Modelling Procedure (ML Fin Data - Project 1)

Hair Albeiro Parra Barrera

Libraries

Getting the data

0.0.1 SP500 Economic Sectors

The following function fetches and extract the economic sectors from the SP500, taken from Wikipedia.

```
# NOTE: not necessary to run anymore
# fetch the sectors as a dataframe
sp500_sectors <- f_get_sp500_sectors()
head(sp500_sectors)
```

```
##   tickers                sectors
## 1     MMM            Industrials
## 2     AOS            Industrials
## 3     ABT            Health Care
## 4    ABBV            Health Care
## 5     ACN Information Technology
## 6    ATVI Communication Services
```

Retrieving top sectors and stocks

The following function will retrieve the top sectors and stocks from the SP500 by weight.

```
# Retrieve top 10 stocks by weight for each sector in the top 5 sectors from the SP500 (by weight)
sector_list <- f_retrieve_top_sp500(top_n_sectors = 6, top_n_stocks = 15, only_tickers=TRUE)
sector_list
```

```
## $Industrials
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
##
## $`Health Care`
##  [1] "ABBV" "ABT"  "AMGN" "BMY"  "DHR"  "ELV"  "GILD" "ISRG" "JNJ"  "LLY"
## [11] "MDT"  "MRK"  "PFE"  "TMO"  "UNH"
##
## $`Information Technology`
##  [1] "AAPL" "ACN"  "ADBE" "AMD"  "AVGO" "CRM"  "CSCO" "IBM"  "INTC" "INTU"
## [11] "MSFT" "NVDA" "ORCL" "QCOM" "TXN"
##
## $`Communication Services`
##  [1] "ATVI"  "CHTR"  "CMCSA" "DIS"   "EA"    "GOOG"  "GOOGL" "META"  "NFLX"
## [10] "OMC"   "T"     "TMUS"  "TTWO"  "VZ"    "WBD"
##
## $Financials
```

```
##  [1] "AXP"  "BAC"  "BLK"  "C"    "CB"   "GS"   "JPM"  "MA"   "MMC"  "MS"
## [11] "PGR"  "SCHW" "SPGI" "V"    "WFC"
##
## $`Consumer Discretionary`
##  [1] "ABNB" "AMZN" "AZO"  "BKNG" "CMG"  "F"    "GM"   "HD"   "MAR"  "MCD"
## [11] "NKE"  "ORLY" "SBUX" "TJX"  "TSLA"
```

Retrieving stock data

We will know use the function `f_fetch_all_tickers` under `functions/fetch_sp500_sectors.R`

```r
# function to fetch all the information for one ticker into a nice xts dataframe
sp500_stocks <- lapply(sector_list,
                       f_fetch_all_tickers,
                       start_date="2016-01-01",
                       end_date="2022-12-01")
```

The result of this function is a list of lists, with elements as below.

```r
# Show the available sectors
names(sp500_stocks)
```

```
## [1] "Industrials"            "Health Care"            "Information Technology"
## [4] "Communication Services" "Financials"             "Consumer Discretionary"
```

```r
# Show available stocks for Industrials
names(sp500_stocks$Industrials)
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```r
# access the xts of the stocks in industrials
head(sp500_stocks$Industrials[[1]])
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-01-06             -1      -0.04944219             NA            NA
## 2016-01-13              1       0.01131390    -0.04944219            NA
## 2016-01-20              1       0.02848331     0.01131390   -0.04944219
## 2016-01-27              1       0.02053790     0.02848331    0.01131390
## 2016-02-03             -1      -0.01619856     0.02053790    0.02848331
## 2016-02-10              1       0.05417793    -0.01619856    0.02053790
##            adjclose_lag2 adjclose_lag3 atr adx aaron bb chaikin_vol clv emv
## 2016-01-06            NA            NA  NA  NA    NA NA          NA  NA  NA
## 2016-01-13            NA            NA  NA  NA   -50 NA          NA  NA  NA
## 2016-01-20            NA            NA  NA  NA  -100 NA          NA  NA  NA
## 2016-01-27   -0.04944219            NA  NA  NA    50 NA          NA  NA  NA
## 2016-02-03    0.01131390   -0.04944219  NA  NA   100 NA          NA  NA  NA
## 2016-02-10    0.02848331    0.01131390  NA  NA    50 NA          NA  NA  NA
##            macd mfi      sar smi volat month_index
## 2016-01-06   NA  NA 79.55761  NA    NA           1
## 2016-01-13   NA  NA 81.71000  NA    NA           1
## 2016-01-20   NA  NA 81.71000  NA    NA           1
## 2016-01-27   NA  NA 77.34000  NA    NA           1
## 2016-02-03   NA  NA 77.34000  NA    NA           2
## 2016-02-10   NA  NA 77.57600  NA    NA           2
```

