

# Untitled

```
#import housing_data
housing_data = read.csv(file = '/Applications/MAT 342/housing_data.csv')

#load packages
pacman::p_load(tidyverse, mlr, mlr3, missForest, skimr, rpart, randomForest, data.table, dplyr, magrittr)

#picking certain data that does not apply in my opinion
housing_data_select = housing_data %>%
  select(approx_year_built, cats_allowed, coop_condo, dogs_allowed, dining_room_type, fuel_type,
         garage_exists, kitchen_type, maintenance_cost, num_bedrooms, num_floors_in_building,
         num_full_bathrooms, num_total_rooms, parking_charges, sale_price, sq_footage, total_taxes, walk_score)

setDT(housing_data_select)

dim(housing_data_select)

## [1] 2230 18

str(housing_data_select)

## Classes 'data.table' and 'data.frame': 2230 obs. of 18 variables:
## $ approx_year_built : int 1955 1955 2004 2002 1949 1938 1950 1960 1960 2005 ...
## $ cats_allowed : chr "no" "no" "no" "no" ...
## $ coop_condo : chr "co-op" "co-op" "condo" "condo" ...
## $ dogs_allowed : chr "no" "no" "no" "no" ...
## $ dining_room_type : chr "combo" "formal" "combo" "combo" ...
## $ fuel_type : chr "gas" "oil" NA "gas" ...
## $ garage_exists : chr NA NA NA NA ...
## $ kitchen_type : chr "eat in" "eat in" "efficiency" "eat in" ...
## $ maintenance_cost : chr NA "$604 " NA NA ...
## $ 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_total_rooms : int 5 4 3 5 4 4 3 2 4 3 ...
## $ parking_charges : chr NA NA NA NA ...
## $ sale_price : chr "$228,000 " "$235,500 " "$137,550 " "$545,000 " ...
## $ sq_footage : int NA 890 550 NA 675 1000 NA 375 NA 681 ...
## $ total_taxes : chr NA NA "$5,500 " "$2,260 " ...
## $ walk_score : int 82 89 90 94 71 90 72 93 70 98 ...
## - attr(*, ".internal.selfref")=<externalptr>

#removing any data with excessive NA
housing_data_drop = housing_data_select %>%
  select(-parking_charges, -sq_footage, -total_taxes, -num_floors_in_building)

#Adjusting the features so they can be used to run the algorithms
housing_data_new = housing_data_drop %>%
  mutate(cats_allowed = ifelse(cats_allowed == "yes", 1, 0)) %>% #set data to binary
  mutate(dogs_allowed = ifelse(dogs_allowed == "yes", 1, 0)) %>% #set data to binary
```

```

mutate(maintenance_cost = as.numeric(gsub('[$,]', '', housing_data_drop$maintenance_cost))) %>% #remo
mutate(sale_price = as.numeric(gsub('[$,]', '', housing_data_drop$sale_price))) %>% #remove $ and , f
mutate(coop_condo = factor(coop_condo, ordered = FALSE)) %>%
mutate(dining_room_type = factor(dining_room_type, ordered = FALSE)) %>%
mutate(fuel_type = factor(fuel_type, ordered = FALSE)) %>%
mutate(kitchen_type = factor(kitchen_type, ordered = FALSE)) %>%
mutate(garage_exists = ifelse(is.na(garage_exists), 0, 1)) #making sure NA is turned to 0

housing_data_new %>%
  filter(!is.na(sale_price))

