# Homework #4

*Taiga Hasegawa(taigah2)*

*Mar 17, 2019*

**(Please read the Homework Policy before you start)**

Description: In this homework, you will gain firsthand experience of analyzing unstructured high frequency financial market data using (shallow) machine learning methods. Consider yourself as a quantitative analyst, and you are analyzing a futures dataset provided directly by a data vendor, which is AlgoSeek. Your goal is to provide models and features that are potentially useful for quantitative strategists / traders.

Dataset: Please download the tick data from Compass 2g, and you will be using E-mini SP500 data from the folders 201603 and 201604. Please note that the license of the data only allows you to do homework and projects for this course, and you should delete the data if you do not take the course.

Practices:

1. Preprocess the unstructured data so that you will have a labeled dataset of a sufficiently large sample size. You can label the price movement direction based on the volume weighted average prices (vwap) of each minute, and take advantage of the 10 levels limit order books to construct any potentially useful features. Some R functions from the fmlr package can be used, such as, fmlr::read_algoseek_futures_fullDepth(), fmlr::istar_CUSUM(), and fmlr::label_meta().

2. This is a relatively open assignment, and there are some necessary sub-tasks, including, but not limited to the following:

    - futures rollovers
    - randomforest models (sequential bootstrap is optional)
    - grid search and parameter tuning with purged k-fold cross validation with embargo

