Modeling Co-op and Condo Sale Prices in Queens

Final project for Math 342W Data Science at Queens

May 25, 2021

By Marin Azhar

In collaboration with:

Kennly Weerasinghe

Sara Jedwab

Hubert Majewski

Enoch Kim

Abstract:

Using raw housing data for Co-ops and Condos in Queens between 2016 and 2017, this project aims to see if we can create more accurate predictive model to compete against Zillow's Zestimate. The three types of models utilized for this project are Regression Trees, OLS, and Random Forest. For each model, we will be discussing the features and key performance metrics. Out of the three models, only the Random Forest had a high enough out of sample R^2 that could be used for predicting real world sales prices on Co-ops and Condos.

Introduction:

Since 2006 websites such as Zillow.com have created predictive models to estimate a home's selling value. While Zillow's Zestimate are popular for people who want to understand the ballpark of the value of their homes, their model is, of course, not perfect. However, a model does not need to be perfect in order to be useful. It only needs to be accurate enough and perform well enough out of sample to be used in the real world. Given nine different areas in Queens, for apartments less than one million dollars, we will be using data collected from Amazon's MTurk, to construct a mathematical model to predict our response variable of sale prices in the units of dollars. Uses that are often unsatisfied with Zillow's Zestimate are confused by what drives a home's price. Many people use outdated information, which can further give them an incorrect approximation for home value. Before using a model, selecting features that have the most correlation to our phenomena is essential. Using our three models, Regression Trees, OLS, and Random Forests, we can evaluate the relevance of critical features and how they affect sales prices.

The Data:

The data was harvested with Amazon's MTurk and download raw from their system. Historically the data set has 2330 rows. However, only 528 rows had a sales price that could be used for a train and test split. This means 77% of the data is missing a response variable there for there is a danger when extrapolating data because the data is not fully represented therefore our model will have a high estimation error. Out of total 55 columns, the data types were characters, integers, doubles, and logical. Due to how the data was collected, over 30 columns were dropped as they were not related to Zillow's housing data. The Remaining dataset could then be inspected for featurization. No other source of data was appended during the data wrangling process.

Featurization:

15 features were selected for the final data set 9 are unordered factors and 6 are numeric variables.

Cats_allowed, dogs_allowed and parking_exists are factor with a category of yes and no. Parking_exists is also a mutation of garadge_exists and a combination if parking charges exists.

coop_condo is a factor with category co-op and condo. Dining_room_types is a factor with four categories: combo, formal and other. Kitchen_type is a factor with three categories: efficiency, eat in and combo. Walk_score has been mutated form a numeric to a factor with four categories: Car-Dependent, Somewhat Walkable, Very Walkable, and Walker's Paradise. Fuel_type is a factor with three categories: gas, oil, other, and electric. Neighborhoods is a factor with nine categories: North Queens, West Queens, Southeast Queens, Southwest Queens, Jamaica, West Central Queens, Northwest Queens, Northeast Queens and Central Queens. These categories were a mutation of the zip codes which were extracted from the full address and categorized based on the distinctions given on the project pdf.

The numeric range for num_bed_rooms is from 0-6 bed rooms , the range of num_total_rooms is from 0-14 rooms, the range of num_full_bathrooms is from 1-3 full bathrooms, the range for num_half_bathrooms from 0-2 half bathrooms, the range of aprox_year_built is from the years 1893-2017, the range of sq_footage is from 100 square foot - 6215 square foot of the apartment and total cost is a mutation of monthly maintenance cost for co-ops and monthly common charges for condos plus the total yearly tax divided by 12 as a dollar range of doubles.

