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")
```

```
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ADP, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CAT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CSX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ETN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker FDX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GE, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker HON, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ITW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker LMT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker NOC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker RTX, skipping...
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker UNP, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker UPS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BMY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DHR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ELV, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GILD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker LLY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MDT, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ACN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AMD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AVGO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CRM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker IBM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker INTC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker INTU, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker QCOM, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TXN, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ATVI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CHTR, skipping...
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CMCSA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker DIS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker EA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GOOG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker GOOGL, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker NFLX, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker OMC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker T, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TMUS, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TTWO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker VZ, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker WBD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CB, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MA, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MMC, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker PGR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker SCHW, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker SPGI, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker V, skipping...
## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ABNB, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker AZO, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker BKNG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker CMG, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MAR, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker MCD, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker ORLY, skipping...

## Warning in f_fetch_ind_base(x, from = from, to = to): No financial ratio data
## for ticker TJX, skipping...
```

```
# clean the environment memory
xts_fama_french <- NULL
xts_financial_ratios <- NULL
xts_realized_vol <- NULL
```

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

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

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

```
# 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"
```

```
# 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      0.053616030
## 2022-11-16       248.2840              1      0.055079400      0.034718650
## 2022-11-23       257.0555              1      0.035328370      0.005923636
## 2022-11-30       258.5827             NA      0.005941215               NA
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3      atr
## 2022-10-26   0.008113008   0.039931040  -0.064535800   0.030150980 9.676399
## 2022-11-02   0.009733913   0.008113008   0.039931040  -0.064535800 9.885942
## 2022-11-09   0.012306040   0.009733913   0.008113008   0.039931040  9.762661
## 2022-11-16   0.053616030   0.012306040   0.009733913   0.008113008 10.232471
## 2022-11-23   0.034718650   0.053616030   0.012306040   0.009733913 10.243009
## 2022-11-30   0.005923636   0.034718650   0.053616030   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  volume     volat month_index
## 2022-10-26 51.52422 260.0428  8.131402 2942400 0.2269538          82
## 2022-11-02 49.23300 258.6055  5.546375 1592400 0.2606250          83
## 2022-11-09 49.20839 257.2257  3.943960 1242900 0.2653165          83
## 2022-11-16 48.83463 256.7200  6.291102 1430800 0.2641173          83
## 2022-11-23 49.31528 224.1100 11.099826 1386300 0.2624611          83
## 2022-11-30 42.97382 224.1100 16.713518 4155500 0.2759187          83
##            Excess_Retun_Mkt Small_minus_Big High_minus_Low Robus_minus_Weak
## 2022-10-26          -0.0066          0.0070         0.0089          -0.0080
## 2022-11-02          -0.0267         -0.0087         0.0161           0.0021
## 2022-11-09          -0.0225         -0.0052         0.0055           0.0095
## 2022-11-16          -0.0103         -0.0107         0.0057           0.0119
## 2022-11-23           0.0063         -0.0024        -0.0094          -0.0075
## 2022-11-30           0.0312         -0.0015        -0.0207          -0.0077
##            Conservative_minus_Aggressive Risk_free_rate Momentum
## 2022-10-26                        0.0067        0.00011   0.0049
## 2022-11-02                        0.0105        0.00014   0.0216
## 2022-11-09                        0.0106        0.00014   0.0164
## 2022-11-16                        0.0093        0.00014   0.0269
## 2022-11-23                       -0.0057        0.00014  -0.0184
## 2022-11-30                       -0.0141        0.00014  -0.0282
```

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.

