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.

```r
# 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.

```r
# 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
tail(sp500_stocks$Industrials[[1]])
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2022-10-26       230.1928              1      0.008146007      0.009733913
## 2022-11-02       232.4444              1      0.009781442      0.012306040
## 2022-11-09       235.3226              1      0.012382070      0.053616090
## 2022-11-16       248.2840              1      0.055079470      0.034718700
## 2022-11-23       257.0555              1      0.035328430      0.005923517
## 2022-11-30       258.5827             NA      0.005941096               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3       atr
## 2022-10-26   0.008113008   0.039930970  -0.064535730   0.030150980  9.676399
## 2022-11-02   0.009733913   0.008113008   0.039930970  -0.064535730  9.885942
## 2022-11-09   0.012306040   0.009733913   0.008113008   0.039930970  9.762661
## 2022-11-16   0.053616090   0.012306040   0.009733913   0.008113008 10.232471
## 2022-11-23   0.034718700   0.053616090   0.012306040   0.009733913 10.243009
## 2022-11-30   0.005923517   0.034718700   0.053616090   0.012306040 10.247795
##                 adx aaron        bb chaikin_vol        clv         emv     macd
## 2022-10-26 13.39493   100 0.6110784 -1.49750300 -0.1320576 -0.01707202 2.049576
## 2022-11-02 13.58997   100 0.6303335  2.90314600 -0.2863719  0.02711271 1.939312
## 2022-11-09 13.77107    50 0.6307783 -0.09676625 -0.3920529  0.04765004 1.866926
## 2022-11-16 14.68326   100 0.8325740 -0.38397100 -0.4461119  0.09074850 1.906715
## 2022-11-23 15.95273   100 0.9310325 -0.20180520 -0.3205142  0.11758529 2.068291
## 2022-11-30 16.53998   100 0.8907336  0.48394890 -0.1089895  0.12144667 2.300754
##                 mfi      sar      smi     volat month_index
## 2022-10-26 51.52422 260.0428 8.131402 0.2269538          82
```

```
## 2022-11-02 49.23300 258.6055  5.546375 0.2606250          83
## 2022-11-09 49.20839 257.2257  3.943960 0.2653165          83
## 2022-11-16 48.83463 256.7200  6.291102 0.2641173          83
## 2022-11-23 49.31528 224.1100 11.099826 0.2624611          83
## 2022-11-30 42.97382 224.1100 16.713518 0.2759187          83
```

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.

```r
# 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"
```

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

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

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

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

```r
# 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 <- 20 # 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, # choose based on tau = based on the run
                          tau=tau, # current run
                          n_months = N_window# size of window
                          )

# compute dynamic features for all stocks
```

```r
list_xts_sector <- lapply(list_xts_sector,
                          f_extract_dynamic_features,
                          arima_col = "adjusted_close",
                          volat_col = "volat"
                          )

####### 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)
tail(list_xts_sector[[1]])
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2019-06-26       149.0287              1     -0.032273380      0.007769978
## 2019-07-03       150.1912              1      0.007800243      0.002738418
## 2019-07-10       150.6030              1      0.002742171      0.005152600
## 2019-07-17       151.3810              1      0.005165897      0.010944030
## 2019-07-24       153.0469             -1      0.011004130     -0.004135125
## 2019-07-31       152.4153             -1     -0.004126588     -0.012387130
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3      atr
## 2019-06-26  -0.032805650   0.022966080   0.015677250   0.017239900 3.984716
## 2019-07-03   0.007769978  -0.032805650   0.022966080   0.015677250 3.889379
## 2019-07-10   0.002738418   0.007769978  -0.032805650   0.022966080 3.757280
## 2019-07-17   0.005152600   0.002738418   0.007769978  -0.032805650 3.618188
## 2019-07-24   0.010944030   0.005152600   0.002738418   0.007769978 3.508318
## 2019-07-31  -0.004135125   0.010944030   0.005152600   0.002738418 3.761295
##                 adx aaron        bb chaikin_vol        clv          emv
## 2019-06-26 21.64867   -50 0.7537743  0.78272240 0.01478129 -0.007624395
## 2019-07-03 20.99122  -100 0.7016395 -0.04666522 0.18156252  0.004073041
## 2019-07-10 20.46727   -50 0.7207805 -1.22019400 0.30184809  0.003596908
## 2019-07-17 20.22681   100 0.7737868  0.31959730 0.27609861  0.005760420
## 2019-07-24 20.22642   100 0.8475027  0.59367060 0.34018571 -0.001639433
## 2019-07-31 20.92201   100 0.8847140 -2.27273300 0.17749593  0.008887518
##                macd      mfi      sar      smi     volat month_index
## 2019-06-26 3.671368 58.32538 162.8200 57.39711 0.1829088          42
## 2019-07-03 3.633028 50.08347 162.8200 56.18796 0.1887240          43
## 2019-07-10 3.576794 50.97254 164.0900 55.17636 0.1434100          43
## 2019-07-17 3.512299 57.43927 164.5400 54.69120 0.1392016          43
## 2019-07-24 3.455591 59.22054 165.3900 55.35288 0.1331613          43
## 2019-07-31 3.391786 63.20159 166.2252 54.40503 0.1690267          43
##            sarima_100_001 sarima_010_001 sarima_110_001 sarima_020_001
## 2019-06-26       152.7432       153.0469       152.9894       154.7127
## 2019-07-03       152.1184       152.4153       152.4371       151.7837
## 2019-07-10       151.8246       152.4153       152.4363       151.1522
## 2019-07-17       151.5340       152.4153       152.4364       150.5206
## 2019-07-24       151.2465       152.4153       152.4364       149.8891
## 2019-07-31       150.9621       152.4153       152.4364       149.2575
##            sarima_120_001 sarima_100_011 sarima_010_011 sarima_110_011
## 2019-06-26       154.2644       152.7432       153.0469       152.9894
## 2019-07-03       152.9438       152.1184       152.4153       152.4371
## 2019-07-10       152.8866       151.8246       152.4153       152.4363
## 2019-07-17       153.1251       151.5340       152.4153       152.4364
## 2019-07-24       153.2142       151.2465       152.4153       152.4364
```

