Online Quantitative Trading Strategies

An Empirical Performance Overview

Glucksman Fellowship

Niko Lahanis[1], Alice Liu[1], Zhengyuan Zhou[2], and Anna Theresa Kuchler[2]

[1]Researchers and Contributors, niko_lahanis@stern.nyu.edu, alice.liu@stern.nyu.edu
[2]Professors and Mentors, zzhou@stern.nyu.edu, tkuchler@stern.nyu.edu

Contents

Abstract

This paper systematically evaluates the empirical performance of online quantitative trading strategies through a standardized Python-based trading framework. We aim to answer a central question: ***which online portfolio selection methods consistently deliver optimal risk-adjusted returns under realistic market conditions?*** *We investigate four primary families of algorithms—including momentum-based Follow-the-Winner, mean-reversion-based Follow-the-Loser, pattern-matching techniques, and ensemble meta-learning strategies—benchmarking them against traditional strategies such as Buy-and-Hold, Best Stock, and Constantly Rebalanced Portfolios (CRP). Rigorous hyperparameter tuning via grid search optimizes each method to maximize key performance metrics, specifically cumulative wealth, exponential growth rate, and the Sharpe ratio. Our findings highlight that adaptive strategies with regularization (e.g., Follow-the-Regularized-Leader) and confidence-based mean-reversion methods (e.g., CWMR, PAMR) provide superior risk-adjusted returns. However, these and other more advanced strategies often incur substantial computational costs, which may limit their practicality for intraday trading. We conclude by discussing practical implications and outlining directions for future work, including enhanced computational efficiency, cloud deployment, and broader asset class applications.*

1. Introduction

Online portfolio selection is a quantitative trading technique that has emerged as a powerful approach to asset allocation. By continuously updating portfolio allocations based on new market data, this adaptive method captures trends and exploits market inefficiencies in real time. Inspired by the original work [18] titled "Online Portfolio Selection – A Survey," this paper implements and compares a wide array of online portfolio selection algorithms discussed in that survey. Our objective is to evaluate not only their risk-adjusted performance—measured through metrics such as cumulative wealth, exponential growth, and the Sharpe ratio—but also their computational efficiency in realistic trading scenarios.

To achieve this, we developed a unified Python-based trading infrastructure that standardizes data preprocessing, portfolio management, and performance evaluation across multiple algorithmic families. These include momentum-based methods (Follow-the-Winner), mean-reversion strategies (Follow-the-Loser), pattern-matching techniques, and meta-learning ensemble approaches. By systematically tuning hyperparameters and benchmarking these strategies against traditional methods like Buy-and-Hold, Best Stock, and Constant Rebalancing, our study bridges theoretical insights with practical applications in quantitative trading.

More concisely, this paper addresses a critical research gap: ***determining which algorithms reliably offer superior risk-adjusted returns in realistic trading environments.*** For access to the complete GitHub repository and installation instructions, please visit the following link: Public Repo [20].

2. Input Data

For this research, the input data is derived from QuantQuote Minute Market Data [22], a comprehensive intra-day stock database. This dataset includes minute-by-minute trading data for securities listed on NASDAQ, NYSE, and AMEX from 1998 to the present, along with select ETFs traded on NYSE ARCA. The data spans regular trading hours from 9:30 AM to 4:00 PM, with optional coverage for pre-market and after-market trading sessions. Key features of the dataset include:

- **Survivorship Bias-Free Indexes:** The dataset maintains survivorship bias-free lists of major indexes.
- **Data Adjustments:** Prices and volumes are automatically adjusted for stock splits and dividends, ensuring continuity in the data.
- **Earnings Data:** Earnings announcements are recorded with their timing, including indicators for pre-market and post-market releases, allowing differentiation of trading behavior during these periods.
- **Dividend Adjustments:** Dividends are factored into historical prices to reflect price continuity, with corresponding values provided in the dataset.
- **Symbol Tracking:** Changes in stock symbols due to mergers, acquisitions, or other corporate actions are tracked automatically. Symbol reuses are managed to avoid ambiguity.

The data for this study was sourced from minute-level historical stock data organized into subdirectories containing individual CSV files for each stock. Each file, named in the format `table_TICKER.csv`, includes key columns such as Date, Time, Open, High, Low, Close, Volume, and adjustments for splits, dividends, and earnings. In our experiments, the breadth of securities was limited to the NASDAQ 100, and the time frame was set from 1998 to 2010.

3. Trading Environment Architecture

To support the exploration of online portfolio selection techniques, we developed a robust Python-based trading infrastructure. This framework integrates data preprocessing, portfolio management, and performance evaluation, enabling efficient implementation and testing of various online portfolio selection strategies. In the following sections, we detail its core components and functionalities.

3.1. Data Collection and Portfolio Initialization

The infrastructure begins with retrieving stock price data stored in structured CSV files. Using the `get_all_tickers()` function, the system scans subdirectories for files named `table_<TICKER>.csv`, extracting unique ticker symbols. Each file contains historical data for individual stocks, which are standardized with uniform column names. Data is further processed to compute price-relative vectors (x_t), where $x_{t,i}$ represents the ratio of the closing price of stock i at time t to its closing price at $t-1$. This step ensures compatibility across datasets and prepares the data for subsequent portfolio calculations.

The portfolio is initialized as a uniform distribution across the available assets using the `initialize_portfolio()` function, where the allocation to each stock is $b_{t,i} = \frac{1}{m}$. Price-relative vectors for multiple assets are aligned side-by-side in a consolidated DataFrame using the `calculate_price_relative_vectors()` function, allowing for the computation of multi-asset portfolio returns.

