Predicting Property Values in Queens, NY

Final project for Math 342W at Queens College

5/26/2024

By Benjamin Minkin

Abstract

For this project, I created various supervised machine learning models to predict the sale price of housing in Queens, NY. To do this, I digested a data set of 2,230 observations with 55 features compiled in 2017. This data set is the raw version that can be found at MLSI. Originally, it was collected through the Amazon MTURK marketplace. First, I extracted the features that I hypothesized were relevant and modified them until they were ready for analysis. Then, I trained an OLS, regression tree, and random forest model on most of the data, leaving the remaining section for validation. Finally, I validated the models and interpreted the results. I then explored a method of improving my model by imputation of missing data.

1. Introduction

Being able to predict the market value of a property is useful for many reasons across industries. For government use, a model can determine the property taxes of a unit. For personal use, both the buyer and seller benefit from having a predicted valuation, which sets an expectation for the transaction. For financial purposes, this model can be used to plan investments in constructing new housing, locating properties to renovate and resell, and determining if a mortgage should be granted. For all the aforementioned uses, a model does not need to be perfectly accurate. Instead, this model needs to be a decent approximation to be practically useful. In comparison, the gold standard Zillow.com's estimate for NYC housing prices in April 2020 was within 5% of the actual sale price only 80.2% of the time.

2. The Data

In this data set, the sample space was limited to condos and co-ops that are listed or sold for under $1,000,000 in mainland Queens. Of the properties that were sold, the date of sale ranges from February 2016 to February 2017. I do not believe that this data set contains all properties that were sold during this period. However, the data that is imported will be treated as trustworthy and as a simple random sample of the overall market. I supplanted this data with six "approx._year_built" values that I collected on streeteasy.com by searching for the address. The goal of the model is to predict the sale_price response variable.

2.2 Featurization

After spending time understanding the data set, the following are the relevant features I selected: Note that the descriptive statistics detailed in the table are calculated on the fully cleaned subset that has no NA sale_price values.

Feature name	**Data Type**	**Description**	**Min/Average/Max**
approx_year_built	Dummy	1 if built after 1978, 0 if built before	%19.1 = 1
coop_condo	Dummy	1 if co-op, 0 if condo	%75.6 = 1
dogs_allowed	Dummy	1 if yes, 0 if no	%27.8 = 1
garage_exists	Dummy	1 if yes, 0 if no	%17.8 = 1
num_bedrooms	Numeric	Number of bedrooms	0 | 1.54 | 3
num_full_bathrooms	Numeric	Number of full bathrooms (1-3)	1 | 1.2 | 3
num_half_bathrooms	Numeric	Number of half bathrooms (0-2)	0 | 0.06 | 2
num_total_rooms	Numeric	Number of rooms in the property	1 | 4 | 8
walk_score	Numeric	1-100 score of walkability, higher is better	15 | 83 | 99
total_com_maint	Numeric	Fees associated with this property	0 | 709 | 4659
full_address_or_zip_code	Categorical	The property's area by zip code	1-9

To get to this point, I had to make decisions when cleaning the data. I set the cutoff for the approx_year_built dummy to 1978 because that was when lead paint was banned federally.

While lead paint was technically banned in NYC in 1960, consumers may be overly cautious and assume that the rules were not enforced until 1978. For dogs_allowed, I set “yes” and “yes89” to 1 and “no” to 0. I presumed “yes89” to be a typo for yes. For garage_exists, I set “yes”, “Yes”, “UG”. “Underground”, and “eys” to 1. I presumed “eys” to be a typo for yes. I also set NA values to 0. I made this decision as there were no properties marked as “no”. This led me to believe that NA was used as a stand-in for “no”. Looking at the subset of sold properties, I noticed that the unique values of num_half_bathrooms were NA, 1, and 2. I presumed that, as in garage_exists, NA was used as a stand-in for 0. To make the total_com_maint column, I cast the common_charges and maintenance_cost columns as numeric. Then I added them together to create the total_com_maint column. I made the decision to combine the two columns to avoid multicollinearity between the individual columns and the coop_condo dummy variable. I wanted this column to capture the relationship between fees and sale price, not the price difference between condos and co-ops.

