stock-price-movement-pred

GuangtingZhou

Required Libraries

```
library(fmlr)
library(lubridate)
library(quantmod)
library(TTR) # for various indicators
library(randomForest)
library(ROCR)
library(caret)
library(MLmetrics) # for logloss
```

Loading datasets

```
mydir = c("201804", '201805','201806')
##################### load data
myfiles <-  list.files(path=mydir, pattern=".zip", full.names=TRUE)
# myfiles
```

Data Preprocessing

```r
l_d3 <- list()
for (i in myfiles) {l_d3[i] <- read_algoseek_equity_taq(i, whichData
= 'NVDA.csv')}

# function to transfer loaded data into the fmlr-friendly format
redef <- function(dat){
  dat <- subset(dat, EventType %in% c("TRADE", "TRADE NB"))
  dat <- subset(dat, lubridate::hour(dat$Timestamp)*60+lubridate::mi
nute(dat$Timestamp) >= 9*60+30)
  dat <- subset(dat, lubridate::hour(dat$Timestamp)*60+lubridate::mi
nute(dat$Timestamp) <= 16*60)
  name <- names(dat)
  name[name=="Timestamp"] <- "tStamp"
  name[name=="Quantity"] <- "Size"
  names(dat) <- name
  dat$tStamp <- as.POSIXct( paste(dat$Date, dat$tStamp), format="%Y-
%m-%d %H:%M:%OS", tz="EST")
  return(dat)
}

# transfer the data
lr_d3 <- list()
for (i in 1:length(l_d3)) {lr_d3[[i]] <- redef(l_d3[[i]])}
```

Setting Bars

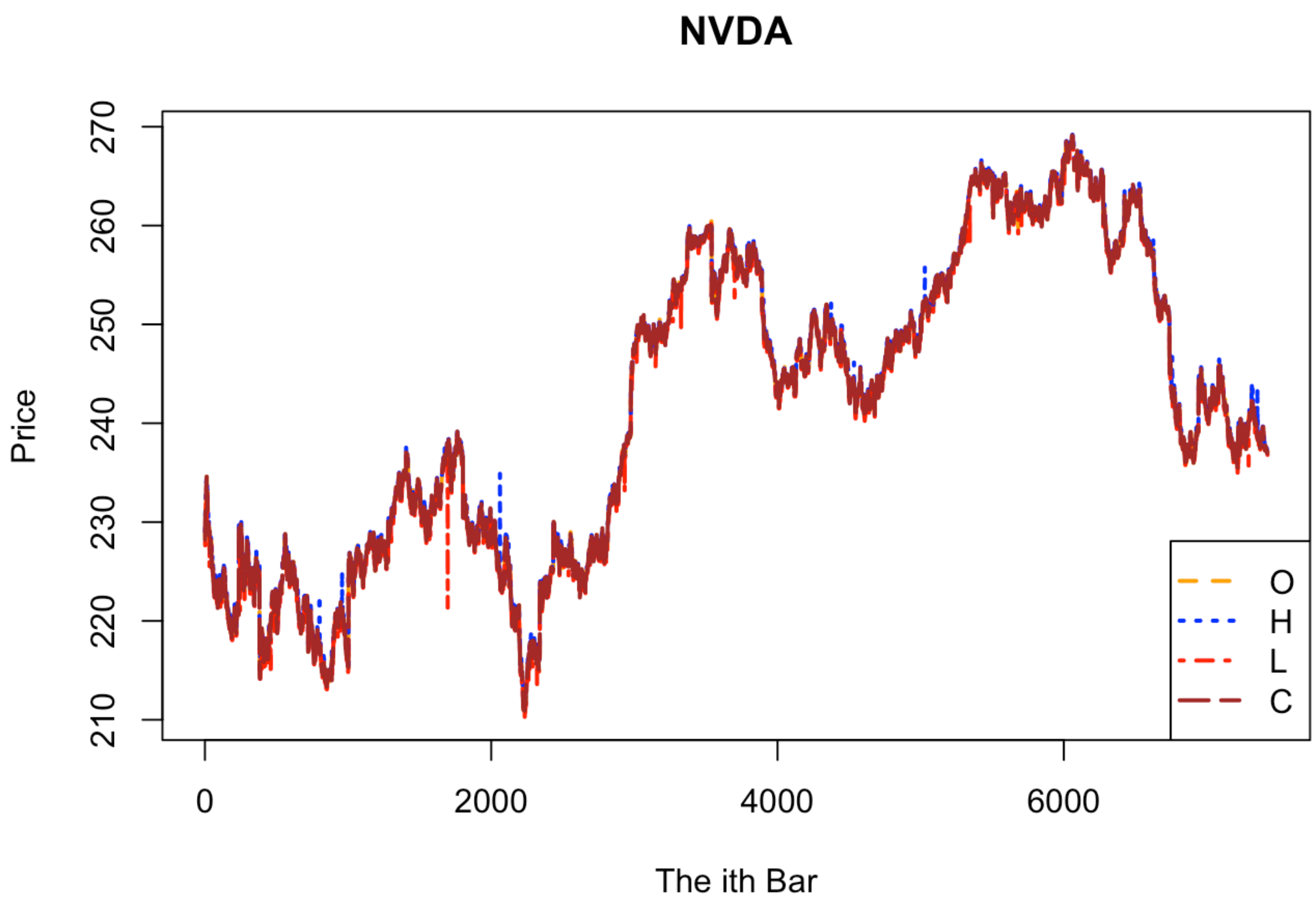
```r
tick_bar <- list()
for (i in 1:length(lr_d3)) {
  tick_bar[[i]] <- bar_tick(lr_d3[[i]], nTic=1000)
}

tick_df <- data.frame()
for (i in 1:length(tick_bar)) {
  tick_bar[[i]] <- as.data.frame(tick_bar[[i]])
  tick_df <- rbind(tick_df, tick_bar[[i]])
}
# write tick data in file
write.csv(tick_df, file = 'tick_df.csv')


unit_bar <- list()
for (i in 1:length(lr_d3)) {
  unit_bar[[i]] <- bar_unit(lr_d3[[i]], unit = 1000000)
}

unit_df <- data.frame()
for (i in 1:length(unit_bar)) {
  unit_bar[[i]] <- as.data.frame(unit_bar[[i]])
  unit_df <- rbind(unit_df, unit_bar[[i]])
}
write.csv(unit_df, file = 'unit_df.csv')
```

```r
tick_df <- read.csv('tick_df.csv')
plot(1, type = 'n',
     xlim=c(0, length(tick_df$H)),
     ylim = c(min(tick_df$L), max(tick_df$H)),
     xlab = 'The ith Bar', ylab = 'Price', main = 'NVDA')
lines(tick_df$O, col = 'orange', lty=2, lwd=2)
lines(tick_df$H, col='blue',  lty=3, lwd=2)
lines(tick_df$L, col='red', lty=4, lwd=2)
lines(tick_df$C, col='brown', lty=5, lwd=2)
legend('bottomright', legend = c('O', 'H', 'L', 'C'),
       col= c('orange', 'blue', 'red', 'brown'),
       lty=c(2, 3, 4, 5), lwd=c(2, 2, 2, 2), merge = T)
```



Adding indicators

```
############################ By Tick Bars
########### Features Setting
DAT_T <- read.csv('tick_df.csv')
dat <- DAT_T[,c('H', 'L', 'O', 'C', 'V')]
dat$V <- as.numeric(dat$V/1e6)
dat$C <- as.numeric(dat$C)
dat$H <- as.numeric(dat$H)
dat$L <- as.numeric(dat$L)
dat$O <- as.numeric(dat$O)
names(dat) <- c("High", "Low", "Open", "Close", "Volume")

# functions used to prepare the following indicators from TTR
HL <- function(dat){cbind(dat$High, dat$Low)}
HLC <- function(dat){cbind(dat$High, dat$Low, dat$Close)}

# add various indicators
dat_used <- cbind(dat,
```

```
ADX=ADX(dat)[,4],
aroon=aroon(HL(dat))[,3],
ATR=ATR(dat)[,2],
BBands(HLC(dat)),
CCI=CCI(HLC(dat)),
chaikinAD=chaikinAD(HLC(dat), dat$Volume),
chaikinVolatility=chaikinVolatility(dat),
CLV=CLV(dat),
CMF=CMF(HLC(dat), dat$Volume),
CMOClose=CMO(dat$Close),
CMOVol=CMO(dat$Volume),
DonchianChannel(HL(dat)),
DPOClose=DPO(dat$Close),
DPOVol=DPO(dat$Volume),
DVI(dat$Close),
EMV=EMV(HLC(dat), dat$Volume)[,1],
GMMA(dat$Close),
GMMA(dat$Volume),
KST=KST(dat$Close)[,1],
MACDClose=MACD(dat$Close)[,1],
MACDVol=MACD(dat$Volume)[,1],
MFI=MFI(HLC(dat), dat$Volume),
OBV=OBV(dat$Close, dat$Volume),
PBands(dat$Close),
ROCClose=ROC(dat$Close),
ROCVol=ROC(dat$Volume),
momentum=momentum(dat$Close),
RSI=RSI(dat$Close),
runPerRankClose=runPercentRank(dat$Close),
runPerRankVolume=runPercentRank(dat$Volume),
SAR=SAR(HL(dat)),
VWAP=VWAP(dat$Close, volume=dat$Volume),
SNR=SNR(HLC(dat), n=30),
stoch(HLC(dat)),
SMI=SMI(HLC(dat))[,1],
TDI=TDI(dat$Close)[,1],
TRIX=TRIX(dat$Close)[,1],
ultimateOsc=ultimateOscillator(HLC(dat)),
VHF=VHF(dat$Close),
vola=volatility(dat),
williamsAD=williamsAD(HLC(dat)),
WPR=WPR(HLC(dat))
```

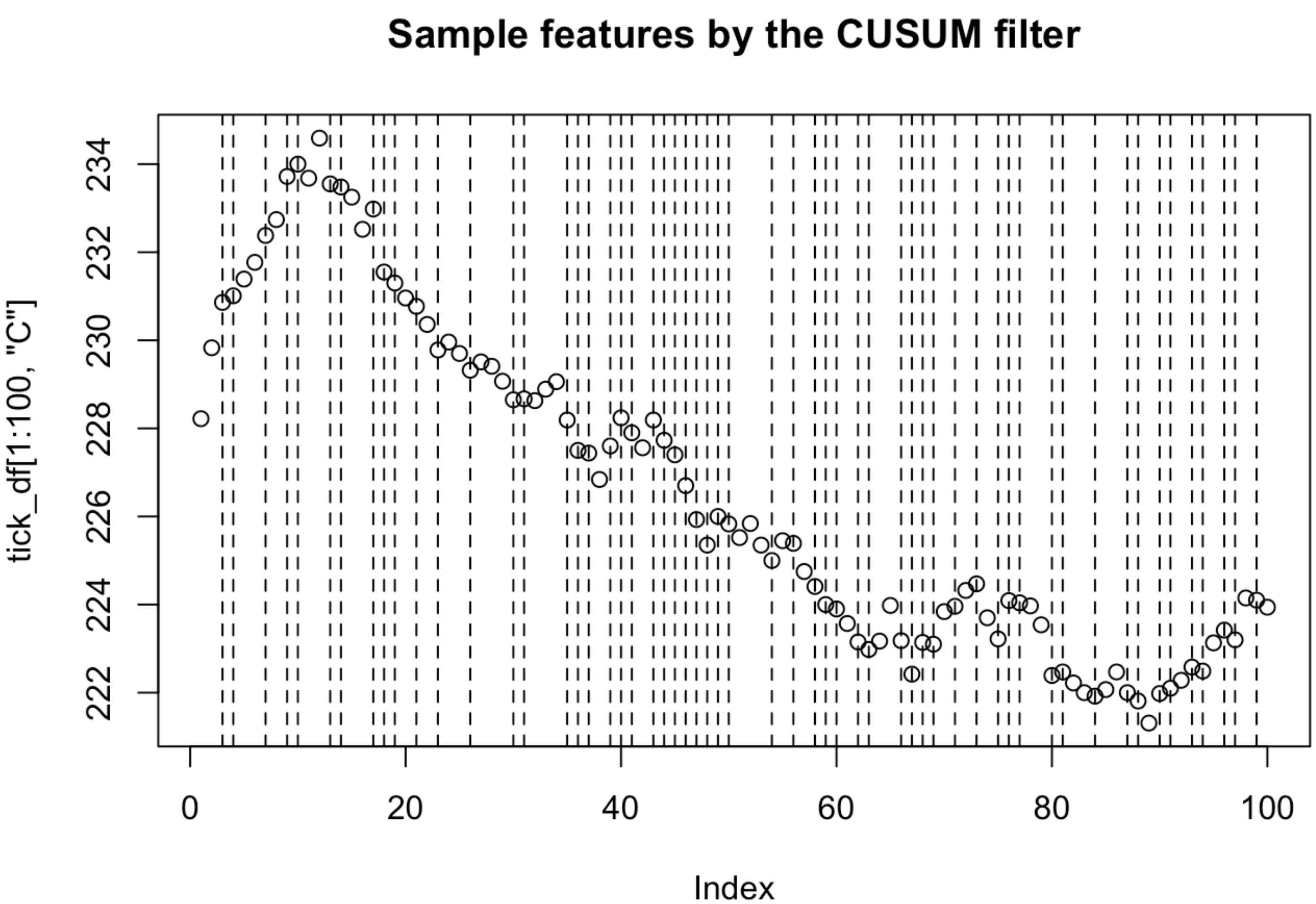
```
)
dim(dat_used)
```

```
## [1] 7423   80
```

