Prediction Model for Apartment Selling Prices

Final Project for Math390 Data Science at Queens College

May24,2020

By: Rameasa Arna

Abstract

The objective of this study is to create a predicting model to predict selling price of the apartment in Queens, NY. The dataset used in this project is the historical apartment sales collected by MLSI Portal from 2016 to 2017. Features in the dataset has been classified based on its important to the sale price of the apartment and predicting models from three different algorithms (Regression Tree, Linear, Random Forest) have been made. Finally, Random Forest model has been discovered to provide a more accurate and stable prediction among the other two in this particular study.

1. Introduction

Predictive modelling is the process that uses data and statistics to predict outcome with data models. These model then later be used to predict outcome in advance for the future. In this study, Apartment sell price models will be generated based on historical sales collected by MLSI portal from 2016 to 2017. After dataset finished cleaning processes, models are created. Among all Random Forest appears to be the best in term of providing accurate and stable prediction.

2. Data

2.1. Description

Data used in this study is apartment description with listing sale price harvested from a collection of Multiple Listing Service (MLS). The raw data contains 2230 observations and 55 different features. Dataset comes from 55 different zip code from mainland Queens except Rockaways which is a peninsular near JFK airport that is geographically distinct from the rest of neighborhoods.

However, some of the variables in this dataset has to be removed since it is not relevant to the study, some are generated based on the feature provided. There are entries in dataset which contains errors, misspell or missing information, data cleaning has been done to ensure data are in good condition to make prediction. Data cleaning process will be performing to many features especially to sale_price variable as it will be the dependent variable of this study. Any missing row data will be ignored. Despite large amount of dataset used, the amount of valid data after cleaning process is reduced therefore there is danger of extrapolation as predicting variables used are outside of the boundary.

2.2. Featurization

Featurization is a crucial step in machine learning pipeline, because the right feature can east the difficulty of the modelling therefore result in a higher quality of the output. There are additional features which can be added into the study such as Zip Code, Pet Allowed, monthly cost, price per sqft etc. However, there are many features have been remove from the dataset which is not relevant

to the model such as HITId, HITTypeId, Title, Description, Keywords etc. After removing and adding necessary features, the dataset contains 19 varibles.

Below table and charts is the description of selected features:

```
approx_year_built community_district_num   coop_condo       dining_room_type garage_exists
Min.   :1893      Min.   : 3.00          Min.   :1.000   combo      :957     Min.   :0.0000
1st Qu.:1950      1st Qu.:25.00          1st Qu.:1.000   dining area:  2     1st Qu.:0.0000
Median :1958      Median :26.00          Median :1.000   formal     :620     Median :0.0000
Mean   :1963      Mean   :26.33          Mean   :1.255   none       :  2     Mean   :0.1812
3rd Qu.:1970      3rd Qu.:28.00          3rd Qu.:2.000   other      :201     3rd Qu.:0.0000
Max.   :2017      Max.   :32.00          Max.   :2.000   NA's       :448     Max.   :1.0000
NA's   :40        NA's   :19
          kitchen_type  num_bedrooms   num_floors_in_building num_full_bathrooms num_half_bathrooms
eatin            :733   Min.   :0.000   Min.   : 1.000         Min.   :1.000      Min.   :0.0000
efficiency kitchen:505   1st Qu.:1.000   1st Qu.: 3.000         1st Qu.:1.000      1st Qu.:1.0000
combo            :349   Median :2.000   Median : 6.000         Median :1.000      Median :1.0000
efficiency       :338   Mean   :1.653   Mean   : 7.785         Mean   :1.231      Mean   :0.9535
eat in           :190   3rd Qu.:2.000   3rd Qu.: 7.000         3rd Qu.:1.000      3rd Qu.:1.0000
(Other)          : 99   Max.   :6.000   Max.   :34.000         Max.   :3.000      Max.   :2.0000
NA's             : 16   NA's   :115     NA's   :650                               NA's   :2058
num_total_rooms  parking_charges pct_tax_deductibl   sale_price      sq_footage      total_taxes
Min.   : 0.000   Min.   : 1.00   Min.   :20.0      Min.   :  1.0   Min.   : 100.0   Min.   :  1.00
1st Qu.: 3.000   1st Qu.:21.00   1st Qu.:40.0      1st Qu.: 68.0   1st Qu.: 743.0   1st Qu.: 76.75
Median : 4.000   Median :41.00   Median :50.0      Median :136.5   Median : 881.0   Median :143.00
Mean   : 4.139   Mean   :44.26   Mean   :45.4      Mean   :144.1   Mean   : 955.4   Mean   :145.90
3rd Qu.: 5.000   3rd Qu.:69.00   3rd Qu.:50.0      3rd Qu.:214.2   3rd Qu.:1100.0   3rd Qu.:219.00
Max.   :14.000   Max.   :89.00   Max.   :75.0      Max.   :315.0   Max.   :6215.0   Max.   :293.00
NA's   :2        NA's   :1671    NA's   :1754      NA's   :1702    NA's   :1210     NA's   :1646
  walk_score     pets_allowed     monthly_cost   price_persqft
Min.   : 7.00   Min.   :0.0000   Min.   :  0.0   Min.   :0.0007
1st Qu.:77.00   1st Qu.:0.0000   1st Qu.:109.0   1st Qu.:0.0982
Median :89.00   Median :0.0000   Median :255.0   Median :0.1485
Mean   :83.92   Mean   :0.3767   Mean   :270.4   Mean   :0.1577
3rd Qu.:95.00   3rd Qu.:1.0000   3rd Qu.:419.0   3rd Qu.:0.1979
Max.   :99.00   Max.   :1.0000   Max.   :842.0   Max.   :1.0900
                                                 NA's   :1425
```