BACKTESTING parameters

The following code is used in the `strategy_design.rmd` markdown to simulate the backtesting. You can ignore most of the code here, but some variables are necessary.

```
# Set up backtesting simulation parameters
sample_xts <- sp500_stocks$Industrials$ADP
sectors <- names(sp500_stocks)
N_sector_best_stocks <- 3 # new strategy: 3x2 = 6

# Formula parameters
slide <- 1
N_months <- length(names(split.xts(sample_xts, f= "months")))
N_window <- 24 # number of months in size for each window
N_runs <- floor((N_months - N_window)/slide)

# display parameters
print(paste0("N_months: ", N_months))
```

```
## [1] "N_months: 83"
```

```
print(paste0("N_runs: ", N_runs))
```

```
## [1] "N_runs: 59"
```

```
print(paste0("slide: ", slide))
```

```
## [1] "slide: 1"
```

```
# setup initial portfolio tracking variables
initial_capital <- 500000
num_tickers <- length(sectors)*N_sector_best_stocks*2 # two sub-strategies for picking
initial_tickers <- rep(NA, num_tickers)
weights <- rep(1/num_tickers, num_tickers) # initialize to 1/n
returns <- rep(NA, N_runs)

# repack the portfolio
portfolio <- list(tickers = initial_tickers,
                  weights = weights,
                  capital = initial_capital,
                  returns = returns,
                  data = NA
                  )
portfolio
```

```
## $tickers
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA
##
## $weights
##  [1] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
##  [7] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [13] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [19] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [25] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
## [31] 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778 0.02777778
##
```

```
## $capital
## [1] 5e+05
##
## $returns
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [51] NA NA NA NA NA NA NA NA NA
##
## $data
## [1] NA
```

MODELLING_PROCEDURE

Recall that the **SECTOR_PROCEDURE**(G, τ) function takes the argument G, which is the **sector name**, and **tau**, which is the current run in the backtesting.

This procedure happens in a loop, for every sector G. Here, we fix one sector only, and a specific τ. The code does the following:

1. Retrieves the actual sector stock data (list of key-value pairs, keys are stock tickers, values are xts full data for that stock.)
2. Creates a variable to store the subset of data that goes into the current window.
3. The `f_extract_window()` function extracts the appropriate window of data corresponding to the τ, with the appropriate window size, for all sectors.
4. Extracts the dynamic features (ARIMA and GARCH) for that each stock in the sector.

```r
# parameters
G <- names(sp500_stocks)[1] # sample sector
tau <- 10 # suppose we are in run 5 of the backtest

####### Inside SECTOR_PROCEDURE ########

# retrieve sector data
sector_data <- sp500_stocks[[G]]

# stocks for sector provided
sector_tickers <- names(sector_data)

# to store subset features for window
sector_stocks_window <- rep(NA, length(sector_tickers))
names(sector_stocks_window) <- sector_tickers

# extract static train-val for all stocks
list_xts_sector <- lapply(sector_data,
                          f_extract_window,
                          tau=tau, # current run
                          n_months = N_window# size of window
                          )

# compute dynamic features for all stocks
list_xts_sector <- lapply(list_xts_sector,
                          f_extract_dynamic_features,
                          return_col = "realized_returns",
                           volat_col = "volat"
                          )
```

```
## Loading required package: forecast
```

```
## Loading required package: rugarch

## Loading required package: parallel

##
## Attaching package: 'rugarch'

## The following object is masked from 'package:purrr':
##
##     reduce

## The following object is masked from 'package:stats':
##
##     sigma
```

```r
####### Inside SECTOR_PROCEDURE ########

# keys are stock tickers for that sector
names(list_xts_sector)
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```r
# each stock has the xts subset (for window)
head(list_xts_sector[[1]])
```

```
##            direction_lead realized_returns actual_returns adjclose_lag1
## 2016-10-05             -1     -0.008140095    0.001485849  -0.016220180
## 2016-10-12              1      0.006425769   -0.008140095   0.001485849
## 2016-10-19             -1     -0.002748745    0.006425769  -0.008140095
## 2016-10-26              1      0.031497620   -0.002748745   0.006425769
## 2016-11-02              1      0.010172550    0.031497620  -0.002748745
## 2016-11-09              1      0.025738290    0.010172550   0.031497620
##            adjclose_lag2 adjclose_lag3      atr      adx aaron        bb
## 2016-10-05   0.024948710  -0.037026870 1.900259 15.44565   -50 0.2934560
## 2016-10-12  -0.016220180   0.024948710 1.872384 15.23639  -100 0.2289285
## 2016-10-19   0.001485849  -0.016220180 1.800070 14.75791   -50 0.3060118
## 2016-10-26  -0.008140095   0.001485849 1.722923 14.44363   100 0.2860935
## 2016-11-02   0.006425769  -0.008140095 1.864142 14.04553    50 0.4910556
## 2016-11-09  -0.002748745   0.006425769 1.989560 13.44222   100 0.5094234
##             chaikin_vol         clv           emv      macd      mfi      sar
## 2016-10-05   -0.4622892  0.18091008 -0.0006643160 1.3477744 46.50802 95.02127
## 2016-10-12    0.3990933  0.24064338 -0.0026850063 1.1358585 37.92195 94.68802
## 2016-10-19   -0.4336751  0.09899013 -0.0019094937 0.9402188 36.19915 94.36810
## 2016-10-26   -1.0188680 -0.01496489 -0.0021492280 0.7585276 30.28217 94.06097
## 2016-11-02 -324.8278000  0.05096933 -0.0009225739 0.6437468 48.88575 93.76613
## 2016-11-09    1.1391500  0.19338517 -0.0009562142 0.5919089 59.37208 93.48309
##                   smi      volat month_index arima_100_001 arima_010_001
## 2016-10-05  -5.331162 0.10247324          10   0.003087307   0.031497620
## 2016-10-12 -11.930732 0.10506831          10   0.005293386   0.010172550
## 2016-10-19 -17.430099 0.10335977          10   0.003683110   0.025738290
## 2016-10-26 -19.828752 0.09985285          10   0.002663115   0.035598080
## 2016-11-02 -18.073978 0.13389984          11   0.007022295  -0.006539954
## 2016-11-09 -13.909935 0.16512456          11   0.004073051   0.021968920
##            arima_110_001 arima_020_001 arima_120_001 arima_100_011
## 2016-10-05   0.012973672    0.06574398  3.470568e-02   0.003087307
## 2016-10-12   0.021707336   -0.01115252  2.857130e-02   0.005293386
```

```
## 2016-10-19   0.017318741    0.04130403  1.493358e-02   0.003683110
## 2016-10-26   0.030264894    0.04545787  4.953662e-02   0.002663115
## 2016-11-02   0.016252618   -0.04867799 -1.150867e-02   0.007022295
## 2016-11-09   0.006548396    0.05047779 -2.234499e-05   0.004073051
##            arima_010_011 arima_110_011 arima_020_011 arima_120_011 vol_forecast
## 2016-10-05   0.031497620   0.012973672    0.06574398  3.470568e-02    0.1338998
## 2016-10-12   0.010172550   0.021707336   -0.01115252  2.857130e-02    0.1651246
## 2016-10-19   0.025738290   0.017318741    0.04130403  1.493358e-02    0.1746223
## 2016-10-26   0.035598080   0.030264894    0.04545787  4.953662e-02    0.1752898
## 2016-11-02  -0.006539954   0.016252618   -0.04867799 -1.150867e-02    0.1772747
## 2016-11-09   0.021968920   0.006548396    0.05047779 -2.234499e-05    0.1757262
```