CUSUM to Access Features and Labels

```
## plot for visualization, just part of the data included
hvec <- na.locf(c(NA,0.5*runSD(tick_df[1:100,'C'])), fromLast = T)
i_CUSUM <- fmlr::istar_CUSUM(tick_df[1:100,'C'], h=hvec)
n_Event <- length(i_CUSUM)

plot(tick_df[1:100,'C'], main="Sample features by the CUSUM filter")
abline(v=i_CUSUM+1, lty = 2)
```



```
################ CUSUMs, prepare features and labels
hvec <- na.locf(c(NA,0.5*runSD(dat_used$Close)), fromLast = T)
i_CUSUM <- fmlr::istar_CUSUM(dat_used$Close, h=hvec)
n_Event <- length(i_CUSUM)

events <- data.frame(t0=i_CUSUM+1,
                     t1 = i_CUSUM+200,
                     trgt = rep(0.001, n_Event),
                     side=rep(1,n_Event))
ptSl <- c(1,1)

out0 <- fmlr::label_meta(dat_used$Close, events, ptSl)
table(out0$label) # imbalanced data, need smote
```

```
##
##    0    1
##  306 4067
```

Combine Labels, Features and Indicators

```
########## Combine labels, features and indicators
fMat0 <- dat_used[out0$t1Fea,]
allSet <- data.frame(Y=as.factor(out0$label),fMat0, t1Fea=out0$t1Fea
, tLabel=out0$tLabel)

# exclude NA at the begining of the indicators
idx_NA <- apply(allSet,1,function(x){sum(is.na(x))>0})
# train-test-split
allSet <- subset(allSet, !idx_NA)
nx <- nrow(allSet)
trainSet <- allSet[1:floor(nx*2/3),]
testSet <- allSet[(floor(nx*2/3)+1):nx,]
dim(allSet)
```

```
## [1] 4166   83
```

```
dim(trainSet)
```

```
## [1] 2777   83
```

```
dim(testSet)
```

```
## [1] 1389   83
```

SMOTE

```
################## SMOTE
tb <- table(trainSet$Y)
ratio <- tb[names(tb)=='1']/tb[names(tb)=='0']
ratio
```

```
##        1
## 19.41912
```

```
if(ratio > 1) perc <- list("0"=ratio, "1"=1) else perc <- list("0"=1
, "1"= (1/ratio))

trainSet_balanced <- UBL::SmoteClassif(Y ~ . - Close - t1Fea - tLabe
l, dat = trainSet, C.perc = perc)
table(trainSet_balanced$Y)
```

```
##
##    0    1
## 2640 2641
```

Model Fitting and Feature Importance Analysis

Feature Importance

```
logistic <- glm(Y~., family = binomial(link='logit'), data=trainSet)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurre
d
```

```
prob_test <- predict(logistic, newdata = testSet, type='response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
test.res <- ifelse(prob_test>=0.5, 1, 0)
table(testSet$Y, test.res)
```

```
##    test.res
##        0    1
##   0   19  129
##   1    2 1239
```

```
pred <- prediction(prob_test, testSet$Y)
tb_test <- table(testSet$Y)
acc_perf <- performance(pred, measure = "acc")
acc_vec <- acc_perf@y.values[[1]]
acc <- acc_vec[max(which(acc_perf@x.values[[1]] >= 0.5))]
acc
```

```
## [1] 0.9056875
```

```
lucky_score <- fmlr::acc_lucky(train_class = table(trainSet$Y),
                               test_class = tb_test,
                               my_acc = acc)
lucky_score
```

```
## $my_accuracy
## [1] 0.9056875
##
## $p_random_guess
## [1] 0
##
## $p_educated_guess
## [1] 0
##
## $mean_random_guess
## [1] 0.5
##
## $mean_educated_guess
## [1] 0.8552268
##
## $acc_majority_guess
## [1] 0.8934485
```

```
summary(logistic)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial(link = "logit"), data = tr
ainSet)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.090e-04   2.100e-08   2.100e-08   2.100e-08   1.168e-04
##
## Coefficients: (5 not defined because of singularities)
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -4.660e+02  1.058e+06   0.000    1.000
## High               2.991e+01  9.039e+04   0.000    1.000
## Low                3.363e+01  6.637e+04   0.001    1.000
## Open              -3.487e+01  1.037e+05   0.000    1.000
## Close             -2.131e+02  6.168e+05   0.000    1.000
## Volume            -1.752e+02  7.300e+05   0.000    1.000
## ADX               -9.188e-01  1.464e+03  -0.001    0.999
## aroon             -8.774e-02  2.696e+02   0.000    1.000
## ATR               -4.242e+00  1.301e+05   0.000    1.000
```

```
## dn                 -3.645e+00  2.241e+04   0.000    1.000
## mavg               -2.005e+02  4.924e+05   0.000    1.000
## up                         NA         NA      NA       NA
## pctB               -2.213e+01  3.069e+05   0.000    1.000
## CCI                 2.016e-03  8.248e+02   0.000    1.000
## chaikinAD          -2.991e+00  1.868e+04   0.000    1.000
## chaikinVolatility  8.029e+00  5.957e+04   0.000    1.000
## CLV                 1.177e+01  3.453e+04   0.000    1.000
## CMF                -6.180e-01  9.307e+04   0.000    1.000
## CMOClose           -1.207e-01  8.200e+02   0.000    1.000
## CMOVol             -7.132e-02  7.117e+02   0.000    1.000
## high                7.476e+00  3.555e+04   0.000    1.000
## mid                -6.156e+00  7.594e+04   0.000    1.000
## low                        NA         NA      NA       NA
## DPOClose            7.531e+00  1.693e+04   0.000    1.000
## DPOVol              1.674e+02  2.114e+05   0.001    0.999
## dvi.mag             1.796e+01  8.833e+04   0.000    1.000
## dvi.str            -1.648e+01  4.507e+04   0.000    1.000
## dvi                        NA         NA      NA       NA
## EMV                -4.384e-05  4.148e-01   0.000    1.000
## short.lag.3         1.589e+03  6.315e+06   0.000    1.000
## short.lag.5        -1.068e+04  4.007e+07   0.000    1.000
## short.lag.8         1.204e+05  3.991e+08   0.000    1.000
## short.lag.10       -3.977e+05  1.272e+09   0.000    1.000
## short.lag.12        4.972e+05  1.579e+09   0.000    1.000
## short.lag.15       -2.619e+05  8.462e+08   0.000    1.000
## long.lag.30         7.030e+05  2.975e+09   0.000    1.000
## long.lag.35        -1.990e+06  9.490e+09   0.000    1.000
## long.lag.40         2.593e+06  1.393e+10   0.000    1.000
## long.lag.45        -1.774e+06  1.072e+10   0.000    1.000
## long.lag.50         5.489e+05  3.708e+09   0.000    1.000
## long.lag.60        -2.845e+04  2.365e+08   0.000    1.000
## short.lag.3.1       3.157e+03  3.275e+07   0.000    1.000
## short.lag.5.1      -6.443e+04  3.950e+08   0.000    1.000
## short.lag.8.1       1.628e+06  6.274e+09   0.000    1.000
## short.lag.10.1     -7.841e+06  2.468e+10   0.000    1.000
## short.lag.12.1      1.342e+07  3.632e+10   0.000    1.000
## short.lag.15.1     -1.042e+07  2.394e+10   0.000    1.000
## long.lag.30.1       9.981e+07  1.571e+11   0.001    0.999
## long.lag.35.1      -3.915e+08  5.742e+11  -0.001    0.999
## long.lag.40.1       6.880e+08  9.482e+11   0.001    0.999
## long.lag.45.1      -6.212e+08  8.087e+11  -0.001    0.999
```

```
## long.lag.50.1      2.487e+08  3.070e+11   0.001    0.999
## long.lag.60.1     -2.054e+07  2.302e+10  -0.001    0.999
## KST                6.447e+00  2.415e+04   0.000    1.000
## MACDClose          1.689e+02  2.125e+06   0.000    1.000
## MACDVol           -1.396e+00  5.307e+03   0.000    1.000
## MFI               -3.172e-01  8.887e+02   0.000    1.000
## OBV                4.520e+00  8.896e+03   0.001    1.000
## dn.1               5.741e-01  2.148e+04   0.000    1.000
## center             1.337e+02  5.256e+05   0.000    1.000
## up.1                      NA         NA      NA       NA
## ROCClose          -1.690e+03  6.369e+07   0.000    1.000
## ROCVol             9.643e-01  2.967e+04   0.000    1.000
## momentum           4.161e+01  2.754e+05   0.000    1.000
## RSI                3.585e-01  3.636e+03   0.000    1.000
## runPerRankClose   -3.344e+01  1.323e+05   0.000    1.000
## runPerRankVolume   6.660e+00  3.689e+04   0.000    1.000
## sar                2.304e+00  9.771e+03   0.000    1.000
## VWAP               2.179e+01  8.972e+04   0.000    1.000
## SNR                8.577e-01  7.036e+03   0.000    1.000
## fastK              7.623e+01  1.615e+05   0.000    1.000
## fastD             -6.485e+01  2.843e+05   0.000    1.000
## slowD             -3.948e+00  2.673e+05   0.000    1.000
## SMI                3.700e-01  9.265e+02   0.000    1.000
## TDI               -1.808e-02  5.462e+02   0.000    1.000
## TRIX              -8.118e+03  1.291e+07  -0.001    0.999
## ultimateOsc        1.584e-01  1.870e+03   0.000    1.000
## VHF                1.867e+01  1.768e+05   0.000    1.000
## vola              -1.568e+02  8.486e+05   0.000    1.000
## williamsAD        -2.101e+00  1.789e+03  -0.001    0.999
## WPR                       NA         NA      NA       NA
## t1Fea              5.921e-01  1.308e+02   0.005    0.996
## tLabel            -5.873e-01  1.367e+02  -0.004    0.997
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.0857e+03  on 2776  degrees of freedom
## Residual deviance: 1.5916e-07  on 2699  degrees of freedom
## AIC: 156
##
## Number of Fisher Scoring iterations: 25
```

```r
varImp(logistic)
```

```
##                        Overall
## High              3.309225e-04
## Low               5.067409e-04
## Open              3.362354e-04
## Close             3.455762e-04
## Volume            2.399767e-04
## ADX               6.277373e-04
## aroon             3.254997e-04
## ATR               3.260446e-05
## dn                1.626675e-04
## mavg              4.072490e-04
## pctB              7.210644e-05
## CCI               2.444625e-06
## chaikinAD         1.601068e-04
## chaikinVolatility 1.347897e-04
## CLV               3.408374e-04
## CMF               6.639887e-06
## CMOClose          1.471934e-04
## CMOVol            1.002060e-04
## high              2.102789e-04
## mid               8.106614e-05
## DPOClose          4.448986e-04
## DPOVol            7.921233e-04
## dvi.mag           2.033636e-04
## dvi.str           3.656828e-04
## EMV               1.056931e-04
## short.lag.3       2.515858e-04
## short.lag.5       2.665423e-04
## short.lag.8       3.016987e-04
## short.lag.10      3.125245e-04
## short.lag.12      3.149197e-04
## short.lag.15      3.095260e-04
## long.lag.30       2.362630e-04
## long.lag.35       2.097400e-04
## long.lag.40       1.860556e-04
## long.lag.45       1.655689e-04
## long.lag.50       1.480249e-04
## long.lag.60       1.202774e-04
## short.lag.3.1     9.639439e-05
```

```
## short.lag.5.1     1.631234e-04
## short.lag.8.1     2.595346e-04
## short.lag.10.1    3.177400e-04
## short.lag.12.1    3.694838e-04
## short.lag.15.1    4.351941e-04
## long.lag.30.1     6.353102e-04
## long.lag.35.1     6.818417e-04
## long.lag.40.1     7.256290e-04
## long.lag.45.1     7.681693e-04
## long.lag.50.1     8.100843e-04
## long.lag.60.1     8.923186e-04
## KST               2.669455e-04
## MACDClose         7.949669e-05
## MACDVol           2.630858e-04
## MFI               3.568721e-04
## OBV               5.081148e-04
## dn.1              2.672338e-05
## center            2.542857e-04
## ROCClose          2.653609e-05
## ROCVol            3.250327e-05
## momentum          1.510734e-04
## RSI               9.859506e-05
## runPerRankClose   2.526896e-04
## runPerRankVolume  1.805488e-04
## sar               2.358324e-04
## VWAP              2.428989e-04
## SNR               1.219082e-04
## fastK             4.719454e-04
## fastD             2.281244e-04
## slowD             1.476797e-05
## SMI               3.993673e-04
## TDI               3.310847e-05
## TRIX              6.290343e-04
## ultimateOsc       8.471317e-05
## VHF               1.055928e-04
## vola              1.848084e-04
## williamsAD        1.174451e-03
## t1Fea             4.526364e-03
## tLabel            4.295291e-03
```

