Predicting Housing Selling Price in Queens

Final Project for Math390 Data Science at Queens College

Date: 25,May 25, 2019

By: Juan D Astudillo

In collaboration with:

Adriana Sham

Sakib Salim

Vincen Miceli

Burhan Hanif

**Abstract**

The purpose of this project is to get a predicting model to explain what will happen in the future with apartment selling prices in Queens New York. Data of Queens Apartments for sale from 2016 to 2017 collected from the portal MLSLI. We classify the more important variables based on the most common algorithms, linear regression modeling, regression tree modeling and random forest modeling. We found that random forest does the best of the job from all the algorithms, yet to get even better performance of it we need a bigger representation of the population.

1. **Introduction**

Prediction Models are useful for explaining people’s ideas about what they believe reality will be in future events. In this project we use a predicting model to explain what will happen in the future with apartment selling prices in Queens New York. To do so, we used raw data of Queens Apartments for sale from 2016 to 2017 collected from the portal MLSLI. We used three of the most currently used algorithms for the prediction of Price. Using the software Rstudio we coded the three chosen algorithms, Linear Modeling, Regression Tree Modeling and Random Forests Modeling. Our raw Data started with 55 variables which were cleaned to 25 for better performance. Finally, after featurization we concluded with 35 variables for better performance of the algorithms.

Linear modeling helped us have a better perception of the influence that each independent variable has on our dependent variable, Sale Price. Regression Tree Modeling guides us to find more significant ways to split our dataset, it gave us as a result a tree with decision nodes and leaf nodes. Finally, Random Forests returned a model based on the creation of multiple decision trees that it merged together to get a more accurate and stable prediction.

1. **The Data**

Data provided in class comes from a collection of listing sale prices out of the portal Multiple Listing Services. The raw data contains 2230 observation all unique and independent. It includes 55 zip codes from Queens excluding Rockaways, a peninsula near JFK airport that is geographically distinct from the rest of the neighborhoods in Queens, NY. Our population of interest is all the apartments listed for sale in Queens county. Therefore, we can say that this dataset is a partial representation of the entire population. Some of the features provided in this data set had to be cleaned since they were not relevant to our study, we did not use external sources for more features, yet we created more variables based on the features that were already provided.

The data set had some outliers that were fixed with the help of featurization. In addition, it had observations with entry errors as misspelled words or missing information. We must make sure we use the boundaries mentioned above once predicting, since the danger for extrapolation is present. The prediction of the data can be done only for the zip codes in the data set, anything outside that range could be considered extrapolation. Therefore, understanding the data set is very important to deliver good predictions.1

**2.2. Featurization**

The selected out of 55 variables the more relevant 24. We started by excluding some features that had no relevant influence on the Sale price, for instance: URL address, id code and work time approval etc. After removing them we were left with features that had useful measures for our prediction. The more relevant features used were self-explanatory here we provide some description for then and also their statistics:

coop\_condo: class of the variable factor, describes if the observation is coop or a condominium.

co-op: 1661 and condo:569

1Important note: We facilitated newly updated data set, same as the original but with the inclusion of the latitude and  
 longitude variables. See featurization

num\_bedrooms: interger, number of bedroms in the coop or condo, the max number of bedrooms we had in the set was 6 and the average was 2.

approx\_year\_built: interger; the max for this variable is 2017 the newest building and the min are 1893 the oldest building.

community\_district\_num: interger; determines school districts, quite important when looking to buy a house since it may determine where kids go to school. Includes district 3 to 32.

num\_floors\_in\_building: variable interger; number of floors in the coop-condo.

num\_full\_bathrooms: variable interger; the number of full bathrooms in the coop-condo.

Some of the features here helped us to create new variables as for instance:

pets\_allowed: cats\_allowed and dog\_allowed were combined into a single variable since they are collinear.

montly charges: maintenance\_cost and common\_charges were combined into one column since they are mutually exclusive.

longitude: We used the package ggmap to get the coordinates from the given feature full\_address\_or\_zip\_code, this facilitated us to have a better approach on distance and location. (Latitude and longitude were added to the excel sheet and updated. We added that to the original data set, so we don’t run continuously the ggmap code to get them)

Latitude: We used the package ggmap to get the coordinates from the given feature full\_address\_or\_zip\_code, this facilitated us to have a better approach on distance and location. (Latitude and longitude were added to the excel sheet and updated. We added that to the original data set, so we don’t run continuously the ggmap code to get them)

Price per square fit: created out of the variables listing\_price\_to\_nearest\_1000 divided by sq\_footage

Distance to the closest LIRR: We used ggmap to get the latitude and longitude of each LIRR station in Queens, then we ran the code to find the nearest distance of each coop/condo to the closest LIRR station.