Your grade for this homework will be based on how much you follow the homework policy, the completeness of the practices, and whether relevant methods are appropriately applied.

```
  rm(list = setdiff(ls(), lsf.str()))
devtools::install_github("larryleihua/randomForestFML")
  library(randomForestFML)
  library(randomForest)
  library(fmlr)
  library(caret)
  library(ROCR)
  library(MLmetrics)

  data_is_loaded <- TRUE
  data_folder <- "201603"
```

```r
data_folder_out <- file.path(data_folder)
if(!file.exists(data_folder_out)) dir.create(data_folder_out)

data_range <- stringr::str_replace_all(
  as.character(as.Date(as.Date("2016-03-01"):as.Date("2016-03-31"),
                       origin = "1970-01-01")), "-", "")
pre_roll_date <- "20160310"
roll_date <- "20160311"

contracts <- rep("ES/ESH6", length(data_range))
contracts[as.numeric(data_range) >= as.numeric(roll_date)] <- "ES/ESM6"

files <- paste0(data_range,".zip")

######################################
# prepare 1 minute data and features #
######################################

# We should run it with R directly, not with Rmarkdown. The codes are included here for reference only
if(data_is_loaded == FALSE)
{
  sapply( (1:length(files)), function(i){

    tmp <- tryCatch(fmlr::read_algoseek_futures_fullDepth( file.path(data_folder, files[i]),
                                                          whichData = paste0(contracts[i], ".csv") ),
                    error=function(e) NA, warning=function(w) NA)

    if(!is.na(tmp)){
      tmp <- tmp[[1]]

      ########
      # SELL #
      ########
      data_sell <- subset(tmp, tmp$Side=="SELL")
      data_sell_smh <- aggregate(list(data_sell$p1*data_sell$v1,
                                      data_sell$p2*data_sell$v2,
                                      data_sell$p3*data_sell$v3,
                                      data_sell$p4*data_sell$v4,
```

```r
                                data_sell$p5*data_sell$v5,
                                data_sell$p6*data_sell$v6,
                                data_sell$p7*data_sell$v7,
                                data_sell$p8*data_sell$v8,
                                data_sell$p9*data_sell$v9,
                                data_sell$p10*data_sell$v10,
                                data_sell$v1, data_sell$v2, data_sell$v3, data_sell$v4,
                                data_sell$v5, data_sell$v6, data_sell$v7, data_sell$v8,
                                data_sell$v9, data_sell$v10),
                           by=list(data_sell$m, data_sell$h), sum)
names(data_sell_smh) <- c("m","h","p1","p2","p3","p4","p5","p6","p7","p8","p9","p10",
                          "v1","v2","v3","v4","v5","v6","v7","v8","v9","v10")
data_sell_smh$vwap1 <- data_sell_smh$p1 / data_sell_smh$v1
data_sell_smh$vwap2 <- data_sell_smh$p2 / data_sell_smh$v2
data_sell_smh$vwap3 <- data_sell_smh$p3 / data_sell_smh$v3
data_sell_smh$vwap4 <- data_sell_smh$p4 / data_sell_smh$v4
data_sell_smh$vwap5 <- data_sell_smh$p5 / data_sell_smh$v5
data_sell_smh$vwap6 <- data_sell_smh$p6 / data_sell_smh$v6
data_sell_smh$vwap7 <- data_sell_smh$p7 / data_sell_smh$v7
data_sell_smh$vwap8 <- data_sell_smh$p8 / data_sell_smh$v8
data_sell_smh$vwap9 <- data_sell_smh$p9 / data_sell_smh$v9
data_sell_smh$vwap10 <- data_sell_smh$p10 / data_sell_smh$v10

data_sell_smh <- data_sell_smh[, names(data_sell_smh)
                               %in% c("m","h", "vwap1","vwap2","vwap3","vwap4","vwap5",
                                      "vwap6","vwap7","vwap8","vwap9","vwap10")]

#######
# BUY #
#######
data_buy <- subset(tmp, Side=="BUY")
data_buy_smh <- aggregate(list(data_buy$p1*data_buy$v1,
                               data_buy$p2*data_buy$v2,
                               data_buy$p3*data_buy$v3,
                               data_buy$p4*data_buy$v4,
                               data_buy$p5*data_buy$v5,
                               data_buy$p6*data_buy$v6,
                               data_buy$p7*data_buy$v7,
```

```
                                data_buy$p8*data_buy$v8,
                                data_buy$p9*data_buy$v9,
                                data_buy$p10*data_buy$v10,
                                data_buy$v1, data_buy$v2, data_buy$v3, data_buy$v4,
                                data_buy$v5, data_buy$v6, data_buy$v7, data_buy$v8,
                                data_buy$v9, data_buy$v10),
                           by=list(data_buy$m, data_buy$h), sum)
names(data_buy_smh) <- c("m","h","p1","p2","p3","p4","p5","p6","p7","p8","p9","p10",