```
## 2019-07-31       153.3788       150.9621       152.4153       152.4364
##            sarima_020_011 sarima_120_011 vol_forecast
## 2019-06-26       154.7127       154.2644    0.1331613
## 2019-07-03       151.7837       152.9438    0.1690267
## 2019-07-10       151.1522       152.8866    0.1719658
## 2019-07-17       150.5206       153.1251    0.1745066
## 2019-07-24       149.8891       153.2142    0.1767083
## 2019-07-31       149.2575       153.3788    0.1786198
```

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.

```
# 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`

```
# 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", # future-lagged log-returns
                    volat_col = "volat", # we always want to keep the volatility col
                    garch_col = "vol_forecast", # GARCH column
                    nvmax = 20, # 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:
```

```
print("")
```

```
## [1] ""
```

```
best_feat_list
```

```
## $featnames
##  [1] "adjusted_close"   "direction_lead"   "discrete_returns" "adjclose_lag2"
##  [5] "mfi"              "smi"              "sarima_100_001"   "sarima_020_001"
##  [9] "sarima_120_001"   "sarima_110_011"   "sarima_120_011"   "volat"
```

```
## [13] "vol_forecast"
##
## $fmla
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag2 + mfi + smi + sarima_100_001 + sarima_020_001 +
##     sarima_120_001 + sarima_110_011 + sarima_120_011 + volat +
##     vol_forecast
## <environment: 0x000001f36a2f2970>
```

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 = 0.01))  # 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
```

```r
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

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

  # 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$train # the rest
  ticker_data_val <- full_train$val # the last month only

  # 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", # forecast future log returns
                    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

  # 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

# re-stack train and val
full_train <- rbind.xts(ticker_data_train, ticker_data_val)

# Calculate the Sharpe Ratio and MSR (on historical discrete returns)
scaling_factor <- as.vector(ticker_data_val$month_index)[1] - as.vector(ticker_data_train$month_index)[1]

# Pack returns and compute mean and std
hist_returns <- na.trim(as.vector(full_train[, "discrete_returns"]))
mean_rets <- mean(hist_returns)
std_rets <- sd(hist_returns)

# Calculate the ES and set risk-free
VaR <- quantile(hist_returns, 0.05)
ES <- mean(hist_returns[hist_returns < VaR])
Rf <- 0 # risk free rate

# Calculate the Sharpe and MSR
stock_sharpe <- ((mean_rets- Rf)/ std_rets ) * sqrt(scaling_factor) # annualized
stock_msr <- ((mean_rets- Rf)/ ES ) * sqrt(scaling_factor) # annualized

### 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("forecasted_ret: ", pred_enet_best))
print(paste("rmse: ", enet_rmse))
print(paste("sharpe: ", stock_sharpe))
```

```r
  print(paste("msr: ", stock_msr))
  print("*****************************************")

