Burhan A Hanif

Collaborators Vincent Miceli Adriana Sham Sakib Salim Juan Diego Astudillo

An attempt to beat Zestimates

A model is a systematic description of a phenomenon that shares valid and important Characteristics of its causation. Prediction model takes important features as inputs of the data-set to calculate if the output is valid given these features. For this project a prediction model will learn from the features (x's) and predict the (y's) and the quantify the errors. The model will predict a future or new entry of an apartment selling price in Queens New York. The training of this model we used the raw data of Queens Apartments for sale from 2016 to 2017 scrapped from MLSLI using MTurk. We used three algorithms for prediction of the Sale Price Linear Model, Regression Tree Model and Random Forest Model. Our raw Data started with 55 features after removing irrelevant features from the data-set to 25 and after featurization we concluded that with these 35 features we will get better performance with our chosen algorithms. The Linear model helped us understand the influence that each independent variable has on our dependent variable (Sale Price). Regression Tree Modeling assisted us to find more substantial ways to make splits on the data. It gave us results a tree with decision nodes and leaf nodes depending on the regressions, finally Random Forests returned splits based on the creation of multiple decision trees that merged together to give us a more accurate and stable predictions. The data provided for this project class comes from a collection of listing sale prices out of the portal Multiple Listing Services. The raw data contains 2230 unique observations (n). 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. The population of interest are apartments in Queens county and our model will be able to predict the price. The dataset is a partial representation of the entire population because it represents a small sample size of Queens with the zip-codes. Certain features have a causation for Sale price and others have been cleaned for their usefulness. Some features are taken out since they were not relevant to our prediction model. We did not use external sources for features, but we created extra variables based on the features that were already provided. "Different models extrapolate differently" there are dangers of extrapolation predicting outside the range of our features. A feature such as zip-code, apartments in different zip-codes could be valued higher or even lower having a dependence on school district and crime per capita. Apartment size per square foot can be valued high in a certain building because of the facilities the building provides. Being outside the range for bedrooms and bathrooms our model will not be able to predict for 4, 5, 6 bedrooms because they can be valued exponentially higher than 0,1,2,3 our model is predicting. The dataset had some outliers that were fixed with help of featurization. For example, there was house worth close to \$1M. Another example total tax had a range of \$11 to \$9300. We had to make those changes manually making them NA's because our model will not account on those weird cases, it will predict poorly on them. In addition, observations had entry errors such as misspelled words. We edited the data in MS Excel and in our code.

approx_year_built: integer; prewar built would be concrete walls, and brick outside, old elevators and the ones built in the modern ones that are built would be wood with new elevators. This would affect the maintenance cost. community_district_num: integer; determines school districts more expensive for better school district. coop_condo: factor; apartments that are either coops or condos. coops are those that have more community charges, you don't own the apartment, however, you own a stock in the corporation that owns the building. To buy co-ops, you must be approved by the corporation board. Co-ops also have less freedom in renting or subletting. Co-ops are also cheaper then condos. In Comparison to co-ops a condo is owned by the person and has the freedom to make the changes in the floor plans also can rent or sublet the apartment. The condos are almost 2x as expensive than a co-op. Adds higher sale price to the condo dining_room_type: factor; this was factorial and was combo, formal or other. Add more to the sale price if more space and fancier. garage_exists: factor; existence of garage in the coop/condo. Having a parking spot would add value to the sale price. kitchen_type: factor; the feature was a factor, but we made it into two different kinds "eat in" and "efficiency" or none. num_bedrooms: integer; number of bedrooms in the coop/condo, more bedrooms add value to the sale price. num_floors_in_building: integer; number of floors in the coop/condo higher floors for views more value added to the sale price. num_full_bathrooms: integer; number of full bathrooms in the coop/condo more bathrooms mean more square foot in the apartment and adds value to the sale price. num_half_bathrooms: integer; number of half bathrooms in the coop/condo. It adds more value because of space and convenience but not as much as full bathrooms. num_total_rooms: integer; number of total rooms in the coop/condo. Amount of rooms as a total beneficial of how the apartment is built. parking_charges: factor; the charge for the parking space in the coop/condo. Being if the parking charges are higher the value of the sale price can be decreased. pct_tax_deductibl: integer; the percentage of tax deduction you can take for owning the property (Tax deductible income). This adds value to the price. sale_price: factor; the price that the coop/condo was sold for our (Y) what we are predicting for, sq_footage: integer; the size of the apartment by square feet. High sq/ft adds more to the sale price. total_taxes: factor; the property tax charged by the local government higher value brings down sale price. walk_score: integer; Metric created by the MLSI, it is the walk to the nearby stores, parks, closest subway. we extracted zip_code from the full address we made the collinearity of dogs allowed and cats allowed into one variable of pets allowed so we got rid of cats and dogs allowed we made Coop-condo into a factor variable because we made a change back from binary. we made maintenance cost and common charges into one numeric variable named monthly cost because these are monthly dues for coops and condos and are mutually exclusive. Then after making the new variable we dropped the two maintence cost and common charges. garage exists to binary variable 0,1 and then turning 0s from NAs to 0 because if they were not listed as yes then they were probably NO. We had to make changes to variable types because the excel file was not reading them as that. In the following code we made changes of variables. We mutated the dining rooms as a factor for its different types. The reason we are changing to character first and then to numeric second for the same variable different times was because as charter we would get a garbage entry. Garage exists, parking charges , sales price , total taxes and price per/sq foot. We created another feature from full address and zip code to a continuous numeric variable of latitude and longitude. We used the package ggmap and the function geocode for calculating these but before we had to edit the excel document to make corrections to addresses. The reason was to give us exact location instead of zip-code which can be anywhere in the 4- or 5-mile radius. We also used latitude and longitude for the closest LIRR Train station distances for fastest way to Midtown Manhattan by public transportation. Then remove the address and zip-code because it is redundant now. Remove the listing price because we put in price per square-foot. Created another temporary variable column-id so we can keep track of the of the real y after separating the y-vector for running Miss forest. After running miss forest bring back the y vector to its real rows. Then we spilt the data into (Xtrain, Ytrain) and (Xtest, Ytest) with a spilt of 80-20 so we can train on the 80% and validate on the 20%. Comparing the in-sample error with our out of sample error. The development of the M matrix with p columns that represents missingness before we run missing forest. The reason for this matrix is because the data is missing and it's taking account of the missingness too see the effects that it takes after running the algorithms. We took out the colinear entries and the features that did not have the missingness. We ran the miss forest on the data without the Y vector because or else it would have calculated the garbage y's. After the implementation of the of Miss forest we merged the y's back to their proper rows using the temp variable and then deleted the temp variable. Regression tree produced these 6 for the splits: coop-condo, square footage, price per sq/ft, parking charges, latitude and monthly costs. The difference between coop-condo was obvious because the price has almost a difference of 2x the times of the condo. Square footage spilt means the bigger the apartment the higher the sale price. The higher the parking charges for the apartment the higher price would be. The latitude if higher would be more north and northern properties are usually more expensive. The higher the monthly costs for the apartment would mean the nicer the building and it would have more features. For Linear regression we ran Ytrain onto Xtrain to get the in-sample fit. Our R^2 was 85% our RMSE was 72,000. The coefficient for coop-condo was 17000, the price for square foot 47,540, latitude 67,000, monthly charges were 163 and parking charges were 395. Like our regression tree these features were important linear model as they had high coefficients. The random forest gave us the best predictions our RMSE was 70,000 and our R^2 was 85%. They can compute different variable types such factors, binary, numeric etc. The algorithm has built in optimizations for decisions and the they are able to find the best splits in the data minimizing the error. For our future predictions it would perform okay but not great since it is 70,000 of from the price and the R^2 was 85%. In conclusion we could have performed better if we had more observations in our dataset. We only had 528 real n observations the rest of the data-set was missing the sale price our y's. We could have run miss forest on the missing y's but that would have been dishonest. It is strange that our linear model was as good as random forest model. If we to include interactions and square terms we could almost certainly beat random forest. To take it a step further we could run forward stepwise that would find the optimal features including interactions, those would be able to create curves instead of a simple line which would fit y much closer.