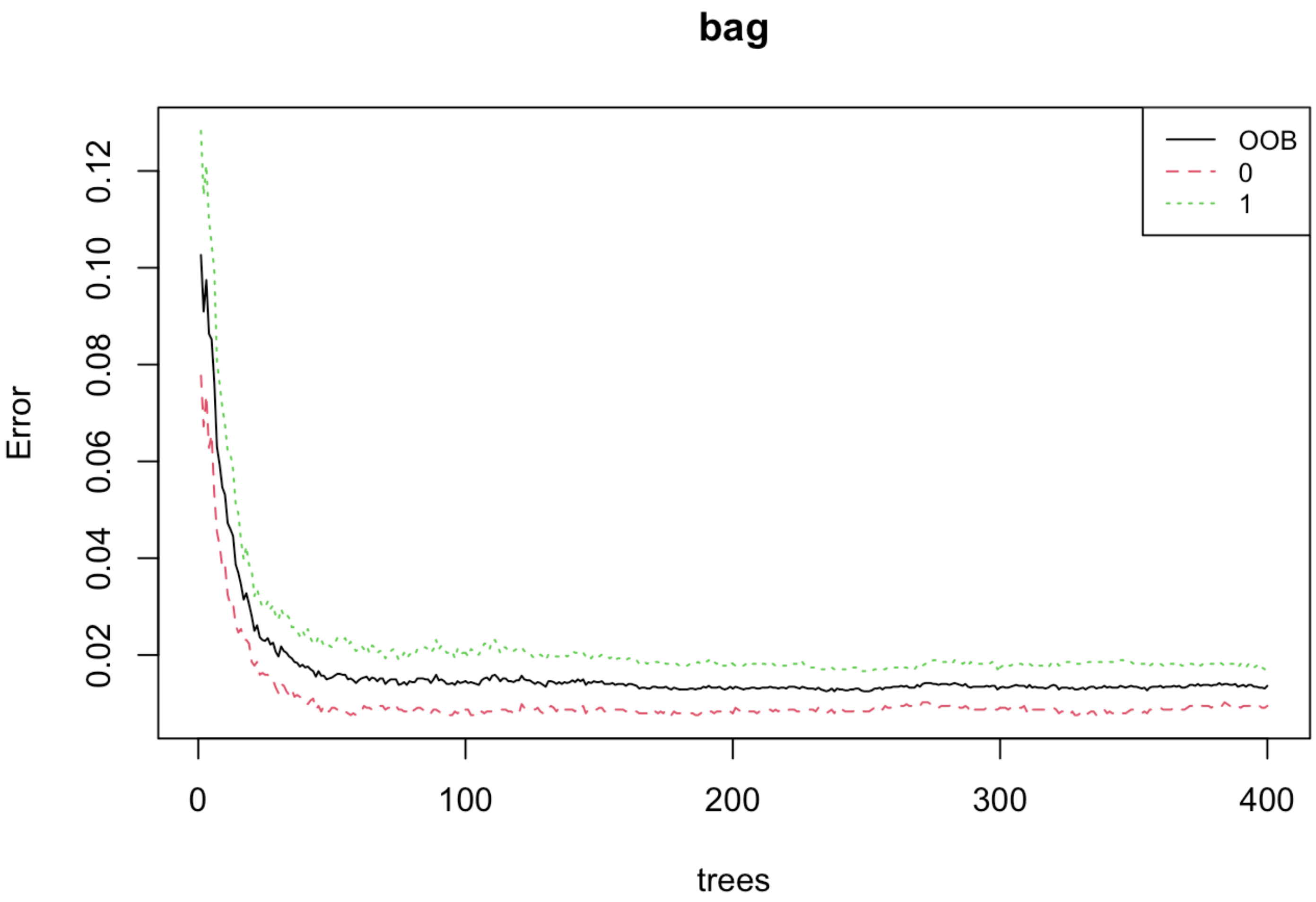
```r
# try random forest
# feature importance
mtry <- tuneRF(trainSet_balanced[,-1], trainSet_balanced$Y, plot=F)
```

```
## mtry = 9  OOB error = 1.42%
## Searching left ...
## mtry = 5     OOB error = 1.33%
## 0.06666667 0.05
## mtry = 3     OOB error = 1.8%
## -0.3571429 0.05
## Searching right ...
## mtry = 18    OOB error = 1.51%
## -0.1428571 0.05
```

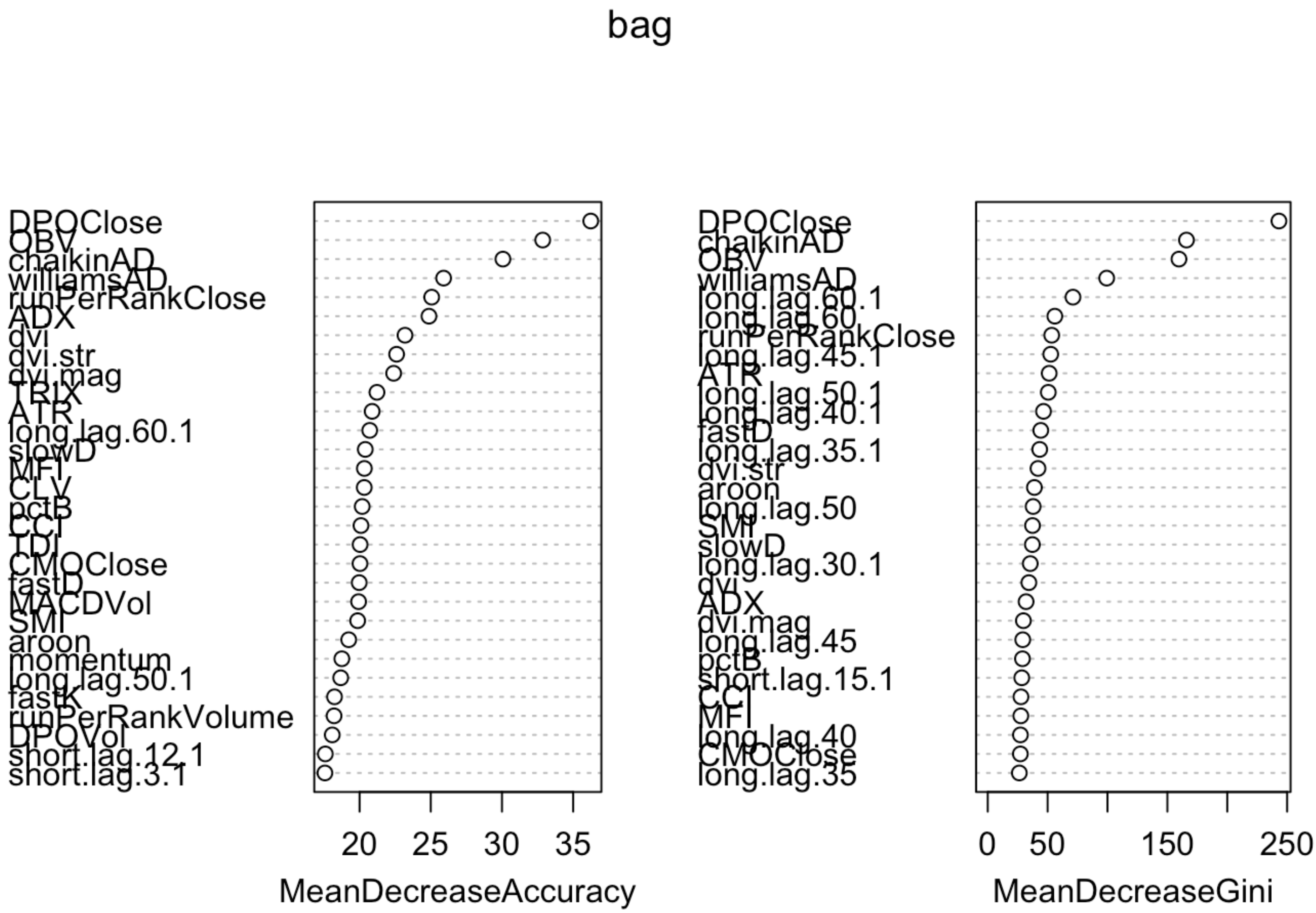
```r
mtry <- mtry[which.min(mtry[,2]),1]
mtry
```

```
## [1] 5
```

```r
bag <- randomForest(Y ~ . - Close - t1Fea - tLabel, data = trainSet_
balanced, mtry = mtry, importance = TRUE, ntree = 400, SB=0)
plot(bag)
legend("topright", colnames(bag$err.rate),col=1:3,cex=0.8,lty=1:3)
```




```
varImpPlot(bag)
```




```
# evaluating auc based on the test set
prob_test <- predict(bag, newdata=testSet, type="prob")
pred <- prediction(prob_test[,2], testSet$Y) # the 2nd column is whe
re the label "1" is
acc_perf <- performance(pred, measure = "acc")
acc_vec <- acc_perf@y.values[[1]]
acc <- acc_vec[max(which(acc_perf@x.values[[1]] >= 0.5))]
acc
```

```
## [1] 0.8920086
```

```
lucky_score <- fmlr::acc_lucky(train_class = table(trainSet$Y),
                               test_class = table(testSet$Y),
                               my_acc = acc)
lucky_score
```

```
## $my_accuracy
## [1] 0.8920086
##
## $p_random_guess
## [1] 0
##
## $p_educated_guess
## [1] 0
##
## $mean_random_guess
## [1] 0.5001353
##
## $mean_educated_guess
## [1] 0.8547437
##
## $acc_majority_guess
## [1] 0.8934485
```

PCA importance

```
# PCA importance
table(testSet$Y, prob_test[,2] >= 0.5)
```

```
##
##     FALSE TRUE
##   0     1  147
##   1     3 1238
```

```
trainFea <- trainSet_balanced[, !(names(trainSet)%in%c('Y', 'Close',
't1Fea', 'tLabel'))]
pca <- prcomp(trainFea, center = TRUE, scale. = TRUE)
summary(pca)
```

