SPX Drawdown Prediction

Data Loading & Preprocessing

First, we need to load the data of *.csv* format into the R.

```
spx <- read.csv("spx.csv", header = F)
hly <- read.csv("hly.csv", header = F)
sox <- read.csv("sox.csv", header = F)
hg <- read.csv("hg.csv", header = F)
vix <- read.csv("vix.csv", header = F)
be5 <- read.csv("be5.csv", header = F)
yc2y10 <- read.csv("yc2y10.csv", header = F)
lmex <- read.csv("lmex.csv", header = F)
```

The loaded data are in the `data.frame` objects, and should be transformed into `xts` format in an automatic manner.

```
# First, convert the variables, which are stored as data frames, to "xts" objects.
fo2 <- quote(which(x == names(dats_xts)))
fo1 <-
  quote(dats <-
          list(
            spx = spx,
            sox = sox,
            yc2y10 = yc2y10,
            hly = hly,
            lmex = lmex,
            be5 = be5
          ))
eval(fo1)
dats_xts <-
  lapply(dats, function(x)
    xts(x$V2, order.by = as.Date(x$V1)))

# Second, assign the "xts" objects to the corresponding variables, and store them into a list.
texts <-
  lapply(names(dats), partial(Create_text, fo = fo2, str = "dats_xts"))
lapply(texts, Assign)
eval(fo1)
```

Now that the data are `xts` objects, we need to align by date for further processing. The easist way is to merge them using `merge.xts` method with "`join` = `inner`". Ignore any observations that contain missing values.

```
# Third, align the data into a data frame by inner join.
dats_daily <-
  as.data.frame(Reduce(function(x, y)
    na.omit(merge(x, y), join = 'inner'), dats))
colnames(dats_daily) <- names(dats)

# Fourth, extract the data out from the data frames,
# assign them to the corresponding variables,
# and then store them into a list.
texts <-
```

```
  lapply(names(dats), partial(Create_text, fo = fo2, str = "dats_daily"))
lapply(texts, Assign)
eval(fo1)

# Last, convert the variable into "xts" objects, and assign them back to the corresponding variables.
dats_xts <-
  lapply(dats, function(x)
    xts(x, order.by = as.Date(row.names(dats_daily))))
texts <-
  lapply(names(dats), partial(Create_text, fo = fo2, str = "dats_xts"))
lapply(texts, Assign)
eval(fo1)
```

Next, specify the parameters of interests, including the period of study, the period of length for computing the exponential moving average (EMA) of the series, the moving average period length for computing the RSI of the series, the threshold of **spx** drawdown considered, the forecast horizon and the length of the lookback period.

```
# Specify the period of interest.
d1 <- as.Date("2007-01-01") - 365
d2 <- as.Date("2016-12-31")
train_period <- c(d1, d2)
ma_len_signal <-
  60 # Set the length of moving average for generating the signals.
RSI_ma_len <-
  90 # Set the length of moving average for generating the RSI's.
thre_optimal <- 0.05

# Specify the forecast parameters.
forecastHorizon <- 60
backwardHorizons <- 180
```

Target & Predicting Variables Generation

Working on the target variable **spx** first, we need to filter all the dates in the whole dataset to locate those days when **spx** is on the rising trend. They can be screened according to the following criteria:

```
spx_EMA <- EMA(spx, n = ma_len_signal)
dat_spx <-
  as.data.table(window(
    merge(spx, spx_EMA, join = "inner"),
    start = d1,
    end = d2
  ))
MA_crossing <- dat_spx[spx - EMA > 0]
MA_crossing <- MA_crossing[index > d1 + 365]
MA_downcross <- MA_crossing[IsDown()]
names(MA_downcross) <- c("date", "spx", "spx_EMA")
```

In short, **spx** should be above its EMA line on particular day and as well as on all the past 10 days before it. Moreover, we restrict our focus on the most recent 10-year data, and truncate the dataset from *Jan 1, 2017*. The first and last five dates selected are displayed.

```
##             date     spx  spx_EMA
##    1: 2007-01-03 1416.60 1388.788
```

```
##    2: 2007-01-04 1418.34 1389.757
##    3: 2007-01-05 1409.71 1390.411
##    4: 2007-01-08 1412.84 1391.146
##    5: 2007-01-09 1412.11 1391.834
##   ---
## 1336: 2016-12-22 2260.96 2195.761
## 1337: 2016-12-23 2263.79 2197.992
## 1338: 2016-12-28 2249.92 2199.694
## 1339: 2016-12-29 2249.26 2201.319
## 1340: 2016-12-30 2238.83 2202.549
```

We then proceed to generate the target variable `y`, i.e. the relative return of `spx` from the day of interest and its minimum level within the next period of length `forecastHorizon`, which is 60 calendar days in our setting. Store the target variable `y` together with `MA_downcross` as a `data.table` object.

```
spx <- window(spx, start = d1 - 360, end = d2 + 120)
index <-
  index(spx[-((length(spx) - forecastHorizon):length(spx))])
y <-
  new(
    "SummaryWindow",
    data = spx,
    date = index,
    origin = 0,
    horizon = forecastHorizon
  )
wins <- CreateWins(y)
target <-
  data.table(date = wins@date, y = sapply(wins@windows, MinRet))
dat_MAcross <- merge(target, MA_downcross, by = "date")
```

