Homework #2

Larry Lei Hua

Feb 10, 2019

In total there are 30 pts:

(1) 5 pts for homework policy and presentation of the homework.
- If the files are not complete or not zipped into one file (should be one zipped file with one pdf and one Rmd), deduct 2 pts
- If the overall presentation is messy and not organized, deduct 3 pts

(2) 4 pts for Question 1
- The critical places to check: (a) the sample size of the derived unit bars is more than 1000; (b) the unit bars match the pattern shown in the solution.

(3) 4 pts for Question 2
- The critical places to check: (a) the sample size of the derived feature bars is more than 100; (b) the sampled feature bars match the pattern shown in the solution.

(4) 5 pts for Question 3
- The critical places to check: (a) whether a standard triple-barrier labeling is used (i.e., side = 0 for all); (b) the "trgt" should be chosen approrpiately so that there are enough labels created.
- If no effective labels have been obtained, deduct at least 2 pts.

(5) 12 pts for Question 4
- 4 pts for meta labeling, and the rule is similar to Question 3.
- 5 pts for correctly applying logistic regression (might be some other relevant models), including constructing feature matrix, model fitting, variable selection.
- 3 pts for evaluating model performance, at least including the following: (a) Confusion matrix (1 pt); (b) ROC curve and AUC (1 pt); (c) F1 score (1 pt)

(Please read the Homework Policy before you start)

Description: In this homework, you will practice implementing algorithms using R, conducting standard triple-barrier and meta labeling, and evaluating the performance of models you choose for the labeling techniques.

Dataset: Please download the tick data of E-Mini SP500 futures from here (yes, click on it). Please read the readme file, and the following questions will be based on the tick data "ESU13" of "ES_Trades.csv".

Practices:

1. Form dollar bars from the 'Unfiltered Price' of the dataset; choose an appropriate threshold so that you have at least thousands of bars to work with.

```
rm(list = setdiff(ls(), lsf.str()))
library(data.table)
library(lubridate)
options(digits.secs=3)

#' @param dat: dat input with at least the following columns: Price, Size
#' @param unit: the total dollar (unit) traded of each window
bar_unit <- function(dat, unit)
{
  cumUnit <- cumsum(dat$Size*dat$Price)
  winIdx <- as.factor(floor(cumUnit / unit))
  H <- aggregate(dat$Price, by = list(winIdx), max)$x
  L <- aggregate(dat$Price, by = list(winIdx), min)$x
  O <- aggregate(dat$Price, by = list(winIdx), function(x){x[1]})$x
  C <- aggregate(dat$Price, by = list(winIdx), function(x){x[length(x)]})$x
  V <- aggregate(dat$Size, by = list(winIdx), sum)$x
  list(H=H,L=L,O=O,C=C,V=V)
}

dat <- fread("~/Dropbox/Teaching/STAT430/datasets_/ES_Trades.csv",na.strings=NULL)
head(dat)
```

```
##    Symbol       Date         Time   Price Volume Market Flag
## 1:  ESU13 09/01/2013 17:00:00.083 1640.25      8           E
## 2:  ESU13 09/01/2013 17:00:00.083 1640.25      1           E
## 3:  ESU13 09/01/2013 17:00:00.083 1640.25      2           E
## 4:  ESU13 09/01/2013 17:00:00.083 1640.25      1           E
## 5:  ESU13 09/01/2013 17:00:00.083 1640.25      1           E
## 6:  ESU13 09/01/2013 17:00:00.083 1640.25     12           E
```

```
##    Sales Condition Exclude Record Flag Unfiltered Price
## 1:               0                  NA          1640.25
## 2:               0                  NA          1640.25
## 3:               0                  NA          1640.25
## 4:               0                  NA          1640.25
## 5:               0                  NA          1640.25
## 6:               0                  NA          1640.25
```

```
  dat$tStamp <- mdy(dat$Date, tz="US/Eastern") + hms(dat$Time)
  # for some unknown reason, 0.00000006 seconds missing after adding hms to mdy !!
  # > lubridate::second(dat$tStamp[1])
  # [1] 0.08299994

  levels(as.factor(dat$Symbol))
```

```
## [1] "ESU13" "ESZ13"
```

```
  # "ESU13" "ESZ13": sep/dec contracts, use ESU13 only in what follows
  dat <- subset(dat, Symbol=="ESU13")

  levels(as.factor(dat$Date))
```

```
##  [1] "09/01/2013" "09/02/2013" "09/03/2013" "09/04/2013" "09/05/2013"
##  [6] "09/06/2013" "09/08/2013" "09/09/2013" "09/10/2013" "09/11/2013"
## [11] "09/12/2013"
```

```
  ##############################################################################################
  # Nnote that there was no trading on Sep 7, 2013                                             #
  # For this homework, we just combine them together as a practice                             #
  # Otherwise the following can be used to split the data into training/validation/testing sets #
  ##############################################################################################
  dat$Type[dat$tStamp <=  mdy("09/07/2013", tz="US/Eastern")] <- "training"
  dat$Type[(dat$tStamp <=  mdy("09/10/2013", tz="US/Eastern")) & (dat$tStamp >  mdy("09/07/2013", tz="US/Eastern"))  ] <- "validation"
  dat$Type[(dat$tStamp <= mdy("09/14/2013", tz="US/Eastern")) & (dat$tStamp >  mdy("09/10/2013", tz="US/Eastern"))  ] <- "testing"
  table(dat$Type)
```

