Report (ML Fin Data - Project 1)

Hair Parra, XiaoXue, Kriti Bhaya, Prateek Sinha

Note: This markdown is for illustration purposes only, so the code is not meant to run as the imports and logic are built in other scripts. The script `run_strategy.R` contains the code to run the simulation. Note that at the present time, the simulation takes ~2h to run, due to the amount of data and techniques employed. Future optimizations are expected.

1. Introduction

In this project, we endeavor to leverage various machine learning techniques to forecast stock returns and build a diversified portfolio. We particularly focus on stocks from varied economic sectors in the S&P 500 dataset. Through feature selection via backward subset modelling, incorporating both static and dynamic predictors like ARIMA and GARCH, and the deployment of the Elastic Net Regression model, our goal is to create a well performing portfolio. Furthermore, a random forest classifier aids in predicting the direction of the returns, strengthening our forecasting model.

To improve our strategies, we select the top stocks based on projected returns, the historical Sharpe ratio and the forecasted returns direction. These selected stocks are then optimized to build portfolios that balance both risk and reward.

2. Data Preparation

2.1 SP500 Economic Sectors

In our project, we extract the SP500 data, focusing on varied economic sectors. This broad and diverse dataset provides us with a comprehensive view of the market dynamics and also ensures that our models are well-trained and capable of handling real-world market fluctuations.

The algorithm fetches the SP500 components and their respective economic sectors from Wikipedia and returns their weight and number of shares held in the SP500 portfolio.

```
# load sp500 data into env
sp500 <- f_load_sp500() # dSP500 components and weights from yahoo finance
sp500_sectors <- f_get_sp500_sectors() # economic sectors from wikipedia
```

2.2 Retrieving top sectors and stocks

It then retrieves the top n stocks with their highest weight for each economic sector identified above.

```
# 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)
```

```
## Example usage:
# # Retrieve top 10 stocks by weight for each sector in the top 5 sectors from the SP500 (by
# weight)
# sector_list <- retrieve_top_sp500(top_n_sectors = 5, top_n_stocks = 10, only_tickers=TRUE)
# sector_list
#
# OUT:
#
#   $`Health Care`
# [1] "ABBV" "ABT"  "AMGN" "DHR"  "JNJ"  "LLY"  "MRK"  "PFE"  "TMO"  "UNH"
#
# $`Information Technology`
# [1] "AAPL" "ACN"  "ADBE" "AMD"  "AVGO" "CRM"  "CSCO" "MSFT" "NVDA" "ORCL"
#
# $`Communication Services`
# [1] "ATVI"  "CMCSA" "DIS"   "GOOG"  "GOOGL" "META"  "NFLX"  "T"     "TMUS"  "VZ"
#
# $Financials
# [1] "BAC"   "BRK-B" "GS"    "JPM"   "MA"    "MMC"   "MS"    "SPGI"  "V"     "WFC"
#
# $`Consumer Discretionary`
# [1] "ABNB" "AMZN" "BKNG" "HD"   "LOW"  "MCD"  "NKE"  "SBUX" "TJX"  "TSLA"
```

2.2.1 Retrieving stock data

The data for each of these stocks is retrieved from Yahoo finance and their weekly returns are calculated. A few financial indicators like ADX, ATR, Aroon, etc are calculated and added as features to this data. It also performs feature engineering to add numerous other parameters like Fama French factors, Financial ratios, etc. retrieved from Compustat.

The resulting dataset is saved as a master list (a list of lists) containing the following: * The first level as the economic sectors * The second level as the stock tickers, each containing an xts time series with their complete dataset

```
# function to fetch all the information for one ticker into a nice xts dataframe
system.time(
  sp500_stocks <- lapply(sector_list,
                         f_fetch_all_tickers,
                         start_date="2017-01-01",
                         end_date="2022-12-01")
)
```

2.2.2 Adding a numeric index

The data-fetching logic includes addition of a numerical index (known as month_index) indicating to which month in the simulation the observations belong.

