Final project for Math 342W Data Science at Queens College
5/25/2021

By: Christian Guaraca

Abstract

Students test their skills against a well-known house hunting application, Zillow. Students were challenged to come up with an algorithm to surpass Zillow with their so called, Zestimates. With strong features and correcting errors throughout the data the predictive outcome could be estimated. Students were to utilize RStudio and various models such as, RandomForest, Linear, and regression trees. With these tools at hand, a model can be made to find a better prediction than Zillow's "Zestimates".

---

1. Introduction

The data given initially is on apartments and the information on these particular apartments are what is provided when looking to buy an apartment. The features given which would be known to be the casual inputs, include sale price, garage, tax, and more. It would be best to know the true casual inputs known as (z_1,z_2,…,z_n) but it is not possible as we do not know all of them.

Therefore, we settle for the data provided to us. Since the data is focusing on apartments for sale in Queens, we will be labeling the apartments sale prices as the response variable. Although, there will be error along the way, thus the predictive model will be the features of the data and the error that comes along with it. In order to solve this model, we will have to utilize models from RStudio like regression tree model, linear models, and random forest model.

2. The Data

The data given represents apartment selling prices for the most part around Queens. It provides us with features that they represent them to be what ultimately decides the price of an apartment selling in Queens. The data comes from a raw data representation from MLSI. The features provided include: approx. year built, cats and dogs allowed, common charges, community district number, coop or condo, date of sale, kitchen and dining room type, fuel type, address, garage, maintenance cost, # of bedrooms, # of floors, # of bathrooms, # of total rooms, Parking charges, tax percent deductible, sale price, total taxes, and walk score. Each feature will be further explained throughout the paper.

2.2. Featurization

The data provided seem to have an abundance of data, although there were many features that had missing data in it. For example, the response metric, sale price, had to be cleaned because for the most part it had NA which means that the sale price for a particular home was not given. Then, there was unnecessary data given. For example, url, last 7 or 30 days approval rate, work time in seconds, requester feedback, rejection time, approval time, submit time, accept time, assignment status, worker ID, and much more.

Of course, these observations had to be taken off when dealing with the model. Essentially, they were taken off because they did not provide any type of effect towards our response metric. A feature which would not be taken off was the coop_condo feature. It was clear from the data that this feature was one of the strongest, if not strongest, feature that affected the response metric. Although, the weakest feature that was included in the model was the # of half bathrooms. This was mainly because most apartments did not have this and the ones that did have them were relatively closer to the average of the other homes that did not.

2.3. Errors and Missingness

There were some errors scattered throughout the dataset. The first one I found was in the feature kitchen_type. There were three types of kitchens combo, efficiency and eat in kitchens. Although, through the data you would catch a fourth type, “efficiency kitchen”. This is the same as efficiency, but they are labeled differently in the observations. This causes problems as it introduces two types while they are both the same type.

Another error was caught by another student. They pointed out that there were certain zip codes from the data that were not included in table 1. The zip codes that did not correspond were: 13627, 27110, 25175, 27010, 26910, 18915, 14433.

