Term Project 390.4- 2019

## R Markdown

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

##Delete variables that we dont need

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

## Clean Data

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

##Change variable type

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

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

#Added latitude and longitude features using ggmap

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

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

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

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

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

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

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

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

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

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

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

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

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

## missForest iteration 1 in progress...

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

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

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

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

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

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

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

## done!

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

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

### REMOVING MISSING Y SECTION

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

Linear Regression

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

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

yhat = predict(linear, Xtest)

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

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

## [1] 83540.97

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

## YARF can now make use of 3 cores.

reg\_tree = YARFCART(Xtrain, Ytrain)

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

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

## [1] 126998.6

sd(e)

## [1] 127303.4

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

Make test, train and selection sets

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

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

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

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

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

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

## [1] 86456.37

sd(e)

## [1] 85004.18