```
# 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,
                          arima_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
```

```
####### 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]])
```

```
##            adjusted_close direction_lead discrete_returns realized_returns
## 2016-10-05       75.58761             -1      0.001486751     -0.008139792
## 2016-10-12       74.97484              1     -0.008106753      0.006425667
## 2016-10-19       75.45815             -1      0.006446356     -0.002749049
## 2016-10-26       75.25100              1     -0.002745274      0.031497830
## 2016-11-02       77.65897              1      0.031999130      0.010172840
## 2016-11-09       78.45301              1      0.010224760      0.025738380
##            adjclose_lag0 adjclose_lag1 adjclose_lag2 adjclose_lag3      atr
## 2016-10-05   0.001485647  -0.016219880   0.024948200  -0.037026570 1.900259
## 2016-10-12  -0.008139792   0.001485647  -0.016219880   0.024948200 1.872384
## 2016-10-19   0.006425667  -0.008139792   0.001485647  -0.016219880 1.800070
## 2016-10-26  -0.002749049   0.006425667  -0.008139792   0.001485647 1.722923
## 2016-11-02   0.031497830  -0.002749049   0.006425667  -0.008139792 1.864142
## 2016-11-09   0.010172840   0.031497830  -0.002749049   0.006425667 1.989560
##                 adx aaron        bb chaikin_vol         clv           emv
## 2016-10-05 15.44565   -50 0.2934560  -0.4622892  0.18091008 -0.0006643160
## 2016-10-12 15.23639  -100 0.2289285   0.3990933  0.24064338 -0.0026850063
## 2016-10-19 14.75791   -50 0.3060118  -0.4336751  0.09899013 -0.0019094937
## 2016-10-26 14.44363   100 0.2860935  -1.0188680 -0.01496489 -0.0021492280
## 2016-11-02 14.04553    50 0.4910556 -324.8278000  0.05096933 -0.0009225739
## 2016-11-09 13.44222   100 0.5094234   1.1391500  0.19338517 -0.0009562142
##                 macd      mfi      sar        smi  volume      volat
## 2016-10-05 1.3477744 46.50802 95.02127  -5.331162 1315500 0.10247324
## 2016-10-12 1.1358585 37.92195 94.68802 -11.930732 1139000 0.10506831
## 2016-10-19 0.9402188 36.19915 94.36810 -17.430099  906400 0.10335977
## 2016-10-26 0.7585276 30.28217 94.06097 -19.828752 1331500 0.09985285
## 2016-11-02 0.6437468 48.88575 93.76613 -18.073978 5356600 0.13389984
## 2016-11-09 0.5919089 59.37208 93.48309 -13.909935 2861300 0.16512456
##            month_index Excess_Retun_Mkt Small_minus_Big High_minus_Low
## 2016-10-05          10           0.0058          0.0042         0.0080
## 2016-10-12          10           0.0006         -0.0022         0.0034
## 2016-10-19          10           0.0025          0.0013         0.0094
## 2016-10-26          10          -0.0023         -0.0073         0.0070
## 2016-11-02          11          -0.0073         -0.0056         0.0025
## 2016-11-09          11           0.0146          0.0213         0.0107
##            Robus_minus_Weak Conservative_minus_Aggressive Risk_free_rate
## 2016-10-05          -0.0048                        0.0044          1e-05
## 2016-10-12           0.0067                        0.0014          1e-05
## 2016-10-19          -0.0011                        0.0049          1e-05
## 2016-10-26           0.0012                        0.0051          1e-05
## 2016-11-02           0.0097                        0.0011          1e-05
## 2016-11-09          -0.0081                        0.0072          1e-05
##            Momentum sarima_100_001 sarima_010_001 sarima_110_001 sarima_020_001
## 2016-10-05  -0.0075    0.003087314     0.03149783    0.012973640     0.06574471
## 2016-10-12   0.0051    0.005293360     0.01017284    0.021707560    -0.01115215
## 2016-10-19  -0.0033    0.003683123     0.02573838    0.017318955     0.04130392
## 2016-10-26  -0.0081    0.002663206     0.03559752    0.030264696     0.04545666
## 2016-11-02   0.0008    0.007022251    -0.00653968    0.016252397    -0.04867688
## 2016-11-09  -0.0200    0.004073060     0.02196901    0.006548615     0.05047770
##            sarima_120_001 sarima_100_011 sarima_010_011 sarima_110_011
## 2016-10-05   3.470593e-02    0.003087314     0.03149783    0.012973640
## 2016-10-12   2.857194e-02    0.005293360     0.01017284    0.021707560
## 2016-10-19   1.493370e-02    0.003683123     0.02573838    0.017318955
## 2016-10-26   4.953573e-02    0.002663206     0.03559752    0.030264696
## 2016-11-02  -1.150867e-02    0.007022251    -0.00653968    0.016252397
## 2016-11-09  -2.165281e-05    0.004073060     0.02196901    0.006548615
##            sarima_020_011 sarima_120_011 best_shifted_arima vol_forecast
## 2016-10-05     0.06574471   3.470593e-02       3.470593e-02    0.1338998
## 2016-10-12    -0.01115215   2.857194e-02       2.857194e-02    0.1651246
## 2016-10-19     0.04130392   1.493370e-02       1.493370e-02    0.1746223
## 2016-10-26     0.04545666   4.953573e-02       4.953573e-02    0.1752898
## 2016-11-02    -0.04867688  -1.150867e-02      -1.150867e-02    0.1772747
## 2016-11-09     0.05047770  -2.165281e-05      -2.165281e-05    0.1757262
```

