[Queens Apartment Price Modeling]

Final Project for Math 342 Data Science at Queens College
5-25-21

By Nasif Khan

Abstract

The goal of this paper is to create and compare predictive models on apartment sale prices in Queens, NY. We will be using public data on Queens apartment prices from February 2016 to February 2017. This data is a raw download from Amazon's MTurk. We will be comparing a Linear Model, Regression Tree Model, and a Random Forest Model. By examining the featurization and using data wrangling, we can compute which model provides the most accurate apartment sale price in Queens.

1. Introduction

Queens, NY is the most ethnically diverse region in the world and also observes many disparities in annual real estate apartment prices. It is challenging to model apartment prices due to many unknowns such as population and foot traffic, property values, state of the economy, new laws affecting real estate prices, etc. Zillow.com attempts to predict these real estate prices with their own disclosed model, however, they are unreliable and don't specify Queens as a borough in particular. The goal of this paper is to predict apartment prices more accurately than Zillow.com using data from February 2016-February 2017.

Models are mathematical representations of phenomenon that occur in nature and are difficult to observe or measure. Most models are inaccurate due to many unknown factors that occur in the natural world that we are unaware of. For example, the COVID-19 Pandemic impacted almost every facet of our lives dramatically and it is extremely unlikely someone was able to predict this event occurring over the last two years. However, using relevant data and features we can result with a strong estimate using these mathematical models. In our case, we will be using apartment prices data from 2016-2017 to predict current Queens apartment prices under one million dollars.

2. The Data

The data used for this model was downloaded via Amazon's MTurk and it is a large spreadsheet with 2,330 rows and 55 columns. This data is representative of Queens as it encompasses all neighborhoods within the region of Queens, and we can observe an expected variation in prices from lower to higher valued apartment prices. The data is uploaded through a raw download via R Studio by accessing the Github repository. There are some missing null values that are worth cleaning to prevent any underfitting or overfitting when creating our model. Figure 1 shows what the cleaned data looks like.

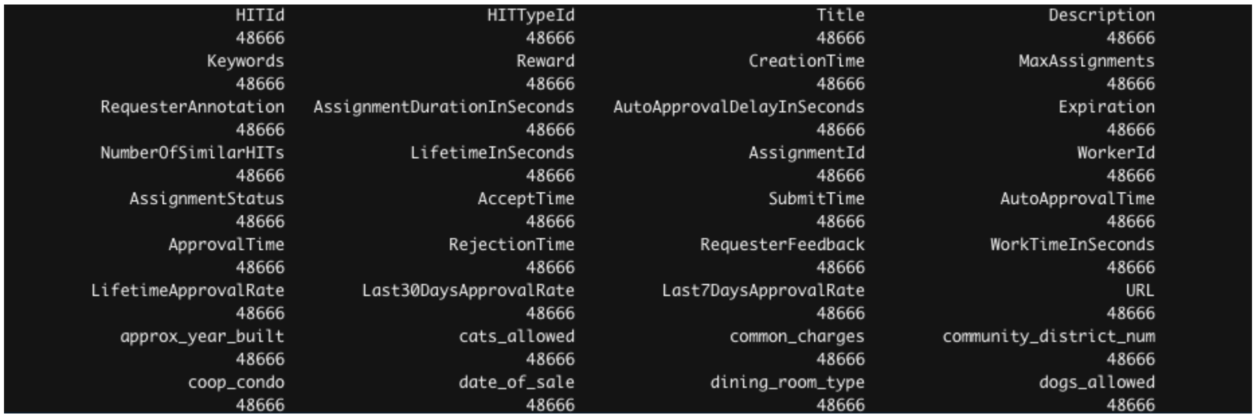

```
                HITId                     HITTypeId                        Title                  Description
                48666                         48666                        48666                        48666
             Keywords                        Reward                 CreationTime               MaxAssignments
                48666                         48666                        48666                        48666
  RequesterAnnotation  AssignmentDurationInSeconds  AutoApprovalDelayInSeconds                    Expiration
                48666                         48666                        48666                        48666
 NumberOfSimilarHITs              LifetimeInSeconds                 AssignmentId                     WorkerId
                48666                         48666                        48666                        48666
     AssignmentStatus                    AcceptTime                   SubmitTime             AutoApprovalTime
                48666                         48666                        48666                        48666
         ApprovalTime                 RejectionTime            RequesterFeedback            WorkTimeInSeconds
                48666                         48666                        48666                        48666
 LifetimeApprovalRate         Last30DaysApprovalRate        Last7DaysApprovalRate                          URL
                48666                         48666                        48666                        48666
     approx_year_built                 cats_allowed               common_charges      community_district_num
                48666                         48666                        48666                        48666
           coop_condo                  date_of_sale              dining_room_type                  dogs_allowed
                48666                         48666                        48666                        48666
```