The regions and the zip codes contained within them are listed below.

Name	**Zip codes included**	**Alias**
Northeast Queens	11361 11362, 11363 11364	1
North Queens	11354 11355 11356 11357 11358 11359 11360	2
Central Queens	11365 11366 11367	3
Jamaica	11412 11423 11432 11433 11434 11435 11436	4
West Central Queens	11374 11375 11379 11385	5
Southeast Queens	11004 11005 11411 11413 11422 11426 11427 11428 11429	6
Southwest Queens	11414 11415 11416 11417 11418 11419 11420 11421	7
West Queens	11368 11369 11370 11372 11373 11377 11378	8
Northwest Queens	11101 11102 11103 11104 11105 11106	9

2.3 Missingness

The table below lists the number of NA values per relevant feature after cleaning and supplanting. As a reminder, there are 2,230 total observations. All features not mentioned have no NA values.

Feature name	**Number of NA values**
approx_year_built	34
num_bedrooms	115
num_total_rooms	2
sale_price	1702

After further subseting the data to include only rows with a sale price, there were no more NA values in any column.

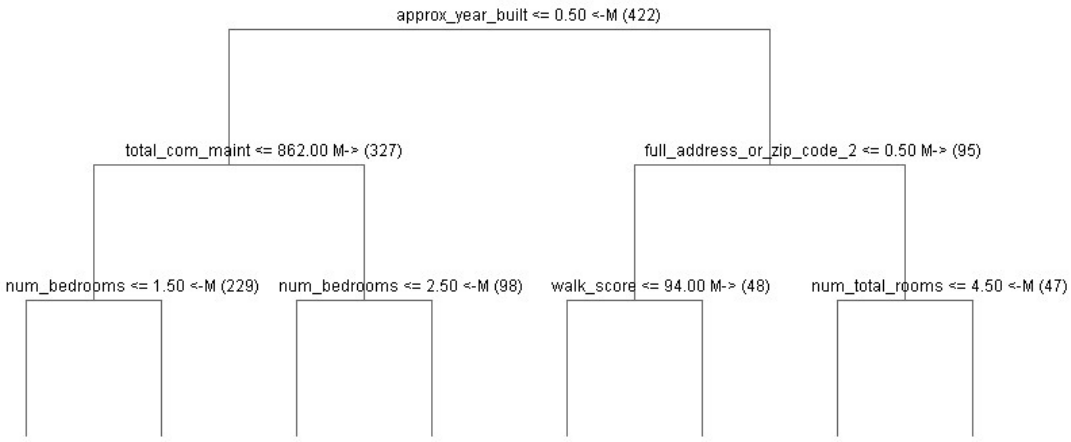
3. Modeling

Before modeling, I split the data into two groups. I made one group containing approximately 80% of the data and designated it for model training. The remaining 20% was set aside for out-of-sample model validation. All models were trained on the same data and validated on the same data. This should allow for comparisons between models. I experimented with three approaches to supervised machine learning models.

3.1 Regression Tree modeling

I used the YARF function from the YARF package to fit the training data to a single regression tree. This model is a flow chart of sorts that uses step functions based on a feature to go either left or right.

The top three layers are pictured below.



The first and most important split is based on whether the property was built after 1978. If it was built after, the next most important feature is whether the property is in North Queens. If the property was built before 1978, the next most important distinction is whether the combined maintenance and common fees are over 862.

3.2 Ordinary Least Squares Modeling

The results of the OLS regression model are tabulated below.