```
# save data in tests
save(list_xts_sector, file = here("tests","jair", "sample_data.rda"))
```

The result is the `list_train_val_sector` oject, 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 = 50, # maximum number of subsets to examine
                    method="backward") #  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, : 6 linear dependencies found
```

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

```
## Warning in rval$lopt[] <- rval$vorder[rval$lopt]: number of items to replace is
## not a multiple of replacement length
```

```
print("")
```

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

```r
best_feat_list
```

```
## $featnames
##  [1] "adjusted_close"                 "direction_lead"
##  [3] "adjclose_lag0"                  "adjclose_lag1"
##  [5] "adjclose_lag2"                  "adjclose_lag3"
##  [7] "clv"                            "emv"
##  [9] "macd"                           "mfi"
## [11] "smi"                            "volume"
## [13] "Conservative_minus_Aggressive" "sarima_110_001"
## [15] "sarima_020_001"                 "sarima_120_001"
## [17] "vol_forecast"                   "volat"
##
## $fmla
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag0 +
##     adjclose_lag1 + adjclose_lag2 + adjclose_lag3 + clv + emv +
##     macd + mfi + smi + volume + Conservative_minus_Aggressive +
##     sarima_110_001 + sarima_020_001 + sarima_120_001 + vol_forecast +
##     volat
## <environment: 0x000001ab2049e5c8>
```

Regularized MLR (Elasticnet)

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

```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 = 0.05))  # 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

```
library("glmnet")

system.time(
  # 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

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

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

    ### 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="forward")
```

```r
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

# Reestimating the model using all training data
X_train <- model.matrix(best_feat_list$fmla, data = ticker_data_train)
X_test <- model.matrix(best_feat_list$fmla, data = ticker_data_val)
y_train <- ticker_data_train[, "realized_returns"]

# refit the model and assign test
refitted_model <- glmnet(X_train, y_train, alpha = best_alpha, lambda = best_lambda, standardize = TRUE)

# Use the best-fitted elastic net regression model to make predictions on the val_data
pred_enet_best <- predict(refitted_model, newx = X_test, s = refitted_model$lambda, type = "response")
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 (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 <- 0.002

# Calculate the Sharpe and MSR
```

```r
    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 = full_train[, c("realized_returns",
                                                   "best_shifted_arima",
                                                   "volat",
                                                   "vol_forecast")] # features to be kept