                         "v1","v2","v3","v4","v5","v6","v7","v8","v9","v10")
data_buy_smh$vwap1 <- data_buy_smh$p1 / data_buy_smh$v1
data_buy_smh$vwap2 <- data_buy_smh$p2 / data_buy_smh$v2
data_buy_smh$vwap3 <- data_buy_smh$p3 / data_buy_smh$v3
data_buy_smh$vwap4 <- data_buy_smh$p4 / data_buy_smh$v4
data_buy_smh$vwap5 <- data_buy_smh$p5 / data_buy_smh$v5
data_buy_smh$vwap6 <- data_buy_smh$p6 / data_buy_smh$v6
data_buy_smh$vwap7 <- data_buy_smh$p7 / data_buy_smh$v7
data_buy_smh$vwap8 <- data_buy_smh$p8 / data_buy_smh$v8
data_buy_smh$vwap9 <- data_buy_smh$p9 / data_buy_smh$v9
data_buy_smh$vwap10 <- data_buy_smh$p10 / data_buy_smh$v10

data_buy_smh <- data_buy_smh[, names(data_buy_smh)
                             %in% c("m","h", "vwap1","vwap2","vwap3","vwap4","vwap5",
                                    "vwap6","vwap7","vwap8","vwap9","vwap10")]

####################
# combine BUY/SELL #
####################
data_buy_sell <- merge(data_buy_smh, data_sell_smh, by=c("h","m"), all=T,
                       suffixes = c("_buy","_sell"), sort=FALSE)

reorder <- c("h","m",
             "vwap10_buy","vwap9_buy","vwap8_buy","vwap7_buy","vwap6_buy",
             "vwap5_buy","vwap4_buy","vwap3_buy","vwap2_buy","vwap1_buy",
             "vwap1_sell","vwap2_sell","vwap3_sell","vwap4_sell","vwap5_sell",
             "vwap6_sell","vwap7_sell","vwap8_sell","vwap9_sell","vwap10_sell")

data_buy_sell <- data_buy_sell[, sapply(reorder, function(x){which(x==names(data_buy_sell))})]
```

```r
      # impute NA by the previous non-NA
      data_buy_sell$vwap10_buy <- zoo::na.locf(data_buy_sell$vwap10_buy)
      data_buy_sell$vwap9_buy <- zoo::na.locf(data_buy_sell$vwap9_buy)
      data_buy_sell$vwap8_buy <- zoo::na.locf(data_buy_sell$vwap8_buy)
      data_buy_sell$vwap7_buy <- zoo::na.locf(data_buy_sell$vwap7_buy)
      data_buy_sell$vwap6_buy <- zoo::na.locf(data_buy_sell$vwap6_buy)
      data_buy_sell$vwap5_buy <- zoo::na.locf(data_buy_sell$vwap5_buy)
      data_buy_sell$vwap4_buy <- zoo::na.locf(data_buy_sell$vwap4_buy)
      data_buy_sell$vwap3_buy <- zoo::na.locf(data_buy_sell$vwap3_buy)
      data_buy_sell$vwap2_buy <- zoo::na.locf(data_buy_sell$vwap2_buy)
      data_buy_sell$vwap1_buy <- zoo::na.locf(data_buy_sell$vwap1_buy)

      data_buy_sell$vwap10_sell <- zoo::na.locf(data_buy_sell$vwap10_sell)
      data_buy_sell$vwap9_sell <- zoo::na.locf(data_buy_sell$vwap9_sell)
      data_buy_sell$vwap8_sell <- zoo::na.locf(data_buy_sell$vwap8_sell)
      data_buy_sell$vwap7_sell <- zoo::na.locf(data_buy_sell$vwap7_sell)
      data_buy_sell$vwap6_sell <- zoo::na.locf(data_buy_sell$vwap6_sell)
      data_buy_sell$vwap5_sell <- zoo::na.locf(data_buy_sell$vwap5_sell)
      data_buy_sell$vwap4_sell <- zoo::na.locf(data_buy_sell$vwap4_sell)
      data_buy_sell$vwap3_sell <- zoo::na.locf(data_buy_sell$vwap3_sell)
      data_buy_sell$vwap2_sell <- zoo::na.locf(data_buy_sell$vwap2_sell)
      data_buy_sell$vwap1_sell <- zoo::na.locf(data_buy_sell$vwap1_sell)

      rm(list=c("data_buy", "data_buy_smh", "data_sell", "data_sell_smh", "tmp")); gc()
      write.table(data_buy_sell, file = file.path(data_folder_out, paste0(data_range[i], "_M1.csv")), sep=",", row.names = F)
    }

  } )
}

####################
# Futures rollover #
####################

# load those data preprocessed from above
dat_roll <- read.csv( file.path( data_folder_out, paste0(roll_date, "_M1.csv")) )
dat_roll_pre <- read.csv( file.path( data_folder_out, paste0(pre_roll_date, "_M1.csv")))
nam <- names(dat_roll)
```

```
gap_roll <- (dat_roll$vwap1_buy[1] + dat_roll$vwap1_sell[1])/2 -
  (dat_roll_pre$vwap1_buy[nx <- nrow(dat_roll_pre)] + dat_roll_pre$vwap1_sell[nx])/2
files_out <- list.files(data_folder_out)
inx_add_gap <- as.numeric(stringr::str_sub(files_out, 1, 8)) < as.numeric(roll_date)
dat <- NULL
for(k in 1:length(files_out)){
  tmp <- read.csv( file.path(data_folder_out, files_out[k]) )
  if(inx_add_gap[k] == TRUE) tmp[,-c(1,2)] <- tmp[,-c(1,2)] + gap_roll
  dat <- rbind(dat, tmp)
}

#############################################
# Label data using mid prices based on vwap #
#############################################

# use mid price to label the data
# you can also use some other reasonable prices
dat$mid_price <- (dat$vwap1_buy + dat$vwap1_sell)/2
ndat <- nrow(dat)

tt_split <- 2:1 # train/test split
idx_train <- (1:floor(tt_split[1]/sum(tt_split)*ndat))
idx_test <- setdiff( (1:ndat), idx_train)
train_data <- dat[idx_train,]
test_data <- dat[idx_test,]
```