Figure 0: Summary of the housing data before imputation:

```
##  approx_year_built cats_allowed common_charges    coop_condo   dining_room
_type
##  Min.   :1893      no :1401     Min.   :   0.00   co-op:1659   combo :957
##  1st Qu.:1950      yes: 826     1st Qu.:   0.00   condo: 568   formal:620
##  Median :1958                   Median :   0.00                other :203
##  Mean   :1963                   Mean   : 147.85                NA's  :447
##  3rd Qu.:1970                   3rd Qu.:  43.42
##  Max.   :2017                   Max.   :2463.00
##  NA's   :40
##  dogs_allowed    fuel_type    parking_exists           kitchen_type
##  no :1681     electric:  62   no :1492       combo             :397
##  yes: 546     gas     :1347   yes: 735       eat in            :943
```

```
##              oil     : 662                 efficiency kitchen:848
##              other   :  44                 NA's              : 39
##              NA's    : 112
##
##
##  maintenance_cost  num_bedrooms   num_full_bathrooms num_half_bathrooms
##  Min.   :   0.0   Min.   :0.000   Min.   :1.000      Min.   :0.00000
##  1st Qu.:   0.0   1st Qu.:1.000   1st Qu.:1.000      1st Qu.:0.00000
##  Median : 672.0   Median :2.000   Median :1.000      Median :0.00000
##  Mean   : 643.8   Mean   :1.653   Mean   :1.231      Mean   :0.07364
##  3rd Qu.: 895.0   3rd Qu.:2.000   3rd Qu.:1.000      3rd Qu.:0.00000
##  Max.   :4659.0   Max.   :6.000   Max.   :3.000      Max.   :2.00000
##  NA's   :61       NA's   :115
##  num_total_rooms    sale_price       sq_footage                walk_score
##  Min.   : 0.000   Min.   : 55000   Min.   : 100.0   Car-Dependent    :  77
##  1st Qu.: 3.000   1st Qu.:171000   1st Qu.: 743.0   Somewhat Walkable: 243
##  Median : 4.000   Median :259000   Median : 881.0   Very Walkable    : 819
##  Mean   : 4.138   Mean   :314492   Mean   : 955.4   Walker's Paradise:1088
##  3rd Qu.: 5.000   3rd Qu.:428250   3rd Qu.:1100.0
##  Max.   :14.000   Max.   :999999   Max.   :6215.0
##  NA's   :2        NA's   :1700     NA's   :1207
##              neighborhood
##  North Queens       :552
##  West Central Queens:457
##  West Queens        :340
##  Southwest Queens   :205
##  Northeast Queens   :179
##  Southeast Queens   :151
##  (Other)            :343
```

Errors, Missingness, and Mutation of Features:

Some obvious errors were misspellings in the raw dataset. For example, "eyes" in garage_exists or the various spellings or format for "efficiency" in kitchen_type. These errors were handled by creating a factor that concatenates different spellings of the same word into a single category. Features such as pct_tax_deductibl and date_of_sale, which had over 77% missingness, these features were dropped. For features that we believed were correlated to sales_prices, such as garage_exists and common_charges and sq_ootage, which also had low completion rates, we mutated the features and extracted as much information from the data set to fill in the missingness as much as possible before imputation.

From the data, we do know if the parking charges were mothing or yearly charges; however, if a charge exists, we can also believe that parking space exists there for this aspect was used to fill in the missingness of the garage_exists feature which was then renamed to parking_exits. Afterward, the parking_charges feature was dropped.

A large percentage of the dataset is for co-ops there, for it makes sense that those co-ops would not have a common charge because co-ops only have a maintenance cost. Therefore, to deal with missingness in common charges, if a building was a co-op, the price would be set to zero. Moreover, some buildings had their charges miscategorized for more accuracy; a swap was made, so only co-ops had maintenance cots charges, and condos had common charges.

While the full address has a completion rate of 1, some of the addresses were incorrect there for when extracting the zip codes. They were incorrect as well. If we imputed on missing zip codes, we would most like to get fake zip codes. As a solution, filtering out incorrect zip codes to manually search for the actual zip codes was done to correct the error to factorize them into neighborhoods.

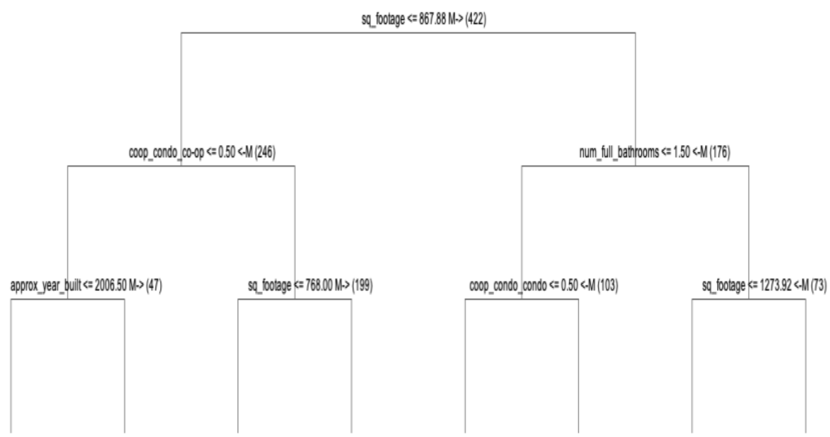
Lastly, other assumptions on the data to reduce missingness were also made. For example, if an apartment reported NA for the number of half-bathrooms, we changed it to a zero. Although it would be wiser to drop features with over 77% missingness, features such as square footage are casual to an apartment sales price there for this is one out of the eight features used to create missingness dummy variables that were appended to the dataset when running missForest for imputation.