Figure 1. Data Cleaning

2.2 Featurization

Featurization is the process of choosing and including features in your model. Features are sets of data that include characteristics that the model will use to predict something. I decided to choose 10 features and split them based on their characteristics. With five categorical features and five numerical features, I believe this will give a reasonable fit for the model when predicting the apartment price.

The five categorical values featured in the model include: **garage_exists, condo_coop, fuel_type, cats_allowed,** and **dining_room_type. garage_exists** is a categorical feature that takes on binary values: **1** if there is a garage, and **0** if the garage does not exist. **condo_coop** is another binary feature where **1** means it is a coop and **0** if not. **fuel_type** describes the types of fuel being used in the apartment and this ranges from the four values of: **electric, gas, oil,** and **other. cats_allowed** is a binary feature where **1** means cats are allowed and **0** if not allowed. Finally, **dining_room_type** has a range of four values between **formal, dining, combo,** and **other**.

These five features are strictly categorical and do not possess any numerical value. We will need to be careful when including these variables later in our model since they aren't assigned a value and may need to dummify them later. Dummy variables are categorical values that are included in a model with a value of 0 or 1 to help represent any shifts with the presence or absence of them when fitting your model.

The five numerical values include: **total_taxes, num_total_rooms, num_floors_in_buildings, num_full_bathrooms,** and **parking_charges. num_total_rooms** is the feature representing the total number of rooms in an apartment, and this ranges from 1 to 8. **num_floors_in_buildings** represents the total number of floors in an apartment building, and this ranges from 1 to 34. **num_full_bathroom** represents the total number of bathrooms in an apartment and this ranges from 1 to 3. **parking_charges** represents the cost of parking and doesn't have a significant range but is observed in United States currency. These numerical values will be important for our model as they will be imperative to predicting a logical price for each apartment in Queens.

2.3 Errors and Missingness

There are several occurrences of errors and missing values in this dataset and they must be mitigated. The first step I took was to identify which column we'll be editing to store the predicted values. The **sale_price** column is the best choice and will be set aside as the resulting prediction. In mathematical modeling, this resulting prediction is read as **y** (such as any equation, x+1 = y, for example). In our case, instead of X we'll use **H** (the hypothesis set) which is the data used to make the predictions.

The first issue with erroneous data that had to be cleared as dealing with the categorical values. I assigned the appropriate categorical values, **garage_exists, cats_allowed,** and **condo_coop** with binary values. Next, I assigned the remaining categorical values that aren't binary features with their appropriate values. Such as assigning **fuel_type** with gas, oil, electric, and other. With the numerical values, I rewrote the values to store the strings as numerical values to avoid any further issues when fitting the model(s). I filtered out null values per **sale_price** column and deleted them accordingly to avoid any messy interactions when fitting the models. These null values will be included in the expanded feature set later with the **null** variable.

3. Modeling

We will be exploring three different models to predict Queens apartment prices: **Regression Tree, Linear Modeling,** and **Random Forest.**

3.1 Regression Tree Modeling

A regression tree is a model that allows you to compute both categorical and numerical values. This is helpful in our case since we already established we will be using a combination of both categorical and numerical values. The top ten most important features that will be computed in this model include: the categorical features **garage_exists, condo_coop, fuel_type, cats_allowed,** and **dining_room_type,** and the numerical features **total_taxes, num_total_rooms, num_floors_in_buildings, num_full_bathrooms,** and **parking_charges.** For this model I started by establishing the tree nodes by parting the X test data with the X and Y training data. This was done to establish the 'leaves' of the regression tree. The reason why we part the X test data with the X and Y training data is because it is larger and removes any potential underfitting of the data. There is room for overfitting but with the use of only the top ten features I feel it is wiser to create more overfitting than underfitting as there will likely be less variation and MSE errors when computing the result.