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

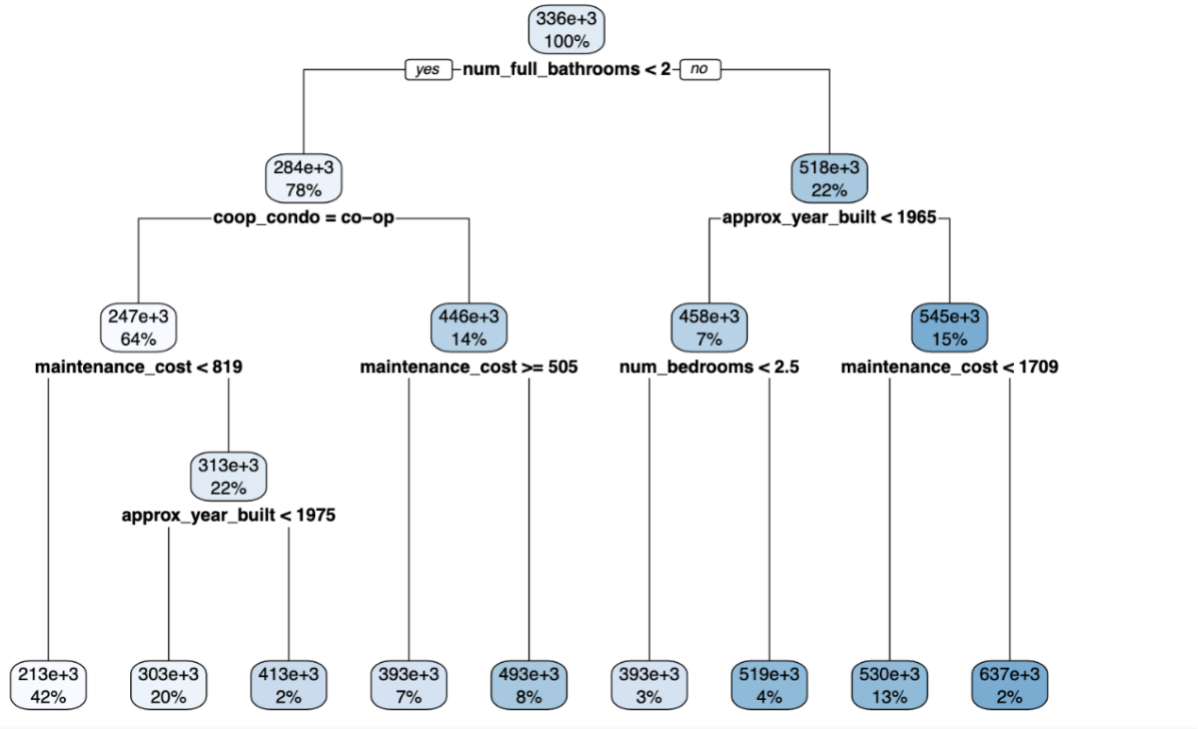
Table 1: The zip codes for the houses in the dataset. They call come from mainland Queens. We are leaving out the Rockaways, a peninsula near JFK airport that is geographically distinct from the rest of the neighborhoods.

3.Modeling

This model that was created is only the tip of the iceberg. Only by utilizing models such as random forest, a great deal could be accomplished. Although, there are still many other models that could be utilized. That is where it begins to become even more interesting, thus it must be explored.

3.1 Regression Tree Modeling

As previously mentioned, a model that would be useful for visualizing the important features that it finds to be crucial for the response metric. Once the data was applied to the regression tree model it resulted in:



This tree shows the features that the tree splits. Immediately, the most crucial split is the top one known as the root node. The top represents the number of full bathrooms < 2. This would mean that an apartment having 2 bathrooms would affect its sale price significantly. Following this is the runner-up which would be whether if the apartment is a co-op or a condo. Following that it splits at whether the maintenance cost is < 819. Finally ending at the approximate year it was built < 1975.