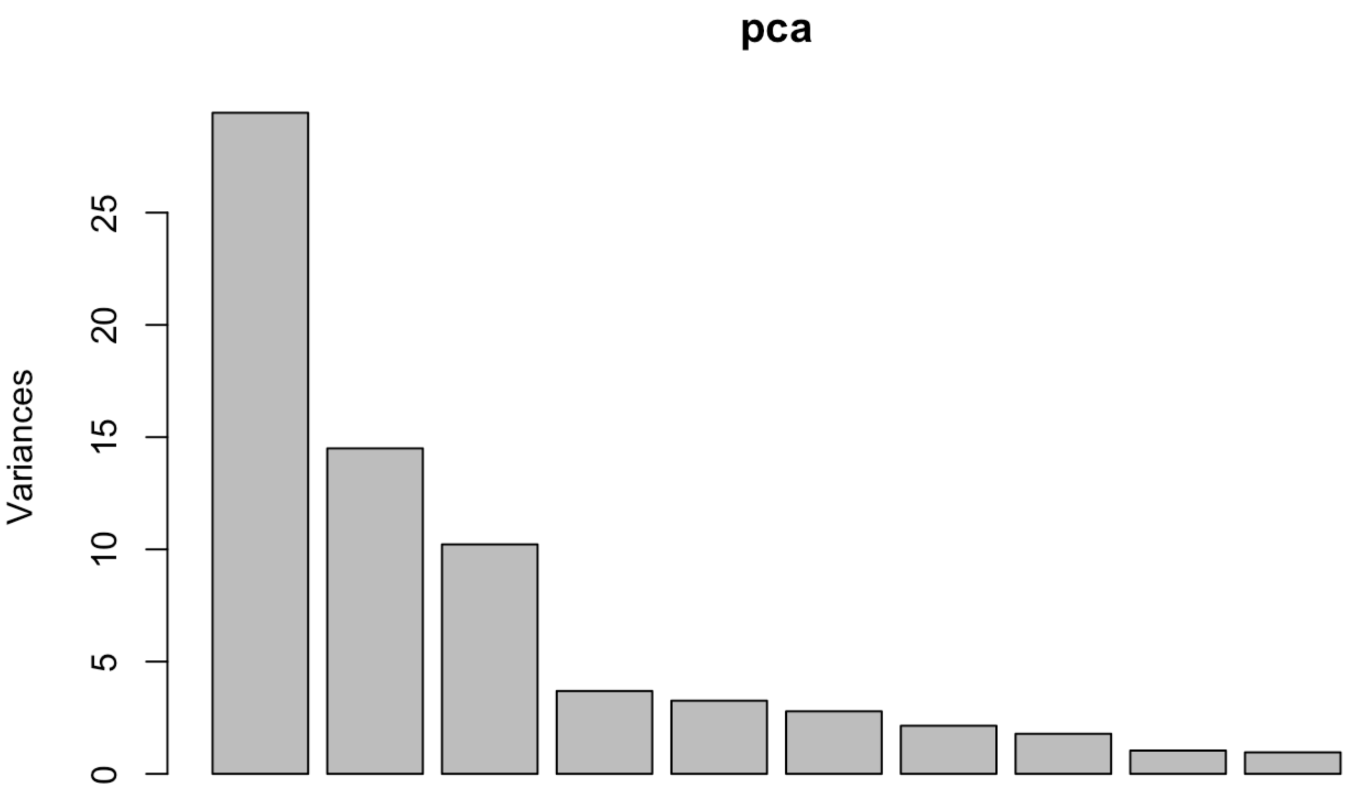
```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     P
C6     PC7
## Standard deviation     5.4268 3.8078 3.1972 1.92099 1.80539 1.669
35 1.46380
## Proportion of Variance 0.3728 0.1835 0.1294 0.04671 0.04126 0.035
```

```
27 0.02712
## Cumulative Proportion  0.3728 0.5563 0.6857 0.73243 0.77369 0.808
96 0.83609
##                            PC8     PC9    PC10    PC11    PC12    
PC13    PC14
## Standard deviation     1.33568 1.01868 0.97902 0.91696 0.88199 0.
85417 0.84952
## Proportion of Variance 0.02258 0.01314 0.01213 0.01064 0.00985 0.
00924 0.00914
## Cumulative Proportion  0.85867 0.87181 0.88394 0.89458 0.90443 0.
91366 0.92280
##                           PC15   PC16    PC17    PC18   PC19     P
C20    PC21
## Standard deviation     0.80058 0.7382 0.72418 0.70712 0.6651 0.65
037 0.62406
## Proportion of Variance 0.00811 0.0069 0.00664 0.00633 0.0056 0.00
535 0.00493
## Cumulative Proportion  0.93091 0.9378 0.94445 0.95078 0.9564 0.96
173 0.96666
##                           PC22    PC23    PC24    PC25    PC26    
PC27    PC28
## Standard deviation     0.61139 0.59447 0.57339 0.53401 0.49139 0.
45977 0.40419
## Proportion of Variance 0.00473 0.00447 0.00416 0.00361 0.00306 0.
00268 0.00207
## Cumulative Proportion  0.97139 0.97587 0.98003 0.98364 0.98669 0.
98937 0.99144
##                           PC29    PC30    PC31    PC32    PC33    
PC34    PC35
## Standard deviation     0.36953 0.36295 0.32319 0.26779 0.23732 0.
22275 0.16314
## Proportion of Variance 0.00173 0.00167 0.00132 0.00091 0.00071 0.
00063 0.00034
## Cumulative Proportion  0.99317 0.99483 0.99616 0.99706 0.99778 0.
99841 0.99874
##                           PC36    PC37    PC38    PC39    PC40    
PC41    PC42
## Standard deviation     0.14846 0.14206 0.11604 0.10193 0.09124 0.
06909 0.06546
## Proportion of Variance 0.00028 0.00026 0.00017 0.00013 0.00011 0.
00006 0.00005
## Cumulative Proportion  0.99902 0.99928 0.99945 0.99958 0.99968 0.
```

```
99974 0.99980
##                           PC43    PC44    PC45    PC46    PC47
PC48    PC49
## Standard deviation     0.05898 0.05789 0.05393 0.03829 0.03321 0.
03110 0.03090
## Proportion of Variance 0.00004 0.00004 0.00004 0.00002 0.00001 0.
00001 0.00001
## Cumulative Proportion  0.99984 0.99988 0.99992 0.99994 0.99995 0.
99997 0.99998
##                           PC50    PC51    PC52    PC53     PC54
PC55
## Standard deviation     0.02741 0.01932 0.01259 0.01158 0.009928 0
.008402
## Proportion of Variance 0.00001 0.00000 0.00000 0.00000 0.000000 0
.000000
## Cumulative Proportion  0.99999 0.99999 0.99999 1.00000 1.000000 1
.000000
##                            PC56     PC57     PC58     PC59     PC
60     PC61
## Standard deviation     0.006929 0.004753 0.003864 0.003239 0.0025
61 0.001692
## Proportion of Variance 0.000000 0.000000 0.000000 0.000000 0.0000
00 0.000000
## Cumulative Proportion  1.000000 1.000000 1.000000 1.000000 1.0000
00 1.000000
##                             PC62      PC63      PC64      PC65
PC66
## Standard deviation     0.0006353 0.0005528 0.0003158 0.0001879 0.
0001325
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.
0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.
0000000
##                             PC67      PC68      PC69      PC70
PC71
## Standard deviation     3.302e-05 2.447e-05 1.389e-05 6.522e-06 1.
896e-06
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.
000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.
000e+00
##                             PC72      PC73      PC74      PC75
```

```
PC76
## Standard deviation     5.167e-07 4.026e-07 3.897e-08 3.429e-15 1.
151e-15
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.
000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.
000e+00
##                             PC77      PC78      PC79
## Standard deviation     1.075e-15 5.319e-16 3.334e-16
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00
```

```
plot(pca)
```



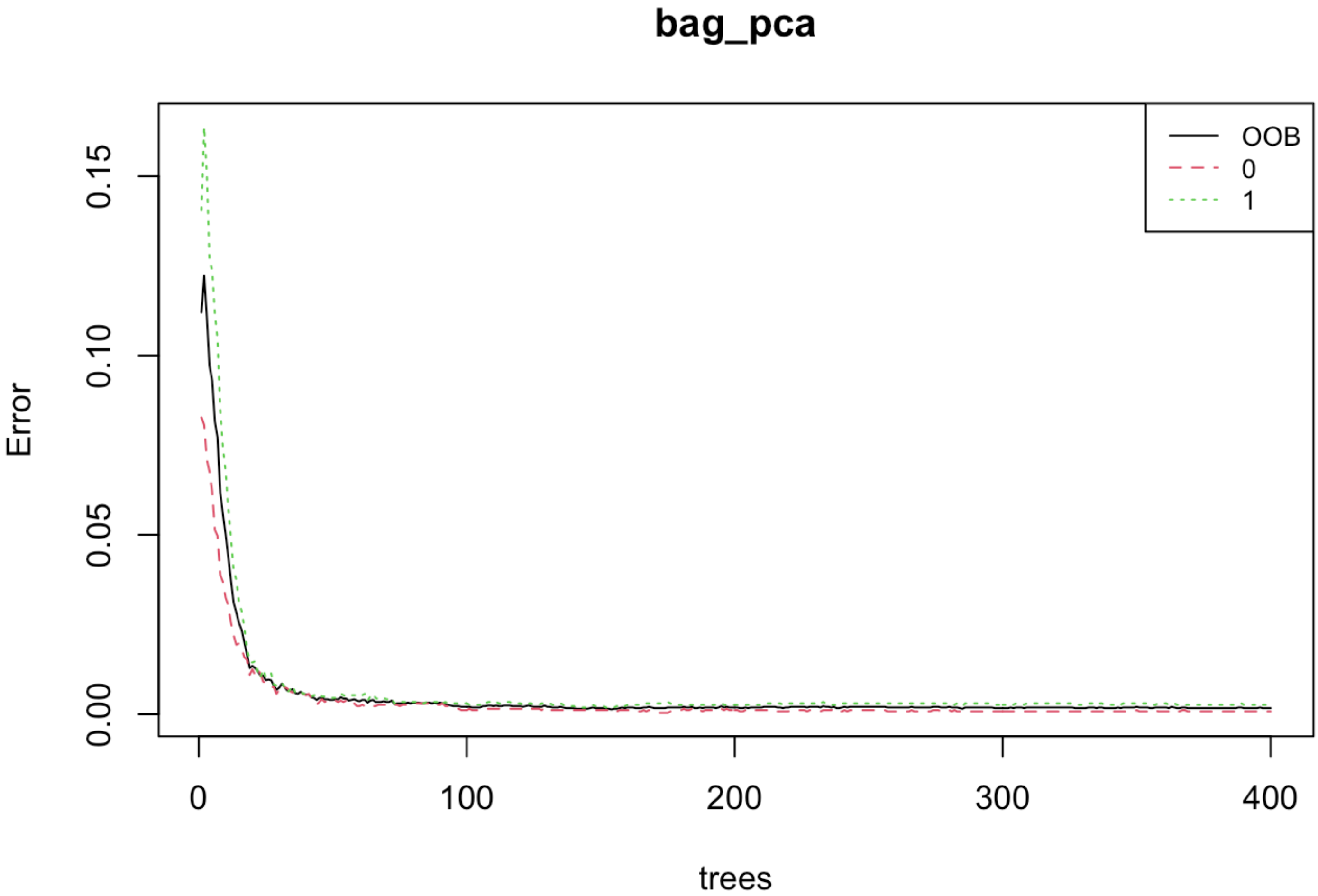

```
trainPCA <- data.frame(Y=trainSet_balanced$Y, pca$x)
mtry_p <- tuneRF(trainPCA[,-1], trainPCA$Y, plot = F)
```

```
## mtry = 8  OOB error = 0.57%
## Searching left ...
## mtry = 4     OOB error = 0.42%
## 0.2666667 0.05
## mtry = 2     OOB error = 0.53%
## -0.2727273 0.05
## Searching right ...
## mtry = 16    OOB error = 0.76%
## -0.8181818 0.05
```

