Gradient-Boosting-Model.R

Shraddha Somani

```
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 3.3.3
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.3.3
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```
# Get market data for all symbols making up the NASDAQ 100 Index
Stocks <- c("MAT","SNE","AAPL","AMGN","MAR")
getSymbols(Stocks,from = '2012-01-01', to = '2017-04-17'  )
```

```
##     As of 0.4-0, 'getSymbols' uses env=parent.frame() and
##  auto.assign=TRUE by default.
##
##  This  behavior  will be  phased out in 0.5-0  when the call  will
##  default to use auto.assign=FALSE. getOption("getSymbols.env") and
##  getOptions("getSymbols.auto.assign") are now checked for alternate
defaults
##
```

```
##  This message is shown once per session and may be disabled by setting
##  options("getSymbols.warning4.0"=FALSE). See ?getSymbols for more details.

## [1] "MAT"  "SNE"  "AAPL" "AMGN" "MAR"

# Merge them all together
stocks <- data.frame(as.xts(merge(AMGN,SNE,AAPL,MAT,MAR)))
head(stocks[,1:12],2)

##            AMGN.Open AMGN.High AMGN.Low AMGN.Close AMGN.Volume
## 2012-01-03     64.95     65.19    63.45      64.11    10216800
## 2012-01-04     64.10     64.13    63.40      63.76     7096400
##            AMGN.Adjusted SNE.Open SNE.High SNE.Low SNE.Close SNE.Volume
## 2012-01-03      57.46720    18.28    18.50   18.28     18.38    1414800
## 2012-01-04      57.15346    18.24    18.27   18.14     18.22    1146400
##            SNE.Adjusted
## 2012-01-03        18.38
## 2012-01-04        18.22

# Set outcome variable
outcomeSymbol <- 'MAR.Close'

# Determine whether the close price of 'Apple' stock is higher or lower than
the current trading day

# Shift outcome value to be on same line as predictors
library(xts)
stocks <- xts(stocks,order.by=as.Date(rownames(stocks)))
stocks <- as.data.frame(merge(stocks, lm1=lag(stocks[,outcomeSymbol],-1)))
stocks$outcome <- ifelse(stocks[,paste0(outcomeSymbol,'.1')] >
stocks[,outcomeSymbol], 1, 0)

# Analysis:
# - Predict down one trading day using the lag function.

# - This will add the close field of our outcome symbol with a lag of 1
trading day so it's on the same line as the predictors. We will rely on this
value for training and testing purposes.
# - A value of 1 means the close price went up, and a 0, that it went down:
# - As we took the outcomeSymbol as close price of Apple stock we can
interpret the output in terms of high and low where 1 being high and 0 being
low.



# Remove shifted down close field as we don't care by the value
stocks <- stocks[,!names(stocks) %in% c(paste0(outcomeSymbol,'.1'))]
head(stocks)

##            AMGN.Open AMGN.High AMGN.Low AMGN.Close AMGN.Volume
## 2012-01-03     64.95     65.19    63.45      64.11    10216800
```

```
## 2012-01-04     64.10     64.13    63.40      63.76     7096400
## 2012-01-05     63.82     64.92    63.58      64.41     6261500
## 2012-01-06     64.24     64.87    64.00      64.76     4587900
## 2012-01-09     64.62     64.64    63.30      64.20     7540500
## 2012-01-10     64.55     65.70    64.21      65.67     6150500
##            AMGN.Adjusted SNE.Open SNE.High SNE.Low SNE.Close SNE.Volume
## 2012-01-03      57.46720    18.28    18.50   18.28     18.38    1414800
## 2012-01-04      57.15346    18.24    18.27   18.14     18.22    1146400
## 2012-01-05      57.73612    17.83    17.85   17.60     17.70    1464900
## 2012-01-06      58.04985    17.57    17.57   17.37     17.44     594100
## 2012-01-09      57.54787    17.51    17.51   17.35     17.47     529400
## 2012-01-10      58.86556    17.70    17.76   17.61     17.70    1037400
##            SNE.Adjusted AAPL.Open AAPL.High AAPL.Low AAPL.Close
## 2012-01-03        18.38    409.40    412.50   409.00     411.23
## 2012-01-04        18.22    410.00    414.68   409.28     413.44
## 2012-01-05        17.70    414.95    418.55   412.67     418.03
## 2012-01-06        17.44    419.77    422.75   419.22     422.40
## 2012-01-09        17.47    425.50    427.75   421.35     421.73
## 2012-01-10        17.70    425.91    426.00   421.50     423.24
##            AAPL.Volume AAPL.Adjusted MAT.Open MAT.High MAT.Low MAT.Close
## 2012-01-03    75555200      53.27877    28.31    28.99   27.73     27.76
## 2012-01-04    65005500      53.56510    27.86    28.30   27.82     28.18
## 2012-01-05    67817400      54.15978    28.12    28.50   27.91     28.47
## 2012-01-06    79573200      54.72595    28.48    28.50   28.11     28.16
## 2012-01-09    98506100      54.63915    28.34    28.57   28.28     28.53
## 2012-01-10    64549100      54.83478    28.81    28.93   28.69     28.84