Figure 1 Descriptive statistics of continuous variables

2.3. Errors and Missingness

There are multiple entries in dataset which contains errors, misspell or missing information. Data cleaning process will be performing on many independent variables especially to sale_price variable as it will be the dependent variable of this study. This figure shows the amount of missing entry (NA) value of each feature in our dataset.

```
  approx_year_built community_district_num            coop_condo       dining_room_type
              12756                   12756                 12756                  12756
      garage_exists            kitchen_type          num_bedrooms num_floors_in_building
              12756                   12756                 12756                  12756
num_full_bathrooms      num_half_bathrooms       num_total_rooms        parking_charges
              12756                   12756                 12756                  12756
 pct_tax_deductibl              sale_price           sq_footage            total_taxes
              12756                   12756                 12756                  12756
         walk_score           pets_allowed          monthly_cost          price_persqft
              12756                   12756                 12756                  12756
```

Figure 2 Count of missing entries in different features

To help with missing entry with our dataset, missforest method will be used to assist with imputing the missing information on the observation. MissForest imputes missing values particularly in the case of mixed-type data. It uses a random forest trained on the observed values of a data matrix to predict the missing values. It can be used to impute continuous and/or categorical data including complex interactions and non-linear relations (Stekhoven 2015). To start using the missforest, first sale price variable is separated from other variables then those data will be input into missforest. After missforest learnt from those input, it will return back data without NA so it can be used to predict by different algorithms. However, Sale Price which has NAs value will be discarded since

it should not be imputed as Imputing dependent variable would create problem to the model. Final dataset which will be used contains 528 observations.

3. Modelling

After dataset has been cleaned, final dataset is obtained. It contains 528 observations with no NAs or any missing information which is suitable to train and test the prediction model.

3.1. Regression Tree

Regression tree is a tree building technique which divide dataset into smaller subgroups, which ideally suited to the generation of clinical decision rules (Lewis 2000). Moreover, Regression Tree helps uncover the complex interaction between predictors which might be difficult or impossible to uncover using traditional multivariate techniques. As seen on figure 3, the most important variable in predicting the sale price of an apartment is the price per square footage and total square footage of the apartment. In addition, we also see the effect of the coop_condo as co-op generally pay less compare to condo style apartment and also the higher monthly cost and year built the higher the sale price of the condo. This method has RMSE of 416 with R^2 of 0.83 by predicting test dataset.