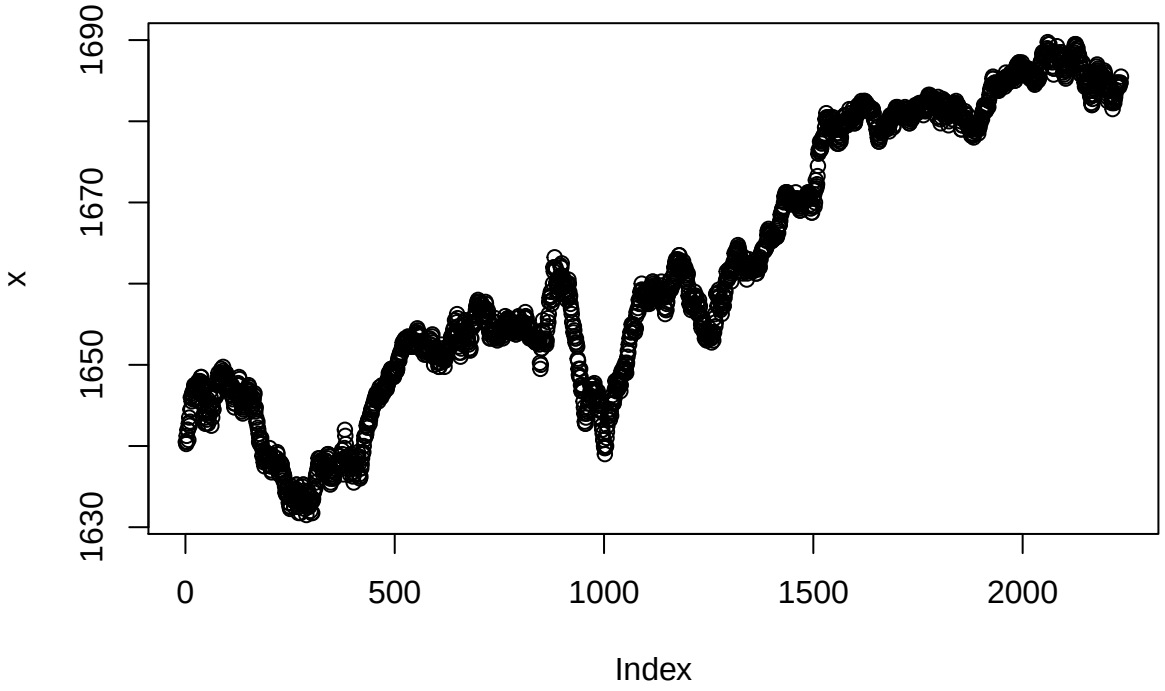
```
##
##    testing   training validation
##    1053148    1943231     357814
```

```
  # Use "Unfiltered Price" as "Price" for our functions
  # The "Price" in the original data is renamed as "filPrice"
```

```r
names(dat)[which(names(dat)=="Price")] <- "filPrice"
names(dat)[which(names(dat)=="Unfiltered Price")] <- "Price"
names(dat)[which(names(dat)=="Volume")] <- "Size"

unit_bar <- bar_unit(dat, unit = 10000000)
x <- unit_bar$C # closed price
v <- unit_bar$V # volume
length(x)
```

```
## [1] 2235
```