The featurization of the variables on the data leaves a lot to the imagination here we just create some extra variables, based on the original ones. Featurization here was done with the purpose of improving the results of our predicting model.

**2.3 Errors and Missingness**

There are many errors done when collecting and entering data, therefore we had to search carefully for errors committed on it. We encounter errors as a misspelling in garage exists, Rstudio function summary and table helped us identify these errors. As well, we encounter errors in a total\_taxes variable, some of the entries in taxes were fake such as entries under 1000 dollars. Therefore, we decided to turn all those observations under 1000 dollars into NAs, so ahead we predicted a better approximation of those observation using missforest. We imputed some of the missing information for the variable community district based on the zipcode, since community district covers bigger areas and zip code was found within it. We corrected minor information on the variables full\_address\_zipcode based on the address of the building and the city where it is located.

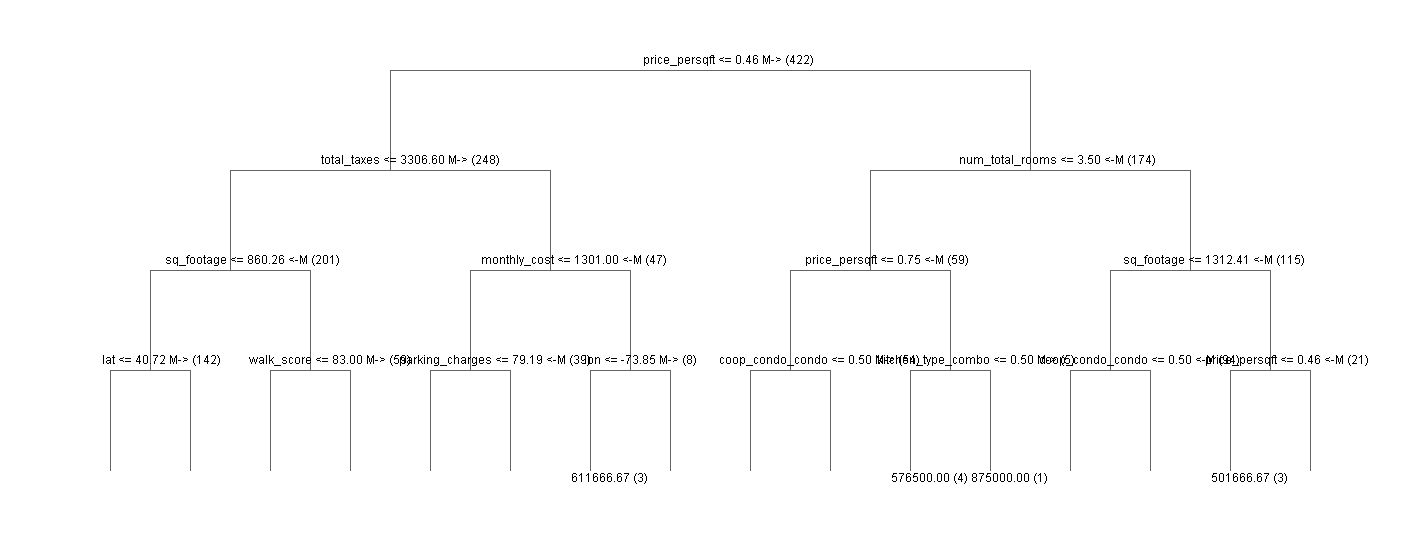
In this project missforest was one of the biggest allied. It helped us to impute missing information on the observation. To achieve better result, we separated Sale Price the variable of interest from the data X. If we keep Sale Price when running missforest on X all our X’s could be corrupted, since they would learn perfectly from Y. Missforest learns from all the data X and inputs the NAs base on the information collected from it. Missforest will return our X without NAs, so we can have more information to create the predicting final algorithm. When using Missforest it is recommended to create a column id, because missforest shuffles all the rows, hence creating a column id for Y and for X will facilitate returning the data when we need. Finally, we discarded all the observation that had Sale Price as NA’s since it wont be honest to impute them. Imputing Y’s would have created problems on the model once we test it out of sample. Our final data set contains 523 observation with no NA’s or missing information.

**3. Modeling**

For the creation of the model we need completed data, the bigger the number of observations that we have in D the best it will be and more accurate results we could get. Since we are creating a model to ship to de world, we want the most optimal performance out of it. We must utilize efficiently our D to achieve good predictions. Our final Data set has 523 observations, this dataset was utilized to run all the algorithms ahead.

