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

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 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"))  
  
## Garage exists conver it 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)) %>%  
 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

#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 )  
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 %<>%  
 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   
## -248278 -38608 -127 39117 333286   
##   
## Coefficients: (3 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -5.201e+07 9.181e+06 -5.665 2.51e-08  
## total\_taxes 5.783e+00 5.728e+00 1.010 0.313111  
## approx\_year\_built -2.022e+02 2.588e+02 -0.781 0.434889  
## community\_district\_num 4.286e+03 1.191e+03 3.600 0.000351  
## coop\_condocondo 1.943e+05 1.534e+04 12.672 < 2e-16  
## dining\_room\_typedining area 1.683e+04 5.466e+04 0.308 0.758314  
## dining\_room\_typeformal 2.882e+04 8.967e+03 3.214 0.001396  
## dining\_room\_typeother 1.167e+04 1.153e+04 1.012 0.312075  
## garage\_exists 1.102e+04 9.275e+03 1.189 0.235129  
## kitchen\_typeeat in -7.375e+03 1.041e+04 -0.708 0.479139  
## kitchen\_typeefficiency -2.647e+04 1.013e+04 -2.613 0.009244  
## num\_bedrooms 5.304e+04 8.256e+03 6.424 3.14e-10  
## num\_floors\_in\_building 2.890e+03 7.479e+02 3.864 0.000126  
## num\_full\_bathrooms 1.883e+04 5.468e+04 0.344 0.730729  
## num\_half\_bathrooms -4.386e+04 3.690e+04 -1.189 0.235110  
## num\_total\_rooms 1.417e+04 5.263e+03 2.693 0.007323  
## parking\_charges 2.870e+02 1.013e+02 2.833 0.004807  
## pct\_tax\_deductibl 1.075e+03 9.316e+02 1.153 0.249324  
## sq\_footage 2.139e+01 1.330e+01 1.608 0.108387  
## walk\_score -5.519e+02 3.487e+02 -1.583 0.114108  
## lat 8.120e+05 1.387e+05 5.853 8.85e-09  
## lon -2.567e+05 8.594e+04 -2.987 0.002958  
## pets\_allowed 9.979e+03 7.091e+03 1.407 0.159973  
## monthly\_cost 1.316e+02 1.475e+01 8.920 < 2e-16  
## price\_persqft 4.215e+05 6.426e+04 6.560 1.37e-10  
## is\_missing\_total\_taxes 2.823e+02 9.484e+03 0.030 0.976266  
## is\_missing\_approx\_year\_built NA NA NA NA  
## is\_missing\_community\_district\_num -5.206e+03 4.377e+04 -0.119 0.905379  
## is\_missing\_dining\_room\_type 2.017e+04 8.682e+03 2.324 0.020543  
## is\_missing\_kitchen\_type -9.046e+04 4.457e+04 -2.030 0.042945  
## is\_missing\_num\_bedrooms -1.658e+04 1.698e+04 -0.977 0.329259  
## is\_missing\_num\_floors\_in\_building 7.295e+03 7.570e+03 0.964 0.335659  
## is\_missing\_num\_half\_bathrooms -1.726e+04 1.356e+04 -1.273 0.203698  
## is\_missing\_num\_total\_rooms NA NA NA NA  
## is\_missing\_parking\_charges 9.349e+03 7.808e+03 1.197 0.231750  
## is\_missing\_pct\_tax\_deductibl 6.183e+02 8.674e+03 0.071 0.943208  
## is\_missing\_sq\_footage 6.885e+03 1.159e+04 0.594 0.552676  
## is\_missing\_lat NA NA NA NA  
## is\_missing\_monthly\_cost 1.296e+04 1.821e+04 0.712 0.476956  
## is\_missing\_price\_persqft -5.842e+03 1.183e+04 -0.494 0.621531  
##   
## (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 \*\*\*  
## 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\_lat   
## is\_missing\_monthly\_cost   
## is\_missing\_price\_persqft   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 74550 on 491 degrees of freedom  