    # show values
    print("******************************************")
    print(paste("forecasted_ret: ", 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 + discrete_returns + adjclose_lag2 +
##     adx + bb + clv + mfi + smi + Conservative_minus_Aggressive +
##     Risk_free_rate + sarima_100_001 + sarima_110_001 + sarima_120_001 +
##     vol_forecast + volat
## <environment: 0x000001ab240665b0>
## [1] "******************************************"
## [1] "forecasted_ret:  0.00555963324121212"
## [1] "rmse:  0.00730273302731386"
## [1] "sharpe:  0.733763077285422"
## [1] "msr:  -0.295462035468019"
## [1] "******************************************"
## [1] "############################################"
## [1] "ticker: BA"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adx + chaikin_vol +
##     clv + macd + mfi + sar + smi + volume + Excess_Retun_Mkt +
##     Conservative_minus_Aggressive + Risk_free_rate + Momentum +
##     sarima_100_001 + sarima_120_001 + sarima_110_011 + vol_forecast +
##     volat
## <environment: 0x000001ab252b11d8>
## [1] "******************************************"
## [1] "forecasted_ret:  0.0103217028136329"
## [1] "rmse:  0.0335982498746814"
## [1] "sharpe:  1.394541522364"
## [1] "msr:  -0.848736740701376"
## [1] "******************************************"
## [1] "############################################"
## [1] "ticker: CAT"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag2 +
##     atr + adx + bb + chaikin_vol + clv + emv + sar + smi + volat +
##     Excess_Retun_Mkt + Small_minus_Big + Robus_minus_Weak + Risk_free_rate +
##     sarima_110_001 + vol_forecast
## <environment: 0x000001ab1822fc48>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00858315573133205"
## [1] "rmse:  0.0285444146315692"
## [1] "sharpe:  0.659415111133307"
## [1] "msr:  -0.354528305758196"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: CSX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag3 +
##     atr + adx + aaron + bb + macd + mfi + sar + volume + volat +
##     Excess_Retun_Mkt + Small_minus_Big + High_minus_Low + Risk_free_rate +
##     Momentum + sarima_010_001 + sarima_020_001 + sarima_120_001 +
##     vol_forecast
## <environment: 0x000001ab190e3678>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00923644755151515"
## [1] "rmse:  0.00993860866720442"
## [1] "sharpe:  0.882532180929076"
## [1] "msr:  -0.642363718881039"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: DE"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     atr + aaron + chaikin_vol + clv + smi + volat + Excess_Retun_Mkt +
##     Small_minus_Big + Momentum + sarima_110_011 + sarima_120_011 +
##     vol_forecast
## <environment: 0x000001ab131f9328>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00571415448282828"
## [1] "rmse:  0.0235265545539995"
## [1] "sharpe:  0.715083150809966"
## [1] "msr:  -0.363791996548408"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: ETN"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     adjclose_lag2 + atr + adx + bb + clv + emv + macd + mfi +
##     sar + smi + volume + volat + Momentum + sarima_010_001 +
##     sarima_110_001 + sarima_120_001 + vol_forecast
## <environment: 0x000001ab15cb2888>
## [1] "****************************************"
## [1] "forecasted_ret:  -0.0570579784377226"
## [1] "rmse:  0.0675961509865496"
## [1] "sharpe:  0.344960643552403"
## [1] "msr:  -0.190670357617281"
## [1] "****************************************"
## [1] "########################################"
## [1] "ticker: FDX"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + atr + adx + aaron + bb +
##     clv + mfi + sar + volume + Excess_Retun_Mkt + sarima_120_011 +
##     vol_forecast + volat
## <environment: 0x000001ab1fc01db8>
## [1] "****************************************"
## [1] "forecasted_ret:  0.00212466279071276"
## [1] "rmse:  0.0276605815308649"
## [1] "sharpe:  0.298804192251413"
## [1] "msr:  -0.144485314671452"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: GE"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adjclose_lag0 + mfi + smi +
##     volat + Small_minus_Big + sarima_100_011 + vol_forecast
## <environment: 0x000001ab2c89fac8>
## [1] "*****************************************"
## [1] "forecasted_ret:  -0.00526786957258477"
## [1] "rmse:  0.0776113628368765"
## [1] "sharpe:  -1.50474376242996"
## [1] "msr:  