that do not have a value for the feature defined. Ordered defines if the factored attribute has any relating structure (e.g smallest to largest). N_unique defines the number of unique levels present in the feature. Top_counts display the most occurring levels with their total count of occurrence.

The ordinal variable in the dataset represented by figure 1 also consists of statistical descriptions. The n_missing and complete_rate column are defined the same as from the factors. The mean is the average of all tuples that have a value for the feature. Sd is the standard deviation of the values present in the feature. P0 to p100 describes the percentile of the feature. P0 is the minimum most value that the feature consists of, while p100 is the max value that occurs. The final column is a textual histogram that shows a low-resolution distribution of the data. All features described correlate with the sale price of an apartment

2.3. Missingness and Errors

The raw dataset consisted of various errors ranging from misspellings to missing values. For example, there was one tuple in the dataset that did not have a useful address as the zip code cannot be computed from the given set of characters; the tuple's zipcode was defined as missing in this case. Any cells that were represented as a collection of characters and a '$' were filtered into their numerical representation. Some features such as the kitchen_type consisted of different spellings to represent the type of the kitchen. Such values were categorized into their respective categories (e.g "eat in" and "Eat in" are an "eat-in" category).

There are several ways of handling missing data. The first way attempted was to simply do a list-wise deletion. This does not work; too much data went missing. Any cell in the dataset that did not have a corresponding value was imputed using the missing forest algorithm. The missing forest algorithm used default hyperparameters specified in the missingForest library in R to predict the values for cells with missing values. Any tuple that had a missing sale_price was dropped after imputing using missing forest. Missing response variables cannot be used for creating a predictive model; such tuples are removed.

3. Modeling

$\mathbb{D}$ is cleaned and imputed. There is no tuple where the response variable is missing. The following models reflect the sale price of an apartment in Queens, New York. The dataset can now be used for predicting the sale price using real observations. The rest of the report will consist of three models generated by Regression Tree, OLS, and Random Forrest algorithms. Each model will have different predictive powers and a real-life application.

3.1. Regression Tree Modeling

Decision trees create levels of decisions at each node that provide a conclusion. The Regression Tree algorithm computes and partitions the dataset at an optimal node for each split. This greedy iterative process divides into a left node and a right one. The algorithm for a Regression Tree model will keep splitting data until no additional splitting can occur. Due to complexities in installing the Yet Another Random Forest (YARF) package, the canonical Comprehensive R Archive Network (CRAN) package rpart will be used instead.

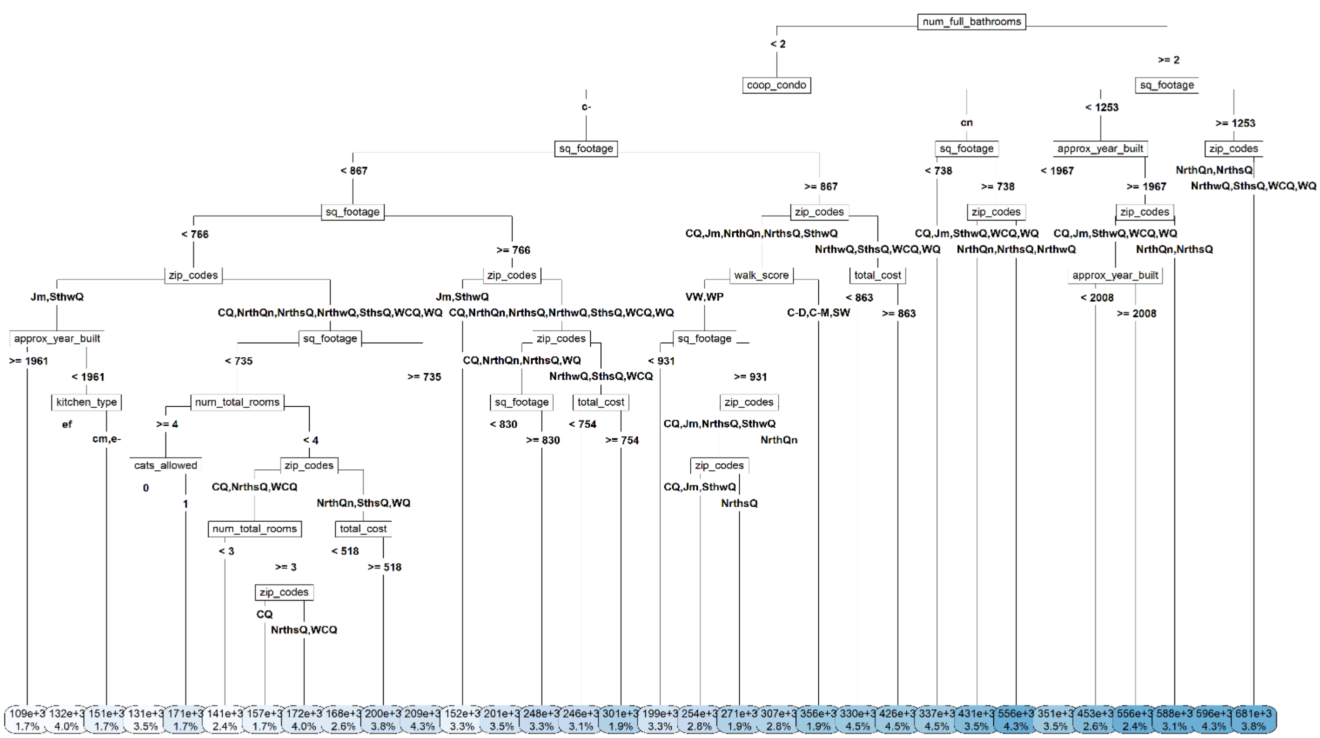

Figure 2: Rpart Regression Tree Model Plot

A single regression tree is known to have high variance thus causing drastic changes for the leaf nodes. It is expected that such models have very poor predictive performance. The top ten layers display the relationship that strongly influences the prediction of the sale price. According to the tree in figure 2, the following are the seemingly most important features num_full_bathrooms, coop_condo, sq_footage, approx_year_built, zip_codes, walk_score, total_cost, kitchen_type, num_total_rooms, and cats_allowed. Num_full_bathrooms are the most important feature in the tree as it strongly determines the sale price of an apartment. Interestingly, it makes sense that the number of full bathrooms. There is more of an influence to an apartment to have a spare bathroom for a guest thus also implying there is an extra room for guests. Such additions bring up an apartment value greatly (Beale, 2012). The coop-condo feature also has an important impact on the sale price. When someone purchases a co-op over a condo, they are buying a share of a corporation. A condo is true real estate that is ownership of the entire unit. The difference between a co-op and a condo will therefore greatly affect the sale price as the entire unit is not to be paid for if it is a co-op (Paley, 2020). Sq_footage is the next most influential feature, and it makes sense that they come after coop-condo as condos measure square footage differently. A co-op will measure square footage only on the interior perimeter, unlike a condo. Condos can often have skewed square footage which can make an apartment seem more affordable (Myers, 2019). Approx_year_built has an influence on the sale price when the building is over 20 years old. Correlated factors such as architecture and fixture designs have a big appearance appeal. With age, however, come higher maintenance costs, which will have a variation in sale price depending on the building standards of the year the building was built in.

Location, or in this specific case, zip_codes does have an influence on sale price. Property values increase with neighboring property. Thus, clustered groups of buildings with positive neighborhood and environmental factors strongly impact the sale price. For example, Figure 2 indicates that apartments located in the north of queens will have a higher cost of about $125,000 starting from $431,000. Walk_score determines how easily an individual can get around in a neighborhood. Walk scores have been found to correlate with increasing property value as it decreased automotive-related expenses. A single increase in walk score can increase the sale price by about $3,000 (Bokhari, 2020). The total_cost is also an influential factor towards the sale price. As defined previously, the total cost is the summation of both the cost of the maintenance and additional costs of a condo. Because the age of a building affects the maintenance costs, which itself affects the sale price, it makes sense that the total cost will also have an effect on the sale price. The type of kitchen also affects the sale price. When placed correctly and is accessible adds monetary value. Unlike a kitchen that is not as accessible or does not have well-kept appliances can decrease the sale prices (Morello, 2016). Finally, the num_total_rooms is also reasonable as it is correlated with square footage. The more rooms an apartment has, the more square footage it can offer. The number of rooms is a partitioning of the square footage which itself influences the sales price of an apartment.