After dealing with the target variable, we turn to work on the predictor variables, also known as X variables. We first generate separate `data.table` objects for each X variable and then merge them via `by = "date"`.

```
############################################################
UpDown <- list(
  spx = T,
  sox = T,
  be5 = T,
  hly = F,
  yc2y10 = T,
  lmex = T
)
dataList <- list()
dataListNames <- character(0)
dataAll <-
  list(
    spx = spx,
    sox = sox,
    lmex = lmex,
    hly = hly,
    be5 = be5,
    yc2y10 = yc2y10
  )
dataNames <- names(dataAll)
xThresholds <-
```

```
  c(
    sox = 7,
    be5 = 7,
    hly = 7,
    yc2y10 = 7,
    lmex = 7
  )
############################################################
for (xName in dataNames) {
  x <- as.xts(dataAll[[xName]])
  isDown <- UpDown[[xName]]
  x_rsi <- na.omit(RSI(x, n = RSI_ma_len))

  data <- x_rsi
  names(data) <- paste0(xName, "_rsi")
  get_data_input <- new(
    "InputsForGetData",
    data = data,
    period = train_period,
    backwardHorizons = backwardHorizons,
    ma_len = 1,
    is_down = isDown
  )

  mrX_mrY <- GetRet(get_data_input)
  names(mrX_mrY) <- Name_data.table(xName)

  data_EMA <- Generate_EMA(
    dataset = list(x_rsi),
    dataset_name = paste0(xName, "_rsi_EMA"),
    ma_lens = c(ma_len_signal),
    external_dataset = list(x, x_rsi),
    external_dataset_name = c(xName, paste0(xName, "_rsi")),
    d_start = first(mrX_mrY$date),
    d_end = last(mrX_mrY$date)
  )
  dat <- merge(data_EMA, mrX_mrY, by = "date")

  dataList <- list.append(dataList, dat)
  dataListNames <- append(dataListNames, xName)
}
names(dataList) <- dataListNames
dataTableAll <- Reduce(partial(merge, by = "date"), dataList)
dataTableAll <- dataTableAll[date > as.Date("2007-01-01")]
```

Here is a brief summary on `dataTableAll`:

```
##             date     spx  spx_rsi spx_rsi_EMA min_ret_spx_rsi
##    1: 2007-01-03 1416.60 57.85101    57.45333      0.14203041
##    2: 2007-01-04 1418.34 57.99385    57.47106      0.28486388
##    3: 2007-01-05 1409.71 57.02463    57.45642      0.00000000
##    4: 2007-01-08 1412.84 57.28644    57.45085      0.26181181
##    5: 2007-01-09 1412.11 57.20425    57.44276      0.17961992
##   ---
```

```
## 2521: 2017-03-21 2344.02 57.00807    57.93412      0.00000000
## 2522: 2017-03-22 2348.45 57.23998    57.91136      0.23191815
## 2523: 2017-03-23 2345.96 57.06501    57.88361      0.05694725
## 2524: 2017-03-24 2343.98 56.92509    57.85218      0.00000000
## 2525: 2017-03-27 2341.59 56.75522    57.81622      0.00000000
##       min_date_spx_rsi max_ret_spx_rsi max_date_spx_rsi       sox  sox_rsi
##    1:       2006-12-22        1.841756       2006-12-15  465.0600 50.13627
##    2:       2006-12-22        1.698923       2006-12-15  474.4200 51.08130
##    3:       2007-01-05        2.668136       2006-12-15  469.0900 50.52984
##    4:       2007-01-05        2.406324       2006-12-15  471.9400 50.81694
##    5:       2007-01-05        2.488516       2006-12-15  475.3800 51.16289
##   ---
## 2521:       2017-03-21        4.474197       2017-03-01  988.4229 59.12640
## 2522:       2017-03-21        4.242279       2017-03-01  999.3426 59.80062
## 2523:       2017-03-21        4.417250       2017-03-01  996.8474 59.57355
## 2524:       2017-03-24        4.557173       2017-03-01 1004.3505 60.03500
## 2525:       2017-03-27        4.727040       2017-03-01 1005.9529 60.13328
##       sox_rsi_EMA min_ret_sox_rsi min_date_sox_rsi max_ret_sox_rsi
##    1:    50.60697       0.1237879       2006-12-22        2.923311
##    2:    50.62252       1.0688118       2006-12-22        1.978287
##    3:    50.61948       0.5173571       2006-12-22        2.529741
##    4:    50.62596       0.8044574       2006-12-22        2.242641
##    5:    50.64356       1.1504019       2006-12-22        1.896697
##   ---
## 2521:    59.68590       0.0000000       2017-03-21        2.270634
## 2522:    59.68967       0.6742255       2017-03-21        1.596408
## 2523:    59.68586       0.4471539       2017-03-21        1.823480
## 2524:    59.69731       0.9086041       2017-03-21        1.362030
## 2525:    59.71160       1.0068812       2017-03-21        1.263753
##       max_date_sox_rsi   lmex lmex_rsi lmex_rsi_EMA min_ret_lmex_rsi
##    1:       2006-11-20 3545.6 49.95721     53.48908        0.0000000
##    2:       2006-11-20 3515.9 49.63646     53.36277        0.0000000
##    3:       2006-11-20 3416.3 48.57875     53.20592        0.0000000
##    4:       2006-11-20 3362.0 48.01468     53.03571        0.0000000
##    5:       2006-11-20 3340.6 47.79351     52.86384        0.0000000
##   ---
## 2521:       2017-02-21 2838.1 54.56784     55.84812        1.2229708
## 2522:       2017-02-21 2852.7 54.88242     55.81646        1.5375444
## 2523:       2017-02-21 2854.1 54.91269     55.78683        1.5678164
## 2524:       2017-02-21 2846.2 54.70325     55.75130        1.3583763
## 2525:       2017-02-21 2816.2 53.91360     55.69105        0.5687309
##       min_date_lmex_rsi max_ret_lmex_rsi max_date_lmex_rsi  hly  hly_rsi
##    1:        2007-01-03         8.692396        2006-07-12 2.91 44.23439
##    2:        2007-01-04         9.013144        2006-07-12 2.95 45.01133
##    3:        2007-01-05        10.070858        2006-07-12 2.93 44.69647
##    4:        2007-01-08        10.634930        2006-07-12 2.92 44.53894
##    5:        2007-01-09        10.426277        2006-07-14 2.89 44.06777
##   ---
## 2521:        2017-03-09         7.757778        2016-11-28 4.04 44.05104
## 2522:        2017-03-09         7.443204        2016-11-28 4.16 45.36695
## 2523:        2017-03-09         7.412932        2016-11-28 4.11 44.92178
## 2524:        2017-03-09         7.622373        2016-11-28 4.07 44.56798
## 2525:        2017-03-09         8.412018        2016-11-28 4.12 45.11439
##       hly_rsi_EMA min_ret_hly_rsi min_date_hly_rsi max_ret_hly_rsi
```

```
##    1:    46.52605        1.810012       2006-12-29       0.0000000
##    2:    46.47639        2.586949       2006-12-29       0.0000000
##    3:    46.41803        2.272091       2006-12-29       0.3148584
##    4:    46.35642        2.114563       2006-12-29       0.4723857
##    5:    46.28138        1.643394       2006-12-29       0.9435556
##   ---
## 2521:    40.69334        6.376909       2017-03-02       0.0000000
## 2522:    40.84657        7.692818       2017-03-02       0.0000000
## 2523:    40.98018        7.247642       2017-03-02       0.4451763
## 2524:    41.09782        6.893842       2017-03-02       0.7989762
## 2525:    41.22951        7.440256       2017-03-02       0.2525621
##       max_date_hly_rsi    be5  be5_rsi be5_rsi_EMA min_ret_be5_rsi
##    1:       2007-01-03 2.2534 47.54914    44.50725      8.18789736
##    2:       2007-01-04 2.1892 45.96379    44.55501      6.60255397
##    3:       2007-01-04 2.2002 46.27416    44.61137      6.91291993
##    4:       2007-01-04 2.2222 46.89109    44.68612      7.52985447
##    5:       2007-01-04 2.2143 46.69638    44.75203      7.33514100
##   ---
## 2521:       2017-03-21 1.9730 56.09974    57.94477      0.00000000
## 2522:       2017-03-22 1.9645 55.84306    57.87587      0.00000000
## 2523:       2017-03-22 1.9532 55.50167    57.79802      0.00000000
## 2524:       2017-03-22 1.9584 55.62791    57.72687      0.12623138
## 2525:       2017-03-22 1.9547 55.51459    57.65434      0.01291787
##       min_date_be5_rsi max_ret_be5_rsi max_date_be5_rsi  yc2y10 yc2y10_rsi
##    1:       2006-10-31        4.480836       2006-08-09  -9.937   47.54306
##    2:       2006-10-31        6.066180       2006-08-09  -9.419   47.76544
##    3:       2006-10-31        5.755814       2006-08-09 -10.842   47.20948
##    4:       2006-10-31        5.138879       2006-08-09 -12.936   46.40573
##    5:       2006-10-31        5.333593       2006-08-09 -13.708   46.11304
##   ---
## 2521:       2017-03-21        5.113030       2017-02-02 115.362   50.67860
## 2522:       2017-03-22        5.369710       2017-02-02 115.318   50.66559
## 2523:       2017-03-23        5.711099       2017-02-02 116.127   50.89983
## 2524:       2017-03-23        5.584867       2017-02-02 115.158   50.60879
## 2525:       2017-03-23        5.698181       2017-02-02 112.142   49.71407
##       yc2y10_rsi_EMA min_ret_yc2y10_rsi min_date_yc2y10_rsi
##    1:       45.81000          5.2129332          2006-11-15
##    2:       45.87411          5.4353077          2006-11-15
##    3:       45.91790          4.8793536          2006-11-15
##    4:       45.93389          4.0756033          2006-11-15
##    5:       45.93976          3.7829127          2006-11-15
##   ---
## 2521:       52.86836          0.4209848          2017-02-28
## 2522:       52.79614          0.4079815          2017-02-28
## 2523:       52.73397          0.6422210          2017-02-28
## 2524:       52.66429          0.3511735          2017-02-28
## 2525:       52.56756          0.0000000          2017-03-27
##       max_ret_yc2y10_rsi max_date_yc2y10_rsi
##    1:          0.6921034          2006-08-09
##    2:          0.4697289          2006-08-09
##    3:          1.0256831          2006-08-09
##    4:          1.8294333          2006-08-09
##    5:          2.1221239          2006-08-09
##   ---
```

```
## 2521:          7.5984368          2016-12-22
## 2522:          7.6114401          2016-12-22
## 2523:          7.3772006          2016-12-22
## 2524:          7.6682481          2016-12-22
## 2525:          8.5629676          2016-12-22
```