3. Machine Learning Components

3.1 Modelling Procedure

The comprehensive **MODELLING_PROCEDURE** oversees the application of a detailed **SECTOR_PROCEDURE** (G, τ) function, systematically iterating through each distinct sector `G` and navigating through different time windows, identified by τ. Here, `G` represents the sector being analyzed, and τ indicates the current iteration within the backtesting framework.

3.1.1 Sector and Time Window Definition

Defining the Sectors For a specific sector `G`, comprising tickers $\{S_1, S_2, \ldots, S_{|G|}\}$, where $|G|$ quantifies the number of stocks within the sector (prior to selection), our goal is to compute and examine a particular time window to ensure the precision and relevance of the subsequent analytic steps.

Time Window Calibration The time window is determined as:

$$[t_1 = \text{week } W_{s\times\tau} \ , \ t_{12} = \text{week } W_{s\times\tau+11}]$$

In a sliding window approach where $S = 1$,

$$\begin{cases} \tau = 1 \implies [t_1 = W_1 \ , \ t_{12} = W_{12}] \\ \tau = 2 \implies [t_1 = W_2 \ , \ t_{12} = W_{13}] \\ \vdots \\ \tau = i \implies [t_1 = W_i \ , \ t_{12} = W_{i+11}] \\ \vdots \\ \tau = T \implies [t_1 = W_{T-12} \ , \ t_{12} = W_T] \end{cases}$$

3.1.2 Sector Procedure

This procedure ultimately grants us a segmented view of our data, allowing us to employ machine learning strategies that are sensitive to both stock and time-specific details

The Sector procedure performs the following: * The sector data is retrieved and the sector's tickers are stored. * Utilize the **`f_extract_window()`** function to extract a segment (window) of data defined by τ and the window size. * Further, dynamic features are extracted via **`f_extract_dynamic_features()`** for each stock within the sector.

```
# function that extracts data for the window only, but does not split the data into train-val
f_extract_window <- function (stock_data, tau = NULL, n_months = 12){
  ##
  ## Params:
  ##  - stock_data (xts): An xts object contianing the features and target for the stock
  ##  - tau (int): the current run number used to split the data
  ##  - n_months (int): the number of months to consider for the window

  # Calculate the beginning and end of the current window
  t_start <- tau
  t_end <- tau + n_months - 1
```

```
  # Subset the appropriate train and test sets for that stock
  data_sub = stock_data[(stock_data$month_index >= t_start) &
                          (stock_data$month_index <= t_end)]

  return(data_sub)
}
```

```
f_extract_dynamic_features <- function(stock_data,
                                       arima_col="realized_returns",
                                       volat_col="volat"){
  ## Computes and incorporates GARCH(1,1) 4-days-shifted forecasted volatility features
  ## as well as 4-days-shifted forecasted returns ARIMA(p,d,q) features for all combinations
  ## of p,d,q in {0,1}.
  ##
  ## Params:
  ##    - stock_data (xts): A date-indexed xts object containing features for a stock.
  ##    - retun_col (str): Specifies the name of the column containing the realized returns
  ##    - volat_col (str): Specifies the name of the column containing the volatility

  require("xts")
  require("forecast")
  require("rugarch")

  # Compute the ARIMA features
  stock_data <- f_add_arima_forecast(stock_data, arima_col=arima_col)

  # Compute the GARCH features
  stock_data <- f_add_garch_forecast(stock_data, volat_col=volat_col)

  return(stock_data)
}
```

3.1.3 Implementing the Regression Model and classification technique

3.1.3.1 Backward Step-wise Regression for Feature Selection In the stock prediction context, efficient feature selection plays a pivotal role in enhancing model performance by simplifying models, reducing over-fitting, and potentially improving prediction accuracy. Backward Step-wise Regression is a viable approach for such a selection, especially when dealing with numerous features. The function **`f_select_features`** is designed to conduct this step.