Now, we have the training set `train_data` and the test set `test_data`. In what follows, we can use the `mid_price` to label the data, and use purged K-fold CV, etc. to fit and evaluate the models.

```
###################################################
# vectors for two parameters to be tuned
hvec <- seq(0.2, 1, length=5)
trgtvec <- seq(0.001, 0.005, length=4)
k <- 5 # k-fold CV
gam <- 0.01 # embargo parameter
run <- FALSE # whether run the grid search? (We need to run it for 1 time only!!)
set.seed(2019)
###################################################
if(run==TRUE)
```

```r
{
  rst <- data.frame("ih"=rep(NA,100), "jtrgt"=rep(NA,100), "iCV"=rep(NA,100), "hCUSUM"=rep(NA,100),
                    "trgt"=rep(NA,100), "acc"=rep(NA,100), "auc"=rep(NA,100), "F1"=rep(NA,100))
  count=1
  for(ih in 1:length(hvec))
  {
    for(jtrgt in 1:length(trgtvec))
    {

      ##############################################
      # some data labeling and analysis work here #
      ##############################################
      i_CUSUM=fmlr::istar_CUSUM(dat$mid_price,h=ih)
      n_Event <- length(i_CUSUM)
      events <- data.frame(t0=i_CUSUM+1,
                           t1 = i_CUSUM+300,
                           trgt = rep(trgtvec[jtrgt], n_Event),
                           side=rep(0,n_Event))
      ptSl <- c(1,1)
      out0 <- fmlr::label_meta(dat$mid_price, events, ptSl, ex_vert = T)
      fMat0 <- dat[out0$t1Fea, !names(dat)%in%c("m","h")]
      fMat <- rbind( rep(NA, ncol(fMat0)),  apply(fMat0, 2, function(x){diff(x)} ) )
      allSet<- data.frame(Y=as.factor(out0$label), fMat)
      allSet[,"t1Fea"]=out0$t1Fea
      allSet[,"tLabel"]=out0$tLabel
      idx_NA <- apply(allSet,1,function(x){sum(is.na(x))>0 })
      allSet <- subset(allSet, !idx_NA)
      CVobj = fmlr::purged_k_CV(allSet)
      for(iCV in 1:5){
        testset=CVobj[[iCV]]$testSet
        trainset=CVobj[[iCV]]$trainSet
        #randomforest models
        mtry <- randomForestFML::tuneRF(trainset[,-1], trainset$Y, trace = FALSE, plot=FALSE)
        mtry <- mtry[which.min(mtry[,2]),1]
        fit_all <- randomForestFML(Y ~ ., data = trainset[,-c(22,23)], mtry = mtry, importance = TRUE,
                                   ntrees = 800)
        pre <- predict(fit_all, newdata = testset)
        acc <- mean(testset$Y==pre)
```

```
        precision <- posPredValue(pre, testset$Y, positive="1")
        recall <- sensitivity(pre, testset$Y, positive="1")
        F1 <- (2 * precision * recall) / (precision + recall)
        prob_test <- predict(fit_all, newdata=testset, type="prob")
        pred <- prediction(prob_test[,2],testset$Y)
        auc <- performance(pred, measure = "auc")@y.values[[1]]
        rst$ih[count]=ih
        rst$jtrgt[count]=jtrgt
        rst$iCV[count]=iCV
        rst$hCUSUM[count]=hvec[ih]
        rst$trgt[count]=trgtvec[jtrgt]
        rst$acc[count]=acc
        rst$auc[count]=auc
        rst$F1[count]=F1
        count=count+1
      }
    } # end of jtrgt loop
  } # end of ih loop

  rst <- data.frame(rst)
  names(rst) <- c("ih", "jtrgt", "iCV", "hCUSUM", "trgt", "acc", "auc", "F1")

  # the result was saved so we don't have to run the analysis again when we knit
  # the Rmarkdown
  write.csv(rst, "rst.csv", row.names = F)

}

########################
# Organize the results #
########################
perfCV <- read.csv("rst.csv", header = T)

# remove those records of which either acc, auc, or F1 are not available
perfCV <- subset(perfCV, (!is.na(acc))&(!is.na(auc))&(!is.na(F1)))

cnt <- aggregate(perfCV$acc, by=list(perfCV$hCUSUM, perfCV$trgt), FUN=length)
acc <- aggregate(perfCV$acc, by=list(perfCV$hCUSUM, perfCV$trgt), FUN=mean)
```

```r
auc <- aggregate(perfCV$auc, by=list(perfCV$hCUSUM, perfCV$trgt), FUN=mean)
f1 <- aggregate(perfCV$F1, by=list(perfCV$hCUSUM, perfCV$trgt), FUN=mean)

# disable warning
options(warn=-1)

# combine results by merging multiple data.frame together
mer <- Reduce(function(...) merge(..., by=c("Group.1","Group.2")),
              list(cnt, acc, auc, f1))
names(mer) <- c("hCUSUM", "trgt", "kCV", "acc", "auc", "f1")

# rule out those the k-fold CV hasn't been successfully conducted
mer <- subset(mer, kCV==5)

####################
# Rank the results #
####################

options(digits = 3)
# rank by f1 scores
rstF1 <- mer[order(mer$f1, decreasing=T),]
rstF1
```

```
##    hCUSUM    trgt kCV   acc   auc    f1
## 12    0.6 0.00500   5 0.630 0.596 0.690
## 6     0.4 0.00233   5 0.554 0.529 0.661
## 16    0.8 0.00500   5 0.570 0.583 0.632
## 20    1.0 0.00500   5 0.553 0.577 0.628
## 13    0.8 0.00100   5 0.516 0.516 0.582
## 5     0.4 0.00100   5 0.533 0.540 0.579
## 14    0.8 0.00233   5 0.515 0.557 0.557
## 11    0.6 0.00367   5 0.473 0.557 0.544
## 1     0.2 0.00100   5 0.521 0.516 0.541
## 9     0.6 0.00100   5 0.521 0.553 0.517
## 18    1.0 0.00233   5 0.470 0.512 0.517
## 17    1.0 0.00100   5 0.461 0.450 0.467
## 10    0.6 0.00233   5 0.464 0.515 0.450
```

```r
########################################################################
# select candidate model based on logloss and evaluate with test_data #
########################################################################
h_selected <- 0.6
trgt_selected <- 0.005 # based on f1

# re-run the model with all train_data
i_CUSUM <- fmlr::istar_CUSUM(train_data$mid_price, h=h_selected)
n_Event <- length(i_CUSUM)
events <- data.frame(t0 = i_CUSUM+1,
                     t1 = i_CUSUM+300,
                     trgt = rep(trgt_selected, n_Event),
                     side = rep(0,n_Event))
ptSl <- c(1,1)
out0 <- fmlr::label_meta(train_data$mid_price, events, ptSl, ex_vert = T)
table(out0$label)
```