title: "Term Project 390.4- 2019" output: word_document: default pdf_document: default Author: Juan D Astudillo, Vincent Miceli, Adriana Sham, Burhan Hanif, Sakib Salim —

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,   Requ
```

Clean Data

```
housing_data %<>%
  mutate( zip_code = str_extract(full_address_or_zip_code, "[0-9]{5}"))
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
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
# 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"))
## Garage exists conver it to binary
d %<>%
  mutate(garage_exists = sjmisc::rec(garage_exists, rec = "NA = 0 ; else = copy")) ##recode from NA to
d %<>%
  mutate(garage_exists = sjmisc::rec(garage_exists, rec = " eys = 1; UG = 1 ; Underground = 1; yes = 1
d %<>%
  select(-c(maintenance_cost , common_charges, model_type))
str(d)
```

```
## 'data.frame':    2230 obs. of  24 variables:
##  $ approx_year_built          : int  1955 1955 2004 2002 1949 1938 1950 1960 1960 2005 ...
##  $ community_district_num     : int  25 25 24 25 26 28 29 28 25 30 ...
##  $ coop_condo                 : Factor w/ 2 levels "co-op","condo": 1 1 2 2 1 1 1 1 1 2 ...
##  $ dining_room_type           : Factor w/ 5 levels "combo","dining area",..: 1 3 1 1 1 1 1 NA NA 5
##  $ full_address_or_zip_code     : Factor w/ 1176 levels " Bayside NY, 11360",..: 1158 562 24 223 497
##  $ garage_exists                : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ kitchen_type                 : Factor w/ 4 levels "combo","eat in",..: 2 2 3 2 2 2 3 3 2 2 ...
##  $ num_bedrooms                 : int  2 1 1 3 2 2 1 0 1 1 ...
##  $ num_floors_in_building       : int  6 7 1 NA 2 6 NA 2 NA 4 ...
##  $ num_full_bathrooms           : int  1 1 1 2 1 1 1 1 1 1 ...
##  $ num_half_bathrooms           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ num_total_rooms              : int  5 4 3 5 4 4 3 2 4 3 ...
##  $ parking_charges              : Factor w/ 90 levels " NA ","100","105",..: 1 1 1 1 1 1 1 1 41 1 ..
##  $ pct_tax_deductibl            : int  NA NA NA NA 39 NA NA NA NA NA ...
##  $ sale_price                   : Factor w/ 316 levels " NA ","100000",..: 107 113 33 252 119 126 38
##  $ sq_footage                   : int  NA 890 550 NA 675 1000 NA 375 NA 681 ...
##  $ total_taxes                  : Factor w/ 294 levels " NA ","100","1024",..: 1 1 255 68 1 1 1 1 1
##  $ walk_score                   : int  82 89 90 94 71 90 72 93 70 98 ...
##  $ listing_price_to_nearest_1000: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ lat                          : num  40.7 40.8 40.7 40.8 40.7 ...
##  $ lon                          : num  -73.8 -73.8 -73.9 -73.8 -73.7 ...
##  $ zip_code                     : chr  "11355" "11354" "11368" "11354" ...
##  $ pets_allowed                 : num  0 0 0 0 1 1 0 0 0 0 ...
##  $ monthly_cost                 : num  767 604 167 275 660 932 660 514 781 NA ...
```

##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)) %>%
  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)) %>%
  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("coord.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
```

```r
  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:

```r
####CREATE A COLUMN ID
d %<>%
  ungroup(d) %>%
  mutate(id = 1 : 2230)
d %<>%
  mutate(total_taxes = ifelse(d$total_taxes < 1000, NA, total_taxes))
real_y = data.frame(d$id, d$sale_price)
real_d = subset(d, (!is.na(d$sale_price)))
fake_d = subset(d, (is.na(d$sale_price)))
real_d$sale_price = NULL
fake_d$sale_price = NULL
```

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

```r
train_indices = sample(1 : nrow(real_d), nrow(real_d)*4/5)
training_data = real_d[train_indices, ]
testing_data = real_d[-train_indices, ]
X = rbind(training_data, testing_data, fake_d)
```

#Let's first create a matrix with p columns that represents missingness

```
M = tbl_df(apply(is.na(X), 2, as.numeric))
colnames(M) = paste("is_missing_", colnames(X), sep = "")
```

#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})
```

