Predicting and Examining Apartment Prices Results in Queens New York 2016-2017 Edition

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

By Enoch Kim

In collaboration with:
Kennly Weerasinghe
Sara Jedwab
Hubert Majewski
Marin Azhar

Abstract

For this final project, the goal is to create and summarize a report to predict models for sale prices in Queens, New York. I will be using the Queens housing market data from February 2016 to February 2017 in order to create three predictive models. They are Regression Tree Model, Ordinary Least Squares Model, and the Random Forest Model. Between the three predictive models that will be built, we hope to get a better understand and accuracy of the Queens housing market sale prices that may apply to the real word.

1. Introduction

I will be using a housing market data set that was collected from February 2016 to February 2017 and within this data set , I will make sale price predictions for condos and co-ops up to a maximum sale price of $1,000,000 located in Queens, New York.

The term “Models” in the world of science are known for approximations, abstractions to reality, absolute truth, systems, and phenomena. A model is still useful for practical purposes even though the model contains errors and is only an approximation within a certain reality. In order for a model to be useful, it needs to capture the phenomenon and measured features, settings of the system, and a useful approximation of reality. In order for a model to attain this usefulness, the model must establish proper metrics because it numerically gauges settings and the phenomena. For this project I will be taking the data that was provided and attempt to create a model in order to have better accurate prediction for the sale prices than the ones that were already provided. From the data, there will be a set of features that will be used to predict a sale prices of each property and there will be three different types of models that will be created. The following models are Regression, Linear and Random Forest Modeling. Each model will provide performances results for the sale price predictions.

2. The Data

The data that I used is from MLSI (Multiple Listing Service of Long Island Inc). This data contains 2230 rows and 55 columns of housing observations that was collected from February 2016 to February 2017. First, I viewed the data using microsoft excel and noticed that a many of the columns were not feasible for this assignment, also there were a lot of NA values within the data. The column that stood out to me the most was sale_price, since the assignment was to build a prediction model for the sale price of apartments in Queens. However, out of the

2230 rows from sale_price, only 528 rows had values that could be used for building a prediction model. After viewing the raw data, I proceeded to clean the data and dealt with the NA values by dropping them and/or imputing the missing data.

2.2 Featurization

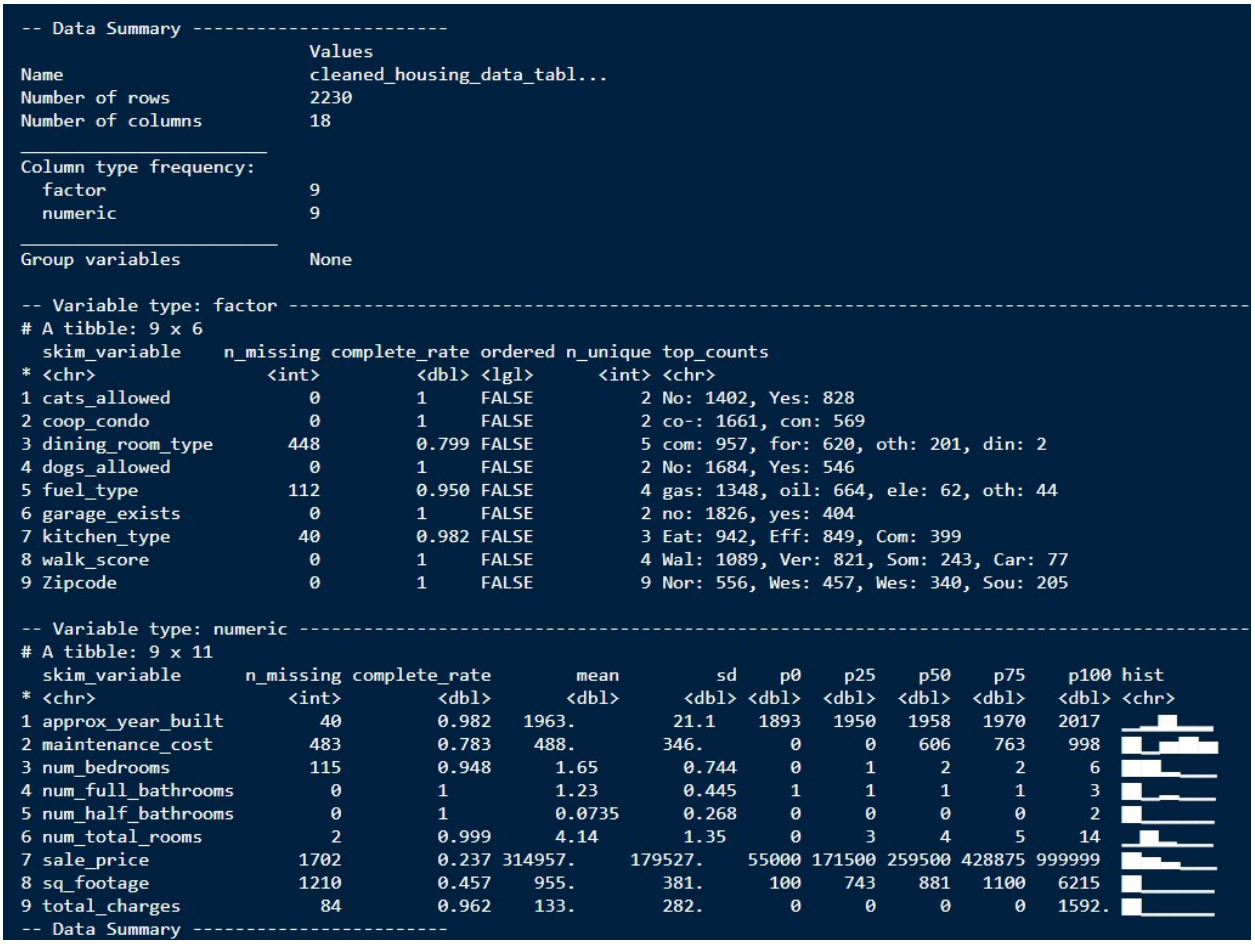
Out of the 55 features that were originally from the raw data, I decided to keep 19 features from the original dataset. The 19 features that were kept were, approx_year_built, cats_allowed, common_charges, coop_condo, dining_room_type, dogs_allowed, fuel_type, garage_exists, kitchen_type, maintenance_cost, num_bedrooms, num_full_bathrooms, num_half_bathrooms, num_total_rooms, total_taxes, sale_price, sq_footage, walk_score, URL(zipcode). The predicted feature that I had chosen and set to y, was sale_price because the main focus on the final project was to make predictions prices for property in Queens NY.

In the dataset, there are numeric variables and categorial variables. The numeric variables for this dataset are, approx_year_built, maintenance_cost, num_bedrooms, num_full_bathrooms, num_half_bathrooms, num_total_rooms, total_charges, and sale_price, sq_footage.

The approx_year_built is the year the building/property was build, its range is from 1893 to 2017. maintenance_cost is cost for building type of condos. num_bedrooms is the total number of bedrooms which ranges from 0 to 3. num_full_bathrooms is the total of number of full bathrooms which ranges from 1 to 3. num_half_bathrooms is the total of number of full bathrooms which ranges from 0 to 2, while cleaning the data the NA were converted to 0. num_total_rooms is the total number of rooms in the building which ranges from 1 to 8. total_charges is the overall charge between total_taxes and common_charges. sale_price is the total value of the building. sq_footage is the overall area number of the square footage in the building. I decided to take the values between total_taxes and common_charges and add them together and create a new column called total_charges because of the difference in cost amongst the two the building type, condo and coop.

The categorial variables for this dataset are cats_allowed, coop_condo, dining room_type, dogs_allowed, fuel_type, garage_exists, kitchen_type, walk_score, URL(zipcode).

The cat_allowed and dogs_allowed is a factor that contains two categories which are Yes or No. The coop_condo is a factor that contains two categories which are coop or condo. dining room_type is a factor that contains three categories, and they are combo, dining area, formal and others. fuel_type is a factor that contains four categories, and they are electric, gas, oil and others. garage_exists is a factor that contains two categories which is no or yes. kitchen_type, is a factor that contains four categories, and they are Efficient, Combo and Eat_In. walk_score is a factor that first had a numerical value but later factored into four categories. Depending on the walk_score the four categories are Car-Dependent, Somewhat Walkable, Very Walkable and Walker's Paradise (Walk Score. n.d.). URL, I first extracted the five-digit zipcode and renamed column to zipcode, and then later converted into a factor that contains nine categories. Depending on the zipcode, the categories are Northeast Queens, North Queens, Central Queens, Jamaica, Northwest Queens, West Central Queens, Southeast Queens, Southwest Queens, and West Queens. These were the feature that stood out to me the most in order to create a feasible and efficient model for the final project.



```
-- Data Summary ------------------------
                           Values
Name                       cleaned_housing_data_tabl...
Number of rows             2230
Number of columns          18
_______________________
Column type frequency:
  factor                   9
  numeric                  9
________________________
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 No: 1402, Yes: 828
2 coop_condo               0         1     FALSE          2 co-: 1661, con: 569
3 dining_room_type       448         0.799 FALSE          5 com: 957, for: 620, oth: 201, din: 2
4 dogs_allowed             0         1     FALSE          2 No: 1684, Yes: 546
5 fuel_type              112         0.950 FALSE          4 gas: 1348, oil: 664, ele: 62, oth: 44
6 garage_exists            0         1     FALSE          2 no: 1826, yes: 404
7 kitchen_type            40         0.982 FALSE          3 Eat: 942, Eff: 849, Com: 399
8 walk_score               0         1     FALSE          4 Wal: 1089, Ver: 821, Som: 243, Car: 77
9 Zipcode                  0         1     FALSE          9 Nor: 556, Wes: 457, Wes: 340, Sou: 205

-- Variable type: numeric -------------------------------------------------------------------------------
# A tibble: 9 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         40         0.982   1963.       21.1  1893   1950   1958   1970   2017 ▁▂▇▁▁
2 maintenance_cost         483         0.783    488.      346.      0      0    606    763    998 ▇▁▅▆▅
3 num_bedrooms             115         0.948      1.65      0.744    0      1      2      2      6 ▇▇▂▁▁
4 num_full_bathrooms         0         1          1.23      0.445    1      1      1      1      3 ▇▁▂▁▁
5 num_half_bathrooms         0         1          0.0735    0.268    0      0      0      0      2 ▇▁▁▁▁
6 num_total_rooms            2         0.999      4.14      1.35     0      3      4      5     14 ▁▇▂▁▁
7 sale_price              1702         0.237 314957.    179527.  55000 171500 259500 428875 999999 ▇▅▃▁▁
8 sq_footage              1210         0.457    955.      381.     100    743    881   1100   6215 ▇▁▁▁▁
9 total_charges             84         0.962    133.      282.       0      0      0      0  1592. ▇▁▁▁▁
-- Data Summary ------------------------
```