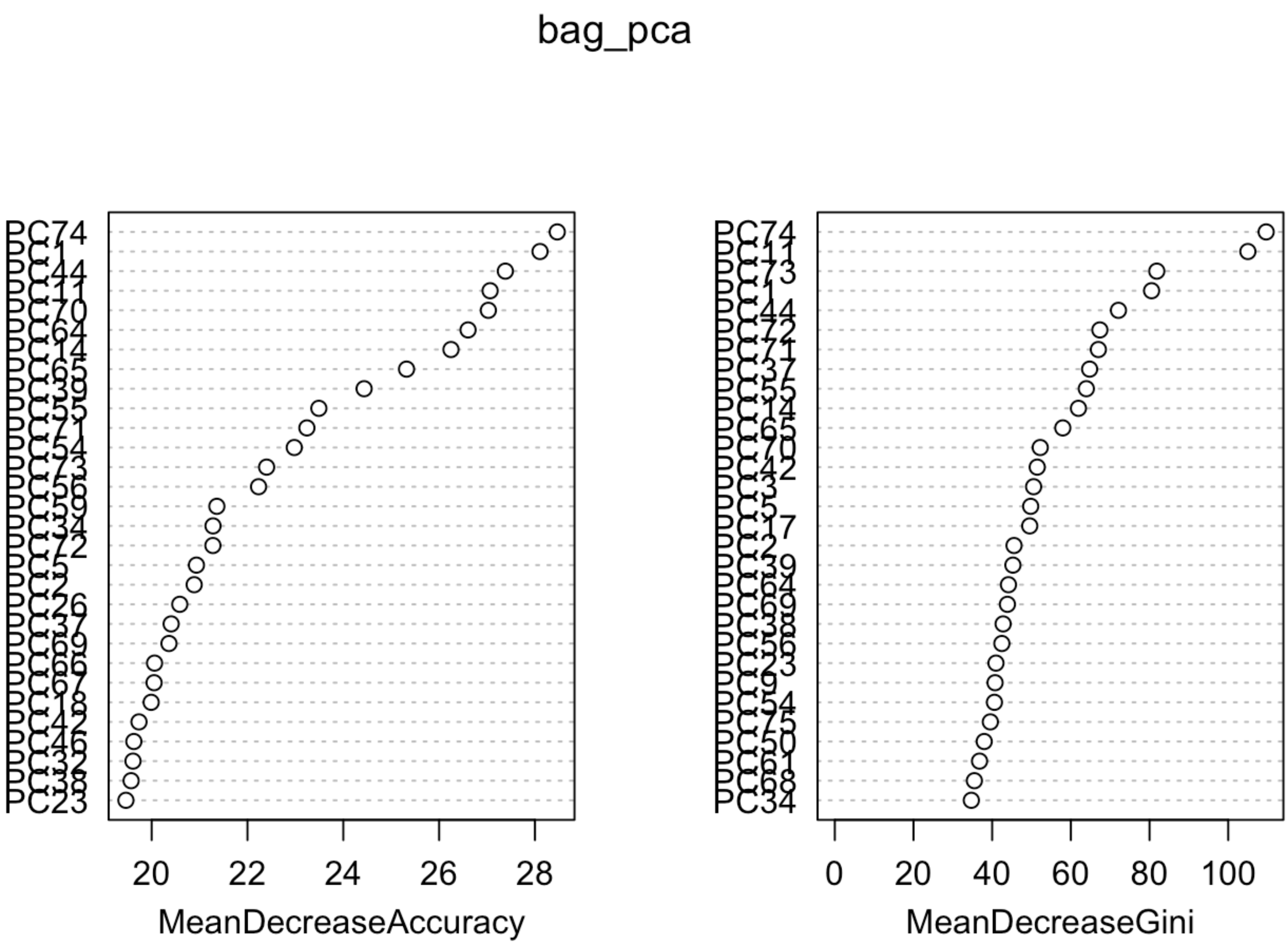
```
mtry_p <- mtry_p[which.min(mtry_p[,2]),1] #mtry=18
mtry_p
```

```
## [1] 4
```

```
bag_pca <- randomForest(Y ~ ., data = trainPCA, mtry = mtry_p, impor
tance = TRUE, ntree = 400, SB=0)
plot(bag_pca)
legend("topright", colnames(bag$err.rate),col=1:3,cex=0.8,lty=1:3)
```




```
varImpPlot(bag_pca)
```




```
testFea <- testSet[, !(names(testSet)%in%c('Y', 'Close', 't1Fea', 't
Label'))]
testPCA <- data.frame(Y=testSet$Y, (scale(testFea, center= pca$cente
r, scale = pca$scale) %*% pca$rotation))

prob_test <- predict(bag_pca, newdata=testPCA, type="prob")
table(testPCA$Y, prob_test[,2] >= 0.5)
```

```
##
##      TRUE
##    0  148
##    1 1241
```

```
pred <- prediction(prob_test[,2], testPCA$Y)
tb_test <- table(testPCA$Y)
acc_perf <- performance(pred, measure = "acc")
acc_vec <- acc_perf@y.values[[1]]
acc <- acc_vec[max(which(acc_perf@x.values[[1]] >= 0.5))]
lucky_score <- fmlr::acc_lucky(train_class = table(trainPCA$Y),
                               test_class = tb_test,
                               my_acc = acc)
lucky_score
```

```
## $my_accuracy
## [1] 0.8934485
##
## $p_random_guess
## [1] 0
##
## $p_educated_guess
## [1] 0
##
## $mean_random_guess
## [1] 0.5000266
##
## $mean_educated_guess
## [1] 0.4999006
##
## $acc_majority_guess
## [1] 0.8934485
```

```
logistic <- glm(Y~., family = binomial(link='logit'), data=trainPCA)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurre
d
```

```
prob_test <- predict(logistic, newdata = testPCA, type='response')
test.res <- ifelse(prob_test>=0.5, 1, 0)
table(testPCA$Y, test.res)
```

```
##    test.res
##        0    1
##   0    7  141
##   1   53 1188
```

```
pred <- prediction(prob_test, testPCA$Y)
tb_test <- table(testSet$Y)
acc_perf <- performance(pred, measure = "acc")
acc_vec <- acc_perf@y.values[[1]]
acc <- acc_vec[max(which(acc_perf@x.values[[1]] >= 0.5))]
acc
```

```
## [1] 0.8603312
```

```
lucky_score <- fmlr::acc_lucky(train_class = table(trainSet$Y),
                               test_class = tb_test,
                               my_acc = acc)
lucky_score
```

```
## $my_accuracy
## [1] 0.8603312
##
## $p_random_guess
## [1] 0
##
## $p_educated_guess
## [1] 0.189
##
## $mean_random_guess
## [1] 0.4999806
##
## $mean_educated_guess
## [1] 0.8548387
##
## $acc_majority_guess
## [1] 0.8934485
```

```r
summary(logistic)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial(link = "logit"), data = tr
ainPCA)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3268  -0.4479   0.0000   0.2223   5.6728
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  9.906e-01  8.697e-02  11.390  < 2e-16 ***
## PC1         -2.107e-01  5.744e-02  -3.668 0.000245 ***
## PC2         -4.061e-02  9.350e-02  -0.434 0.664036
## PC3         -1.760e-01  9.284e-02  -1.896 0.057954 .
## PC4         -5.826e-02  3.333e-02  -1.748 0.080460 .
## PC5          3.963e-01  5.134e-02   7.718 1.18e-14 ***
## PC6         -3.067e-01  8.747e-02  -3.506 0.000455 ***
## PC7          2.435e-01  5.222e-02   4.663 3.11e-06 ***
## PC8          5.235e-01  7.921e-02   6.609 3.87e-11 ***
## PC9         -1.303e+00  8.358e-02 -15.587  < 2e-16 ***
## PC10         5.046e-01  2.347e-01   2.150 0.031547 *
## PC11         2.104e+00  1.222e-01  17.224  < 2e-16 ***
## PC12        -4.164e-01  1.027e-01  -4.053 5.06e-05 ***
## PC13        -8.102e-01  1.111e-01  -7.292 3.06e-13 ***
## PC14         1.160e+00  1.143e-01  10.150  < 2e-16 ***
## PC15         1.021e-01  1.016e-01   1.005 0.314749
## PC16        -3.185e-03  1.730e-01  -0.018 0.985316
## PC17         1.273e+00  9.673e-02  13.161  < 2e-16 ***
## PC18         2.137e-01  8.841e-02   2.417 0.015650 *
## PC19        -1.674e-01  1.555e-01  -1.076 0.281776
## PC20         3.701e-01  1.084e-01   3.415 0.000638 ***
## PC21         5.330e-01  1.460e-01   3.650 0.000262 ***
## PC22         7.131e-01  9.395e-02   7.591 3.18e-14 ***
## PC23         7.788e-01  1.731e-01   4.498 6.85e-06 ***
## PC24        -2.707e-01  1.145e-01  -2.363 0.018117 *
## PC25         2.199e-01  1.624e-01   1.354 0.175776
## PC26        -5.311e-01  1.176e-01  -4.515 6.32e-06 ***
## PC27         1.121e+00  1.398e-01   8.017 1.09e-15 ***
```

```
## PC28         -4.965e-01  1.933e-01  -2.569 0.010208 *
## PC29          1.061e+00  1.681e-01   6.312 2.76e-10 ***
## PC30         -8.088e-01  2.786e-01  -2.903 0.003694 **
## PC31         -1.903e+00  1.862e-01 -10.223  < 2e-16 ***
## PC32         -1.789e+00  2.245e-01  -7.968 1.61e-15 ***
## PC33          3.575e-01  2.828e-01   1.264 0.206139
## PC34          1.494e+00  3.017e-01   4.952 7.36e-07 ***
## PC35          1.041e+00  3.310e-01   3.146 0.001654 **
## PC36         -2.093e+00  3.988e-01  -5.247 1.55e-07 ***
## PC37          4.983e+00  4.083e-01  12.203  < 2e-16 ***
## PC38          2.751e+00  4.971e-01   5.534 3.13e-08 ***
## PC39          1.498e+00  5.506e-01   2.721 0.006510 **
## PC40         -9.417e-01  5.924e-01  -1.590 0.111920
## PC41         -3.098e+00  9.496e-01  -3.262 0.001105 **
## PC42          6.135e+00  1.016e+00   6.040 1.54e-09 ***
## PC43         -3.057e+00  1.721e+00  -1.776 0.075689 .
## PC44          1.193e+01  1.202e+00   9.927  < 2e-16 ***
## PC45         -7.297e+00  2.455e+00  -2.973 0.002949 **
## PC46         -2.096e+01  2.945e+00  -7.119 1.09e-12 ***
## PC47          3.071e+00  2.662e+00   1.154 0.248628
## PC48         -1.140e+01  2.511e+00  -4.539 5.65e-06 ***
## PC49          1.627e+01  2.637e+00   6.171 6.80e-10 ***
## PC50          1.116e+01  2.194e+00   5.088 3.62e-07 ***
## PC51          3.659e+00  3.143e+00   1.164 0.244408
## PC52         -4.235e+00  5.930e+00  -0.714 0.475142
## PC53          3.849e+00  5.160e+00   0.746 0.455783
## PC54         -1.283e+01  5.764e+00  -2.226 0.026002 *
## PC55         -7.825e+01  7.196e+00 -10.875  < 2e-16 ***
## PC56          4.380e+01  8.420e+00   5.201 1.98e-07 ***
## PC57         -1.181e+00  1.308e+01  -0.090 0.928055
## PC58          8.699e+01  1.633e+01   5.329 9.89e-08 ***
## PC59          8.540e+01  1.619e+01   5.274 1.34e-07 ***
## PC60          4.123e+01  2.281e+01   1.807 0.070710 .
## PC61         -1.952e+02  3.273e+01  -5.964 2.46e-09 ***
## PC62         -1.187e+03  9.139e+01 -12.990  < 2e-16 ***
## PC63          2.385e+02  1.118e+02   2.133 0.032911 *
## PC64         -1.330e+02  1.937e+02  -0.687 0.492323
## PC65          4.019e+03  3.065e+02  13.112  < 2e-16 ***
## PC66          1.981e+02  4.332e+02   0.457 0.647511
## PC67         -7.602e+03  1.669e+03  -4.555 5.24e-06 ***
## PC68         -1.918e+04  2.349e+03  -8.163 3.27e-16 ***
## PC69         -3.789e+03  4.108e+03  -0.922 0.356356
```

```
## PC70          5.224e+04  9.070e+03   5.760 8.40e-09 ***
## PC71         -4.575e+04  2.876e+04  -1.591 0.111609    
## PC72          4.647e+05  1.076e+05   4.319 1.57e-05 ***
## PC73         -1.335e+06  1.438e+05  -9.284  < 2e-16 ***
## PC74          8.038e+06  1.481e+06   5.428 5.68e-08 ***
## PC75         -6.531e+13  8.727e+13  -0.748 0.454225    
## PC76          2.154e+14  9.709e+13   2.218 0.026543 *  
## PC77          5.082e+14  2.941e+14   1.728 0.083976 .  
## PC78         -3.492e+14  2.021e+14  -1.728 0.084001 .  
## PC79         -5.082e+12  3.419e+14  -0.015 0.988139    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 7321.0  on 5280  degrees of freedom
## Residual deviance: 2880.4  on 5201  degrees of freedom
## AIC: 3040.4
## 
## Number of Fisher Scoring iterations: 7
```

```
varImp(logistic)
```

```
##          Overall
## PC1   3.66783128
## PC2   0.43434722
## PC3   1.89604801
## PC4   1.74802113
## PC5   7.71818271
## PC6   3.50619755
## PC7   4.66338921
## PC8   6.60910971
## PC9  15.58671928
## PC10  2.15010089
## PC11 17.22364450
## PC12  4.05274382
## PC13  7.29186485
## PC14 10.14967671
## PC15  1.00530743
## PC16  0.01840491
## PC17 13.16056601
```

```
## PC18  2.41698040
## PC19  1.07633777
## PC20  3.41483438
## PC21  3.65007866
## PC22  7.59066799
## PC23  4.49827341
## PC24  2.36321434
## PC25  1.35387681
## PC26  4.51525925
## PC27  8.01658693
## PC28  2.56870174
## PC29  6.31150440
## PC30  2.90316135
## PC31 10.22282582
## PC32  7.96799845
## PC33  1.26425338
## PC34  4.95172278
## PC35  3.14626721
## PC36  5.24687627
## PC37 12.20348098
## PC38  5.53383533
## PC39  2.72090731
## PC40  1.58962374
## PC41  3.26229046
## PC42  6.03991331
## PC43  1.77626667
## PC44  9.92691658
## PC45  2.97304830
## PC46  7.11924101
## PC47  1.15368714
## PC48  4.53892340
## PC49  6.17073021
## PC50  5.08795439
## PC51  1.16403894
## PC52  0.71413827
## PC53  0.74580786
## PC54  2.22618572
## PC55 10.87465972
## PC56  5.20129645
## PC57  0.09029194
## PC58  5.32880014
## PC59  5.27362494
```

```
## PC60  1.80733524
## PC61  5.96408217
## PC62 12.98990098
## PC63  2.13316886
## PC64  0.68661935
## PC65 13.11197617
## PC66  0.45722316
## PC67  4.55502543
## PC68  8.16309518
## PC69  0.92233172
## PC70  5.76026831
## PC71  1.59100370
## PC72  4.31873864
## PC73  9.28363559
## PC74  5.42844111
## PC75  0.74839037
## PC76  2.21817257
## PC77  1.72806880
## PC78  1.72792938
## PC79  0.01486598
```