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!
```

```r
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
## -293856  -37685    -230   35325  330462
##
## Coefficients: (3 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -3.357e+07  9.826e+06  -3.416 0.000688
## approx_year_built                 2.207e+02  2.505e+02   0.881 0.378626
## community_district_num            3.948e+03  1.153e+03   3.425 0.000666
## coop_condocondo                   1.538e+05  1.750e+04   8.786  < 2e-16
## dining_room_typedining area       2.233e+04  5.317e+04   0.420 0.674717
## dining_room_typeformal            2.970e+04  8.567e+03   3.467 0.000573
## dining_room_typeother             1.567e+04  1.148e+04   1.364 0.173112
## garage_exists                     9.419e+03  9.119e+03   1.033 0.302137
## kitchen_typeeat in               -4.888e+03  1.017e+04  -0.481 0.630826
## kitchen_typeefficiency           -2.185e+04  1.003e+04  -2.179 0.029829
## num_bedrooms                      5.092e+04  7.920e+03   6.430 3.03e-10
## num_floors_in_building            3.209e+03  7.318e+02   4.385 1.42e-05
## num_full_bathrooms               -3.149e+04  5.313e+04  -0.593 0.553620
## num_half_bathrooms                3.175e+03  3.219e+04   0.099 0.921471
## num_total_rooms                   1.480e+04  5.153e+03   2.871 0.004263
## parking_charges                   2.982e+02  9.827e+01   3.034 0.002537
## pct_tax_deductibl                -2.108e+02  9.659e+02  -0.218 0.827303
## sq_footage                        2.207e+01  1.288e+01   1.713 0.087404
## total_taxes                       7.707e+00  5.536e+00   1.392 0.164540
## walk_score                       -7.445e+02  3.931e+02  -1.894 0.058822
## lat                               5.826e+05  1.420e+05   4.104 4.75e-05
## lon                              -1.239e+05  8.992e+04  -1.378 0.168817
## pets_allowed                      1.448e+04  7.054e+03   2.052 0.040665
## monthly_cost                      1.288e+02  1.429e+01   9.013  < 2e-16
## price_persqft                     4.349e+05  6.241e+04   6.968 1.04e-11
## shortest_dist                    -5.476e+03  6.120e+03  -0.895 0.371319
## is_missing_approx_year_built     -1.284e+04  3.415e+04  -0.376 0.707044
## is_missing_community_district_num -1.732e+05  7.526e+04  -2.301 0.021787
## is_missing_dining_room_type       1.484e+04  7.974e+03   1.861 0.063348
## is_missing_kitchen_type           3.431e+04  2.944e+04   1.165 0.244450
## is_missing_num_bedrooms                  NA         NA      NA       NA
## is_missing_num_floors_in_building -1.305e+04  8.451e+03  -1.544 0.123123
## is_missing_num_half_bathrooms     1.579e+02  1.423e+04   0.011 0.991153
## is_missing_num_total_rooms                NA         NA      NA       NA
## is_missing_parking_charges         7.221e+03  7.862e+03   0.919 0.358797
## is_missing_pct_tax_deductibl      -6.287e+03  8.654e+03  -0.726 0.467908
## is_missing_sq_footage              7.290e+03  6.779e+03   1.075 0.282729
## is_missing_total_taxes            -6.671e+03  9.359e+03  -0.713 0.476313
## is_missing_monthly_cost           -1.910e+04  2.009e+04  -0.951 0.342214
## is_missing_price_persqft                  NA         NA      NA       NA
##
## (Intercept)                       ***
## 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                        .
## total_taxes
## walk_score                        .
## lat                               ***
## lon
## pets_allowed                      *
## monthly_cost                      ***
## price_persqft                     ***
## shortest_dist
## 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_total_taxes
## is_missing_monthly_cost
## is_missing_price_persqft
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 73000 on 491 degrees of freedom
##   (1702 observations deleted due to missingness)
## Multiple R-squared:  0.846,  Adjusted R-squared:  0.8347
## F-statistic:  74.9 on 36 and 491 DF,  p-value: < 2.2e-16
```

REMOVING MISSING Y SECTION

```
Data = Xnew
### sale price is our imputed Y
Y = Data$price
Data %<>%
  filter(!is.na(price)) %>%
  select(-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)
```

Dropping colinear features

```
Xtrain %<>%
  select(-c(is_missing_num_total_rooms, is_missing_num_bedrooms, is_missing_price_persqft))
```

Linear Regression

```
linear = lm(Ytrain ~ ., data = Xtrain)## simple linear model
summary(linear)
```