For each ticker, the closing prices are extracted, and price-relative vectors are calculated by dividing each day's closing price by the previous day's. This normalization removes the influence of the absolute price level of a stock, enabling a direct comparison of performance across different securities and ensuring compatibility for portfolio optimization and algorithmic trading strategies. The

Date	AAPL	ADBE	ADI	ADP	ADSK	AEP	ALGN	AMAT	AMD
2020-03-20	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
2020-03-23	1.001272	1.064626	1.013781	0.990994	0.990994	0.998042	0.955933	1.087877	1.070791
2020-03-24	1.077576	0.989776	1.111111	1.131817	1.038761	1.010376	1.200000	1.093676	1.087204
2020-03-25	1.006179	0.987088	0.996277	1.002895	0.993306	1.017210	1.143315	1.010011	0.993243
2020-03-26	1.045977	1.043492	0.989002	1.115085	1.076266	1.114477	1.010635	1.089866	1.043450

Figure 1. Price Relative Vectors

data is cleaned by handling missing values and adjusting non-positive values to small positive numbers to avoid computational issues. The final dataset is structured as a single Python DataFrame, with rows representing daily price relatives and columns for each ticker, as shown in Figure 1.

3.2. Performance Metrics and Evaluation

The infrastructure includes utilities for calculating key performance metrics such as:

- **Cumulative Wealth [5]:** The `calculate_cumulative_wealth()` function computes the wealth trajectory over a sequence of trading periods based on the portfolio allocation and price-relative vectors. The initial cumulative wealth value is set to 1.0.
- **Exponential Growth Rate [14]:** Defined as $W_n = \frac{1}{n} \log\left(\frac{S_n}{S_0}\right)$, this metric evaluates the long-term growth potential of a strategy.
- **Sharpe Ratio [23]:** The `compute_sharpe_ratio()` function calculates risk-adjusted performance by annualizing the excess returns over a risk-free rate (assumed to be a constant 5%).

These metrics ensure a balanced assessment of both profitability and risk, directly aligning with industry standards. The infrastructure is designed to evaluate various online portfolio selection strategies. Parameter tuning for each algorithm is automated via a grid search over predefined hyperparameter values (e.g., β and δ), with performance primarily ranked by the Sharpe ratio.

3.3. Scalability and Practicality

To ensure scalability, the framework leverages efficient vectorized operations provided by `NumPy` and `pandas`. Its modular design allows seamless integration of additional strategies, datasets, and performance metrics. This makes the infrastructure a powerful tool for exploring state-of-the-art methods in online portfolio selection.

ALGORITHM 1: Online portfolio selection framework.

Input: $\mathbf{x}_1^n$: Historical market sequence
Output: S_n: Final cumulative wealth
Initialize $S_0 = 1, \mathbf{b}_1 = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$
for $t = 1, 2, \ldots, n$ **do**
  Portfolio manager computes a portfolio $\mathbf{b}_t$;
  Market reveals the market price relative $\mathbf{x}_t$;
  Portfolio incurs period return $\mathbf{b}_t^\top \mathbf{x}_t$ and updates cumulative return $S_t = S_{t-1} \times \left(\mathbf{b}_t^\top \mathbf{x}_t\right)$;
  Portfolio manager updates his/her online portfolio selection rules ;
end

Figure 2. Environment Architecture and Algorithm Flow

Figure 2 summarizes the entire trading environment developed in Python, illustrating the process across each trading strategy. This framework was inspired by the survey assessing trading strategies in [18].

4. Algorithms Tested

Below is an overview of the trading strategies systematically compared in this study, as discussed in [18].

Classifications	Algorithms	Representative References
Benchmarks	Buy And Hold	
	Best Stock	
	Constant Rebalanced Portfolios	Kelly [1956]; Cover [1991]
Follow-the-Winner	Universal Portfolios	Cover [1991]; Cover and Ordentlich [1996]
	Exponential Gradient	Helmbold et al. [1996, 1998]
	Follow the Leader	Gaivoronski and Stella [2000]
	Follow the Regularized Leader	Agarwal et al. [2006]
	Aggregating-type Algorithms	Vovk and Watkins [1998]
Follow-the-Loser	Anti Correlation	Borodin et al. [2003, 2004]
	Passive Aggressive Mean Reversion	Li et al. [2012]
	Confidence Weighted Mean Reversion	Li et al. [2011b, 2013]
	Online Moving Average Reversion	Li and Hoi [2012]
	Robust Median Reversion	Huang et al. [2013]
Pattern-Matching Approaches	Nonparametric Histogram Log-optimal Strategy	Györfi et al. [2006]
	Nonparametric Kernel-based Log-optimal Strategy	Györfi et al. [2006]
	Nonparametric Nearest Neighbor Log-optimal Strategy	Györfi et al. [2008]
	Correlation-driven Nonparametric Learning Strategy	Li et al. [2011a]
	Nonparametric Kernel-based Semi-log-optimal Strategy	Györfi et al. [2007]
	Nonparametric Kernel-based Markowitz-type Strategy	Ottucsák and Vajda [2007]
	Nonparametric Kernel-based GV-type Strategy	Györfi and Vajda [2008]
Meta-Learning Algorithms	Aggregating Algorithm	Vovk [1990][1998]
	Fast Universalization Algorithm	Akcoglu et al. [2002, 2004]
	Online Gradient Updates	Das and Banerjee [2011]
	Online Newton Updates	Das and Banerjee [2011]
	Follow the Leading History	Hazan and Seshadhri [2009]

Figure 3. Algorithms Tested and Their Original References

4.1. Benchmarks

To evaluate the performance of online portfolio selection algorithms, we compare them against three benchmark strategies commonly used in financial literature. These benchmarks serve as simple, interpretable reference points that help contextualize the efficacy of more sophisticated portfolio selection methods.

1. **Buy-and-Hold (BAH)**
   The Buy-and-Hold strategy is one of the simplest investment approaches. Under this strategy, an initial allocation of wealth is distributed across assets at the beginning of the trading period, and no rebalancing occurs thereafter. The portfolio holdings evolve passively based on the relative price changes of the assets. Mathematically, the portfolio weight at time t is given by:

$$b_t = \frac{b_{t-1} \odot x_{t-1}}{\sum_{i=1}^{N} (b_{t-1,i} x_{t-1,i})}. \tag{1}$$

2. **Best Stock Strategy**
   The Best Stock strategy assumes perfect hindsight by investing all wealth in the single best-performing stock over the period. This strategy provides an upper bound on the performance that any portfolio selection method can achieve based solely on past price movements. Formally, at each time step:

$$b_t = e_{\arg\max x_{t-1}}, \tag{2}$$

   where e_i is the one-hot vector representing full allocation in the best-performing asset.