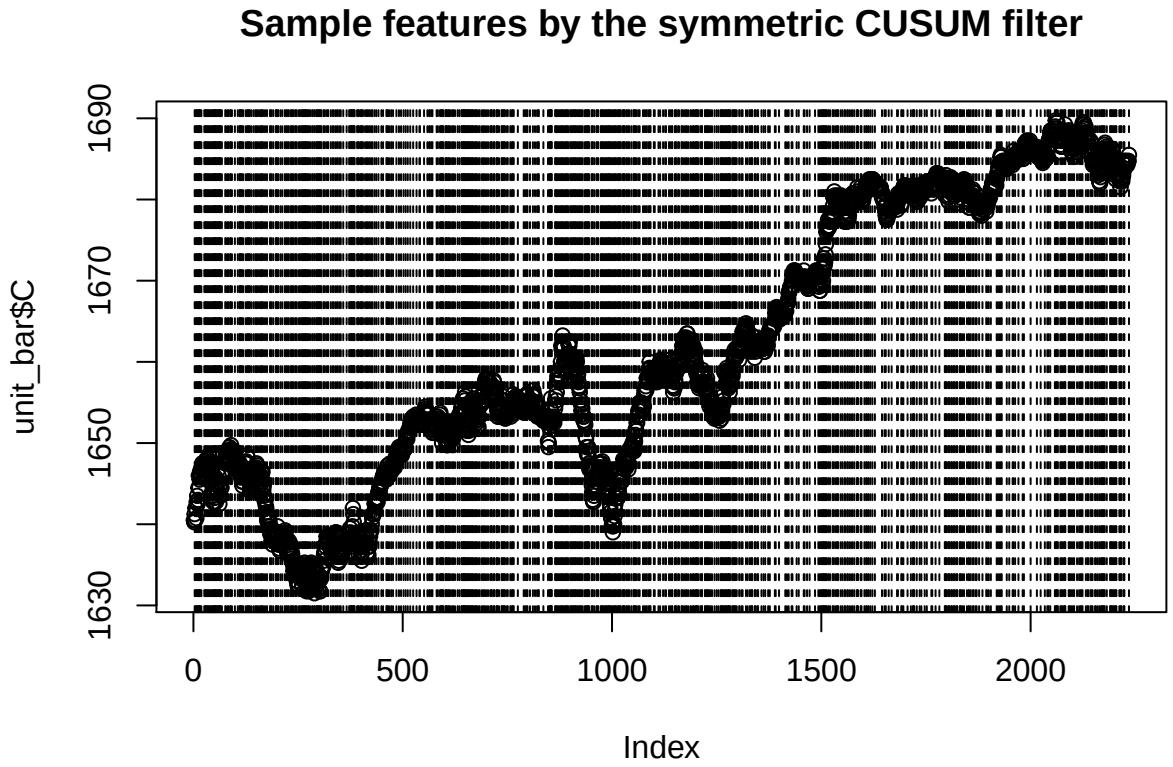
```r
plot(x)
```



2. Apply a symmetric CUSUM filter and set up a reasonable threshold so that you have at least hundreds of feature bars.

```r
#' ith point that triggers alarm for CUSUM filter
#' @param yvec: a y vector for time series of some key financial features
#' @param h: the threshold
```

```r
#' @examples
#' tick_bar <- bar_tick(dat, nTic=100)
#' i_CUSUM <- istar_CUSUM(tick_bar$C, h=4000)
#' plot(tick_bar$C, main="Sample feastures by the CUSUM filter")
#' abline(v=i_CUSUM, lty = 2)
istar_CUSUM <- function(yvec, h)
{
  S_pos <- S_neg <- 0
  istar <- NULL
  yminusEy <- diff(yvec)
  n <- length(yminusEy)
  for(i in 1:n)
  {
    S_pos <- max(0, S_pos + yminusEy[i])
    S_neg <- min(0, S_neg + yminusEy[i])
    if(max(S_pos, -S_neg) >= h) # note that Snippet 2.4 in AFML does not follow the definition of S_t
    {
      istar <- c(istar, i+1)
      S_pos <- S_neg <- 0
    }
  }
  return(istar)
}

# apply CUSUM filter
i_CUSUM <- istar_CUSUM(unit_bar$C, h=1)
n_Event <- length(i_CUSUM)
# [1] 642

plot(unit_bar$C, main="Sample features by the symmetric CUSUM filter")
abline(v=i_CUSUM, lty = 2)
```