```
##
## Call:
## lm(formula = Ytrain ~ ., data = Xtrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -298745  -35010    1864   34077  344502
##
## Coefficients: (1 not defined because of singularities)
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      -3.772e+07  1.098e+07  -3.434 0.000658
## approx_year_built                 1.563e+02  2.755e+02   0.567 0.570968
## community_district_num            3.699e+03  1.227e+03   3.015 0.002742
## coop_condocondo                   1.844e+05  1.946e+04   9.478  < 2e-16
## dining_room_typedining area       2.711e+04  5.238e+04   0.518 0.605018
## dining_room_typeformal            2.990e+04  9.489e+03   3.151 0.001753
## dining_room_typeother             1.476e+04  1.241e+04   1.189 0.235031
## garage_exists                     9.091e+03  1.043e+04   0.871 0.384133
## kitchen_typeeat in                4.328e+03  1.125e+04   0.385 0.700557
## kitchen_typeefficiency           -1.895e+04  1.112e+04  -1.705 0.089084
## num_bedrooms                      4.185e+04  8.803e+03   4.754 2.82e-06
## num_floors_in_building            3.078e+03  8.085e+02   3.807 0.000164
## num_full_bathrooms              -2.523e+04  5.235e+04  -0.482 0.630070
## num_half_bathrooms              -7.751e+03  3.575e+04  -0.217 0.828459
## num_total_rooms                  1.805e+04  5.752e+03   3.138 0.001832
## parking_charges                  3.581e+02  1.032e+02   3.469 0.000581
## pct_tax_deductibl                6.141e+02  1.361e+03   0.451 0.652121
## sq_footage                       2.006e+01  1.332e+01   1.506 0.132818
## total_taxes                      4.295e+00  5.996e+00   0.716 0.474255
## walk_score                      -6.772e+02  4.342e+02  -1.560 0.119677
## lat                              6.370e+05  1.549e+05   4.111 4.81e-05
## lon                             -1.514e+05  1.009e+05  -1.501 0.134254
## pets_allowed                     9.384e+03  7.774e+03   1.207 0.228117
## monthly_cost                     1.574e+02  1.862e+01   8.452 5.93e-16
## price_persqft                    3.563e+05  6.818e+04   5.226 2.85e-07
## shortest_dist                   -1.090e+03  6.817e+03  -0.160 0.873029
## is_missing_approx_year_built    -3.826e+04  4.278e+04  -0.894 0.371726
## is_missing_community_district_num         NA         NA      NA       NA
## is_missing_dining_room_type      1.047e+04  8.986e+03   1.165 0.244809
## is_missing_kitchen_type          3.542e+04  3.314e+04   1.069 0.285939
## is_missing_num_floors_in_building -1.260e+04  9.043e+03  -1.393 0.164439
## is_missing_num_half_bathrooms    1.013e+04  1.543e+04   0.657 0.511777
## is_missing_parking_charges      -5.098e+02  8.769e+03  -0.058 0.953666
## is_missing_pct_tax_deductibl    -2.134e+03  9.433e+03  -0.226 0.821148
## is_missing_sq_footage            6.485e+03  7.406e+03   0.876 0.381762
## is_missing_total_taxes          -1.262e+04  1.017e+04  -1.241 0.215502
## is_missing_monthly_cost         -2.948e+04  2.381e+04  -1.238 0.216331
##
## (Intercept)                     ***
## 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
## total_taxes
## walk_score
## lat                             ***
## lon
## pets_allowed
## monthly_cost                    ***
## price_persqft                   ***
## shortest_dist
## is_missing_approx_year_built
## is_missing_community_district_num
## is_missing_dining_room_type
## is_missing_kitchen_type
## is_missing_num_floors_in_building
## is_missing_num_half_bathrooms
## is_missing_parking_charges
## is_missing_pct_tax_deductibl
## is_missing_sq_footage
## is_missing_total_taxes
## is_missing_monthly_cost
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71400 on 386 degrees of freedom
## Multiple R-squared:  0.8506, Adjusted R-squared:  0.837
## F-statistic: 62.79 on 35 and 386 DF,  p-value: < 2.2e-16
```

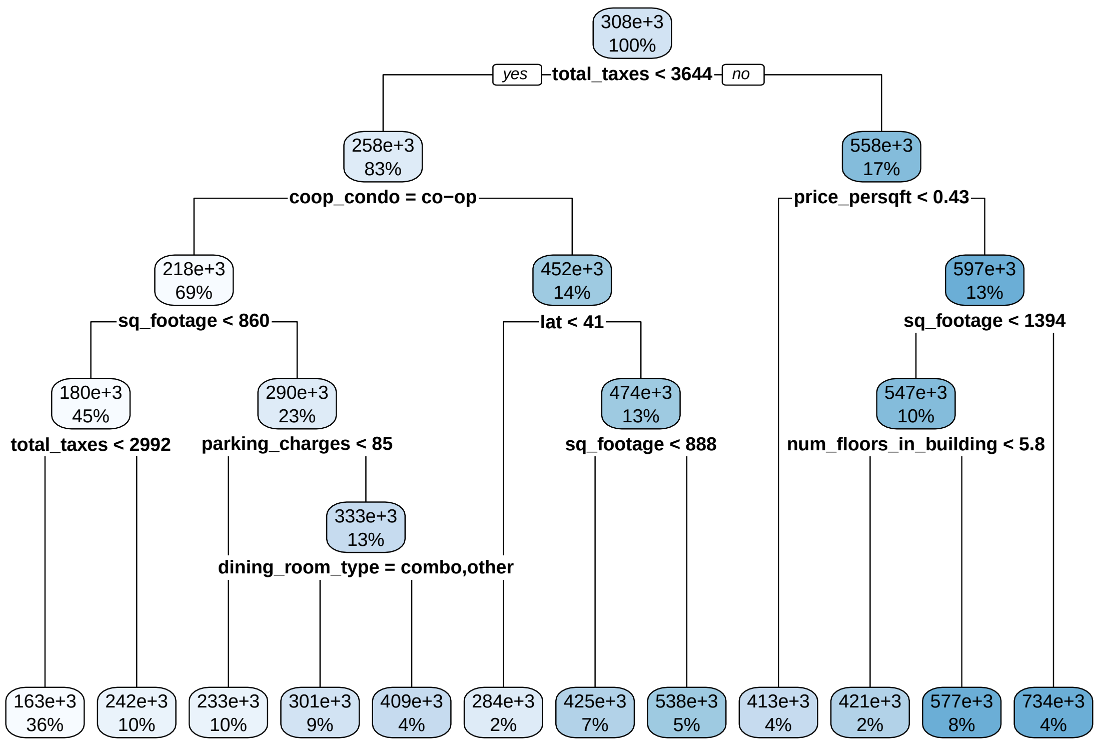
```
yhat = predict(linear, Xtest)
```

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

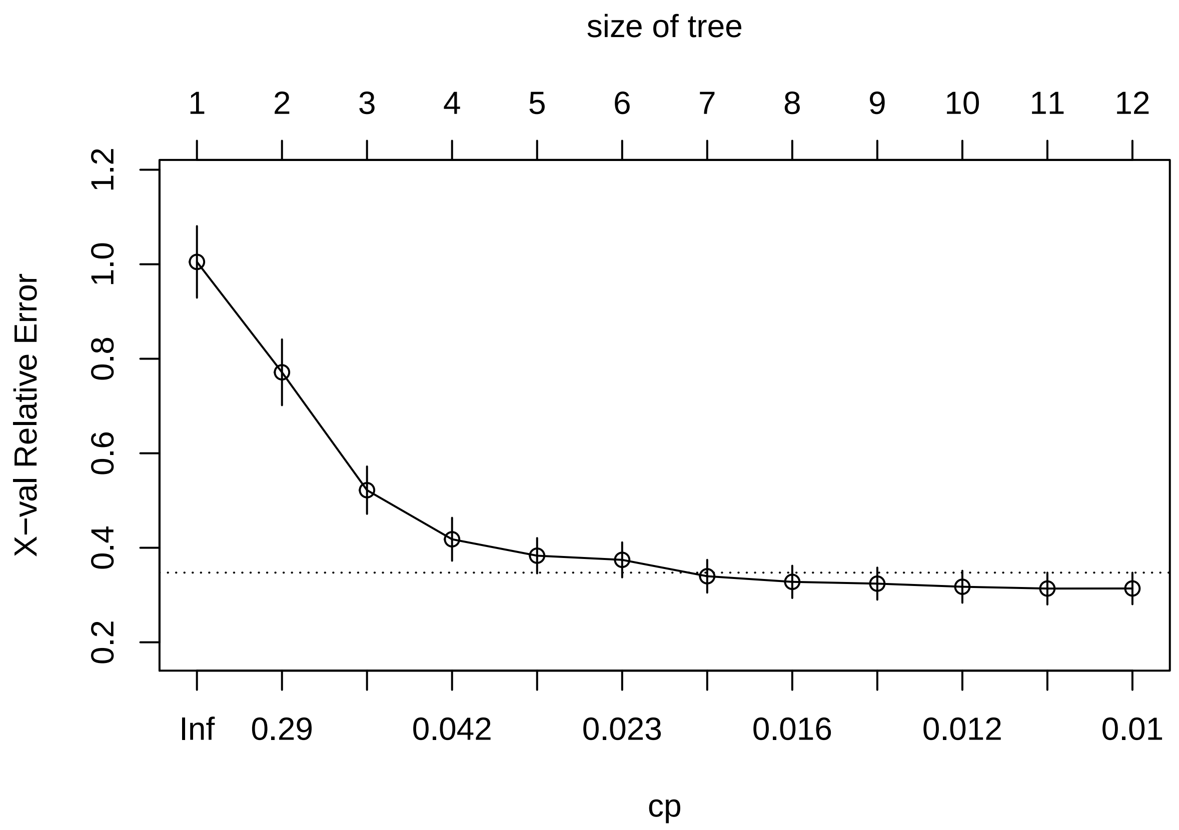
```
e = yhat - Ytest
sqrt(sum(e^2) / nrow(Xtest))
```