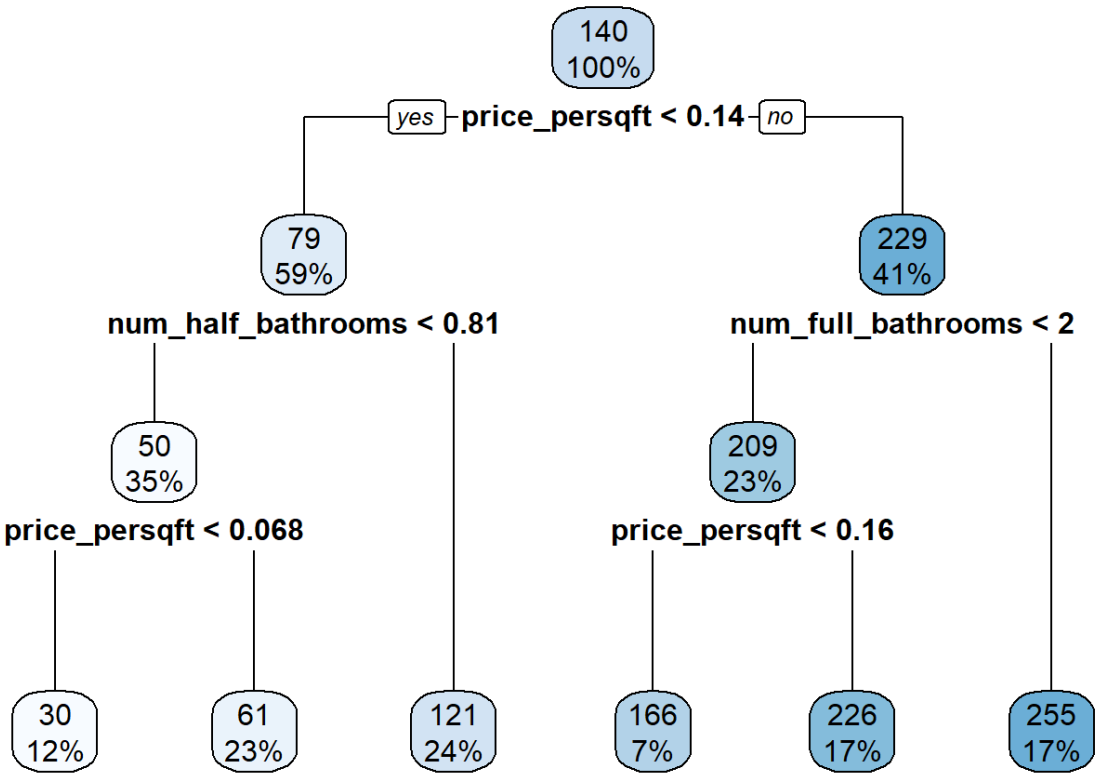


Figure 3 Regression Tree Prediction

3.2. Linear Modelling

From the result of Linear Modelling figure 4, we noticed some similarity in important input between Linear Model and Regression Tree we discuss earlier. Important inputs are number of full bathroom, price per sqft, coop_condo etc. R^2 out sample of 0.88 with RMSE of 348 is achieved. A Linear Model is good but not ideal to be used for this prediction as linear model is assume to have linear interaction where in reality this assumption might not valid.

```
##
## Call:
## lm(formula = sale_price ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -80.460 -13.944  -0.202  12.345 122.765
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2.948e+02  2.232e+02   1.321 0.187420
## approx_year_built              -2.183e-01  1.138e-01  -1.919 0.055699 .
## community_district_num          4.843e-02  4.796e-01   0.101 0.919630
## coop_condo                     -2.842e+01  6.528e+00  -4.353 1.74e-05 ***
## dining_room_typedining area     7.757e+00  1.590e+01   0.488 0.625920
## dining_room_typeformal          7.943e+00  3.600e+00   2.207 0.027951 *
## dining_room_typeother           4.587e+00  4.795e+00   0.957 0.339380
## garage_exists                   2.941e+00  3.932e+00   0.748 0.454883
## kitchen_typecombo              -3.824e+00  2.801e+01  -0.137 0.891480
## kitchen_typeCombo              -7.943e+00  2.777e+01  -0.286 0.775000
## kitchen_typeeat in              4.571e-01  2.741e+01   0.017 0.986705
## kitchen_typeEat in              6.333e+01  3.350e+01   1.891 0.059468 .
## kitchen_typeEat In             -3.467e-01  2.833e+01  -0.012 0.990242
## kitchen_typeeatin               2.348e+01  3.357e+01   0.699 0.484764
## kitchen_typeefficiency          9.654e-01  2.746e+01   0.035 0.971973
## kitchen_typeefficiency kitchen -6.976e+00  3.865e+01  -0.180 0.856875
## num_bedrooms                    9.888e+00  3.376e+00   2.928 0.003619 **
## num_floors_in_building          8.665e-01  2.711e-01   3.196 0.001515 **
## num_full_bathrooms              4.026e+01  4.934e+00   8.160 5.39e-15 ***
## num_half_bathrooms              5.072e+01  1.030e+01   4.925 1.28e-06 ***
## num_total_rooms                -6.707e-01  2.251e+00  -0.298 0.765947
## parking_charges                 1.995e-01  9.403e-02   2.122 0.034543 *
## pct_tax_deductibl               4.787e-01  4.948e-01   0.967 0.334004
## sq_footage                      1.800e-02  4.881e-03   3.687 0.000261 ***
## total_taxes                     2.753e-02  3.903e-02   0.705 0.481005
## walk_score                     -2.537e-01  1.096e-01  -2.315 0.021161 *
## pets_allowed                    5.471e+00  2.896e+00   1.889 0.059632 .
## monthly_cost                   -6.044e-03  1.043e-02  -0.579 0.562796
## price_persqft                   1.241e+03  4.873e+01  25.461  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.57 on 367 degrees of freedom
## Multiple R-squared:  0.9129, Adjusted R-squared:  0.9063
## F-statistic: 137.4 on 28 and 367 DF,  p-value: < 2.2e-16
```

Figure 4 Linear Modelling Output Table

3.3. Random Forest Modelling

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of

the individual trees in the forest and the correlation between them. Random Forest is a substantial modification of bagging that builds a large collection of de-correlated trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune. As a consequence, random forests are popular, and are implemented in a variety of packages (Friedman, Hastie, and Tibshirani 2001). However, the tree in bagging have not complete independence of one to another since all the original predictions are connected to the original first splits. It prevents bagging from being optimal. Yet more data is needed for its effectiveness.

Result from Random Forest modelling of in sample data has R^2 of 0.92 with RMSE of 310.

```
##
## Call:
##  randomForest(formula = sale_price ~ ., data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 566.7436
##                     % Var explained: 92.45
```

Figure 5: Random Forest Output

4. Comparison Result of Forecast Model

	Test Data	
Model	RMSE	R^2
Regression Tree	416	0.83
Linear Model	348	0.88
Random Forest	310	0.92

Table 1 Summary Result of Prediction Models

From Table 1, Random Forest has a good predicting capability for test data while linear model perform good with train data but poorly with test data, it is prompt for over fitting. Where regression tree has lowest performance among all other models.

5. Discussion

Dues to low quality of the dataset which causes many observations to be removed from the model. This result in low amount of sample size to be used, it would be better prediction for regression tree and random forest if the sample size is adequate. Despite all this, this study gives us a good understanding of how those prediction modelling performs. In conclusion, Random Forest model based on creation of multiple decision tree provides a more accurate and stable prediction both in train and test data.