Note that we also need to combine the X varibales with the target variable, i.e. to merge `dataTableAll` with `dat_MAcross`.

```
d <- merge(dat_MAcross, dataTableAll, by = "date")
```

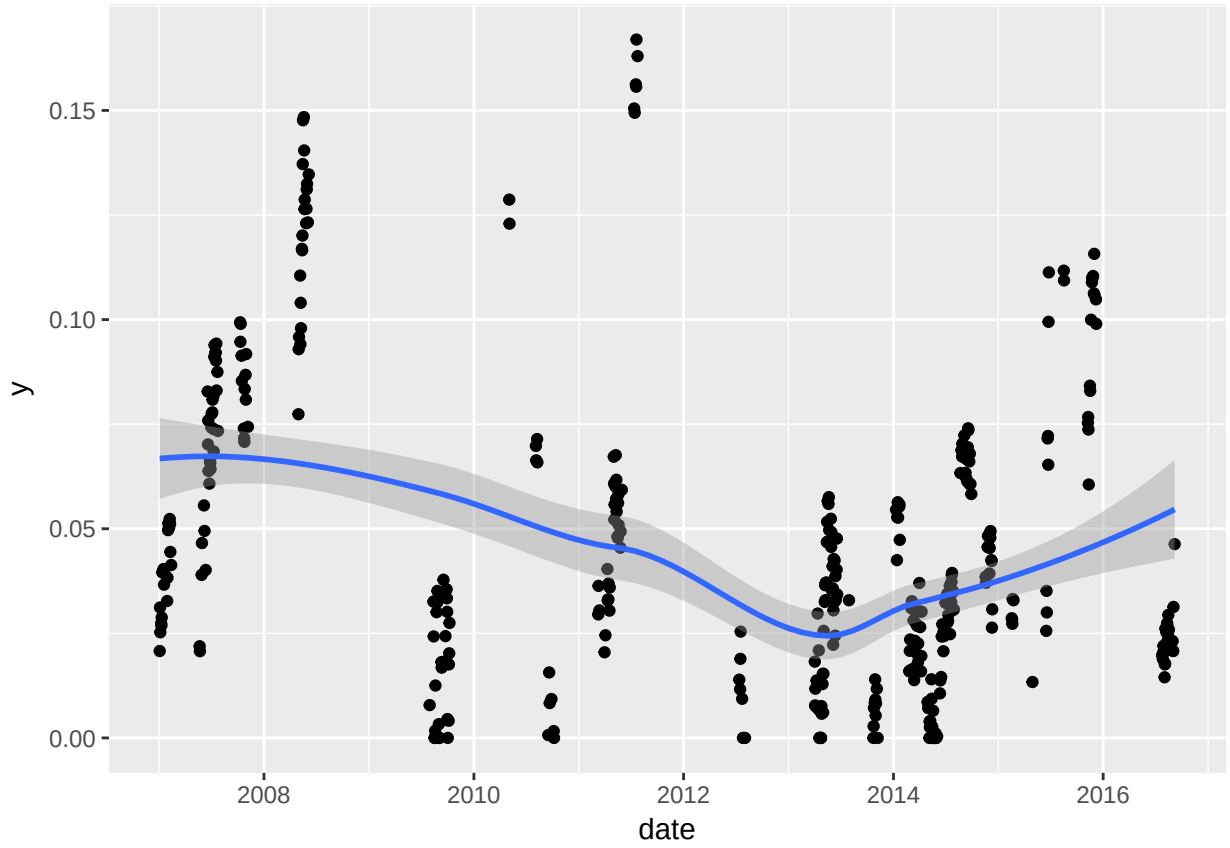
The meaning of each variable in `d` is pretty self-explained. Variable `min_ret_X` refers to the relative return between the current level of X and the minimum level of X over the past lookback period of length `backwardHorizons`, which is 180 calendar days in our setting, where X is a generic term and can be used to represent varibales such as `sox`, `hly`, `spx`, `be5`, `yc2y10` and `lmex`. The meaning of `max_ret_X` can be inferred by the same token. Variable `min_date_X` refers to the date when variables X attains the minimum level within the stipulated lookback period. Same can be said on `max_date_X`.