0.591755081874018"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: HON"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     adjclose_lag2 + adjclose_lag3 + atr + adx + aaron + clv +
##     emv + macd + mfi + sar + smi + volat + Robus_minus_Weak +
##     Risk_free_rate + sarima_110_001 + sarima_020_001 + sarima_120_001 +
##     vol_forecast
## <environment: 0x000001ab2bd98248>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00413636705997845"
## [1] "rmse:  0.00655217990339337"
## [1] "sharpe:  0.50829473304487"
## [1] "msr:  -0.221461303419631"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: ITW"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + adjclose_lag1 + adjclose_lag2 +
##     atr + bb + clv + macd + mfi + sar + volume + volat + Small_minus_Big +
##     Robus_minus_Weak + Risk_free_rate + sarima_110_001 + sarima_020_001 +
##     sarima_120_011 + vol_forecast
## <environment: 0x000001ab2716d2a8>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00207220955858586"
## [1] "rmse:  0.022399258755137"
## [1] "sharpe:  0.0654746204178788"
## [1] "msr:  -0.0281379147725384"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: LMT"
## Reordering variables and trying again:
## realized_returns ~ direction_lead + discrete_returns + adjclose_lag0 +
##     adjclose_lag1 + adjclose_lag2 + adjclose_lag3 + chaikin_vol +
##     clv + macd + mfi + volume + volat + Excess_Retun_Mkt + Small_minus_Big +
##     Conservative_minus_Aggressive + Momentum + sarima_110_001 +
##     sarima_020_001 + sarima_120_011 + vol_forecast
## <environment: 0x000001ab182805e0>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00360129766646465"
## [1] "rmse:  0.0206388618044935"
## [1] "sharpe:  0.401847818694071"
## [1] "msr:  -0.19914259326524"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: NOC"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + discrete_returns +
##     adjclose_lag1 + adjclose_lag2 + atr + adx + aaron + mfi +
##     sar + smi + volat + Excess_Retun_Mkt + Small_minus_Big +
##     High_minus_Low + Conservative_minus_Aggressive + Momentum +
##     sarima_010_001 + sarima_020_001 + sarima_120_001 + vol_forecast
## <environment: 0x000001ab24d22950>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00501053137986081"
## [1] "rmse:  0.014843163077175"
## [1] "sharpe:  0.400962762641132"
## [1] "msr:  -0.167683873611341"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: RTX"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag1 +
##     adjclose_lag3 + atr + aaron + chaikin_vol + clv + macd +
##     smi + volat + Excess_Retun_Mkt + Small_minus_Big + Robus_minus_Weak +
##     Risk_free_rate + Momentum + sarima_110_001 + sarima_120_001 +
##     vol_forecast
## <environment: 0x000001ab2c7be5b8>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00313481875674293"
## [1] "rmse:  0.0234876948444603"
## [1] "sharpe:  0.38724148489548"
## [1] "msr:  -0.188194889426545"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: UNP"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag0 +
##     adjclose_lag1 + adjclose_lag2 + adjclose_lag3 + atr + adx +
##     aaron + clv + emv + macd + smi + volat + Excess_Retun_Mkt +
##     Small_minus_Big + Conservative_minus_Aggressive + Risk_free_rate +
##     sarima_110_001 + sarima_120_011 + vol_forecast
## <environment: 0x000001ab1c20d018>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00566941089555962"
## [1] "rmse:  0.016917578580181"
## [1] "sharpe:  0.70386459479901"
## [1] "msr:  -0.379628544552588"
## [1] "*****************************************"
## [1] "#########################################"
## [1] "ticker: UPS"
## Reordering variables and trying again:
## realized_returns ~ adjusted_close + direction_lead + adjclose_lag0 +
##     adjclose_lag1 + adjclose_lag2 + adjclose_lag3 + atr + adx +
##     bb + chaikin_vol + clv + macd + sar + smi + volat + Excess_Retun_Mkt +
##     Robus_minus_Weak + Conservative_minus_Aggressive + sarima_020_001 +
##     sarima_120_001 + sarima_100_011 + vol_forecast
## <environment: 0x000001ab18076db8>
## [1] "*****************************************"
## [1] "forecasted_ret:  0.00112842013438798"
## [1] "rmse:  0.0237769076660908"
## [1] "sharpe:  -0.0684476408320746"
## [1] "msr:  0.0278995188162262"
## [1] "*******************************************"
## [1] "###########################################"
```