3.2. Linear Modeling

Linear models provided from the OLS algorithm are helpful to analyze the relationships between linearly independent variables. One assumption with OLS is that that dataset is homoscedastic. This is a requirement of OLS as the noise can not change invariance as it will change the dataset's shape over a feature. In the dataset, many features were imputed and can have an innate incompleteness/missingness that changes the variation of noise between different features.

```
Call:
lm(formula = train$sale_price ~ ., data = train %>% select(-sale_price))

Residuals:
    Min      1Q  Median      3Q     Max
-304722  -37652   -5250   38896  291323

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -871200.29  618314.59  -1.409 0.159647
approx_year_built               402.10     317.17   1.268 0.205637
cats_allowed1                 13237.38   11610.70   1.140 0.254955
coop_condocondo              195813.57   14943.78  13.103  < 2e-16 ***
dining_room_typedining area  -12854.59   56693.30  -0.227 0.820747
dining_room_typeformal        19282.43   10272.25   1.877 0.061257 .
dining_room_typeother         18864.24   13710.98   1.376 0.169670
dogs_allowed1                  6377.21   12808.55   0.498 0.618849
fuel_typegas                   7724.92   31767.65   0.243 0.808004
fuel_typeoil                   7157.50   32295.12   0.222 0.824721
fuel_typeother                47194.56   42494.76   1.111 0.267437
garage_exists1                13462.51   10618.03   1.268 0.205605
kitchen_typeeat-in            -9506.04   12179.22  -0.781 0.435570
kitchen_typeefficiency       -21892.16   11940.86  -1.833 0.067520 .
num_bedrooms                  31794.83    8898.93   3.573 0.000398 ***
num_full_bathrooms            74037.57   14499.74   5.106 5.19e-07 ***
num_half_bathrooms            -8693.49   20039.84  -0.434 0.664670
num_total_rooms                5694.28    6319.11   0.901 0.368089
sq_footage                       37.47      14.38   2.606 0.009519 **
walk_score.L                  -8519.54   53762.61  -0.158 0.874173
walk_score.Q                  61888.07   45647.66   1.356 0.175967
walk_score.C                  -9866.65   34027.89  -0.290 0.772005
walk_score^4                  45591.92   21206.46   2.150 0.032187 *
zip_codesJamaica             -36085.44   22569.27  -1.599 0.110671
zip_codesNorth Queens         45962.55   18967.42   2.423 0.015844 *
zip_codesNortheast Queens     44014.75   20323.41   2.166 0.030948 *
zip_codesNorthwest Queens    160123.29   30136.70   5.313 1.83e-07 ***
zip_codesSoutheast Queens     31995.52   22727.27   1.408 0.159998
zip_codesSouthwest Queens    -39895.82   19734.84  -2.022 0.043912 *
zip_codesWest Central Queens  53642.44   19750.58   2.716 0.006906 **
zip_codesWest Queens          51714.08   20198.64   2.560 0.010841 *
total_cost                      155.96      16.60   9.396  < 2e-16 ***
approx_year_built_missing     31849.71   35437.75   0.899 0.369349
cats_allowed_missing                NA         NA      NA       NA
dining_room_type_missing      -4597.05    9892.39  -0.465 0.642406
fuel_type_missing              3219.34   19184.57   0.168 0.866822
kitchen_type_missing         -68239.88   38940.93  -1.752 0.080504 .
maintenance_cost_missing     -24954.69   19771.33  -1.262 0.207655
sq_footage_missing            -8167.27    8523.15  -0.958 0.338542
condoCharges_missing          48631.03   32307.49   1.505 0.133080
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 77530 on 384 degrees of freedom
Multiple R-squared:  0.8228,    Adjusted R-squared:  0.8053
F-statistic: 46.93 on 38 and 384 DF,  p-value: < 2.2e-16
```

Figure 3: Summary of OLS Linear Model

Figure 3 displays a table of the OLS model generated in R. The call defines how the OLS was defined to run. The residuals measure a sample of error for each tuple. The residuals for the model assess the difference between the data points. The residuals in figure 3 appear to be symmetrical around $-5,250 of the actual sale price. Therefore, having a good and consistent linear fit of the data. Next are the coefficients of the model. The intercept defines the sale price for a tuple that has negligible values for its features. The linear model extrapolates that such negative sales are possible, but do not reflect reality as negative sale prices do not occur. All the estimated weights created by the OLS algorithm are applied to the intercept value set. Many features spoken about before have a positive correlation with sale price given all other features are held constant. Factors that were turned into dummies in figure 3 display how each individual presence of a level affects the sale price of an apartment if they are present as a 1. One of the most influential features discussed is if the apartment sold is a condo. Co-op apartments are shares that have a much lower cost compared to condo apartments. Therefore, the estimated weight for the presence of a condo apartment increases the value of the sale price by $195,000. Location is another notable influential feature on the sale price generated from the OLS algorithm. Zipcodes that are centered around Northwest Queens provide an additional $160,123 to the sale price of an apartment, similar to the regression tree model.

The in-sample error statistics show that the model performs adequately. The standard error of the residuals is about $77530 from the observed sale price. The R^2 error metric, which displays how well predictions fit the data, is at 82% or .822. The RMSE which describes the root mean squared error states that the prediction of the sale price deviates by about $73,867.23. The out-of-sample error metrics states that the model performed only slightly worse with the testing dataset. The out-of-sample R^2 is 83% or .834 and has an RMSE of $80,213.91 which is only up to about $7,000 off from the in-sample RMSE.

3.3. Random Forest Modeling

Random forest models are known to have much more impressive error metrics. The term forest is derived from creating a set of trees that the model predicts. A single decision tree is created by creating a local optimum of the weight of a node. An algorithm that takes a local optimum is known as a greedy algorithm in computer science. In a regression tree algorithm, the dataset

$\mathbb{D}$ is used to compute all possible orthogonal-to-axis splits where each bucket-split there are two putative daughter nodes, and the prediction is the mean of the observations that landed in the bucket. As decision trees do not have their complexity stem from the sample size of the training dataset, random forests are non-parametric models. The sum of squares error (SSE) is computed for the splits as follows

$$SSE_W \coloneqq \frac{N_L SSE_L + N_R SSE_R}{N_L + N_R}$$

Where N_L represents the bucket size on the left, SSE_L represents the SSE of the bucket on the left, and respectively for the right bucket. The local optimum is taken by finding the minimum of the SSE_W at each node. The iterative process repeats until there are no additional splits left.

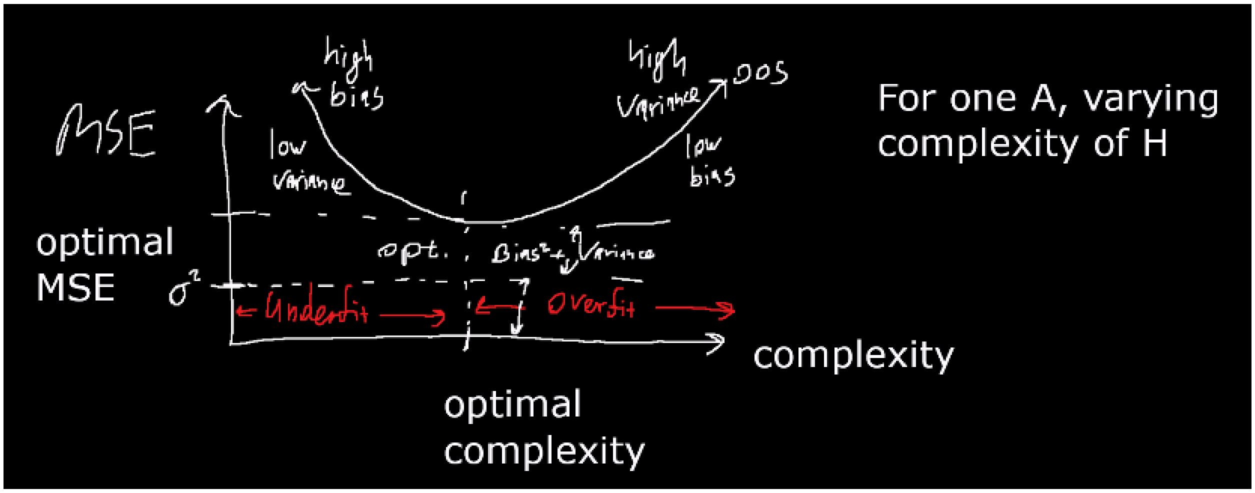

Figure 4: Variance and Bias of Predictions with Respect to MSE

Regression models, however, tend to have a large variance and a low bias, thus overfitting by increasing the complexity. The Mean Squared Error (MSE) is defined as follows:

$$MSE \coloneqq \delta^2 + Bias[G(x_\alpha)]^2 + Var[G(x_\alpha)]$$

Bootstrap aggregation, known also as bagging, allows for reducing the problem of high variance by increasing bias, thus decreasing the MSE as the bias of the model is already low. The bagging procedure constructs n number of regression trees using n training datasets. The predictions created by the set of tree models are averaged or taken the mode of if it is a classification model. Each individual tree consists of high variance but averaging the predictions increases bias as each tree may consist of shared observed tuples that are included in the final prediction. Therefore, the final output will have lower errors produced than a regression tree, becoming less complex, when decreasing MSE as in figure 4. While random forests decrease error by taking advantage of multiple trees is a positive benefit, a loss is that the trees lose interpretability.

As bagging requires creating a set of trees and sets of training data, additional parameters are required. Such parameters are called hyperparameters. A combination of different hyperparameters will strongly influence the predictive power of the final produced model. The Machine Learning in R (mlr) library allows for the optimization of the hyperparameters to pass into the Random Forest algorithm to bag the trees. The hyperparameters that the mlr library created for the random forest is having 12 features (mtry), 176 trees, and a node size of 19. As the bagging procedure in a random forest only reduces the high variance it is not likely that the random forest model will underfit; the averaged model will only decrease the overfitting by optimizing the hyperparameters to minimize MSE in figure 4.

It is not possible to tell the true causal variables of the sale price of apartment buildings in Queens. The truth function t or any of its parameters $z_1, \dots z_n$ will never be known, thus it is impossible, thus improvable, to know if there are any features in the model that will cause the sale price; there will always be an error due to ignorance because of this. However, the relationships between the sale price and the majority of the features are correlated. Proving a correlation requires research on the phenomenon and finding further factors that can influence the sale price. Features such as the walk_score have been researched in the past and have proven that there is a causation between said feature and the sale price of real estate (Bokhari, 2020).