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

```
## [1] "ticker: ADP"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag2 + adx + macd + mfi + smi + sarima_100_011 +
##     sarima_010_011 + sarima_120_011 + volat + vol_forecast
## <environment: 0x000001f3746df218>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00350236346872877"
## [1] "rmse:  0.00859642398737655"
## [1] "sharpe:  0.889944339789262"
## [1] "msr:  -0.388083007960868"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: BA"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     atr + adx + aaron + clv + smi + volat + sarima_120_001 +
##     sarima_110_011 + vol_forecast
## <environment: 0x000001f3649c41a0>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00385792977265609"
## [1] "rmse:  0.0389181877273309"
## [1] "sharpe:  0.606180867847129"
## [1] "msr:  -0.316654391938"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: CAT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag2 +
##     atr + adx + bb + emv + sarima_020_001 + sarima_110_011 +
##     sarima_120_011 + vol_forecast + volat
## <environment: 0x000001f3675cc878>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.000879888058304028"
## [1] "rmse:  0.0426615114260573"
## [1] "sharpe:  0.327638399697175"
## [1] "msr:  -0.137015287468138"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: CSX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + atr + chaikin_vol +
##     emv + sar + sarima_120_001 + sarima_100_011 + vol_forecast +
##     volat
## <environment: 0x000001f366b0c1d8>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00529721245408163"
## [1] "rmse:  0.051593680097498"
## [1] "sharpe:  0.528667722941504"
## [1] "msr:  -0.242919009110399"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: DE"
```

```
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     emv + sarima_100_001 + sarima_110_001 + sarima_120_001 +
##     sarima_110_011 + volat + vol_forecast
## <environment: 0x000001f36ace3c10>
## [1] "******************************************"
## [1] "forecasted_ret:  0.000826998211211032"
## [1] "rmse:  0.0449125414430164"
## [1] "sharpe:  0.464535455970236"
## [1] "msr:  -0.214227466051945"
## [1] "******************************************"
## [1] "##########################################"
## [1] "ticker: ETN"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag2 + macd + sarima_010_011 + sarima_020_011 +
##     sarima_120_011 + volat + vol_forecast
## <environment: 0x000001f373c63420>
## [1] "******************************************"
## [1] "forecasted_ret:  0.00119418376329643"
## [1] "rmse:  0.0327711636734526"
## [1] "sharpe:  0.213981351336753"
## [1] "msr:  -0.0921431033485743"
## [1] "******************************************"
## [1] "##########################################"
## [1] "ticker: FDX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag0 + adjclose_lag1 + adjclose_lag2 + adjclose_lag3 +
##     chaikin_vol + clv + emv + macd + smi + sarima_110_011 + sarima_120_011 +
##     volat + vol_forecast
## <environment: 0x000001f362ee5658>
## [1] "******************************************"
## [1] "forecasted_ret:  -0.00243119359489796"
## [1] "rmse:  0.0441449680660968"
## [1] "sharpe:  -0.153673160336959"
## [1] "msr:  0.0571916185822411"
## [1] "******************************************"
## [1] "##########################################"
## [1] "ticker: GE"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag0 +
##     adjclose_lag3 + adx + bb + smi + sarima_020_001 + sarima_110_011 +
##     vol_forecast + volat
## <environment: 0x000001f375b1f8c8>
## [1] "******************************************"
## [1] "forecasted_ret:  -0.0161287852817406"
## [1] "rmse:  0.0540444010371181"
## [1] "sharpe:  -0.52708799830933"
## [1] "msr:  0.232855916072306"
## [1] "******************************************"
## [1] "##########################################"
## [1] "ticker: HON"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + atr + adx +
##     chaikin_vol + macd + mfi + smi + sarima_110_011 + sarima_120_011 +
##     vol_forecast + volat
## <environment: 0x000001f36f2f04d0>
## [1] "******************************************"
## [1] "forecasted_ret:  0.00341322116346939"
```