```
##    user  system elapsed
##   40.04    0.17   54.36
```

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:

```
names(sector_tracker)
```

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

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

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

```
source(here("functions","modelling.R"))

# Obtain the top picks with the function
best_sector_stocks <- f_select_top_stocks(sector_tracker, n=3)
names(best_sector_stocks)
```

```
## [1] "BA"  "CSX" "ADP" "CAT"
```

```
best_sector_stocks
```

```
## $BA
## $BA$forecasted_ret
## [1] 0.0103217
##
## $BA$sharpe
## [1] 1.394542
##
## $BA$msr
## [1] -0.8487367
##
## $BA$rmse
## [1] 0.03359825
##
## $BA$data
##            realized_returns best_shifted_arima     volat vol_forecast
## 2016-10-05    -0.0112018172       -0.036045569 0.1215014    0.2190119
## 2016-10-12     0.0224261785       -0.006408533 0.1234677    0.2294365
## 2016-10-19     0.0664736246        0.048285397 0.1197529    0.2335310
## 2016-10-26    -0.0334661111        0.007056710 0.2165795    0.2328868
## 2016-11-02     0.0380191872        0.008819670 0.2190119    0.2368343
## 2016-11-09     0.0092614702        0.017965135 0.2294365    0.2391726
## 2016-11-16     0.0222846534        0.007417973 0.2335310    0.2369913
## 2016-11-23     0.0054615221        0.010412039 0.2328868    0.2381453
## 2016-11-30     0.0234993535       -0.006297921 0.2368343    0.2360335
## 2016-12-07     0.0021384930        0.005049842 0.2391726    0.1529022
##        ...
## 2018-07-25    -0.0089181155        0.065700737 0.2219085    0.2383571
## 2018-08-01    -0.0142177462        0.048047696 0.2317340    0.2350262
## 2018-08-08    -0.0422293370       -0.049422468 0.2311597    0.2307443
## 2018-08-15     0.0536070401        0.021594299 0.2433951    0.2134922
## 2018-08-22     0.0004570828        0.057101563 0.2383571    0.2207011
## 2018-08-29    -0.0100737307       -0.002107368 0.2350262    0.2103641
## 2018-09-05     0.0192266918        0.072968341 0.2307443    0.2101598
## 2018-09-12     0.0328710072        0.122334867 0.2134922    0.2099818
## 2018-09-19    -0.0005202921        0.138943845 0.2207011    0.2098267
## 2018-09-26     0.0720473115        0.177702599 0.2103641    0.2096916
##
##
## $CSX
## $CSX$forecasted_ret
## [1] 0.009236448
##
## $CSX$sharpe
## [1] 0.8825322
##
## $CSX$msr
## [1] -0.6423637
##
## $CSX$rmse
## [1] 0.009938609
##
## $CSX$data
##            realized_returns best_shifted_arima     volat vol_forecast
## 2016-10-05    -0.0164151114      -0.0163788250 0.1525324    0.1623525
## 2016-10-12     0.0280696933       0.1450207009 0.1445769    0.2170363
## 2016-10-19    -0.0224584189       0.0368840596 0.1564746    0.2146174
## 2016-10-26     0.0117804360      -0.0327855146 0.1562740    0.2039793
## 2016-11-02     0.0972600941       0.0526983311 0.1623525    0.2093311
## 2016-11-09    -0.0002951691       0.0372371675 0.2170363    0.2207179
## 2016-11-16     0.0308163362      -0.0273185628 0.2146174    0.2191759
## 2016-11-23     0.0300096128      -0.0342932378 0.2039793    0.2193127
## 2016-11-30     0.0361979812       0.0139516552 0.2093311    0.2120542
## 2016-12-07    -0.0176611988       0.0036563835 0.2207179    0.2157915
##        ...
## 2018-07-25    -0.0046620147      -0.0059180962 0.2144367    0.2012677
## 2018-08-01     0.0268246068       0.0159480038 0.2152854    0.1948769
## 2018-08-08     0.0085107647      -0.0038755710 0.2122352    0.2010700
## 2018-08-15     0.0077608611      -0.0154304602 0.2063714    0.1845599
## 2018-08-22     0.0146753724      -0.0006112724 0.2012677    0.1781337
## 2018-08-29    -0.0040294802      -0.0009075892 0.1948769    0.1537379
## 2018-09-05    -0.0021559605       0.0194215352 0.2010700    0.1624511
## 2018-09-12    -0.0020249639       0.0323569077 0.1845599    0.1698206
## 2018-09-19    -0.0012171806       0.0392887201 0.1781337    0.1761119
## 2018-09-26     0.0146419150       0.0506398347 0.1537379    0.1815207
##
##
## $ADP
## $ADP$forecasted_ret
## [1] 0.005559633
##
## $ADP$sharpe
## [1] 0.7337631
##
## $ADP$msr
## [1] -0.295462
##
```

