Predicting Sale Prices for Condos and Co-ops in Queens

Final project for Math 390 Data Science at Queens College
May 24, 2020

By Alin Carrera

Abstract

The goal of this project was to predict sale prices for both condos and co-ops in Queens, NY, using data from February, 2016 to February, 2017. This was done by implementing 3 different algorithms: linear modeling, regression tree modeling, and random forest modeling. Proper analysis of each algorithm was discussed. However, selecting features that would appropriately predict the selling prices was the most important step needed in order for the algorithms to be implemented, along with properly assessing missing entries.

1. Introduction

For this project, data that was collected in the span of a year from 2016 to 2017 is used to make predictions for the sale prices of condos and co-ops in Queens, NY. The purpose is to create a model that will do so as accurately as possible. Models in fact will never be perfect because there is no way to model reality in its entirety. However, it is possible to get better predictions by enhancing the model. In this case, using past data to create the models, making predictions on it, and comparing it to the actual recorded sale price is a good way to determine how well these models can predict in the future. There will be three different algorithms used to create the models: Linear modeling, Regression Tree modeling and Random Forest modeling. For each, there will be a set of features from the data set that will be used to give predictions on the sale price per apartment.

2. The Data

The data was gathered from Multiple Listing Service of Long Island, Inc. This data was collected from 2016 to 2017. Initially, the data set consisted of 55 features and 2,230 observations, which are the columns and rows, respectively. Looking through the data there were some features that stood out in terms of explaining the price for the apartments. For example, if an apartment has a garage then it will be more of a suitable fit for someone moving in that owns a car and explains why the price of said apartment will be higher than others. Due to the fact that there were features that seemed useless in regards to the sale price of an apartment, features had to be picked with careful consideration.

2.1. Featurization

From the 55 features, 16 were chosen including the feature being predicted, *sale_price*, which is a continuous variable. The remaining 15 features were chosen because they give relevant information on the apartments having a higher/lower selling price compared to others. Using *skim(housing_data)* I was able to analyze the features carefully and select the ones that would help explain the price for the apartments, as it breaks down the variables and shows the number of missing entries. The following variables are continuous: *approx_year_built, num_bedrooms, num_floors_in_building, num_total_rooms, num_full_bathrooms, sq_footage, walk_score, maintence_cost*. The remaining are categorical variables: *cats_allowed, coop_condo, dogs_allowed, dining_room_type, fuel_type, garage_exists, kitchen_type*.

Since the values for *sale_price* and *maintence_cost* are given with the “$” symbol, they had to be adjusted to be numerical values. That way the algorithms can be run on the data. The variables *dogs_allowed* and *cats_allowed* were converted to binary, as the responses for both were either yes or no, so it would be 1 for yes and 0 for no. I chose to convert *garage_exists* to binary as well because some of the responses for this variable were variations of "yes", either it was spelt in all lower case, in all capital letters, or a combination of both. The different variations of "yes" were assigned the value of 1 and the rest of the entries, which were NA's, were assigned the value of 0.

Majority of these features are self explanatory, in terms of describing aspects of the apartment and the pricing for them. For example, *sq_footage, num_full_bathrooms, num_bedrooms, num_floors_in_building, num_total_rooms, and approx_year_built*. The remaining features are extras that might incentivize a potential buyer on deciding on which apartment to buy.

2.2. Errors and Missingness

While going through the data, it was quite noticeable that there was a lot of missing data for some features and errors. An error that also led me to convert *garage_exists* into a binary feature was that from the different variations of "yes" one was spelt incorrectly, "eys". Thus, changing the variable would make it easy to work with and will avoid issues with the misspelled response. Missingness in the data was prominent, as a lot of features from the initial data set had to be dropped because it would result in poor performance.