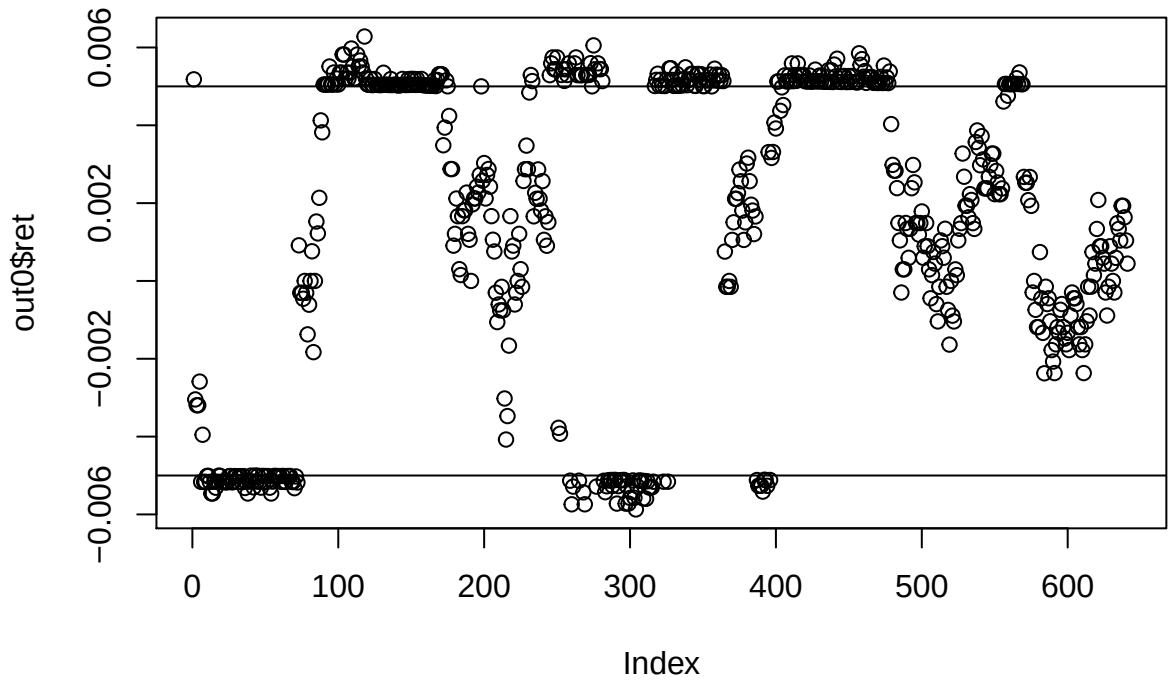


3. On those sampled features, apply the standard triple-barrier method, where ptSl = [1,1] and t1 can be set up as a reasonable value based on your preference.

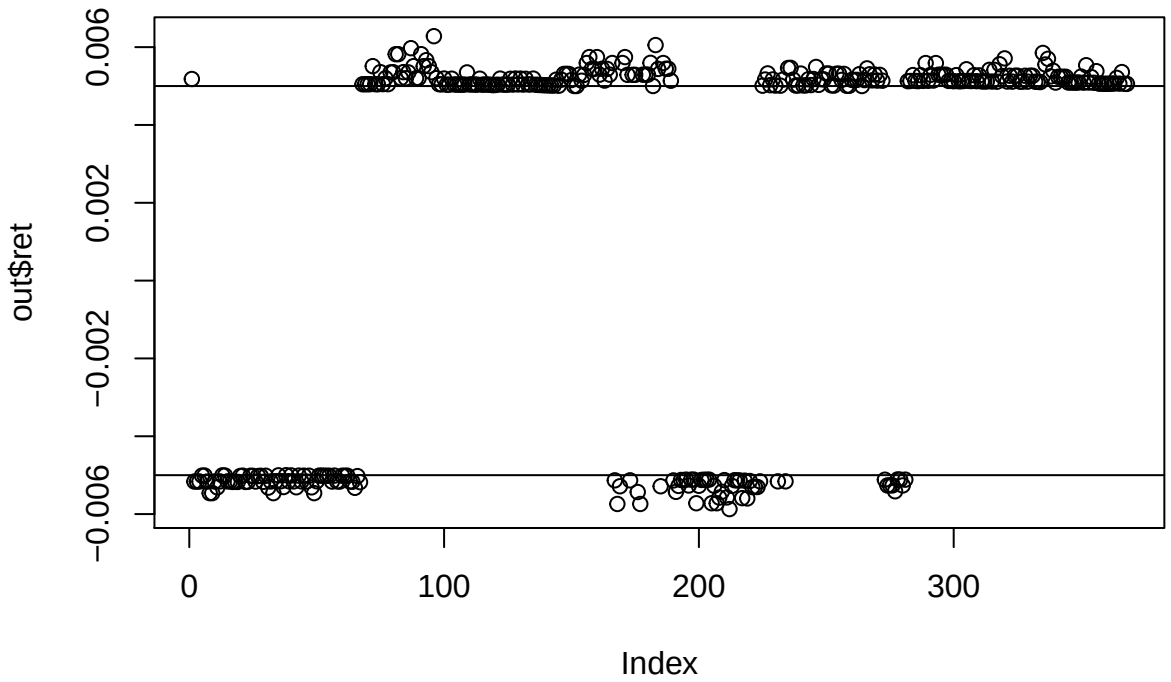
```
#' @param x: time series to be labeled
#' @param events: dataframe, has the following two columns:
#'                t0: t0 is event's start time index
#'                t1: t1 is event's end time index; Inf: no vertical barrier;
#'                trgt: the unit absolute return used to set up upper and lower barrier
#'                side: 0: no side; 1: up; -1: down
#' @param ptSl: a vector of two multipliers upper and lower barriers
label_meta <- function(x, events, ptSl)
{
  nBar <- length(x)
  t0 <- events$t0
  t1 <- sapply(events$t1, function(tt){min(tt,nBar)})
```

```
trgt <- events$trgt
side <- events$side
u <- ptSl[1]
l <- ptSl[2]

out <- sapply(1:length(t0),
              function(i)
              {
                i_t0 <- t0[i]
                i_t1 <- min(t1[i], length(x))
                i_x <- x[i_t0:i_t1]
                i_trgt <- trgt[i]
                i_side <- side[i]
                if(i_side==0)
                {
                  up <- i_trgt*u
                  lo <- i_trgt*l
                  isup <- (i_x/i_x[1]-1) >= up
                  islo <- -(i_x/i_x[1]-1) >= lo
                  T_up <- ifelse(sum(isup)>0, min(which(isup)), Inf)
                  T_lo <- ifelse(sum(islo)>0, min(which(islo)), Inf)
                }else if(i_side == 1)
                {
                  up <- i_trgt*u
                  isup <- (i_x/i_x[1]-1) >= up
                  T_up <- ifelse(sum(isup)>0, min(which(isup)), Inf)
                  T_lo <- Inf
                }else
                {
                  lo <- i_trgt*l
                  islo <- -(i_x/i_x[1]-1) >= lo
                  T_up <- Inf
                  T_lo <- ifelse(sum(islo)>0, min(which(islo)), Inf)
                }
                ret <- i_x[min(T_up, T_lo, length(i_x))] / i_x[1] - 1
                rst <- c(T_up, T_lo, length(i_x), ret)
                return(rst)
              }
```

```
  )
  out <- data.frame(t(out))
  names(out) <- c("T_up", "T_lo", "t1", "ret")
  return(out)
}

## Note that if trgt is too large, you may not get any samples
events <- data.frame(t0=i_CUSUM+1, t1 = i_CUSUM+200, trgt = rep(0.005, n_Event), side=rep(0,n_Event))
ptSl <- c(1,1)

## standard triple-barrier labeling
out0 <- label_meta(x, events, ptSl)
plot(out0$ret)
abline(h=events$trgt[1])
abline(h=-events$trgt[1])
```