Modeling:

Using only apartments that are not missing a response variable. A train and test split was made from the imputed data in order to fit three different kinds of models.

Regression Tree Modeling:

Figure 1: A Regression Tree visualization of depth 3



Th Regression Tree model using YARF has 291 nodes, 146 leaves and a max depth of 23. The in-sample RMSE is 34866.63$ and the R^2 is 80.04%. The out of sample RMSE is 111024.3$ and the R^2 is 3.43% clearly the model under performs.

To comment on the top features, we must look at a regression tree image. We can then see the main split or the root node is sq_footage. This makes sense because there is a high correlation between sales prices and space. The larger the space the more you would expect to pay for. For apartment’s with larger square footage the next globally best node is number full bathrooms. And for apartments with smaller than 867 square footage, it then matters if your apartment is co-op vs condo. This also makes sense as well because co-ops are also cheaper than condos.

Linear Modeling:

Figure 2: OLS output as table

```
lm(formula = housing_ytrain ~ ., data = housing_Xtrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -305391  -36708   -3317   37233  289916
##
## Coefficients: (1 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -1.867e+06  6.267e+05  -2.980 0.003067 **
## cats_allowedyes                   1.154e+04  1.119e+04   1.030 0.303464
## coop_condocondo                   1.866e+05  1.499e+04  12.449  < 2e-16 ***
## dining_room_typeformal            3.056e+04  1.072e+04   2.852 0.004584 **
## dining_room_typeother             2.834e+03  1.395e+04   0.203 0.839070
## dogs_allowedyes                   4.465e+03  1.256e+04   0.355 0.722521
## fuel_typegas                     -2.047e+04  2.641e+04  -0.775 0.438762
## fuel_typeoil                     -9.346e+03  2.740e+04  -0.341 0.733201
## fuel_typeother                   -1.852e+04  3.693e+04  -0.502 0.616236
## parking_existsyes                -6.585e+03  8.916e+03  -0.739 0.460640
## kitchen_typeeat in               -7.460e+03  1.252e+04  -0.596 0.551515
## kitchen_typeefficiency kitchen   -2.277e+04  1.223e+04  -1.863 0.063287 .
## num_bedrooms                      3.591e+04  9.104e+03   3.944 9.51e-05 ***
## num_full_bathrooms                5.234e+04  1.381e+04   3.790 0.000175 ***
## num_half_bathrooms                3.678e+04  1.624e+04   2.264 0.024131 *
```

```
## num_total_rooms                    5.966e+03  6.217e+03   0.960 0.337849
## sq_footage                         3.103e+01  1.369e+01   2.267 0.023973 *
## walk_scoreSomewhat Walkable       -4.662e+03  3.461e+04  -0.135 0.892933
## walk_scoreVery Walkable           -3.579e+04  3.344e+04  -1.070 0.285176
## walk_scoreWalker's Paradise        2.045e+04  3.430e+04   0.596 0.551314
## neighborhoodJamaica               -1.738e+04  2.235e+04  -0.777 0.437395
## neighborhoodNorth Queens           4.620e+04  1.875e+04   2.464 0.014160 *
## neighborhoodNortheast Queens       4.030e+04  2.010e+04   2.005 0.045694 *
## neighborhoodNorthwest Queens       1.751e+05  2.593e+04   6.751 5.40e-11 ***
## neighborhoodSoutheast Queens       2.911e+04  2.400e+04   1.213 0.225851
## neighborhoodSouthwest Queens      -4.318e+04  2.033e+04  -2.124 0.034288 *
## neighborhoodWest Central Queens    6.830e+04  1.937e+04   3.525 0.000473 ***
## neighborhoodWest Queens            3.314e+04  2.006e+04   1.652 0.099358 .
## approx_year_built_is_missing       2.169e+04  3.709e+04   0.585 0.558974
## dining_room_type_is_missing       -2.628e+03  9.693e+03  -0.271 0.786436
## fuel_type_is_missing               1.720e+04  1.821e+04   0.945 0.345443
## kitchen_type_is_missing            1.697e+04  4.205e+04   0.404 0.686803
## maintenance_cost_is_missing       -4.283e+04  3.163e+04  -1.354 0.176549
## num_bedrooms_is_missing                   NA         NA      NA       NA
## sq_footage_is_missing             -2.826e+03  8.632e+03  -0.327 0.743523
## approx_year_built                  9.406e+02  3.222e+02   2.919 0.003713 **
## total_cost                         1.405e+02  1.395e+01  10.076  < 2e-16
```