Feature Extraction

With over 1000 observations in the dataset, we need to filter the data such that we only select those days when X variables have gone down from their most recent peak (or gone up from the most recent trough in the case of `hly`), while `spx` remains an upward trend.

The resulting dataset, named `data_indicator`, has 409 rows (observations) and 46 columns (variables), and can be visualized using the following graph:

```
ggplot(data = data_indicator, aes(x = date, y = y)) +
  geom_point() +
  geom_smooth()
```



Note that `geom_smooth()` uses `method = 'loess'`.

Generate the binary target variable `y_binary` via

```
dat <- within(data_indicator, {
  y_binary <- as.factor(ifelse(y > thre_optimal, 1, 0))
})
```

Feature engineering is conducted to extract information that could be useful for predicting `y_binary`.

Classification Using Random Forest

First, we divide the whole period of study into four parts with starting and ending dates as listed below.

```
startDays <- c("2007-01-01", "2008-01-01", "2012-01-01", "2015-01-01")
endDays <- c("2007-12-31", "2011-12-31", "2014-12-31", "2016-12-31")
```

The back testing procedure is conducted by the following way: a testing period is first selected for a given position in the vectors `startDays` and `endDays`; the data corresponding to the remaining period are automatically used as the training data. Yet we should check whether there exists any look-ahead bias in the training dataset beforehand.

```
print(as.data.frame(dat[date > "2007-12-01" & date < "2008-03-01"][, 1 : 3]))
```

```
## [1] date     y_binary y
## <0 rows> (or 0-length row.names)
```

```
print(as.data.frame(dat[date > "2011-12-01" & date < "2012-03-01"][, 1 : 3]))
```

```
## [1] date     y_binary y
```

```
## <0 rows> (or 0-length row.names)
```

```
print(as.data.frame(dat[date > "2014-12-01" & date < "2015-03-01"][, 1 : 3]))
```

```
##          date y_binary          y
## 1  2014-12-02        0 0.04539450
## 2  2014-12-03        0 0.04897485
## 3  2014-12-04        0 0.04786864
## 4  2014-12-05        0 0.04945142
## 5  2014-12-08        0 0.04250331
## 6  2014-12-09        0 0.04227554
## 7  2014-12-10        0 0.02635553
## 8  2014-12-11        0 0.03075177
## 9  2015-02-17        0 0.02861441
## 10 2015-02-18        0 0.02830908
## 11 2015-02-19        0 0.02727598
## 12 2015-02-20        0 0.03319907
## 13 2015-02-23        0 0.03290578
```

```
as.Date("2014-12-11") + 60
```

```
## [1] "2015-02-09"
```

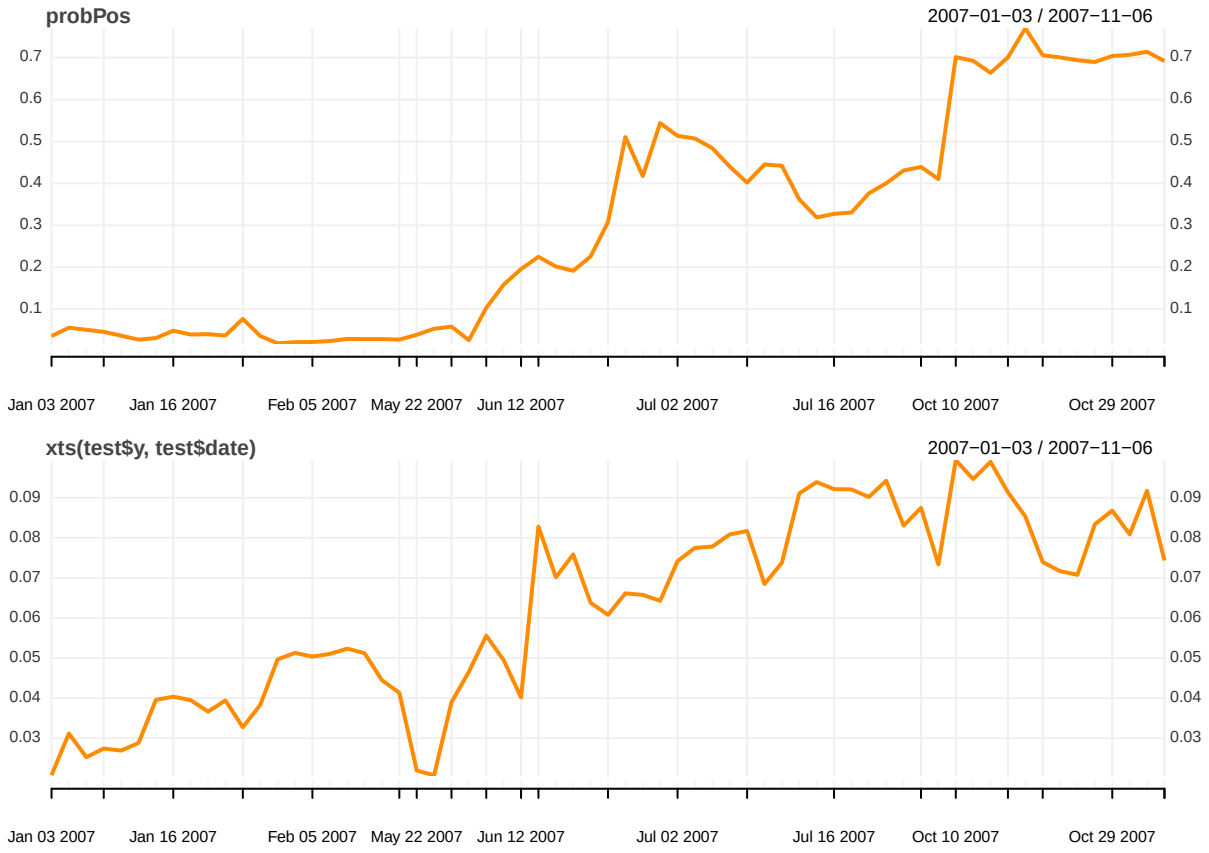
Apparently, there is no evidence of overlapping information contained in the training and testing datasets. A random forest model of size 10000 is built for each of the testing periods.