**3.1 Regression Tree**

Regression trees divide a data set into smaller subgroups. These different partitions help us take better decisions once picking the more important variables for predicting price in our model. In our regression tree the most important split was the price per squared fit, the reason behind this is because having more space represents more comfort. If we have a bigger space in the apartment this will be directly reflected in the price of it. Similarly happened with the variable num\_total\_rooms the number of rooms increments the Price. Following the tree a relevant variable is monthly cost and finally coop-condo.



**3.2 Linear Modeling**

Our linear regression model has 25 independent variables, some of the more important inputs in the regression are community district, coop\_condo, number of bathrooms, latitude, longitude, total taxes, kitchen type, number of bedrooms and monthly cost, all of them with very significant p values (three, two and one star). The R^2 of the model in sample 84% is almost the same than out of sample 85%, in the sample is higher because it is prompt for overfitting. Out of sample RMSE is +71006.00, its R^2 is the same as in sample since it is working with the testing set and it’s indicative of how our predictions would look.

For the interpretation of the coefficients we need to set everything constant but the two coefficients of interest, then we state that when we compare two naturally observed observations PRICE and coop condo we can see that price will go up by 9710.00 dollars if it is a coop. Yet, if we take into consideration latitude, PRICE will increase by 677200.00 dollars. Linear models are good alternatives for prediction yet in our case it won’t be the best option for predicting PRICE. A linear model will be good, but not be ideal for predicting sale price. In the linear model we are assuming linear interactions when in reality we don’t really know if our assumption is valid or not.

**3.3 Random Forest Modeling**

Random Forests performs well with big amounts of data. It creates many decision trees therefore if we have more decision trees in the forest the better will be our predictions. Random forest is created in the same base as decision trees and bagging. Bagging trees include randomness in the process of building trees, it will reduce the variance of each prediction of the single trees. Yet, the trees in bagging have not complete independence of one to another since all the original predictions are connected to the original first splits. These characteristics are known as tree correlation which prevents bagging from being 1optimal, therefore it reduces the variance of the predicted values, consequently it improves the performance of the model overall. However, since we need more data to finally be able to see the real power of random forest we can say that the variables that were initially mentioned in the other model are the more effective to create our prediction model. It is logical to the think that the price will increase based on price per footage, the causal relationship between these two variables have a big impact on price.

**4. Performance Result for Random Forest Model**

The overall results for our different algorithms are for linear regression RMSE 83540.97, Regression Tree modeling 86456.92, and for our for Our RMSE for random forest 86456. representative.

**5. Discussion**

Some of the variables perform better when using the linear regression algorithm. It would be good to increase N the sample size to have a better prediction on random forest and tree modeling. The results obtained by the linear regression so far gave us a good idea of how the algorithms are performing and it was confirmed by tree modeling and random forest. The overall results for our different algorithms are for linear regression RMSE 83540.97, Regression Tree modeling 86456.92, and for our for Our RMSE for random forest 86456.

## R Markdown

pacman::p\_load(dplyr, tidyr, ggplot2, magrittr, stringr, mlr)  
housing\_data = read.csv("housing\_data\_2016\_2017.csv")

##Delete variables that we dont need

housing\_data %<>%  
 select(-c(HITId, HITTypeId, Title, Description, Keywords, Reward, CreationTime, MaxAssignments, RequesterAnnotation, AssignmentDurationInSeconds, AutoApprovalDelayInSeconds, Expiration, NumberOfSimilarHITs, LifetimeInSeconds, AssignmentId, WorkerId, AssignmentStatus, AcceptTime, SubmitTime, AutoApprovalTime, ApprovalTime, RejectionTime, RequesterFeedback, WorkTimeInSeconds, LifetimeApprovalRate, Last30DaysApprovalRate, Last7DaysApprovalRate, URL, url, date\_of\_sale))

## Clean Data

#extract zip codes as a separate feature  
housing\_data %<>%  
 mutate( zip\_code = str\_extract(full\_address\_or\_zip\_code, "[0-9]{5}"))   
  
#make pets allowed binary and combine them  
housing\_data %<>%  
 mutate(dogs\_allowed = ifelse(substr(housing\_data$dogs\_allowed, 1, 3) == "yes", 1, 0)) %>%  
 mutate(cats\_allowed = ifelse(substr(housing\_data$cats\_allowed, 1, 3) == "yes", 1, 0)) %>%  
 mutate( pets\_allowed = ifelse( cats\_allowed + dogs\_allowed > 0, 1, 0)) %>%  
 mutate(coop\_condo = factor(tolower(coop\_condo)))  
  
housing\_data %<>%  
 select(-c(dogs\_allowed,cats\_allowed, fuel\_type))  
  
d = housing\_data  
  
#convert NA's to 0 for charges  
d %<>%  
 mutate(maintenance\_cost = sjmisc::rec(maintenance\_cost, rec = "NA = 0 ; else = copy")) %<>%  
 mutate(common\_charges = sjmisc::rec(common\_charges, rec = "NA = 0 ; else = copy"))##recode from NA to 0.  
  