The result is the `list_train_val_sector` object, which is a list of lists. - The first level are the stock tickers - The second level are train and val xts for each stock.

```r
# Check num of rows (weeks) for window
nrow(list_xts_sector[[1]])
```

```
## [1] 103
```

Feature Selection

Notes: - This will use **forward selection** to extract the features from a sample stock for the current sector. - The `target_var` argument specifies the target variable, in this case is called "realized_returns". - `f_select_features()` is found under `functions/feature_engineering.R`

```r
# Extract a sample stock in the list_xts_sector
sample_sector_stock <- list_xts_sector[[1]]

# Define the formula for regression
fmla <- realized_returns ~ . -realized_returns -month_index

# try obtaining best features for a sample train set for a stock in the sample sector
best_feat_list <- f_select_features(
                    fmla = fmla, # formula for regression
                    data = sample_sector_stock, # for one stock of current sector
                    target_var = "realized_returns", # y
                    volat_col = "volat", # we always want to keep the volatility col
                    garch_col = "vol_forecast", # GARCH column
                    nvmax = 15, # examine all possible subsets
                    method="exhaustive") #  we always want to use forward selection
```

```
## Loading required package: leaps
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in =
## force.in, : 5 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
## Warning in f_select_features(fmla = fmla, data = sample_sector_stock,
## target_var = "realized_returns", : garch_col must be present in columns
```

```r
best_feat_list
```

```
## $featnames
##  [1] "direction_lead" "actual_returns" "adjclose_lag1"  "adjclose_lag2"
```

```
##  [5] "adjclose_lag3"  "clv"            "macd"           "mfi"
##  [9] "sar"            "smi"            "arima_110_001"  "arima_020_001"
## [13] "arima_120_001"  "vol_forecast"   "volat"
##
## $fmla
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + clv + macd + mfi + sar +
##     smi + arima_110_001 + arima_020_001 + arima_120_001 + vol_forecast +
##     volat
## <environment: 0x00000141f15167f8>
```

The result of this object is a list `best_feat_list` in this case, containing two objects: - **`featnames`**: a list of features selected. - **`fmla`**: An R formula (for regression, etc)

NOTE: - This is just for illustration and to visualize the data. The actual feature selection is performed in a loop for every stock as illustrated in the next section. - There will always be linear dependencies because of the ARIMA features. This is normal.

Regularized MLR (Elasticnet)

After feature selection, we want to fit the following model:

$$\mathcal{L}(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\left[\alpha||\beta||_1 + (1-\alpha)||\beta||_2^2\right]$$

First, we wil do the following: 1. Specify the general formula 2. Create the grid of parameters to use in the Elasticnet models 3. Create a tracking variable to save the forecasted returns, MSEs and Sharpe Ratios computed

```r
# load required libraries
library("caret")
library("Metrics")

# Define the formula for regression
fmla <- realized_returns ~ . -realized_returns -month_index

# Create a grid for elastic net regression hyperparameters
grid_enet <- expand.grid(alpha = seq(from = 0, to = 1, by = 0.1),  # Elastic net mixing parameter
                         lambda = seq(from = 0, to = 0.05, by = 0.005))  # Regularization strength

# Initialize variable to save forecasted returns, MSEs and Sharpe Ratios
sector_tracker <- as.list(rep(NA, length(sector_tickers)))
names(sector_tracker) <- sector_tickers

# transform into a list of lists
sector_tracker <- lapply(sector_tracker, function(x) list(
  forecasted_ret = NA,
  sharpe = NA,
  msr = NA, # modified sharpe ratio
  rmse = NA,
  data = NA
))

# display values
fmla # all initial variables
```

```
## realized_returns ~ . - realized_returns - month_index
```

```
names(sector_tracker) # list of lists
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```
names(sector_tracker[[1]]) # to store the values as the loop happens
```

```
## [1] "forecasted_ret" "sharpe"         "msr"            "rmse"
## [5] "data"
```