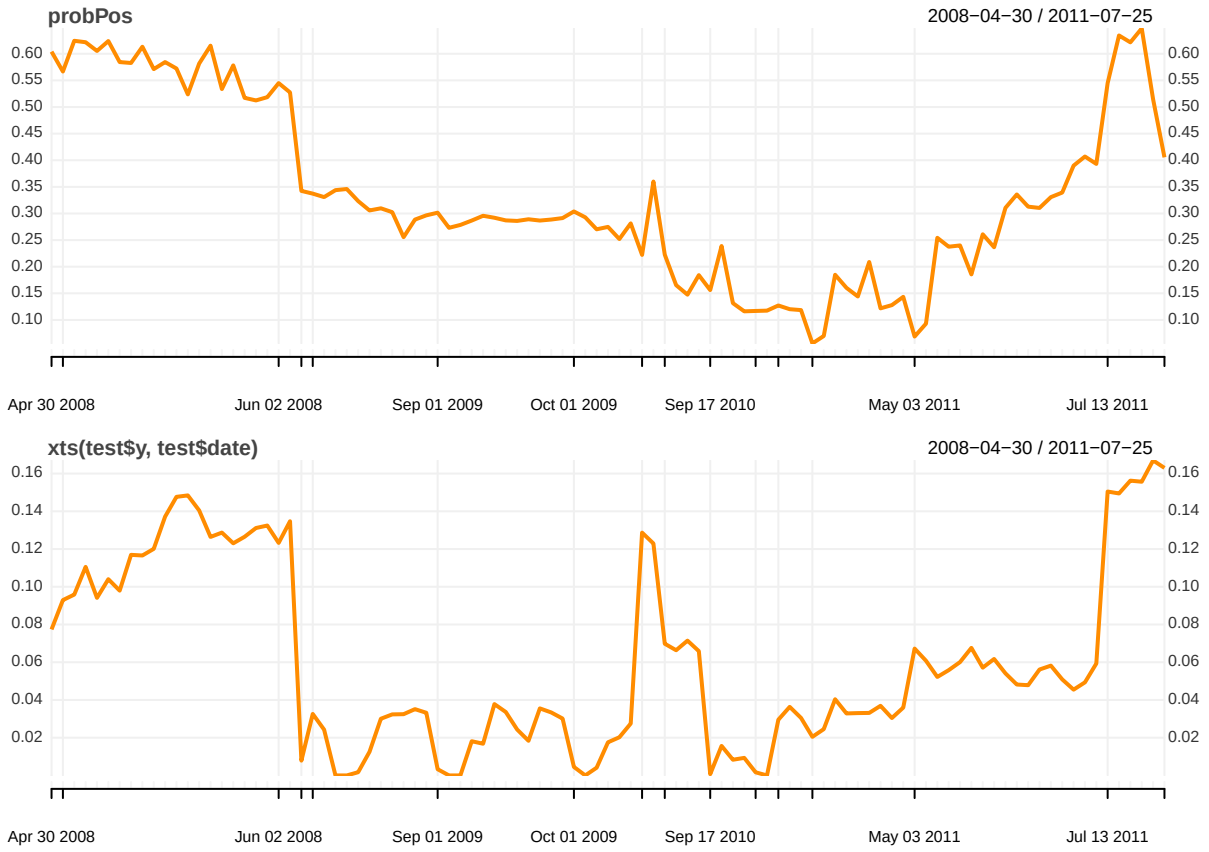
```
# Run the random forest model.
library(randomForest)

testDats <- list()
probs <- list()
ps <- list()
for (i in 1 : length(startDays)) {
  startTest <- startDays[i]
  endTest <- endDays[i]
  expr <- quote(!(date > startTest & date < endTest))
  condition <- expr
  train <- dat[eval(condition)]
  test <- dat[!eval(condition)]
  costs <- table(train$y_binary)
  costs <- 1 / costs

  model <- randomForest(
    y_binary ~ hly_rsi + hly_ret + yc2y10_diff + sox_rsi + lmex_ret_diff + be5_diff,
    data = train,
    ntree = 10000,
    importance = TRUE,
    classwt = costs
  )
  importance(model)
  probPos <- predict(model, newdata = test, type = "prob")[, 2]
  probs[[i]] <- probPos
  testDats[[i]] <- test
  ps[[i]] <- as.numeric(probPos > 0.5)
}
```

One of the meaningful ways to gauge whether it is a robust model for the out-of-sample data is to visualize