3.2 Linear Modeling

Linear models can help you make predictions by regressing continuous variables as a function with predictor variables. After setting the training and testing data for X, Y, and B, I set the linear model to **lm(b_train~ y_train + x_train).** I also created an OOS error for the line model to **predict(line_model, b_test)**. There is an R^2 out of sample error is .81 with a RMSE error of 340. I can observe in-error statistical errors when computing the OOS because there is a large variation when comparing the standard errors of in-sample OOS and out of sample OOS. This gross statistical error leaves enough room for another model, such as the Random Forest Model, to have an improved prediction since RF models do not handle OOS errors like Linear modeling in the instance that it compares different node predictions instead of a simple vanilla linear model.

Another issue with my linear model is that it shows a steady growth with the X and Y training datas, implying that if a house were to have *more* of a certain feature, such as dining rooms or bathrooms, that it will simply be a higher cost. This is obviously incorrect, as there are more factors that determine an apartment's overall price: such as if it has a garage or parking charges. A linear model can be a good predictor for Queens apartment prices but can be improved by adding more relevant features to prevent underfitting, such as features I did not include from the raw data.

3.3 Random Forest Modeling

Random Forest is a non-parametric supervised machine learning model that uses a number of decision trees to make accurate predictions usually with the 'bagging' method. The bagging method is when the Random Forest algorithm takes multiple decision trees and combines the predictions from all these groups. This is done by iterating the samples of all the decision tree ensembles and returning a new sample prediction. When replacing the samples it reduces correlation from the decision trees in the same area. After this is done, the Random Forest model takes the Mean Square Error and determines whether the model is underfit or overfit. With the task of predicting Queens apartment prices and acknowledging that overfitting isn't as dangerous as underfitting given the 10 features I included, the Random Forest Model is likely the best choice for making these predictions.

4. Performance Results for your Random Forest Model

The Random Forest model has the greatest R^2 and RMSE in comparison to the Linear and Regression Models. When testing the split-testing data, the R^2 and RMSE in out of sample values have the best range in terms of accuracy in predicting the prices. In my Random Forest model, I included the **predict(rf_model, x_test)** after modeling **rf_model** with **rf_data** and the training data for both x and y values. The Random Forest model avoids overfitting as predicted from creating the earlier Linear and Regression models. Table 1 has a summary of these errors.

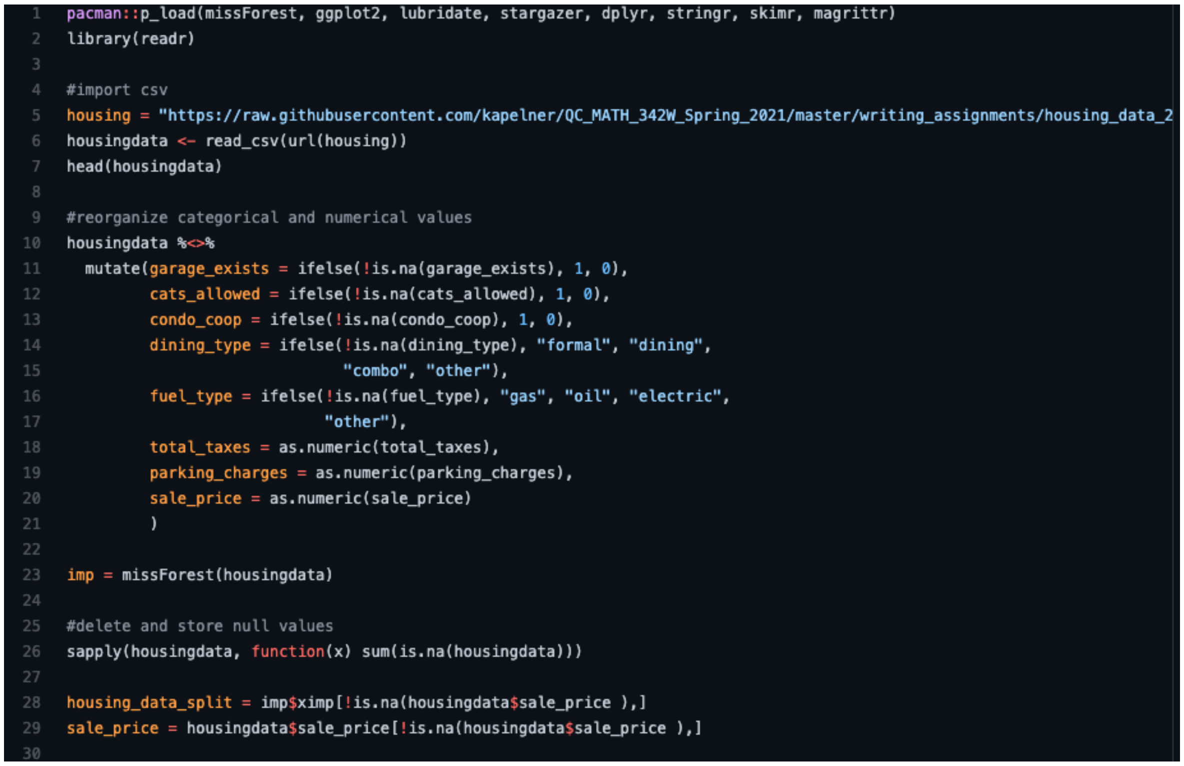
Model	R^2	RMSE
Regression Model	.81	405
Linear Model	.87	331
Random Forest	.88	301