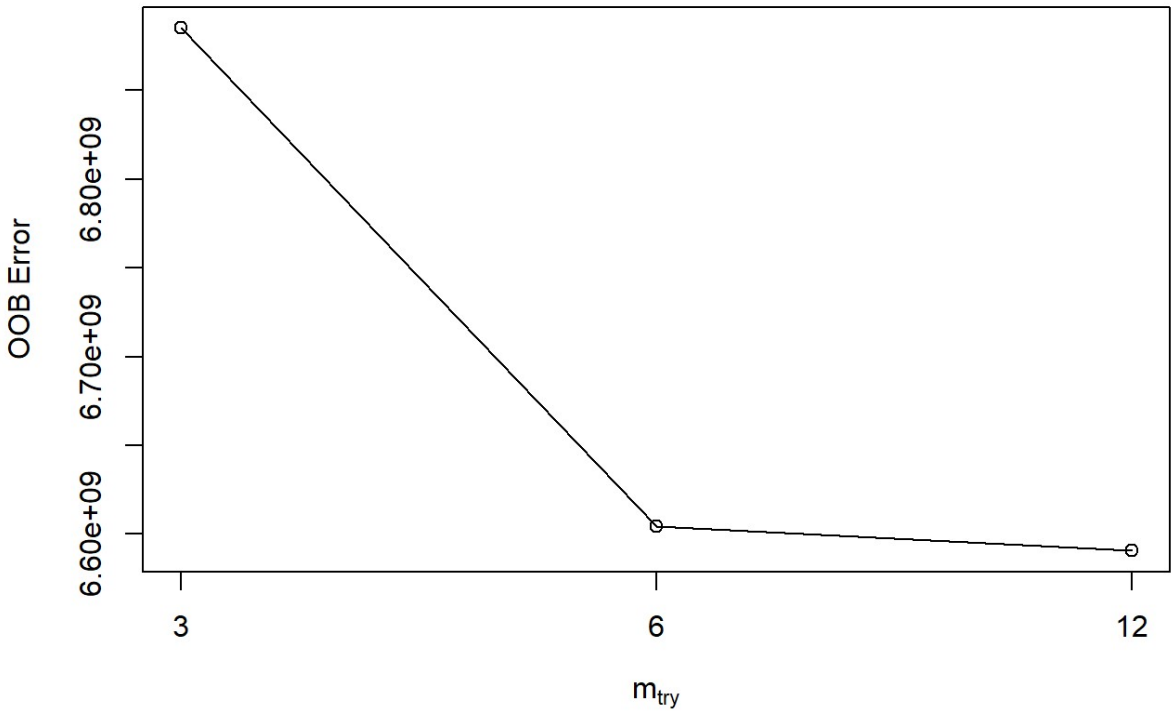
approx_year_built	14,588.150
	(20,848.920)
coop_condo	-216,870.800***
	(19,463.920)
dogs_allowed	27,867.080***
	(9,321.611)
garage_exists	11,917.370
	(10,930.610)
num_bedrooms	40,605.490***
	(9,224.459)
num_full_bathrooms	95,350.940***
	(13,037.910)
num_half_bathrooms	52,838.130***
	(17,640.520)
num_total_rooms	6,928.698
	(6,038.764)
walk_score	1,461.956***
	(401.418)
total_com_maint	152.775***
	(13.914)
full_address_or_zip_code_2	17,003.930
	(14,640.910)
full_address_or_zip_code_3	-51,196.880**
	(20,009.990)
full_address_or_zip_code_4	-103,405.800***
	(19,623.740)
full_address_or_zip_code_5	36,836.750**
	(16,548.090)
full_address_or_zip_code_6	-7,455.474
	(19,360.160)
full_address_or_zip_code_7	-84,522.640***
	(15,937.530)
full_address_or_zip_code_8	10,405.100
	(17,371.560)
full_address_or_zip_code_9	133,637.000***
	(24,769.710)
Constant	32,849.430
	(42,414.740)
Observations	422
R2	0.819
Adjusted R2	0.811
Residual Std. Error	79,266.630 (df = 403)
F Statistic	101.538*** (df = 18; 403)
Note:	*p<0.1; **p<0.05; ***p<0.01

What was surprising about these results is the positive sign of the total_com_maint variable. I would have thought that an increase in fees would decrease the value of the unit, but the opposite is true. This is likely because the increase in fees correlates with more expensive apartments that have more amenities. In this way, the total_com_maint is a proxy for amenities rather than the cost of ownership. Another surprising result was that although I have heard the saying “location,

location, location" in reference to real estate, the difference in price prediction between areas was massive. All things being the same, a property in Northwest Queens is predicted to be worth $237,000 more than a property in Jamaica, Queens. The approx_year_built variable was also surprising. I would have thought that the age of the building would have more influence and be decidedly positive. This result of a near-zero value is likely due to confounding variables. It is also especially strange considering that this was the most important decision for the single regression tree model.