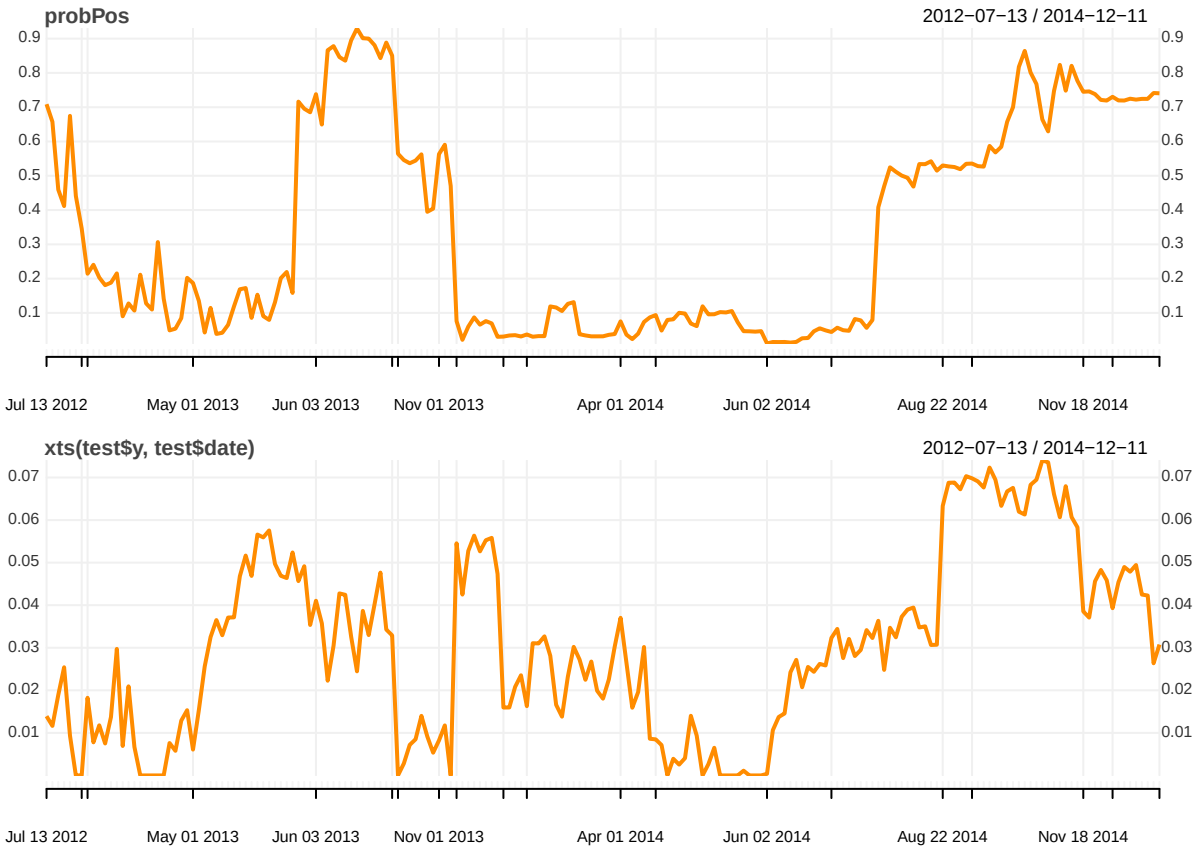
the relationship between the fitted probabilities and the actual target variable value over the testing period.

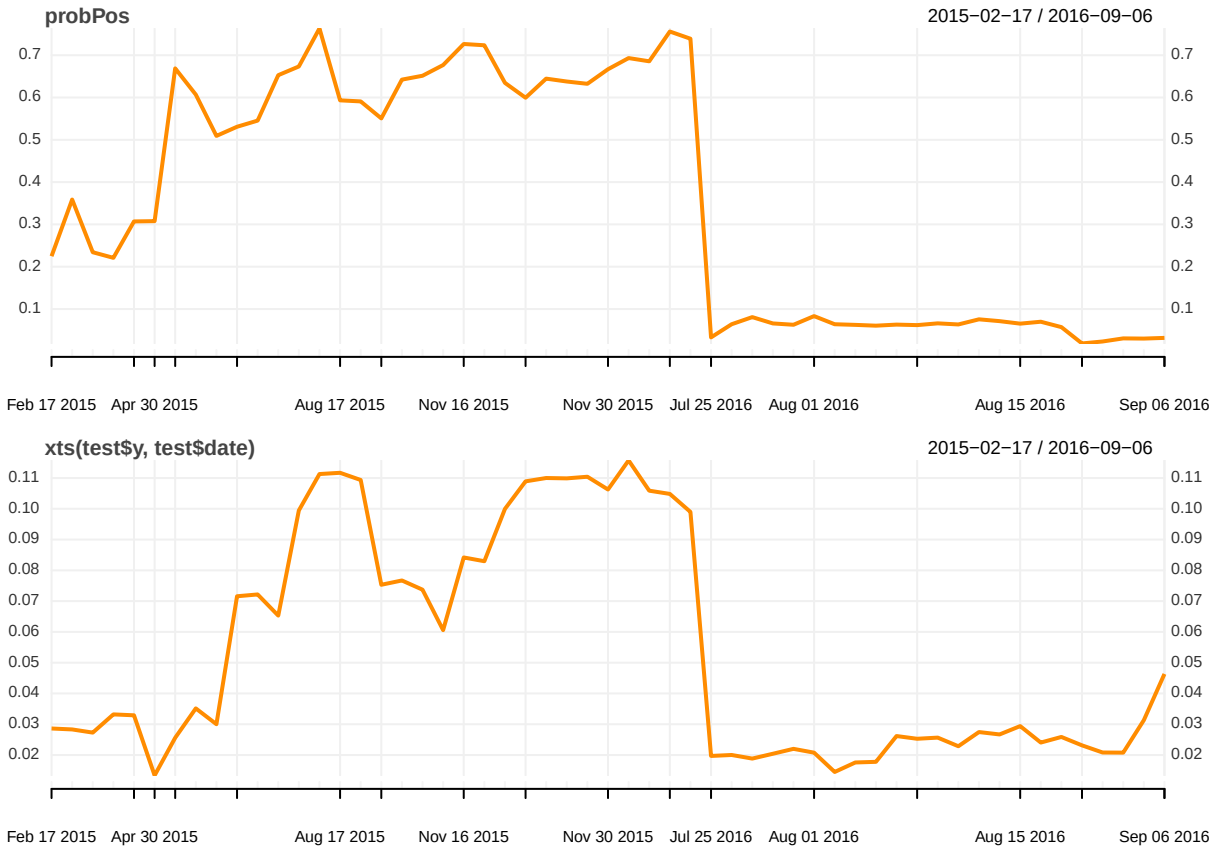
```
# Compare the probabilities with the target values.
for (i in 1 : length(startDays)) {
  test <- testDats[[i]]
  probPos <- xts(probs[[i]], order.by = test$date)
  p <- ps[[i]]

  par(mfrow = c(2, 1))
  print(chart_Series(probPos),
        name = "Fitted Conditional Probabilities")
  print(chart_Series(xts(test$y, test$date)),
        name = "Relative Return from the Current Day over Next Two Months")
}
```