After selecting which features would be used, the data still had 2,230 observations, from which some of these observations were *sale_price*. The observations that had the *sale_price* missing were dropped as well because trying to predict the *sale_price* in supervised learning would not make sense. Once these observations were dropped, there were only 528 observations left in our data set. From these 528 observations, some of them had missing values for features like f*uel_type, maintenance_cost, sq_footage*, and so on. MissForest was used to impute the missing values. This replaced the missing values with predictions in the corresponding entries. The final data set consisted of 15 features (not counting the one being predicted on), 528 observations and no missing data whatsoever.

```
── Data Summary ────────────────────────
                           Values
Name                       housing_tbl_imp
Number of rows             528
Number of columns          16
_______________________
Column type frequency:
  factor                   4
  numeric                  12
________________________
Group variables            None

── Variable type: factor ──────────────────────────────────────────────────────
────────
  skim_variable    n_missing complete_rate ordered n_unique top_counts
1 coop_condo               0             1 FALSE          2 co-: 399,
con: 129
2 dining_room_type         0             1 FALSE          4 com: 329,
for: 136, oth: 60, din: 3
3 fuel_type                0             1 FALSE          6 gas: 317,
oil: 188, ele: 11, oth: 8
4 kitchen_type             0             1 FALSE          7 eff: 232,
eat: 194, Com: 51, com: 31

── Variable type: numeric ─────────────────────────────────────────────────────
───────
   skim_variable         n_missing complete_rate       mean
sd     p0     p25     p50     p75
 1 approx_year_built             0             1   1962.         20.5
1915   1950    1956    1966.
 2 cats_allowed                  0             1      0.460
0.499     0      0       0       1
 3 dogs_allowed                  0             1      0.278
0.449     0      0       0       1
 4 garage_exists                 0             1      0.178
0.383     0      0       0       0
 5 maintenance_cost              0             1    813.       350.
155    638.    728.    869
 6 num_bedrooms                  0             1      1.54
0.748     0      1       1       2
 7 num_floors_in_building        0             1      7.10       6.31
1      3       6       7
 8 num_full_bathrooms            0             1      1.20
0.422     1      1       1       1
 9 num_total_rooms               0             1      4.02       1.20
1      3       4       5
10 sale_price                    0             1 314957.    179527.
55000 171500  259500  428875
```

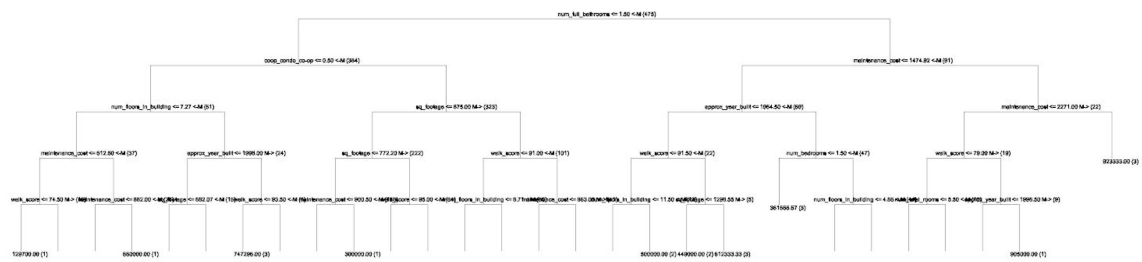
3. Modeling

3.1. Regression Tree Modeling

The Regression Tree algorithm created splits based on the features that have the most importance on the *sale_price* of an apartment. The tree made the first split on the num_full_bathrooms in the apartment if it is less than 1.5, thus telling us that this feature has the greatest effect on the price. After this split there are two more, one for when the number of bathrooms is less than 1.5 and one for when it is greater than 1.5. Looking at the left hand side, which corresponds to less than 1.5, the next split is made on the type of apartment, whether it is condo or co-op.

The splits that follow are based on *maintenance_cost* of the apartments and the *approx_year_built.* These splits are reasonable because pricing does vary if the apartment is a condo or co-op, as well as what the maintenance cost is and when it was built.

Now, for the left hand side of the initial split based on the *num _full_bathroom*s, the next split is made on the *maintenance_cost*. If the cost is more than $1475, then the next important factor again is *maintenance_cost*. If the cost is less than $1475 then the next split will be on the *approx_year_built*. Again, these splits make sense because the more bathrooms there are the higher the price which is increased by the maintenance cost.



3.2. Linear Modeling

Fitting the OLS linear model on the data resulted with the value of R^2 being 79%, nearly 80%, and am RMSE of 85,290. The performance of this model isn't all that great, as it performed worse than the Regression Tree model. The R^2 is a little low and the RSME is relatively high. Looking at the same features that the Regression tree found to be more important it is noticeable that in the linear model the feature that has the greatest effect on the price is the type of apartment, that is condo or co-op. The next feature that also has a big impact on the price is *num_full_bathrooms,* which is not surprising. The reason that these two features are important in the linear model is not surprising is because in the tree they were top features. Another important feature in the linear model is num_bedrooms. From this point forward the features that are important for the tree are different than the ones from the linear model. This doesn't mean that the linear model is bad at predicting the sale price.