```
## [1] 85814.03
```

```
#REGRESSION TREE
pacman::p_load(rsample)#data spliting
pacman::p_load(rpart) #performing reg tree
pacman::p_load(rpart.plot) #ploting reg tree
pacman::p_load(ipred) #bagging
pacman::p_load(caret) #bagging
m1 = rpart(
  formula = Ytrain ~ .,
  data    = Xtrain,
  method  = "anova"
  )
rpart.plot(m1)
```


308e+3
100%
yes
total_taxes < 3644
no
258e+3
83%
coop_condo = co−op
558e+3
17%
price_persqft < 0.43
218e+3
69%
sq_footage < 860
452e+3
14%
lat < 41
597e+3
13%
sq_footage < 1394
180e+3
45%
total_taxes < 2992
290e+3
23%
parking_charges < 85
474e+3
13%
sq_footage < 888
547e+3
10%
num_floors_in_building < 5.8
333e+3
13%
dining_room_type = combo,other
163e+3
36%
242e+3
10%
233e+3
10%
301e+3
9%
409e+3
4%
284e+3
2%
425e+3
7%
538e+3
5%
413e+3
4%
421e+3
2%
577e+3
8%
734e+3
4%

```r
plotcp(m1)
```



```r
summary(m1)
```

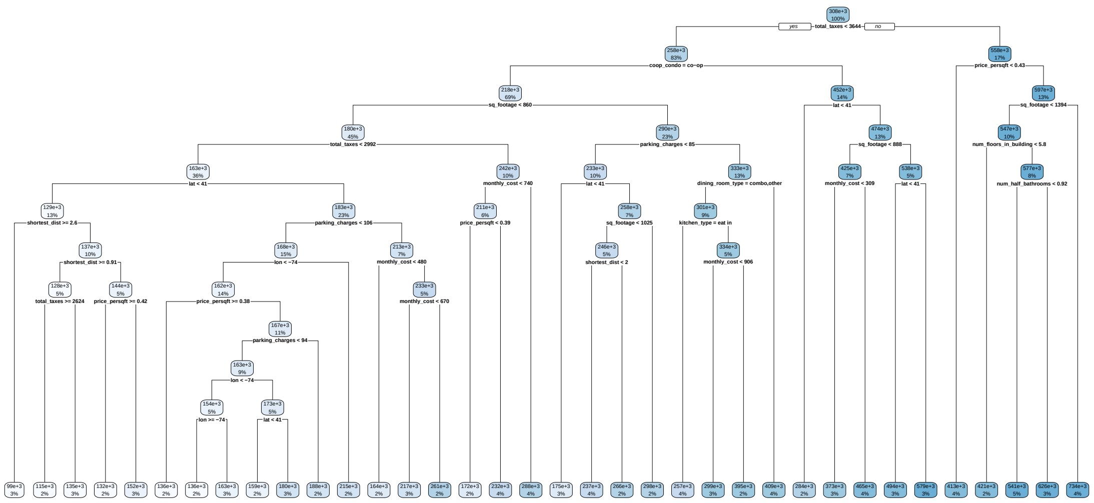
```
## Call:
## rpart(formula = Ytrain ~ ., data = Xtrain, method = "anova")
##   n= 422
##
##            CP nsplit rel error    xerror       xstd
## 1  0.40432061      0 1.0000000 1.0049289 0.07578280
## 2  0.20772469      1 0.5956794 0.7713344 0.06970579
## 3  0.05926418      2 0.3879547 0.5219011 0.05020183
## 4  0.03029997      3 0.3286905 0.4179873 0.04548378
## 5  0.02909995      4 0.2983905 0.3832397 0.03735022
## 6  0.01871160      5 0.2692906 0.3744220 0.03710430
## 7  0.01695005      6 0.2505790 0.3397632 0.03471459
## 8  0.01558743      7 0.2336289 0.3278590 0.03416186
## 9  0.01266841      8 0.2180415 0.3240925 0.03397891
## 10 0.01195052      9 0.2053731 0.3174986 0.03385444
## 11 0.01056997     10 0.1934226 0.3137086 0.03371290
## 12 0.01000000     11 0.1828526 0.3138752 0.03339476
##
## Variable importance
##           total_taxes              sq_footage            monthly_cost
##                    23                      15                      12
##         price_persqft       approx_year_built              coop_condo
##                    11                      10                      10
##        num_total_rooms           num_bedrooms        parking_charges
##                      5                      3                      2
##          shortest_dist     num_half_bathrooms             walk_score
##                      2                      2                      1
## num_floors_in_building                    lat                    lon
##                      1                      1                      1
##        dining_room_type
##                      1
##
## Node number 1: 422 observations,    complexity param=0.4043206
##   mean=308191.7, MSE=3.121006e+10
##   left son=2 (351 obs) right son=3 (71 obs)
##   Primary splits:
##       total_taxes       < 3644.032  to the left,  improve=0.4043206, (0 missing)
##       price_persqft     < 0.4294476 to the left,  improve=0.3775690, (0 missing)
##       coop_condo        splits as  LR, improve=0.3754617, (0 missing)
##       approx_year_built < 1970.5    to the left,  improve=0.3463094, (0 missing)
##       sq_footage        < 860.765   to the left,  improve=0.2937378, (0 missing)
##   Surrogate splits:
##       sq_footage        < 1231.15   to the left,  agree=0.912, adj=0.479, (0 split)
##       monthly_cost      < 1461.5    to the left,  agree=0.865, adj=0.197, (0 split)
##       price_persqft     < 0.6932388 to the left,  agree=0.865, adj=0.197, (0 split)
##       approx_year_built < 2008.5    to the left,  agree=0.853, adj=0.127, (0 split)
##       num_total_rooms   < 6.5       to the left,  agree=0.851, adj=0.113, (0 split)
##
## Node number 2: 351 observations,    complexity param=0.2077247
##   mean=257669.1, MSE=1.691212e+10
##   left son=4 (291 obs) right son=5 (60 obs)
##   Primary splits:
##       coop_condo        splits as  LR, improve=0.4608824, (0 missing)
##       price_persqft     < 0.4786456 to the left,  improve=0.3680961, (0 missing)
##       approx_year_built < 1974.5    to the left,  improve=0.3604524, (0 missing)
##       sq_footage        < 860.765   to the left,  improve=0.2263521, (0 missing)
##       monthly_cost      < 430.5     to the right, improve=0.1962185, (0 missing)
##   Surrogate splits:
##       approx_year_built < 1976.5    to the left,  agree=0.952, adj=0.717, (0 split)
##       monthly_cost      < 430.5     to the right, agree=0.932, adj=0.600, (0 split)
##       price_persqft     < 0.5107067 to the left,  agree=0.906, adj=0.450, (0 split)
##       total_taxes       < 1969.57   to the right, agree=0.858, adj=0.167, (0 split)
##       num_total_rooms   < 6.5       to the left,  agree=0.838, adj=0.050, (0 split)
##
## Node number 3: 71 observations,    complexity param=0.03029997
##   mean=557958.5, MSE=2.689197e+10
##   left son=6 (15 obs) right son=7 (56 obs)
##   Primary splits:
##       price_persqft          < 0.4274787 to the left,  improve=0.2090106, (0 missing)
##       monthly_cost           < 1478.5    to the left,  improve=0.1409474, (0 missing)
##       community_district_num < 27.5      to the left,  improve=0.1294134, (0 missing)
##       parking_charges        < 83.7      to the left,  improve=0.1240996, (0 missing)
##       num_bedrooms           < 2.5       to the left,  improve=0.1217794, (0 missing)
##   Surrogate splits:
##       parking_charges    < 83.7      to the left,  agree=0.915, adj=0.600, (0 split)
##       approx_year_built  < 1961      to the left,  agree=0.845, adj=0.267, (0 split)
##       num_half_bathrooms < 1.025     to the right, agree=0.831, adj=0.200, (0 split)
##       num_total_rooms    < 7.5       to the right, agree=0.817, adj=0.133, (0 split)
##       lat                < 40.69337  to the left,  agree=0.817, adj=0.133, (0 split)
##
## Node number 4: 291 observations,    complexity param=0.05926418
##   mean=217580.2, MSE=8.434212e+09
##   left son=8 (192 obs) right son=9 (99 obs)
##   Primary splits:
##       sq_footage      < 860.325   to the left,  improve=0.3180254, (0 missing)
##       monthly_cost    < 854.5     to the left,  improve=0.2689731, (0 missing)
##       total_taxes     < 2990.305  to the left,  improve=0.2511625, (0 missing)
##       num_bedrooms    < 1.5       to the left,  improve=0.2239259, (0 missing)
##       num_total_rooms < 4.5       to the left,  improve=0.1688384, (0 missing)
##   Surrogate splits:
##       num_bedrooms       < 1.5       to the left,  agree=0.890, adj=0.677, (0 split)
##       num_total_rooms    < 4.5       to the left,  agree=0.838, adj=0.525, (0 split)
##       monthly_cost       < 812       to the left,  agree=0.818, adj=0.465, (0 split)
##       num_half_bathrooms < 0.965     to the left,  agree=0.763, adj=0.303, (0 split)
##       total_taxes        < 3054.13   to the left,  agree=0.749, adj=0.263, (0 split)
##
## Node number 5: 60 observations,    complexity param=0.01695005
##   mean=452099.8, MSE=1.243214e+10
##   left son=10 (7 obs) right son=11 (53 obs)
##   Primary splits:
##       lat          < 40.70383  to the left,  improve=0.2992821, (0 missing)
##       total_taxes  < 2240      to the left,  improve=0.2224080, (0 missing)
##       sq_footage   < 887.815   to the left,  improve=0.2081316, (0 missing)
##       kitchen_type splits as  RRL-, improve=0.1716706, (0 missing)
##       num_bedrooms < 1.5       to the left,  improve=0.1537354, (0 missing)
##   Surrogate splits:
##       parking_charges    < 69.058    to the left,  agree=0.933, adj=0.429, (0 split)
##       price_persqft      < 0.4647468 to the left,  agree=0.933, adj=0.429, (0 split)
##       shortest_dist      < 2.508986  to the right, agree=0.933, adj=0.429, (0 split)
##       num_half_bathrooms < 1.105     to the right, agree=0.900, adj=0.143, (0 split)
##
## Node number 6: 15 observations
##   mean=413100, MSE=1.287201e+10
##
## Node number 7: 56 observations,    complexity param=0.02909995
##   mean=596759.8, MSE=2.352106e+10
##   left son=14 (41 obs) right son=15 (15 obs)
##   Primary splits:
##       sq_footage         < 1394.21   to the left,  improve=0.2909742, (0 missing)
##       num_bedrooms       < 1.5       to the left,  improve=0.2123309, (0 missing)
##       monthly_cost       < 1456      to the left,  improve=0.1946976, (0 missing)
##       num_total_rooms    < 4.5       to the left,  improve=0.1680623, (0 missing)
##       num_half_bathrooms < 0.91      to the left,  improve=0.1349023, (0 missing)
##   Surrogate splits:
##       walk_score    < 79        to the right, agree=0.839, adj=0.400, (0 split)
##       monthly_cost  < 1778      to the left,  agree=0.821, adj=0.333, (0 split)
##       shortest_dist < 1.272243  to the left,  agree=0.821, adj=0.333, (0 split)
##       lon           < -73.78052 to the left,  agree=0.804, adj=0.267, (0 split)
##       num_bedrooms  < 2.5       to the left,  agree=0.786, adj=0.200, (0 split)
##
## Node number 8: 192 observations,    complexity param=0.01558743
##   mean=180390.8, MSE=3.879615e+09
##   left son=16 (150 obs) right son=17 (42 obs)
##   Primary splits:
##       total_taxes     < 2991.555  to the left,  improve=0.2756080, (0 missing)
##       lat             < 40.71999  to the left,  improve=0.2696933, (0 missing)
##       price_persqft   < 0.3846746 to the left,  improve=0.1940024, (0 missing)
##       monthly_cost    < 859       to the left,  improve=0.1603974, (0 missing)
##       parking_charges < 132.8583  to the left,  improve=0.1531287, (0 missing)
##   Surrogate splits:
##       sq_footage    < 823.405   to the left,  agree=0.818, adj=0.167, (0 split)
##       walk_score    < 96.5      to the left,  agree=0.812, adj=0.143, (0 split)
##       monthly_cost  < 823       to the left,  agree=0.807, adj=0.119, (0 split)
##       shortest_dist < 0.2946642 to the right, agree=0.807, adj=0.119, (0 split)
##       num_bedrooms  < 2.5       to the left,  agree=0.792, adj=0.048, (0 split)
##
## Node number 9: 99 observations,    complexity param=0.0187116
##   mean=289705.3, MSE=9.383053e+09
##   left son=18 (43 obs) right son=19 (56 obs)
##   Primary splits:
##       parking_charges        < 84.89     to the left,  improve=0.2653009, (0 missing)
##       price_persqft          < 0.4136942 to the left,  improve=0.2601914, (0 missing)
##       num_floors_in_building < 8.71      to the left,  improve=0.2601534, (0 missing)
##       walk_score             < 91.5      to the left,  improve=0.2032182, (0 missing)
##       shortest_dist          < 0.7582295 to the right, improve=0.2015569, (0 missing)
##   Surrogate splits:
##       price_persqft          < 0.3633966 to the left,  agree=0.879, adj=0.721, (0 split)
##       shortest_dist          < 1.186745  to the right, agree=0.859, adj=0.674, (0 split)
##       walk_score             < 86.5      to the left,  agree=0.848, adj=0.651, (0 split)
##       lon                    < -73.81268 to the right, agree=0.788, adj=0.512, (0 split)
##       num_floors_in_building < 4.88      to the left,  agree=0.768, adj=0.465, (0 split)
##
## Node number 10: 7 observations
##   mean=284257.1, MSE=1.353748e+10
##
## Node number 11: 53 observations,    complexity param=0.01266841
##   mean=474267.7, MSE=8.074022e+09
##   left son=22 (30 obs) right son=23 (23 obs)
##   Primary splits:
##       sq_footage             < 887.815   to the left,  improve=0.3899091, (0 missing)
##       num_total_rooms        < 3.5       to the left,  improve=0.2876267, (0 missing)
##       total_taxes            < 2143      to the left,  improve=0.2498324, (0 missing)
##       community_district_num < 24.5      to the left,  improve=0.2151755, (0 missing)
##       num_bedrooms           < 1.5       to the left,  improve=0.2031233, (0 missing)
##   Surrogate splits:
##       num_total_rooms    < 4.5       to the left,  agree=0.849, adj=0.652, (0 split)
##       num_bedrooms       < 1.5       to the left,  agree=0.774, adj=0.478, (0 split)
##       num_half_bathrooms < 0.975     to the left,  agree=0.774, adj=0.478, (0 split)
##       price_persqft      < 0.5667094 to the right, agree=0.736, adj=0.391, (0 split)
##       parking_charges    < 98.885    to the right, agree=0.717, adj=0.348, (0 split)
##
## Node number 14: 41 observations,    complexity param=0.01195052
##   mean=546720.7, MSE=1.721348e+10
##   left son=28 (8 obs) right son=29 (33 obs)
##   Primary splits:
```