2.3 Errors and Missingness

When first viewing the raw data, I noticed there were a decent amount of NA's and some of data entries seem to have errors. The feature kitchen_type had entries that were misspelled such as efficiemcy and efficiency kitchene. dogs_allowed had an entry of yes89 instead of yes and cat_allowed had an entry of y instead of yes. The NA's entries, depending on the features, they were either dropped or imputed. An example would be, the feature num_half_bathrooms, if the entry contained NA due to no response, it was converted to 0. The rest of the missing values were dealt with by adding an "is_missing_ column and due to that, an additional 10 feature columns were added to the dataset. The data was imputed through the library R package called missForest.

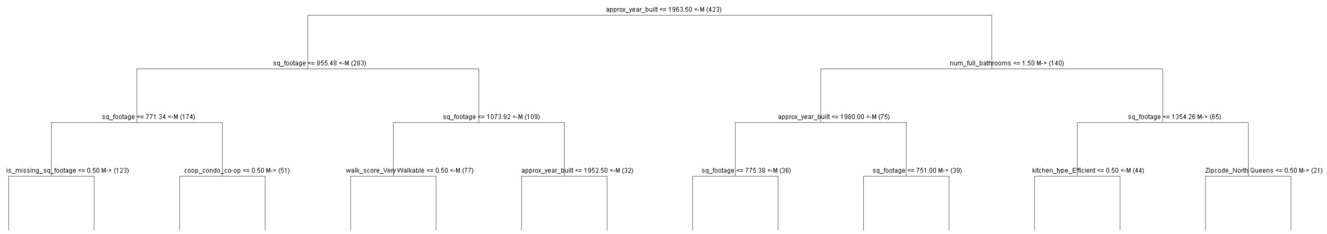
3. Modeling

After cleaning the raw data and imputing the data, the data went through training and testing on where sale_price was not NA. I created three models, they are Regression Tree Modeling, Linear Modeling and Random Forest Modeling. The goal of the models were to predicted the sale_price of apartments located in Queens.

3.1 Regression Tree Modeling

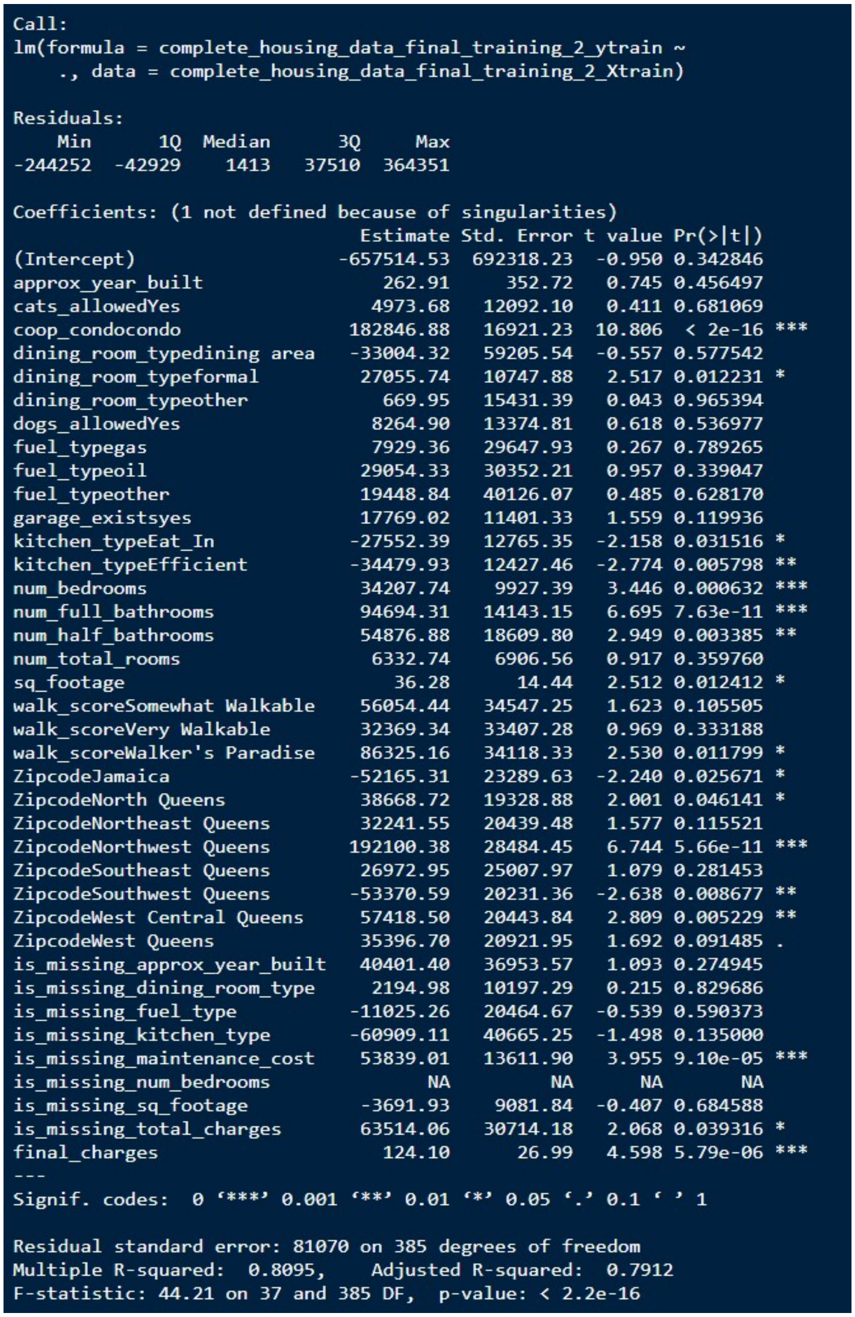
The Regression Tree Modeling is known as to be an algorithm that is based on creating split-based features, in this case the sale_price would be the first feature since that is our y in our model. The figure shown below is the result of the Regression Tree Model in four layers. Starting with the root node which considered the most important node, the split starts with the feature with approx_year_build less than or equal to 1963. The reason for this is that, usually the starting point on determining the price of an apartment in Queens is when year it was built. Moving from the root node and going left of the tree, the next feature is sq_footage less than equal 871.83. Continuing down the leaf node, the next feature that shows up is the sq_footage on both the left and right side, but the only difference is the size of the sq_footage. This shows that the next important feature when predicting the price of an apartment in Queens is the size of the area. After seeing the first three levels of the left side, the two most important features are the year the place was build and the square area of the apartment.

Going back to the root node and if the building was built after 1963, the next important feature on the right side is the num_full_bathrooms less than equal 1.5. The next split between the num_full_bathrooms is the left node approx_year_built less than equal 1979 and the right node being the sq_footage less than equal to 1135.10. The first three levels of the right side of the tree, it can be determined that the most important features when predicting an apartment price in Queens is the year it was built, the number of bathrooms and the square footage. The in-sample is, R^2 is 78% and RMSE is 38211.04. Overall, after modeling the Regression Tree and viewing both sides, it can be stated that the approx_year_built, num_full_bathrooms and sq_footage are the most important features for this model. Also, the results of the figure show that the top features are sq_footage, approx_year_built, final_charges, num_full_bathroom, coop_condo, kitchen_type_Eat_in, walk_score_Paradise, kitchen_type_Effcient, and Zipcode_North Queens


approx_year_built <= 1963.50 <-M (423)
sq_footage <= 855.48 <-M (283)
num_full_bathrooms <= 1.50 M-> (140)
sq_footage <= 771.34 <-M (174)
sq_footage <= 1073.92 <-M (109)
approx_year_built <= 1980.00 <-M (75)
sq_footage <= 1354.26 M-> (65)
is_missing_sq_footage <= 0.50 M-> (123)
coop_condo_co-op <= 0.50 M-> (51)
walk_score_Very Walkable <= 0.50 <-M (77)
approx_year_built <= 1952.50 <-M (32)
sq_footage <= 775.38 <-M (36)
sq_footage <= 751.00 M-> (39)
kitchen_type_Efficient <= 0.50 <-M (44)
Zipcode_North Queens <= 0.50 M-> (21)

3.2 Linear Modeling

The next model that was created was the Linear Model and the type of model that was fitted was the vanilla OLS linear model. After creating and running the OLS linear model, the in-sample result for R^2 is 80.9% and the RMSE is $81,069.79 and the out-of-sample result for R^2 is 51.4% and the RMSE is $91,253.66. There is a huge drop off for R^2 between the in-sample and out-of-sample. According to the summary after running the model, the feature that had the most impact was the Zip-code location Northwest Queens followed by coop or condo. The features that had the worst impact were Zip-code location Jamaica and Zip-code location Southwest Queens. From this Linear Model, Zip-code location seems to have one the biggest impacts whether it is positive or negative. These results are surprising since the Regression Tree Model had shown the year the apartment was build and the square footage of the apartment. Between the two models that had ran so far, the Linear Model performed better than the Regression Tree Model, as expected.