```

```

##      approx_year_built cats_allowed coop_condo dogs_allowed dining_room_type
##  1:          1955          0      co-op          0          combo
##  2:          1955          0      co-op          0          formal
##  3:          2004          0      condo          0          combo
##  4:          2002          0      condo          0          combo
##  5:          1949          1      co-op          1          combo
##  ---
## 524:          1950          0      co-op          0          <NA>
## 525:          1947          0      co-op          0          formal
## 526:          2010          0      condo          0          combo
## 527:          2006          0      condo          0          combo
## 528:          1958          0      co-op          0          other
##      fuel_type garage_exists kitchen_type maintenance_cost num_bedrooms
##  1:      gas          0      eat in          NA          2
##  2:      oil          0      eat in          604          1
##  3:    <NA>          0  efficiency          NA          1
##  4:      gas          0      eat in          NA          3
##  5:      gas          0      eat in          660          2
##  ---
## 524:      gas          0      eat in          725          2
## 525:      gas          0      Combo          680          1
## 526:      gas          0      Eat In          NA          2
## 527: electric          0      Combo          NA          2
## 528:      other          0      eat in          659          2
##      num_full_bathrooms num_total_rooms sale_price walk_score
##  1:          1          5      228000          82
##  2:          1          4      235500          89
##  3:          1          3      137550          90
##  4:          2          5      545000          94
##  5:          1          4      241700          71
##  ---
## 524:          1          4      216000          83
## 525:          1          5      232500          94
## 526:          2          5      428000          96
## 527:          2          4      635000          99
## 528:          1          4      310000          96

```

```
missing_data = tbl_df(apply(is.na(housing_data_new), 2, as.numeric))
```

```

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

```

```

colnames(missing_data) = paste("missing_data_", colnames(housing_data_new), sep = "")
missing_data %<>%

```

```

select_if(function(x){sum(x) > 0})
housing_imp = missForest(data.frame(housing_data_new))$ximp

## missForest iteration 1 in progress...done!
## missForest iteration 2 in progress...done!
## missForest iteration 3 in progress...done!
## missForest iteration 4 in progress...done!
## missForest iteration 5 in progress...done!
## missForest iteration 6 in progress...done!
## missForest iteration 7 in progress...done!
## missForest iteration 8 in progress...done!

housing = cbind(housing_imp,missing_data)

#making train and test split
test_prop = 0.1

#test
test_indices = sample(1:nrow(housing), round((test_prop)*nrow(housing)))
housing_test = housing[test_indices,]
y_test = housing_test$sale_price
X_test = cbind(1, housing_test)
X_test$sale_price = NULL

#train
train_indices = setdiff(1:nrow(housing), test_indices)
housing_train = housing[train_indices,]
y_train = housing_train$sale_price
X_train = cbind(1, housing_train)
X_train$sale_price = NULL
n_train = nrow(X_train)

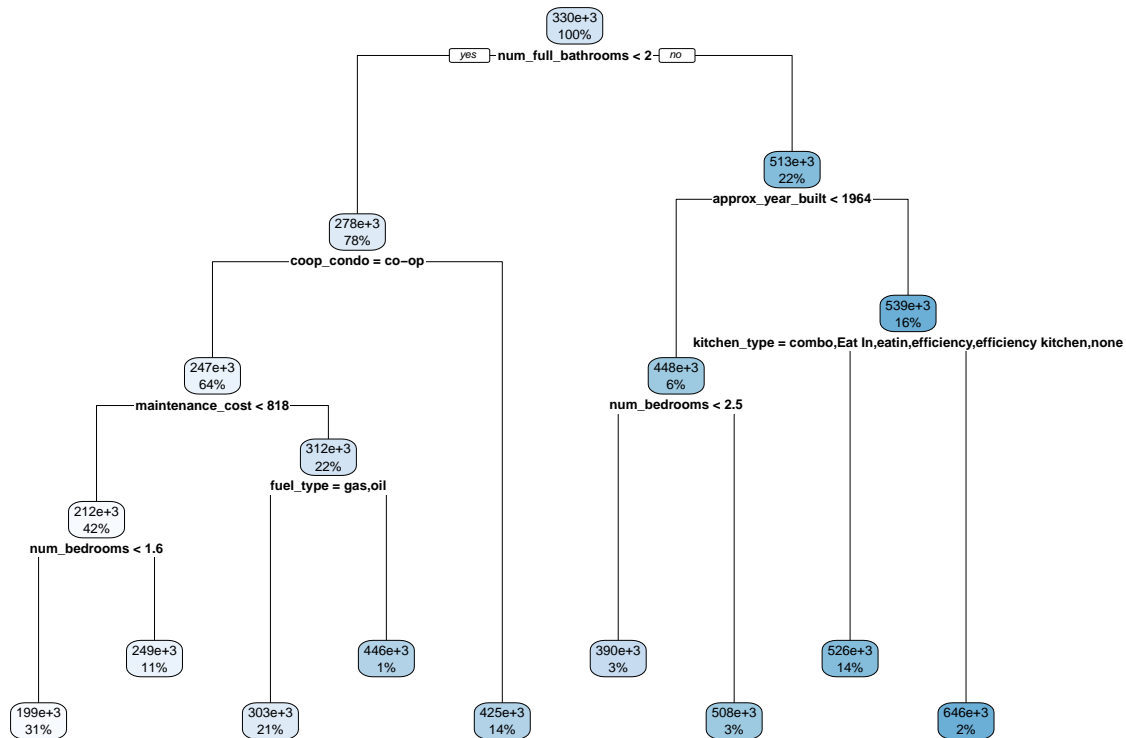
#Create Regression tree model

#use rpart, YARF not available for me
#In-sample Error
Reg_tree = rpart(y_train~., housing_train)
y_hat_train = predict(Reg_tree, housing_train)
e_in = y_train - y_hat_train
rsme_in = sd(e_in)
r_squared_in = (var(y_train)-var(e_in)) / var(y_train)

#OOSE
y_hat_test = predict(Reg_tree, housing_test)
e_oose = y_test - y_hat_test
rsme_oose = sd(e_oose)
rsquared_oose = (var(y_test) - var(e_oose)) / var(y_test)

fit_model = rpart(housing_train$sale_price~., data.frame(X_train), method="anova")
rpart.plot(fit_model)