R project

Rameasa Arna

5.1. Load libraries

```
pacman::p_load(dplyr,tidyr,missForest, ggplot2,magrittr,stringr,mlr,ggmap)
```

5.2. Load Housing Data

```
set.seed(5)
raw_housing_data <- read.csv("housing_data_2016_2017.csv")
hd <- raw_housing_data[,-c(0:28)]
hd <- select(hd,-c(model_type,fuel_type,date_of_sale,url))
unique(hd$coop_condo)
```

```
## [1] co-op condo
## Levels: co-op condo
```

5.3. Process data (cleaning and extracting)

```
#process pet allow columns
hd <- mutate(hd, dogs_allowed = ifelse(substr(hd$dogs_allowed,1,3)=="no",0,1)
)
hd <- mutate(hd, cats_allowed = ifelse(substr(hd$cats_allowed,1,3)=="no",0,1)
)
hd <- mutate(hd,pets_allowed = ifelse(cats_allowed + dogs_allowed > 0, 1, 0))
#process CoopCondo
hd <- mutate(hd, coop_condo = factor(tolower(coop_condo)))
#convert garage column to binary
#if those keyword true replace with 1
hd <- mutate(hd,garage_exists = replace(garage_exists, (garage_exists== "eys"
| garage_exists== "UG" | garage_exists== "Underground" |garage_exists == "yes
" | garage_exists== "Yes"),1))

#change variable Types
hd$listing_price_to_nearest_1000 <- as.numeric(hd$listing_price_to_nearest_10
00)
hd$approx_year_built <-  as.numeric(hd$approx_year_built)
hd$community_district_num <- as.numeric(hd$community_district_num)
hd$num_bedrooms <- as.numeric(hd$num_bedrooms)
hd$num_floors_in_building  <- as.numeric(hd$num_floors_in_building)
hd$num_full_bathrooms <-as.numeric(hd$num_full_bathrooms)
hd$num_half_bathrooms  <- as.numeric(hd$num_half_bathrooms)
hd$num_total_rooms <-  as.numeric(hd$num_total_rooms)
hd$pct_tax_deductibl <-  as.numeric(hd$pct_tax_deductibl)
hd$dining_room_type <- as.factor(hd$dining_room_type)
hd$kitchen_type <- as.factor(hd$kitchen_type)
hd$garage_exists <-  as.numeric(hd$garage_exists)
hd$garage_exists[is.na(hd$garage_exists )] <- 0
hd$coop_condo <-  as.numeric(hd$coop_condo)
hd$parking_charges <-  as.numeric(hd$parking_charges)
```

```r
hd$sale_price <- as.numeric(hd$sale_price)
hd$sq_footage <-  as.numeric(hd$sq_footage)
hd$total_taxes <- as.numeric(hd$total_taxes)
hd$walk_score <- as.numeric(hd$walk_score)
hd$common_charges <- as.numeric(hd$common_charges)
hd$maintenance_cost <- as.numeric(hd$maintenance_cost)
hd$maintenance_cost[is.na(hd$maintenance_cost)] <- 0
hd$common_charges[is.na(hd$common_charges)] <- 0
hd$monthly_cost <- as.numeric(hd$common_charges) + as.numeric(hd$maintenance_
cost)

#calculate price per square feet
hd$price_persqft <-  as.numeric(hd$listing_price_to_nearest_1000/hd$sq_footag
e)

#remove full address
hd <- select(hd,-c(full_address_or_zip_code,listing_price_to_nearest_1000))
hd <- select(hd,-c(common_charges,maintenance_cost))
#remove dog and cat allow column
hd <- select(hd,-c(dogs_allowed,cats_allowed))

#printSummary Continuos
summary(hd)
```

```
##  approx_year_built community_district_num   coop_condo       dining_room_t
ype
##  Min.   :1893      Min.   : 3.00          Min.   :1.000   combo      :957
##  1st Qu.:1950      1st Qu.:25.00          1st Qu.:1.000   dining area:  2
##  Median :1958      Median :26.00          Median :1.000   formal     :620
##  Mean   :1963      Mean   :26.33          Mean   :1.255   none       :  2
##  3rd Qu.:1970      3rd Qu.:28.00          3rd Qu.:2.000   other      :201
##  Max.   :2017      Max.   :32.00          Max.   :2.000   NA's       :448
##  NA's   :40        NA's   :19
##  garage_exists                kitchen_type  num_bedrooms
##  Min.   :0.0000   eatin              :733   Min.   :0.000
##  1st Qu.:0.0000   efficiency kitchen:505   1st Qu.:1.000
##  Median :0.0000   combo              :349   Median :2.000
##  Mean   :0.1812   efficiency         :338   Mean   :1.653
##  3rd Qu.:0.0000   eat in             :190   3rd Qu.:2.000
##  Max.   :1.0000   (Other)            : 99   Max.   :6.000
##                   NA's               : 16   NA's   :115
##  num_floors_in_building num_full_bathrooms num_half_bathrooms num_total_ro
oms
##  Min.   : 1.000         Min.   :1.000      Min.   :0.0000     Min.   : 0.0
00
##  1st Qu.: 3.000         1st Qu.:1.000      1st Qu.:1.0000     1st Qu.: 3.0
00
##  Median : 6.000         Median :1.000      Median :1.0000     Median : 4.0
00
##  Mean   : 7.785         Mean   :1.231      Mean   :0.9535     Mean   : 4.1
```

```
39
##  3rd Qu.: 7.000         3rd Qu.:1.000      3rd Qu.:1.0000     3rd Qu.: 5.0
00
##  Max.   :34.000         Max.   :3.000      Max.   :2.0000     Max.   :14.0
00
##  NA's   :650                               NA's   :2058       NA's   :2
##  parking_charges pct_tax_deductibl   sale_price      sq_footage
##  Min.   : 1.00   Min.   :20.0      Min.   :  1.0   Min.   : 100.0
##  1st Qu.:21.00   1st Qu.:40.0      1st Qu.: 68.0   1st Qu.: 743.0
##  Median :41.00   Median :50.0      Median :136.5   Median : 881.0
##  Mean   :44.26   Mean   :45.4      Mean   :144.1   Mean   : 955.4
##  3rd Qu.:69.00   3rd Qu.:50.0      3rd Qu.:214.2   3rd Qu.:1100.0
##  Max.   :89.00   Max.   :75.0      Max.   :315.0   Max.   :6215.0
##  NA's   :1671    NA's   :1754      NA's   :1702    NA's   :1210
##   total_taxes       walk_score     pets_allowed     monthly_cost
##  Min.   :  1.00   Min.   : 7.00   Min.   :0.0000   Min.   :  0.0
##  1st Qu.: 76.75   1st Qu.:77.00   1st Qu.:0.0000   1st Qu.:109.0
##  Median :143.00   Median :89.00   Median :0.0000   Median :255.0
##  Mean   :145.90   Mean   :83.92   Mean   :0.3767   Mean   :270.4
##  3rd Qu.:219.00   3rd Qu.:95.00   3rd Qu.:1.0000   3rd Qu.:419.0
##  Max.   :293.00   Max.   :99.00   Max.   :1.0000   Max.   :842.0
##  NA's   :1646
##  price_persqft
##  Min.   :0.0007
##  1st Qu.:0.0982
##  Median :0.1485
##  Mean   :0.1577
##  3rd Qu.:0.1979
##  Max.   :1.0900
##  NA's   :1425
```