3.1.3.2 Model Training with Elastic Net Regression Elastic Net regression is a hybrid model that combines penalties from both Lasso and Ridge regression. The objective function it minimizes is defined as:

$$\min_{\beta_0,\beta} \frac{1}{N} \sum_{i=1}^{N} l\left(y_i, \beta_0 + \beta' x_i\right) + \lambda \left[\frac{1}{2}(1-\alpha)\|\beta\|_2^2 + \alpha\|\beta\|_1\right]$$

Here $l(y, \eta)$ is the negative log-likelihood contribution for observation i. The elastic-net penalty is controlled by α, and bridges the gap between lasso ($\alpha = 1$, the default) and ridge $\alpha = 0$. The tuning parameter λ controls the overall strength of the penalty (Simon et al., 2011).

Training the Model to get the best hyperparameters: α^* and λ^* The model is trained using a 10-fold cross-validation, which helps in obtaining a model with optimal hyperparameters while avoiding overfitting.

Refitting and predicting The elastic net model will be refitted by using the α^* and λ^* and the total training data.

Model Validation The model's predictive accuracy is evaluated using the Root Mean Square Error (RMSE), which is a standard metric to assess the model prediction error. Mathematically, RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where: - y_i is the actual value, - $\hat{y}_i$ is the predicted value, and n is the total number of observations.

3.1.3.3 Classification Technique: Random Forest Model Random Forest leverages the power of multiple decision trees for classification. In the context of stock prediction, especially regarding the direction of stock movement (i.e., whether it will go "up" or "down"), Random Forest acts as a viable predictive model due to its ability to manage non-linearities and interactions among variables.

Model Training and Hyperparameter Tuning Utilizing the `ranger` algorithm, a variant of Random Forest, the model is trained with a 10-fold cross-validation to ensure robustness and to minimize overfitting. This technique divides the dataset into ten subsets, sequentially using nine for training and one for validation, across all possible arrangements, mitigating bias and offering a dependable error estimation.

Predictive Modeling for Stock Movement The trained model predicts the probability of stock movement direction (`up` or `down`) for the validation set. It provides a probabilistic forecast for each class, offering a binary outcome.

Determining Aggregate Movement Direction Based on the average probability, a final forecast is formed: if the mean probability of `up` movements exceeds 0.5, the overall movement is predicted as `up`; otherwise, it's designated as `down`. This approach ensures that the final prediction considers the collective insight from all decision trees within the Random Forest model.

3.2 Fitting all the models

We loop through every stock doing the following: - Extracting the train and validation sets, and filter NAs - Perform feature selection for every stock - Fit an Elasticnet model for that stock, and obtain predictions for the returns - Compute the RMSE - Fit the Random Forest classification model and obtain the prediction for the direction lead - Compute the Sharpe Ratio and Modified Sharpe - Save everything

3.3 Identifying the best selected stocks

In the next phase, after fine-tuning and validating our models, the subsequent action is employing them to ascertain the stocks with the highest potential for returns. The procedure `f_select_top_stocks` is devised to scrutinize through the modeling phase outcomes, identifying stocks that not only are predicted to render the highest returns but also secure a favorable Sharpe ratio.

Balancing Sharpe Ratio and Forecasted Returns for Stock Selection The function `f_select_top_stocks` intuitively amalgamates the two metrics to identify and prioritize the most promising stocks for investment:

1. **Sharpe Ratio**: The Sharpe ratio is a measure of risk-adjusted return, formulated as:

$$\text{Sharpe Ratio} = \frac{\text{Expected Return of selected stocks} - \text{Risk-Free Rate}}{\text{Standard Deviation of selected stocks' Return}}$$

2. **Forecasted Returns**: These are the returns predicted by our preceding model, steering our selection toward stocks projected to maximize potential gains.

Now that all the models have been trained and the metrics recorded, we now simply choose the top stocks based on: - positive forecasted return and upward forecasted return direction - historical sharpe ratio greater than 0.3 and upward forecasted return direction - historical sharpe higher than 0.5.

The selected stocks and their data are then saved as a list containing the following: - The first level of the list as the stock tickers - The second level as the xts of each stock containing the realized returns, best shifted Arima, actual volatility and the volatility forecast.

```
## Example

#  head(best_stocks_data$ELV, 3)
#            realized_returns best_shifted_arima     volat vol_forecast month_index
# 2016-10-05     -0.010044382        -0.02765337 0.1784826    0.2822681          10
# 2016-10-12      0.024359314         0.05894012 0.1693443    0.3747814          10
# 2016-10-19      0.002742052         0.11664366 0.1653891    0.3719133          10
```

3.4. Portfolio optimization

Once the best stocks have been selected via the modelling procedure, they are then used as input for the portfolio optimization algorithm.