3.3 Random Forest Modeling

I used the tuneRF function from the randomForest library to determine the best mtry variable to use for the random forest.



I used the YARF function again, this time setting mtry to the optimal value of 6. The model used its default of 500 trees to determine the best splits.

4. Performance Results

Model	In-sample R^2	In-sample RMSE	Oos-R^2	Oos-RMSE
Tree	0.84	117,927	0.49	140,993
OLS	0.82	77,462	0.72	90,242
Random forest	0.78	85,984	0.83	69,655

The random forest model performed best out-of-sample. This matched my a priori, expectation as I do not believe housing prices are linear in their variables. As expected, all models did worse out-of-sample. The OLS model did slightly worse, indicating minimal evidence of overfitting.

5. Imputation

One major feature I was unable to include was square footage. I made the decision to exclude this feature because of the high number of NA values in that column. After completing the models without square footage, I set out to include it using imputation. First, I created a dummy value to keep track of whether the square footage was NA. Then, I used the entire data set to determine the missing values of square footage. I made sure to exclude the sale_price column to avoid inducing or exacerbating a correlation. I added the finished sq_footage column to the training and testing data and proceeded as before.

I got the following performance results:

5.2 Imputing Square ft. Performance

Model	In-sample R^2	In-sample RMSE	Oos-R^2	Oos-RMSE
Tree	0.79	139,549	0.50	126,762
OLS	0.81	79,853	0.62	107,963
Random forest	0.78	86,313	0.78	70,550

It seems that imputing square feet had little effect on the tree and forest model while harming the OLS model. Surprisingly, no model improved out-of-sample. It is important to note that these models' performance varies, and the imputed models may perform better in the long run. Both markdown files will be available on my GitHub if one would like to experiment further.

6. Discussion

I was satisfied with the results of the random forest model, particularly without imputation. Due to its non-parametric nature, this model should have more flexibility in real-world applications. Generally, the downside to random forests tends to be a lack of interpretability. While this is true, I would argue that the accuracy of this model makes up for its shortcomings. The out-of-sample performance RMSE of 70,000 is low enough to be useful, and the R^2 of 0.83 implies that the model explains the data's variance well. While there are certainly external factors that influence housing prices, I would presume that this model will perform well in real-life use cases.

Acknowledgements

I would like to thank my brother Jacob for his input about the NYC housing market and help with navigating the YARF R library.