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

Residuals:
    Min      1Q  Median      3Q     Max 
-375325  -48822   -2968   39322  391557 

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)    
(Intercept)                 400204.23  642037.66   0.623   0.5334    
approx_year_built             -314.24     326.20  -0.963   0.3359    
cats_allowed                 16332.03   11164.78   1.463   0.1442    
coop_condocondo             220097.15   15238.27  14.444  < 2e-16 ***
dogs_allowed                  7464.55   12451.45   0.599   0.5491    
dining_room_typedining area   8830.28   50721.82   0.174   0.8619    
dining_room_typeformal        7909.04   10751.93   0.736   0.4624    
dining_room_typeother        29687.94   13497.26   2.200   0.0283 *  
fuel_typegas                 22867.48   32995.93   0.693   0.4886    
fuel_typenone                72894.66   60354.15   1.208   0.2278    
fuel_typeoil                 23671.60   33865.53   0.699   0.4849    
fuel_typeother               49905.45   44806.50   1.114   0.2660    
fuel_typeOther              153610.43   93499.48   1.643   0.1011    
garage_exists               -10553.32   11375.06  -0.928   0.3540    
kitchen_typecombo           -20983.22   88609.87  -0.237   0.8129    
kitchen_typeCombo             6345.58   87940.22   0.072   0.9425    
kitchen_typeeat in           -8162.61   87182.47  -0.094   0.9254    
kitchen_typeEat in           88141.74  107698.47   0.818   0.4136    
kitchen_typeEat In          -16636.22   89968.62  -0.185   0.8534    
kitchen_typeefficiency      -33573.51   87309.39  -0.385   0.7008    
maintenance_cost               119.35      19.54   6.107 2.20e-09 ***
num_bedrooms                 62355.33    9530.63   6.543 1.65e-10 ***
num_floors_in_building        6236.51     856.54   7.281 1.50e-12 ***
num_full_bathrooms           69648.54   14183.71   4.910 1.28e-06 ***
num_total_rooms                785.65    6326.71   0.124   0.9012    
sq_footage                      11.78      16.14   0.730   0.4656    
walk_score                    1498.90     355.17   4.220 2.96e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 85290 on 448 degrees of freedom
Multiple R-squared:  0.7891,	Adjusted R-squared:  0.7768 
F-statistic: 64.46 on 26 and 448 DF,  p-value: < 2.2e-16
```

3.3. Random Forest Modeling

The Random Forest algorithm should be the go to algorithm when making the predictions. That is because when using Random Forest it decorrelates the trees during construction by taking a subset of the features and this introduces randomness. From this the trees are split randomly and are grown as they normally would in the tree algorithm. This algorithm decreases variance a lot, without really increasing bias. With this algorithm there is no worry of underfitting as this algorithm takes the averages of all the different trees which leads to the decrease of the variance in the model. This model is non-parametric because the size of the subsets chosen can increase depending on how big the sample size is. Even though all the features used in the models affect the sale price in one way or another, I would have to say that the following features have the greatest impact on the price: *coop_condo, num_bedrooms, num_full_bathrooms, num_total_rooms, sq_footage*. These features seem to be very similar to the ones chosen in all the three models.

4. **Performance Results for your Random Forest Model**

Clearly, Random Forest performed extremely well compared to the two other algorithms. This model yielded an oob R^2 of 97% and an RMSE of 28,719. There is a vast increase in R^2 and a huge decrease in RSME. These results are a valid estimate because the way the algorithm works, creating random splits on a subset of the features. When validating these results to a hold-out test set created from the initial train-test split done to the data, the R^2 value is 98.5% and the RSME is 20,599. This gives incredible results in the validation of the Random Forest algorithm. These values are good at indicating that the model will predict well enough on data it has not seen.