```
## $ADP$rmse
## [1] 0.007302733
##
## $ADP$data
##            realized_returns best_shifted_arima      volat vol_forecast
## 2016-10-05     -0.008139792       3.470593e-02 0.10247324    0.1338998
## 2016-10-12      0.006425667       2.857194e-02 0.10506831    0.1651246
## 2016-10-19     -0.002749049       1.493370e-02 0.10335977    0.1746223
## 2016-10-26      0.031497830       4.953573e-02 0.09985285    0.1752898
## 2016-11-02      0.010172840      -1.150867e-02 0.13389984    0.1772747
## 2016-11-09      0.025738380      -2.165281e-05 0.16512456    0.1757262
## 2016-11-16      0.035597520       1.569318e-02 0.17462225    0.1786333
## 2016-11-23     -0.006539680       4.621780e-02 0.17528980    0.1788125
## 2016-11-30      0.021969010       2.690016e-02 0.17727467    0.1800747
## 2016-12-07      0.001229041      -2.413743e-02 0.17572623    0.1792691
##        ...
## 2018-07-25     -0.048140270       1.593988e-03 0.14677140    0.1629356
## 2018-08-01      0.038536310       1.576618e-02 0.16670545    0.1629302
## 2018-08-08      0.024511920      -3.147352e-03 0.16536364    0.1650939
## 2018-08-15      0.014479860      -6.994768e-03 0.16624802    0.1513876
## 2018-08-22      0.021060650       8.570461e-03 0.16293565    0.1481179
## 2018-08-29     -0.001435956       1.969220e-02 0.16293021    0.1538378
## 2018-09-05      0.006751476       2.190612e-02 0.16509395    0.1580478
## 2018-09-12      0.003612930       2.099944e-02 0.15138758    0.1613970
## 2018-09-19      0.017869940       2.704952e-02 0.14811786    0.1640755
## 2018-09-26      0.013080660       2.812675e-02 0.15383781    0.1662261
##
##
## $CAT
## $CAT$forecasted_ret
## [1] 0.008583156
##
## $CAT$sharpe
## [1] 0.6594151
##
## $CAT$msr
## [1] -0.3545283
##
## $CAT$rmse
## [1] 0.02854441
##
## $CAT$data
##            realized_returns best_shifted_arima     volat vol_forecast
## 2016-10-05     -0.020791766       -0.064210821 0.1566718    0.1741676
## 2016-10-12      0.004783920        0.167444490 0.1514927    0.2232971
## 2016-10-19     -0.036184948        0.095847696 0.1583610    0.2283652
## 2016-10-26     -0.036556952       -0.031116308 0.1664951    0.2319594
## 2016-11-02      0.117249043       -0.013971664 0.1741676    0.2346487
## 2016-11-09      0.023300482        0.002076906 0.2232971    0.2272987
## 2016-11-16      0.029865659       -0.039248173 0.2283652    0.2279674
## 2016-11-23     -0.006467126       -0.024130381 0.2319594    0.2267511
## 2016-11-30      0.018352774        0.016031658 0.2346487    0.2293654
## 2016-12-07     -0.037582057        0.002935270 0.2272987    0.2220408
##        ...
## 2018-07-25     -0.013906258        0.049294171 0.2327273    0.2499901
## 2018-08-01      0.008409750        0.078368811 0.2445062    0.2482561
## 2018-08-08     -0.056615388       -0.044153282 0.2386050    0.2493593
## 2018-08-15      0.056042618        0.020663568 0.2547456    0.2356708
## 2018-08-22      0.015844655        0.090800966 0.2499901    0.2277853
## 2018-08-29     -0.008993156        0.005797540 0.2482561    0.2237265
## 2018-09-05      0.025908101        0.005618888 0.2493593    0.2241639
## 2018-09-12      0.057112280        0.016820483 0.2356708    0.2245925
## 2018-09-19      0.002680369        0.002471323 0.2277853    0.2250125
## 2018-09-26      0.032438174        0.005292756 0.2237265    0.2254241
```