4. Performance Results from Random Forest

The final model produced by the random forest algorithm has the following in-sample metrics, the R^2 is 91% or .91 and the RMSE is $51,096.27. For the training data, the model can very accurately predict the sale price of an apartment of about $42K from the actual sale price of the apartment. The model can also predict a $\hat{y}$ that only 9% of the predictions do not fit the data. The generalization error is also known as the out of sample error for the model is R^2 as 84% or .846 and the RMSE is $76,131.56. The performance metrics are much better than the performance metrics of a single regression tree model. The in-sample error statistics should be better than a regression tree as a regression tree consists of much more variance than a random forest which through bagging reduces variance in the error metric. So, the final model will have a deviation of about $76,131.56 then predicting the sales prices of an apartment using the random forest model.

5. Discussion

Many interesting things have come up while exploring and modeling this dataset. It was surprising to see the final oos error metrics for the random forest. There are many interpretations of that dataset that can take place such as converting the addresses to geological coordinates as a location has been researched to be a very influential factor for determining the sale price of any real estate. There are many areas of which could have been refined more such as data cleaning. The data cleaned in this report may have perhaps been over-generalized. The linear model and the regression tree show that while specific zip code areas such as Northwest Queens have a strong impact on the sale price, other zip code nodes in the regression tree have other specific breakdowns that determine the price of the apartment. One idea that has come up is to use a classification model such as a K Nearest Neighbors (KNN) to group zip codes or geological coordinates instead of separating them manually with a table that does not reflect all of Queens. Another small thing that fell short was a check for linear independence. While there was a code segment that checked for linear independence across the missing dummies, it falls short when checking across all the columns. Converting the columns into numeric values and computing the rank would be an improvement in this area where 0 columns are removed, thus keeping the dataset linearly independent.

In conclusion, it is unlikely that the random forest model will beat Zillow for predicting the sale price of apartments. Zillow is able to put much more amount of time and research into contributing factors that correlate a sale price more precisely. The random forest model error metrics clearly indicate that it has outperformed the regression tree model and the linear model. I believe the models are adequate to be production-ready, but also have room for improvement. With more time, money, and research, more precise and accurate models can be created that will further encapsulate the sale price of any apartment building.

References

Beale, L. (2012, March 2). Wealthy home buyers demand bathrooms; lots of bathrooms. Los Angeles Times. https://www.latimes.com/home/la-xpm-2012-mar-02-la-fi-many-bathrooms-20120303-story.html.

Bokhari, S. (2020, October 7). *How Much is a Point of Walk Score Worth?* Redfin Real Estate News. https://www.redfin.com/news/how-much-is-a-point-of-walk-score-worth/.

Crook, D. (2019, December 10). How NYC Property Taxes Are Calculated: StreetEasy. StreetEasy Blog. https://streeteasy.com/blog/nyc-property-taxes/#:~:text=In%20New%20York%20City%2C%20the,rise%20to%2021.167%25%20in%202020.

Evans, J. R. (2017, September 13). What Is a Full Bath? It Has 4 Separate Parts.

https://www.realtor.com/advice/sell/if-i-take-out-the-tub-does-a-bathroom-still-count-as-a-full-bath/.

Morello, R. (2016, September 27). Does a Small Kitchen Affect the Resale Value of Your Home? Budgeting Money - The Nest. https://budgeting.thenest.com/small-kitchen-affect-resale-value-home-23365.html.

Myers, E. (2019, April 1). How important is price per square foot, really? Brick Underground. https://www.brickunderground.com/sell/price-per-square-foot-how-important-really.

Paley, L. (2020, April 7). *Co-op vs. Condo: What to Know Before Buying: StreetEasy*. StreetEasy Blog. https://streeteasy.com/blog/co-ops-vs-condos-nyc-home-buyers-guide/.

Walk Score. (n.d.). Walk Score Methodology. https://www.walkscore.com/methodology.shtml#:~:text=Walk%20Score%20measures%20the%20walkability%20of%20any%20address%20using%20a,miles)%20are%20given%20maximum%20points.

Code Appendix

Final Project Code by Hubert Majewski

```
#Load required libraries
pacman::p_load(data.table, R.utils, tidyverse, skimr, mlr, rpart, rpart.plot,
missForest, randomForest, caret)

#Turn off warnings for presentation
options(warn = -1)

#Set randomization seed to make deterministic
set.seed(342)

#Load in the raw data
housing <- fread("https://raw.githubusercontent.com/kapelner/QC_MATH_342W_Spr
ing_2021/master/writing_assignments/housing_data_2016_2017.csv")

#Load it as a data.table object
housing <- data.table(housing)

#Summary of columns and table using skim
skim(housing)
```

Data summary

Name	housing
Number of rows	2230
Number of columns	55
_______________________	
Column type frequency:	
character	36
logical	5
numeric	14
________________________	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
HITId	758	0.66	30	30	0	1472	0
HITTypeId	758	0.66	30	30	0	2	0

Title	758	0.66	69	69	0	1	0
Description	758	0.66	46	47	0	2	0
Reward	758	0.66	5	5	0	1	0
CreationTime	758	0.66	28	28	0	62	0
RequesterAnnotation	758	0.66	48	48	0	2	0
Expiration	758	0.66	28	28	0	62	0
AssignmentId	758	0.66	30	30	0	1472	0
WorkerId	758	0.66	13	14	0	73	0
AssignmentStatus	758	0.66	8	8	0	1	0
AcceptTime	758	0.66	28	28	0	1457	0
SubmitTime	758	0.66	28	28	0	1460	0
AutoApprovalTime	758	0.66	28	28	0	1460	0
ApprovalTime	758	0.66	23	23	0	929	0
LifetimeApprovalRate	758	0.66	10	14	0	32	0
Last30DaysApprovalRate	758	0.66	10	14	0	32	0
Last7DaysApprovalRate	758	0.66	10	14	0	32	0
URL	758	0.66	73	105	0	1450	0
cats_allowed	0	1.00	1	3	0	3	0
common_charges	1684	0.24	3	7	0	258	0
coop_condo	0	1.00	5	5	0	2	0
date_of_sale	1702	0.24	8	10	0	222	0
dining_room_type	448	0.80	4	11	0	5	0
dogs_allowed	0	1.00	2	5	0	3	0
fuel_type	112	0.95	3	8	0	6	0
full_address_or_zip_code	0	1.00	5	59	0	1177	0
garage_exists	1826	0.18	1	11	0	6	0
kitchen_type	16	0.99	4	19	0	13	0
maintenance_cost	623	0.72	4	7	0	609	0
model_type	40	0.98	1	40	0	875	0
parking_charges	1671	0.25	2	4	0	89	0
sale_price	1702	0.24	8	9	0	315	0
total_taxes	1646	0.26	3	7	0	293	0
listing_price_to_nearest_1000	534	0.76	3	7	0	292	0

url	758	0.66	73	105	0	1450	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
Keywords	2230	0	NaN	:
NumberOfSimilarHITs	2230	0	NaN	:
LifetimeInSeconds	2230	0	NaN	:
RejectionTime	2230	0	NaN	:
RequesterFeedback	2230	0	NaN	:

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
MaxAssignments	758	0.66	1.00	0.00	1	1	1	1	1	▁▁▇▁▁
AssignmentDurationInSeconds	758	0.66	900.00	0.00	900	900	900	900	900	▁▁▇▁▁
AutoApprovalDelayInSeconds	758	0.66	60.00	0.00	60	60	60	60	60	▁▁▇▁▁
WorkTimeInSeconds	758	0.66	162.39	111.69	22	89	127	197	815	▇▂▁▁▁
approx_year_built	40	0.98	1962.71	21.08	1893	1950	1958	1970	2017	▁▃▇▂
community_district_num	19	0.99	26.33	2.95	3	25	26	28	32	▁▁▁▇
num_bedrooms	115	0.95	1.65	0.74	0	1	2	2	6	▇▃▁▁
num_floors_in_building	650	0.71	7.79	7.52	1	3	6	7	34	▇▁▁▁▁
num_full_bathrooms	0	1.00	1.23	0.44	1	1	1	1	3	▇▁▂▁▁
num_half_bathrooms	2058	0.08	0.95	0.30	0	1	1	1	2	▁▁▇▁▁
num_total_rooms	2	1.00	4.14	1.35	0	3	4	5	14	▁▇▂▁▁

pct_tax_deductibl	1754	0.21	45.40	6.95	20	40	50	50	75	
sq_footage	1210	0.46	955.36	380.86	100	743	881	1100	6215	
walk_score	0	1.00	83.92	14.75	7	77	89	95	99	

List-wise deletion attempt (if only it were this easy)

```
#Immediate List-wise deletion
LWhousing <- na.omit(housing)

rawCols <- ncol(LWhousing)
rawTotal <- nrow(LWhousing)

cat("Total LW columns", rawCols ,"\n", "Total LW tuples is", rawTotal, "\n")

## Total LW columns 55
##  Total LW tuples is 0

#It doesn't work so we need to first filter columns
```