```
OOB results on all observations:
  R^2: 0.97436
  RMSE: 28719.01
  MAE: 14829.55
  L2: 435484592399
  L1: 7830000
[1] 20599.22
[1] 0.9853821
```

5. **Discussion**

Through the application of the three different algorithms, I was able to get better predictions on the sale prices with Random Forest. The second one with the next best results was the Regression Tree and Linear model came in last. Looking at the results we got from the different algorithms it is seen that they performed slightly better than the estimates provided by Zillow. There is no doubt that these models could give more accurate predictions if maybe there were more observations to use.

More than half observations were dropped when removing observations that had NA as a response for *sale_price,* this could lead the models to give more predictions in the future. Using these models for other data sets that have similar features and have similar numbers of observations will most likely give good predictions, of course that it is if there isn’t a lot of data missing. However, if they were used on data sets that have more features and a larger number of observations then it can give poor predictions. It’s hard to explicitly say what modifications would be made on the model when new data sets contain more relevant features that affect pricing. For example, if the apartment is closer to train stations or if it's located on a popular avenue/street/neighborhood then pricing would be higher than an apartment that is not near any any of these things. Then, cleaning of the data would require more attention, as such features would need to be properly adjusted in order to use in the model. Luckily, like all models, it can be improved to give better predictions for larger data sets for the future.

Final Project

Alin Carrera

May 24, 2020

```r
pacman::p_load(tidyverse, magrittr, data.table, missForest, skimr)

housing_data = read.csv("housing_data_2016_2017.csv", header = TRUE)

skim(housing_data)
```

Data summary

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

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
HITId	758	0.66	FALSE	1472	301: 1, 301: 1, 301: 1, 302: 1

HITTypeId	758	0.66	FALSE	2	310: 944, 36B: 528
Title	758	0.66	FALSE	1	Fin: 1472
Description	758	0.66	FALSE	2	Got: 944, Go : 528
Reward	758	0.66	FALSE	1	$0.: 1472
CreationTime	758	0.66	FALSE	62	Thu: 43, Thu: 40, Wed: 39, Thu: 37
RequesterAnnotation	758	0.66	FALSE	2	Bat: 944, Bat: 528
Expiration	758	0.66	FALSE	62	Thu: 43, Thu: 40, Wed: 39, Thu: 37
AssignmentId	758	0.66	FALSE	1472	301: 1, 301: 1, 304: 1, 304: 1
WorkerId	758	0.66	FALSE	73	A23: 187, A1S: 129, A3C: 124, AHX: 114
AssignmentStatus	758	0.66	FALSE	1	App: 1472
AcceptTime	758	0.66	FALSE	1457	Thu: 2, Thu: 2, Thu: 2, Thu: 2
SubmitTime	758	0.66	FALSE	1460	Thu: 2, Thu: 2, Thu: 2, Thu: 2
AutoApprovalTime	758	0.66	FALSE	1460	Thu: 2, Thu: 2, Thu: 2, Thu: 2
ApprovalTime	758	0.66	FALSE	929	201: 6, 201: 6, 201: 5, 201: 5

LifetimeApprovalRate	758	0.66	FALSE	32	100: 187, 100: 126, 100: 124, 100: 106
Last30DaysApprovalRate	758	0.66	FALSE	32	100: 187, 100: 126, 100: 124, 100: 106
Last7DaysApprovalRate	758	0.66	FALSE	32	100: 187, 100: 126, 100: 124, 100: 106
URL	758	0.66	FALSE	1450	htt: 2, htt: 2, htt: 2, htt: 2
cats_allowed	0	1.00	FALSE	3	no: 1402, yes: 826, y: 2
common_charges	1684	0.24	FALSE	258	$25: 11, $17: 10, $27: 9, $29: 8
coop_condo	0	1.00	FALSE	2	co-: 1661, con: 569
date_of_sale	1702	0.24	FALSE	222	6/3: 7, 10/: 6, 12/: 6, 2/2: 6
dining_room_type	448	0.80	FALSE	5	com: 957, for: 620, oth: 201, din: 2
dogs_allowed	0	1.00	FALSE	3	no: 1684, yes: 544, yes: 2
fuel_type	112	0.95	FALSE	6	gas: 1348, oil: 664, ele: 62, oth: 40
full_address_or_zip_code	0	1.00	FALSE	1177	70-: 22, 269: 17, 270: 16, 73-: 14
garage_exists	1826	0.18	FALSE	6	yes: 361, Yes: 39, 1: 1, eys: 1
kitchen_type	16	0.99	FALSE	13	eat: 733, eff: 505, com: 349, eff: 338

maintenance_cost	623	0.72	FALSE	609	$54: 10, $67: 10, $68: 10, $70: 10
model_type	40	0.98	FALSE	875	1 B: 63, One: 59, 2 B: 50, Hi-: 41
parking_charges	1671	0.25	FALSE	89	$15: 42, $60: 41, $75: 27, $13: 23
sale_price	1702	0.24	FALSE	315	$15: 11, $17: 10, $13: 7, $22: 7
total_taxes	1646	0.26	FALSE	293	$13: 13, $25: 12, $4,: 11, $2,: 10
listing_price_to_nearest_1000	534	0.76	FALSE	292	$34: 28, $39: 26, $28: 25, $23: 23
url	758	0.66	FALSE	1450	htt: 2, htt: 2, htt: 2, htt: 2

Variable type: logical

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

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist

MaxAssignments	758	0.66	1.00	0.00	1	1	1	1	1	
AssignmentDurationInSeconds	758	0.66	900.00	0.00	900	900	900	900	900	
AutoApprovalDelayInSeconds	758	0.66	60.00	0.00	60	60	60	60	60	
WorkTimeInSeconds	758	0.66	162.39	111.69	22	89	127	197	815	
approx_year_built	40	0.98	1962.71	21.08	1893	1950	1958	1970	2017	
community_district_num	19	0.99	26.33	2.95	3	25	26	28	32	
num_bedrooms	115	0.95	1.65	0.74	0	1	2	2	6	
num_floors_in_building	650	0.71	7.79	7.52	1	3	6	7	34	
num_full_bathrooms	0	1.00	1.23	0.44	1	1	1	1	3	
num_half_bathrooms	2058	0.08	0.95	0.30	0	1	1	1	2	
num_total_rooms	2	1.00	4.14	1.35	0	3	4	5	14	
pct_tax_deductibl	1754	0.21	45.40	6.95	20	40	50	50	75	
sq_footage	1210	0.46	955.36	380.86	100	743	881	1100	6215	
walk_score	0	1.00	83.92	14.75	7	77	89	95	99	