3. **Constant Rebalanced Portfolio (CRP)**
   The CRP strategy maintains a fixed proportion of wealth in each asset by rebalancing at every time step. Given an initial portfolio weight vector b, the strategy enforces:

$$b_t = b, \quad \forall t. \tag{3}$$

   This approach ensures stability and leverages volatility pumping, where the portfolio benefits from price fluctuations. The optimal CRP in hindsight, referred to as the Best Constant Rebalanced Portfolio (BCRP), is computed as:

$$b^* = \arg\max_{b \in \Delta_N} \sum_{t=1}^{T} \log(b \cdot x_t), \tag{4}$$

   where Δ_N denotes the probability simplex over N assets.

4.2. Follow-the-Winner

The Follow-the-Winner (FTW) approaches allocate more wealth to assets that have historically performed well. These strategies often attempt to track or improve upon the BCRP by dynamically adjusting portfolio allocations.

1. **Universal Portfolios (Approximation)**
   Cover's Universal Portfolio [5] is a theoretically optimal FTW strategy that continually updates portfolio weights by averaging across a continuum of constant rebalanced portfolios, weighted by their past performance. Due to computational constraints, we approximate this strategy using a Monte Carlo method by sampling M random portfolios and updating their weights based on historical returns:

$$b_t = \frac{\sum_{i=1}^{M} w_{t-1}^{(i)} p_{t-1}^{(i)}}{\sum_{i=1}^{M} w_{t-1}^{(i)}}, \tag{5}$$

   where $p_t^{(i)}$ represents the i^{th} sampled portfolio and $w_t^{(i)}$ is its corresponding wealth.

2. **Exponential Gradient (EG)**
   The Exponential Gradient (EG) algorithm, proposed by Helmbold et al. [12], updates portfolio weights using a multiplicative adjustment based on past returns:

$$b_{t+1,i} = b_{t,i} \cdot \exp\left(\eta \left(\frac{x_{t,i}}{b_t \cdot x_t} - 1\right)\right), \tag{6}$$

   where η is the learning rate. The weights are normalized to maintain a valid portfolio.

3. **Follow-The-Leader (FTL)**
   The FTL strategy selects the portfolio that would have been optimal in hindsight, based on cumulative past log returns:

$$b_t = \arg\max_b \sum_{\tau=1}^{t-1} \log(b \cdot x_\tau). \tag{7}$$

   While this method reallocates capital to the historically best-performing assets, it can be vulnerable to market instability due to overfitting on short-term trends [8].

4. **Follow-the-Regularized-Leader (FTRL)**
   A variation of FTL, FTRL introduces a regularization term to mitigate overreaction to short-term fluctuations:

$$b_t = \arg\max_b \left\{ \sum_{\tau=1}^{t-1} \log(b \cdot x_\tau) - \frac{\beta}{2} \|b\|^2 \right\}, \tag{8}$$

   where β controls the strength of regularization. This formulation, inspired by the Online Newton Step (ONS) [1], improves the stability of portfolio updates.

5. **Aggregation-Based Strategy**
   The Aggregation-Based method [27] combines multiple base portfolios by weighting them according to their historical performance:

$$b_t = \sum_{i=1}^{N} w_t^{(i)} p^{(i)}, \tag{9}$$

   with the weights updated using an exponential rule:

$$w_t^{(i)} = \frac{w_{t-1}^{(i)} \cdot \left(p^{(i)} \cdot x_{t-1}\right)^\eta}{\sum_{j=1}^{N} w_{t-1}^{(j)} \cdot \left(p^{(j)} \cdot x_{t-1}\right)^\eta}. \tag{10}$$

   This approach provides adaptive allocation while maintaining diversification.

4.3. Follow-the-Loser

In contrast to Follow-the-Winner approaches, Follow-the-Loser strategies are based on the principle of mean reversion—that is, assets that underperform tend to revert to their historical average performance. These methods aim to exploit market inefficiencies caused by overreaction to short-term trends [4]. The main strategies implemented in this research are:

1. **Anti-Correlation (Anticor)**
   Introduced by Borodin et al. [4], the Anticor strategy exploits statistical mean reversion by transferring wealth from assets with high recent returns to those with lower returns. It does so by computing cross-correlation matrices over a rolling window of past price relatives:

$$Mcov(i,j) = \frac{1}{w-1} \sum_{t=1}^{w} \left(y_{t,i} - \bar{y}_i\right)\left(y_{t,j} - \bar{y}_j\right), \tag{11}$$

   where $y_{t,i}$ represents the log price relative of asset i, and w is the look-back window size.

2. **Passive Aggressive Mean Reversion (PAMR)**
   Proposed by Li et al. [17], PAMR formalizes mean reversion in an online learning framework. It penalizes portfolios when the predicted return exceeds a predefined threshold ϵ:

$$\ell_\epsilon(b_t; x_t) = \max\{0, b_t \cdot x_t - \epsilon\}. \tag{12}$$

   The next portfolio is computed by solving a constrained optimization problem that minimizes deviation from the previous allocation while enforcing the mean reversion principle.

3. **Confidence Weighted Mean Reversion (CWMR)**
   CWMR [16] extends PAMR by incorporating confidence-weighted learning. It models the portfolio as a Gaussian distribution with mean μ_t and covariance Σ_t, allowing the strategy to adjust the confidence of weight estimates dynamically:

$$(\mu_{t+1}, \Sigma_{t+1}) = \arg\min_{\mu,\Sigma} D_{KL}\left(N(\mu,\Sigma) \,\|\, N(\mu_t,\Sigma_t)\right), \tag{13}$$

   where D_{KL} denotes the Kullback-Leibler divergence.