Data Filtering

```
#Remove attributes which are not related with the cost of housing
housing2 <- housing %>%
    select(-HITId, -HITTypeId, -AssignmentStatus, -Title, -Description, -Assi
gnmentId, -AcceptTime, -SubmitTime, -URL, -url, -WorkerId, -date_of_sale, -Ke
ywords, -model_type, -NumberOfSimilarHITs, -community_district_num, -Lifetime
InSeconds, -AcceptTime, -ApprovalTime, -AutoApprovalTime, -RejectionTime, -Re
questerFeedback, -Reward, -MaxAssignments, -RequesterAnnotation, -AssignmentD
urationInSeconds, -AutoApprovalDelayInSeconds, -Expiration, -Last30DaysApprov
alRate, -Last7DaysApprovalRate, -date_of_sale, -WorkTimeInSeconds, -model_typ
e, -LifetimeApprovalRate, -parking_charges, -MaxAssignments, -CreationTime, -
SubmitTime, -pct_tax_deductibl, -listing_price_to_nearest_1000, -num_floors_i
n_building) #%>% select(-garage_exists) # I disagree with this. May add value
to the entire building/apartment if it is a part of it.


#Convert costs to continuous as it can be anything in between
housing2 <- housing2 %>% mutate(sale_price = as.numeric(str_remove_all(sale_p
rice, "[$,]" )))
housing2 <- housing2 %>% mutate(total_taxes = as.numeric(str_remove_all(total
_taxes, "[$,]" )))
housing2 <- housing2 %>% mutate(common_charges = as.numeric(str_remove_all(co
mmon_charges, "[$,]" )))
housing2 <- housing2 %>% mutate(maintenance_cost = as.numeric(str_remove_all(
maintenance_cost, "[$,]" )))
```

```
#Convert address into zipcodes
zip_codes <- gsub("[^0-9.-]", "", housing2$full_address_or_zip_code)
housing2$zip_codes = str_sub(zip_codes, -5, -1)

#Specific cases
housing2$zip_codes[housing2$zip_codes == "1367."] <- "11367" #Specific cases
where the initial zip code was malformed
housing2$zip_codes[housing2$zip_codes == ".1136"] <- "11369"
housing2$zip_codes[housing2$zip_codes == "1355."] <- "11355"

#Factor all attributes that are categories
housing2 <- housing2 %>%
  mutate(zip_codes = as.factor(case_when(
    zip_codes == "11361" | zip_codes == "11362" | zip_codes == "11363" | zip_
codes == "11364" ~ "Northeast Queens",
    zip_codes == "11354" | zip_codes == "11355" | zip_codes == "11356" | zip_
codes == "11357" | zip_codes == "11358" | zip_codes == "11359" | zip_codes ==
"11360" ~ "North Queens",
    zip_codes == "11365" | zip_codes == "11366" | zip_codes == "11367" ~ "Cen
tral Queens",
    zip_codes == "11412" | zip_codes == "11423" | zip_codes == "11432" | zip_
codes == "11433" | zip_codes == "11434" | zip_codes == "11435" | zip_codes ==
"11436" ~ "Jamaica",
    zip_codes == "11101" | zip_codes == "11102" | zip_codes == "11103" | zip_
codes == "11104" | zip_codes == "11105" | zip_codes == "11106" ~ "Northwest Q
ueens",
    zip_codes == "11374" | zip_codes == "11375" | zip_codes == "11379" | zip_
codes == "11385" ~ "West Central Queens",
    zip_codes == "11004" | zip_codes == "11005" | zip_codes == "11411" | zip_
codes == "11413" | zip_codes == "11422" | zip_codes == "11426" | zip_codes ==
"11427" | zip_codes == "11428" | zip_codes == "11429" ~ "Southeast Queens",
    zip_codes == "11414" | zip_codes == "11415" | zip_codes == "11416" | zip_
codes == "11417" | zip_codes == "11418" | zip_codes == "11419" | zip_codes ==
"11420"  | zip_codes == "11421" ~ "Southwest Queens",
    zip_codes == "11368" | zip_codes == "11369" | zip_codes == "11370" | zip_
codes == "11372" | zip_codes == "11373" | zip_codes == "11377" | zip_codes ==
"11378"  ~ "West Queens"
           )))

#Using website as city definition https://www.walkscore.com/methodology.shtml
#:~:text=Walk%20Score%20measures%20the%20walkability%20of%20any%20address%20u
sing%20a,miles)%20are%20given%20maximum%20points
housing2$walk_score <- ordered(as.factor(case_when(housing2$walk_score < 25 ~
"Car-Dependent",
                                                        housing2$walk_score >
= 25 & housing2$walk_score < 50 ~ "Car-Mostly-Dependent",
                                                        housing2$walk_score >
= 50 & housing2$walk_score < 70 ~ "Somewhat Walkable",
                                                        housing2$walk_score >
```

```
= 70 & housing2$walk_score < 90 ~ "Very Walkable",
                            housing2$walk_score >= 90 ~ "Walker's Paradise"))
)

#ordering the walk_score because it is that way
housing2$walk_score <- ordered(housing2$walk_score, levels = c("Car-Dependent
", "Car-Mostly-Dependent", "Somewhat Walkable", "Very Walkable", "Walker's Pa
radise"))

housing2$approx_year_built <- as.integer(housing2$approx_year_built)

housing2 <- housing2 %>%
  mutate(kitchen_type = as.factor(case_when(
    kitchen_type == "efficiency" | kitchen_type == "efficiency kitchene" | ki
tchen_type == "efficiency ktchen" | kitchen_type == "efficiency kitchen" | ki
tchen_type == "efficiemcy" ~ "efficiency",
    kitchen_type == "Combo" | kitchen_type == "combo" ~ "combo",
    kitchen_type == "eat in" | kitchen_type == "Eat In" | kitchen_type == "ea
tin" | kitchen_type == "Eat in" ~ "eat-in")))

housing2$num_half_bathrooms <- ifelse(is.na(housing2$num_half_bathrooms), 0,
housing2$num_half_bathrooms)

housing2 <- housing2 %>%
  mutate(cats_allowed = as.factor(ifelse(cats_allowed == "no", 0, 1)))

housing2 <- housing2 %>%
  mutate(dogs_allowed = as.factor(ifelse(dogs_allowed == "no", 0, 1)))

housing2 <- housing2 %>%
  mutate(garage_exists = as.factor(ifelse(is.na(garage_exists), 0, 1)))

housing2 <- housing2 %>% mutate(fuel_type = as.factor(ifelse(fuel_type == "Ot
her" | fuel_type == "none", "other", fuel_type)))

housing2 <- housing2 %>% mutate(dining_room_type = as.factor(dining_room_type
))

housing2 <- housing2 %>% mutate(maintenance_cost = ifelse(coop_condo == "cond
o", replace(maintenance_cost, is.na(maintenance_cost), 0), maintenance_cost))

housing2 <- housing2 %>% mutate(total_taxes = replace(total_taxes, is.na(tota
l_taxes), 0 )) %>%
                                                mutate(common_charges = ifel
se(coop_condo == "co-op", replace(common_charges, is.na(common_charges), 0),
common_charges)) %>%
                                                mutate(condoCharges = ifelse
(coop_condo == "condo", common_charges + (total_taxes / 12), 0))

housing2 <- housing2 %>% select(-total_taxes, -common_charges, -full_address_
```

```
or_zip_code)

housing <- housing2 %>% mutate(coop_condo = as.factor(coop_condo))

#Print cleaned
skim(housing)
```

Data summary

Name	housing
Number of rows	2230
Number of columns	18
_______________________	
Column type frequency:	
factor	9
numeric	9
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1.00	FALSE	2	0: 1402, 1: 828
coop_condo	0	1.00	FALSE	2	co-: 1661, con: 569
dining_room_type	448	0.80	FALSE	5	com: 957, for: 620, oth: 201, din: 2
dogs_allowed	0	1.00	FALSE	2	0: 1684, 1: 546
fuel_type	112	0.95	FALSE	4	gas: 1348, oil: 664, ele: 62, oth: 44
garage_exists	0	1.00	FALSE	2	0: 1826, 1: 404
kitchen_type	40	0.98	FALSE	3	eat: 942, eff: 849, com: 399
walk_score	0	1.00	TRUE	5	Wal: 1089, Ver: 821, Som: 243, Car: 67
zip_codes	13	0.99	FALSE	9	Nor: 551, Wes: 455, Wes: 337, Sou: 205

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist

approx_year_built	40	0.98	1962.71	21.08	1893	1950	1958	1970	2017.00	
maintenance_cost	109	0.95	650.73	498.20	0	310	673	900	4659.00	
num_bedrooms	115	0.95	1.65	0.74	0	1	2	2	6.00	
num_full_bathrooms	0	1.00	1.23	0.44	1	1	1	1	3.00	
num_half_bathrooms	0	1.00	0.07	0.27	0	0	0	0	2.00	
num_total_rooms	2	1.00	4.14	1.35	0	3	4	5	14.00	
sale_price	1702	0.24	314956.56	179526.60	55000	171500	259500	428875	999999.00	
sq_footage	1210	0.46	955.36	380.86	100	743	881	1100	6215.00	
condoCharges	84	0.96	133.49	281.76	0	0	0	0	1591.67	

```
head(housing)

##    approx_year_built cats_allowed coop_condo dining_room_type dogs_allowed
## 1:              1955            0      co-op            combo            0
## 2:              1955            0      co-op           formal            0
## 3:              2004            0      condo            combo            0
## 4:              2002            0      condo            combo            0
## 5:              1949            1      co-op            combo            1
## 6:              1938            1      co-op            combo            1
##    fuel_type garage_exists kitchen_type maintenance_cost num_bedrooms
## 1:       gas             0       eat-in               NA            2
## 2:       oil             0       eat-in              604            1
## 3:      <NA>             0   efficiency                0            1
## 4:       gas             0       eat-in                0            3
## 5:       gas             0       eat-in              660            2
## 6:       oil             0       eat-in              932            2
##    num_full_bathrooms num_half_bathrooms num_total_rooms sale_price sq_foo
tage
## 1:                  1                  0               5     228000
NA
## 2:                  1                  0               4     235500
890
```

```
## 3:                       1                   0               3      137550
550
## 4:                       2                   0               5      545000
NA
## 5:                       1                   0               4      241700
675
## 6:                       1                   0               4      250000
1000
##           walk_score        zip_codes condoCharges
## 1:     Very Walkable     North Queens       0.0000
## 2:     Very Walkable     North Queens       0.0000
## 3: Walker's Paradise      West Queens     625.3333
## 4: Walker's Paradise     North Queens     463.3333
## 5:     Very Walkable Southeast Queens       0.0000
## 6: Walker's Paradise Southwest Queens       0.0000
```