```
#Feature Selection

housing = 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, sale_price, sq_footage,
walk_score)

#Adjusting the features so they can be used to run the algorithms

housing_data_tbl = housing %>%

  mutate(coop_condo = factor(coop_condo, ordered = FALSE)) %>%

  mutate(cats_allowed = ifelse(cats_allowed == "no", 0, 1)) %>%

  mutate(dogs_allowed = ifelse(dogs_allowed == "no", 0, 1)) %>%

  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(maintenance_cost = as.numeric(gsub('[$,]', '',
housing$maintenance_cost))) %>%

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

  mutate(garage_exists = ifelse(is.na(garage_exists), 0, 1))

skim(housing)
```

Data summary

Name	housing
Number of rows	2230
Number of columns	16

Column type frequency:

factor	9
numeric	7

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
cats_allowed	0	1.00	FALSE	3	no: 1402, yes: 826, y: 2
coop_condo	0	1.00	FALSE	2	co-: 1661, con: 569
dogs_allowed	0	1.00	FALSE	3	no: 1684, yes: 544, yes: 2
dining_room_type	448	0.80	FALSE	5	com: 957, for: 620, oth: 201, din: 2
fuel_type	112	0.95	FALSE	6	gas: 1348, oil: 664, ele: 62, oth: 40
garage_exists	1826	0.18	FALSE	6	yes: 361, Yes: 39, 1: 1, eys: 1
kitchen_type	16	0.99	FALSE	13	eat: 733, eff: 505, com: 349, eff: 338
maintenance_cost	623	0.72	FALSE	609	$54: 10, $67: 10, $68: 10, $70: 10

sale_price	1702	0.24	FALSE	315	$15: 11, $17: 10, $13: 7, $22: 7

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	
num_bedrooms	115	0.95	1.65	0.74	0	1	2	2	6	
num_floors_in_building	650	0.71	7.79	7.52	1	3	6	7	34	
num_full_bathrooms	0	1.00	1.23	0.44	1	1	1	1	3	
num_total_rooms	2	1.00	4.14	1.35	0	3	4	5	14	
sq_footage	1210	0.46	955.36	380.86	100	743	881	1100	6215	
walk_score	0	1.00	83.92	14.75	7	77	89	95	99	

```r
#Dropping all the observations that had NA as a response for sale_price

housing_tbl = housing_data_tbl %>%

  filter(!is.na(sale_price))