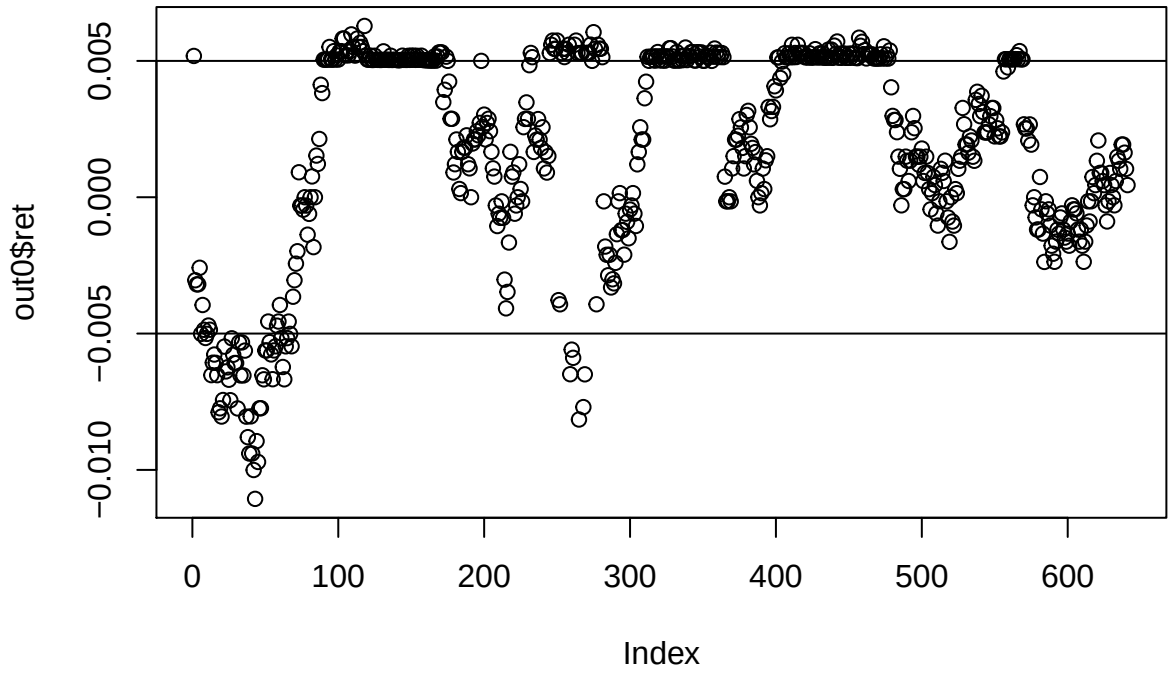
```r
# remove those with returns below targets
out <- subset(out0, ret>=events$trgt*ptSl[1] | ret<=-events$trgt*ptSl[2])
plot(out$ret)
abline(h=events$trgt[1])
abline(h=-events$trgt[1])
```



4. On those sampled features, apply the meta labeling method and only consider the labels associated with the upper barriers, where ptSl = [1,0] and t1 can be set up as a reasonable value based on your preference. After you get the labels and the feature bars, choose appropriate features from the feature bars as the predictors, and use the labels (either 1 or 0) from the meta labeling method to conduct a regression analysis. Your report for the analysis should include at least how you evaluate the performance of the models based on the confusion matrix, ROC curves, F1 scores, etc.

```r
events <- data.frame(t0=i_CUSUM+1, t1 = i_CUSUM+200, trgt = rep(0.005, n_Event), side=rep(1,n_Event))
ptSl <- c(1,1)

## meta labeling for side = rep(1,n_Event), i.e., only upper barrier (and the vertical barrier as always)
out0 <- label_meta(x, events, ptSl)
```