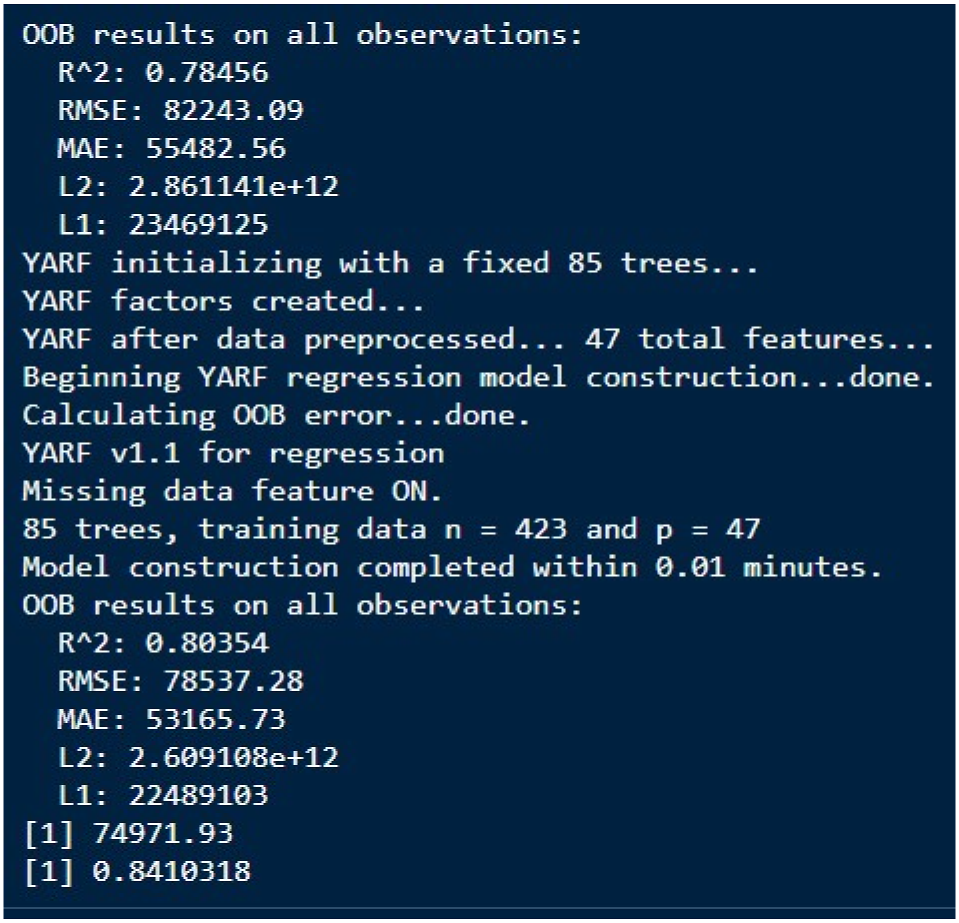
```
Call:
lm(formula = complete_housing_data_final_training_2_ytrain ~ 
    ., data = complete_housing_data_final_training_2_Xtrain)

Residuals:
    Min      1Q  Median      3Q     Max 
-244252  -42929    1413   37510  364351 

Coefficients: (1 not defined because of singularities)
                               Estimate Std. Error t value Pr(>|t|)    
(Intercept)                  -657514.53  692318.23  -0.950 0.342846    
approx_year_built                262.91     352.72   0.745 0.456497    
cats_allowedYes                 4973.68   12092.10   0.411 0.681069    
coop_condocondo               182846.88   16921.23  10.806  < 2e-16 ***
dining_room_typedining area   -33004.32   59205.54  -0.557 0.577542    
dining_room_typeformal         27055.74   10747.88   2.517 0.012231 *  
dining_room_typeother            669.95   15431.39   0.043 0.965394    
dogs_allowedYes                 8264.90   13374.81   0.618 0.536977    
fuel_typegas                    7929.36   29647.93   0.267 0.789265    
fuel_typeoil                   29054.33   30352.21   0.957 0.339047    
fuel_typeother                 19448.84   40126.07   0.485 0.628170    
garage_existsyes               17769.02   11401.33   1.559 0.119936    
kitchen_typeEat_In            -27552.39   12765.35  -2.158 0.031516 *  
kitchen_typeEfficient         -34479.93   12427.46  -2.774 0.005798 ** 
num_bedrooms                   34207.74    9927.39   3.446 0.000632 ***
num_full_bathrooms             94694.31   14143.15   6.695 7.63e-11 ***
num_half_bathrooms             54876.88   18609.80   2.949 0.003385 ** 
num_total_rooms                 6332.74    6906.56   0.917 0.359760    
sq_footage                        36.28      14.44   2.512 0.012412 *  
walk_scoreSomewhat Walkable    56054.44   34547.25   1.623 0.105505    
walk_scoreVery Walkable        32369.34   33407.28   0.969 0.333188    
walk_scoreWalker's Paradise    86325.16   34118.33   2.530 0.011799 *  
ZipcodeJamaica                -52165.31   23289.63  -2.240 0.025671 *  
ZipcodeNorth Queens            38668.72   19328.88   2.001 0.046141 *  
ZipcodeNortheast Queens        32241.55   20439.48   1.577 0.115521    
ZipcodeNorthwest Queens       192100.38   28484.45   6.744 5.66e-11 ***
ZipcodeSoutheast Queens        26972.95   25007.97   1.079 0.281453    
ZipcodeSouthwest Queens       -53370.59   20231.36  -2.638 0.008677 ** 
ZipcodeWest Central Queens     57418.50   20443.84   2.809 0.005229 ** 
ZipcodeWest Queens             35396.70   20921.95   1.692 0.091485 .  
is_missing_approx_year_built   40401.40   36953.57   1.093 0.274945    
is_missing_dining_room_type     2194.98   10197.29   0.215 0.829686    
is_missing_fuel_type          -11025.26   20464.67  -0.539 0.590373    
is_missing_kitchen_type       -60909.11   40665.25  -1.498 0.135000    
is_missing_maintenance_cost    53839.01   13611.90   3.955 9.10e-05 ***
is_missing_num_bedrooms              NA         NA      NA       NA    
is_missing_sq_footage          -3691.93    9081.84  -0.407 0.684588    
is_missing_total_charges       63514.06   30714.18   2.068 0.039316 *  
final_charges                    124.10      26.99   4.598 5.79e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 81070 on 385 degrees of freedom
Multiple R-squared:  0.8095,	Adjusted R-squared:  0.7912 
F-statistic: 44.21 on 37 and 385 DF,  p-value: < 2.2e-16
```

3.3 Random Forest Modeling

The final model that was used for this final project was Random Forest Modeling. Random Forest Modeling is a non-parametric algorithm that constructs multiple decision trees and eventually combines all of them together to provide a prediction which is more accurate than the first two models that were used. It uses the method called bootstrap aggregation, which is also known as "bagging" to predict an out-of-sample tends to have better accuracy. It also cross validates the data which is added bonus while also giving a discount on the variance without increasing the bias. Out of the three Models, the Random Forest Modeling should have the best prediction price for the apartments located in Queens.

4. Performance Results for your Random Forest Model

The performance results for the Random Forest Model out-of-bag result was, R^2 is 80.5% and the RMSE is $78091.14. This result is similar but slightly worse than the Linear model. This is surprising considering that the Random Forest Model is known to outperform the other two models. We then conduct the out-of-sample for the final model and the results were, R^2 is 84.1% and RMSE is $72220.06. Overall, these results were great because the results for the out-of-sample results for the Linear Model was, R^2 was 51.4% and the RMSE was $91,253.66 while the Regression Tree Model results were, R^2 was 48.6% and the RMSE was $96,560.07. The models performed better as expected when it came to the out-of-sample results.



```
OOB results on all observations:
  R^2: 0.78456
  RMSE: 82243.09
  MAE: 55482.56
  L2: 2.861141e+12
  L1: 23469125
YARF initializing with a fixed 85 trees...
YARF factors created...
YARF after data preprocessed... 47 total features...
Beginning YARF regression model construction...done.
Calculating OOB error...done.
YARF v1.1 for regression
Missing data feature ON.
85 trees, training data n = 423 and p = 47
Model construction completed within 0.01 minutes.
OOB results on all observations:
  R^2: 0.80354
  RMSE: 78537.28
  MAE: 53165.73
  L2: 2.609108e+12
  L1: 22489103
[1] 74971.93
[1] 0.8410318
```

5. Discussion

The main goal of this final project was to create a model and predict the apartment prices in Queens. With the raw data that was collected by MILSI (Multiple Listing Service of Long Island Inc), I first cleaned the data, used the features which I believe was valid, in order to created three model and summarized the results of the model in detail. Regression Tree model performed the worst out the three as expect but when running the Linear Model and the Random Forest Model, Linear Modeling had performed slightly better than Random Forest Model which was a bit unexpected. However, I do believe my model is production ready due the end results for the out-of-sample numbers for all three models. Also, I believe that the modeling part fell a bit short because normally, the Random Forest Model is supposed to perform better than Linear

Model, but this was not the case. We might be able to plug in these holes by cleaning data in a more efficient way and/or selecting different and perhaps better features for the models.