4. **Online Moving Average Reversion (OLMAR)**
   OLMAR [17] improves upon single-period mean reversion strategies by using a moving average of past price relatives as a forecast:

$$\hat{x}_{t+1} = \frac{1}{w} \sum_{i=1}^{w} x_{t-i}. \tag{14}$$

   This method helps smooth out short-term noise, resulting in a more robust rebalancing strategy.

5. **Robust Median Reversion (RMR)**
   The RMR strategy [13] further enhances mean reversion by replacing the simple moving average with an ℓ_1 median estimator:

$$\hat{x}_{t+1} = \arg\min_{\mu} \sum_{i=1}^{w} \|x_{t-i} - \mu\|. \tag{15}$$

   This approach reduces the influence of outliers, leading to a more stable allocation.

4.4. Pattern Matching

Pattern-matching approaches leverage historical market behavior to identify and apply trading strategies that were effective under similar conditions in the past. These nonparametric techniques select historical data samples based on similarity measures and then optimize portfolio allocation accordingly. Unlike the Follow-the-Winner and Follow-the-Loser strategies, pattern matching makes no explicit assumptions about market structure and relies solely on data-driven selection criteria.

Following the classification in [18], we implement four sample selection methods and three portfolio optimization techniques:

1. **Sample Selection Methods**

(a) **Histogram-Based Selection** [9]: Historical price-relative vectors are segmented into predefined bins, and past periods with similar mean behavior (falling into the same bin as the most recent market window) are selected into the candidate set C_t.

(b) **Kernel-Based Selection** [9]: A smooth similarity function based on Euclidean distance is applied between the most recent price-relative window and all past windows. Historical windows that fall within a predefined threshold are selected.

(c) **Nearest Neighbor Selection** [15]: Historical windows are ranked by their distance from the latest price-relative window, and the k nearest neighbors are selected.

(d) **Correlation-Based Selection** [3]: The Pearson correlation coefficient between the most recent window and each historical window is computed, and those with correlation above a threshold ρ are included in C_t.

2. **Portfolio Optimization Methods**

(a) **Log-Optimal Portfolio** [9]: The optimal portfolio b^* is determined by maximizing the expected logarithmic return over the selected sample set C_t:

$$b^* = \arg\max_b \sum_{i \in C_t} \log(b \cdot x_i), \tag{16}$$

subject to $b \in \Delta_N$.

(b) **Semi-Log-Optimal Portfolio** [10]: This method uses a smoothed objective function $f_z(z) = z - 0.5(z-1)^2$ in place of $\log(z)$ to reduce sensitivity to extreme values:

$$b^* = \arg\max_b \sum_{i \in C_t} f_z(b \cdot x_i). \tag{17}$$

(c) **Markowitz Portfolio** [21]: Inspired by classical mean-variance optimization, this method balances expected return and risk using:

$$b^* = \arg\max_b \left[\mu_C \cdot b - \lambda\, b^T \Sigma_C\, b\right], \tag{18}$$

where μ_C is the mean of the selected price-relative vectors, Σ_C is their covariance matrix, and λ controls the risk-return tradeoff.

3. **Pattern-Matching Portfolio Framework**

(a) Select a candidate set C_t from historical data using one of the sample selection methods.

(b) Compute the optimal portfolio weights using one of the portfolio optimization methods.

(c) Rebalance the portfolio to b_t.

The table below summarizes the combinations of sample selection and portfolio optimization strategies that were tested:

Sample Selection Method	**Portfolio Optimization Method**
Histogram-Based Selection	Log-Optimal Portfolio
Histogram-Based Selection	Semi-Log-Optimal Portfolio
Histogram-Based Selection	Markowitz Portfolio
Kernel-Based Selection	Log-Optimal Portfolio
Kernel-Based Selection	Semi-Log-Optimal Portfolio
Kernel-Based Selection	Markowitz Portfolio
Nearest Neighbor Selection	Log-Optimal Portfolio
Nearest Neighbor Selection	Semi-Log-Optimal Portfolio
Nearest Neighbor Selection	Markowitz Portfolio
Correlation-Based Selection	Log-Optimal Portfolio
Correlation-Based Selection	Semi-Log-Optimal Portfolio
Correlation-Based Selection	Markowitz Portfolio

Table 1. Pattern-Matching Strategy Combinations

By leveraging past market patterns, these nonparametric strategies provide an adaptive approach to portfolio selection. Our empirical results compare the performance of these methods against other online trading algorithms.

4.5. Meta-Learning

Meta-learning algorithms in online portfolio selection enhance performance by dynamically combining multiple base strategies. Rather than relying on a single predefined approach, these methods form an ensemble of algorithms, adjusting allocations based on past performance to optimize returns. The goal is to achieve adaptability in non-stationary market conditions while maintaining theoretical performance guarantees.

The primary meta-learning strategies implemented in this research include Aggregation Algorithms, Fast Universalization, Online Gradient Updates, Online Newton Updates, and Follow-the-Leading-History. These methods build upon earlier work on ensemble learning and regret minimization in online convex optimization [26, 2, 6, 11].