```
## [1] "rmse:  0.0354457717256096"
## [1] "sharpe:  0.676224835230474"
## [1] "msr:  -0.289148114906077"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: ITW"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag0 + adjclose_lag2 + adx + mfi + sarima_110_001 +
##     sarima_120_001 + volat + vol_forecast
## <environment: 0x000001f36eaf0a60>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00108522868163265"
## [1] "rmse:  0.0359857204941772"
## [1] "sharpe:  0.303539164202787"
## [1] "msr:  -0.126029104188648"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: LMT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag1 + adjclose_lag2 + aaron + chaikin_vol + smi +
##     sarima_110_001 + sarima_020_001 + sarima_120_001 + sarima_010_011 +
##     sarima_020_011 + volat + vol_forecast
## <environment: 0x000001f36f751788>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00245390640908315"
## [1] "rmse:  0.0216040507227898"
## [1] "sharpe:  0.510576395891447"
## [1] "msr:  -0.220913683417775"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: NOC"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag0 + adjclose_lag1 + atr + adx + aaron + bb +
##     emv + mfi + sarima_110_001 + sarima_120_011 + volat + vol_forecast
## <environment: 0x000001f371708d20>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00233639526938775"
## [1] "rmse:  0.033577751975452"
## [1] "sharpe:  0.490363650070614"
## [1] "msr:  -0.212512300791679"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: RTX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     chaikin_vol + sarima_010_001 + sarima_120_011 + volat + vol_forecast
## <environment: 0x000001f372dcf7e0>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.000659869551292678"
## [1] "rmse:  0.0230484094398749"
## [1] "sharpe:  0.332256967269015"
## [1] "msr:  -0.140923453879974"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: UNP"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
```

```
##     bb + smi + sarima_010_001 + sarima_110_001 + sarima_120_011 +
##     volat + vol_forecast
## <environment: 0x000001f3729858a8>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00515718972294101"
## [1] "rmse:  0.0479702843868829"
## [1] "sharpe:  0.985734581558049"
## [1] "msr:  -0.532727538072992"
## [1] "*****************************************"
## [1] "##########################################"
## [1] "ticker: UPS"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag0 + atr + aaron + mfi + sar + sarima_120_001 +
##     sarima_110_011 + volat + vol_forecast
## <environment: 0x000001f372b21888>
## [1] "*****************************************"
## [1] "forecasted_ret:  -0.000102464363265306"
## [1] "rmse:  0.0565886856154367"
## [1] "sharpe:  0.253379204482838"
## [1] "msr:  -0.10300576569634"
## [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] 0.003502363
##
## $ADP$sharpe
## [1] 0.8899443
##
## $ADP$msr
## [1] -0.388083
##
## $ADP$rmse
## [1] 0.008596424
##
## $ADP$data
## [1] NA
##
##
```

```
## $BA
## $BA$forecasted_ret
## [1] 0.00385793
##
## $BA$sharpe
## [1] 0.6061809
##
## $BA$msr
## [1] -0.3166544
##
## $BA$rmse
## [1] 0.03891819
##
## $BA$data
## [1] NA
##
##
## $CAT
## $CAT$forecasted_ret
## [1] 0.0008798881
##
## $CAT$sharpe
## [1] 0.3276384
##
## $CAT$msr
## [1] -0.1370153
##
## $CAT$rmse
## [1] 0.04266151
##
## $CAT$data
## [1] NA
##
##
## $CSX
## $CSX$forecasted_ret
## [1] 0.005297212
##
## $CSX$sharpe
## [1] 0.5286677
##
## $CSX$msr
## [1] -0.242919
##
## $CSX$rmse
## [1] 0.05159368
##
## $CSX$data
## [1] NA
##
##
## $DE
## $DE$forecasted_ret
## [1] 0.0008269982
##
## $DE$sharpe
## [1] 0.4645355
##
## $DE$msr
## [1] -0.2142275
```

```