The in-sample R^2 is 83% and the RMSE is 77057.37$. The out of sample R^2 is 4.77% and the RMSE is 88348.06. We likely had a high in-sample R^2 due to overfitting because of how poor the out of sample error performs. This model is not ideal for predictions however out of sample the linear model does better than the regression tree. This could be because sales prices and corelating features are linear.

The most important feature on the regression tree was square footage, here on the linear model we can see that if all other features are held constant, a one unit increase in square footage will increase the sales price of an apartment by 31.03$.

Random Forest Modeling:

The Random Forest model uses a non-parametric algorithm by using bootstrap aggregation which samples the data with replacement. By using this algorithm, we gain the ability to decorrelate the data which then helps reduce out bias. Here we are allowed to overfit the model. And by setting a range of possible mtrys, ntrees, and nodesizes as our hyperparameters we can have the algorithm to iteratively pick the best possible parameters for a model. Mtrys must be less than equal to the number of columns we have while ntrees and nodes does not have an upper bound for the size. We can most likely expect the random forest to perform the best out of the three models because of the bagging process as well as MLR which will return the optimal node size, tree size and mtry. (note optimal node size was not used in model)

Performance Results for your Random Forest Model:

Split	R^2	RMSE
First Train	80.4%	80515.63%
Second Train	81%	79289.99$
Final Test	73.5%	86654.80$

As expected the oob R^2 on the final test is much better than the out of sample errors of the Regression Tree and OLS models. The in sample and out of sample errors are much closer which also indicates the random forest is better for using in the real world. The generalization error is the oob out of sample RMSE score the error is much higher than the in-sample error thus nnd the R^2 is lower thus the model did not underfit.

Discussion:

Many assumptions were made when futurizing the data in order to deal with missingness. To make the featurization better the walk score, kitchen type and dinning room type could have been made into an ordered factor. This might have helped our out of sample errors in the linear model. I do not believe these models are production ready, we need more data and possibly better features because many of given features had some degree of linear correlation. I can confirm this because when initially building my OLS model I had a rank deficient matrix which means some of the columns were linearly dependent. Therefore no, I do not believe this model can beat Zillow.

References:

. (2018, July). *Set hyperparameters to a learner in mlr after parameter tuning*. Stack Overflow. https://stackoverflow.com/questions/51207549/set-hyperparameters-to-a-learner-in-mlr-after-parameter-tuning.

Crook, D. (2019, December 10). *How NYC Property Taxes Are Calculated: StreetEasy*. StreetEasy Blog. https://streeteasy.com/blog/nyc-property-taxes/#:~:text=In%20New%20York%20City%2C%20the,rise%20to%2021.167%25%20in%202020.

Walk Score Methodology. Walk Score. (n.d.). https://www.walkscore.com/methodology.shtml#:~:text=Walk%20Score%20measures%20the%20walkability%20of%20any%20address%20using%20a,miles)%20are%20given%20maximum%20points.

Acknowledgements:

Thank you, to all my collaborators for talking your time to meet and work on the project together. I found it very valuable to discuss featurization and model building in a group. You guys made cleaning data fun! And special shutout to Janine for providing the code snippet needed to extract zip codes.