```



fit\_model

```

## n= 2007
##
## node), split, n, deviance, yval
##      * denotes terminal node
##
## 1) root 2007 4.005688e+13 329560.0
##    2) num_full_bathrooms< 1.5 1569 1.687576e+13 278308.6
##      4) coop_condo=co-op 1290 7.481944e+12 246679.2
##        8) maintenance_cost< 818.4143 845 2.243681e+12 212130.7
##          16) num_bedrooms< 1.578333 618 1.087800e+12 198749.1 *
##          17) num_bedrooms>=1.578333 227 7.439367e+11 248561.8 *
##          9) maintenance_cost>=818.4143 445 2.314478e+12 312282.6
##            18) fuel_type=gas,oil 415 1.556165e+12 302582.1 *
##            19) fuel_type=electric,other,Other 30 1.790469e+11 446473.3 *
##        5) coop_condo=condo 279 2.136293e+12 424551.6 *
##    3) num_full_bathrooms>=1.5 438 4.296514e+12 513152.4
##      6) approx_year_built< 1963.5 126 1.268307e+12 448110.9
##        12) num_bedrooms< 2.5 64 2.590364e+11 390148.5 *
##        13) num_bedrooms>=2.5 62 5.723010e+11 507943.0 *
##      7) approx_year_built>=1963.5 312 2.279914e+12 539419.2
##        14) kitchen_type=combo,Eat In,eatin,efficiency,efficiency kitchen,none 276 1.219770e+12 525566
##        15) kitchen_type=Combo,eat in 36 6.010933e+11 645627.0 *

```

#mlr attempt

```

mod_task = makeRegrTask(data = data.frame(X_train), target = 'housing_data$sale_price')
algor = makeLearner("regr.rpart")
valid = makeResampleDesc("CV", iteration = 5)
resample = resample(algor, mod_task, valid, measures = list(rmse))
resample

```

```
mean(resample$measures.test$rmse)
```

```
#Random Forest
```

```
random_forest = randomForest(sale_price~.,housing_imp)  
random_forest
```

```
##
```

```
## Call:
```

```
## randomForest(formula = sale_price ~ ., data = housing_imp)
```

```
##           Type of random forest: regression
```

```
##           Number of trees: 500
```

```
## No. of variables tried at each split: 4
```

```
##
```

```
##           Mean of squared residuals: 1579484795
```

```
##           % Var explained: 92.08
```