```
##       num_floors_in_building < 5.845     to the left,  improve=0.2230187, (0 missing)
##       num_bedrooms           < 1.5       to the left,  improve=0.1838297, (0 missing)
##       sq_footage             < 866       to the left,  improve=0.1394050, (0 missing)
##       num_half_bathrooms     < 0.87      to the left,  improve=0.1308696, (0 missing)
##       num_total_rooms        < 3.5       to the left,  improve=0.1052890, (0 missing)
##   Surrogate splits:
##       monthly_cost      < 203.5     to the left,  agree=0.854, adj=0.250, (0 split)
##       pct_tax_deductibl < 40.445    to the left,  agree=0.829, adj=0.125, (0 split)
##
## Node number 15: 15 observations
##   mean=733533.3, MSE=1.521078e+10
##
## Node number 16: 150 observations
##   mean=163087.8, MSE=2.283312e+09
##
## Node number 17: 42 observations
##   mean=242186.9, MSE=4.692681e+09
##
## Node number 18: 43 observations
##   mean=232767.4, MSE=3.356388e+09
##
## Node number 19: 56 observations,    complexity param=0.01056997
##   mean=333425.5, MSE=9.609887e+09
##   left son=38 (39 obs) right son=39 (17 obs)
##   Primary splits:
##       dining_room_type       splits as  L-R-L, improve=0.2586870, (0 missing)
##       monthly_cost           < 946       to the left,  improve=0.2437953, (0 missing)
##       num_floors_in_building < 8.658     to the left,  improve=0.2431465, (0 missing)
##       total_taxes            < 3022.85   to the left,  improve=0.1875480, (0 missing)
##       kitchen_type           splits as  RLR-, improve=0.1671011, (0 missing)
##   Surrogate splits:
##       monthly_cost           < 1112.5    to the left,  agree=0.786, adj=0.294, (0 split)
##       sq_footage             < 1081.9    to the left,  agree=0.768, adj=0.235, (0 split)
##       num_floors_in_building < 1.5       to the right, agree=0.750, adj=0.176, (0 split)
##       total_taxes            < 3458.067  to the left,  agree=0.750, adj=0.176, (0 split)
##       shortest_dist          < 0.432921  to the right, agree=0.750, adj=0.176, (0 split)
##
## Node number 22: 30 observations
##   mean=425139.6, MSE=5.490577e+09
##
## Node number 23: 23 observations
##   mean=538347.8, MSE=4.189336e+09
##
## Node number 28: 8 observations
##   mean=420881.2, MSE=1.8221e+10
##
## Node number 29: 33 observations
##   mean=577227.3, MSE=1.219965e+10
##
## Node number 38: 39 observations
##   mean=300507.1, MSE=6.035842e+09
##
## Node number 39: 17 observations
##   mean=408944.1, MSE=9.620147e+09
```

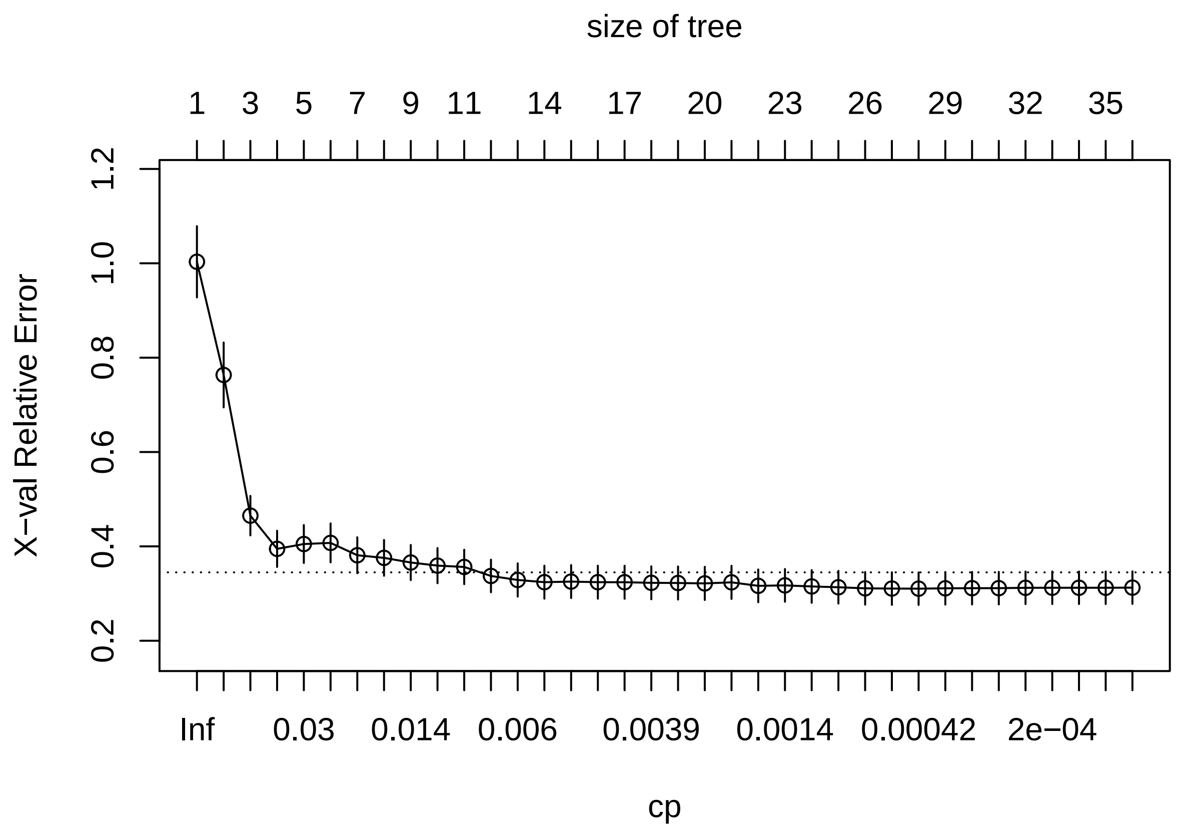
```r
yhat = predict(m1, Xtest)
e = yhat - Ytest
sqrt(sum(e^2)/106)
```