Final Project Code:

#Loading the 2016-2017 Housing Data for Queens

```
pacman::p_load(tidyverse, magrittr, data.table, R.utils, skimr)
housing_data = fread("https://raw.githubusercontent.com/kapelner/QC_MATH_342W
_Spring_2021/master/writing_assignments/housing_data_2016_2017.csv")

housing_data = data.frame(housing_data)
```

We have 55 cols and 2230 rows in total. Our repose variable is sales_price.

#Now we are dropping data that is not relevant to our data set.

```
housing_data = housing_data%<>%
  select(-HITId, -HITTypeId, -Title, -Description, -Keywords, -Reward, -Creat
ionTime, -MaxAssignments, -RequesterAnnotation, -AssignmentDurationInSeconds,
-AutoApprovalDelayInSeconds, -NumberOfSimilarHITs, -LifetimeInSeconds, -Assig
nmentId, -WorkerId, -AssignmentStatus, -AcceptTime, -SubmitTime, -AutoApprova
lTime, -ApprovalTime,  -RejectionTime, -RequesterFeedback, -URL, -url, -Expir
ation, -Last30DaysApprovalRate, -Last7DaysApprovalRate ,-WorkTimeInSeconds, -
LifetimeApprovalRate )


#after looking out our data set there are extra cols that we can delete
housing_data = housing_data%<>%
  select( -model_type , -listing_price_to_nearest_1000, -num_floors_in_buildi
ng, -pct_tax_deductibl, -date_of_sale, -community_district_num  )

#-fuel_type
```

#Cleaning the data to reduce unique catigories

#Create zip codes to categorize the locations

```
#from Janine
zip_codes = gsub("[^0-9.-]", "", housing_data$full_address_or_zip_code)
housing_data$zip_codes = str_sub(zip_codes, -5, -1)
```

#Clean Zipcodes

```
#unique(housing_data$zip_codes) # check for any incorrect zipcodes
 housing_data%>%
  filter( housing_data$zip_codes =="1355." |  housing_data$zip_codes == "1367
." | housing_data$zip_codes == "17-30" | housing_data$zip_codes == ".1136" |
housing_data$zip_codes =="71137" |  housing_data$zip_codes == "01137" | housi
ng_data$zip_codes == "81137" | housing_data$zip_codes == "01137" | housing_da
ta$zip_codes == "71136" | housing_data$zip_codes == "51142"| housing_data$zip
_codes == "51135")%>>% #find correct zips and manually change them
```

```r
#manually fix it

#Clean Zip codes

housing_data$zip_codes[housing_data$zip_codes == "1367."] <-"11367"
#housing_data$zip_codes[housing_data$zip_codes == "17-30"] <-NA no address
housing_data$zip_codes[housing_data$zip_codes == ".1136"] <-"11369"
housing_data$zip_codes[housing_data$zip_codes == "1355."] <-"11355"

housing_data$zip_codes[housing_data$zip_codes == "81137"] <-"11372"