Code Appendix

{r}

```
library("randomForest")

library("YARF")

library("fastDummies")

library("stargazer")

library("dplyr")

library("stringr")


#import the data

housing_data = read.csv("C:\\Users\\benmi\\OneDrive\\Desktop\\Math 342w\\housing_data_2016_2017.csv")


#vector of columns that are relevant for the model

cols_to_keep = c(29, 31:33,36, 38:39, 41, 43, 45:48, 50, 53)


#number of columns used in the model

num_cols_kept = length(cols_to_keep)


#subset of housing data that only has relevant columns

relevant_data = housing_data %>% select(all_of(cols_to_keep))


#Dummify garage 1 for yes, Yes, eys, UG, and Underground, 0 for NA

relevant_data$garage_exists = ifelse(relevant_data$garage_exists == "Yes" | relevant_data$garage_exists == "yes" | relevant_data$garage_exists == "UG" | relevant_data$garage_exists == "1" | relevant_data$garage_exists == "eys" | relevant_data$garage_exists == "Underground", 1, 0)
```

```
#set NA values to 0
relevant_data$garage_exists = ifelse(is.na(relevant_data$garage_exists), 0, 1)

#set half bathrooms, NA to 0
relevant_data$num_half_bathrooms = ifelse(is.na(relevant_data$num_half_bathrooms), 0,
relevant_data$num_half_bathrooms)

#turn common charge's NA values to 0
relevant_data$common_charges = ifelse(is.na(relevant_data$common_charges), 0,
relevant_data$common_charges)

#remove dollar sign and comma
relevant_data$common_charges = str_replace(relevant_data$common_charges, "\\$", "")
relevant_data$common_charges = str_replace(relevant_data$common_charges, ",", "")

#convert to numeric
relevant_data$common_charges = as.numeric(relevant_data$common_charges)

#set NA values to 0
relevant_data$maintenance_cost = ifelse(is.na(relevant_data$maintenance_cost), 0,
relevant_data$maintenance_cost)

#remove dollar sign and comma
relevant_data$maintenance_cost = str_replace(relevant_data$maintenance_cost, "\\$", "")
relevant_data$maintenance_cost = str_replace(relevant_data$maintenance_cost, ",", "")

#set to numeric
relevant_data$maintenance_cost = as.numeric(relevant_data$maintenance_cost)
```

```
#create new col that is total maintenance and common_charges

relevant_data$total_com_maint = relevant_data$maintenance_cost +
relevant_data$common_charges


#add missing "year built" data

relevant_data[relevant_data$full_address_or_zip_code == "34-20 Parsons Blvd, Flushing NY,
11354", ]$approx_year_built = 1962


relevant_data[relevant_data$full_address_or_zip_code == "34-41 78th Street, Jackson Heights,
NY 11372", ]$approx_year_built = 1939


relevant_data[relevant_data$full_address_or_zip_code == "92-31 57th Ave,  Elmhurst NY,
11373", ]$approx_year_built = 1965


relevant_data[relevant_data$full_address_or_zip_code == "102-32 65th Ave,  Forest Hills NY,
11375", ]$approx_year_built = 1955


relevant_data[relevant_data$full_address_or_zip_code == "170-06 Crocheron Ave,  Flushing
NY, 11358", ]$approx_year_built = 1951


relevant_data[relevant_data$full_address_or_zip_code == "74-63 220th Street,  Bayside NY,
11364", ]$approx_year_built = 2017


# Dummify co-op_condo to be 1 for co-op and 0 for condo

relevant_data$coop_condo = ifelse(relevant_data$coop_condo == "co-op", 1, 0)


# Dummify dogs_allowed to be 1 for "yes" and 0 for "no"

relevant_data$dogs_allowed = ifelse(relevant_data$dogs_allowed == "yes" |
relevant_data$dogs_allowed == "yes89", 1, 0)
```

```
#remove “$” and “,”
relevant_data$sale_price = str_replace(relevant_data$sale_price, "\\$", "")
relevant_data$sale_price = str_replace(relevant_data$sale_price, ",", "")

#convert to numeric
relevant_data$sale_price = as.numeric(relevant_data$sale_price)

# Dummify approx_year_built to 0 if built before 1978, 1 if built after 1978. (When lead paint was outlawed federally)
relevant_data$approx_year_built = ifelse(relevant_data$approx_year_built<1978, 0, 1)

#further subset to rows with sale prices
non_NA_sale = relevant_data[!is.na(relevant_data$sale_price),]

#extract zip codes from address string
non_NA_sale$full_address_or_zip_code = str_sub(non_NA_sale$full_address_or_zip_code, start = -5)

#handle exception
non_NA_sale$full_address_or_zip_code[non_NA_sale$full_address_or_zip_code == "Share"] = "11354"

#convert to numeric