Fitting all the models

Next, we loop through every stock doing the following: 1. Extracting the train and validation sets, and filter NAs 2. Perform feature selection for every stock 3. Fit an Elasticnet model for that stock, and obtain predictions for the returns 4. Compute the RMSE 5. Compute the Sharpe Ratio and Modified Sharpe 6. Save everything

```
# Loop for every stock ticker in sector G
for(ticker in sector_tickers){
  print(paste0("ticker: ", ticker))

  ### Step 0: Data Preparation

  ############################################################################
  ### NOTE: Need to refactor

  # fetch data for that ticker
  full_train <- list_xts_sector[[ticker]]

  # Re-extract train and val with full features
  full_train <- f_extract_train_val_no_window(full_train,
                                              val_lag = 1) # number of months in val

  # Reassign to train and val
  ticker_data_train <- full_train$train
  ticker_data_val <- full_train$val

  # remove nas
  ticker_data_train <- na.omit(ticker_data_train) # data cannot contain nas
  ticker_data_val <- na.omit(ticker_data_val) # data cannot contain nas

  ############################################################################

  ### Step 1: Feature Selection

  # Perform feature selection for that stock
  best_feat_list <- f_select_features(
                    fmla = fmla, # formula for regression
                    data = ticker_data_train, # train data for one stock of current sector
                    target_var = "realized_returns", # y
                    volat_col = "volat", # always keep the actual volatility
                    garch_col = "vol_forecast",
                    nvmax = 20, # total number of max subsets
                    method="exhaustive")

  print(best_feat_list$fmla)

  ### Step 2: Elasticnet
```

```r
  # Set up time-slice cross-validation parameters
  ctr_train <- trainControl(method = "timeslice", # cross validation
                            initialWindow = 52,  # Consecutive number of weeks
                            horizon = 4,         # Horizon is one month prediction (4 weeks)
                            skip = 1,            # No skip, our data will overlap in practice
                            fixedWindow = TRUE,   # Use a fixed window
                            allowParallel = TRUE) # Enable parallel processing


  # Train the elastic net regression model using time-slice cross-validation
  model_enet_best <- train(form = best_feat_list$fmla,           # Formula from feature selection
                           data = ticker_data_train,             # Training data
                           method = "glmnet",                    # Model method = Elasticnet
                           tuneGrid = grid_enet,                 # Hyperparameter grid
                           trControl = ctr_train,                # Cross-validation control
                           preProc = c("center", "scale"),       # Preprocessing steps
                           metric = "Rsquared",                  # Metric for selecting the best model
                           threshold = 0.2)

  # Extract the best alpha and beta fitted
  best_alpha <- model_enet_best$bestTune$alpha
  best_lambda <- model_enet_best$bestTune$lambda

  # Use the best-fitted elastic net regression model to make predictions on the val_data
  pred_enet_best <- predict(model_enet_best, ticker_data_val) # predict on val
  pred_enet_best <- mean(pred_enet_best) # take the average

  # Compute the RMSE on the validation set
  enet_rmse <- sqrt(mse(actual = ticker_data_val[, "realized_returns"], predicted = pred_enet_best))

  ### Step 3: Sharpe Ratio

  # Calculate the Sharpe Ratio and MSR
  stock_sharpe <- SharpeRatio(ticker_data_train[, "realized_returns"], Rf=0.02, FUN="StdDev")
  stock_msr <- SharpeRatio(ticker_data_train[, "realized_returns"], Rf=0.02, FUN="ES")

  ### Step 4: Track the measures

  sector_tracker[[ticker]]$forecasted_ret = pred_enet_best
  sector_tracker[[ticker]]$rmse = enet_rmse
  sector_tracker[[ticker]]$sharpe = stock_sharpe