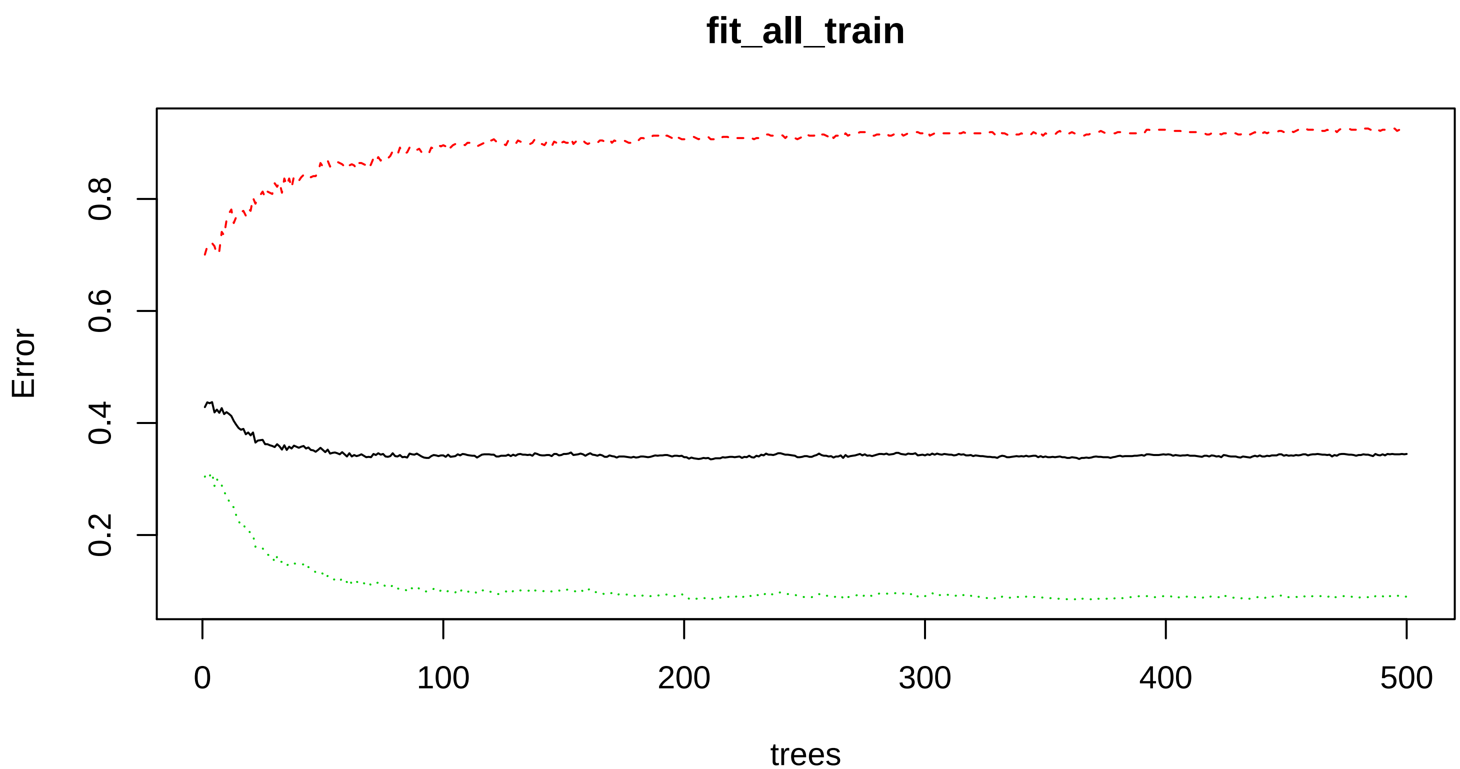
```
##
##   -1    1
##  472 1078
```

```r
fMat0 <- train_data[out0$t1Fea, !names(train_data)%in%c("m","h")]
fMat <- rbind( rep(NA, ncol(fMat0)),  apply(fMat0, 2,  function(x){diff(x)} ) )
allSet_train <- data.frame(Y=as.factor(out0$label), fMat)
idx_NA <- apply(allSet_train,1,function(x){sum(is.na(x))>0 })
allSet_train <- subset(allSet_train, !idx_NA)

mtry <- randomForestFML::tuneRF(allSet_train[,-1], allSet_train$Y, trace = FALSE, plot=FALSE)
```

```
## 0.00896 0.05
## -0.0179 0.05
```

```r
mtry <- mtry[which.min(mtry[,2]),1]
fit_all_train <- randomForestFML(Y ~ ., data = allSet_train, mtry = mtry, importance = TRUE, ntrees = 800)
plot(fit_all_train)
```




```
# run the above fitted model for all test_data
i_CUSUM <- fmlr::istar_CUSUM(test_data$mid_price, h=h_selected)
n_Event <- length(i_CUSUM)
events <- data.frame(t0 = i_CUSUM+1,
                     t1 = i_CUSUM+300,
                     trgt = rep(trgt_selected, n_Event),
                     side = rep(0,n_Event))
ptSl <- c(1,1)
out0 <- fmlr::label_meta(test_data$mid_price, events, ptSl, ex_vert = T)
table(out0$label)
```

```
##
##  -1   1
##  93 200
```

```r