housing_data$zip_codes[housing_data$zip_codes == "71137"] <-"11372"
housing_data$zip_codes[housing_data$zip_codes == "01137"] <-"11375"
housing_data$zip_codes[housing_data$zip_codes == "1136"] <-"11364"
housing_data$zip_codes[housing_data$zip_codes == "51142"] <-"11427"

housing_data$zip_codes[housing_data$zip_codes == "51135"] <-"11355"
housing_data$zip_codes[housing_data$zip_codes == "71136"] <-"11364"



housing_data = housing_data[housing_data$zip_codes !="17-30",] #remove rows
#unique(housing_data$zip_codes)

#Mutate Zipcodes into Neigbhorhoods


house2 = housing_data %>%
  mutate(

 zip_codes =    ifelse(zip_codes == "11361" | zip_codes == "11362" | zip_code
s == "11363" | zip_codes == "11364" , "Northeast Queens",zip_codes),
  zip_codes =  ifelse( zip_codes == "11354" | zip_codes == "11355" | zip_code
s == "11356" | zip_codes == "11357" | zip_codes == "11358" | zip_codes == "11
359" | zip_codes == "11360" , "North Queens", zip_codes),

 zip_codes =   ifelse( zip_codes == "11365" | zip_codes == "11366" | zip_code
s == "11367" ,"Central Queens",zip_codes),

zip_codes =    ifelse( zip_codes == "11412" | zip_codes == "11423" | zip_code
s == "11432" | zip_codes == "11433" | zip_codes == "11434" | zip_codes == "11
435" | zip_codes == "11436" , "Jamaica" , zip_codes) ,

zip_codes =    ifelse( zip_codes == "11101" | zip_codes == "11102" | zip_code
s == "11103" | zip_codes == "11104" | zip_codes == "11105" | zip_codes == "11
106","Northwest Queens", zip_codes),

zip_codes =     ifelse(zip_codes == "11374" | zip_codes == "11375" | zip_code
s == "11379" | zip_codes == "11385" , "West Central Queens", zip_codes),
```

```
 zip_codes =   ifelse(zip_codes == "11004" | zip_codes == "11005" | zip_codes
== "11411" | zip_codes == "11413" | zip_codes == "11422" | zip_codes == "1142
6" | zip_codes == "11427" | zip_codes == "11428" | zip_codes == "11429" , "So
utheast Queens", zip_codes),

 zip_codes =   ifelse(zip_codes == "11414" | zip_codes == "11415" | zip_codes
== "11416" | zip_codes == "11417" | zip_codes == "11418" | zip_codes == "1141
9" | zip_codes == "11420"  | zip_codes == "11421" , "Southwest Queens", zip_c
odes),

zip_codes =    ifelse(zip_codes == "11368" | zip_codes == "11369" | zip_codes
== "11370" | zip_codes == "11372" | zip_codes == "11373" | zip_codes == "1137
7" | zip_codes == "11378"  ,"West Queens", zip_codes)



)
#unique(house2$zip_codes)
```