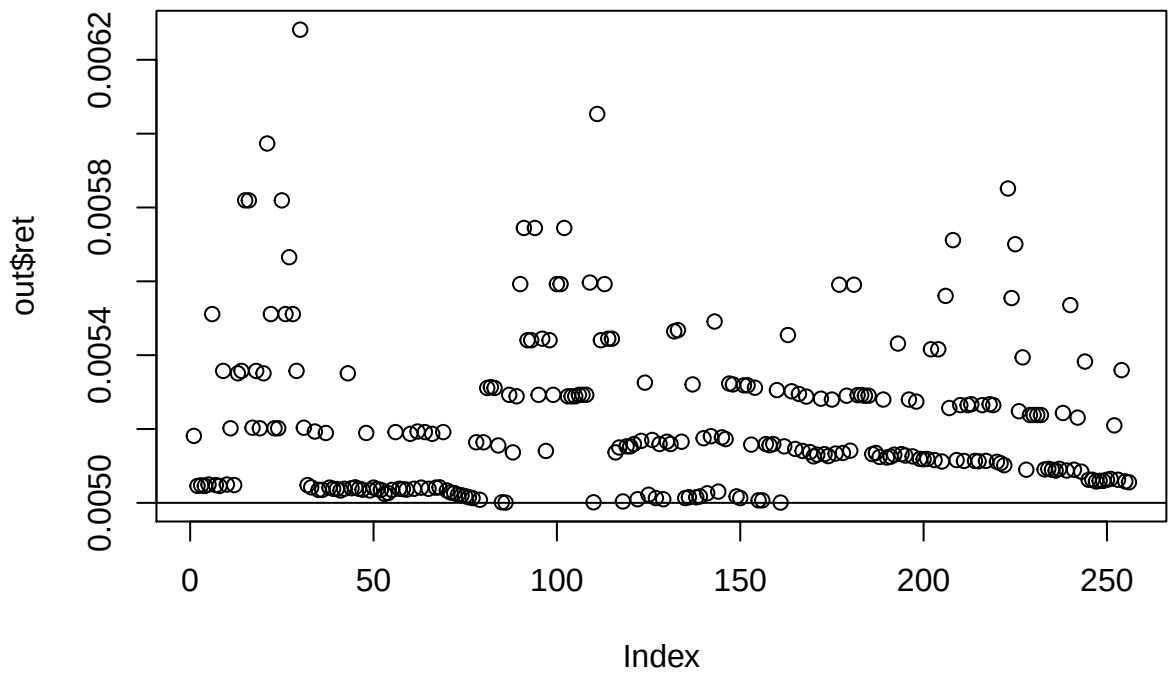
```
plot(out0$ret)
abline(h=events$trgt[1])
abline(h=-events$trgt[1])
```




```
# There are lots of unfavarable returns because the vertical barrier was hit
# remove those with returns below targets
out <- subset(out0, ret>=events$trgt*ptSl[1])
nrow(out) # sample size for regression analysis
```

```
## [1] 256
```

```
plot(out$ret)
abline(h=events$trgt[1])
```

