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)  
#register\_google(key = 'AIzaSyAA9F9nKHHRtmc2shoe4OHx24rFS4ZTjDA')  
#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!  
## 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 want  
## 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 want  
## to do regression?

## done!  
## missForest iteration 7 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 8 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 9 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   
## -319736 -41187 2393 38019 343305   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -5.520e+07 9.778e+06 -5.646 2.79e-08  
## total\_taxes -3.465e+00 5.931e+00 -0.584 0.55939  
## approx\_year\_built -3.845e+01 2.642e+02 -0.146 0.88435  
## community\_district\_num 3.732e+03 1.217e+03 3.067 0.00228  
## coop\_condocondo 2.020e+05 1.546e+04 13.066 < 2e-16  
## dining\_room\_typedining area 2.274e+04 5.603e+04 0.406 0.68510  
## dining\_room\_typeformal 2.251e+04 8.957e+03 2.513 0.01231  
## dining\_room\_typeother 1.889e+04 1.194e+04 1.582 0.11439  
## garage\_exists 1.629e+04 9.500e+03 1.715 0.08702  
## kitchen\_typeeat in -9.171e+03 1.063e+04 -0.863 0.38870  
## kitchen\_typeefficiency -2.467e+04 1.047e+04 -2.356 0.01885  
## num\_bedrooms 4.600e+04 8.389e+03 5.483 6.69e-08  
## num\_floors\_in\_building 3.007e+03 7.581e+02 3.966 8.39e-05  
## num\_full\_bathrooms 1.641e+04 5.663e+04 0.290 0.77216  
## num\_half\_bathrooms -5.903e+03 2.665e+04 -0.222 0.82477  
## num\_total\_rooms 1.829e+04 5.531e+03 3.306 0.00101  
## parking\_charges 2.707e+02 1.111e+02 2.436 0.01519  
## pct\_tax\_deductibl -5.856e+02 1.166e+03 -0.502 0.61559  
## sq\_footage 1.822e+01 1.341e+01 1.359 0.17488  
## walk\_score -6.446e+02 4.124e+02 -1.563 0.11873  
## lat 8.584e+05 1.392e+05 6.167 1.45e-09  
## lon -2.722e+05 9.162e+04 -2.971 0.00311  
## pets\_allowed 1.271e+04 7.345e+03 1.731 0.08415  
## monthly\_cost 1.411e+02 1.494e+01 9.443 < 2e-16  
## price\_persqft 3.572e+05 6.721e+04 5.315 1.63e-07  
## shortest\_dist -2.626e+02 6.401e+03 -0.041 0.96730  
## is\_missing\_total\_taxes -6.179e+03 1.009e+04 -0.612 0.54050  
## is\_missing\_approx\_year\_built NA NA NA NA  
## is\_missing\_community\_district\_num -3.347e+04 3.952e+04 -0.847 0.39744  
## is\_missing\_dining\_room\_type 1.916e+03 8.588e+03 0.223 0.82357  
## is\_missing\_kitchen\_type -2.173e+04 5.536e+04 -0.393 0.69484  
## is\_missing\_num\_bedrooms -2.248e+04 1.731e+04 -1.298 0.19484  
## is\_missing\_num\_floors\_in\_building 4.066e+03 8.030e+03 0.506 0.61284  
## is\_missing\_num\_half\_bathrooms -1.227e+04 1.346e+04 -0.911 0.36270  
## is\_missing\_num\_total\_rooms NA NA NA NA  
## is\_missing\_parking\_charges 7.397e+03 8.017e+03 0.923 0.35669  
## is\_missing\_pct\_tax\_deductibl -1.118e+04 8.462e+03 -1.321 0.18707  
## is\_missing\_sq\_footage 3.392e+03 1.142e+04 0.297 0.76650  
## is\_missing\_monthly\_cost -3.886e+04 1.798e+04 -2.161 0.03115  
## is\_missing\_price\_persqft -6.752e+03 1.184e+04 -0.570 0.56861  
##   
## (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: 76120 on 490 degrees of freedom  