Dealing with missingness

```
#Record the nulls into their own columns
M <- tibble::as_tibble(apply(is.na(housing), 2, as.numeric))
colnames(M) = paste(colnames(housing), "_missing", sep = "")
M <- tibble::as_tibble(t(unique(t(M))))
m <- M %>%
  select_if(function(x){sum(x) > 0})

housing2 <- cbind(M, housing)


#Prep for missing forest
housing2NA = housing2 %>%
  filter(is.na(sale_price))
housing2 = housing2 %>%
  filter(!is.na(sale_price))

#Split
n = nrow(housing2)
k = 5

test_indices <- sample(1 : n, 1 / k * n)
train_indices <- setdiff(1 : n, test_indices)

training <- housing2[train_indices, ]
testing <- housing2[test_indices, ]

XTest <- testing %>%
  mutate(sale_price = NA)
yTest <- testing$sale_price

#Print a summary of the data before imputation
summary(housing2)
```

```
##  approx_year_built_missing cats_allowed_missing dining_room_type_missing
##  Min.   :0.00000           Min.   :0            Min.   :0.0000
##  1st Qu.:0.00000           1st Qu.:0            1st Qu.:0.0000
##  Median :0.00000           Median :0            Median :0.0000
##  Mean   :0.01136           Mean   :0            Mean   :0.2273
##  3rd Qu.:0.00000           3rd Qu.:0            3rd Qu.:0.0000
##  Max.   :1.00000           Max.   :0            Max.   :1.0000
##
##  fuel_type_missing kitchen_type_missing maintenance_cost_missing
##  Min.   :0.00000   Min.   :0.00000      Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000      1st Qu.:0.00000
##  Median :0.00000   Median :0.00000      Median :0.00000
##  Mean   :0.04545   Mean   :0.01326      Mean   :0.03977
##  3rd Qu.:0.00000   3rd Qu.:0.00000      3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000      Max.   :1.00000
##
##  num_bedrooms_missing num_total_rooms_missing sale_price_missing
##  Min.   :0            Min.   :0               Min.   :0
##  1st Qu.:0            1st Qu.:0               1st Qu.:0
##  Median :0            Median :0               Median :0
##  Mean   :0            Mean   :0               Mean   :0
##  3rd Qu.:0            3rd Qu.:0               3rd Qu.:0
##  Max.   :0            Max.   :0               Max.   :0
##
##  sq_footage_missing zip_codes_missing condoCharges_missing approx_year_bui
lt
##  Min.   :0.0000     Min.   :0         Min.   :0.00000      Min.   :1915
##  1st Qu.:0.0000     1st Qu.:0         1st Qu.:0.00000      1st Qu.:1950
##  Median :1.0000     Median :0         Median :0.00000      Median :1957
##  Mean   :0.5966     Mean   :0         Mean   :0.01894      Mean   :1962
##  3rd Qu.:1.0000     3rd Qu.:0         3rd Qu.:0.00000      3rd Qu.:1968
##  Max.   :1.0000     Max.   :0         Max.   :1.00000      Max.   :2016
##                                                            NA's   :6
##  cats_allowed coop_condo     dining_room_type dogs_allowed    fuel_type
##  0:285        co-op:399   combo       :241     0:381        electric: 11
##  1:243        condo:129   dining area:  2     1:147        gas     :301
##                           formal     :116                  oil     :180
##                           none       :  0                  other   : 12
##                           other      : 49                  NA's    : 24
##                           NA's       :120
##
##  garage_exists     kitchen_type maintenance_cost  num_bedrooms
##  0:434          combo     : 81   Min.   :   0.0   Min.   :0.000
##  1: 94          eat-in    :209   1st Qu.: 387.0   1st Qu.:1.000
##                 efficiency:231   Median : 670.0   Median :1.000
##                 NA's      :  7   Mean   : 625.7   Mean   :1.538
##                                  3rd Qu.: 827.0   3rd Qu.:2.000
##                                  Max.   :4659.0   Max.   :3.000
##                                  NA's   :21
##  num_full_bathrooms num_half_bathrooms num_total_rooms   sale_price
```

```
##  Min.   :1.000      Min.   :0.00000    Min.   :1.000   Min.   : 55000
##  1st Qu.:1.000      1st Qu.:0.00000    1st Qu.:3.000   1st Qu.:171500
##  Median :1.000      Median :0.00000    Median :4.000   Median :259500
##  Mean   :1.205      Mean   :0.05871    Mean   :4.025   Mean   :314957
##  3rd Qu.:1.000      3rd Qu.:0.00000    3rd Qu.:5.000   3rd Qu.:428875
##  Max.   :3.000      Max.   :2.00000    Max.   :8.000   Max.   :999999
##
##    sq_footage                     walk_score                   zip_codes
##  Min.   : 375.0   Car-Dependent        :  2   North Queens        :113
##  1st Qu.: 750.0   Car-Mostly-Dependent:  9   West Central Queens: 93
##  Median : 874.0   Somewhat Walkable    : 61   Northeast Queens   : 72
##  Mean   : 965.3   Very Walkable        :237   West Queens        : 69
##  3rd Qu.:1010.0   Walker's Paradise    :219   Southwest Queens   : 59
##  Max.   :6215.0                                Central Queens     : 34
##  NA's   :315                                   (Other)            : 88
##   condoCharges
##  Min.   :   0.0
##  1st Qu.:   0.0
##  Median :   0.0
##  Mean   : 135.3
##  3rd Qu.:   0.0
##  Max.   :1500.9
##  NA's   :10

skim(housing2)
```

Data summary

Name	housing2
Number of rows	528
Number of columns	30
_______________________	
Column type frequency:	
factor	9
numeric	21
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1.00	FALSE	2	0: 285, 1: 243
coop_condo	0	1.00	FALSE	2	co-: 399, con: 129

dining_room_type	120	0.77	FALSE	4	com: 241, for: 116, oth: 49, din: 2
dogs_allowed	0	1.00	FALSE	2	0: 381, 1: 147
fuel_type	24	0.95	FALSE	4	gas: 301, oil: 180, oth: 12, ele: 11
garage_exists	0	1.00	FALSE	2	0: 434, 1: 94
kitchen_type	7	0.99	FALSE	3	eff: 231, eat: 209, com: 81
walk_score	0	1.00	TRUE	5	Ver: 237, Wal: 219, Som: 61, Car: 9
zip_codes	0	1.00	FALSE	9	Nor: 113, Wes: 93, Nor: 72, Wes: 69

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
approx_year_built_missing	0	1.00	0.01	0.11	0	0	0	0	1.00	▇▁▁▁▁
cats_allowed_missing	0	1.00	0.00	0.00	0	0	0	0	0.00	▁▁▇▁▁
dining_room_type_missing	0	1.00	0.23	0.42	0	0	0	0	1.00	▇▁▁▁▂
fuel_type_missing	0	1.00	0.05	0.21	0	0	0	0	1.00	▇▁▁▁▁
kitchen_type_missing	0	1.00	0.01	0.11	0	0	0	0	1.00	▇▁▁▁▁
maintenance_cost_missing	0	1.00	0.04	0.20	0	0	0	0	1.00	▇▁▁▁▁
num_bedrooms_missing	0	1.00	0.00	0.00	0	0	0	0	0.00	▁▁▇▁▁
num_total_rooms_missing	0	1.00	0.00	0.00	0	0	0	0	0.00	▁▁▇▁▁