non_NA_sale$full_address_or_zip_code = as.numeric(non_NA_sale$full_address_or_zip_code)

#Categorize the zip codes into regions
Northeast = c(11361, 11362, 11363, 11364)
North = c(11354, 11355, 11356, 11357, 11358, 11359, 11360)
Central = c(11365, 11366, 11367)
```

```
Jamaica = c(11412, 11423, 11432, 11433, 11434, 11435, 11436)
Northwest = c(11101, 11102, 11103, 11104, 11105, 11106)
West_Central = c(11374, 11375, 11379, 11385)
Southeast = c(11004, 11005, 11411, 11413, 11422, 11426, 11427, 11428, 11429)
Southwest = c(11414, 11415, 11416, 11417, 11418, 11419, 11420, 11421)
West = c(11368, 11369, 11370, 11372, 11373, 11377, 11378)

non_NA_sale$full_address_or_zip_code = case_when(
  non_NA_sale$full_address_or_zip_code %in% Northeast ~ 1,
  non_NA_sale$full_address_or_zip_code %in% North ~ 2,
  non_NA_sale$full_address_or_zip_code %in% Central ~ 3,
  non_NA_sale$full_address_or_zip_code %in% Jamaica ~ 4,
  non_NA_sale$full_address_or_zip_code %in% West_Central ~ 5,
  non_NA_sale$full_address_or_zip_code %in% Southeast ~ 6,
  non_NA_sale$full_address_or_zip_code %in% Southwest ~ 7,
  non_NA_sale$full_address_or_zip_code %in% West ~ 8,
  non_NA_sale$full_address_or_zip_code %in% Northwest ~ 9)

# Dummify the zip code categorical variable
non_NA_sale = dummy_cols(non_NA_sale, select_columns = c("full_address_or_zip_code"),
remove_first_dummy = TRUE, remove_selected_columns = TRUE)

# Filter out columns that will not be used in the final model
features_vec = c(1,4:6,8:11,13,14,15:23)
select_data = non_NA_sale %>% select(all_of(features_vec))

#randomly pick indices
split_index = sample(nrow(select_data), size = nrow(select_data), replace = FALSE)
```

```
#create subset of 80%
splitting_point = split_index[1:round(0.8*nrow(select_data), 0)]

#create training and testing sets
train_data = select_data[splitting_point, ]
test_data = select_data[-splitting_point, ]

#The Linear Model
ols_model = lm(sale_price ~., data = train_data)
stargazer(ols_model, type = "text")

#The Tree Model
tree_model = YARF(X = train_data[,-9], y = as.vector(train_data $sale_price), num_trees = 1)
tree_model

#Tune the random forest to find the best m_try parameter
tuneRF(x = train_data[,-9],
      y = as.vector(train_data$sale_price),
      stepFactor = 0.5,
      ntreeTry=300,
      trace=TRUE,
      improve = 0.05,
      plot = TRUE)

#The Random Forest Model
yarf_model = YARF(X = train_data[,-9], y = as.vector(train_data $sale_price), mtry = 6)
yarf_model
```

```
#Test OLS model in-sample
cat("OLS in-sample r_sq is ", summary(ols_model)$r.squared, " \n")
cat("OLS in-sample RMSE is ", sqrt(mean(ols_model$residuals^2)))

#Test OLS model oos
ols_hat = predict(ols_model, test_data[,-9])
cat("\nOLS out-of-sample r_sq is ", cor(ols_hat, test_data$sale_price)^2, " \n")
cat("OLS out-of-sample RMSE is ", sqrt(mean((test_data$sale_price - ols_hat)^2)), " \n")

#Test tree model oos
tree_hat = predict(tree_model, test_data[,-9])
cat("tree out-of-sample r_sq is ", cor(tree_hat, test_data$sale_price)^2, " \n")
cat("tree out-of-sample RMSE is ", sqrt(mean((test_data$sale_price - tree_hat)^2)), " \n")

#Test yarf model oos
forest_hat = predict(yarf_model, test_data[,-9])
cat("Forest out-of-sample r_sq is ", cor(forest_hat, test_data$sale_price)^2, " \n")
cat("Forest out-of-sample RMSE is ", sqrt(mean((test_data$sale_price - forest_hat)^2)), " \n")
```

Citations

Adam Kapelner, Justin Bleich (2016). bartMachine: Machine Learning with Bayesian Additive Regression Trees. Journal of Statistical Software, 70(4), 1-40. doi:10.18637/jss.v070.i04

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables. R package version 5.2.3.

https://CRAN.R-project.org/package=stargazer

Kaplan J (2023). _fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables_. R package version 1.7.3,

https://CRAN.R project.org/package=fastDummies

Wickham H, François R, Henry L, Müller K, Vaughan D (2023). _dplyr: A Grammar of Data Manipulation_. R package version 1.1.4,

<https://CRAN.R-project.org/package=dplyr>.

Wickham H (2023). _stringr: Simple, Consistent Wrappers for Common String Operations_. R package version 1.5.1,

<https://CRAN.R-project.org/package=stringr>.