  sector_tracker[[ticker]]$msr = stock_msr
  # sector_tracker[[ticker]]$data = rbind.xts(ticker_data_train, ticker_data_val) # This should be included at


  # show values
  print("*******************************************")
  print(paste("predicted return: ", pred_enet_best))
  print(paste("rmse: ", enet_rmse))
  print(paste("sharpe: ", stock_sharpe))
  print(paste("msr: ", stock_msr))
  print("*******************************************")

  print("###########################################")
}
```

```
## [1] "ticker: ADP"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
```

```
##     adjclose_lag2 + adjclose_lag3 + clv + macd + mfi + sar +
##     smi + arima_110_001 + arima_100_011 + arima_120_011 + vol_forecast +
##     volat
## <environment: 0x00000141ef950968>
## [1] "*****************************************"
## [1] "predicted return:  0.00555963109676768"
## [1] "rmse:  0.00730287341585979"
## [1] "sharpe:  -0.540595795366778"
## [1] "msr:  -0.203838655634037"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: BA"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + adx + bb + chaikin_vol +
##     clv + emv + macd + mfi + sar + smi + arima_100_011 + arima_020_011 +
##     arima_120_011 + volat + vol_forecast
## <environment: 0x00000141f0fec020>
## [1] "*****************************************"
## [1] "predicted return:  0.0102579557473477"
## [1] "rmse:  0.0336373062894727"
## [1] "sharpe:  -0.326256711825058"
## [1] "msr:  -0.200331723857771"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: CAT"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adjclose_lag2 + atr + bb +
##     chaikin_vol + macd + mfi + smi + arima_110_011 + vol_forecast +
##     volat
## <environment: 0x00000141f0f06a50>
## [1] "*****************************************"
## [1] "predicted return:  0.00858855224301536"
## [1] "rmse:  0.0285403479309924"
## [1] "sharpe:  -0.446026591848937"
## [1] "msr:  -0.262220863439266"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: CSX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adx + bb + emv + macd + mfi +
##     volat + arima_010_001 + arima_020_001 + arima_120_011 + vol_forecast
## <environment: 0x00000141ef762868>
## [1] "*****************************************"
## [1] "predicted return:  0.00923644795656566"
## [1] "rmse:  0.00993856135394996"
## [1] "sharpe:  -0.259991032044992"
## [1] "msr:  -0.119501922420566"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: DE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + aaron + clv + smi +
##     arima_110_011 + vol_forecast + volat
## <environment: 0x00000141e768a320>
## [1] "*****************************************"
## [1] "predicted return:  0.0057141577989899"
## [1] "rmse:  0.0235264762494487"
## [1] "sharpe:  -0.46322438600633"
## [1] "msr:  -0.247710287652867"
## [1] "*****************************************"
```

```
## [1] "###########################################"
## [1] "ticker: ETN"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adx + bb + clv + emv + mfi +
##     sar + smi + volat + arima_110_001 + arima_120_001 + arima_100_011 +
##     vol_forecast
## <environment: 0x00000141f6a06158>
## [1] "*****************************************"
## [1] "predicted return:  0.00322158336889028"
## [1] "rmse:  0.0147719595042477"
## [1] "sharpe:  -0.691845453248933"
## [1] "msr:  -0.390496497074725"
## [1] "*****************************************"
## [1] "###########################################"
## [1] "ticker: FDX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + adx + aaron + bb +
##     clv + mfi + sar + arima_120_011 + vol_forecast + volat
## <environment: 0x00000141f51f4158>
## [1] "*****************************************"
## [1] "predicted return:  0.00299406002617908"
## [1] "rmse:  0.0278432434260068"
## [1] "sharpe:  -0.642659195395819"
## [1] "msr:  -0.273001916126461"
## [1] "*****************************************"
## [1] "###########################################"
## [1] "ticker: GE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + mfi + smi +
##     volat + arima_100_011 + vol_forecast
## <environment: 0x00000141f2fadb78>
## [1] "*****************************************"
## [1] "predicted return:  -0.00526248091890984"
## [1] "rmse:  0.0776111625581303"
## [1] "sharpe:  -0.895360296735942"
## [1] "msr:  -0.354019449139713"
## [1] "*****************************************"
## [1] "###########################################"
## [1] "ticker: HON"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
##     adjclose_lag3 + adx + aaron + bb + clv + emv + macd + smi +
##     volat + arima_100_001 + arima_110_001 + arima_120_011 + vol_forecast
## <environment: 0x00000141f10d0b60>
## [1] "*****************************************"
## [1] "predicted return:  0.00383980360868687"
## [1] "rmse:  0.00674137587142724"
## [1] "sharpe:  -0.841046141581542"
## [1] "msr:  -0.305106131483384"
## [1] "*****************************************"
## [1] "###########################################"
## [1] "ticker: ITW"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag3 +
##     atr + adx + bb + emv + macd + mfi + sar + smi + volat + arima_100_001 +
##     arima_110_011 + arima_120_011 + vol_forecast
## <environment: 0x00000141f0cc6c10>
## [1] "*****************************************"
## [1] "predicted return:  0.00207220713030303"
## [1] "rmse:  0.0223989737648339"
```

```
## [1] "sharpe:  -0.742468522316493"
## [1] "msr:  -0.294107204882337"
## [1] "****************************************"
## [1] "##########################################"
## [1] "ticker: LMT"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + atr + adx + chaikin_vol +
##     macd + mfi + arima_120_001 + arima_110_011 + vol_forecast +
##     volat
## <environment: 0x00000141f82d0fb8>
## [1] "****************************************"
## [1] "predicted return:  0.00360129875666667"
## [1] "rmse:  0.0206388910126488"
## [1] "sharpe:  -0.690320635139213"
## [1] "msr:  -0.315759475212704"
## [1] "****************************************"
## [1] "##########################################"
## [1] "ticker: NOC"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + atr + adx + aaron + clv + smi + vol_forecast +
##     volat
## <environment: 0x00000141f1ec2538>
## [1] "****************************************"
## [1] "predicted return:  0.00366770812626263"
## [1] "rmse:  0.0159475666396349"
## [1] "sharpe:  -0.614257863675188"
## [1] "msr:  -0.225579250414986"
## [1] "****************************************"
## [1] "##########################################"
## [1] "ticker: RTX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + atr + chaikin_vol + clv + mfi + smi + volat +
##     arima_120_001 + arima_110_011 + vol_forecast
## <environment: 0x00000141f4c90278>
## [1] "****************************************"
## [1] "predicted return:  0.00326232324083587"
## [1] "rmse:  0.0234233361011908"
## [1] "sharpe:  -0.813518114828589"
## [1] "msr:  -0.291496260798377"
## [1] "****************************************"
## [1] "##########################################"
## [1] "ticker: UNP"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + atr + adx + clv + emv + macd +
##     smi + volat + arima_110_011 + arima_120_011 + vol_forecast
## <environment: 0x00000141f4eecad0>
## [1] "****************************************"
## [1] "predicted return:  0.00666851908425754"
## [1] "rmse:  0.0164476592363265"
## [1] "sharpe:  -0.561155407289055"
## [1] "msr:  -0.265298271723924"
## [1] "****************************************"
## [1] "##########################################"
## [1] "ticker: UPS"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + actual_returns + adjclose_lag2 +
```

```
##     atr + adx + aaron + bb + chaikin_vol + clv + macd + mfi +
##     smi + arima_100_001 + volat + vol_forecast
## <environment: 0x00000141ec274e98>
## [1] "*****************************************"
## [1] "predicted return:  0.00195920636262626"
## [1] "rmse:  0.0243205789890724"
## [1] "sharpe:  -0.648506184960835"
## [1] "msr:  -0.1786587673575"
## [1] "*****************************************"
## [1] "############################################"
```