Parameter Tuning

```
#################################################### Tuning
# vectors for two parameters to be tuned
hvec <- seq(0.5, 2,length=5)
trgtvec <- seq(0.001, 0.01, length=4)
k <- 5 # k-fold CV
gam <- 0.01 # embargo parameter
run <- FALSE # whether run the grid search?
run <- TRUE
####################################################
if(run==TRUE)
{
  rst <- NULL
  for(ih in 1:length(hvec))
  {
    for(jtrgt in 1:length(trgtvec))
    {
      ####################################################
      i_CUSUM <- fmlr::istar_CUSUM(dat_used$Close, h=hvec[ih])  # <-
```

```
--------------- tuning parameter 1
      n_Event <- length(i_CUSUM)

      events <- data.frame(t0=i_CUSUM+1,
                           t1 = i_CUSUM+200,
                           trgt = rep(trgtvec[jtrgt], n_Event), # <-
--------------- tuning parameter 2
                           side=rep(1,n_Event))
      ptSl <- c(1,1)

      out0 <- fmlr::label_meta(dat_used$Close, events, ptSl)
      table(out0$label)

      # feature matrix
      fMat0 <- dat_used[out0$t1Fea,]

      # t1Fea and tLabel have to be included in order to use purged
k-CV
      allSet <- data.frame(Y=as.factor(out0$label),fMat0, t1Fea=out0
$t1Fea, tLabel=out0$tLabel)

      # exclude NA at the begining of the indicators
      idx_NA <- apply(allSet,1,function(x){sum(is.na(x))>0})
      allSet <- subset(allSet, !idx_NA)
      nx <- nrow(allSet)

      #####################################
      # prepare data for purged k-fold CV #
      #####################################
      CVobj <- fmlr::purged_k_CV(allSet, k=k, gam=gam)

      ##################
      ## randomforest ##
      ##################
      set.seed(1)
      for(i in 1:k)
      {
        trainSet <- CVobj[[i]]$trainSet
        trainSet <- trainSet[,!names(trainSet)%in%c("Close", "t1Fea"
, "tLabel")]

        testSet <- CVobj[[i]]$testSet
```

```
        testSet <- testSet[,!names(testSet)%in%c("Close", "t1Fea", "
tLabel")]

        # smote
        (tb <- table(trainSet$Y))
        (ratio <- tb[names(tb)=="1"] / tb[names(tb)=="0"])
        if(ratio > 1) perc <- list("0"=ratio, "1"=1) else perc <- li
st("0"=1, "1"=(1/ratio) )

        trainSet_balanced <- UBL::SmoteClassif(Y ~ ., dat = trainSet
, C.perc = perc)
        table(trainSet_balanced$Y)

        # automatically choose mtry
        # mtry <- tuneRF(trainSet_balanced[,-1], trainSet_balanced$Y
, plot = F)
        # mtry <- mtry[which.min(mtry[,2]),1]

        fit <- randomForest(Y ~ ., data = trainSet_balanced, importa
nce = FALSE, ntrees = 500) # use default mtry

        pre <- predict(fit, newdata = testSet) #predicted labels
        acc <- mean(testSet$Y==pre)

        # can also use R caret package to calculate F1 score
        # predictions <- predict(fit, newdata=testSet)
        precision <- posPredValue(pre, testSet$Y, positive="1")
        recall <- sensitivity(pre, testSet$Y, positive="1")
        F1 <- (2 * precision * recall) / (precision + recall)

        roc_prob <- predict(fit, newdata=testSet, type="prob")
        pred <- prediction(roc_prob[,2], testSet$Y)
        # the 2nd column is where the label "1" is
        # the default order of factors 0 and 1 is 0 < 1
        # so "1" is treated as positive, and a ligher prob.
        # means being closer to "1"

        auc <- tryCatch(performance(pred, measure = "auc")@y.values[
[1]],
                        error=function(e) NA, warning=function(w) NA
)
```

```
        # logloss / cross entropy loss
        logloss <- MLmetrics::LogLoss(roc_prob[,2], as.numeric(testS
et$Y==1))

        rst <- rbind(rst, c(ih, jtrgt, i, hvec[ih], trgtvec[jtrgt],
                            acc, auc, F1, logloss, table(trainSet$Y)
, table(testSet$Y), table(trainSet_balanced$Y)))
        cat(ih, jtrgt, i, hvec[ih], trgtvec[jtrgt], acc, auc, F1, lo
gloss,
            table(trainSet$Y), table(testSet$Y), table(trainSet_bala
nced$Y), "\n")
      }
    } # end of jtrgt loop
  } # end of ih loop

  rst <- data.frame(rst)
  names(rst) <- c("ih", "jtrgt", "iCV", "hCUSUM", "trgt", "acc", "au
c", "F1", "logloss",
                  "train0", "train1", "test0", "test1", "train_bal0"
, "train_bal1")
  write.csv(rst, "tuning_purgedCV_logloss-2021.csv", row.names = F)

}
```

```
## 1 1 1 0.5 0.001 0.9465479 0.7274786 0.97254 0.1863251 120 1625 19
430 1624 1625
## 1 1 2 0.5 0.001 0.922049 0.7043099 0.9594438 0.2440575 108 1642 2
6 423 1642 1642
## 1 1 3 0.5 0.001 0.8641425 0.835009 0.9257004 0.4002259 113 1657 1
8 431 1656 1657
## 1 1 4 0.5 0.001 0.7861915 0.6973068 0.8769231 0.5276717 114 1646
22 427 1646 1646
## 1 1 5 0.5 0.001 0.8802661 0.7018612 0.9363208 0.3408185 85 1706 5
4 397 1705 1706
## 1 2 1 0.5 0.004 0.8797327 0.5918451 0.9349398 0.4042885 319 1412
42 407 1411 1412
## 1 2 2 0.5 0.004 0.7550111 0.6601283 0.8578811 0.4469143 290 1435
63 386 1434 1435
## 1 2 3 0.5 0.004 0.5657016 0.6392221 0.6869984 0.7127793 268 1459
56 393 1458 1459
## 1 2 4 0.5 0.004 0.3340757 0.64375 0.3809524 1.064399 288 1465 65
```

```
384 1464 1465
## 1 2 5 0.5 0.004 0.7206208 0.557255 0.8333333 0.6041613 226 1557 1
35 316 1556 1557
## 1 3 1 0.5 0.007 0.8285078 0.6095482 0.9006452 0.571814 484 1246 6
2 387 1246 1246
## 1 3 2 0.5 0.007 0.7282851 0.6833156 0.8267045 0.5536874 437 1261
103 346 1261 1261
## 1 3 3 0.5 0.007 0.4565702 0.6267858 0.5378788 0.7990909 414 1303
83 366 1303 1303
## 1 3 4 0.5 0.007 0.3429844 0.6597233 0.256927 1.134714 414 1329 11
1 338 1329 1329
## 1 3 5 0.5 0.007 0.6563193 0.6416011 0.762634 0.6697295 359 1420 1
90 261 1419 1420
## 1 4 1 0.5 0.01 0.7349666 0.6802646 0.832158 0.6183699 637 1088 86
363 1088 1088
## 1 4 2 0.5 0.01 0.7126949 0.718498 0.8006182 0.5615654 566 1117 14
7 302 1117 1117
## 1 4 3 0.5 0.01 0.4476615 0.6034068 0.4655172 0.792238 549 1161 12
0 329 1160 1161
## 1 4 4 0.5 0.01 0.3853007 0.6770886 0.2290503 1.245097 553 1186 14
2 307 1186 1186
## 1 4 5 0.5 0.01 0.616408 0.6750394 0.7042735 0.7280787 495 1283 23
5 216 1283 1283
## 2 1 1 0.875 0.001 0.9495798 0.7328795 0.974026 0.182735 64 857 11
227 857 857
## 2 1 2 0.875 0.001 0.9327731 0.6732827 0.9652174 0.2461342 55 858
16 222 857 858
## 2 1 3 0.875 0.001 0.8739496 0.6729915 0.9315068 0.3863093 61 871
13 225 871 871
## 2 1 4 0.875 0.001 0.7731092 0.742491 0.868932 0.4909453 64 878 11
227 878 878
## 2 1 5 0.875 0.001 0.9004149 0.6950845 0.9475983 0.3063902 51 901
24 217 901 901
## 2 2 1 0.875 0.004 0.8823529 0.6069899 0.9369369 0.3902035 163 750
24 214 750 750
## 2 2 2 0.875 0.004 0.7941176 0.675895 0.8841608 0.4169666 149 743
32 206 742 743
## 2 2 3 0.875 0.004 0.6344538 0.6625 0.7535411 0.6439493 150 774 30
208 774 774
## 2 2 4 0.875 0.004 0.2436975 0.6472284 0.25 1.094655 151 784 33 20
5 783 784
## 2 2 5 0.875 0.004 0.7261411 0.6623172 0.8390244 0.567105 119 831
```

```
68 173 831 831
## 2 3 1 0.875 0.007 0.8529412 0.6801548 0.9148418 0.4476398 248 665
35 203 665 665
## 2 3 2 0.875 0.007 0.6806723 0.6795412 0.8 0.5365182 219 670 60 17
8 669 670
## 2 3 3 0.875 0.007 0.5882353 0.6730904 0.6918239 0.6812455 221 690
44 194 690 690
## 2 3 4 0.875 0.007 0.2647059 0.622182 0.1116751 1.074351 225 707 5
4 184 707 707
## 2 3 5 0.875 0.007 0.6514523 0.632295 0.7666667 0.6481227 193 755
92 149 755 755
## 2 4 1 0.875 0.01 0.7731092 0.6807447 0.856383 0.6027973 335 576 5
0 188 576 576
## 2 4 2 0.875 0.01 0.6638655 0.7284007 0.7727273 0.5884457 290 594
87 151 593 594
## 2 4 3 0.875 0.01 0.4663866 0.5604869 0.5448029 0.7345251 299 605
60 178 605 605
## 2 4 4 0.875 0.01 0.3403361 0.6782965 0.122905 1.079182 305 625 72
166 624 625
## 2 4 5 0.875 0.01 0.5311203 0.5865702 0.6686217 0.7399103 268 678
120 121 678 678
## 3 1 1 1.25 0.001 0.9605263 0.6495434 0.9798658 0.180556 39 553 6
146 552 553
## 3 1 2 1.25 0.001 0.9605263 0.6101598 0.9798658 0.2023527 37 547 6
146 547 547
## 3 1 3 1.25 0.001 0.8552632 0.5709459 0.9214286 0.3978358 41 558 4
148 557 558
## 3 1 4 1.25 0.001 0.7105263 0.7272727 0.8225806 0.5694272 36 568 9
143 568 568
## 3 1 5 1.25 0.001 0.869281 0.5902256 0.9300699 0.449675 25 583 20
133 583 583
## 3 2 1 1.25 0.004 0.8881579 0.6333702 0.9407666 0.3133227 106 480
13 139 479 480
## 3 2 2 1.25 0.004 0.8355263 0.6880682 0.906367 0.3980836 96 474 20
132 474 474
## 3 2 3 1.25 0.004 0.625 0.5930607 0.7574468 0.632831 96 496 16 136
496 496
## 3 2 4 1.25 0.004 0.2697368 0.5982517 0.283871 1.006761 93 503 22
130 502 503
## 3 2 5 1.25 0.004 0.6928105 0.7043651 0.8065844 0.6261458 71 536 4
8 105 535 536
## 3 3 1 1.25 0.007 0.8618421 0.7332168 0.9201521 0.3931838 163 422
```

```
22 130 422 422
## 3 3 2 1.25 0.007 0.7434211 0.747433 0.8340426 0.5000326 142 423 4
0 112 422 423
## 3 3 3 1.25 0.007 0.5592105 0.5961481 0.6731707 0.6952202 145 437
27 125 437 437
## 3 3 4 1.25 0.007 0.2894737 0.6677468 0.1940299 0.9778991 146 446
34 118 446 446
## 3 3 5 1.25 0.007 0.620915 0.658642 0.7478261 0.6618651 123 483 63
90 482 483
## 3 4 1 1.25 0.01 0.7828947 0.6714326 0.8653061 0.5378806 214 370 2
9 123 370 370
## 3 4 2 1.25 0.01 0.6578947 0.7479539 0.7636364 0.56553 182 378 56
96 378 378
## 3 4 3 1.25 0.01 0.4802632 0.519536 0.5635359 0.7499483 190 387 35
117 386 387
## 3 4 4 1.25 0.01 0.4013158 0.6544073 0.2834646 0.9853298 188 403 4
7 105 403 403
## 3 4 5 1.25 0.01 0.5620915 0.7033333 0.685446 0.6984709 166 438 78
75 437 438
## 4 1 1 1.625 0.001 0.9417476 0.7214286 0.97 0.2153325 21 380 5 98
379 380
## 4 1 2 1.625 0.001 0.9126214 0.6636905 0.9543147 0.2580052 17 373
7 96 373 373
## 4 1 3 1.625 0.001 0.9514563 0.5148515 0.9751244 0.3143214 22 369
2 101 369 369
## 4 1 4 1.625 0.001 0.9126214 0.5833333 0.9543147 0.2965675 22 384
3 100 383 384
## 4 1 5 1.625 0.001 0.9038462 0.5549708 0.9494949 0.3043294 17 394
9 95 393 394
## 4 2 1 1.625 0.004 0.8932039 0.6215054 0.9435897 0.342518 65 332 1
0 93 331 332
## 4 2 2 1.625 0.004 0.8543689 0.649679 0.9197861 0.4124518 59 328 1
4 89 328 328
## 4 2 3 1.625 0.004 0.6796117 0.6089744 0.797546 0.6151803 56 330 1
2 91 330 330
## 4 2 4 1.625 0.004 0.5339806 0.628663 0.6619718 0.710452 58 335 12
91 334 335
## 4 2 5 1.625 0.004 0.7403846 0.7243867 0.8508287 0.5409816 46 364
27 77 363 364
## 4 3 1 1.625 0.007 0.8737864 0.6868132 0.9273743 0.3991779 107 289
12 91 289 289
## 4 3 2 1.625 0.007 0.7087379 0.7271429 0.8192771 0.5070449 90 294
```

```
28 75 293 294
## 4 3 3 1.625 0.007 0.6407767 0.5846405 0.7516779 0.6454488 87 295
18 85 295 295
## 4 3 4 1.625 0.007 0.407767 0.6214927 0.440367 0.8707899 88 303 22
81 303 303
## 4 3 5 1.625 0.007 0.6153846 0.6853516 0.7619048 0.6574453 77 332
40 64 332 332
## 4 4 1 1.625 0.01 0.7864078 0.6691729 0.8658537 0.5171232 148 247
19 84 246 247
## 4 4 2 1.625 0.01 0.6407767 0.7121457 0.7730061 0.6195288 125 255
38 65 255 255
## 4 4 3 1.625 0.01 0.5728155 0.579191 0.6666667 0.6919464 123 256 2
7 76 256 256
## 4 4 4 1.625 0.01 0.4174757 0.5530303 0.3181818 0.893949 120 269 3
3 70 269 269
## 4 4 5 1.625 0.01 0.5384615 0.6678994 0.68 0.7128237 112 295 52 52
295 295
## 5 1 1 2 0.001 0.96 0.7986111 0.9795918 0.1754521 17 272 3 72 272
272
## 5 1 2 2 0.001 0.96 0.7685185 0.9795918 0.1507117 15 277 3 72 276
277
## 5 1 3 2 0.001 0.9466667 0.1609589 0.9726027 0.2525171 16 278 2 73
278 278
## 5 1 4 2 0.001 0.9066667 0.5014286 0.951049 0.352432 14 272 5 70 2
71 272
## 5 1 5 2 0.001 0.9066667 0.6060924 0.9503546 0.3214022 12 287 7 68
287 287
## 5 2 1 2 0.004 0.8933333 0.6940299 0.943662 0.3241498 51 238 8 67
238 238
## 5 2 2 2 0.004 0.8666667 0.6184615 0.9285714 0.4095463 47 240 10 6
5 240 240
## 5 2 3 2 0.004 0.76 0.4638462 0.8615385 0.5670327 43 237 10 65 237
237
## 5 2 4 2 0.004 0.5466667 0.5561538 0.6730769 0.6962343 42 242 10 6
5 242 242
## 5 2 5 2 0.004 0.72 0.5074956 0.8372093 0.6242655 37 262 21 54 261
262
## 5 3 1 2 0.007 0.84 0.6335227 0.9076923 0.4481415 80 206 11 64 206
206
## 5 3 2 2 0.007 0.6933333 0.6315789 0.8188976 0.5413576 70 214 19 5
6 213 214
## 5 3 3 2 0.007 0.68 0.5927778 0.7931034 0.6085855 67 211 15 60 211
```

```
211
## 5 3 4 2 0.007 0.5066667 0.6049696 0.5747126 0.7359551 66 218 17 5
8 218 218
## 5 3 5 2 0.007 0.6266667 0.6068216 0.7666667 0.6791999 60 238 29 4
6 238 238
## 5 4 1 2 0.01 0.76 0.7133333 0.8474576 0.4832997 101 185 15 60 185
185
## 5 4 2 2 0.01 0.64 0.802 0.7804878 0.5424208 88 196 25 50 195 196
## 5 4 3 2 0.01 0.5466667 0.5074956 0.6666667 0.6827297 84 191 21 54
190 191
## 5 4 4 2 0.01 0.44 0.5969125 0.4166667 0.8137944 83 199 22 53 199
199
## 5 4 5 2 0.01 0.6 0.6649928 0.7272727 0.6709207 80 217 34 41 216 2
17
```