1. **Aggregation Algorithm (AA)**
   The Aggregation Algorithm weights individual base experts using an exponentially weighted scheme, as introduced by Vovk [26, 27]. At each time step, the expert weights are updated according to:

$$w_t^{(i)} = \frac{w_{t-1}^{(i)} \exp(-\eta \ell_t^{(i)})}{\sum_j w_{t-1}^{(j)} \exp(-\eta \ell_t^{(j)})}, \tag{19}$$

   where $\ell_t^{(i)}$ is the loss for expert i at time t and η is the learning rate. The overall portfolio allocation is then computed as:

$$b_t = \sum_i w_t^{(i)} b_t^{(i)}. \tag{20}$$

2. **Fast Universalization (FU)**
   Fast Universalization accelerates convergence by efficiently combining a subset of base strategies [2]. The method initializes with uniform weights over a set of pre-selected algorithms (e.g., CWMR, FTRL, and PAMR) and dynamically reweights them based on their cumulative performance:

$$b_t = \sum_i w_t^{(i)} b_t^{(i)}, \quad \text{where } w_t^{(i)} = \frac{w_{t-1}^{(i)} S_{t-1}^{(i)}}{\sum_j w_{t-1}^{(j)} S_{t-1}^{(j)}}. \tag{21}$$

3. **Online Gradient Updates (OGU)**
   Inspired by exponential gradient methods in online convex optimization [6], Online Gradient Updates adjust the portfolio allocation via gradient descent updates over expert predictions:

$$b_{t+1,i} = b_{t,i} \cdot \exp\left(\eta \left(\frac{x_{t,i}}{b_t \cdot x_t} - 1\right)\right). \tag{22}$$

   The weights are normalized to ensure a valid portfolio.

4. **Online Newton Updates (ONU)**
   ONU incorporates second-order gradient information to adaptively adjust the learning rate [6]. The portfolio update follows the Newton step:

$$b_t = b_{t-1} - \eta A^{-1} \nabla \ell_t, \tag{23}$$

   where A is the Hessian matrix of historical losses.

5. **Follow-the-Leading-History (FTLH)**
   The FTLH algorithm [11] maintains a set of historical experts, updating their portfolios using the Online Newton Step (ONS). At each time step, a new expert is introduced, and the Weighted Majority Algorithm (WMA) is applied to combine past expert performances:

$$w_t^{(i)} = \frac{w_{t-1}^{(i)} e^{-\eta \ell_t^{(i)}}}{\sum_j w_{t-1}^{(j)} e^{-\eta \ell_t^{(j)}}}. \tag{24}$$

   Experts with poor historical performance (below a threshold) are pruned, ensuring the system focuses on the more effective strategies.

Meta-learning approaches enable dynamic portfolio adaptation by combining multiple experts into a robust framework. In our experiments, ensemble methods combining high-performing base strategies (CWMR, FTRL, PAMR) using gradient or Newton-based updates demonstrate enhanced risk-adjusted performance.

5. Experiments and Results

5.1. Training and Hyperparameter Tuning

Before officially comparing performances, we conducted hyperparameter tuning using a grid search. During initial experiments, we discovered that the performance of some algorithms was highly sensitive to their parameter settings. Although default parameters are convenient, they may not be optimal for our historical market data, potentially leading to suboptimal performance. By systematically exploring a range of candidate values with a grid search, we ensured that each algorithm was finely tuned to maximize key performance metrics such as the Sharpe ratio, enabling a fair and robust comparison across all methods.

Using the Anti-Correlation (Anticor) algorithm as an example, we defined a grid of candidate values for its key hyperparameters, including the rolling window size for computing correlations, the α parameter governing the degree of adjustment, and the correlation threshold that determines when to trigger wealth transfers between assets. As shown in Figure 4, this grid search resulted in eight unique hyperparameter configurations.

```
anticor_grid = {
    'window_size': [3, 5],
    'alpha': [1.5, 2.0],
    'corr_threshold': [0.4, 0.5]
}
```

Figure 4. Hyperparameter Grid Search Example – 8 Combinations

For each combination, the algorithm was executed on historical price-relative vectors, and its performance was evaluated using metrics such as cumulative wealth, exponential growth rate, and the Sharpe ratio (see Section 3.2). The hyperparameter set yielding the highest Sharpe ratio was selected as optimal.

Window Size	Alpha	Corr. Threshold	Final Wealth	Exp. Growth	Sharpe Ratio
3	2.0	0.5	36.88	0.00195042	1.01752
3	1.5	0.5	36.88	0.00115903	1.01751
3	2.0	0.4	35.57	0.00118301	1.00748
3	1.5	0.4	35.59	0.00116523	1.00777
5	2.0	0.5	28.65	0.00111968	0.93937
5	1.5	0.5	28.65	0.00110729	0.93727
5	2.0	0.4	26.50	0.00099422	0.89344
5	1.5	0.4	28.50	0.00110062	0.93440

Table 2. Anticor – Hyperparameter Tuning Results

This systematic tuning procedure was applied across all algorithms in the study, ensuring that each method was optimized under comparable conditions. To expedite the grid search, we employed the joblib library [25] for parallelization, significantly reducing computation time. For a comprehensive list of the optimal hyperparameters for each algorithm, please refer to Exhibit A in the Appendix.

5.2. Output

5.2.1 Follow-the-Winner

Figure 5 presents the cumulative wealth (on a logarithmic scale) for the Follow-the-Winner strategies over the sample period, compared against benchmark strategies (BAH, Best Stock, and CRP). The main observations are as follows:

- **Follow-the-Regularized Leader (FTRL)** stands out with a final wealth of 52.6363 and the highest Sharpe ratio of 1.1370, indicating superior risk-adjusted performance. Although it lags in the early years, its robust final performance suggests that incorporating regularization into the allocation process significantly improves stability and returns.
- **Constant Rebalancing (CRP)**, **Exponential Gradient (EG)**, and **Follow the Leader (FTL)** exhibit nearly identical performance, with final wealth around 12.1584 and Sharpe ratios of approximately 0.7553. **Universal Portfolios** deliver solid results with a final wealth of 11.7161 and a Sharpe ratio of 0.7515, albeit at a longer runtime (0.0768 seconds).
- **Aggregation-Based Simple** underperforms relative to the other dynamic strategies, achieving a final wealth of 10.0203 and a Sharpe ratio of 0.6608, which suggests that its simplistic aggregation mechanism may not fully capture market dynamics.

Table 3 summarizes the performance metrics for each strategy, ordered by their Sharpe ratio. The results underscore the effectiveness of regularization in online portfolio selection, as evidenced by the performance of Follow-the-Regularized Leader.