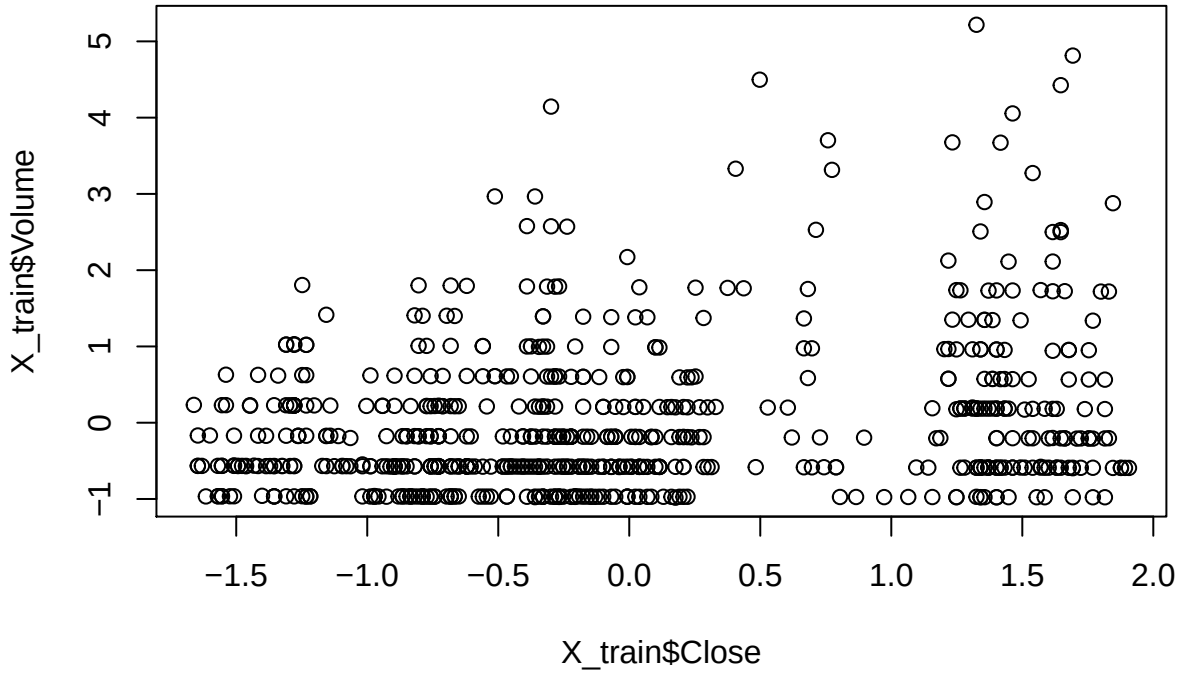



```
## construct X_train using various features obtained from sampled feature bars
# feature matrix with (C)losed price and (V)olume
# You can also try some other features
iTmp <- c(0, i_CUSUM)
fMat0 <- t(sapply(1:(length(i_CUSUM)),
                  function(i)
                  {
                    winTmp <- x[(iTmp[i]+1):(iTmp[i+1])]
                    C <- winTmp[length(winTmp)]
                    V <- sum(v[(iTmp[i]+1):(iTmp[i+1])])
                    return(c(C,V))
                  }
))
fMat0 <- data.frame(fMat0)
names(fMat0) <- c("Close", "Volume")

## no need to change the feature matrix, use fMat0
```

```r
fMat <- fMat0

## use logistic regression with out0
Y_train <- rep(0, n_Event)
Y_train[out0$ret>=events$trgt*ptSl[1]] <- 1 # 1 for positive resturns >= trgt*ptSl[1]
table(Y_train)/length(Y_train) # percentage of 0/1
```

```
## Y_train
##         0         1
## 0.6012461 0.3987539
```

```r
X_train <- data.frame(Close=scale(fMat$Close), Volume=scale(fMat$Volume))
plot(X_train$Volume ~ X_train$Close) # briefly check collinearity
```



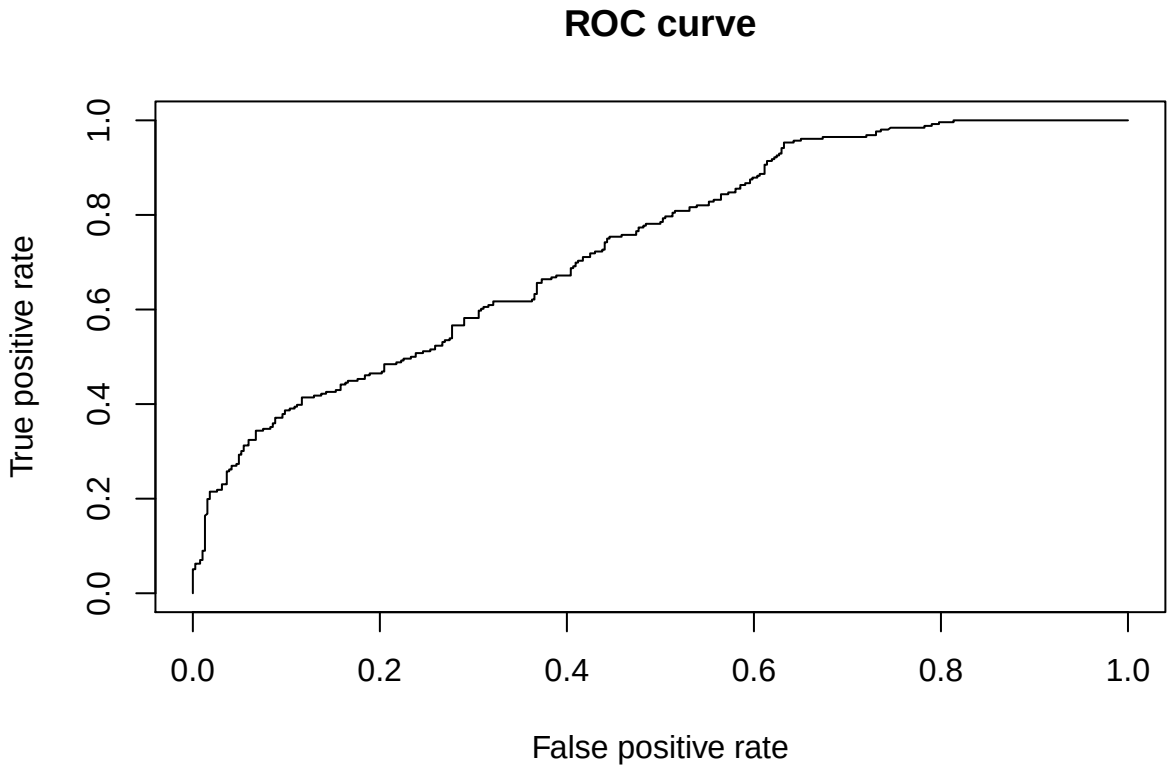

```r
## fit logistic regression
## Both Close and Volume were significant, and keep both
fit1 <- glm(Y_train ~ X_train$Close + X_train$Volume, family = "binomial")
summary(fit1)
```

```
##
## Call:
## glm(formula = Y_train ~ X_train$Close + X_train$Volume, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6493  -0.9996  -0.4748   1.0473   2.1006
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -0.53442    0.09232  -5.789 7.08e-09 ***
## X_train$Close  -1.05764    0.11069  -9.555  < 2e-16 ***
## X_train$Volume  0.24033    0.09314   2.580  0.00987 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 863.49  on 641  degrees of freedom
## Residual deviance: 743.54  on 639  degrees of freedom
## AIC: 749.54
##
## Number of Fisher Scoring iterations: 4
```

```
## Evaluating performance
pred <- predict(fit1, type="response")
# confusion matrix
table(Y_train, pred > 0.5)
```

```
##
## Y_train FALSE TRUE
##       0   290   96
##       1   125  131
```