sale_price_missing	0	1.00	0.00	0.00	0	0	0	0	0.00	▁▁▇▁▁
sq_footage_missing	0	1.00	0.60	0.49	0	0	1	1	1.00	▅▁▁▁▇
zip_codes_missing	0	1.00	0.00	0.00	0	0	0	0	0.00	▁▁▇▁▁
condoCharges_missing	0	1.00	0.02	0.14	0	0	0	0	1.00	▇▁▁▁▁
approx_year_built	6	0.99	1962.38	20.56	1915	1950	1957	1968	2016.00	▁▁▆▇▂
maintenance_cost	21	0.96	625.71	481.80	0	387	670	827	4659.00	▇▃▁▁▁
num_bedrooms	0	1.00	1.54	0.75	0	1	1	2	3.00	▁▇▁▇▃
num_full_bathrooms	0	1.00	1.20	0.42	1	1	1	1	3.00	▇▁▂▁▁
num_half_bathrooms	0	1.00	0.06	0.24	0	0	0	0	2.00	▇▁▁▁▁
num_total_rooms	0	1.00	4.02	1.20	1	3	4	5	8.00	▁▅▇▂▁
sale_price	0	1.00	314956.56	179526.60	55000	171500	259500	428875	999999.00	▇▃▂▁▁
sq_footage	315	0.40	965.28	490.42	375	750	874	1010	6215.00	▇▁▁▁▁
condoCharges	10	0.98	135.26	284.11	0	0	0	0	1500.92	▇▁▁▁▁

```
#Fill in missingness
housing3 <- missForest(rbind(training, XTest, housing2NA))$ximp

##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
```

```
##   missForest iteration 3 in progress...done!
##   missForest iteration 4 in progress...done!

#Remove origional y that was missing for modeling
housing3 <- housing3 %>% filter(sale_price_missing == 0) %>%
    select(-sale_price_missing)

#Remove origional zipcodes that was missing (about 1 tuple?)
#housing3 <- housing3 %>% filter(zip_codes_missing == 0) %>%
#   select(-zip_codes_missing)

#Compute imputed costs on tuple
housing3 <- housing3 %>%
  mutate(total_cost = maintenance_cost + condoCharges) %>%
  select(-maintenance_cost, -condoCharges)

#Retain linear independence
#Note: REMOVES NUMERIC AND FACTORS FROM TABLE AND SETS THEM AS CHARACTERS DUE
TO COL COMPARISONS
housing3 <- cbind(housing3[, -(1:11)], tibble::as_tibble(t(unique(t(housing3[
, (1:11)])))))
#housing3 <- housing3[, qr(housing3)$pivot[seq_len(qr(housing3)$rank)]]
#housing3 <- cbind(housing3[, -(1:ncol(housing3))], tbl_df(t(unique(t(housing
3[, (1:ncol(housing3))])))))
#housing3 <- sapply(1:ncol(housing3), function (x) qr(housing3[,-x])$rank)
#which(rankifremoved == max(rankifremoved))

#Reinsert the yTest into the testing dataset
train <- housing3[1:as.integer(n - as.integer(1 / k * n)), ]
test <- housing3[(as.integer(n - as.integer(1 / k * n)) + 1):n, ]
test$sale_price <- yTest

#Print filled
skim(housing3)
```

Data summary

Name	housing3
Number of rows	528
Number of columns	25
_______________________	
Column type frequency:	
factor	9
numeric	16
________________________	

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1	FALSE	2	0: 285, 1: 243
coop_condo	0	1	FALSE	2	co-: 399, con: 129
dining_room_type	0	1	FALSE	4	com: 330, for: 135, oth: 60, din: 3
dogs_allowed	0	1	FALSE	2	0: 381, 1: 147
fuel_type	0	1	FALSE	4	gas: 312, oil: 192, oth: 13, ele: 11
garage_exists	0	1	FALSE	2	0: 434, 1: 94
kitchen_type	0	1	FALSE	3	eff: 233, eat: 213, com: 82
walk_score	0	1	TRUE	5	Ver: 237, Wal: 219, Som: 61, Car: 9
zip_codes	0	1	FALSE	9	Nor: 113, Wes: 93, Nor: 72, Wes: 69

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
approx_year_built	0	1	1962.28	20.47	1915.00	1950.00	1956.00	1966.50	2016	▁▇▆▂▂
num_bedrooms	0	1	1.54	0.75	0.00	1.00	1.00	2.00	3	▁▇▁▇▂
num_full_bathrooms	0	1	1.20	0.42	1.00	1.00	1.00	1.00	3	▇▁▂▁▁
num_half_bathrooms	0	1	0.06	0.24	0.00	0.00	0.00	0.00	2	▇▁▁▁▁
num_total_rooms	0	1	4.02	1.20	1.00	3.00	4.00	5.00	8	▁▆▇▂▁
sale_price	0	1	314264.84	170472.90	55000.00	173750.00	262000.00	430000.00	950000	▇▆▃▁▁
sq_footage	0	1	894.67	359.42	375.00	711.03	828.81	984.27	6215	▇▁▁▁▁

total_cost	0	1	774.36	367.60	148.92	584.00	713.00	869.25	4659	▇▁▁▁▁
approx_year_built_missing	0	1	0.01	0.11	0.00	0.00	0.00	0.00	1	▇▁▁▁▁
cats_allowed_missing	0	1	0.00	0.00	0.00	0.00	0.00	0.00	0	▁▁▇▁▁
dining_room_type_missing	0	1	0.23	0.42	0.00	0.00	0.00	0.00	1	▇▁▁▁▂
fuel_type_missing	0	1	0.05	0.21	0.00	0.00	0.00	0.00	1	▇▁▁▁▁
kitchen_type_missing	0	1	0.01	0.11	0.00	0.00	0.00	0.00	1	▇▁▁▁▁
maintenance_cost_missing	0	1	0.04	0.20	0.00	0.00	0.00	0.00	1	▇▁▁▁▁
sq_footage_missing	0	1	0.60	0.49	0.00	0.00	1.00	1.00	1	▅▁▁▁▇
condoCharges_missing	0	1	0.02	0.14	0.00	0.00	0.00	0.00	1	▇▁▁▁▁

```
head(housing3)

##   approx_year_built cats_allowed coop_condo dining_room_type dogs_allowed
## 1              1955            0      co-op            combo            0
## 2              1955            0      co-op           formal            0
## 3              2004            0      condo            combo            0
## 4              2002            0      condo            combo            0
## 5              1949            1      co-op            combo            1
## 6              1938            1      co-op            combo            1
##   fuel_type garage_exists kitchen_type num_bedrooms num_full_bathrooms
## 1       gas             0       eat-in            2                  1
## 2       oil             0       eat-in            1                  1
## 3       gas             0   efficiency            1                  1
## 4       gas             0       eat-in            3                  2
## 5       gas             0       eat-in            2                  1
## 6       oil             0       eat-in            2                  1
##   num_half_bathrooms num_total_rooms sale_price sq_footage        walk_sco
```

```
re
## 1                  0               5     228000   878.7562     Very Walkab
le
## 2                  0               4     235500   890.0000     Very Walkab
le
## 3                  0               3     137550   550.0000 Walker's Paradi
se
## 4                  0               5     545000  1077.9034 Walker's Paradi
se
## 5                  0               4     241700   675.0000     Very Walkab
le
## 6                  0               4     250000  1000.0000 Walker's Paradi
se
##          zip_codes total_cost approx_year_built_missing cats_allowed_missi
ng
## 1     North Queens   845.8436                         0
0
## 2     North Queens   604.0000                         0
0
## 3      West Queens   625.3333                         0
0
## 4     North Queens   463.3333                         0
0
## 5 Southeast Queens   660.0000                         0
0
## 6 Southwest Queens   932.0000                         0
0
##   dining_room_type_missing fuel_type_missing kitchen_type_missing
## 1                        0                 0                    0
## 2                        0                 0                    0
## 3                        0                 1                    0
## 4                        0                 0                    0
## 5                        0                 0                    0
## 6                        0                 0                    0
##   maintenance_cost_missing sq_footage_missing condoCharges_missing
## 1                        1                  1                    0
## 2                        0                  0                    0
## 3                        0                  0                    0
## 4                        0                  1                    0
## 5                        0                  0                    0
## 6                        0                  0                    0
```