Summarizing Performance From Tuning

```
perfCV <- read.csv("tuning_purgedCV_logloss-2021.csv", header = T)
perfCV
```

```
##      ih jtrgt iCV hCUSUM  trgt       acc       auc        F1   log
loss train0
## 1     1     1   1  0.500 0.001 0.9465479 0.7274786 0.9725400 0.186
3251    120
## 2     1     1   2  0.500 0.001 0.9220490 0.7043099 0.9594438 0.244
0575    108
## 3     1     1   3  0.500 0.001 0.8641425 0.8350090 0.9257004 0.400
2259    113
## 4     1     1   4  0.500 0.001 0.7861915 0.6973068 0.8769231 0.527
6717    114
## 5     1     1   5  0.500 0.001 0.8802661 0.7018612 0.9363208 0.340
8185     85
## 6     1     2   1  0.500 0.004 0.8797327 0.5918451 0.9349398 0.404
2885    319
## 7     1     2   2  0.500 0.004 0.7550111 0.6601283 0.8578811 0.446
9143    290
## 8     1     2   3  0.500 0.004 0.5657016 0.6392221 0.6869984 0.712
7793    268
## 9     1     2   4  0.500 0.004 0.3340757 0.6437500 0.3809524 1.064
3986    288
```

```
## 10    1     2   5  0.500 0.004 0.7206208 0.5572550 0.8333333 0.604
1613    226
## 11    1     3   1  0.500 0.007 0.8285078 0.6095482 0.9006452 0.571
8140    484
## 12    1     3   2  0.500 0.007 0.7282851 0.6833156 0.8267045 0.553
6874    437
## 13    1     3   3  0.500 0.007 0.4565702 0.6267858 0.5378788 0.799
0909    414
## 14    1     3   4  0.500 0.007 0.3429844 0.6597233 0.2569270 1.134
7143    414
## 15    1     3   5  0.500 0.007 0.6563193 0.6416011 0.7626340 0.669
7295    359
## 16    1     4   1  0.500 0.010 0.7349666 0.6802646 0.8321580 0.618
3699    637
## 17    1     4   2  0.500 0.010 0.7126949 0.7184980 0.8006182 0.561
5654    566
## 18    1     4   3  0.500 0.010 0.4476615 0.6034068 0.4655172 0.792
2380    549
## 19    1     4   4  0.500 0.010 0.3853007 0.6770886 0.2290503 1.245
0972    553
## 20    1     4   5  0.500 0.010 0.6164080 0.6750394 0.7042735 0.728
0787    495
## 21    2     1   1  0.875 0.001 0.9495798 0.7328795 0.9740260 0.182
7350     64
## 22    2     1   2  0.875 0.001 0.9327731 0.6732827 0.9652174 0.246
1342     55
## 23    2     1   3  0.875 0.001 0.8739496 0.6729915 0.9315068 0.386
3093     61
## 24    2     1   4  0.875 0.001 0.7731092 0.7424910 0.8689320 0.490
9453     64
## 25    2     1   5  0.875 0.001 0.9004149 0.6950845 0.9475983 0.306
3902     51
## 26    2     2   1  0.875 0.004 0.8823529 0.6069899 0.9369369 0.390
2035    163
## 27    2     2   2  0.875 0.004 0.7941176 0.6758950 0.8841608 0.416
9666    149
## 28    2     2   3  0.875 0.004 0.6344538 0.6625000 0.7535411 0.643
9493    150
## 29    2     2   4  0.875 0.004 0.2436975 0.6472284 0.2500000 1.094
6551    151
## 30    2     2   5  0.875 0.004 0.7261411 0.6623172 0.8390244 0.567
1050    119
```

```
## 31   2      3   1  0.875 0.007 0.8529412 0.6801548 0.9148418 0.447
6398    248
## 32   2      3   2  0.875 0.007 0.6806723 0.6795412 0.8000000 0.536
5182    219
## 33   2      3   3  0.875 0.007 0.5882353 0.6730904 0.6918239 0.681
2455    221
## 34   2      3   4  0.875 0.007 0.2647059 0.6221820 0.1116751 1.074
3505    225
## 35   2      3   5  0.875 0.007 0.6514523 0.6322950 0.7666667 0.648
1227    193
## 36   2      4   1  0.875 0.010 0.7731092 0.6807447 0.8563830 0.602
7973    335
## 37   2      4   2  0.875 0.010 0.6638655 0.7284007 0.7727273 0.588
4457    290
## 38   2      4   3  0.875 0.010 0.4663866 0.5604869 0.5448029 0.734
5251    299
## 39   2      4   4  0.875 0.010 0.3403361 0.6782965 0.1229050 1.079
1820    305
## 40   2      4   5  0.875 0.010 0.5311203 0.5865702 0.6686217 0.739
9103    268
## 41   3      1   1  1.250 0.001 0.9605263 0.6495434 0.9798658 0.180
5560     39
## 42   3      1   2  1.250 0.001 0.9605263 0.6101598 0.9798658 0.202
3527     37
## 43   3      1   3  1.250 0.001 0.8552632 0.5709459 0.9214286 0.397
8358     41
## 44   3      1   4  1.250 0.001 0.7105263 0.7272727 0.8225806 0.569
4272     36
## 45   3      1   5  1.250 0.001 0.8692810 0.5902256 0.9300699 0.449
6750     25
## 46   3      2   1  1.250 0.004 0.8881579 0.6333702 0.9407666 0.313
3227    106
## 47   3      2   2  1.250 0.004 0.8355263 0.6880682 0.9063670 0.398
0836     96
## 48   3      2   3  1.250 0.004 0.6250000 0.5930607 0.7574468 0.632
8310     96
## 49   3      2   4  1.250 0.004 0.2697368 0.5982517 0.2838710 1.006
7612     93
## 50   3      2   5  1.250 0.004 0.6928105 0.7043651 0.8065844 0.626
1458     71
## 51   3      3   1  1.250 0.007 0.8618421 0.7332168 0.9201521 0.393
1838    163
```

```
## 52   3     3   2  1.250 0.007 0.7434211 0.7474330 0.8340426 0.500
0326    142
## 53   3     3   3  1.250 0.007 0.5592105 0.5961481 0.6731707 0.695
2202    145
## 54   3     3   4  1.250 0.007 0.2894737 0.6677468 0.1940299 0.977
8991    146
## 55   3     3   5  1.250 0.007 0.6209150 0.6586420 0.7478261 0.661
8651    123
## 56   3     4   1  1.250 0.010 0.7828947 0.6714326 0.8653061 0.537
8806    214
## 57   3     4   2  1.250 0.010 0.6578947 0.7479539 0.7636364 0.565
5300    182
## 58   3     4   3  1.250 0.010 0.4802632 0.5195360 0.5635359 0.749
9483    190
## 59   3     4   4  1.250 0.010 0.4013158 0.6544073 0.2834646 0.985
3298    188
## 60   3     4   5  1.250 0.010 0.5620915 0.7033333 0.6854460 0.698
4709    166
## 61   4     1   1  1.625 0.001 0.9417476 0.7214286 0.9700000 0.215
3325     21
## 62   4     1   2  1.625 0.001 0.9126214 0.6636905 0.9543147 0.258
0052     17
## 63   4     1   3  1.625 0.001 0.9514563 0.5148515 0.9751244 0.314
3214     22
## 64   4     1   4  1.625 0.001 0.9126214 0.5833333 0.9543147 0.296
5675     22
## 65   4     1   5  1.625 0.001 0.9038462 0.5549708 0.9494949 0.304
3294     17
## 66   4     2   1  1.625 0.004 0.8932039 0.6215054 0.9435897 0.342
5180     65
## 67   4     2   2  1.625 0.004 0.8543689 0.6496790 0.9197861 0.412
4518     59
## 68   4     2   3  1.625 0.004 0.6796117 0.6089744 0.7975460 0.615
1803     56
## 69   4     2   4  1.625 0.004 0.5339806 0.6286630 0.6619718 0.710
4520     58
## 70   4     2   5  1.625 0.004 0.7403846 0.7243867 0.8508287 0.540
9816     46
## 71   4     3   1  1.625 0.007 0.8737864 0.6868132 0.9273743 0.399
1779    107
## 72   4     3   2  1.625 0.007 0.7087379 0.7271429 0.8192771 0.507
0449     90
```

```
## 73   4     3   3  1.625 0.007 0.6407767 0.5846405 0.7516779 0.645
4488      87
## 74   4     3   4  1.625 0.007 0.4077670 0.6214927 0.4403670 0.870
7899      88
## 75   4     3   5  1.625 0.007 0.6153846 0.6853516 0.7619048 0.657
4453      77
## 76   4     4   1  1.625 0.010 0.7864078 0.6691729 0.8658537 0.517
1232     148
## 77   4     4   2  1.625 0.010 0.6407767 0.7121457 0.7730061 0.619
5288     125
## 78   4     4   3  1.625 0.010 0.5728155 0.5791910 0.6666667 0.691
9464     123
## 79   4     4   4  1.625 0.010 0.4174757 0.5530303 0.3181818 0.893
9490     120
## 80   4     4   5  1.625 0.010 0.5384615 0.6678994 0.6800000 0.712
8237     112
## 81   5     1   1  2.000 0.001 0.9600000 0.7986111 0.9795918 0.175
4521      17
## 82   5     1   2  2.000 0.001 0.9600000 0.7685185 0.9795918 0.150
7117      15
## 83   5     1   3  2.000 0.001 0.9466667 0.1609589 0.9726027 0.252
5171      16
## 84   5     1   4  2.000 0.001 0.9066667 0.5014286 0.9510490 0.352
4320      14
## 85   5     1   5  2.000 0.001 0.9066667 0.6060924 0.9503546 0.321
4022      12
## 86   5     2   1  2.000 0.004 0.8933333 0.6940299 0.9436620 0.324
1498      51
## 87   5     2   2  2.000 0.004 0.8666667 0.6184615 0.9285714 0.409
5463      47
## 88   5     2   3  2.000 0.004 0.7600000 0.4638462 0.8615385 0.567
0327      43
## 89   5     2   4  2.000 0.004 0.5466667 0.5561538 0.6730769 0.696
2343      42
## 90   5     2   5  2.000 0.004 0.7200000 0.5074956 0.8372093 0.624
2655      37
## 91   5     3   1  2.000 0.007 0.8400000 0.6335227 0.9076923 0.448
1415      80
## 92   5     3   2  2.000 0.007 0.6933333 0.6315789 0.8188976 0.541
3576      70
## 93   5     3   3  2.000 0.007 0.6800000 0.5927778 0.7931034 0.608
5855      67
```

```
## 94   5     3   4  2.000 0.007 0.5066667 0.6049696 0.5747126 0.735
9551      66