#Mutate other features

```
house2%<>%
  mutate(

    cats_allowed = as.factor(ifelse(cats_allowed == "yes" | cats_allowed == "
y", "yes","no")),

    coop_condo =as.factor(coop_condo),
    zip_codes = as.factor(zip_codes),

    dogs_allowed = as.factor(ifelse(dogs_allowed == "yes" | dogs_allowed == "
yes89", "yes","no")),

    kitchen_type = as.factor(case_when(kitchen_type == "efficiency kitchen" |
kitchen_type == "efficiency" |
                            kitchen_type == "efficiemcy"| kitchen_type ==  "e
fficiency ktchen"|
                            kitchen_type == "efficiency kitchene" ~"efficienc
y kitchen",
                            kitchen_type == "eat in" | kitchen_type == "Eat I
n" | kitchen_type == "eatin" |
                            kitchen_type =="1955"|kitchen_type == "Eat in" ~
"eat in",
                            kitchen_type =="Combo"| kitchen_type == "combo" ~
"combo")),

    dining_room_type= as.factor(case_when(dining_room_type == "none" |
                                dining_room_type == "other" ~"other" , dining
_room_type == "dining area" |
```

```
                                dining_room_type =="combo" ~ "combo", dining_
room_type =="formal" ~"formal")),

    fuel_type = as.factor(ifelse(fuel_type == "other" | fuel_type == "Other"
| fuel_type =="none", "other", fuel_type)),        #not
#using fuel type because the zipcode/location dictates what kind of infrastru
cture a building can built, there fore fuel type is a Spurious relationship..
..


    #change garage to a parking space a parking space that is non underground
could exist and we can tell based on if a parking charges exist also how do w
e know if yes = underground or if it's different,thus make it all the same ca
tegory

    garage_exists = as.factor(ifelse(garage_exists =="Underground"| garage_ex
ists == "Yes" | garage_exists == "yes" |
                              garage_exists == "1" |garage_exists =="eys"  |
is.na(parking_charges) ==FALSE ,"yes","no")), #dealing with missingnes minima
lly

    walk_score = as.factor(case_when( walk_score < 50 ~ "Car-Dependent",
                                  walk_score >49 &walk_score <70 ~ "Somewhat
Walkable",
                                  walk_score>69 & walk_score <90 ~ "Very Walk
able",
                                  walk_score>89 & walk_score <= 100 ~ "Walker
's Paradise")),


   # approx_year_built = as.factor(case_when(approx_year_built < 1939 ~ "Pre
war",
                                        #  approx_year_built >=1939  &  appro
x_year_built  <= 1990~ "During or Post war" ,
                                         # approx_year_built >=1990 ~ "Contem
porary (after 1990)" )),

    common_charges = as.integer(str_remove_all(common_charges, "[$,]")),
    maintenance_cost = as.integer(str_remove_all(maintenance_cost, "[$,]")),
    total_taxes = as.integer(str_remove_all(total_taxes, "[$,]" )), # need na
    total_taxes =(total_taxes/12), # make it a monthly cost
    sale_price = as.integer(str_remove_all(sale_price, "[$,]")) # this is our
reponse varable



   )

 house2%<>%
select(-parking_charges, -full_address_or_zip_code )
```

```
house2 %<>%
  rename(
    parking_exists = garage_exists, #it could have a parking lot isntead of a
garage moreover because we had the parking prices we know a spot somowhere ex
isted
    neighborhood = zip_codes
    )

house2$parking_exists[is.na(house2$parking_exists) ==TRUE ] <-"no" #we are ma
king an assumption here
house2$num_half_bathrooms[is.na(house2$num_half_bathrooms) ==TRUE] <-0 # asum
ption

#Problem sometimes the co_op's do not have a maintenance cost by they have a
common charage, and vis versa one way to deal with this is to swap the charge
s in the right place. co_ops should only have a monthy maintance cost and con
dos should only have a monthy common chrage + total taxtes.


house2$fix_mc_swaps = rowSums(house2[ , c("common_charges","maintenance_cost"
)], na.rm =TRUE)
house2$fix_cc_swaps_add_total_taxes = rowSums(house2[ ,c("common_charges","ma
intenance_cost", "total_taxes")], na.rm =TRUE) #maybe dont add total taxes ye
t
house2$maintenance_cost = house2$fix_mc_swaps
house2$common_charges = house2$fix_cc_swaps_add_total_taxes

#Fill back the Zeros


house_condo = house2%>%
filter(coop_condo =="condo")

house_coop =house2%>%
  filter(coop_condo == "co-op")

house_condo$maintenance_cost <- 0 #condos do not have maintenance_cost
house_condo$common_charges[house_condo$common_charge == 0] <- NA #to impute o
n common_charfes
house_coop$common_charges <- 0 #coops do not have common_charges
house_coop$maintenance_cost[house_coop$maintenance_cost == 0]<- NA #if a zero
impute cost

#Combind back the Data set


house2 =rbind(house_condo, house_coop) #stich back the two data frames

house2 =house2%>%
  select(-fix_mc_swaps, -fix_cc_swaps_add_total_taxes, -total_taxes) # drop u
```

```
necessary cols

house2 <- house2[sample(1:nrow(house2)), ] # to randmize the order of the dat
a again

# final clean drop rows so features have 0% missingness


house2 = house2[is.na(house2$common_charges) == FALSE,] #only one missing jus
t drop row
#house2 = house2[is.na(house2$num_total_rooms) == FALSE,] #only two missing j
ust drop row //dropped it so the lin model can work
```