fMat0 <- test_data[out0$t1Fea, !names(test_data)%in%c("m","h")]
fMat <- rbind( rep(NA, ncol(fMat0)),  apply(fMat0, 2,  function(x){diff(x)} ) )
allSet_test <- data.frame(Y=as.factor(out0$label), fMat)
idx_NA <- apply(allSet_test,1,function(x){sum(is.na(x))>0 })
allSet_test <- subset(allSet_test, !idx_NA)

pre <- predict(fit_all_train, newdata = allSet_test)
cat("Confusion Matrix", "\n")
```

```
## Confusion Matrix
```

```r
table(allSet_test$Y, pre == 1) # associate TRUE with "1"
```

```
##
##      FALSE TRUE
##   -1     9   83
##   1      9  191
```

```r
acc <- mean(allSet_test$Y==pre)
precision <- posPredValue(pre, allSet_test$Y, positive="1")
recall <- sensitivity(pre, allSet_test$Y, positive="1")
F1 <- (2 * precision * recall) / (precision + recall)
cat("acc, precision, recall, F1", "\n")
```

```
## acc, precision, recall, F1
```

```r
cat(c(acc, precision, recall, F1))
```

```
## 0.685 0.697 0.955 0.806
```

```r
acc_lucky(table(allSet_train$Y), table(allSet_test$Y), acc)
```

```
## $my_accuracy
## [1] 0.685
##
## $p_random_guess
## [1] 0
##
## $p_educated_guess
## [1] 0
##
## $mean_random_guess
```

```
## [1] 0.5
##
## $mean_educated_guess
## [1] 0.573
##
## $acc_majority_guess
## [1] 0.685
```