Acknowledgments

I would like to give an acknowledgment with the people I collaborated with, Kennly Weerasinghe, Sara Jedwab, Hubert Majewski and Marin Azhar. Each played a role in helping understand and reviewing the different modeling types with this project, assisted me in picking certain features and cleaning the data.

References

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.

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

Code Appendix

Math 342W Final Project

Enoch Kim

11:59PM May 25, 2021

```
pacman::p_load(dplyr,magrittr,stringr, skimr)

#Importing the Housing data
housing_data = read.csv("C:\\Users\\Enoch Kim\\Desktop\\Spring 2021\\Math 342
w\\myGit\\Math342W_Spring21_QC\\final_project\\housing_data_2016_2017.csv")
#View(housing_data)

#Feature Selection, selecting the columns I'll be using for the model
cleaned_housing_data = housing_data %<>%
  select(approx_year_built ,cats_allowed, common_charges ,coop_condo, dining_
room_type, dogs_allowed, fuel_type, garage_exists, kitchen_type, maintenance_
cost, num_bedrooms, num_full_bathrooms, num_half_bathrooms, num_total_rooms,
total_taxes ,sale_price, sq_footage, walk_score, URL)

#View(cleaned_housing_data)
```

```
#Cleaning the Feature Selection, so I can run the algorithms and create a
model

cleaned_housing_data_table = cleaned_housing_data %>%

  mutate(cats_allowed = as.factor(ifelse(cats_allowed == "no", "No", "Yes")))
%>%

  mutate(common_charges = as.numeric(gsub('[$,]', '', common_charges))) %>%

  mutate(common_charges = as.numeric(ifelse(coop_condo == "co_op",
replace(common_charges, is.na(common_charges), 0), common_charges))) %>%

  mutate(coop_condo = as.factor(coop_condo)) %>%

  mutate(dining_room_type = as.factor(dining_room_type)) %>%

  mutate(dogs_allowed = as.factor(ifelse(dogs_allowed == "no", "No", "Yes")))
%>%

  mutate(fuel_type = as.factor(ifelse(fuel_type == "none" | fuel_type ==
"Other", "other", fuel_type))) %>%

  mutate(garage_exists = as.factor(ifelse(is.na(garage_exists), "no",
", "yes"))) %>%

  mutate(kitchen_type = as.factor(case_when(kitchen_type == "Combo" |
kitchen_type == "combo" ~ "Combo",
                                            kitchen_type == "eat in" |
kitchen_type == "eatin" | kitchen_type == "Eat In" | kitchen_type == "Eat in"
~ "Eat_In",
                                            kitchen_type == "efficiemcy" |
kitchen_type == "efficiency" | kitchen_type == "efficiency kitchen" |
                                            kitchen_type == "efficiency
kitchene" | kitchen_type == "efficiency ktchen" ~ "Efficient"))) %>%

  mutate(maintenance_cost = as.numeric(str_remove_all(maintenance_cost,
"[$]"))) %>%

  mutate(maintenance_cost = ifelse(coop_condo == "condo",
replace(maintenance_cost, is.na(maintenance_cost), 0), maintenance_cost)) %>%

  mutate(num_half_bathrooms = replace(num_half_bathrooms,
is.na(num_half_bathrooms), 0)) %>%

  mutate(total_taxes = as.numeric(gsub('[$,]', '', total_taxes))) %>%

  mutate(total_taxes = replace(total_taxes, is.na(total_taxes), 0)) %>%

  mutate(sale_price = as.numeric(gsub('[$,]', '', sale_price))) %>%

  #Got score from this site,
https://www.walkscore.com/methodology.shtml#:~:text=Walk%20Score%20measures%2
0the%20walkability%20of%20any%20address%20using%20a,miles)%20are%20given%20ma
ximum%20points

  mutate(walk_score = as.factor(case_when(
    walk_score <= 24 ~ "Car-Dependent",
    walk_score > 24 & walk_score < 50 ~ "Car-Dependent" ,
    walk_score > 49 & walk_score < 70 ~ "Somewhat Walkable",
    walk_score > 69 & walk_score < 90 ~ "Very Walkable",
    walk_score > 89 & walk_score <= 100 ~ "Walker's Paradise"))) %>%

  rename(Zipcode = URL) %>%

  mutate(Zipcode = as.numeric(str_remove(str_sub(Zipcode, start = -15, end =
-10), pattern = "-"))) %>%

  mutate(Zipcode = as.factor(case_when(
    Zipcode == "11361" | Zipcode == "11362" | Zipcode == "11363" | Zipcode ==
"11364" ~ "Northeast Queens",
    Zipcode == "11354" | Zipcode == "11355" | Zipcode == "11356" | Zipcode == "11357" | Zipcode == "11358" | Zipcode == "11359" | Zipcode == "11360" ~ "North Queens",
    Zipcode == "11365" | Zipcode == "11366" | Zipcode == "11367" ~ "Central Queens",
    Zipcode == "11412" | Zipcode == "11423" | Zipcode == "11432" | Zipcode == "11433" | Zipcode == "11434" | Zipcode == "11435" | Zipcode == "11436" ~ "Jamaica",
    Zipcode == "11101" | Zipcode == "11102" | Zipcode == "11103" | Zipcode == "11104" | Zipcode == "11105" | Zipcode == "11106"~ "Northwest Queens",
    Zipcode == "11374" | Zipcode == "11375" | Zipcode == "11379" | Zipcode == "11385" ~ "West Central Queens",
    Zipcode == "11004" | Zipcode == "11005" | Zipcode == "11411" | Zipcode == "11413" | Zipcode == "11422" | Zipcode == "11426" | Zipcode == "11427" | Zipcode == "11428" | Zipcode == "11429"~ "Southeast Queens",
    Zipcode == "11414" | Zipcode == "11415" | Zipcode == "11416" | Zipcode == "11417" | Zipcode == "11418" | Zipcode == "11419" | Zipcode == "11420" | Zipcode == "11421" ~  "Southwest Queens",
    Zipcode == "11368" | Zipcode == "11369" | Zipcode == "11370" | Zipcode == "11372" | Zipcode == "11373" | Zipcode == "11377" | Zipcode == "11378"  ~ "West Queens",
    TRUE ~ "Other" ))) %>%

  mutate(total_charges = ifelse(coop_condo == "condo", (common_charges + (total_taxes/12)), 0)) %>%

  select(-total_taxes, -common_charges)
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
skim(cleaned_housing_data_table)
```

Data summary

Name	cleaned_housing_data_tabl...
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	No: 1402, Yes: 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	No: 1684, Yes: 546
fuel_type	112	0.95	FALSE	4	gas: 1348, oil: 664, ele: 62, oth: 44
garage_exists	0	1.00	FALSE	2	no: 1826, yes: 404
kitchen_type	40	0.98	FALSE	3	Eat: 942, Eff: 849, Com: 399
walk_score	0	1.00	FALSE	4	Wal: 1089, Ver: 821, Som: 243, Car: 77
Zipcode	0	1.00	FALSE	9	Nor: 556, Wes: 457, Wes: 340, 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	483	0.78	488.26	345.85	0	0	606	763	998.00	▇▁▆▇▅
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	▇▁▁▁▁
total_charges	84	0.96	133.49	281.76	0	0	0	0	1591.67	▇▁▁▁▁

```
#View(cleaned_housing_data_table)

#write.csv(cleaned_housing_data_table, "C:\\Users\\Enoch Kim\\Desktop\\Spring
2021\\Math 342w\\myGit\\Math342W_Spring21_QC\\final_project\\cleaned_housing_
data_table.csv")

#Since there are missing values, I will deal with the missing data by adding
dummy features and create is_missing columns.
pacman::p_load(tidyverse, missForest)
set.seed(1989)
missing_data = tbl_df(apply(is.na(cleaned_housing_data_table), 2, as.numeric)
)

## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.

colnames(missing_data) = paste("is_missing_", colnames(cleaned_housing_data_t
able), sep = "")
missing_data = tbl_df(t(unique(t(missing_data))))
```

```r
missing_data %<>%
  select_if(function(x){sum(x) > 0})

skim(missing_data)
```