# combine maintaince cost and common charges  
d %<>%   
 mutate( monthly\_cost = common\_charges + maintenance\_cost)  
  
d %<>%  
 mutate(monthly\_cost = sjmisc::rec(monthly\_cost, rec = "0 = NA ; else = copy"))  
  
## Convert garage to binary  
d %<>%  
 mutate(garage\_exists = sjmisc::rec(garage\_exists, rec = "NA = 0 ; else = copy")) ##recode from NA to 0.   
  
d %<>%  
 mutate(garage\_exists = sjmisc::rec(garage\_exists, rec = " eys = 1; UG = 1 ; Underground = 1; yes = 1 ; Yes = 1 ; else = copy")) ##recode from NA to 0.  
  
d %<>%  
 select(-c(maintenance\_cost , common\_charges, model\_type))  
  
#str(d)

##Change variable type

d %<>%  
 mutate( dining\_room\_type = as.factor(dining\_room\_type)) %>%  
 mutate(garage\_exists = as.character(garage\_exists)) %>%  
 mutate(garage\_exists = as.numeric(garage\_exists)) %>%  
 mutate( parking\_charges = as.character(parking\_charges)) %>%  
 mutate( parking\_charges = as.numeric(parking\_charges)) %>%  
 #sale to numeric for regression  
 mutate(sale\_price = as.character(sale\_price)) %>%  
 mutate(sale\_price = as.numeric(sale\_price)) %>%  
 mutate(total\_taxes = as.character(total\_taxes)) %>%  
 mutate(total\_taxes = as.numeric(total\_taxes)) %>%  
   
 #new feature 'price/sq ft'  
 mutate(price\_persqft = listing\_price\_to\_nearest\_1000 / sq\_footage)

## Warning: NAs introduced by coercion  
  
## Warning: NAs introduced by coercion  
  
## Warning: NAs introduced by coercion

#Added latitude and longitude features using ggmap

#Already run and included in the data  
#pacman::p\_load(ggmap)  
#d %<>%  
# mutate(lat = geocode(full\_address\_or\_zip\_code)$lat, lon = #geocode(full\_address\_or\_zip\_code)$lon )  
  
#geocoordinates for relevant LIRR stations  
lirr\_coord = read.csv("all\_lirr\_geocoordinates.csv")  
  
  
RAD\_EARTH = 3958.8  
degrees\_to\_radians = function(angle\_degrees){  
 for(i in 1:length(angle\_degrees))  
 angle\_degrees[i] = angle\_degrees[i]\*pi/180  
 return(angle\_degrees)  
}  
  
compute\_globe\_distance = function(destination, origin){  
 destination\_rad = degrees\_to\_radians(destination)  
 origin\_rad = degrees\_to\_radians(origin)  
 delta\_lat = destination\_rad[1] - origin\_rad[1]  
 delta\_lon = destination\_rad[2] - origin\_rad[2]  
 h = (sin(delta\_lat/2))^2 + cos(origin\_rad[1]) \* cos(destination\_rad[1]) \* (sin(delta\_lon/2))^2  
 central\_angle = 2 \* asin(sqrt(h))  
 return(RAD\_EARTH \* central\_angle)  
}  
  
#find the closest LIRR station and compute distance  
shortest\_lirr\_distance = function(all\_lirr\_coords, house\_coords){  
 shortest\_dist = Inf  
 for (i in 1: nrow(all\_lirr\_coords)){  
 ith\_lirr = c(all\_lirr\_coords$lat[i], all\_lirr\_coords$lon[i])  
 new\_dist = compute\_globe\_distance(ith\_lirr, house\_coords)  
 if( new\_dist < shortest\_dist){  
 shortest\_dist = new\_dist  
 }  
 }  
 return(shortest\_dist)  
}  
  
d %<>%  
 rowwise() %>%  
 mutate(shortest\_dist = shortest\_lirr\_distance(lirr\_coord, c(lat, lon)) )  
  
#makes any other addresses redundant  
d %<>%  
 select(-c(zip\_code, full\_address\_or\_zip\_code, listing\_price\_to\_nearest\_1000))

We are trying to predict sale\_price. So let’s section our dataset:

####CREATE A COLUMN ID  
  
d %<>%  
 ungroup(d) %>%  
 mutate(id = 1 : 2230)  
  
real\_y = data.frame(d$id, d$sale\_price)  
  
j = d %>%  
 select(total\_taxes)  
  
d %<>%  
 select(-c(total\_taxes, sale\_price))  
  
d = cbind(j, d)  
  
d[,1][d[, 1] < 1000] = NA ## number 1 is total taxes  
  
real\_d = subset(d, (!is.na(d[,2]))) ## sale price  
fake\_d = subset(d, (is.na(d[,2])))

#Split the data that has y into train and test sets

train\_indices = sample(1 : nrow(real\_d), nrow(real\_d)\*4/5)  
training\_data = real\_d[train\_indices, ]  
testing\_data = real\_d[-train\_indices, ]  
  
#testing\_data %<>%  
# mutate(sale\_price = NA)  
  
  
X = rbind(training\_data, testing\_data, fake\_d)  
  
  
#table(X$total\_taxes)  
  
#str(X)

Let’s first create a matrix with columns that represents missingness

M = tbl\_df(apply(is.na(X), 2, as.numeric))  
colnames(M) = paste("is\_missing\_", colnames(X), sep = "")  
# head(M)  
# summary(M)

Some of these missing indicators are collinear because they share all the rows they are missing on. Let’s filter those out:

M = tbl\_df(t(unique(t(M))))

Some featuers did not have missingness so let’s remove them:

M %<>% select\_if(function(x){sum(x) > 0})  
# head(M)  
# dim(M)  
# colSums(M)

Now let’s impute using the package. we cannot fit RF models to the entire dataset (it’s 26,000! observations) so we will sample 5 for X1 and for each of the trees and then average. That will be good enough.

pacman::p\_load(missForest)  
Ximp = missForest(data.frame(X), sampsize = rep(172, ncol(X)))$ximp

## missForest iteration 1 in progress...

## Warning in randomForest.default(x = obsX, y = obsY, ntree = ntree, mtry =  
## mtry, : The response has five or fewer unique values. Are you sure you want  
## to do regression?

## done!  
## missForest iteration 2 in progress...

## Warning in randomForest.default(x = obsX, y = obsY, ntree = ntree, mtry =  
## mtry, : The response has five or fewer unique values. Are you sure you want  
## to do regression?

## done!  
## missForest iteration 3 in progress...

## Warning in randomForest.default(x = obsX, y = obsY, ntree = ntree, mtry =  
## mtry, : The response has five or fewer unique values. Are you sure you want  
## to do regression?

## done!  
## missForest iteration 4 in progress...

## Warning in randomForest.default(x = obsX, y = obsY, ntree = ntree, mtry =  
## mtry, : The response has five or fewer unique values. Are you sure you want  
## to do regression?

## done!

Ximp %<>%  
 arrange(id)  
  
  
Xnew = data.frame(cbind(Ximp, M, real\_y))  
  
Xnew %<>%  
 mutate(price = d.sale\_price) %>%  
 select(-c(id, d.id, d.sale\_price))  
   
  
linear\_mod\_impute\_and\_missing\_dummies = lm(price ~ ., data = Xnew)  
summary(linear\_mod\_impute\_and\_missing\_dummies)