Regression Tree Model

```
#Create one Regression Tree (anova -> regression)
rtModel <- rpart(train$sale_price ~ ., data = train %>% select(-sale_price),
method = "anova", control = list(cp = 0, xval = 10))

png("Regression_Tree_Model_Plot.png",width = 5888, height = 3312, res = 250)
rpart.plot(rtModel, tweak = 1.235, fallen.leaves = TRUE, type = 5, faclen = 2
```

```
, digits = 3)
dev.off()

## png
##   2

#plotcp(rtModel)
rtModel

## n= 423
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##    1) root 423 1.302784e+13 313311.3
##      2) num_full_bathrooms< 1.5 340 5.707659e+12 257218.0
##        4) coop_condo=co-op 288 2.508365e+12 224205.3
##          8) sq_footage< 867.322 196 6.066295e+11 182076.2
##           16) sq_footage< 765.906 132 2.286658e+11 162556.6
##             32) zip_codes=Jamaica,Southwest Queens 31 2.468297e+10 130967.
7
##               64) approx_year_built>=1960.5 7 6.807429e+09 108714.3 *
##               65) approx_year_built< 1960.5 24 1.339796e+10 137458.3
##                130) kitchen_type=efficiency 17 4.572941e+09 132058.8 *
##                131) kitchen_type=combo,eat-in 7 7.125714e+09 150571.4 *
##             33) zip_codes=Central Queens,North Queens,Northeast Queens,Nor
thwest Queens,Southeast Queens,West Central Queens,West Queens 101 1.635547e+
11 172252.2
##               66) sq_footage< 734.8331 83 1.121606e+11 164261.2
##                132) num_total_rooms>=3.5 22 2.446729e+10 143413.6
##                  264) cats_allowed=0 15 9.438229e+09 130593.3 *
##                  265) cats_allowed=1 7 7.280649e+09 170885.7 *
##                133) num_total_rooms< 3.5 61 7.468318e+10 171780.0
##                  266) zip_codes=Central Queens,Northeast Queens,West Centr
al Queens 34 2.315201e+10 159691.2
##                    532) num_total_rooms< 2.5 10 3.804500e+09 141000.0 *
##                    533) num_total_rooms>=2.5 24 1.439824e+10 167479.2
##                     1066) zip_codes=Central Queens 7 7.162857e+09 157142.9
*
##                     1067) zip_codes=Northeast Queens,West Central Queens 1
7 6.179559e+09 171735.3 *
##                  267) zip_codes=North Queens,Southeast Queens,West Queens
27 4.030558e+10 187002.9
##                    534) total_cost< 517.5 11 5.405833e+09 167925.2 *
##                    535) total_cost>=517.5 16 2.814378e+10 200118.8 *
##               67) sq_footage>=734.8331 18 2.165436e+10 209100.0 *
##           17) sq_footage>=765.906 64 2.239399e+11 222335.1
##             34) zip_codes=Jamaica,Southwest Queens 14 1.200886e+10 152285.
7 *
##             35) zip_codes=Central Queens,North Queens,Northeast Queens,Nor
```

```
thwest Queens,Southeast Queens,West Central Queens,West Queens 50 1.239990e+1
1 241949.0
##               70) zip_codes=Central Queens,North Queens,Northeast Queens,W
est Queens 29 6.316092e+10 223758.2
##                140) sq_footage< 830.281 15 2.719203e+10 200732.6 *
##                141) sq_footage>=830.281 14 1.949543e+10 248428.6 *
##               71) zip_codes=Northwest Queens,Southeast Queens,West Central
Queens 21 3.799003e+10 267069.5
##                142) total_cost< 753.5 13 7.366118e+09 245920.0 *
##                143) total_cost>=753.5 8 1.535972e+10 301437.5 *
##          9) sq_footage>=867.322 92 8.127418e+11 313958.6
##           18) zip_codes=Central Queens,Jamaica,North Queens,Northeast Quee
ns,Southwest Queens 54 2.586232e+11 269172.2
##             36) walk_score=Very Walkable,Walker's Paradise 46 1.689628e+11
254006.5
##               72) sq_footage< 930.947 14 3.132250e+10 199000.0 *
##               73) sq_footage>=930.947 32 7.674772e+10 278071.9
##                146) zip_codes=Central Queens,Jamaica,Northeast Queens,Sout
hwest Queens 20 1.153196e+10 260790.0
##                  292) zip_codes=Central Queens,Jamaica,Southwest Queens 12
9.385667e+09 253833.3 *
##                  293) zip_codes=Northeast Queens 8 6.944350e+08 271225.0 *
##                147) zip_codes=North Queens 12 4.928706e+10 306875.0 *
##             37) walk_score=Car-Dependent,Car-Mostly-Dependent,Somewhat Wal
kable 8 1.824588e+10 356375.0 *
##           19) zip_codes=Northwest Queens,Southeast Queens,West Central Que
ens,West Queens 38 2.918845e+11 377602.3
##             38) total_cost< 863 19 1.238806e+11 329546.7 *
##             39) total_cost>=863 19 8.024903e+10 425657.9 *
##        5) coop_condo=condo 52 1.147037e+12 440058.0
##         10) sq_footage< 737.5752 19 3.138451e+11 337444.1 *
##         11) sq_footage>=737.5752 33 5.179420e+11 499138.7
##           22) zip_codes=Central Queens,Jamaica,Southwest Queens,West Centr
al Queens,West Queens 15 1.275730e+11 431479.2 *
##           23) zip_codes=North Queens,Northeast Queens,Northwest Queens 18
2.644794e+11 555521.6 *
##      3) num_full_bathrooms>=1.5 83 1.868090e+12 543091.0
##        6) sq_footage< 1253.179 49 7.567748e+11 478490.8
##         12) approx_year_built< 1966.5 15 1.219071e+11 350870.0 *
##         13) approx_year_built>=1966.5 34 2.827796e+11 534794.1
##           26) zip_codes=Central Queens,Jamaica,Southwest Queens,West Centr
al Queens,West Queens 21 1.916778e+11 502095.2
##             52) approx_year_built< 2007.5 11 8.497164e+10 453181.8 *
##             53) approx_year_built>=2007.5 10 5.143890e+10 555900.0 *
##           27) zip_codes=North Queens,Northeast Queens 13 3.237708e+10 5876
15.4 *
##        7) sq_footage>=1253.179 34 6.121290e+11 636191.2
##         14) zip_codes=North Queens,Northeast Queens 18 2.776612e+11 595972
.2 *
```

```
##         15) zip_codes=Northwest Queens,Southeast Queens,West Central Queen
s,West Queens 16 2.725959e+11 681437.5 *
```

```
#RMSE IS
predictions <- rtModel %>% predict(train %>% select(-sale_price))
RMSE(predictions, train$sale_price)

## [1] 69597.16

R2(predictions, train$sale_price)

## [1] 0.8427281
```

Linear Model

```
#Creating one linear model with intercept
lmModel = lm(train$sale_price ~ ., train %>% select(-sale_price))
lmModel

## 
## Call:
## lm(formula = train$sale_price ~ ., data = train %>% select(-sale_price))
## 
## Coefficients:
##                  (Intercept)               approx_year_built  
##                   -871200.29                          402.10  
##                 cats_allowed1                coop_condocondo  
##                     13237.38                       195813.57  
##   dining_room_typedining area          dining_room_typeformal  
##                    -12854.59                         19282.43  
##         dining_room_typeother                   dogs_allowed1  
##                     18864.24                          6377.21  
##                fuel_typegas                    fuel_typeoil  
##                      7724.92                          7157.50  
##              fuel_typeother                  garage_exists1  
##                     47194.56                         13462.51  
##          kitchen_typeeat-in          kitchen_typeefficiency  
##                     -9506.04                        -21892.16  
##                num_bedrooms              num_full_bathrooms  
##                     31794.83                         74037.57  
##          num_half_bathrooms                 num_total_rooms  
##                     -8693.49                          5694.28  
##                   sq_footage                     walk_score.L  
##                        37.47                         -8519.54  
##                 walk_score.Q                    walk_score.C  
##                     61888.07                         -9866.65  
##                 walk_score^4                 zip_codesJamaica  
##                     45591.92                        -36085.44  
##        zip_codesNorth Queens        zip_codesNortheast Queens  
##                     45962.55                         44014.75  
##    zip_codesNorthwest Queens        zip_codesSoutheast Queens
```

```
##                     160123.29                         31995.52
##    zip_codesSouthwest Queens  zip_codesWest Central Queens
##                    -39895.82                         53642.44
##         zip_codesWest Queens                       total_cost
##                     51714.08                           155.96
##    approx_year_built_missing              cats_allowed_missing
##                     31849.71                               NA
##     dining_room_type_missing                fuel_type_missing
##                     -4597.05                          3219.34
##         kitchen_type_missing         maintenance_cost_missing
##                    -68239.88                        -24954.69
##           sq_footage_missing              condoCharges_missing
##                     -8167.27                         48631.03
```

```
#in-sample stats to report
lmModelSum <- summary(lmModel)
lmModelSum
```

```
##
## Call:
## lm(formula = train$sale_price ~ ., data = train %>% select(-sale_price))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -304722  -37652   -5250   38896  291323
##
## Coefficients: (1 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -871200.29  618314.59  -1.409 0.159647
## approx_year_built                402.10     317.17   1.268 0.205637
## cats_allowed1                  13237.38   11610.70   1.140 0.254955
## coop_condocondo               195813.57   14943.78  13.103  < 2e-16 ***
## dining_room_typedining area   -12854.59   56693.30  -0.227 0.820747
## dining_room_typeformal         19282.43   10272.25   1.877 0.061257 .
## dining_room_typeother          18864.24   13710.98   1.376 0.169670
## dogs_allowed1                   6377.21   12808.55   0.498 0.618849
## fuel_typegas                    7724.92   31767.65   0.243 0.808004
## fuel_typeoil                    7157.50   32295.12   0.222 0.824721
## fuel_typeother                 47194.56   42494.76   1.111 0.267437
## garage_exists1                 13462.51   10618.03   1.268 0.205605
## kitchen_typeeat-in             -9506.04   12179.22  -0.781 0.435570
## kitchen_typeefficiency        -21892.16   11940.86  -1.833 0.067520 .
## num_bedrooms                   31794.83    8898.93   3.573 0.000398 ***
## num_full_bathrooms             74037.57   14499.74   5.106 5.19e-07 ***
## num_half_bathrooms             -8693.49   20039.84  -0.434 0.664670
## num_total_rooms                 5694.28    6319.11   0.901 0.368089
## sq_footage                        37.47      14.38   2.606 0.009519 **
## walk_score.L                   -8519.54   53762.61  -0.158 0.874173
## walk_score.Q                   61888.07   45647.66   1.356 0.175967
## walk_score.C                   -9866.65   34027.89  -0.290 0.772005
```

```
## walk_score^4                    45591.92   21206.46   2.150 0.032187 *
## zip_codesJamaica               -36085.44   22569.27  -1.599 0.110671
## zip_codesNorth Queens           45962.55   18967.42   2.423 0.015844 *
## zip_codesNortheast Queens       44014.75   20323.41   2.166 0.030948 *
## zip_codesNorthwest Queens      160123.29   30136.70   5.313 1.83e-07 ***
## zip_codesSoutheast Queens       31995.52   22727.27   1.408 0.159998
## zip_codesSouthwest Queens      -39895.82   19734.84  -2.022 0.043912 *
## zip_codesWest Central Queens    53642.44   19750.58   2.716 0.006906 **
## zip_codesWest Queens            51714.08   20198.64   2.560 0.010841 *
## total_cost                        155.96      16.60   9.396  < 2e-16 ***
## approx_year_built_missing       31849.71   35437.75   0.899 0.369349
## cats_allowed_missing                  NA         NA      NA       NA
## dining_room_type_missing        -4597.05    9892.39  -0.465 0.642406
## fuel_type_missing                3219.34   19184.57   0.168 0.866822
## kitchen_type_missing           -68239.88   38940.93  -1.752 0.080504 .
## maintenance_cost_missing       -24954.69   19771.33  -1.262 0.207655
## sq_footage_missing              -8167.27    8523.15  -0.958 0.338542
## condoCharges_missing            48631.03   32307.49   1.505 0.133080
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 77530 on 384 degrees of freedom
## Multiple R-squared:  0.8228, Adjusted R-squared:  0.8053
## F-statistic: 46.93 on 38 and 384 DF,  p-value: < 2.2e-16

#RMSE IS
predictions <- lmModel %>% predict(train %>% select(-sale_price))
RMSE(predictions, train$sale_price)

## [1] 73867.23

R2(predictions, train$sale_price)

## [1] 0.8228376

#RMSE OOS
predictions <- lmModel %>% predict(test %>% select(-sale_price))
RMSE(predictions, test$sale_price)

## [1] 80213.91

R2(predictions, test$sale_price)

## [1] 0.8346043
```