## 95   5     3   5  2.000 0.007 0.6266667 0.6068216 0.7666667 0.679
1999      60
## 96   5     4   1  2.000 0.010 0.7600000 0.7133333 0.8474576 0.483
2997     101
## 97   5     4   2  2.000 0.010 0.6400000 0.8020000 0.7804878 0.542
4208      88
## 98   5     4   3  2.000 0.010 0.5466667 0.5074956 0.6666667 0.682
7297      84
## 99   5     4   4  2.000 0.010 0.4400000 0.5969125 0.4166667 0.813
7944      83
## 100  5     4   5  2.000 0.010 0.6000000 0.6649928 0.7272727 0.670
9207      80
##      train1 test0 test1 train_bal0 train_bal1
## 1      1625    19   430       1624       1625
## 2      1642    26   423       1642       1642
## 3      1657    18   431       1656       1657
## 4      1646    22   427       1646       1646
## 5      1706    54   397       1705       1706
## 6      1412    42   407       1411       1412
## 7      1435    63   386       1434       1435
## 8      1459    56   393       1458       1459
## 9      1465    65   384       1464       1465
## 10     1557   135   316       1556       1557
## 11     1246    62   387       1246       1246
## 12     1261   103   346       1261       1261
## 13     1303    83   366       1303       1303
## 14     1329   111   338       1329       1329
## 15     1420   190   261       1419       1420
## 16     1088    86   363       1088       1088
## 17     1117   147   302       1117       1117
## 18     1161   120   329       1160       1161
## 19     1186   142   307       1186       1186
## 20     1283   235   216       1283       1283
## 21      857    11   227        857        857
## 22      858    16   222        857        858
## 23      871    13   225        871        871
## 24      878    11   227        878        878
## 25      901    24   217        901        901
## 26      750    24   214        750        750
## 27      743    32   206        742        743
```

```
## 28     774   30  208        774        774
## 29     784   33  205        783        784
## 30     831   68  173        831        831
## 31     665   35  203        665        665
## 32     670   60  178        669        670
## 33     690   44  194        690        690
## 34     707   54  184        707        707
## 35     755   92  149        755        755
## 36     576   50  188        576        576
## 37     594   87  151        593        594
## 38     605   60  178        605        605
## 39     625   72  166        624        625
## 40     678  120  121        678        678
## 41     553    6  146        552        553
## 42     547    6  146        547        547
## 43     558    4  148        557        558
## 44     568    9  143        568        568
## 45     583   20  133        583        583
## 46     480   13  139        479        480
## 47     474   20  132        474        474
## 48     496   16  136        496        496
## 49     503   22  130        502        503
## 50     536   48  105        535        536
## 51     422   22  130        422        422
## 52     423   40  112        422        423
## 53     437   27  125        437        437
## 54     446   34  118        446        446
## 55     483   63   90        482        483
## 56     370   29  123        370        370
## 57     378   56   96        378        378
## 58     387   35  117        386        387
## 59     403   47  105        403        403
## 60     438   78   75        437        438
## 61     380    5   98        379        380
## 62     373    7   96        373        373
## 63     369    2  101        369        369
## 64     384    3  100        383        384
## 65     394    9   95        393        394
## 66     332   10   93        331        332
## 67     328   14   89        328        328
## 68     330   12   91        330        330
## 69     335   12   91        334        335
```

```
## 70      364    27    77          363         364
## 71      289    12    91          289         289
## 72      294    28    75          293         294
## 73      295    18    85          295         295
## 74      303    22    81          303         303
## 75      332    40    64          332         332
## 76      247    19    84          246         247
## 77      255    38    65          255         255
## 78      256    27    76          256         256
## 79      269    33    70          269         269
## 80      295    52    52          295         295
## 81      272     3    72          272         272
## 82      277     3    72          276         277
## 83      278     2    73          278         278
## 84      272     5    70          271         272
## 85      287     7    68          287         287
## 86      238     8    67          238         238
## 87      240    10    65          240         240
## 88      237    10    65          237         237
## 89      242    10    65          242         242
## 90      262    21    54          261         262
## 91      206    11    64          206         206
## 92      214    19    56          213         214
## 93      211    15    60          211         211
## 94      218    17    58          218         218
## 95      238    29    46          238         238
## 96      185    15    60          185         185
## 97      196    25    50          195         196
## 98      191    21    54          190         191
## 99      199    22    53          199         199
## 100     217    34    41          216         217
```

```
perfCV <- subset(perfCV, (!is.na(acc))&(!is.na(auc))&(!is.na(F1))&(!
is.na(logloss)))
dim(perfCV)
```

```
## [1] 100  15
```

```
cnt <- aggregate(perfCV$acc, by=list(perfCV$hCUSUM, perfCV$trgt), FU
N=length)
acc <- aggregate(perfCV$acc, by=list(perfCV$hCUSUM, perfCV$trgt), FU
N=mean)
auc <- aggregate(perfCV$auc, by=list(perfCV$hCUSUM, perfCV$trgt), FU
N=mean)
f1 <- aggregate(perfCV$F1, by=list(perfCV$hCUSUM, perfCV$trgt), FUN=
mean)
logloss <- aggregate(perfCV$logloss, by=list(perfCV$hCUSUM, perfCV$t
rgt), FUN=mean)
train1 <- aggregate(perfCV$train1, by=list(perfCV$hCUSUM, perfCV$trg
t), FUN=mean)
train0 <- aggregate(perfCV$train0, by=list(perfCV$hCUSUM, perfCV$trg
t), FUN=mean)
test1 <- aggregate(perfCV$test1, by=list(perfCV$hCUSUM, perfCV$trgt)
, FUN=mean)
test0 <- aggregate(perfCV$test0, by=list(perfCV$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,logloss, train1,train0,test1,te
st0))
names(mer) <- c("hCUSUM", "trgt", "kCV", "acc", "auc", "f1", "loglos
s",
                "train1","train0","test1", "test0")

tail(mer)
```

```
##    hCUSUM  trgt kCV       acc       auc        f1   logloss train
1 train0 test1
## 15  1.625 0.007   5 0.6492905 0.6610882 0.7401202 0.6159814  302.
6   89.8  79.2
## 16  1.625 0.010   5 0.5911875 0.6362879 0.6607417 0.6870742  264.
4  125.6  69.4
## 17  2.000 0.001   5 0.9360000 0.5671219 0.9666380 0.2505030  277.
2   14.8  71.0
## 18  2.000 0.004   5 0.7573333 0.5679974 0.8488116 0.5242457  243.
8   44.0  63.2
## 19  2.000 0.007   5 0.6693333 0.6139341 0.7722145 0.6026479  217.
4   68.6  56.8
## 20  2.000 0.010   5 0.5973333 0.6569469 0.6877103 0.6386331  197.
6   87.2  51.6
##    test0
## 15  24.0
## 16  33.8
## 17   4.0
## 18  11.8
## 19  18.2
## 20  23.4
```

```
dim(mer)
```

```
## [1] 20 11
```

```
# rstF1 <- mer[order(mer$f1, decreasing=T),]
# rstF1
rstlogloss <- mer[order(mer$logloss, decreasing=F),]
head(rstlogloss)
```

```
##    hCUSUM  trgt kCV       acc       auc        f1   logloss train
1 train0 test1
## 17  2.000 0.001   5 0.9360000 0.5671219 0.9666380 0.2505030  277.
2   14.8  71.0
## 13  1.625 0.001   5 0.9244586 0.6076549 0.9606498 0.2777112  380.
0   19.8  98.0
## 5   0.875 0.001   5 0.8859653 0.7033458 0.9374561 0.3225028  873.
0   59.0 223.6
## 1   0.500 0.001   5 0.8798394 0.7331931 0.9341856 0.3398198 1655.
2  108.0 421.6
## 9   1.250 0.001   5 0.8712246 0.6296295 0.9267621 0.3599693  561.
8   35.6 143.2
## 18  2.000 0.004   5 0.7573333 0.5679974 0.8488116 0.5242457  243.
8   44.0  63.2
##    test0
## 17   4.0
## 13   5.2
## 5   15.0
## 1   27.8
## 9    9.0
## 18  11.8
```