probPos
2008−04−30 / 2011−07−25
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
Apr 30 2008
Jun 02 2008
Sep 01 2009
Oct 01 2009
Sep 17 2010
May 03 2011
Jul 13 2011

xts(test$y, test$date)
2008−04−30 / 2011−07−25
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
Apr 30 2008
Jun 02 2008
Sep 01 2009
Oct 01 2009
Sep 17 2010
May 03 2011
Jul 13 2011


probPos
2012−07−13 / 2014−12−11
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Jul 13 2012
May 01 2013
Jun 03 2013
Nov 01 2013
Apr 01 2014
Jun 02 2014
Aug 22 2014
Nov 18 2014
xts(test$y, test$date)
2012−07−13 / 2014−12−11
0.07
0.06
0.05
0.04
0.03
0.02
0.01
Jul 13 2012
May 01 2013
Jun 03 2013
Nov 01 2013
Apr 01 2014
Jun 02 2014
Aug 22 2014
Nov 18 2014



If a cutoff value of 0.5 is set for the fitted conditional probabilities, a confusion matrix can be generated for each testing period.

```
# Confusion matrix.
for (i in 1 : length(startDays)) {
  test <- testDats[[i]]
  p <- ps[[i]]

  evalRes <-
     confusionMatrix(p, test$y_binary, positive = levels(test$y_binary)[2])
  print(paste0("Period from ", startDays[i], " to ",endDays[i]))
  print(evalRes)
}
```

```
## [1] "Period from 2007-01-01 to 2007-12-31"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22 26
##          1  0 17
##
##                Accuracy : 0.6
##                  95% CI : (0.471, 0.7196)
##     No Information Rate : 0.6615
##     P-Value [Acc > NIR] : 0.88
##
```

```
##                  Kappa : 0.3068
##  Mcnemar's Test P-Value : 9.443e-07
##
##             Sensitivity : 0.3953
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.4583
##              Prevalence : 0.6615
##          Detection Rate : 0.2615
##    Detection Prevalence : 0.2615
##       Balanced Accuracy : 0.6977
##
##        'Positive' Class : 1
##
## [1] "Period from 2008-01-01 to 2011-12-31"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52 20
##          1  0 27
##
##                Accuracy : 0.798
##                  95% CI : (0.7054, 0.872)
##     No Information Rate : 0.5253
##     P-Value [Acc > NIR] : 1.613e-08
##
##                   Kappa : 0.5865
##  Mcnemar's Test P-Value : 2.152e-05
##
##             Sensitivity : 0.5745
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.7222
##              Prevalence : 0.4747
##          Detection Rate : 0.2727
##    Detection Prevalence : 0.2727
##       Balanced Accuracy : 0.7872
##
##        'Positive' Class : 1
##
## [1] "Period from 2012-01-01 to 2014-12-31"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 108  11
##          1  48  24
##
##                Accuracy : 0.6911
##                  95% CI : (0.6203, 0.7558)
##     No Information Rate : 0.8168
##     P-Value [Acc > NIR] : 1
##
```

```
##                   Kappa : 0.2681
##  Mcnemar's Test P-Value : 2.775e-06
##
##             Sensitivity : 0.6857
##             Specificity : 0.6923
##          Pos Pred Value : 0.3333
##          Neg Pred Value : 0.9076
##              Prevalence : 0.1832
##          Detection Rate : 0.1257
##    Detection Prevalence : 0.3770
##       Balanced Accuracy : 0.6890
##
##        'Positive' Class : 1
##
## [1] "Period from 2015-01-01 to 2016-12-31"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  0
##          1  3 23
##
##                Accuracy : 0.9455
##                  95% CI : (0.8488, 0.9886)
##     No Information Rate : 0.5818
##     P-Value [Acc > NIR] : 1.22e-09
##
##                   Kappa : 0.8899
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 1.0000
##             Specificity : 0.9062
##          Pos Pred Value : 0.8846
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4182
##          Detection Rate : 0.4182
##    Detection Prevalence : 0.4727
##       Balanced Accuracy : 0.9531
##
##        'Positive' Class : 1
##
```