5.4. Missing Values

```
sapply(hd, function(x) sum(is.na(hd)))
```

```
##      approx_year_built community_district_num             coop_condo
##                  12756                  12756                  12756
##       dining_room_type          garage_exists           kitchen_type
##                  12756                  12756                  12756
##           num_bedrooms num_floors_in_building     num_full_bathrooms
##                  12756                  12756                  12756
##     num_half_bathrooms        num_total_rooms        parking_charges
##                  12756                  12756                  12756
##      pct_tax_deductibl             sale_price            sq_footage
##                  12756                  12756                  12756
##            total_taxes             walk_score          pets_allowed
##                  12756                  12756                  12756
##           monthly_cost          price_persqft
##                  12756                  12756
```

5.5. Features Imputation

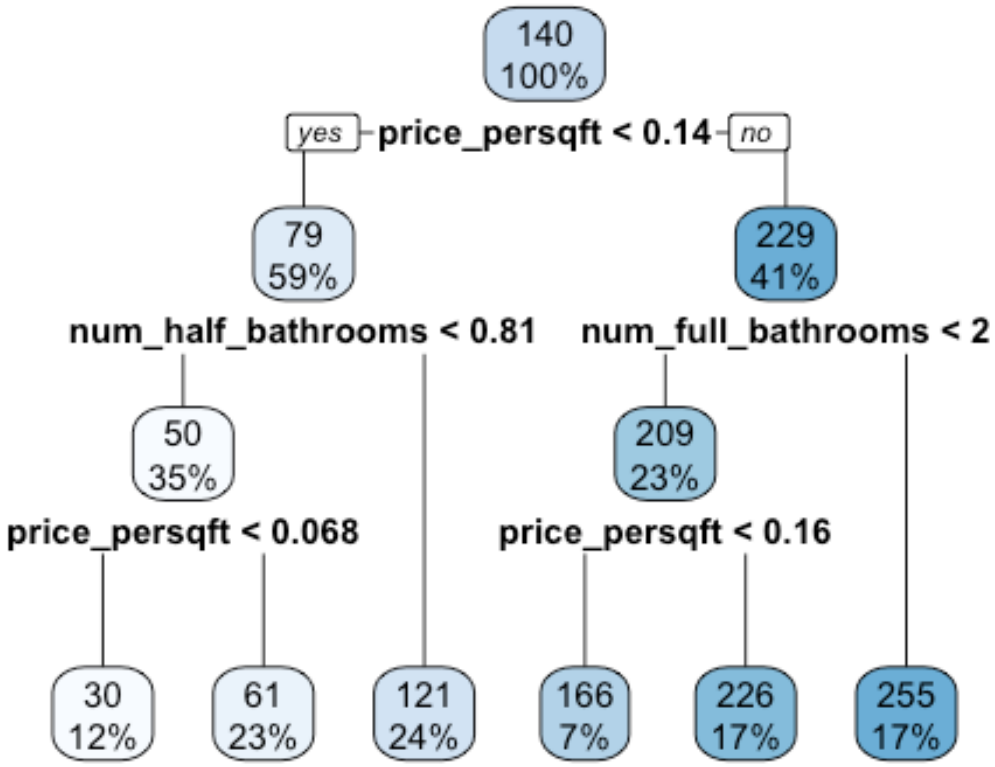
```
imputed_hd <- missForest(hd)

##   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 wan
t 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 wan
t 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 wan
t 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 wan
t to
## do regression?

## done!
##   missForest iteration 5 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 wan
t to
## do regression?

## done!
##   missForest iteration 6 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 wan
t to
## do regression?

## done!
```