##   
## Call:  
## lm(formula = price ~ ., data = Xnew)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -279749 -35955 348 38112 344912   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -3.895e+07 9.524e+06 -4.089 5.06e-05  
## total\_taxes 6.256e+00 5.664e+00 1.104 0.269919  
## approx\_year\_built 1.987e+02 2.469e+02 0.805 0.421356  
## community\_district\_num 3.332e+03 1.172e+03 2.844 0.004641  
## coop\_condocondo 1.476e+05 1.833e+04 8.050 6.33e-15  
## dining\_room\_typedining area 3.404e+04 5.321e+04 0.640 0.522661  
## dining\_room\_typeformal 2.898e+04 8.567e+03 3.383 0.000776  
## dining\_room\_typeother 1.409e+04 1.143e+04 1.233 0.218120  
## garage\_exists 1.433e+04 9.112e+03 1.572 0.116481  
## kitchen\_typeeat in 6.103e+02 1.021e+04 0.060 0.952383  
## kitchen\_typeefficiency -1.358e+04 1.008e+04 -1.348 0.178359  
## num\_bedrooms 4.981e+04 7.932e+03 6.280 7.48e-10  
## num\_floors\_in\_building 3.001e+03 7.251e+02 4.139 4.11e-05  
## num\_full\_bathrooms -1.287e+04 5.311e+04 -0.242 0.808632  
## num\_half\_bathrooms -4.615e+04 3.198e+04 -1.443 0.149665  
## num\_total\_rooms 1.653e+04 5.188e+03 3.186 0.001536  
## parking\_charges 3.254e+02 9.632e+01 3.378 0.000789  
## pct\_tax\_deductibl 6.536e+02 8.746e+02 0.747 0.455261  
## sq\_footage 2.472e+01 1.284e+01 1.925 0.054792  
## walk\_score -7.634e+02 3.938e+02 -1.939 0.053130  
## lat 6.038e+05 1.413e+05 4.273 2.32e-05  
## lon -1.854e+05 8.662e+04 -2.141 0.032772  
## pets\_allowed 1.190e+04 6.989e+03 1.703 0.089207  
## monthly\_cost 1.294e+02 1.409e+01 9.182 < 2e-16  
## price\_persqft 5.023e+05 6.882e+04 7.298 1.18e-12  
## shortest\_dist -4.051e+03 6.237e+03 -0.650 0.516308  
## is\_missing\_total\_taxes -9.201e+02 9.349e+03 -0.098 0.921646  
## is\_missing\_approx\_year\_built NA NA NA NA  
## is\_missing\_community\_district\_num 7.303e+03 3.081e+04 0.237 0.812696  
## is\_missing\_dining\_room\_type -2.105e+04 7.956e+03 -2.646 0.008402  
## is\_missing\_kitchen\_type 2.073e+04 5.280e+04 0.393 0.694739  
## is\_missing\_num\_bedrooms 1.610e+04 1.606e+04 1.003 0.316360  
## is\_missing\_num\_floors\_in\_building -1.162e+04 7.587e+03 -1.532 0.126272  
## is\_missing\_num\_half\_bathrooms -1.524e+04 1.376e+04 -1.107 0.268684  
## is\_missing\_num\_total\_rooms NA NA NA NA  
## is\_missing\_parking\_charges 1.270e+04 7.978e+03 1.592 0.111987  
## is\_missing\_pct\_tax\_deductibl -6.021e+03 8.465e+03 -0.711 0.477237  
## is\_missing\_sq\_footage 1.427e+04 1.098e+04 1.299 0.194401  
## is\_missing\_monthly\_cost 8.883e+03 1.622e+04 0.548 0.584128  
## is\_missing\_price\_persqft -8.711e+03 1.139e+04 -0.765 0.444755  
##   
## (Intercept) \*\*\*  
## total\_taxes   
## approx\_year\_built   
## community\_district\_num \*\*   
## coop\_condocondo \*\*\*  
## dining\_room\_typedining area   
## dining\_room\_typeformal \*\*\*  
## dining\_room\_typeother   
## garage\_exists   
## kitchen\_typeeat in   
## kitchen\_typeefficiency   
## num\_bedrooms \*\*\*  
## num\_floors\_in\_building \*\*\*  
## num\_full\_bathrooms   
## num\_half\_bathrooms   
## num\_total\_rooms \*\*   
## parking\_charges \*\*\*  
## pct\_tax\_deductibl   
## sq\_footage .   
## walk\_score .   
## lat \*\*\*  
## lon \*   
## pets\_allowed .   
## monthly\_cost \*\*\*  
## price\_persqft \*\*\*  
## shortest\_dist   
## is\_missing\_total\_taxes   
## is\_missing\_approx\_year\_built   
## is\_missing\_community\_district\_num   
## is\_missing\_dining\_room\_type \*\*   
## is\_missing\_kitchen\_type   
## is\_missing\_num\_bedrooms   
## is\_missing\_num\_floors\_in\_building   
## is\_missing\_num\_half\_bathrooms   
## is\_missing\_num\_total\_rooms   
## is\_missing\_parking\_charges   
## is\_missing\_pct\_tax\_deductibl   
## is\_missing\_sq\_footage   
## is\_missing\_monthly\_cost   
## is\_missing\_price\_persqft   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 72740 on 490 degrees of freedom  
## (1702 observations deleted due to missingness)  
## Multiple R-squared: 0.8473, Adjusted R-squared: 0.8358   
## F-statistic: 73.51 on 37 and 490 DF, p-value: < 2.2e-16

### REMOVING MISSING Y SECTION

Data = Xnew  
### sale price is our imputed Y  
  
  
Data %<>%  
 filter(!is.na(price))  
  
  
Y = Data$price  
  
Xtrain = Data[1:422, ]  
Xtest = Data[423:528, ]  
  
Ytrain = Y[1:422]  
Ytest = Y[423:528]  
  
dtrain = cbind(Xtrain, Ytrain) ## combine x train with y train, x test with y test  
dtest = cbind(Xtest, Ytest)