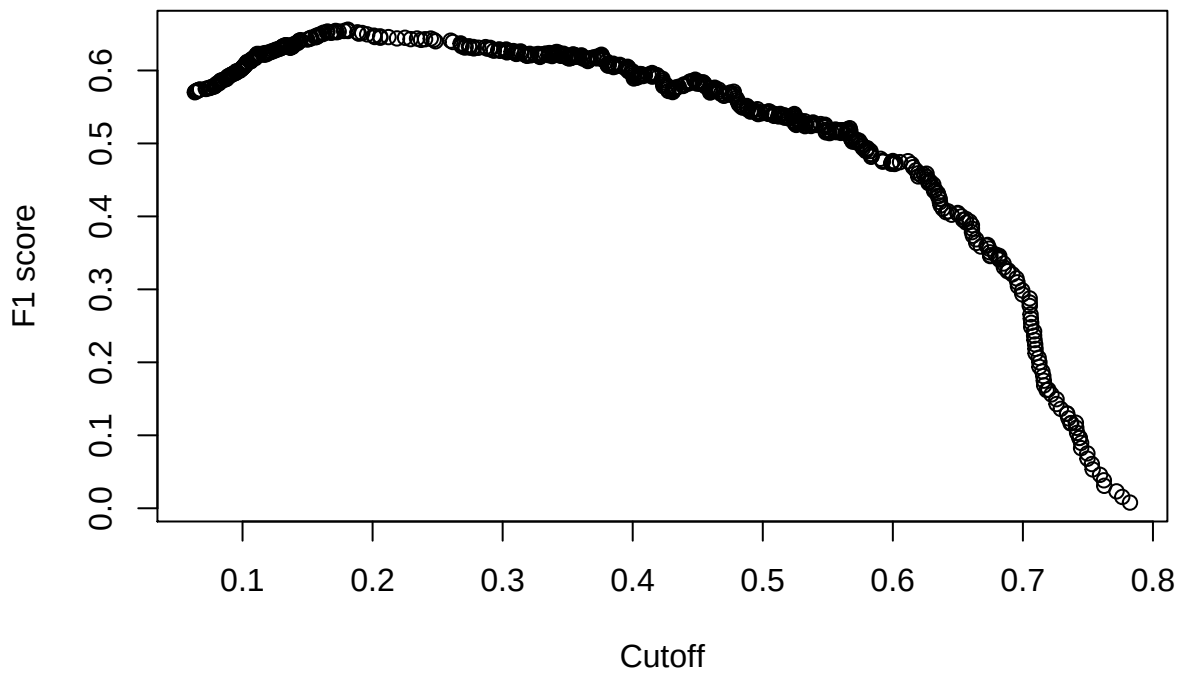
```
# ROC curve
library(ROCR)
ROC <- prediction(pred, Y_train)
ROC_perf <- performance(ROC, 'tpr','fpr')
plot(ROC_perf, main = "ROC curve")
```




```
  # AUC
  AUC <- performance(ROC, "auc")
  AUC@y.values[[1]]
```

```
## [1] 0.7337779
```

```
  # F1 score
  F1 <- performance(ROC, 'f')
  plot(F1@y.values[[1]][-1]~F1@x.values[[1]][-1], xlab="Cutoff", ylab="F1 score")
```




```
  (optCut <- F1@x.values[[1]][-1][which.max(F1@y.values[[1]][-1])])
```

```
##      474
## 0.181045
```

```
  table(Y_train, pred >= optCut)
```

```
##
## Y_train FALSE TRUE
##       0   142  244
##       1    12  244
```

```
  # Note that, F1 score is just a reference for modeling
  # For trading purpose, F1 is probably not a good metric as you might want to avoid any wrong buys
```