#Create dummy varaiables for imputation on features that have less than 90% missing

```
set.seed(718)

M = tbl_df(apply(is.na(house2), 2, as.numeric))

## Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.

colnames(M) = paste(colnames(house2), "_is_missing", sep = "")
M = tbl_df(t(unique(t(M))))
M %<>%
  select_if(function(x){sum(x) > 0})

house2 = cbind(M, house2)

missing_col = ncol(M)
```

#create train and test splits

```
obs_without_responses = house2 %>%
  filter(is.na(sale_price))
obs_with_responses = house2 %>%
  filter(!is.na(sale_price))

n = nrow(obs_with_responses) #there are 528 observations with responses that
we can use later on
k = 5

test_indices = sample(1 : n, 1 / k * n)
train_indices = setdiff(1 : n, test_indices)

n_test = as.integer((1 / k) * n)
n_train = as.integer(n - n_test)

training = obs_with_responses[train_indices, ]
```

```
testing = obs_with_responses[test_indices, ]

X_test = testing %>%
  mutate(sale_price = NA)
y_test = testing$sale_price
```

#impute on data using missForest

```
pacman::p_load(missForest)

#fill in missingness
housing_missing = rbind(training, X_test, obs_without_responses) #can use all
data except y_test (to use it would be cheating)

housing_complete = missForest(housing_missing)$ximp

#housing_complete
sum(is.na(housing_complete))

housing = housing_complete %>%
  filter(sale_price_is_missing == 0) %>%
  select(-sale_price_is_missing)

housing = cbind(housing[, -(1:missing_col)], tbl_df(t(unique(t(housing[,(1:mi
ssing_col)]))))) #make sure all col are linearly independent

housing_training = housing[1:n_train, ]
housing_test = housing[(n_train+1):n, ]

housing_test$sale_price = y_test

#combine charges with maintenance cost after imputation before creating model
s
housing_test %<>%
  mutate(total_cost = maintenance_cost + common_charges) %<>%
  select(-maintenance_cost, -common_charges)

housing_training %<>%
  mutate(total_cost = maintenance_cost + common_charges) %<>%
  select(-maintenance_cost, -common_charges)

housing_ytest = housing_test$sale_price
housing_Xtest = housing_test
housing_Xtest$sale_price = NULL

housing_ytrain = housing_training$sale_price
housing_Xtrain = housing_training
housing_Xtrain$sale_price = NULL
```

```r
#Regression Tree Model
pacman::p_load(YARF)
options(java.parameters = "-Xmx4000m")

reg_tree = YARFCART(housing_Xtrain, housing_ytrain)

get_tree_num_nodes_leaves_max_depths(reg_tree)

tree_image = illustrate_trees(reg_tree, max_depth = 5, open_file = TRUE, leng
th_in_px_per_half_split = 40) # will give the locally best nodes

#in-sample stats
y_hat_train = predict(reg_tree, housing_Xtrain)
e = housing_ytrain - y_hat_train
sd(e) #s_e

insamp_r_sq = 1 - sd(e) / sd(housing_ytrain) #R^2
cat("in sample r^2 = ", insamp_r_sq , "\n ")

#oos stats
y_hat_test_tree = predict(reg_tree, housing_Xtest)
e = housing_ytest - y_hat_test_tree
sd(e)

oos_r_sq = 1 - sd(e) / sd(housing_ytest)
cat("oos r^2 =" , oos_r_sq )

#Linear Modeling
pacman::p_load(xtable)

lin_mod = lm(housing_ytrain ~ . , housing_Xtrain)
lin_mod

#in-sample stats
summary(lin_mod)$sigma

lin_insample_rsq =summary(lin_mod)$r.squared
cat("insaple r^2 =" ,lin_insample_rsq, "\n")

xtable(lin_mod)

#oos stats
y_hat_test_linear = predict(lin_mod, housing_Xtest)

e = housing_ytest - y_hat_test_linear
sd(e)

lin_oos_rsq = 1 - sd(e) / sd(housing_ytest)
cat("oos r^2 =" , lin_oos_rsq, "\n")

summary(lin_mod)
```

```
#Random Forest


pacman::p_load(mlr)
#housing_Xcomplete = union_all(housing_Xtrain, housing_Xtest) # its iliegal t
o use the test data
housing_Xcomplete = housing_Xtrain
#y_salesprice = union_all(housing_ytrain, housing_ytest)
y_salesprice = housing_ytrain
mlr_data = cbind(y_salesprice, housing_Xcomplete)
colnames(mlr_data)[1] = "sales_price"
task = makeRegrTask(data = mlr_data, target = "sales_price")

parms = makeParamSet(
  makeIntegerParam("mtry", lower = 2, upper = ncol(housing_Xcomplete)), #feat
ure dependant mtry can not be greater than num of col
  makeIntegerParam("ntree", lower = 2, upper = 100 ), # it is possible to go
higher
  makeIntegerParam("nodesize", lower =2, upper = 100)
)


desc <- makeResampleDesc("CV", iters = 20)
ctrl <- makeTuneControlRandom(maxit = 20)
mlr_ret <- tuneParams("regr.randomForest", task = task, resampling = desc, pa
r.set = parms, control = ctrl, measures = list(rmse))



#Optimal result
mlr_ret

rf_is_mod = YARF(housing_Xtest, housing_ytest, mtry= as.integer(mlr_ret$x[1])
, num_trees = as.integer(mlr_ret$x[2]))

rf_is_mod

yhat = predict(rf_is_mod, housing_Xtest)






#gfinal oos for the final model

oos_rmse = sqrt(mean((housing_ytest - yhat)^2))
oos_rsq = 1 - sum((housing_ytest - yhat)^2)/sum((housing_ytest - mean(y_sales
price))^2)
oos_rmse
```

```
oos_rsq
```