##            MAT.Volume MAT.Adjusted MAR.Open MAR.High MAR.Low MAR.Close
## 2012-01-03    3353500     21.88299    29.88    30.28   29.82     30.00
## 2012-01-04    2587800     22.21407    29.93    30.57   29.73     30.47
## 2012-01-05    2948000     22.44268    30.20    31.60   30.03     31.47
## 2012-01-06    3046000     22.19831    31.39    31.99   31.19     31.74
## 2012-01-09    5016300     22.48997    31.78    32.50   31.70     32.17
## 2012-01-10    2260900     22.73435    33.00    33.28   32.43     32.70
##            MAR.Volume MAR.Adjusted outcome
## 2012-01-03    3290500     27.89194       1
## 2012-01-04    5360800     28.32891       1
## 2012-01-05    7517800     29.25864       1
## 2012-01-06    5495200     29.50967       1
## 2012-01-09    5306000     29.90945       1
## 2012-01-10    6334900     30.40221       1
```

```r
# Cast date to true date and order in decreasing order
stocks$date <- as.Date(row.names(stocks))
stocks <- stocks[order(as.Date(stocks$date, "%m/%d/%Y"), decreasing = TRUE),]
# Analysis:

# - Here is the pattern maker function. This will take our raw market data
and scale it so that we can compare any symbol with any other symbol.
```

```
# - It then subtracts the different day ranges requested by the days
parameter using the diff and lag calls and puts them all on the same row
along with the outcome.
# - To make things even more compatible, the roundByScaler parameter can
round results.


# Calculate all day differences and populate them on same row
Difference_in_Days <- function(objDF,days=c(10),
offLimitsSymbols=c('outcome'), roundByScaler=3) {
  # needs to be sorted by date in decreasing order
  ind <- sapply(objDF, is.numeric)
  for (sym in names(objDF)[ind]) {
    if (!sym %in% offLimitsSymbols) {
      print(paste('*********', sym))
      objDF[,sym] <- round(scale(objDF[,sym]),roundByScaler)

      print(paste('theColName', sym))
      for (day in days) {
        objDF[paste0(sym,'_',day)] <- c(diff(objDF[,sym],lag =
day),rep(x=0,day)) * -1
      }
    }
  }
  return (objDF)
}


# Call the function with the following differences
stocks <- Difference_in_Days(stocks, days=c(1,2,3,4,5,10,20),
offLimitsSymbols=c('outcome'), roundByScaler=2)
```

```
## [1] "********* AMGN.Open"
## [1] "theColName AMGN.Open"
## [1] "********* AMGN.High"
## [1] "theColName AMGN.High"
## [1] "********* AMGN.Low"
## [1] "theColName AMGN.Low"
## [1] "********* AMGN.Close"
## [1] "theColName AMGN.Close"
## [1] "********* AMGN.Volume"
## [1] "theColName AMGN.Volume"
## [1] "********* AMGN.Adjusted"
## [1] "theColName AMGN.Adjusted"
## [1] "********* SNE.Open"
## [1] "theColName SNE.Open"
## [1] "********* SNE.High"
## [1] "theColName SNE.High"
## [1] "********* SNE.Low"
## [1] "theColName SNE.Low"
```

```
## [1] "********* SNE.Close"
## [1] "theColName SNE.Close"
## [1] "********* SNE.Volume"
## [1] "theColName SNE.Volume"
## [1] "********* SNE.Adjusted"
## [1] "theColName SNE.Adjusted"
## [1] "********* AAPL.Open"
## [1] "theColName AAPL.Open"
## [1] "********* AAPL.High"
## [1] "theColName AAPL.High"
## [1] "********* AAPL.Low"
## [1] "theColName AAPL.Low"
## [1] "********* AAPL.Close"
## [1] "theColName AAPL.Close"
## [1] "********* AAPL.Volume"
## [1] "theColName AAPL.Volume"
## [1] "********* AAPL.Adjusted"
## [1] "theColName AAPL.Adjusted"
## [1] "********* MAT.Open"
## [1] "theColName MAT.Open"
## [1] "********* MAT.High"
## [1] "theColName MAT.High"
## [1] "********* MAT.Low"
## [1] "theColName MAT.Low"
## [1] "********* MAT.Close"
## [1] "theColName MAT.Close"
## [1] "********* MAT.Volume"
## [1] "theColName MAT.Volume"
## [1] "********* MAT.Adjusted"
## [1] "theColName MAT.Adjusted"
## [1] "********* MAR.Open"
## [1] "theColName MAR.Open"
## [1] "********* MAR.High"
## [1] "theColName MAR.High"
## [1] "********* MAR.Low"
## [1] "theColName MAR.Low"
## [1] "********* MAR.Close"
## [1] "theColName MAR.Close"
## [1] "********* MAR.Volume"
## [1] "theColName MAR.Volume"
## [1] "********* MAR.Adjusted"
## [1] "theColName MAR.Adjusted"
```

```r
# Drop most recent entry as we don't have an outcome
stocks <- stocks[2:nrow(stocks),]
```

```r
# Extract the day of the week, day of the month, day of the year as
predictors using POSIXlt:
stocks$wday <- as.POSIXlt(stocks$date)$wday
stocks$yday <- as.POSIXlt(stocks$date)$mday
stocks$mon<- as.POSIXlt(stocks$date)$mon


# Next we remove the date field as it won't help us as a predictor
# As they are all unique and we shuffle the data set using the sample
function:


# Remove date field and shuffle data frame
stocks <- subset(stocks, select=-c(date))
stocks <- stocks[sample(nrow(stocks)),]