5. Discussion

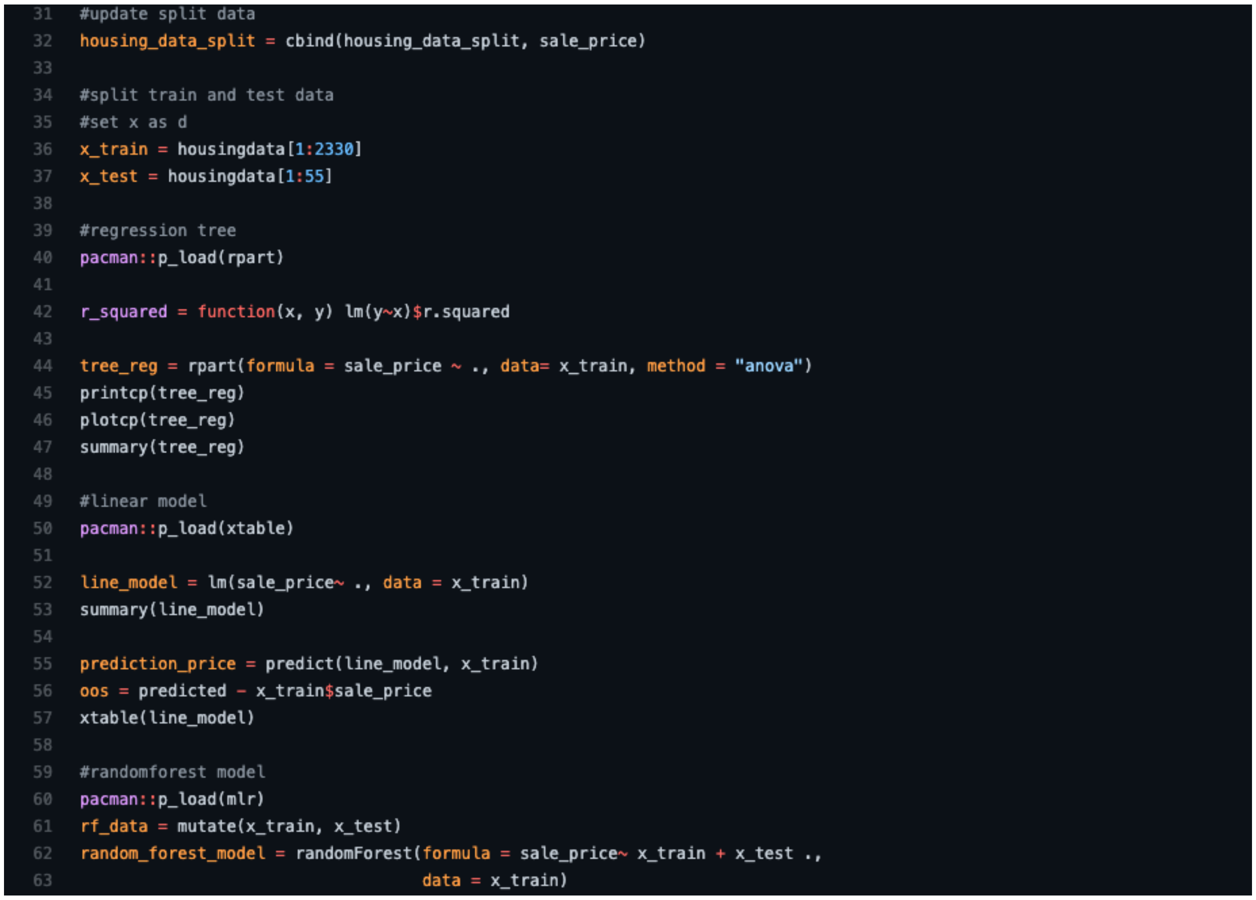
The goal of this paper is to create and compare predictive models on apartment sale prices in Queens, NY. The Random Forest Model remains the most accurate predictor in terms of estimating Queens apartment prices. This is largely due to the fact that Regression and Linear models aren't fit to measure the impact of certain categorical variables. However, Regression and Linear models can handle dummifying these variables and giving some insight to explaining the value of these categorical values even if they aren't assigned a numerical value.