Data summary

Name	missing_data
Number of rows	2230
Number of columns	10
_______________________	
Column type frequency:	
numeric	10
________________________	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
is_missing_approx_year_built	0	1	0.02	0.13	0	0	0	0	1	▇▁▁▁▁▁
is_missing_dining_room_type	0	1	0.20	0.40	0	0	0	0	1	▇▁▁▁▁▂
is_missing_fuel_type	0	1	0.05	0.22	0	0	0	0	1	▇▁▁▁▁▁
is_missing_kitchen_type	0	1	0.02	0.13	0	0	0	0	1	▇▁▁▁▁▁
is_missing_maintenance_cost	0	1	0.22	0.41	0	0	0	0	1	▇▁▁▁▁▂
is_missing_num_bedrooms	0	1	0.05	0.22	0	0	0	0	1	▇▁▁▁▁▁

is_missing_num_total_rooms	0	1	0.00	0.03	0	0	0	0	1	▇▁▁▁▁
is_missing_sale_price	0	1	0.76	0.43	0	1	1	1	1	▂▁▁▁▇
is_missing_sq_footage	0	1	0.54	0.50	0	0	1	1	1	▇▁▁▁▇
is_missing_total_charges	0	1	0.04	0.19	0	0	0	0	1	▇▁▁▁▁

```
final_cleaned_housing_data_table = cbind(missing_data, cleaned_housing_data_t
able)
skim(final_cleaned_housing_data_table)
```

Data summary

Name	final_cleaned_housing_dat…
Number of rows	2230
Number of columns	28
_______________________	
Column type frequency:	
factor	9
numeric	19
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1.00	FALSE	2	No: 1402, Yes: 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	No: 1684, Yes: 546
fuel_type	112	0.95	FALSE	4	gas: 1348, oil: 664, ele: 62, oth: 44
garage_exists	0	1.00	FALSE	2	no: 1826, yes: 404

kitchen_type	40	0.98	FALSE	3	Eat: 942, Eff: 849, Com: 399
walk_score	0	1.00	FALSE	4	Wal: 1089, Ver: 821, Som: 243, Car: 77
Zipcode	0	1.00	FALSE	9	Nor: 556, Wes: 457, Wes: 340, Sou: 205

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
is_missing_approx_year_built	0	1.00	0.02	0.13	0	0	0	0	1.00	▇▁▁▁▁
is_missing_dining_room_type	0	1.00	0.20	0.40	0	0	0	0	1.00	▇▁▁▁▂
is_missing_fuel_type	0	1.00	0.05	0.22	0	0	0	0	1.00	▇▁▁▁▁
is_missing_kitchen_type	0	1.00	0.02	0.13	0	0	0	0	1.00	▇▁▁▁▁
is_missing_maintenance_cost	0	1.00	0.22	0.41	0	0	0	0	1.00	▇▁▁▁▂
is_missing_num_bedrooms	0	1.00	0.05	0.22	0	0	0	0	1.00	▇▁▁▁▁

is_missing_num_total_rooms	0	1.00	0.00	0.03	0	0	0	0	1.00	▇▁▁▁▁
is_missing_sale_price	0	1.00	0.76	0.43	0	1	1	1	1.00	▂▁▁▁▇
is_missing_sq_footage	0	1.00	0.54	0.50	0	0	1	1	1.00	▇▁▁▁▇
is_missing_total_charges	0	1.00	0.04	0.19	0	0	0	0	1.00	▇▁▁▁▁
approx_year_built	40	0.98	1962.71	21.08	1893	1950	1958	1970	2017.00	▁▂▇▂▂
maintenance_cost	483	0.78	488.26	345.85	0	0	606	763	998.00	▇▁▅▇▅
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	▇▁▁▁▁
total_charges	84	0.96	133.49	281.76	0	0	0	0	1591.67	▇▁▁▁▁

```
summary(final_cleaned_housing_data_table)

##  is_missing_approx_year_built is_missing_dining_room_type is_missing_fuel_
type
##  Min.   :0.00000              Min.   :0.0000              Min.   :0.00000
##  1st Qu.:0.00000              1st Qu.:0.0000              1st Qu.:0.00000
##  Median :0.00000              Median :0.0000              Median :0.00000
##  Mean   :0.01794              Mean   :0.2009              Mean   :0.05022
##  3rd Qu.:0.00000              3rd Qu.:0.0000              3rd Qu.:0.00000
##  Max.   :1.00000              Max.   :1.0000              Max.   :1.00000
##
##  is_missing_kitchen_type is_missing_maintenance_cost is_missing_num_bedroo
ms
##  Min.   :0.00000         Min.   :0.0000              Min.   :0.00000
##  1st Qu.:0.00000         1st Qu.:0.0000              1st Qu.:0.00000
##  Median :0.00000         Median :0.0000              Median :0.00000
##  Mean   :0.01794         Mean   :0.2166            Mean   :0.05157  
##  3rd Qu.:0.00000         3rd Qu.:0.0000            3rd Qu.:0.00000  
##  Max.   :1.00000         Max.   :1.0000            Max.   :1.00000  
##                                                                     
##  is_missing_num_total_rooms is_missing_sale_price is_missing_sq_footage
##  Min.   :0.0000000          Min.   :0.0000        Min.   :0.0000       
##  1st Qu.:0.0000000          1st Qu.:1.0000        1st Qu.:0.0000       
##  Median :0.0000000          Median :1.0000        Median :1.0000       
##  Mean   :0.0008969          Mean   :0.7632        Mean   :0.5426       
##  3rd Qu.:0.0000000          3rd Qu.:1.0000        3rd Qu.:1.0000       
##  Max.   :1.0000000          Max.   :1.0000        Max.   :1.0000       
##                                                                        
##  is_missing_total_charges approx_year_built cats_allowed coop_condo 
##  Min.   :0.00000          Min.   :1893      No :1402     co-op:1661  
##  1st Qu.:0.00000          1st Qu.:1950      Yes: 828     condo: 569  
##  Median :0.00000          Median :1958                              
##  Mean   :0.03767          Mean   :1963                              
##  3rd Qu.:0.00000          3rd Qu.:1970                              
##  Max.   :1.00000          Max.   :2017                              
##                           NA's   :40                                
##     dining_room_type dogs_allowed    fuel_type    garage_exists    kitchen
_type
##  combo      :957     No :1684     electric:  62   no :1826      Combo    :
399  
##  dining area:  2     Yes: 546     gas     :1348   yes: 404      Eat_In   :
942  
##  formal     :620                  oil     : 664                 Efficient:
849  
##  none       :  2                  other   :  44                 NA's     :
40  
##  other      :201                  NA's    : 112                           
##  NA's       :448                                                          
##                                                                           
##  maintenance_cost  num_bedrooms   num_full_bathrooms num_half_bathrooms
##  Min.   :  0.0    Min.   :0.000   Min.   :1.000      Min.   :0.00000   
##  1st Qu.:  0.0    1st Qu.:1.000   1st Qu.:1.000      1st Qu.:0.00000   
##  Median :606.0    Median :2.000   Median :1.000      Median :0.00000   
##  Mean   :488.3    Mean   :1.653   Mean   :1.231      Mean   :0.07354   
##  3rd Qu.:763.0    3rd Qu.:2.000   3rd Qu.:1.000      3rd Qu.:0.00000   
##  Max.   :998.0    Max.   :6.000   Max.   :3.000      Max.   :2.00000   
##  NA's   :483      NA's   :115                                          
##  num_total_rooms    sale_price       sq_footage                walk_score
##  Min.   : 0.000   Min.   : 55000   Min.   : 100.0   Car-Dependent    :  77
##  1st Qu.: 3.000   1st Qu.:171500   1st Qu.: 743.0   Somewhat Walkable: 243
##  Median : 4.000   Median :259500   Median : 881.0   Very Walkable    : 821
##  Mean   : 4.139   Mean   :314957   Mean   : 955.4   Walker's Paradise:1089
##  3rd Qu.: 5.000   3rd Qu.:428875   3rd Qu.:1100.0                         
##  Max.   :14.000   Max.   :999999   Max.   :6215.0                         
##  NA's   :2        NA's   :1702     NA's   :1210                           
##                 Zipcode    total_charges   
## North Queens        :556   Min.   :   0.0  
## West Central Queens:457   1st Qu.:   0.0  
## West Queens         :340   Median :   0.0  
## Southwest Queens    :205   Mean   : 133.5  
## Northeast Queens    :179   3rd Qu.:   0.0  
## Southeast Queens    :151   Max.   :1591.7  
## (Other)             :342   NA's   :84
```

```
#write.csv(final_cleaned_housing_data_table, "C:\\Users\\Enoch Kim\\Desktop\\Spring 2021\\Math 342w\\myGit\\Math342W_Spring21_QC\\final_project\\final_cleaned_housing_data_table.csv")

#Since some sale_price are NA, I will be dropping the rows that have NA.
final_cleaned_housing_data_table_missing_responses = final_cleaned_housing_data_table %>%
  filter(is.na(sale_price))

final_cleaned_housing_data_table_not_missing_responses = final_cleaned_housing_data_table %>%
  filter(!is.na(sale_price))

#We now must setup train and testing, please note: In the final_cleaned_housing_data_table_not_missing_responses there are 528 observation

n = nrow(final_cleaned_housing_data_table_not_missing_responses)
k = 5

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

n_test = as.integer((1 / k) * n)
n_train = as.integer(n - n_test)

training_data = final_cleaned_housing_data_table_not_missing_responses[train_indices, ]
testing_data = final_cleaned_housing_data_table_not_missing_responses[test_indices, ]

X_test = testing_data %>%
  mutate(sale_price = NA)

y_test = testing_data$sale_price

skim(final_cleaned_housing_data_table_not_missing_responses)
```