Linear Regression

Xtrain$price = NULL  
Xtest$price = NULL  
linear = lm(Ytrain ~ ., data = Xtrain)## simple linear model  
summary(linear)

##   
## Call:  
## lm(formula = Ytrain ~ ., data = Xtrain)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -297241 -35024 1334 34032 344865   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -4.133e+07 1.077e+07 -3.837 0.000146  
## total\_taxes 3.381e+00 6.032e+00 0.560 0.575480  
## approx\_year\_built 1.914e+02 2.710e+02 0.706 0.480496  
## community\_district\_num 3.149e+03 1.249e+03 2.520 0.012124  
## coop\_condocondo 1.733e+05 2.078e+04 8.341 1.34e-15  
## dining\_room\_typedining area 3.919e+04 5.248e+04 0.747 0.455749  
## dining\_room\_typeformal 3.001e+04 9.333e+03 3.215 0.001413  
## dining\_room\_typeother 1.446e+04 1.263e+04 1.145 0.252940  
## garage\_exists 1.595e+04 1.035e+04 1.541 0.124057  
## kitchen\_typeeat in 6.006e+03 1.136e+04 0.529 0.597344  
## kitchen\_typeefficiency -1.676e+04 1.129e+04 -1.484 0.138653  
## num\_bedrooms 3.818e+04 8.834e+03 4.322 1.97e-05  
## num\_floors\_in\_building 2.891e+03 8.112e+02 3.563 0.000413  
## num\_full\_bathrooms -5.906e+03 5.233e+04 -0.113 0.910201  
## num\_half\_bathrooms -4.943e+04 3.356e+04 -1.473 0.141591  
## num\_total\_rooms 2.127e+04 5.755e+03 3.697 0.000250  
## parking\_charges 3.535e+02 1.008e+02 3.507 0.000506  
## pct\_tax\_deductibl 2.170e+03 1.185e+03 1.832 0.067699  
## sq\_footage 2.118e+01 1.325e+01 1.598 0.110832  
## walk\_score -6.061e+02 4.329e+02 -1.400 0.162249  
## lat 6.184e+05 1.554e+05 3.980 8.24e-05  
## lon -2.087e+05 9.827e+04 -2.124 0.034315  
## pets\_allowed 6.496e+03 7.664e+03 0.848 0.397187  
## monthly\_cost 1.524e+02 1.869e+01 8.153 5.07e-15  
## price\_persqft 4.309e+05 7.539e+04 5.716 2.20e-08  
## shortest\_dist 1.023e+02 6.916e+03 0.015 0.988210  
## is\_missing\_total\_taxes 2.622e+03 9.990e+03 0.262 0.793128  
## is\_missing\_approx\_year\_built NA NA NA NA  
## is\_missing\_community\_district\_num 9.286e+03 3.048e+04 0.305 0.760783  
## is\_missing\_dining\_room\_type -2.072e+04 8.721e+03 -2.376 0.017976  
## is\_missing\_kitchen\_type 3.806e+04 5.197e+04 0.732 0.464377  
## is\_missing\_num\_bedrooms 8.820e+03 1.965e+04 0.449 0.653758  
## is\_missing\_num\_floors\_in\_building -9.331e+03 8.395e+03 -1.111 0.267051  
## is\_missing\_num\_half\_bathrooms -1.047e+04 1.582e+04 -0.662 0.508459  
## is\_missing\_num\_total\_rooms NA NA NA NA  
## is\_missing\_parking\_charges 2.108e+04 8.989e+03 2.345 0.019512  
## is\_missing\_pct\_tax\_deductibl 3.802e+02 9.186e+03 0.041 0.967008  
## is\_missing\_sq\_footage 2.236e+04 1.190e+04 1.878 0.061075  
## is\_missing\_monthly\_cost 1.069e+04 1.785e+04 0.599 0.549792  
## is\_missing\_price\_persqft -1.557e+04 1.233e+04 -1.263 0.207513  
##   
## (Intercept) \*\*\*  
## total\_taxes   
## approx\_year\_built   
## community\_district\_num \*   
## coop\_condocondo \*\*\*  
## dining\_room\_typedining area   
## dining\_room\_typeformal \*\*   
## dining\_room\_typeother   
## garage\_exists   
## kitchen\_typeeat in   
## kitchen\_typeefficiency   
## num\_bedrooms \*\*\*  
## num\_floors\_in\_building \*\*\*  
## num\_full\_bathrooms   
## num\_half\_bathrooms   
## num\_total\_rooms \*\*\*  
## parking\_charges \*\*\*  
## pct\_tax\_deductibl .   
## sq\_footage   
## walk\_score   
## lat \*\*\*  
## lon \*   
## pets\_allowed   
## monthly\_cost \*\*\*  
## price\_persqft \*\*\*  
## shortest\_dist   
## is\_missing\_total\_taxes   
## is\_missing\_approx\_year\_built   
## is\_missing\_community\_district\_num   
## is\_missing\_dining\_room\_type \*   
## is\_missing\_kitchen\_type   
## is\_missing\_num\_bedrooms   
## is\_missing\_num\_floors\_in\_building   
## is\_missing\_num\_half\_bathrooms   
## is\_missing\_num\_total\_rooms   
## is\_missing\_parking\_charges \*   
## is\_missing\_pct\_tax\_deductibl   
## is\_missing\_sq\_footage .   
## is\_missing\_monthly\_cost   
## is\_missing\_price\_persqft   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 71060 on 384 degrees of freedom  
## Multiple R-squared: 0.8528, Adjusted R-squared: 0.8386   
## F-statistic: 60.12 on 37 and 384 DF, p-value: < 2.2e-16