5.6. Train-Test Split

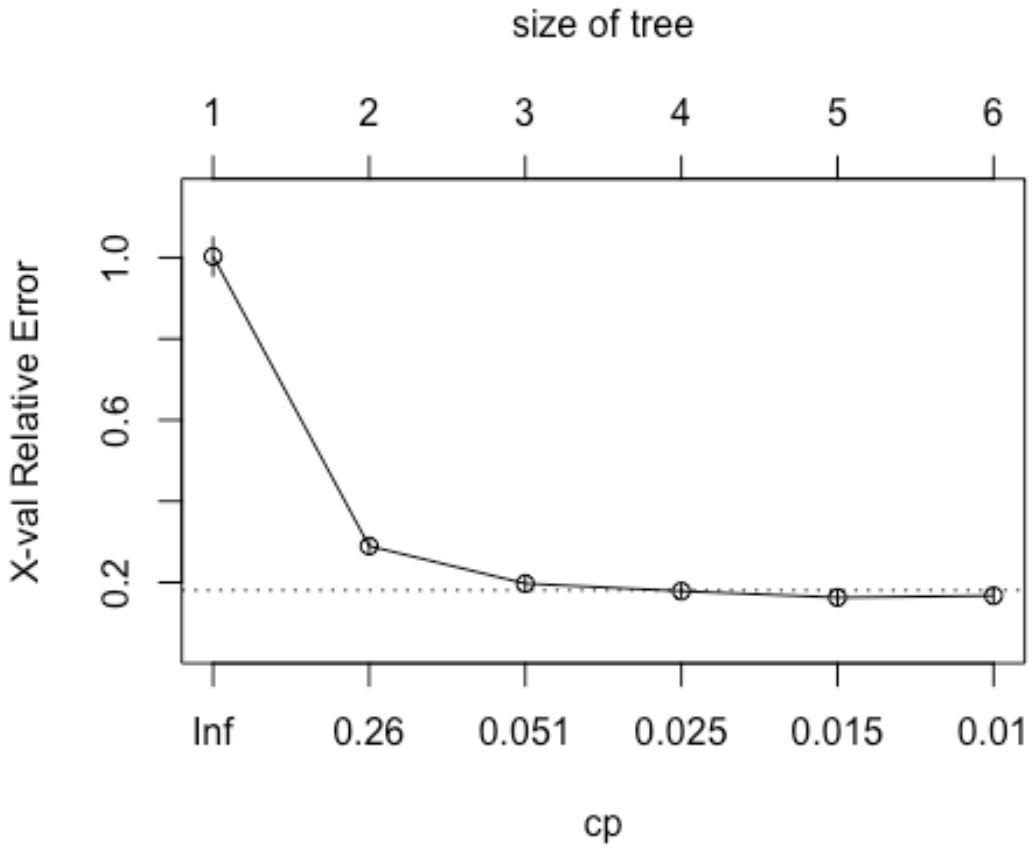
```
#Split data to train and test data
hd_new <- imputed_hd$ximp[!is.na(hd$sale_price ),]


sale_price <- hd$sale_price[!is.na(hd$sale_price )]
hd_new<-cbind(hd_new,sale_price)




train <- hd_new[1:396,]

test <- hd_new[397:528,]
```