3.4.1 Assumptions for portfolio optimization The portfolio strategy is based on the following assumptions: - No short sales are allowed - No leverage - Fractional investment is possible - No transaction costs

3.4.2 Portfolio optimization procedure The following steps are performed for portfolio optimization:

1. Compute the covariance matrix using the 1-month volatility forecasted via GARCH
2. Perform the portfolio optimization satisfying the following:

- The minimum desirable return is the mean of the current month's forecasted returns computed via ARIMA
- No short selling is allowed
- Minimum weight of 5% allocated to each stock to ensure diversification

3. The optimization technique provides the optimal weight for each stock in the portfolio.

```r
# Perform min-variance portfolio optimization
output <- tryCatch({
  quadprog::solve.QP(Dmat = cov_matrix, # covariance matrix to be minimized
                     dvec = mean_returns, # minimum returns
                     Amat = cbind(matrix(rep(1,length(mean_returns)), ncol = 1),
                                  diag(length(mean_returns))),
                     bvec = c(1, rep(min_alloc,length(mean_returns))),
                     # sum of weights = 1 & no short selling
                     meq = 1)
}, error = function(e){
  print("(f_optimize_portfolio) -->error in portfolio optimization,
        default equally weighted portfolio returned.")
  return(NULL)
})

# if error in  optimization, return equally weighted porf
if (is.null(output)){
  return(rep(1/length(mean_returns), length(mean_returns)))
}

# Weights allocated
weights_mv <- output$solution
names(weights_mv) <- names(mean_returns)

# return optimized weights
return(weights_mv)
```

4. Backtesting procedure

4.1 Time window and number of runs

Assuming we have N_{years} of weekly data, giving a total of N_{months}, we want to fix a window of $N_W = 12$ months at the time (i.e. a year of data). Thus, the total number of runs is given by

$$N^{runs} = \left\lfloor \frac{N_{months} - N_W}{s} \right\rfloor + 1$$

where $s = 1$ is the number of months to move at the time (because of monthly re-balance).

i.e., we can move N^{runs} times when predicting one month at the time, starting with having all the data until month 12.

That is, $\tau = 1, \ldots, 48$

4.2 Backtesting procedure

Backtesting is performed for the strategy on a rolling window basis. The procedure can be summarized as a repeated process of the following steps:

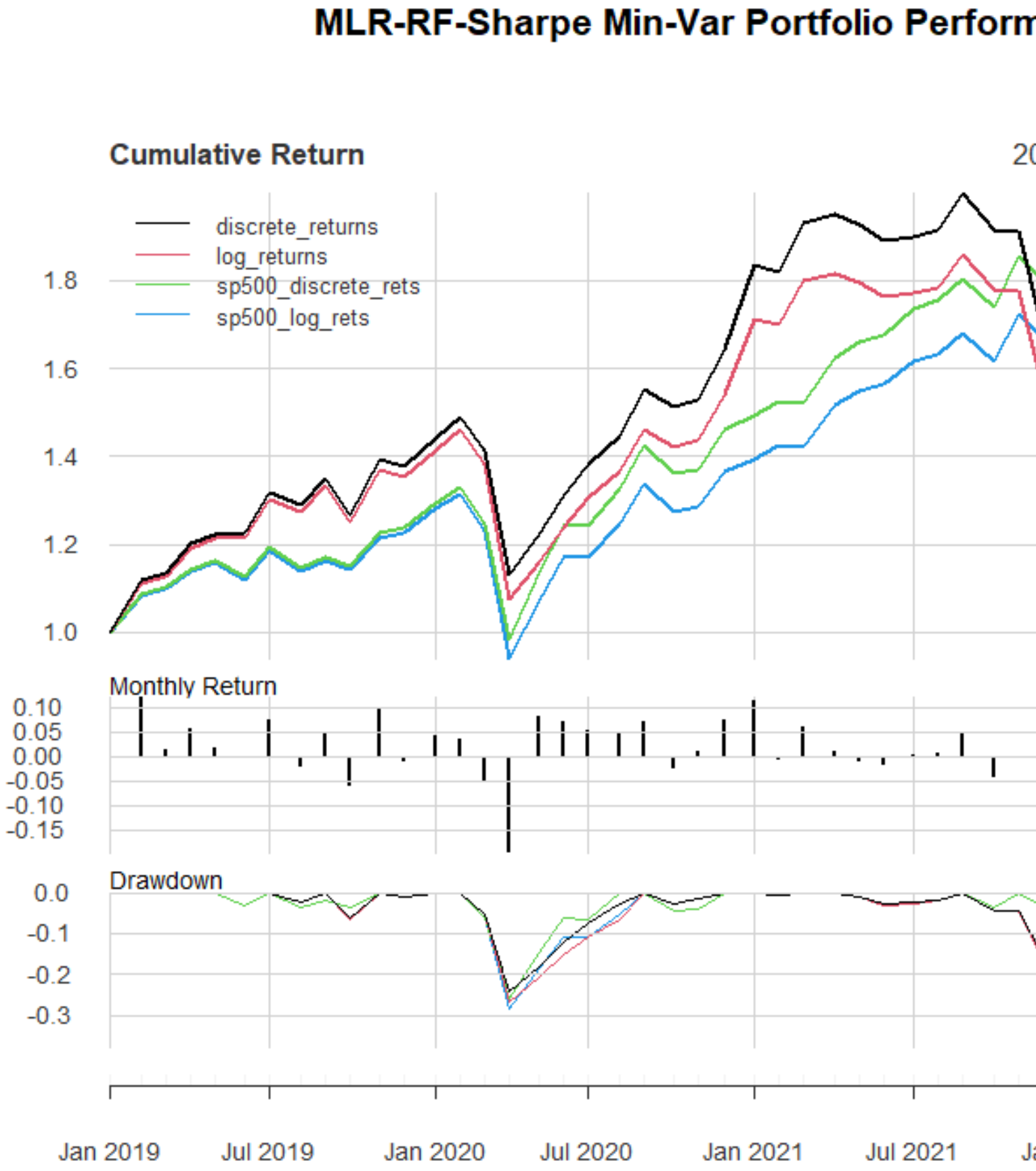
1. Long the portfolio as per the optimized weights and keep the position for a month.

2. After a month, close all the positions by shorting the portfolio. The profit/loss generated forms part of the capital and is reinvested.
3. Perform the modelling procedure on the new dataset generated for the next time step to get a new selection of stocks. Again perform the portfolio optimization technique to obtain the optimal weights.
4. Long this new portfolio

This process is repeated monthly for all the time steps and the results obtained after each run are noted and presented.

5. Performance Evaluation

The following graphs represent the performance of our portfolio over the period.


MLR-RF-Sharpe Min-Var Portfolio Perform
Cumulative Return
discrete_returns
log_returns
sp500_discrete_rets
sp500_log_rets
1.8
1.6
1.4
1.2
1.0
Monthly Return
0.10
0.05
0.00
-0.05
-0.10
-0.15
Drawdown
0.0
-0.1
-0.2
-0.3
Jan 2019
Jul 2019
Jan 2020
Jul 2020
Jan 2021
Jul 2021

The backtest spans over a 5-year period, using a rolling 24-month window to forecast the metrics for the subsequent month. Our data, however, consists of weekly Wednesday adjusted close. We benchmarked the performance against the S&P 500 index. As evident from the graph above, the portfolio has consistently outperformed the index on a monthly basis. The portfolio, however, also displays significant drawdowns indicating high volatility and large declines from peaks. Towards the latter part of the backtest, portfolio's performance appears to align closely with that of the S&P 500 before eventually intersecting with it. This pattern suggests an asymmetric response of the portfolio to market dynamics. Specifically, the portfolio doesn't demonstrate equal sensitivity to market uptrends as it does to downtrends.