Now that all the models have been trained and the metrics recorded, we now simply choose the top 3 stocks based on the return, and the top 3 based on the best sharpe or modified sharpe ratio.

Let's first show some values for the `sector_tracker` object:

```r
names(sector_tracker)
```

```
##  [1] "ADP" "BA"  "CAT" "CSX" "DE"  "ETN" "FDX" "GE"  "HON" "ITW" "LMT" "NOC"
## [13] "RTX" "UNP" "UPS"
```

```r
names(sector_tracker[[1]])
```

```
## [1] "forecasted_ret" "sharpe"         "msr"            "rmse"
## [5] "data"
```

```r
sector_tracker
```

```
## $ADP
## $ADP$forecasted_ret
## [1] 0.005559631
##
## $ADP$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.5405958
##
## $ADP$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2038387
##
## $ADP$rmse
## [1] 0.007302873
##
## $ADP$data
## [1] NA
##
##
## $BA
## $BA$forecasted_ret
## [1] 0.01025796
##
## $BA$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.3262567
##
## $BA$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2003317
```

```
##
## $BA$rmse
## [1] 0.03363731
##
## $BA$data
## [1] NA
##
##
## $CAT
## $CAT$forecasted_ret
## [1] 0.008588552
##
## $CAT$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.4460266
##
## $CAT$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2622209
##
## $CAT$rmse
## [1] 0.02854035
##
## $CAT$data
## [1] NA
##
##
## $CSX
## $CSX$forecasted_ret
## [1] 0.009236448
##
## $CSX$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):        -0.259991
##
## $CSX$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.1195019
##
## $CSX$rmse
## [1] 0.009938561
##
## $CSX$data
## [1] NA
##
##
## $DE
## $DE$forecasted_ret
## [1] 0.005714158
##
## $DE$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.4632244
##
## $DE$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2477103
##
## $DE$rmse
## [1] 0.02352648
```

```
##
## $DE$data
## [1] NA
##
##
## $ETN
## $ETN$forecasted_ret
## [1] 0.003221583
##
## $ETN$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.6918455
##
## $ETN$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.3904965
##
## $ETN$rmse
## [1] 0.01477196
##
## $ETN$data
## [1] NA
##
##
## $FDX
## $FDX$forecasted_ret
## [1] 0.00299406
##
## $FDX$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.6426592
##
## $FDX$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2730019
##
## $FDX$rmse
## [1] 0.02784324
##
## $FDX$data
## [1] NA
##
##
## $GE
## $GE$forecasted_ret
## [1] -0.005262481
##
## $GE$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.8953603
##
## $GE$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.3540194
##
## $GE$rmse
## [1] 0.07761116
##
## $GE$data
## [1] NA
```

```
##
##
## $HON
## $HON$forecasted_ret
## [1] 0.003839804
##
## $HON$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.8410461
##
## $HON$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.3051061
##
## $HON$rmse
## [1] 0.006741376
##
## $HON$data
## [1] NA
##
##
## $ITW
## $ITW$forecasted_ret
## [1] 0.002072207
##
## $ITW$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.7424685
##
## $ITW$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2941072
##
## $ITW$rmse
## [1] 0.02239897
##
## $ITW$data
## [1] NA
##
##
## $LMT
## $LMT$forecasted_ret
## [1] 0.003601299
##
## $LMT$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.6903206
##
## $LMT$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.3157595
##
## $LMT$rmse
## [1] 0.02063889
##
## $LMT$data
## [1] NA
##
##
## $NOC
```

```
## $NOC$forecasted_ret
## [1] 0.003667708
##
## $NOC$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.6142579
##
## $NOC$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2255793
##
## $NOC$rmse
## [1] 0.01594757
##
## $NOC$data
## [1] NA
##
##
## $RTX
## $RTX$forecasted_ret
## [1] 0.003262323
##
## $RTX$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.8135181
##
## $RTX$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2914963
##
## $RTX$rmse
## [1] 0.02342334
##
## $RTX$data
## [1] NA
##
##
## $UNP
## $UNP$forecasted_ret
## [1] 0.006668519
##
## $UNP$sharpe
##                                 realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.5611554
##
## $UNP$msr
##                             realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.2652983
##
## $UNP$rmse
## [1] 0.01644766
##
## $UNP$data
## [1] NA
##
##
## $UPS
## $UPS$forecasted_ret
## [1] 0.001959206
##
```

```
## $UPS$sharpe
##                               realized_returns
## StdDev Sharpe (Rf=2%, p=95%):       -0.6485062
##
## $UPS$msr
##                           realized_returns
## ES Sharpe (Rf=2%, p=95%):       -0.1786588
##
## $UPS$rmse
## [1] 0.02432058
##
## $UPS$data
## [1] NA
```

```
## TODO: Complete the function, keep the name and parameters
f_select_top_stocks <- function(sector_tracker, n=3, sharpe_criterion = "MSR"){
  ## selects the top n + n stocks (n based on forecasted return, n based on sharpe)
  ##
  ## Params:
  ##    - sector_tracker (list of lists): generated by the Loop for every stock ticker in sector G
  ##    - n (int): number of top stocks to choos efor each method. Top n for the predicted returns,
  ##              and top n for the sharpe-based.


  # should be a list of n + n stocks
  # keys are the chosen tickers, values are the xts data for that stock
  # i.e. you just need to select the ticker and xts data from the sector_tracker object.
  best_stocks <- NA
}
```