5.7. REGRESSION TREE

```
#calculate Rsquare
rsq <- function(x, y) summary(lm(y~x))$r.squared

pacman::p_load(rsample,rpart,rpart.plot,ipred,caret)
regTree = rpart(
  formula = sale_price ~ .,
  data = train,
  method = "anova"
)
printcp(regTree)

##
## Regression tree:
## rpart(formula = sale_price ~ ., data = train, method = "anova")
##
## Variables actually used in tree construction:
## [1] num_full_bathrooms num_half_bathrooms price_persqft
##
## Root node error: 2974058/396 = 7510.2
##
## n= 396
##
##         CP nsplit rel error  xerror     xstd
## 1 0.719771      0   1.00000 1.00283 0.046970
## 2 0.094069      1   0.28023 0.28965 0.019274
## 3 0.027725      2   0.18616 0.19731 0.018343
## 4 0.023074      3   0.15843 0.17860 0.016645
## 5 0.010305      4   0.13536 0.16296 0.017751
## 6 0.010000      5   0.12506 0.16728 0.017889

rpart.plot(regTree)
```




```
plotcp(regTree)
```




```
summary(regTree)
```

```
## Call:
## rpart(formula = sale_price ~ ., data = train, method = "anova")
##   n= 396
##
##           CP nsplit rel error    xerror       xstd
## 1 0.71977149      0 1.0000000 1.0028313 0.04696964
## 2 0.09406914      1 0.2802285 0.2896517 0.01927391
## 3 0.02772514      2 0.1861594 0.1973133 0.01834278
## 4 0.02307395      3 0.1584342 0.1785963 0.01664507
## 5 0.01030517      4 0.1353603 0.1629607 0.01775130
## 6 0.01000000      5 0.1250551 0.1672805 0.01788860
##
## Variable importance
##      price_persqft num_half_bathrooms       monthly_cost         coop_cond
o
##                 30                 15                 14                  1
3
##  approx_year_built num_full_bathrooms         sq_footage    num_total_room
s
##                 12                 10                  2
2
```

```
##       num_bedrooms  pct_tax_deductibl
##                  1                  1
##
## Node number 1: 396 observations,    complexity param=0.7197715
##   mean=139.7071, MSE=7510.248
##   left son=2 (235 obs) right son=3 (161 obs)
##   Primary splits:
##       price_persqft      < 0.1395036  to the left,  improve=0.7197715, (0
missing)
##       num_half_bathrooms < 0.875      to the left,  improve=0.4904524, (0
missing)
##       sq_footage         < 850.9969   to the left,  improve=0.3336604, (0
missing)
##       monthly_cost       < 213.5      to the right, improve=0.3333498, (0
missing)
##       num_full_bathrooms < 1.5        to the left,  improve=0.3306706, (0
missing)
##   Surrogate splits:
##       monthly_cost       < 213.5      to the right, agree=0.803, adj=0.516
, (0 split)
##       coop_condo         < 1.5        to the left,  agree=0.793, adj=0.491
, (0 split)
##       approx_year_built  < 1963.5     to the left,  agree=0.783, adj=0.466
, (0 split)
##       num_half_bathrooms < 0.875      to the left,  agree=0.770, adj=0.435
, (0 split)
##       num_full_bathrooms < 1.5        to the left,  agree=0.735, adj=0.348
, (0 split)
##
## Node number 2: 235 observations,    complexity param=0.09406914
##   mean=78.85106, MSE=2070.322
##   left son=4 (140 obs) right son=5 (95 obs)
##   Primary splits:
##       num_half_bathrooms < 0.805      to the left,  improve=0.5750303, (0
missing)
##       price_persqft      < 0.1020952  to the left,  improve=0.5466081, (0
missing)
##       sq_footage         < 811.9005   to the left,  improve=0.3274644, (0
missing)
##       num_total_rooms    < 4.5        to the left,  improve=0.2195340, (0
missing)
##       num_bedrooms       < 1.5        to the left,  improve=0.1985692, (0
missing)
##   Surrogate splits:
##       price_persqft     < 0.1020952  to the left,  agree=0.826, adj=0.568,
(0 split)
##       sq_footage        < 892.6875   to the left,  agree=0.757, adj=0.400,
(0 split)
##       num_total_rooms   < 4.5        to the left,  agree=0.719, adj=0.305,
(0 split)
```

```
##       num_bedrooms      < 1.5        to the left,  agree=0.706, adj=0.274,
(0 split)
##       pct_tax_deductibl < 48.555     to the left,  agree=0.655, adj=0.147,
(0 split)
##
## Node number 3: 161 observations,    complexity param=0.02772514
##   mean=228.5342, MSE=2154.597
##   left son=6 (93 obs) right son=7 (68 obs)
##   Primary splits:
##       num_full_bathrooms < 1.5        to the left,  improve=0.2377012, (0
missing)
##       monthly_cost       < 448.5      to the right, improve=0.2085590, (0
missing)
##       price_persqft      < 0.1619639  to the left,  improve=0.2016065, (0
missing)
##       sq_footage         < 1312.292   to the left,  improve=0.1642609, (0
missing)
##       total_taxes        < 155.89     to the left,  improve=0.1464875, (0
missing)
##   Surrogate splits:
##       sq_footage       < 1023.965   to the left,  agree=0.814, adj=0.559,
(0 split)
##       num_total_rooms  < 4.5        to the left,  agree=0.776, adj=0.471,
(0 split)
##       num_bedrooms     < 2.5        to the left,  agree=0.714, adj=0.324,
(0 split)
##       monthly_cost     < 210.5      to the right, agree=0.677, adj=0.235,
(0 split)
##       dining_room_type splits as  L-R-L, agree=0.671, adj=0.221, (0 split)
##
## Node number 4: 140 observations,    complexity param=0.01030517
##   mean=50.42857, MSE=803.602
##   left son=8 (49 obs) right son=9 (91 obs)
##   Primary splits:
##       price_persqft          < 0.06836786 to the left,  improve=0.27241770
, (0 missing)
##       community_district_num < 26.5       to the right, improve=0.18313590
, (0 missing)
##       sq_footage             < 877.9283   to the left,  improve=0.12219670
, (0 missing)
##       kitchen_type           splits as  -LLRRL--L----, improve=0.05089122,
(0 missing)
##       monthly_cost           < 456.5      to the left,  improve=0.05080973
, (0 missing)
##   Surrogate splits:
##       total_taxes       < 124.37     to the right, agree=0.764, adj=0.327,
(0 split)
##       parking_charges   < 36.56833   to the right, agree=0.750, adj=0.286,
(0 split)
##       pct_tax_deductibl < 45.515     to the right, agree=0.686, adj=0.102,
```

```
(0 split)
##       sq_footage        < 580.075    to the left,  agree=0.686, adj=0.102,
(0 split)
##       approx_year_built < 1957.5     to the right, agree=0.679, adj=0.082,
(0 split)
##
## Node number 5: 95 observations
##   mean=120.7368, MSE=992.1518
##
## Node number 6: 93 observations,    complexity param=0.02307395
##   mean=209.1828, MSE=2123.805
##   left son=12 (26 obs) right son=13 (67 obs)
##   Primary splits:
##       price_persqft      < 0.1619639  to the left,  improve=0.34743520, (0
missing)
##       parking_charges    < 48.14      to the left,  improve=0.23886450, (0
missing)
##       monthly_cost       < 448.5      to the right, improve=0.15315870, (0
missing)
##       total_taxes        < 148.855    to the left,  improve=0.14601210, (0
missing)
##       num_half_bathrooms < 0.935      to the left,  improve=0.09637415, (0
missing)
##   Surrogate splits:
##       pct_tax_deductibl < 50.659     to the right, agree=0.753, adj=0.115,
(0 split)
##       sq_footage        < 1016.775   to the right, agree=0.753, adj=0.115,
(0 split)
##       monthly_cost      < 20         to the left,  agree=0.753, adj=0.115,
(0 split)
##       parking_charges   < 3.5        to the left,  agree=0.742, adj=0.077,
(0 split)
##       kitchen_type      splits as  LRRR-R--RR---, agree=0.731, adj=0.038,
(0 split)
##
## Node number 7: 68 observations
##   mean=255, MSE=984.1176
##
## Node number 8: 49 observations
##   mean=30.26531, MSE=249.9908
##
## Node number 9: 91 observations
##   mean=61.28571, MSE=764.9074
##
## Node number 12: 26 observations
##   mean=165.5769, MSE=1104.859
##
## Node number 13: 67 observations
##   mean=226.1045, MSE=1494.989
```

```r
predicted = predict(regTree, test)
e = predicted - test$sale_price
sqrt(sum(e^2))
```

```
## [1] 415.9834
```

```r
rsq(predicted, test$sale_price)
```

```
## [1] 0.8332083
```