Data summary

Name	final_cleaned_housing_dat...
Number of rows	528
Number of columns	28

Column type frequency:

factor	9
numeric	19

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1.00	FALSE	2	No: 285, Yes: 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	No: 381, Yes: 147
fuel_type	24	0.95	FALSE	4	gas: 301, oil: 180, oth: 12, ele: 11
garage_exists	0	1.00	FALSE	2	no: 434, yes: 94
kitchen_type	7	0.99	FALSE	3	Eff: 231, Eat: 209, Com: 81
walk_score	0	1.00	FALSE	4	Ver: 237, Wal: 219, Som: 61, Car: 11
Zipcode	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
is_missing_approx_year_built	0	1.00	0.01	0.11	0	0	0.0	0.00	1.00	▇▁▁▁▁
is_missing_dining_room_type	0	1.00	0.23	0.42	0	0	0.0	0.00	1.00	▇▁▁▁▂

is_missing_fuel_type	0	1.00	0.05	0.21	0	0	0.0	0.00	1.00	▇▁▁▁▁
is_missing_kitchen_type	0	1.00	0.01	0.11	0	0	0.0	0.00	1.00	▇▁▁▁▁
is_missing_maintenance_cost	0	1.00	0.16	0.37	0	0	0.0	0.00	1.00	▇▁▁▁▂
is_missing_num_bedrooms	0	1.00	0.00	0.00	0	0	0.0	0.00	0.00	▁▁▇▁▁
is_missing_num_total_rooms	0	1.00	0.00	0.00	0	0	0.0	0.00	0.00	▁▁▇▁▁
is_missing_sale_price	0	1.00	0.00	0.00	0	0	0.0	0.00	0.00	▁▁▇▁▁
is_missing_sq_footage	0	1.00	0.60	0.49	0	0	1.0	1.00	1.00	▅▁▁▁▇
is_missing_total_charges	0	1.00	0.02	0.14	0	0	0.0	0.00	1.00	▇▁▁▁▁

approx_year_built	6	0.99	1962.38	20.56	1915	1950	1957.0	1968.00	2016.00	▁▇▇▂▂
maintenance_cost	86	0.84	504.81	335.66	0	0	641.5	756.75	996.00	▇▁▂▇▂
num_bedrooms	0	1.00	1.54	0.75	0	1	1.0	2.00	3.00	▁▇▁▇▂
num_full_bathrooms	0	1.00	1.20	0.42	1	1	1.0	1.00	3.00	▇▁▂▁▁
num_half_bathrooms	0	1.00	0.06	0.24	0	0	0.0	0.00	2.00	▇▁▁▁▁
num_total_rooms	0	1.00	4.02	1.20	1	3	4.0	5.00	8.00	▁▅▇▂▁
sale_price	0	1.00	314956.56	179526.60	55000	171500	259500.0	428875.00	999999.00	▇▆▂▁▁
sq_footage	315	0.40	965.28	490.42	375	750	874.0	1010.00	6215.00	▇▁▁▁▁

total_charges	10	0.98	135.26	284.11	0	0	0.0	0.00	1500.92	▇▁▁▁▁

```
#I will deal by the missing data by now imputing the data table.
missing_data2 = rbind(training_data, X_test, final_cleaned_housing_data_table
_missing_responses)

complete_housing_data = missForest(missing_data2)$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!
##   missForest iteration 5 in progress...done!
##   missForest iteration 6 in progress...done!

##(left out missForest iternation due to length of the file)

sum(is.na(complete_housing_data))

## [1] 0

skim(complete_housing_data)
```

Data summary

Name	complete_housing_data
Number of rows	2230
Number of columns	28
________________________	
Column type frequency:	
factor	9
numeric	19
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1	FALSE	2	No: 1402, Yes: 828
coop_condo	0	1	FALSE	2	co-: 1661, con: 569
dining_room_type	0	1	FALSE	5	com: 1241, for: 745, oth: 239, din: 3

dogs_allowed	0	1	FALSE	2	No: 1684, Yes: 546
fuel_type	0	1	FALSE	4	gas: 1410, oil: 708, ele: 66, oth: 46
garage_exists	0	1	FALSE	2	no: 1826, yes: 404
kitchen_type	0	1	FALSE	3	Eat: 956, Eff: 866, Com: 408
walk_score	0	1	FALSE	4	Wal: 1089, Ver: 821, Som: 243, Car: 77
Zipcode	0	1	FALSE	9	Nor: 556, Wes: 457, Wes: 340, Sou: 205

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
is_missing_approx_year_built	0	1	0.02	0.13	0	0.00	0	0.00	1.00	▇▁▁▁▁
is_missing_dining_room_type	0	1	0.20	0.40	0	0.00	0	0.00	1.00	▇▁▁▁▂
is_missing_fuel_type	0	1	0.05	0.22	0	0.00	0	0.00	1.00	▇▁▁▁▁
is_missing_kitchen_type	0	1	0.02	0.13	0	0.00	0	0.00	1.00	▇▁▁▁▁
is_missing_maintenance_cost	0	1	0.22	0.41	0	0.00	0	0.00	1.00	▇▁▁▁▂

is_missing_num_bedrooms	0	1	0.05	0.22	0	0.00	0	0.00	1.00	▇▁▁▁▁
is_missing_num_total_rooms	0	1	0.00	0.03	0	0.00	0	0.00	1.00	▇▁▁▁▁
is_missing_sale_price	0	1	0.76	0.43	0	1.00	1	1.00	1.00	▂▁▁▁▇
is_missing_sq_footage	0	1	0.54	0.50	0	0.00	1	1.00	1.00	▇▁▁▁▇
is_missing_total_charges	0	1	0.04	0.19	0	0.00	0	0.00	1.00	▇▁▁▁▁
approx_year_built	0	1	1962.72	20.98	1893	1950.00	1958	1970.00	2017.00	▁▂▇▂▂
maintenance_cost	0	1	554.80	333.11	0	389.75	677	811.86	998.00	▇▁▂▇▇
num_bedrooms	0	1	1.62	0.75	0	1.00	2	2.00	6.00	▇▇▂