3.2 Linear Modeling

For linear modeling the OLS is utilized. The in sample for R-Squared was 81.94% and the RMSE was 60,670. On the other hand, the R-Squared for the OOS OLS was 55.35% and the RMSE was 67829. The out of sample model is the better model because the prediction is plus or minus $67,829. From the table below you can also see that coop or condo seem to have a high estimate meaning that it must have been crucial to the OLS.

```
Call:
lm(formula = sale_price ~ ., data = housing_train)

Residuals:
   Min     1Q  Median     3Q    Max
-329834  -31317   -3130   29837  364635

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                     -2.545e+06  2.037e+05 -12.497  < 2e-16 ***
approx_year_built                1.264e+03  9.804e+01  12.896  < 2e-16 ***
cats_allowed                    -8.503e+03  4.190e+03  -2.029 0.042575 *
coop_condocondo                  1.648e+05  6.557e+03  25.140  < 2e-16 ***
dogs_allowed                     1.824e+04  4.687e+03   3.892 0.000103 ***
dining_room_typedining area     -1.362e+04  4.326e+04  -0.315 0.752964
dining_room_typeformal           7.623e+03  3.357e+03   2.271 0.023262 *
dining_room_typenone             4.861e+04  3.540e+04   1.373 0.169833
dining_room_typeother            2.602e+04  4.782e+03   5.441 5.96e-08 ***
fuel_typegas                    -3.317e+04  8.512e+03  -3.897 0.000101 ***
fuel_typenone                   -1.229e+04  3.632e+04  -0.339 0.735015
fuel_typeoil                    -3.210e+04  8.708e+03  -3.686 0.000234 ***
fuel_typeother                  -8.048e+03  1.354e+04  -0.594 0.552422
fuel_typeOther                   9.053e+04  6.212e+04   1.457 0.145202
garage_exists                    2.918e+03  3.879e+03   0.752 0.451977
kitchen_typecombo               -1.319e+04  6.257e+04  -0.211 0.833014
kitchen_typeCombo                1.404e+04  6.287e+04   0.223 0.823275
kitchen_typeeat in              -8.665e+03  6.233e+04  -0.139 0.889443
kitchen_typeEat in               3.343e+04  7.574e+04   0.441 0.658953
kitchen_typeEat In              -2.221e+04  6.392e+04  -0.348 0.728213
kitchen_typeeatin               -2.287e+04  6.252e+04  -0.366 0.714595
kitchen_typeefficiemcy          -3.342e+04  7.580e+04  -0.441 0.659343
kitchen_typeefficiency          -3.682e+04  6.234e+04  -0.591 0.554771
kitchen_typeefficiency kitchen  -3.061e+04  6.259e+04  -0.489 0.624800
kitchen_typeefficiency kitchene -5.359e+03  7.692e+04  -0.070 0.944459
kitchen_typeefficiency ktchen   -2.599e+04  7.610e+04  -0.342 0.732723
kitchen_typenone                -1.523e+04  6.421e+04  -0.237 0.812496
maintenance_cost                 1.622e+02  6.015e+00  26.969  < 2e-16 ***
num_bedrooms                     2.920e+04  3.305e+03   8.834  < 2e-16 ***
num_full_bathrooms               6.779e+04  4.546e+03  14.913  < 2e-16 ***
num_total_rooms                  1.852e+03  1.926e+03   0.962 0.336304
walk_score                       1.511e+03  1.054e+02  14.334  < 2e-16 ***
missing_data_approx_year_built  -3.117e+03  1.123e+04  -0.278 0.781406
missing_data_dining_room_type    3.543e+03  3.533e+03   1.003 0.316091
missing_data_fuel_type           4.154e+02  6.642e+03   0.063 0.950139
missing_data_kitchen_type        1.028e+04  1.781e+04   0.577 0.564000
missing_data_maintenance_cost   -2.392e+04  5.367e+03  -4.456 8.82e-06 ***
missing_data_num_bedrooms       -6.777e+03  7.037e+03  -0.963 0.335646
missing_data_num_total_rooms     6.056e+04  4.674e+04   1.296 0.195236
```

```
missing_data_sale_price        1.218e+04  6.296e+03   1.934 0.053300 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 60670 on 1967 degrees of freedom
Multiple R-squared:  0.8194,            Adjusted R-squared:  0.8159
F-statistic: 228.9 on 39 and 1967 DF,  p-value: < 2.2e-16
```

3.3 Random Forest Modeling

Finally, the random forest model will be presented last, but is definitely one of the algorithms which should be your choice for prediction models. Its name refers to a forest because there are a bunch of regression trees that go into it, thus resulting in a forest. With the random forest there are multiple trees and each one creates a prediction; the best prediction is the outcome. Although, the random forest model is non-parametric. Also, a lose taken when using the random forest is that it is not as interpretable as its fellow regression tree and linear models. This model is also great since it has a reduced bias and variance. With all this at hand the random forest model will get the best prediction it can by acquiring as much data as it can from the data set.

4.Performance Results for your Random Forest

YARF was not available to be used, therefore random forest was used. The R-squared retrieved was 92.89.

```
Call:
 randomForest(formula = sale_price ~ ., data = housing_imp)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 1540265839
                    % Var explained: 92.89
```

5.Discussion

Ultimately, these three models were an amazing tool to use. The results could have been better if it was not for all the missing data that there was. There were even unnecessary data that were included in the data set which had to be dropped. The phenomenon of apartment sale price was therefore affected by these issues. As a result of this, the models created did not reach the potential it could have had.