5.8. Linear Regression

```r
linear = lm(sale_price ~ ., data = train)## simple linear model
summary(linear)
```

```
##
## Call:
## lm(formula = sale_price ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -80.460 -13.944  -0.202  12.345 122.765
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2.948e+02  2.232e+02   1.321 0.187420
## approx_year_built              -2.183e-01  1.138e-01  -1.919 0.055699 .
## community_district_num          4.843e-02  4.796e-01   0.101 0.919630
## coop_condo                     -2.842e+01  6.528e+00  -4.353 1.74e-05 ***
## dining_room_typedining area     7.757e+00  1.590e+01   0.488 0.625920
## dining_room_typeformal          7.943e+00  3.600e+00   2.207 0.027951 *
## dining_room_typeother           4.587e+00  4.795e+00   0.957 0.339380
## garage_exists                   2.941e+00  3.932e+00   0.748 0.454883
## kitchen_typecombo              -3.824e+00  2.801e+01  -0.137 0.891480
## kitchen_typeCombo              -7.943e+00  2.777e+01  -0.286 0.775000
## kitchen_typeeat in              4.571e-01  2.741e+01   0.017 0.986705
## kitchen_typeEat in              6.333e+01  3.350e+01   1.891 0.059468 .
## kitchen_typeEat In             -3.467e-01  2.833e+01  -0.012 0.990242
## kitchen_typeeatin               2.348e+01  3.357e+01   0.699 0.484764
## kitchen_typeefficiency          9.654e-01  2.746e+01   0.035 0.971973
## kitchen_typeefficiency kitchen -6.976e+00  3.865e+01  -0.180 0.856875
## num_bedrooms                    9.888e+00  3.376e+00   2.928 0.003619 **
## num_floors_in_building          8.665e-01  2.711e-01   3.196 0.001515 **
## num_full_bathrooms              4.026e+01  4.934e+00   8.160 5.39e-15 ***
## num_half_bathrooms              5.072e+01  1.030e+01   4.925 1.28e-06 ***
## num_total_rooms                -6.707e-01  2.251e+00  -0.298 0.765947
## parking_charges                 1.995e-01  9.403e-02   2.122 0.034543 *
## pct_tax_deductibl               4.787e-01  4.948e-01   0.967 0.334004
## sq_footage                      1.800e-02  4.881e-03   3.687 0.000261 ***
## total_taxes                     2.753e-02  3.903e-02   0.705 0.481005
## walk_score                     -2.537e-01  1.096e-01  -2.315 0.021161 *
## pets_allowed                    5.471e+00  2.896e+00   1.889 0.059632 .
```

```
## monthly_cost                   -6.044e-03  1.043e-02  -0.579 0.562796
## price_persqft                   1.241e+03  4.873e+01  25.461  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.57 on 367 degrees of freedom
## Multiple R-squared:  0.9129, Adjusted R-squared:  0.9063
## F-statistic: 137.4 on 28 and 367 DF,  p-value: < 2.2e-16
```

```
predicted = predict(linear, test)
e = predicted - test$sale_price
sqrt(sum(e^2) )
```

```
## [1] 347.7819
```

```
rsq(predicted, test$sale_price)
```

```
## [1] 0.8830125
```

5.9. Random Forest

```
RF <- randomForest(
  formula = sale_price ~ .,
  data = train
)
RF
```

```
##
## Call:
##  randomForest(formula = sale_price ~ ., data = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 566.7436
##                     % Var explained: 92.45
```

```
which.min(RF$mse)
```

```
## [1] 467
```

```
sqrt(RF$mse[which.min(RF$mse)])
```

```
## [1] 23.78569
```

```
predicted = predict(RF,test)
e = predicted - test$sale_price
sqrt(sum(e^2))
```

```
## [1] 310.0141
```

```
rsq(predicted, test$sale_price)
```

```
## [1] 0.9152029
```

References

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. V ol. 1. 10. Springer series in statistics New Y ork. (https://web.stanford.edu/~hastie/ElemStatLearn/)

Lewis, Roger J. 2000. "An Introduction to Classification and Regression Tree (CART) Analysis." In *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California*.Vol.14. (https://www.researchgate.net/publication/240719582_An_Introduction_to_Classification_and_Regression_Tree_CART_Analysis)

Stekhoven, Daniel J. 2015. "MissForest: Nonparametric Missing Value Imputation Using Random Forest." *Astrophysics Source Code Library*. (https://ui.adsabs.harvard.edu/abs/2015ascl.soft05011S/abstract)