```
## [1] 115939.5
```

```r
m2 <- rpart(
    formula = Ytrain ~ .,
    data    = Xtrain,
    method  = "anova",
    control = list(cp = 0, xval = 10)
)
rpart.plot(m2)
```



```r
plotcp(m2)
```



```r
yhat = predict(m2, Xtest)
e = yhat - Ytest
sqrt(sum(e^2)/106)
```

```
## [1] 116058.3
```

```r
jpeg(file = "save_m2.jpeg")
```

```r
###Tuning
m3 <- rpart(
    formula = Ytrain ~ .,
    data    = Xtrain,
    method  = "anova",
    control = list(minsplit = 10, maxdepth = 12, xval = 10)
)
yhat = predict(m3, Xtest)
e = yhat - Ytest
sqrt(sum(e^2)/106)
```

```
## [1] 115939.5
```

```r
m3$cptable
```

```
##              CP nsplit rel error    xerror       xstd
## 1  0.40432061      0 1.0000000 1.0046626 0.07561982
## 2  0.20772469      1 0.5956794 0.6735084 0.05022728
## 3  0.05926418      2 0.3879547 0.4335064 0.03790875
## 4  0.03029997      3 0.3286905 0.3784264 0.03678666
## 5  0.02909995      4 0.2983905 0.3916900 0.04016449
## 6  0.01871160      5 0.2692906 0.3837787 0.03950903
## 7  0.01695005      6 0.2505790 0.3542221 0.03774597
## 8  0.01558743      7 0.2336289 0.3632621 0.03911695
## 9  0.01266841      8 0.2180415 0.3599131 0.03928099
## 10 0.01195052      9 0.2053731 0.3644700 0.04119890
## 11 0.01056997     10 0.1934226 0.3591691 0.04108622
## 12 0.01000000     11 0.1828526 0.3487342 0.04080870
```

```
# function to get optimal cp
get_cp <- function(x) {
  min    <- which.min(x$cptable[, "xerror"])
  cp <- x$cptable[min, "CP"]
}
# function to get minimum error
get_min_error <- function(x) {
  min    <- which.min(x$cptable[, "xerror"])
  xerror <- x$cptable[min, "xerror"]
}
```

```
optimal_tree <- rpart(
    formula = Ytrain ~ .,
    data    = Xtrain,
    method  = "anova",
    control = list(minsplit = 11, maxdepth = 8, cp = 0.01)
    )
pred <- predict(optimal_tree, newdata = Xtrain)
RMSE(pred = pred, obs = Ytrain)
```

```
## [1] 75543.64
```

##RANDOM FORESTS

```
m1 <- randomForest(
  formula = Ytrain ~ .,
  data    = Xtrain
)
m1
```

```
##
## Call:
##  randomForest(formula = Ytrain ~ ., data = Xtrain)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##           Mean of squared residuals: 4894874424
##                     % Var explained: 84.32
```

```r
which.min(m1$mse)
```

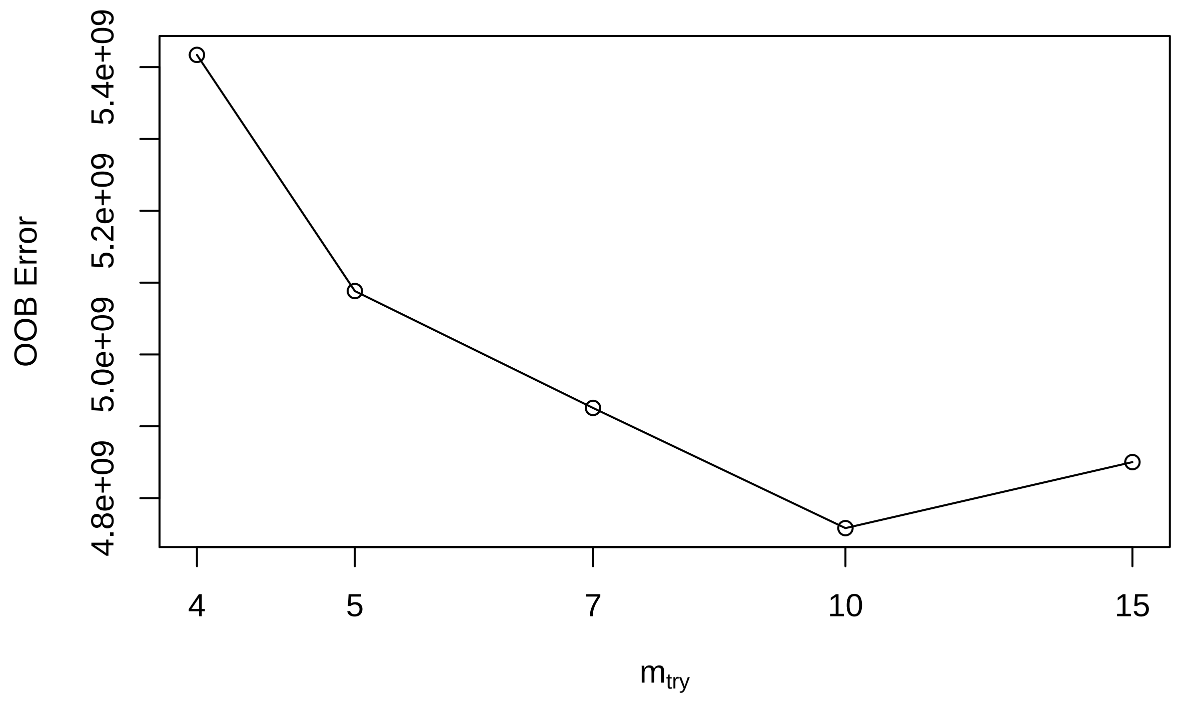
```
## [1] 365
```

```r
# RMSE of this optimal random forest
sqrt(m1$mse[which.min(m1$mse)])
```

```
## [1] 69782.39
```

```r
features <- setdiff(names(Xtrain), Ytrain)
set.seed(1989)
m2 <- tuneRF(
  x          = Xtrain,
  y          = Ytrain,
  ntreeTry   = 500,
  mtryStart  = 5,
  stepFactor = 1.5,
  improve    = 0.01,
  trace      = FALSE      # to not show real-time progress
)
```

```
## -0.06460655 0.01
## 0.03198041 0.01
## 0.03396325 0.01
## -0.01930881 0.01
```


OOB Error
4.8e+09
5.0e+09
5.2e+09
5.4e+09
m_{try}