										▁▁
num_full_bathrooms	0	1	1.23	0.44	1	1.00	1	1.00	3.00	▇▁▂▁▁
num_half_bathrooms	0	1	0.07	0.27	0	0.00	0	0.00	2.00	▇▁▁▁▁
num_total_rooms	0	1	4.14	1.35	0	3.00	4	5.00	14.00	▁▇▂▁▁
sale_price	0	1	329944.85	149995.40	55000	201616.76	291133	444841.96	950000.00	▇▇▇▁▁
sq_footage	0	1	916.31	315.17	100	720.71	860	1019.58	6215.00	▇▁▁▁▁
total_charges	0	1	146.74	285.47	0	0.00	0	174.50	1591.67	▇▁▁▁▁

```
complete_housing_data_final = complete_housing_data %>%
  filter(is_missing_sale_price == 0) %>%
  select(-is_missing_sale_price)

complete_housing_data_final = cbind(complete_housing_data_final[, -(1 : 9)],
tbl_df(t(unique(t(complete_housing_data_final[, (1 : 9)])))))

complete_housing_data_final_training = complete_housing_data_final[1 : n_trai
n, ]
```

```
complete_housing_data_final_test = complete_housing_data_final[(n_train + 1):
n, ]

complete_housing_data_final_test$sale_price = y_test

#Before creating models and after imputing, I need to merge charges with main
tenance cost.

complete_housing_data_final_test_2 = complete_housing_data_final_test %>%
  mutate(final_charges = maintenance_cost + total_charges) %>%
  select(-maintenance_cost, -total_charges)

complete_housing_data_final_training_2 = complete_housing_data_final_training
%>%
  mutate(final_charges = maintenance_cost + total_charges) %>%
  select(-maintenance_cost, -total_charges)

complete_housing_data_final_ytest = complete_housing_data_final_test_2$sale_p
rice
complete_housing_data_final_Xtest = complete_housing_data_final_test_2
complete_housing_data_final_Xtest$sale_price = NULL

complete_housing_data_final_training_2_ytrain = complete_housing_data_final_t
raining_2$sale_price
complete_housing_data_final_training_2_Xtrain = complete_housing_data_final_t
raining_2
complete_housing_data_final_training_2_Xtrain$sale_price = NULL

#Regression Tree Model
pacman::p_load(YARF)

## YARF can now make use of 7 cores.

options(java.parameters = "-Xmx4000m")

reg_tree = YARFCART(complete_housing_data_final_training_2_Xtrain, complete_h
ousing_data_final_training_2_ytrain)

## YARF initializing with a fixed 1 trees...
## YARF factors created...
## YARF after data preprocessed... 47 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.

reg_tree

## YARF v1.1 for regression
## Missing data feature ON.
## 1 trees, training data n = 423 and p = 47
## Model construction completed within 0.01 minutes.
## No OOB results to show (no trees have been fit as of yet).
```

```
get_tree_num_nodes_leaves_max_depths(reg_tree)

## $num_nodes
## [1] 303
##
## $num_leaves
## [1] 152
##
## $max_depth
## [1] 23

tree_image = illustrate_trees(reg_tree, max_depth = 5, open_file = TRUE, leng
th_in_px_per_half_split = 40)

#In-Sample for numbers for Regression Tree Model
y_hat_train = predict(reg_tree, complete_housing_data_final_training_2_Xtrain
)
e = complete_housing_data_final_training_2_ytrain - y_hat_train
sd(e) #This is s_e

## [1] 38211.04

1 - sd(e) / sd(complete_housing_data_final_training_2_ytrain) #This is R squa
red

## [1] 0.7846028

#Out of Sample numbers for Regression Tree Model
y_hat_test_tree = predict(reg_tree, complete_housing_data_final_Xtest)
e = complete_housing_data_final_ytest - y_hat_test_tree
sd(e) #This is s_e

## [1] 104290.4

1 - sd(e) / sd(complete_housing_data_final_ytest) #This is R squared

## [1] 0.4450623

#Linear Model
pacman::p_load(xtable)

lin_mod = lm(complete_housing_data_final_training_2_ytrain ~ ., complete_hous
ing_data_final_training_2_Xtrain)
lin_mod

##
## Call:
## lm(formula = complete_housing_data_final_training_2_ytrain ~
##     ., data = complete_housing_data_final_training_2_Xtrain)
##
## Coefficients:
##                   (Intercept)             approx_year_built
##                   -657514.53                        262.91
##              cats_allowedYes               coop_condocondo
##                      4973.68                     182846.88
##  dining_room_typedining area        dining_room_typeformal
##                    -33004.32                      27055.74
##        dining_room_typeother               dogs_allowedYes
##                       669.95                       8264.90
##                 fuel_typegas                  fuel_typeoil
##                      7929.36                      29054.33
##               fuel_typeother              garage_existsyes
##                     19448.84                      17769.02
##           kitchen_typeEat_In         kitchen_typeEfficient
##                    -27552.39                     -34479.93
##                 num_bedrooms            num_full_bathrooms
##                     34207.74                      94694.31
##           num_half_bathrooms               num_total_rooms
##                     54876.88                       6332.74
##                   sq_footage   walk_scoreSomewhat Walkable
##                        36.28                      56054.44
##      walk_scoreVery Walkable   walk_scoreWalker's Paradise
##                     32369.34                      86325.16
##               ZipcodeJamaica           ZipcodeNorth Queens
##                    -52165.31                      38668.72
##      ZipcodeNortheast Queens       ZipcodeNorthwest Queens
##                     32241.55                     192100.38
##      ZipcodeSoutheast Queens       ZipcodeSouthwest Queens
##                     26972.95                     -53370.59
##   ZipcodeWest Central Queens            ZipcodeWest Queens
##                     57418.50                      35396.70
## is_missing_approx_year_built   is_missing_dining_room_type
##                     40401.40                       2194.98
##         is_missing_fuel_type       is_missing_kitchen_type
##                    -11025.26                     -60909.11
##  is_missing_maintenance_cost       is_missing_num_bedrooms
##                     53839.01                            NA
##        is_missing_sq_footage      is_missing_total_charges
##                     -3691.93                      63514.06
##                final_charges
##                       124.10
```

```
#In-Sample for numbers for Linear Model
summary(lin_mod)$sigma
```

```
## [1] 81069.79
```

```
summary(lin_mod)$r.squared
```

```
## [1] 0.8094677
```

```
xtable(lin_mod)
```

```
## % latex table generated in R 4.0.1 by xtable 1.8-4 package
## % Tue May 25 17:33:57 2021
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrr}
##   \hline
##  & Estimate & Std. Error & t value & Pr($>$$|$t$|$) \\
##   \hline
## (Intercept) & -657514.5278 & 692318.2293 & -0.95 & 0.3428 \\
##   approx\_year\_built & 262.9084 & 352.7171 & 0.75 & 0.4565 \\
##   cats\_allowedYes & 4973.6813 & 12092.1046 & 0.41 & 0.6811 \\
##   coop\_condocondo & 182846.8762 & 16921.2283 & 10.81 & 0.0000 \\
##   dining\_room\_typedining area & -33004.3193 & 59205.5423 & -0.56 & 0.577
5 \\
##   dining\_room\_typeformal & 27055.7385 & 10747.8809 & 2.52 & 0.0122 \\
##   dining\_room\_typeother & 669.9466 & 15431.3865 & 0.04 & 0.9654 \\
##   dogs\_allowedYes & 8264.9038 & 13374.8095 & 0.62 & 0.5370 \\
##   fuel\_typegas & 7929.3615 & 29647.9250 & 0.27 & 0.7893 \\
##   fuel\_typeoil & 29054.3289 & 30352.2065 & 0.96 & 0.3390 \\
##   fuel\_typeother & 19448.8370 & 40126.0657 & 0.48 & 0.6282 \\
##   garage\_existsyes & 17769.0198 & 11401.3343 & 1.56 & 0.1199 \\
##   kitchen\_typeEat\_In & -27552.3874 & 12765.3496 & -2.16 & 0.0315 \\
##   kitchen\_typeEfficient & -34479.9340 & 12427.4623 & -2.77 & 0.0058 \\
##   num\_bedrooms & 34207.7358 & 9927.3915 & 3.45 & 0.0006 \\
##   num\_full\_bathrooms & 94694.3109 & 14143.1493 & 6.70 & 0.0000 \\
##   num\_half\_bathrooms & 54876.8786 & 18609.7951 & 2.95 & 0.0034 \\
##   num\_total\_rooms & 6332.7396 & 6906.5600 & 0.92 & 0.3598 \\
##   sq\_footage & 36.2757 & 14.4408 & 2.51 & 0.0124 \\
##   walk\_scoreSomewhat Walkable & 56054.4387 & 34547.2491 & 1.62 & 0.1055 \
\
##   walk\_scoreVery Walkable & 32369.3364 & 33407.2812 & 0.97 & 0.3332 \\
##   walk\_scoreWalker's Paradise & 86325.1612 & 34118.3281 & 2.53 & 0.0118 \
\
##   ZipcodeJamaica & -52165.3064 & 23289.6333 & -2.24 & 0.0257 \\
##   ZipcodeNorth Queens & 38668.7179 & 19328.8764 & 2.00 & 0.0461 \\
##   ZipcodeNortheast Queens & 32241.5537 & 20439.4752 & 1.58 & 0.1155 \\
##   ZipcodeNorthwest Queens & 192100.3774 & 28484.4487 & 6.74 & 0.0000 \\
##   ZipcodeSoutheast Queens & 26972.9500 & 25007.9689 & 1.08 & 0.2815 \\
##   ZipcodeSouthwest Queens & -53370.5922 & 20231.3608 & -2.64 & 0.0087 \\
##   ZipcodeWest Central Queens & 57418.4988 & 20443.8373 & 2.81 & 0.0052 \\
##   ZipcodeWest Queens & 35396.6990 & 20921.9540 & 1.69 & 0.0915 \\
##   is\_missing\_approx\_year\_built & 40401.3985 & 36953.5726 & 1.09 & 0.27
49 \\
##   is\_missing\_dining\_room\_type & 2194.9765 & 10197.2871 & 0.22 & 0.8297
\\
##   is\_missing\_fuel\_type & -11025.2622 & 20464.6737 & -0.54 & 0.5904 \\
##   is\_missing\_kitchen\_type & -60909.1095 & 40665.2509 & -1.50 & 0.1350 \
\
##   is\_missing\_maintenance\_cost & 53839.0148 & 13611.9007 & 3.96 & 0.0001
\\
##   is\_missing\_sq\_footage & -3691.9261 & 9081.8391 & -0.41 & 0.6846 \\
##   is\_missing\_total\_charges & 63514.0559 & 30714.1795 & 2.07 & 0.0393 \\
##   final\_charges & 124.0997 & 26.9897 & 4.60 & 0.0000 \\
##    \hline
## \end{tabular}
## \end{table}
```