## (1702 observations deleted due to missingness)  
## Multiple R-squared: 0.8393, Adjusted R-squared: 0.8276   
## F-statistic: 71.26 on 36 and 491 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   
## -251218 -35085 3772 33806 327502   
##   
## Coefficients: (3 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -5.064e+07 1.005e+07 -5.041 7.14e-07  
## total\_taxes 1.215e+00 6.154e+00 0.197 0.843614  
## approx\_year\_built -2.319e+02 2.808e+02 -0.826 0.409389  
## community\_district\_num 3.698e+03 1.256e+03 2.944 0.003438  
## coop\_condocondo 2.139e+05 1.738e+04 12.310 < 2e-16  
## dining\_room\_typedining area 2.047e+04 5.319e+04 0.385 0.700604  
## dining\_room\_typeformal 2.597e+04 9.753e+03 2.663 0.008071  
## dining\_room\_typeother 6.665e+03 1.237e+04 0.539 0.590242  
## garage\_exists 1.192e+04 1.048e+04 1.138 0.255904  
## kitchen\_typeeat in 2.375e+03 1.141e+04 0.208 0.835185  
## kitchen\_typeefficiency -2.348e+04 1.116e+04 -2.104 0.036040  
## num\_bedrooms 4.294e+04 9.108e+03 4.714 3.39e-06  
## num\_floors\_in\_building 2.739e+03 8.248e+02 3.321 0.000983  
## num\_full\_bathrooms 2.117e+04 5.335e+04 0.397 0.691694  
## num\_half\_bathrooms -3.269e+04 4.021e+04 -0.813 0.416814  
## num\_total\_rooms 1.739e+04 5.816e+03 2.989 0.002974  
## parking\_charges 3.227e+02 1.068e+02 3.022 0.002678  
## pct\_tax\_deductibl 2.028e+03 1.252e+03 1.620 0.106058  
## sq\_footage 1.940e+01 1.365e+01 1.421 0.156046  
## walk\_score -5.915e+02 3.849e+02 -1.537 0.125214  
## lat 8.109e+05 1.493e+05 5.432 9.90e-08  
## lon -2.391e+05 9.629e+04 -2.483 0.013447  
## pets\_allowed 6.532e+03 7.720e+03 0.846 0.397990  
## monthly\_cost 1.628e+02 1.896e+01 8.587 2.25e-16  
## price\_persqft 3.947e+05 7.211e+04 5.474 7.94e-08  
## is\_missing\_total\_taxes -1.144e+03 1.000e+04 -0.114 0.908994  
## is\_missing\_approx\_year\_built NA NA NA NA  
## is\_missing\_community\_district\_num -2.166e+03 4.236e+04 -0.051 0.959257  
## is\_missing\_dining\_room\_type 2.024e+04 9.681e+03 2.090 0.037233  
## is\_missing\_kitchen\_type -9.051e+04 4.345e+04 -2.083 0.037886  
## is\_missing\_num\_bedrooms -2.598e+04 1.853e+04 -1.402 0.161796  
## is\_missing\_num\_floors\_in\_building 3.825e+03 8.186e+03 0.467 0.640529  
## is\_missing\_num\_half\_bathrooms -1.202e+04 1.467e+04 -0.819 0.413125  
## is\_missing\_num\_total\_rooms NA NA NA NA  
## is\_missing\_parking\_charges 7.906e+03 8.487e+03 0.932 0.352122  
## is\_missing\_pct\_tax\_deductibl 1.237e+03 9.676e+03 0.128 0.898298  
## is\_missing\_sq\_footage 1.156e+04 1.273e+04 0.909 0.364106  
## is\_missing\_lat NA NA NA NA  
## is\_missing\_monthly\_cost -8.642e+02 2.067e+04 -0.042 0.966681  
## is\_missing\_price\_persqft -1.436e+04 1.293e+04 -1.110 0.267645  
##   
## (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 \*\*\*  
## 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\_lat   
## is\_missing\_monthly\_cost   
## is\_missing\_price\_persqft   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 71920 on 385 degrees of freedom  
## Multiple R-squared: 0.8488, Adjusted R-squared: 0.8347   
## F-statistic: 60.04 on 36 and 385 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] 87429.6

#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] 108649.1

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] 88400.89