#Imputing the missing data

missing = tbl_df(apply(is.na(housing_tbl), 2, as.numeric))

colnames(missing) = paste("is_missing_", colnames(housing_tbl), sep = "")

missing %<>%

  select_if(function(x){sum(x) > 0})
```

```r
housing_tbl_imp = missForest(data.frame(housing_tbl))$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!
skim(housing_tbl_imp)
```

Data summary

Name	housing_tbl_imp
Number of rows	528
Number of columns	16
_______________________	
Column type frequency:	
factor	4
numeric	12
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
coop_condo	0	1	FALSE	2	co-: 399, con: 129
dining_room_type	0	1	FALSE	4	com: 335, for: 136, oth: 55, din: 2

fuel_type	0	1	FALSE	6	gas: 318, oil: 187, ele: 11, oth: 8
kitchen_type	0	1	FALSE	7	eff: 232, eat: 194, Com: 51, com: 31

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
approx_year_built	0	1	1962.27	20.47	1915	1950.00	1956.00	1966.50	2016	
cats_allowed	0	1	0.46	0.50	0	0.00	0.00	1.00	1	
dogs_allowed	0	1	0.28	0.45	0	0.00	0.00	1.00	1	
garage_exists	0	1	0.18	0.38	0	0.00	0.00	0.00	1	
maintenance_cost	0	1	805.09	350.31	155	626.75	722.50	866.00	4659	
num_bedrooms	0	1	1.54	0.75	0	1.00	1.00	2.00	3	
num_floors_in_building	0	1	7.12	6.31	1	3.00	6.00	7.00	34	
num_full_bathrooms	0	1	1.20	0.42	1	1.00	1.00	1.00	3	
num_total_rooms	0	1	4.02	1.20	1	3.00	4.00	5.00	8	
sale_price	0	1	314956.56	179526.60	55000	171500.00	259500.00	428875.00	999999	
sq_footage	0	1	904.21	368.60	375	705.72	833.68	996.68	6215	

walk_score	0	1	83.10	13.09	15	76.00	85.00	94.00	99

```r
#Creating the train-test split that will be used for the different algorithms
set.seed(1)
test_prop = 0.10
train_indices = sample(1 : nrow(housing_tbl_imp), round((1 - test_prop) *
nrow(housing_tbl_imp)))
housing_tbl_imp_train = housing_tbl_imp[train_indices, ]
y_train = housing_tbl_imp_train$sale_price
X_train = housing_tbl_imp_train
X_train$sale_price = NULL
n_train = nrow(X_train)
test_indices = setdiff(1 : nrow(housing_tbl_imp), train_indices)
housing_tbl_imp_test = housing_tbl_imp[test_indices, ]
y_test = housing_tbl_imp_test$sale_price
X_test = housing_tbl_imp_test
X_test$sale_price = NULL
#Regression Tree Modeling
if (!pacman::p_isinstalled(YARF)){
  pacman::p_install_gh("kapelner/YARF/YARFJARs", ref = "dev")
  pacman::p_install_gh("kapelner/YARF/YARF", ref = "dev")
}
pacman::p_load(YARF)
options(java.parameters = "-Xmx4000m")

tree_mod = YARFCART(X_train, y_train,
```

```
            bootstrap_indices = 1 : n_train, calculate_oob_error = FALSE)

## YARF initializing with a fixed 1 trees...

## YARF factors created...

## YARF after data preprocessed... 30 total features...

## Beginning YARF regression model construction...done.

illustrate_trees(tree_mod, max_depth = 5, open_file = TRUE)

get_tree_num_nodes_leaves_max_depths(tree_mod)

## $num_nodes

## [1] 385

##

## $num_leaves

## [1] 193

##

## $max_depths

## [1] 19

y_hat_train = predict(tree_mod, housing_tbl_imp_train)

## Warning in predict.YARF(tree_mod, housing_tbl_imp_train): Prediction set
column names did not match training set column names.