## (1702 observations deleted due to missingness)  
## Multiple R-squared: 0.8328, Adjusted R-squared: 0.8202   
## F-statistic: 65.98 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   
## -325030 -34975 2279 35937 368341   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -6.227e+07 1.101e+07 -5.657 3.02e-08  
## total\_taxes -9.545e+00 6.374e+00 -1.497 0.135104  
## approx\_year\_built -1.225e+02 2.885e+02 -0.425 0.671398  
## community\_district\_num 3.720e+03 1.296e+03 2.870 0.004326  
## coop\_condocondo 2.301e+05 1.765e+04 13.041 < 2e-16  
## dining\_room\_typedining area 1.336e+04 5.517e+04 0.242 0.808806  
## dining\_room\_typeformal 2.215e+04 9.893e+03 2.239 0.025753  
## dining\_room\_typeother 1.469e+04 1.306e+04 1.125 0.261376  
## garage\_exists 1.561e+04 1.091e+04 1.431 0.153365  
## kitchen\_typeeat in 2.080e+03 1.177e+04 0.177 0.859882  
## kitchen\_typeefficiency -2.083e+04 1.169e+04 -1.781 0.075626  
## num\_bedrooms 3.573e+04 9.364e+03 3.816 0.000158  
## num\_floors\_in\_building 3.029e+03 8.498e+02 3.565 0.000410  
## num\_full\_bathrooms 1.104e+04 5.612e+04 0.197 0.844118  
## num\_half\_bathrooms -3.575e+04 3.165e+04 -1.130 0.259368  
## num\_total\_rooms 2.304e+04 6.186e+03 3.725 0.000225  
## parking\_charges 3.139e+02 1.175e+02 2.672 0.007862  
## pct\_tax\_deductibl 2.021e+03 1.657e+03 1.219 0.223412  
## sq\_footage 2.083e+01 1.392e+01 1.496 0.135346  
## walk\_score -5.591e+02 4.538e+02 -1.232 0.218714  
## lat 9.347e+05 1.528e+05 6.117 2.36e-09  
## lon -3.268e+05 1.040e+05 -3.143 0.001801  
## pets\_allowed 8.204e+03 8.121e+03 1.010 0.312993  
## monthly\_cost 1.742e+02 1.990e+01 8.757 < 2e-16  
## price\_persqft 3.051e+05 7.389e+04 4.128 4.48e-05  
## shortest\_dist 3.787e+03 7.114e+03 0.532 0.594853  
## is\_missing\_total\_taxes -1.740e+04 1.122e+04 -1.552 0.121600  
## is\_missing\_approx\_year\_built NA NA NA NA  
## is\_missing\_community\_district\_num -1.285e+04 4.496e+04 -0.286 0.775090  
## is\_missing\_dining\_room\_type -5.183e+03 9.692e+03 -0.535 0.593122  
## is\_missing\_kitchen\_type -1.780e+04 5.454e+04 -0.326 0.744392  
## is\_missing\_num\_bedrooms -1.362e+03 2.118e+04 -0.064 0.948766  
## is\_missing\_num\_floors\_in\_building 9.633e+03 8.805e+03 1.094 0.274641  
## is\_missing\_num\_half\_bathrooms -9.413e+03 1.495e+04 -0.630 0.529347  
## is\_missing\_num\_total\_rooms NA NA NA NA  
## is\_missing\_parking\_charges 2.543e+03 8.741e+03 0.291 0.771278  
## is\_missing\_pct\_tax\_deductibl -1.792e+04 9.254e+03 -1.936 0.053587  
## is\_missing\_sq\_footage 1.254e+03 1.272e+04 0.099 0.921517  
## is\_missing\_monthly\_cost -2.266e+04 1.870e+04 -1.212 0.226288  
## is\_missing\_price\_persqft -7.440e+03 1.308e+04 -0.569 0.569765  
##   
## (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: 74490 on 384 degrees of freedom  
## Multiple R-squared: 0.8382, Adjusted R-squared: 0.8227   
## F-statistic: 53.78 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] 89529.95

#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] 108793.3

sd(e)

## [1] 108524.9

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] 87784.45

sd(e)

## [1] 86770.77