# Fit the model
# Use a simple gbm model to get an AUC score.
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.3.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
predictorNames <- names(stocks)[names(stocks) != 'outcome']
head(predictorNames)
```

```
## [1] "AMGN.Open"     "AMGN.High"     "AMGN.Low"      "AMGN.Close"
## [5] "AMGN.Volume"   "AMGN.Adjusted"
```

```r
set.seed(1234)
split <- sample(nrow(stocks), floor(0.7*nrow(stocks)))
train <-stocks[split,]
test <- stocks[-split,]


# As this is a binary classification, we need to force gbm into using the
classification mode. We do this by changing the outcome variable to a factor:
train$outcome <- ifelse(train$outcome==1,'yes','nope')
head(train$outcome)
```

```
## [1] "yes"  "yes"  "nope" "yes"  "yes"  "yes"
```

```r
# Create caret trainControl object to control the number of cross-validations
performed
objControl <- trainControl(method='cv', number=2, returnResamp='none',
summaryFunction = twoClassSummary, classProbs = TRUE)
```

```r
# Run model
bst <- train(train[,predictorNames],  as.factor(train$outcome),
             method='gbm',
             trControl=objControl,
             metric = "ROC",
             tuneGrid = expand.grid(n.trees = 5, interaction.depth = 3,
shrinkage = 0.1, n.minobsinnode = 1)
)
```

```
## Loading required package: gbm
```

```
## Warning: package 'gbm' was built under R version 3.3.3
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.3
```

```
## Loading required package: plyr
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:lubridate':
##
##     here
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3691             nan     0.1000    0.0002
##      2        1.3557             nan     0.1000    0.0002
##      3        1.3442             nan     0.1000   -0.0054
##      4        1.3306             nan     0.1000   -0.0025
##      5        1.3158             nan     0.1000   -0.0004
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3729             nan     0.1000   -0.0016
##      2        1.3607             nan     0.1000   -0.0027
##      3        1.3505             nan     0.1000   -0.0031
##      4        1.3363             nan     0.1000   -0.0005
##      5        1.3270             nan     0.1000   -0.0040
##
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.3768             nan     0.1000   -0.0014
##      2        1.3714             nan     0.1000   -0.0019
##      3        1.3621             nan     0.1000   -0.0009
##      4        1.3567             nan     0.1000   -0.0025
##      5        1.3519             nan     0.1000   -0.0028
```

```
print(bst)
```

```
## Stochastic Gradient Boosting
##
## 930 samples
## 243 predictors
##   2 classes: 'nope', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold)
## Summary of sample sizes: 465, 465
## Resampling results:
##
##   ROC        Sens       Spec
##   0.4813517  0.3220721  0.6769547
##
## Tuning parameter 'n.trees' was held constant at a value of 5
##  3
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 1
```

```
# Analysis:

# - There are two types of evaluation we can do here, raw or prob.
# - Raw gives you a class prediction, in our case yes and nope,
# - While prob gives you the probability on how sure the model is about it's
choice.
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 3.3.3
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
predictions <- predict(object=bst, train[,predictorNames], type='prob')
head(predictions)
```

```
##        nope       yes
## 1 0.5210962 0.4789038
## 2 0.4455583 0.5544417
## 3 0.4609229 0.5390771
## 4 0.4643479 0.5356521
## 5 0.4575425 0.5424575
## 6 0.4200981 0.5799019

auc <- auc(train$outcome,predictions[[2]])
print(paste('AUC score:', auc))

## [1] "AUC score: 0.655993956919883"

# Analysis:

# - AUC of ROC is a better measure than accuracy
# - AUC as a criteria for comparing learning algorithms
# - AUC of our model is 0.649285290455624 (remember that an AUC ranges
between 0.5 and 1, where 0.5 is random and 1 is perfect).
# - AUC replaces accuracy when comparing classifiers
# - Experimental results show AUC indicates a difference in performance
between decision trees and Naïve Bayes (significantly better)
```