Hyperparameter Tuning for random forest

```
#Random Forest MLR
housing_Xcomplete <- train %>% select(-sale_price)
y_salesprice <- train$sale_price

data = cbind(y_salesprice, housing_Xcomplete)
```

```
colnames(data)[1] = "sales_price"

task = makeRegrTask(data = data, target = "sales_price")

## Warning in makeTask(type = type, data = data, weights = weights, blocking
=
## blocking, : Empty factor levels were dropped for columns: dining_room_type

parms = makeParamSet(
    #Must have atleast 1 of everthing. Mtry cannot be larger than the number
of columns present
  makeIntegerParam("mtry", lower = 1, upper = ncol(housing_Xcomplete)),
  makeIntegerParam("ntree", lower = 1, upper = 1000),
  makeIntegerParam("nodesize", lower = 1, upper = 1000)
)

desc <- makeResampleDesc("Bootstrap", iters = 30)

ctrl <- makeTuneControlRandom(maxit = 30)

mlr_ret <- tuneParams("regr.randomForest", task = task, resampling = desc, pa
r.set = parms, control = ctrl, measures = list(rmse))

## [Tune] Started tuning learner regr.randomForest for parameter set:

##               Type len Def     Constr Req Tunable Trafo
## mtry       integer   -   -    1 to 24   -    TRUE     -
## ntree      integer   -   - 1 to 1e+03   -    TRUE     -
## nodesize integer   -   - 1 to 1e+03   -    TRUE     -

## With control class: TuneControlRandom

## Imputation value: Inf

## [Tune-x] 1: mtry=23; ntree=865; nodesize=645

## [Tune-y] 1: rmse.test.rmse=129545.0654666; time: 0.1 min

## [Tune-x] 2: mtry=4; ntree=707; nodesize=3

## [Tune-y] 2: rmse.test.rmse=83353.7686475; time: 0.2 min

## [Tune-x] 3: mtry=9; ntree=2; nodesize=334

## [Tune-y] 3: rmse.test.rmse=131649.3940247; time: 0.0 min

## [Tune-x] 4: mtry=17; ntree=352; nodesize=531

## [Tune-y] 4: rmse.test.rmse=127429.2035513; time: 0.0 min

## [Tune-x] 5: mtry=14; ntree=763; nodesize=595

## [Tune-y] 5: rmse.test.rmse=126318.0111683; time: 0.1 min
```

```
## [Tune-x] 6: mtry=7; ntree=598; nodesize=997
## [Tune-y] 6: rmse.test.rmse=128210.2057682; time: 0.0 min
## [Tune-x] 7: mtry=12; ntree=176; nodesize=19
## [Tune-y] 7: rmse.test.rmse=82510.6222085; time: 0.1 min
## [Tune-x] 8: mtry=5; ntree=466; nodesize=77
## [Tune-y] 8: rmse.test.rmse=94428.9631364; time: 0.1 min
## [Tune-x] 9: mtry=6; ntree=174; nodesize=771
## [Tune-y] 9: rmse.test.rmse=129684.4163061; time: 0.0 min
## [Tune-x] 10: mtry=10; ntree=209; nodesize=627
## [Tune-y] 10: rmse.test.rmse=126164.3897233; time: 0.0 min
## [Tune-x] 11: mtry=1; ntree=310; nodesize=68
## [Tune-y] 11: rmse.test.rmse=125641.3085135; time: 0.0 min
## [Tune-x] 12: mtry=5; ntree=400; nodesize=739
## [Tune-y] 12: rmse.test.rmse=131835.4844586; time: 0.0 min
## [Tune-x] 13: mtry=22; ntree=675; nodesize=451
## [Tune-y] 13: rmse.test.rmse=129109.7189636; time: 0.1 min
## [Tune-x] 14: mtry=11; ntree=395; nodesize=243
## [Tune-y] 14: rmse.test.rmse=105316.7349310; time: 0.0 min
## [Tune-x] 15: mtry=8; ntree=924; nodesize=430
## [Tune-y] 15: rmse.test.rmse=126941.2995949; time: 0.1 min
## [Tune-x] 16: mtry=11; ntree=918; nodesize=126
## [Tune-y] 16: rmse.test.rmse=98838.8128998; time: 0.1 min
## [Tune-x] 17: mtry=6; ntree=609; nodesize=682
## [Tune-y] 17: rmse.test.rmse=129266.3946418; time: 0.0 min
## [Tune-x] 18: mtry=9; ntree=22; nodesize=311
## [Tune-y] 18: rmse.test.rmse=113904.1910302; time: 0.0 min
## [Tune-x] 19: mtry=20; ntree=108; nodesize=821
## [Tune-y] 19: rmse.test.rmse=128749.3162588; time: 0.0 min
```

```
## [Tune-x] 20: mtry=11; ntree=728; nodesize=879

## [Tune-y] 20: rmse.test.rmse=125797.3869341; time: 0.1 min

## [Tune-x] 21: mtry=22; ntree=884; nodesize=848

## [Tune-y] 21: rmse.test.rmse=129195.8047492; time: 0.1 min

## [Tune-x] 22: mtry=13; ntree=182; nodesize=458

## [Tune-y] 22: rmse.test.rmse=125919.5809180; time: 0.0 min

## [Tune-x] 23: mtry=22; ntree=465; nodesize=154

## [Tune-y] 23: rmse.test.rmse=102967.4489259; time: 0.1 min

## [Tune-x] 24: mtry=1; ntree=738; nodesize=18

## [Tune-y] 24: rmse.test.rmse=120192.8752707; time: 0.1 min

## [Tune-x] 25: mtry=2; ntree=136; nodesize=83

## [Tune-y] 25: rmse.test.rmse=106609.0524688; time: 0.0 min

## [Tune-x] 26: mtry=19; ntree=695; nodesize=845

## [Tune-y] 26: rmse.test.rmse=128313.7575757; time: 0.1 min

## [Tune-x] 27: mtry=8; ntree=307; nodesize=162

## [Tune-y] 27: rmse.test.rmse=101865.1278355; time: 0.0 min

## [Tune-x] 28: mtry=22; ntree=806; nodesize=504

## [Tune-y] 28: rmse.test.rmse=129190.8710698; time: 0.1 min

## [Tune-x] 29: mtry=16; ntree=855; nodesize=357

## [Tune-y] 29: rmse.test.rmse=125628.7978122; time: 0.1 min

## [Tune-x] 30: mtry=12; ntree=17; nodesize=499

## [Tune-y] 30: rmse.test.rmse=127170.7898171; time: 0.0 min

## [Tune] Result: mtry=12; ntree=176; nodesize=19 : rmse.test.rmse=82510.6222
085

#Optimal hyperparameter result
mlr_ret$x

## $mtry
## [1] 12
##
## $ntree
## [1] 176
```

```
##
## $nodesize
## [1] 19
```

RandomForest Model

```
#Model
rfModel = randomForest(housing_Xcomplete, y_salesprice, mtry = as.integer(mlr
_ret$x[1]), num_trees = as.integer(mlr_ret$x[2]), nodesize = as.integer(mlr_r
et$x[3]))
rfModel

##
## Call:
##  randomForest(x = housing_Xcomplete, y = y_salesprice, mtry = as.integer(m
lr_ret$x[1]),      nodesize = as.integer(mlr_ret$x[3]), num_trees = as.intege
r(mlr_ret$x[2]))
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 6032153605
##                     % Var explained: 80.41

yhat = predict(rfModel,  train %>% select(-sale_price))
is_rmse = sqrt(mean((train$sale_price - yhat)^2))
is_rsq = 1 - sum((train$sale_price - yhat)^2)/sum((train$sale_price - mean(y_
salesprice))^2)
is_rmse

## [1] 51096.27

is_rsq

## [1] 0.9152292

#Compute errors using model of entire dataset
#Once this is evaluated, there is no going back, otherwise it is cheating!
#Run and submit, there is no going back.
yhat = predict(rfModel,  test %>% select(-sale_price))
oos_rmse = sqrt(mean((test$sale_price - yhat)^2))
oos_rsq = 1 - sum((test$sale_price - yhat)^2)/sum((test$sale_price - mean(y_s
alesprice))^2)
oos_rmse

## [1] 76131.56

oos_rsq

## [1] 0.8462672
```