Another argument for why we didn't get the most accurate prediction is due to a low sample size. This was articulated when building the Regression and Linear Models. Underfitting was the biggest issue with my model given that I only used the top ten features when computing the models. With a larger sample size, we may observe greater variance when comparing the nodes but overall a more accurate prediction.

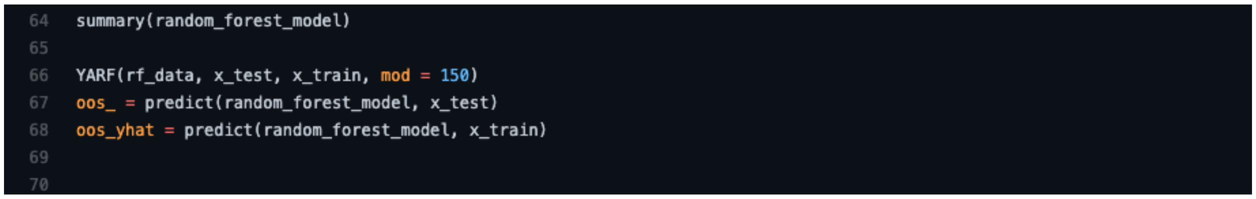
Code Appendix



```r
pacman::p_load(missForest, ggplot2, lubridate, stargazer, dplyr, stringr, skimr, magrittr)
library(readr)

#import csv
housing = "https://raw.githubusercontent.com/kapelner/QC_MATH_342W_Spring_2021/master/writing_assignments/housing_data_2
housingdata <- read_csv(url(housing))
head(housingdata)

#reorganize categorical and numerical values
housingdata %<>%
  mutate(garage_exists = ifelse(!is.na(garage_exists), 1, 0),
         cats_allowed = ifelse(!is.na(cats_allowed), 1, 0),
         condo_coop = ifelse(!is.na(condo_coop), 1, 0),
         dining_type = ifelse(!is.na(dining_type), "formal", "dining",
                              "combo", "other"),
         fuel_type = ifelse(!is.na(fuel_type), "gas", "oil", "electric",
                            "other"),
         total_taxes = as.numeric(total_taxes),
         parking_charges = as.numeric(parking_charges),
         sale_price = as.numeric(sale_price)
         )

imp = missForest(housingdata)

#delete and store null values
sapply(housingdata, function(x) sum(is.na(housingdata)))

housing_data_split = imp$ximp[!is.na(housingdata$sale_price ),]
sale_price = housingdata$sale_price[!is.na(housingdata$sale_price ),]

```



```r
#update split data
housing_data_split = cbind(housing_data_split, sale_price)

#split train and test data
#set x as d
x_train = housingdata[1:2330]
x_test = housingdata[1:55]

#regression tree
pacman::p_load(rpart)

r_squared = function(x, y) lm(y~x)$r.squared

tree_reg = rpart(formula = sale_price ~ ., data= x_train, method = "anova")
printcp(tree_reg)
plotcp(tree_reg)
summary(tree_reg)

#linear model
pacman::p_load(xtable)

line_model = lm(sale_price~ ., data = x_train)
summary(line_model)

prediction_price = predict(line_model, x_train)
oos = predicted - x_train$sale_price
xtable(line_model)

#randomforest model
pacman::p_load(mlr)
rf_data = mutate(x_train, x_test)
random_forest_model = randomForest(formula = sale_price~ x_train + x_test .,
                                   data = x_train)
```



```r
summary(random_forest_model)

YARF(rf_data, x_test, x_train, mod = 150)
oos_ = predict(random_forest_model, x_test)
oos_yhat = predict(random_forest_model, x_train)


```