Furthermore, we can check which days are selected by the 0.5 cutoff value. The result shows that our method is quite resillient against the false positive error, as majority of the interesting periods have been selected, whereas most of the misclassified days are actually interesting days with future drawdowns slightly smaller than 5% (mostly above 3.5%).

```
# Print the positively classified observations using a cutoff value of 0.5.
d_1s <- list()
for (i in 1 : length(startDays)) {
  test <- testDats[[i]]
  p <- ps[[i]]

  d_1 <- as.data.frame(test[p == 1])[, 1 : 3]
  d_1s[[i]] <- d_1
```

```
  print(d_1)
}
```

```
##          date y_binary          y
## 1  2007-06-27        1 0.06614708
## 2  2007-06-29        1 0.06428975
## 3  2007-07-02        1 0.07419230
## 4  2007-07-03        1 0.07749513
## 5  2007-10-10        1 0.09936191
## 6  2007-10-11        1 0.09469188
## 7  2007-10-12        1 0.09897554
## 8  2007-10-15        1 0.09135991
## 9  2007-10-16        1 0.08534770
## 10 2007-10-23        1 0.07394758
## 11 2007-10-24        1 0.07168114
## 12 2007-10-25        1 0.07077390
## 13 2007-10-26        1 0.08341149
## 14 2007-10-29        1 0.08680191
## 15 2007-10-30        1 0.08086113
## 16 2007-10-31        1 0.09175283
## 17 2007-11-06        1 0.07436179
##          date y_binary          y
## 1  2008-04-30        1 0.07737498
## 2  2008-05-01        1 0.09292293
## 3  2008-05-02        1 0.09584836
## 4  2008-05-06        1 0.11051570
## 5  2008-05-07        1 0.09410658
## 6  2008-05-08        1 0.10400807
## 7  2008-05-09        1 0.09794134
## 8  2008-05-12        1 0.11690819
## 9  2008-05-13        1 0.11656831
## 10 2008-05-14        1 0.12009285
## 11 2008-05-15        1 0.13716923
## 12 2008-05-16        1 0.14764093
## 13 2008-05-19        1 0.14840568
## 14 2008-05-20        1 0.14043441
## 15 2008-05-21        1 0.12641025
## 16 2008-05-22        1 0.12869079
## 17 2008-05-27        1 0.12303028
## 18 2008-05-28        1 0.12649190
## 19 2008-05-29        1 0.13112726
## 20 2008-05-30        1 0.13244262
## 21 2008-06-02        1 0.12323280
## 22 2008-06-05        1 0.13471030
## 23 2011-07-13        1 0.15045685
## 24 2011-07-15        1 0.14943699
## 25 2011-07-19        1 0.15622621
## 26 2011-07-20        1 0.15565981
## 27 2011-07-21        1 0.16694449
##          date y_binary           y
## 1  2012-07-13        0 0.013922670
## 2  2012-07-16        0 0.011635294
## 3  2012-07-23        0 0.009351953
```

```
## 4  2013-05-29        0 0.045663569
## 5  2013-05-30        0 0.049153475
## 6  2013-05-31        0 0.035352049
## 7  2013-06-03        0 0.041044367
## 8  2013-06-04        0 0.035730486
## 9  2013-06-05        0 0.022257443
## 10 2013-06-06        0 0.030488857
## 11 2013-06-07        0 0.042771605
## 12 2013-06-10        0 0.042439479
## 13 2013-06-11        0 0.032617318
## 14 2013-06-12        0 0.024452410
## 15 2013-06-13        0 0.038665086
## 16 2013-06-14        0 0.032974126
## 17 2013-06-17        0 0.040236968
## 18 2013-06-18        0 0.047656813
## 19 2013-06-19        0 0.034280172
## 20 2013-07-30        0 0.032907068
## 21 2013-10-23        0 0.000000000
## 22 2013-10-24        0 0.002808107
## 23 2013-10-25        0 0.007171392
## 24 2013-10-28        0 0.008489822
## 25 2013-10-29        0 0.013995880
## 26 2013-11-01        0 0.008225290
## 27 2013-11-04        0 0.011753859
## 28 2014-07-18        0 0.034702915
## 29 2014-07-21        0 0.032457958
## 30 2014-07-22        0 0.037287059
## 31 2014-07-25        0 0.034761467
## 32 2014-07-28        0 0.035039491
## 33 2014-07-29        0 0.030650524
## 34 2014-07-30        0 0.030709569
## 35 2014-08-22        1 0.063322269
## 36 2014-08-26        1 0.068764312
## 37 2014-08-27        1 0.068810871
## 38 2014-08-28        1 0.067234592
## 39 2014-08-29        1 0.070321508
## 40 2014-09-02        1 0.069815410
## 41 2014-09-03        1 0.069090128
## 42 2014-09-04        1 0.067659500
## 43 2014-09-05        1 0.072331163
## 44 2014-09-08        1 0.069471507
## 45 2014-09-09        1 0.063341112
## 46 2014-09-10        1 0.066743833
## 47 2014-09-11        1 0.067566147
## 48 2014-09-12        1 0.061973065
## 49 2014-09-15        1 0.061306467
## 50 2014-09-16        1 0.068279823
## 51 2014-09-17        1 0.069485454
## 52 2014-09-18        1 0.074014597
## 53 2014-09-19        1 0.073572423
## 54 2014-09-22        1 0.066088683
## 55 2014-09-23        1 0.060662608
## 56 2014-09-24        1 0.067962768
## 57 2014-09-26        1 0.060700507
```

```
## 58 2014-09-29        1 0.058302154
## 59 2014-11-18        0 0.038532021
## 60 2014-11-19        0 0.037086571
## 61 2014-11-25        0 0.045616174
## 62 2014-11-26        0 0.048286642
## 63 2014-11-28        0 0.045860821
## 64 2014-12-01        0 0.039299906
## 65 2014-12-02        0 0.045394498
## 66 2014-12-03        0 0.048974850
## 67 2014-12-04        0 0.047868644
## 68 2014-12-05        0 0.049451423
## 69 2014-12-08        0 0.042503313
## 70 2014-12-09        0 0.042275539
## 71 2014-12-10        0 0.026355533
## 72 2014-12-11        0 0.030751770
##          date y_binary          y
## 1  2015-06-17        0 0.02559464
## 2  2015-06-18        0 0.03514925
## 3  2015-06-19        0 0.03000488
## 4  2015-06-22        1 0.07158301
## 5  2015-06-23        1 0.07217305
## 6  2015-06-24        1 0.06529987
## 7  2015-06-25        1 0.09946202
## 8  2015-06-26        1 0.11129246
## 9  2015-08-17        1 0.11169403
## 10 2015-08-18        1 0.10935563
## 11 2015-11-09        1 0.07531584
## 12 2015-11-10        1 0.07671060
## 13 2015-11-11        1 0.07372048
## 14 2015-11-12        1 0.06057762
## 15 2015-11-16        1 0.08419094
## 16 2015-11-17        1 0.08296268
## 17 2015-11-20        1 0.09996314
## 18 2015-11-23        1 0.10891454
## 19 2015-11-24        1 0.11000220
## 20 2015-11-25        1 0.10988716
## 21 2015-11-27        1 0.11041524
## 22 2015-11-30        1 0.10626751
## 23 2015-12-01        1 0.11571223
## 24 2015-12-02        1 0.10588071
## 25 2015-12-07        1 0.10483036
## 26 2015-12-08        1 0.09898284
```

The overall true positive error rate can be computed.

```
d_1 <- Reduce(rbind, d_1s)
(tpRate <- table(d_1$y_binary)['1'] / nrow(d_1))
```

```
##         1
## 0.6408451
```

What if we care more about a `spx` drawdown of 3%? For reference, the true positive rate is over 90%.

```
(tpRate_0.03 <- sum(d_1$y > 0.03) / nrow(d_1))
```

```
## [1] 0.9014085
```

Similar performance can be expected from `xgboost`, which may perform slightly better on the 1st & 3rd testing period but noticeably worse on the 2nd. In practice, one can consider aggregate results from several models for better overall performance.