In summary, meta-strategies that adaptively adjust weights based on recent performance—particularly those employing regularization (e.g., FTRL)—tend to deliver significantly improved risk-adjusted returns compared to both traditional benchmarks and less adaptive methods.

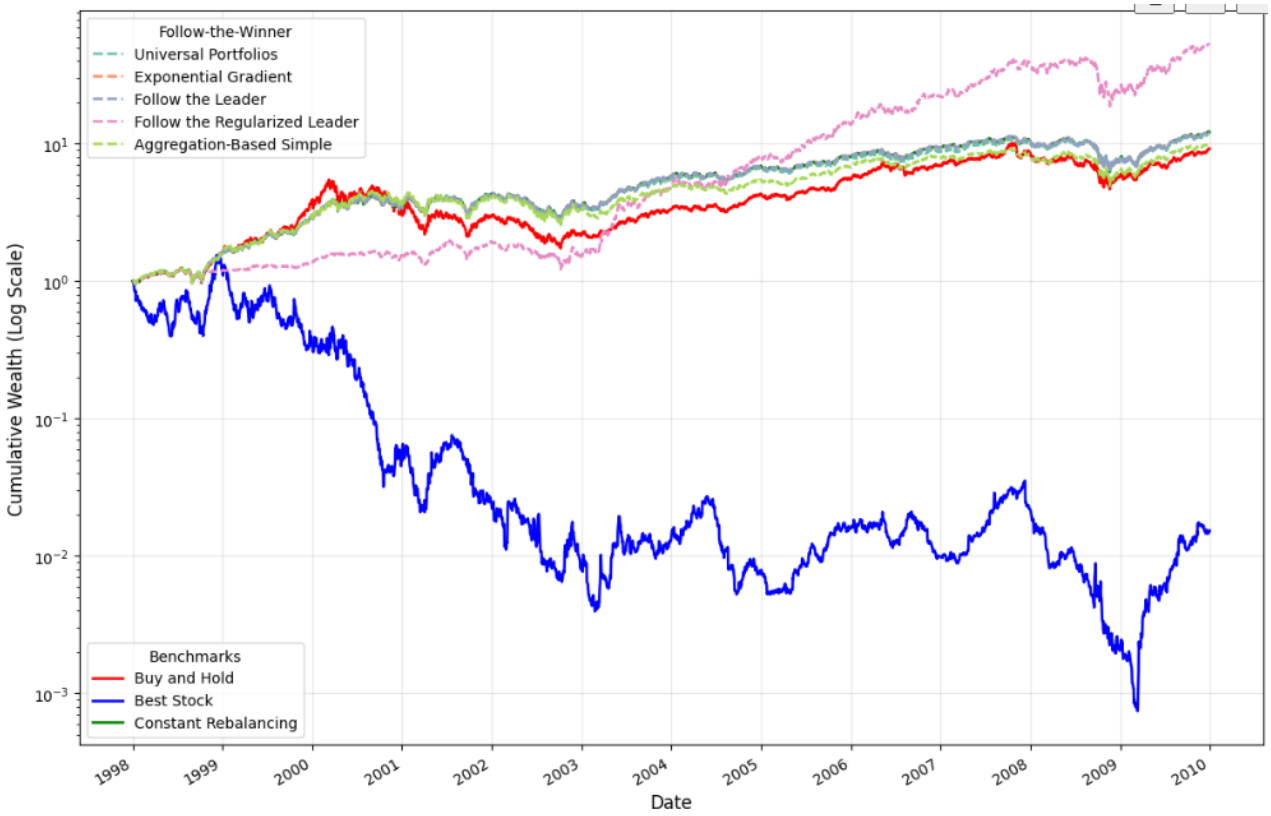

Figure 5. Cumulative Wealth of Follow-the-Winner Algorithms vs. Benchmarks (Log Scale)

Strategy	Final Wealth	Exponential Growth	Sharpe Ratio	Time (sec)
Follow the Regularized Leader	52.6363	0.0013	1.1370	0.6616
Constant Rebalancing	12.1584	0.0008	0.7553	0.0058
Exponential Gradient	12.1584	0.0008	0.7553	0.0499
Follow the Leader	12.1584	0.0008	0.7553	0.0575
Universal Portfolios	11.7161	0.0008	0.7515	0.0768
Aggregation-Based Simple	10.0203	0.0008	0.6608	0.0257
Buy and Hold	9.1742	0.0007	0.5797	0.0159
Best Stock	0.0153	-0.0014	0.0905	0.0091

Table 3. Performance Summary – Follow-the-Winner Strategies

5.2.2 Follow-the-Loser

Figure 6 displays the cumulative wealth (log scale) for the Follow-the-Loser strategies over the sample period, compared against benchmark strategies (BAH, Best Stock, and CRP). Key observations include:

- **Confidence Weighted Mean Reversion (CWMR)** delivers the highest final wealth (1349.6859) and the top Sharpe ratio (1.7710). Its strong performance in later stages indicates that dynamically adjusting portfolio weights based on confidence intervals effectively exploits mean-reverting behavior.
- **Passive Aggressive Mean Reversion (PAMR)** follows closely, achieving a final wealth of 788.2571 and a Sharpe ratio of 1.6295. Its penalization mechanism for excessive returns allows it to capture substantial mean-reversion gains.
- **Anti-Correlation** outperforms simpler benchmarks with a final wealth of 36.8860 and a Sharpe ratio of 1.0175. Its approach of reallocating capital among negatively correlated assets helps capture short-term reversals.
- **Robust Median Reversion (RMR)** and **Online Moving Average Reversion (OLMAR)** exhibit moderate performance, with Sharpe ratios of 0.9249 and 0.8389, respectively.

Table 4 summarizes the performance metrics for each Follow-the-Loser strategy, ordered by Sharpe ratio. The results indicate that advanced mean-reversion methods—especially those incorporating adaptive or probabilistic elements (CWMR, PAMR)—can substantially outperform traditional benchmarks.