Code Appendix

````
---
title: "Final Project"
author: "Christian Guaraca"
output:
  pdf_document: default
  html_document: default
---

```{r}
#import housing_data
housing_data = read.csv(file = '/Applications/MAT 342/housing_data.csv')
```



```{r}
#load packages
pacman::p_load(tidyverse, mlr, mlr3, missForest, skimr, rpart, randomForest, data.table,
dplyr, magrittr, rpart.plot)
```

```{r}
#picking certain data that does not apply in my opinion
housing_data_select = housing_data %>%
  select(approx_year_built, cats_allowed, coop_condo, dogs_allowed,
dining_room_type, fuel_type,
       garage_exists, kitchen_type, maintenance_cost, num_bedrooms,
num_floors_in_building,
       num_full_bathrooms, num_total_rooms, parking_charges, sale_price, sq_footage,
total_taxes, walk_score)
```

```{r}
setDT (housing_data_select)
```

```{r}
dim(housing_data_select)
str(housing_data_select)
```
````

```{r}
#removing any data with excessive NA
housing_data_drop = housing_data_select %>%
  select(-parking_charges, -sq_footage, -total_taxes, -num_floors_in_building)
```

```{r}
#Adjusting the features so they can be used to run the algorithms
housing_data_new = housing_data_drop %>%
  mutate(cats_allowed = ifelse(cats_allowed == "yes", 1, 0)) %>% #set data to binary
  mutate(dogs_allowed = ifelse(dogs_allowed == "yes", 1, 0)) %>% #set data to binary
  mutate(maintenance_cost = as.numeric(gsub('[$,]', '',
housing_data_drop$maintenance_cost))) %>% #remove $ and , from obs
  mutate(sale_price = as.numeric(gsub('[$,]', '', housing_data_drop$sale_price))) %>%
#remove $ and , from obs
  mutate(coop_condo = factor(coop_condo, ordered = FALSE)) %>%
  mutate(dining_room_type = factor(dining_room_type, ordered = FALSE)) %>%
  mutate(fuel_type = factor(fuel_type, ordered = FALSE)) %>%
  mutate(kitchen_type = factor(kitchen_type, ordered = FALSE)) %>%
  mutate(garage_exists = ifelse(is.na(garage_exists), 0, 1)) #making sure NA is turned to 0
```

```{r}
housing_data_new %>%
  filter(!is.na(sale_price))
```

```{r}
missing_data = tbl_df(apply(is.na(housing_data_new), 2, as.numeric))
colnames(missing_data) = paste("missing_data_", colnames(housing_data_new), sep =
"")
missing_data %<>%
  select_if(function(x){sum(x) > 0})
housing_imp = missForest(data.frame(housing_data_new))$ximp
housing = cbind(housing_imp,missing_data)
```

```{r}
#making train and test split
test_prop = 0.1

#test
```

```
test_indices = sample(1:nrow(housing), round((test_prop)*nrow(housing)))
housing_test = housing[test_indices,]
y_test = housing_test$sale_price
X_test = cbind(1, housing_test)
X_test$sale_price = NULL

#train
train_indices = setdiff(1:nrow(housing), test_indices)
housing_train = housing[train_indices,]
y_train = housing_train$sale_price
X_train = cbind(1, housing_train)
X_train$sale_price = NULL
n_train = nrow(X_train)
```

```{r}
#Create Regression tree model

#use rpart, YARF not availible for me
#In-sample Error
Reg_tree = rpart(y_train~., housing_train)
y_hat_train = predict(Reg_tree, housing_train)
e_in = y_train - y_hat_train
rsme_in = sd(e_in)
r_squared_in = (var(y_train)-var(e_in)) / var(y_train)

#OOSE
y_hat_test = predict(Reg_tree, housing_test)
e_oose = y_test - y_hat_test
rsme_oose = sd(e_oose)
rsquared_oose = (var(y_test) - var(e_oose)) / var(y_test)
```

```{r}
fit_model = rpart(housing_train$sale_price~., data.frame(X_train), method="anova")
rpart.plot(fit_model)
fit_model
```

```{r}
#OLS In-sample
linear_model = lm(sale_price ~., housing_train)
```

```
summary(linear_model)
```

```{r}
#OLS OOS
y_hat_linear = predict(linear_model, housing_test)
e = y_test - y_hat_linear
RMSE = sd(e)
rsquared = 1 - RMSE / sd(y_test)
RMSE
rsquared
```

```{r eval=FALSE}
#mlr attempt
mod_task = makeRegrTask(data = data.frame(X_train), target =
'housing_data$sale_price')
algor = makeLearner("regr.rpart")
valid = makeResampleDesc("CV", iteration = 5)
resample = resample(algor, mod_task, valid, measures = list(rmse))
resample
mean(resample$measures.test$rmse)
```

```{r}
#Random Forest
random_forest = randomForest(sale_price~.,housing_imp)
random_forest

```