yhat = predict(linear, Xtest)

## Warning in predict.lm(linear, Xtest): prediction from a rank-deficient fit  
## may be misleading

e = yhat - Ytest  
  
#RMSE  
sqrt(sum(e^2) / 108)

## [1] 83540.97

#REGRESSION TREE  
pacman::p\_load(YARF)

## YARF can now make use of 3 cores.

reg\_tree = YARFCART(Xtrain, Ytrain)

## YARF initializing with a fixed 1 trees...  
## YARF factors created...  
## YARF after data preprocessed... 42 total features...  
## Beginning YARF regression model construction...done.  
## Calculating OOB error...done.

y\_hat\_test\_tree = predict(reg\_tree, Xtest)  
e = Ytest - y\_hat\_test\_tree  
#RMSE  
sqrt(sum(e^2)/108)

## [1] 126998.6

sd(e)

## [1] 127303.4

illustrate\_trees(reg\_tree, max\_depth = 4, open\_file = TRUE)

Make test, train and selection sets

n = nrow(Data)  
K = 5  
test\_indices = sample(1 : n, size = n \* 1 / K)  
master\_train\_indices = setdiff(1 : n, test\_indices)  
select\_indices = sample(master\_train\_indices, size = n \* 1 / K)  
train\_indices = setdiff(master\_train\_indices, select\_indices)  
rm(master\_train\_indices)  
  
houses\_train = Data[train\_indices, ]  
houses\_select = Data[select\_indices, ]  
houses\_test = Data[test\_indices, ]

Hyperparameter Tuning for Random Forest Running this chunk gives the optimal hyperparameters used in the model.

# train\_task = makeRegrTask(data = houses\_train, target = "price")  
# test\_task = makeRegrTask(data = houses\_test, target = "price")  
#   
# algorithm = makeLearner("regr.randomForest", predict.type = "response")  
#   
# all\_mtry = seq(1, 10, by = 1)  
# all\_nodesize = seq(1, 10, by = 1)  
# all\_sampsize = seq(100, 110, by = 1)  
# all\_hyperparams = makeParamSet(  
# makeDiscreteParam(id = "nodesize", default = 5, values = all\_nodesize),  
# makeDiscreteParam(id = "mtry", default = 5, values = all\_mtry)  
# )  
# inner = makeResampleDesc("CV", iters = 3)  
# lrn = makeTuneWrapper("regr.randomForest",   
# n resampling = inner,   
# par.set = all\_hyperparams,   
# control = makeTuneControlGrid())  
#   
#   
# outer = makeResampleDesc("CV", iters = 5)  
# r = resample(lrn, train\_task,   
# resampling = outer,   
# extract = getTuneResult)  
#   
# r #overall estimate of oos error of the whole procedure if it were used on all of $\mathbb{D}$  
# print(getNestedTuneResultsOptPathDf(r)) #results of each inner validation over all outer iterations  
# r$extract #"winning" model for each outer iteration

rf\_mod = YARF(Xtrain, Ytrain, mtry = 10, nodesize = 4)

## YARF initializing with a fixed 500 trees...  
## YARF factors created...  
## YARF after data preprocessed... 42 total features...  
## Beginning YARF regression model construction...done.  
## Calculating OOB error...done.

y\_hat\_test\_tree = predict(rf\_mod, Xtest)  
e = Ytest - y\_hat\_test\_tree  
#RMSE  
sqrt(sum(e^2)/108)

## [1] 86456.37

sd(e)

## [1] 85004.18