```
summary(lin_mod)
```

```
##
## Call:
## lm(formula = complete_housing_data_final_training_2_ytrain ~
##     ., data = complete_housing_data_final_training_2_Xtrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -244252  -42929    1413   37510  364351
##
## Coefficients: (1 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -657514.53  692318.23  -0.950 0.342846
## approx_year_built                262.91     352.72   0.745 0.456497
## cats_allowedYes                 4973.68   12092.10   0.411 0.681069
## coop_condocondo               182846.88   16921.23  10.806  < 2e-16 ***
## dining_room_typedining area   -33004.32   59205.54  -0.557 0.577542
## dining_room_typeformal         27055.74   10747.88   2.517 0.012231 *
## dining_room_typeother            669.95   15431.39   0.043 0.965394
## dogs_allowedYes                 8264.90   13374.81   0.618 0.536977
## fuel_typegas                    7929.36   29647.93   0.267 0.789265
## fuel_typeoil                   29054.33   30352.21   0.957 0.339047
## fuel_typeother                 19448.84   40126.07   0.485 0.628170
## garage_existsyes               17769.02   11401.33   1.559 0.119936
## kitchen_typeEat_In            -27552.39   12765.35  -2.158 0.031516 *
## kitchen_typeEfficient         -34479.93   12427.46  -2.774 0.005798 **
## num_bedrooms                   34207.74    9927.39   3.446 0.000632 ***
## num_full_bathrooms             94694.31   14143.15   6.695 7.63e-11 ***
## num_half_bathrooms             54876.88   18609.80   2.949 0.003385 **
## num_total_rooms                 6332.74    6906.56   0.917 0.359760
## sq_footage                        36.28      14.44   2.512 0.012412 *
## walk_scoreSomewhat Walkable    56054.44   34547.25   1.623 0.105505
## walk_scoreVery Walkable        32369.34   33407.28   0.969 0.333188
## walk_scoreWalker's Paradise    86325.16   34118.33   2.530 0.011799 *
## ZipcodeJamaica                -52165.31   23289.63  -2.240 0.025671 *
## ZipcodeNorth Queens            38668.72   19328.88   2.001 0.046141 *
## ZipcodeNortheast Queens        32241.55   20439.48   1.577 0.115521
## ZipcodeNorthwest Queens       192100.38   28484.45   6.744 5.66e-11 ***
## ZipcodeSoutheast Queens        26972.95   25007.97   1.079 0.281453
## ZipcodeSouthwest Queens       -53370.59   20231.36  -2.638 0.008677 **
## ZipcodeWest Central Queens     57418.50   20443.84   2.809 0.005229 **
## ZipcodeWest Queens             35396.70   20921.95   1.692 0.091485 .
## is_missing_approx_year_built   40401.40   36953.57   1.093 0.274945
## is_missing_dining_room_type     2194.98   10197.29   0.215 0.829686
## is_missing_fuel_type          -11025.26   20464.67  -0.539 0.590373
## is_missing_kitchen_type       -60909.11   40665.25  -1.498 0.135000
## is_missing_maintenance_cost    53839.01   13611.90   3.955 9.10e-05 ***
## is_missing_num_bedrooms              NA         NA      NA       NA
## is_missing_sq_footage          -3691.93    9081.84  -0.407 0.684588
## is_missing_total_charges       63514.06   30714.18   2.068 0.039316 *
## final_charges                    124.10      26.99   4.598 5.79e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 81070 on 385 degrees of freedom
## Multiple R-squared:  0.8095, Adjusted R-squared:  0.7912
## F-statistic: 44.21 on 37 and 385 DF,  p-value: < 2.2e-16
```

```
#Out of Sample numbers for Linear Model
y_hat_test_linear = predict(lin_mod, complete_housing_data_final_Xtest)
```

```
## Warning in predict.lm(lin_mod, complete_housing_data_final_Xtest): predict
ion
## from a rank-deficient fit may be misleading
```

```
e = complete_housing_data_final_ytest - y_hat_test_linear
sd(e) #This is s_e
```

```
## [1] 91253.66
```

```
1 - sd(e) / sd(complete_housing_data_final_ytest) #This is R squared
```

```
## [1] 0.5144319
```

```
pacman::p_load(mlr)
complete_housing_data_X = complete_housing_data_final_training_2_Xtrain
y_salesprice_data = complete_housing_data_final_training_2_ytrain
mlr_data = cbind(y_salesprice_data, complete_housing_data_X)
colnames(mlr_data)[1] = "sales_price"
task = makeRegrTask(data = mlr_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(
  makeIntegerParam("mtry", lower = 2, upper = ncol(complete_housing_data_fina
l_training_2_Xtrain)),
  makeIntegerParam("ntree", lower = 2, upper = 90),
  makeIntegerParam("nodesize", lower = 2, upper = 90)
)


desc <- makeResampleDesc("CV", iters = 20)
```

```
ctrl <- makeTuneControlRandom(maxit = 20)
mlr_ret <- tuneParams("regr.randomForest", task = task, resampling = desc, par.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   -   - 2 to 24   -    TRUE     -
## ntree     integer   -   - 2 to 90   -    TRUE     -
## nodesize integer   -   - 2 to 90   -    TRUE     -
```

```
## With control class: TuneControlRandom
```

```
## Imputation value: Inf
```

```
## [Tune-x] 1: mtry=9; ntree=56; nodesize=81
```

```
## [Tune-y] 1: rmse.test.rmse=93968.7962312; time: 0.0 min
```

```
## [Tune-x] 2: mtry=6; ntree=44; nodesize=48
```

```
## [Tune-y] 2: rmse.test.rmse=88590.2512933; time: 0.0 min
```

```
## [Tune-x] 3: mtry=21; ntree=16; nodesize=45
```

```
## [Tune-y] 3: rmse.test.rmse=83149.0843894; time: 0.0 min
```

```
## [Tune-x] 4: mtry=7; ntree=46; nodesize=67
```

```
## [Tune-y] 4: rmse.test.rmse=92940.6621926; time: 0.0 min
```

```
## [Tune-x] 5: mtry=12; ntree=23; nodesize=36
```

```
## [Tune-y] 5: rmse.test.rmse=82895.5614079; time: 0.0 min
```

```
## [Tune-x] 6: mtry=22; ntree=20; nodesize=70
```

```
## [Tune-y] 6: rmse.test.rmse=88042.3371926; time: 0.0 min
```

```
## [Tune-x] 7: mtry=5; ntree=61; nodesize=74
```

```
## [Tune-y] 7: rmse.test.rmse=95605.6154323; time: 0.0 min
```

```
## [Tune-x] 8: mtry=11; ntree=67; nodesize=40
```

```
## [Tune-y] 8: rmse.test.rmse=83063.6255813; time: 0.0 min
```

```
## [Tune-x] 9: mtry=19; ntree=44; nodesize=37
```

```
## [Tune-y] 9: rmse.test.rmse=79965.9162643; time: 0.0 min
```

```
## [Tune-x] 10: mtry=11; ntree=63; nodesize=70
```

```
## [Tune-y] 10: rmse.test.rmse=90584.3898156; time: 0.0 min
## [Tune-x] 11: mtry=7; ntree=72; nodesize=28

## [Tune-y] 11: rmse.test.rmse=82294.3713784; time: 0.0 min

## [Tune-x] 12: mtry=11; ntree=72; nodesize=63

## [Tune-y] 12: rmse.test.rmse=89181.1946054; time: 0.0 min

## [Tune-x] 13: mtry=9; ntree=75; nodesize=89

## [Tune-y] 13: rmse.test.rmse=95771.1153543; time: 0.0 min

## [Tune-x] 14: mtry=17; ntree=31; nodesize=70

## [Tune-y] 14: rmse.test.rmse=89823.5401941; time: 0.0 min

## [Tune-x] 15: mtry=14; ntree=11; nodesize=89

## [Tune-y] 15: rmse.test.rmse=97161.2506390; time: 0.0 min

## [Tune-x] 16: mtry=22; ntree=85; nodesize=12

## [Tune-y] 16: rmse.test.rmse=73718.3282497; time: 0.0 min

## [Tune-x] 17: mtry=21; ntree=72; nodesize=42

## [Tune-y] 17: rmse.test.rmse=82403.1286488; time: 0.0 min

## [Tune-x] 18: mtry=19; ntree=28; nodesize=11

## [Tune-y] 18: rmse.test.rmse=74627.3836438; time: 0.0 min

## [Tune-x] 19: mtry=11; ntree=75; nodesize=88

## [Tune-y] 19: rmse.test.rmse=94345.0541190; time: 0.0 min

## [Tune-x] 20: mtry=9; ntree=14; nodesize=71

## [Tune-y] 20: rmse.test.rmse=93410.7827133; time: 0.0 min

## [Tune] Result: mtry=22; ntree=85; nodesize=12 : rmse.test.rmse=73718.32824
97
```

```
#The most optimal result
mlr_ret
```

```
## Tune result:
## Op. pars: mtry=22; ntree=85; nodesize=12
## rmse.test.rmse=73718.3282497
```

```
#Getting the In=Sample for Final model
rf_mod = YARF(complete_housing_data_X, y_salesprice_data, mtry= as.integer(ml
r_ret$x[1]), num_trees = as.integer(mlr_ret$x[2]))
```

```
## YARF initializing with a fixed 85 trees...
## YARF factors created...
## YARF after data preprocessed... 47 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.
```

```
rf_mod
```

```
## YARF v1.1 for regression
## Missing data feature ON.
## 85 trees, training data n = 423 and p = 47
## Model construction completed within 0.01 minutes.
## OOB results on all observations:
##   R^2: 0.79109
##   RMSE: 80987.07
##   MAE: 54789.77
##   L2: 2.774417e+12
##   L1: 23176072
```

```
rf_is_mod = YARF(complete_housing_data_final_training_2_Xtrain, complete_hous
ing_data_final_training_2_ytrain, mtry= as.integer(mlr_ret$x[1]), num_trees =
as.integer(mlr_ret$x[2]))
```

```
## YARF initializing with a fixed 85 trees...
## YARF factors created...
## YARF after data preprocessed... 47 total features...
## Beginning YARF regression model construction...done.
## Calculating OOB error...done.
```

```
rf_is_mod
```

```
## YARF v1.1 for regression
## Missing data feature ON.
## 85 trees, training data n = 423 and p = 47
## Model construction completed within 0.01 minutes.
## OOB results on all observations:
##   R^2: 0.789
##   RMSE: 81391.65
##   MAE: 55605.18
##   L2: 2.802206e+12
##   L1: 23520992
```

```
yhat = predict(rf_is_mod, complete_housing_data_final_Xtest)
```

```
#Getting the Out of Sample for Final model
oos_rmse = sqrt(mean((complete_housing_data_final_ytest - yhat) ^ 2))
oos_rsq = 1 - sum((complete_housing_data_final_ytest - yhat) ^ 2) / sum((comp
lete_housing_data_final_ytest - mean(y_salesprice_data)) ^ 2)
oos_rmse
```

```
## [1] 73797.01
```

```
oos_rsq

## [1] 0.8459753
```