## Attempting to subset to training set columns.

e = y_train - y_hat_train

sd(e)

## [1] 22590.98

1 - sd(e) / sd(y_train)

## [1] 0.8748724

#Linear Modeling
```

```r
linear_mod = lm(sale_price ~ ., housing_tbl_imp_train)
summary(linear_mod)$r.squared
## [1] 0.7884677
summary(linear_mod)$sigma
## [1] 85412.3
sd(linear_mod$residuals)
## [1] 83036.73
summary(linear_mod)
##
## Call:
## lm(formula = sale_price ~ ., data = housing_tbl_imp_train)
##
## Residuals:
##    Min    1Q  Median      3Q    Max
## -381330 -48607   -2933   40558  393565
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                430834.96  642418.01   0.671  0.503
## approx_year_built            -329.33     326.24  -1.009    0.313
## cats_allowed                17034.00   11174.01   1.524  0.128
## coop_condocondo             222709.34   15308.18  14.548  < 2e-16 ***
## dogs_allowed                8314.67   12406.87   0.670    0.503
## dining_room_typedining area  16871.24   61581.28   0.274 0.784
## dining_room_typeformal     10578.84   10735.88   0.985   0.325
```

```
## dining_room_typeother      29988.02   13854.55   2.164   0.031 *
## fuel_typegas               21257.97   33049.07   0.643   0.520
## fuel_typenone              67513.64   60494.86   1.116   0.265
## fuel_typeoil               21923.77   33933.58   0.646   0.519
## fuel_typeother             43899.73   44846.98   0.979   0.328
## fuel_typeOther             152592.79   93742.43   1.628  0.104
## garage_exists              -10878.39   11387.17  -0.955  0.340
## kitchen_typecombo          -17812.02   88780.62  -0.201  0.841
## kitchen_typeCombo          7639.22   88113.96   0.087    0.931
## kitchen_typeeat in         -4564.93   87407.56  -0.052   0.958
## kitchen_typeEat in         90819.59  107978.39   0.841   0.401
## kitchen_typeEat In         -10116.61   90171.28  -0.112  0.911
## kitchen_typeefficiency     -31073.96   87501.96  -0.355  0.723
## maintenance_cost                 115.29      19.57   5.890 7.57e-09 ***
## num_bedrooms               63228.71    9539.15   6.628 9.78e-11 ***
## num_floors_in_building     6329.77     858.65   7.372 8.20e-13 ***
## num_full_bathrooms         72087.05   14125.38   5.103 4.94e-07 ***
## num_total_rooms                  -89.80      6356.25  -0.014   0.989
## sq_footage                       12.19       16.43   0.742     0.458
## walk_score                 1480.61     354.50   4.177 3.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 85410 on 448 degrees of freedom
## Multiple R-squared:  0.7885, Adjusted R-squared:  0.7762
```

```
## F-statistic: 64.23 on 26 and 448 DF,  p-value: < 2.2e-16

#Random Forest Modeling

y = housing_tbl_imp$sale_price

X = housing_tbl_imp

X$sell_price = NULL


num_trees = 500

mod_rf = YARF(X, y, num_trees = num_trees)

## YARF initializing with a fixed 500 trees...

## YARF factors created...

## YARF after data preprocessed... 31 total features...

## Beginning YARF regression model construction...done.

## Calculating OOB error...done.

mod_rf

## YARF v1.0 for regression

## Missing data feature ON.

## 500 trees, training data n = 528 and p = 31

## Model construction completed within 0.18 minutes.

## OOB results on all observations:

##   R^2: 0.97409

##   RMSE: 28867.76

##   MAE: 15016.79

##   L2: 440007627567

##   L1: 7928864

illustrate_trees(mod_rf, max_depth = 4, open_file = TRUE)
```

```r
#In sample Random Forest

holdout_rf = YARF(housing_tbl_imp_train, housing_tbl_imp_train$sale_price,
num_trees = num_trees)

## YARF initializing with a fixed 500 trees...

## YARF factors created...

## YARF after data preprocessed... 31 total features...

## Beginning YARF regression model construction...done.

## Calculating OOB error...done.

mod_rf

## YARF v1.0 for regression

## Missing data feature ON.

## 500 trees, training data n = 528 and p = 31

## Model construction completed within 0.18 minutes.

## OOB results on all observations:

##   R^2: 0.97409

##   RMSE: 28867.76

##   MAE: 15016.79

##   L2: 440007627567

##   L1: 7928864

#Out of sample RSME for the Random Forest

rmse_rf = sd(y_test - predict(holdout_rf, housing_tbl_imp_test))

rmse_rf

## [1] 22477.11

r_squared = 1 - (sum((y_test - predict(holdout_rf, housing_tbl_imp_test))^2)/
sum((y_test - mean(y))^2))
```

```
r_squared

## [1] 0.9825974
```