##
## $DE$rmse
## [1] 0.04491254
##
## $DE$data
## [1] NA
##
##
## $ETN
## $ETN$forecasted_ret
## [1] 0.001194184
##
## $ETN$sharpe
## [1] 0.2139814
##
## $ETN$msr
## [1] -0.0921431
##
## $ETN$rmse
## [1] 0.03277116
##
## $ETN$data
## [1] NA
##
##
## $FDX
## $FDX$forecasted_ret
## [1] -0.002431194
##
## $FDX$sharpe
## [1] -0.1536732
##
## $FDX$msr
## [1] 0.05719162
##
## $FDX$rmse
## [1] 0.04414497
##
## $FDX$data
## [1] NA
##
##
## $GE
## $GE$forecasted_ret
## [1] -0.01612879
##
## $GE$sharpe
## [1] -0.527088
##
## $GE$msr
## [1] 0.2328559
##
## $GE$rmse
## [1] 0.0540444
##
## $GE$data
## [1] NA
##
##
## $HON
```

```
## $HON$forecasted_ret
## [1] 0.003413221
##
## $HON$sharpe
## [1] 0.6762248
##
## $HON$msr
## [1] -0.2891481
##
## $HON$rmse
## [1] 0.03544577
##
## $HON$data
## [1] NA
##
##
## $ITW
## $ITW$forecasted_ret
## [1] 0.001085229
##
## $ITW$sharpe
## [1] 0.3035392
##
## $ITW$msr
## [1] -0.1260291
##
## $ITW$rmse
## [1] 0.03598572
##
## $ITW$data
## [1] NA
##
##
## $LMT
## $LMT$forecasted_ret
## [1] 0.002453906
##
## $LMT$sharpe
## [1] 0.5105764
##
## $LMT$msr
## [1] -0.2209137
##
## $LMT$rmse
## [1] 0.02160405
##
## $LMT$data
## [1] NA
##
##
## $NOC
## $NOC$forecasted_ret
## [1] 0.002336395
##
## $NOC$sharpe
## [1] 0.4903637
##
## $NOC$msr
## [1] -0.2125123
##
```

```
## $NOC$rmse
## [1] 0.03357775
##
## $NOC$data
## [1] NA
##
##
## $RTX
## $RTX$forecasted_ret
## [1] 0.0006598696
##
## $RTX$sharpe
## [1] 0.332257
##
## $RTX$msr
## [1] -0.1409235
##
## $RTX$rmse
## [1] 0.02304841
##
## $RTX$data
## [1] NA
##
##
## $UNP
## $UNP$forecasted_ret
## [1] 0.00515719
##
## $UNP$sharpe
## [1] 0.9857346
##
## $UNP$msr
## [1] -0.5327275
##
## $UNP$rmse
## [1] 0.04797028
##
## $UNP$data
## [1] NA
##
##
## $UPS
## $UPS$forecasted_ret
## [1] -0.0001024644
##
## $UPS$sharpe
## [1] 0.2533792
##
## $UPS$msr
## [1] -0.1030058
##
## $UPS$rmse
## [1] 0.05658869
##
## $UPS$data
## [1] NA
```

```r
# Extract the top 3 tickers with the highest Sharpe ratio
top_sharpe <- names(sort(sapply(sector_tracker, function(x) x$sharpe), decreasing=TRUE))[1:3]
top_fore_rets <- names(sort(sapply(sector_tracker, function(x) x$forecasted_ret), decreasing=TRUE))[1:3]
```

```
# display selected stocks
top_fore_rets
```

```
## [1] "CSX" "UNP" "BA"
```

```
top_sharpe
```

```
## [1] "UNP" "ADP" "HON"
```

```
## TODO: Complete the function, keep the name and parameters
f_select_top_stocks <- function(sector_tracker, n=3){
  ## 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.

  # Extract the top 3 tickers with the highest Sharpe ratio
  top_sharpe <- names(sort(sapply(sector_tracker, function(x) x$sharpe), decreasing=TRUE))[1:n]
  top_fore_rets <- names(sort(sapply(sector_tracker, function(x) x$forecasted_ret), decreasing=TRUE))[1:n]

  # Concat in one list
  top_tickers  <-c(top_sharpe, top_fore_rets) # fix so that becomes a set

  # Create a new named list with tickers and their corresponding data
  best_stocks_data <- lapply(top_tickers, function(x) sector_tracker[[x]])
  names(best_stocks_data) <- top_tickers

  return(best_stocks_data)
}
```

```
# Obtain data for the top n*2 stocks (best forecasted rets and best sharpe together)
best_stocks_data <- f_select_top_stocks(sector_tracker = sector_tracker, n = 2)
names(best_stocks_data)
```

```
## [1] "UNP" "ADP" "CSX" "UNP"
```

```
names(best_stocks_data[[1]])
```

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

```
best_stocks_data
```

```
## $UNP
## $UNP$forecasted_ret
## [1] 0.00515719
##
## $UNP$sharpe
## [1] 0.9857346
##
## $UNP$msr
## [1] -0.5327275
##
```

```
## $UNP$rmse
## [1] 0.04797028
##
## $UNP$data
## [1] NA
##
##
## $ADP
## $ADP$forecasted_ret
## [1] 0.003502363
##
## $ADP$sharpe
## [1] 0.8899443
##
## $ADP$msr
## [1] -0.388083
##
## $ADP$rmse
## [1] 0.008596424
##
## $ADP$data
## [1] NA
##
##
## $CSX
## $CSX$forecasted_ret
## [1] 0.005297212
##
## $CSX$sharpe
## [1] 0.5286677
##
## $CSX$msr
## [1] -0.242919
##
## $CSX$rmse
## [1] 0.05159368
##
## $CSX$data
## [1] NA
##
##
## $UNP
## $UNP$forecasted_ret
## [1] 0.00515719
##
## $UNP$sharpe
## [1] 0.9857346
##
## $UNP$msr
## [1] -0.5327275
##
## $UNP$rmse
## [1] 0.04797028
##
## $UNP$data
## [1] NA
```