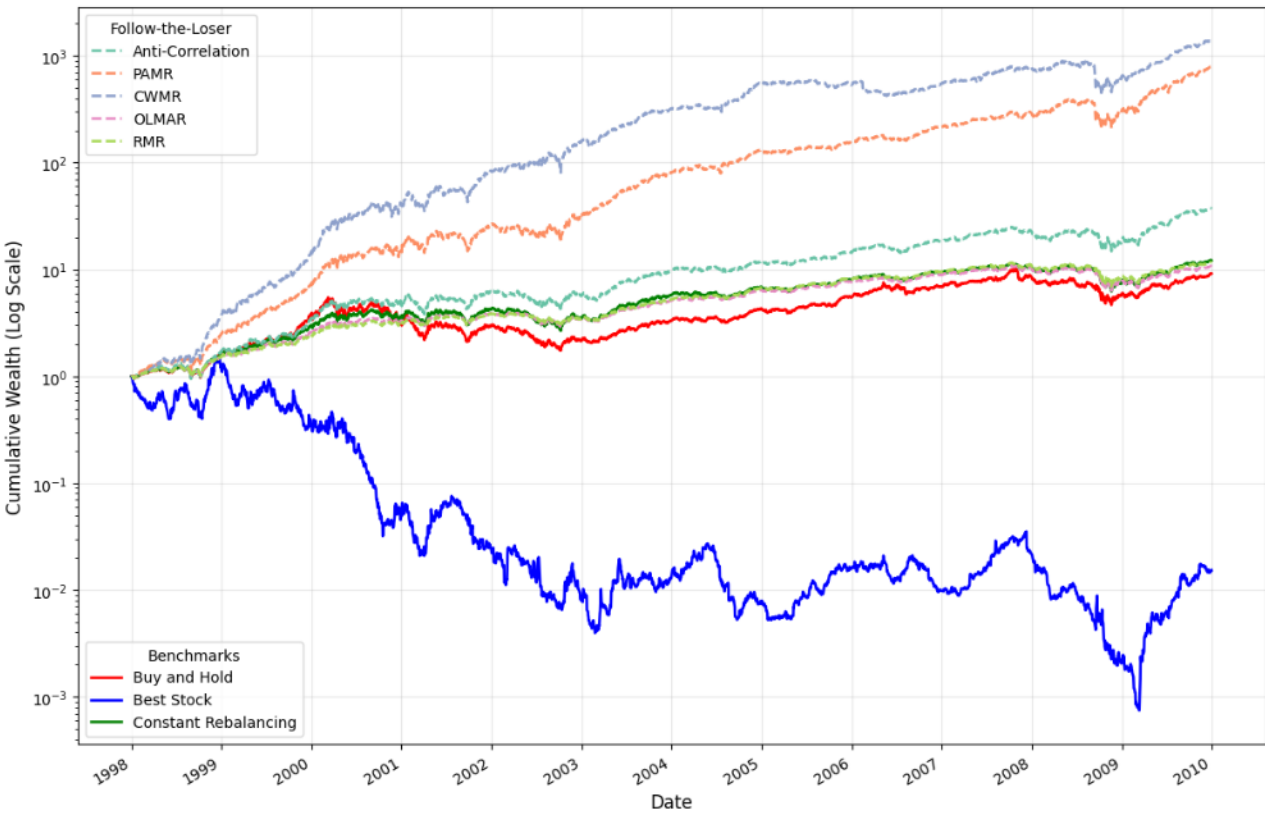

Figure 6. Cumulative Wealth of Follow-the-Loser Strategies vs. Benchmarks (Log Scale)

Strategy	Final Wealth	Exp. Growth	Sharpe Ratio	Time (sec)
CWMR	1349.6859	0.0024	1.7710	1.0463
PAMR	788.2571	0.0022	1.6295	0.0581
Anti-Correlation	36.8860	0.0012	1.0175	0.6792
RMR	11.6045	0.0008	0.9249	0.6320
OLMAR	10.6686	0.0008	0.8389	0.0553
Constant Rebalancing	12.1584	0.0008	0.7553	0.0077
Buy and Hold	9.1742	0.0007	0.5797	0.0198
Best Stock	0.0153	-0.0014	0.0905	0.0099

Table 4. Performance Summary – Follow-the-Loser Strategies (Ordered by Sharpe Ratio)

5.2.3 Pattern Matching

Figure 7 illustrates the cumulative wealth (log scale) of the pattern-matching strategies over the sample period, compared against benchmark strategies (BAH, Best Stock, and CRP). The key findings are:

- **Histogram-Based Selection** combined with any of the three portfolio optimization methods (Log-Optimal, Semi-Log-Optimal, Markowitz) significantly outperforms the other pattern-matching approaches, achieving final wealth levels between 500 and 610 with Sharpe ratios above 1.14. This indicates that discretizing historical returns into bins and focusing on similar market conditions can be very effective, albeit at high computational cost.
- **Kernel-Based Selection** produces moderate returns, with final wealth near 13.4 and Sharpe ratios of approximately 0.7510, comparable to the performance of CRP.
- **Nearest Neighbor Selection** leads to relatively low final wealth (around 1.4–1.8) and Sharpe ratios near 0.40, suggesting that focusing solely on the closest historical windows may be too restrictive or prone to overfitting.
- **Correlation-Based Selection** exhibits the weakest performance among pattern-matching approaches, with final wealth around 0.12–0.17 and Sharpe ratios between 0.09 and 0.12.

Table 5 summarizes the final performance metrics for each pattern-matching strategy, ordered by Sharpe ratio. Notably, the combination of Histogram-Based Selection and Markowitz Portfolio optimization achieves the highest Sharpe ratio of 1.1642, underscoring the potential benefits of pairing an effective sample selection method with mean-variance optimization. However, this method does incur a significantly higher computational cost.

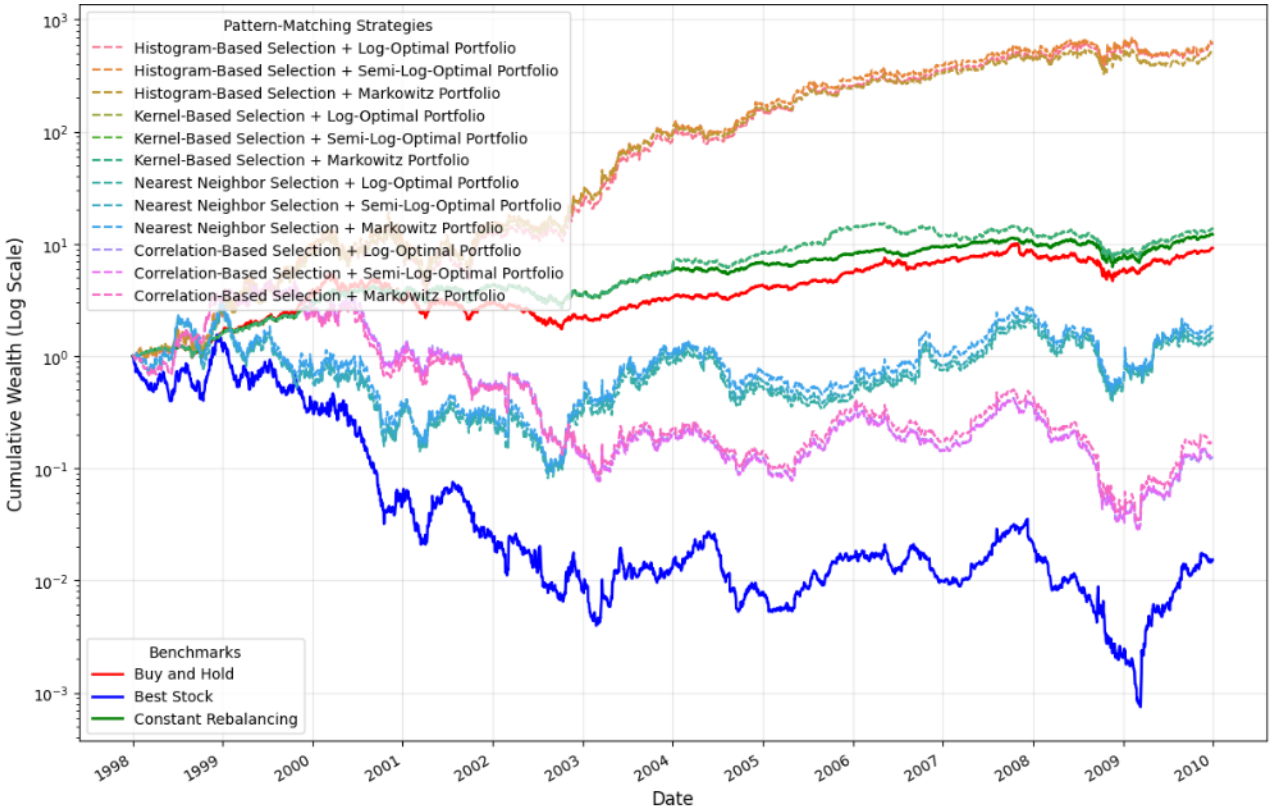

Figure 7. Cumulative Wealth of Pattern-Matching Strategies vs. Benchmarks (Log Scale)

Strategy	Final Wealth	Exp. Growth	Sharpe Ratio	Time (sec)
Histogram-Based Selection + Markowitz Portfolio	501.1874	0.0021	1.1642	772.2721
Histogram-Based Selection + Semi-Log-Optimal Portfolio	610.4945	0.0021	1.1584	1233.4941
Histogram-Based Selection + Log-Optimal Portfolio	598.7398	0.0021	1.1492	1120.7117
Constant Rebalancing	12.1584	0.0008	0.7553	0.0098
Kernel-Based Selection + Markowitz Portfolio	13.4051	0.0009	0.7510	151.5789
Kernel-Based Selection + Semi-Log-Optimal Portfolio	13.4052	0.0009	0.7510	150.5620
Kernel-Based Selection + Log-Optimal Portfolio	13.4052	0.0009	0.7510	129.4660
Buy and Hold	9.1742	0.0007	0.5797	0.0164
Nearest Neighbor Selection + Markowitz Portfolio	1.8390	0.0002	0.4064	550.3752
Nearest Neighbor Selection + Semi-Log-Optimal Portfolio	1.6012	0.0002	0.4053	521.6035
Nearest Neighbor Selection + Log-Optimal Portfolio	1.4205	0.0001	0.3948	492.5187
Correlation-Based Selection + Markowitz Portfolio	0.1681	-0.0006	0.1211	473.1760
Correlation-Based Selection + Semi-Log-Optimal Portfolio	0.1282	-0.0007	0.0954	834.6027
Correlation-Based Selection + Log-Optimal Portfolio	0.1225	-0.0007	0.0914	845.7227
Best Stock	0.0153	-0.0014	0.0905	0.0156

Table 5. Performance Summary – Pattern-Matching Strategies (Ordered by Sharpe Ratio)

5.2.4 Meta-Learning

Figure 8 shows the cumulative wealth (log scale) of the meta-learning strategies over the sample period, compared with benchmarks (BAH, Best Stock, and CRP). In our experiments, we tested various combinations of base experts, eventually finding that an ensemble combining **CWMR**, **FTRL**, and **PAMR** yielded consistently superior performance. The key findings are:

- **Online Newton Update (ONU)** achieves the highest final wealth of 387.4096 and the best Sharpe ratio of 1.5562, indicating that incorporating second-order (Hessian-based) information into the update process can lead to strong risk-adjusted returns.
- **Online Gradient Update (OGU)** and **Fast Universalization** also show robust performance, with final wealth around 102 and 101, respectively, and Sharpe ratios of 1.3648 and 1.3490. These results illustrate that simpler first-order updates or more computationally efficient portfolio averaging schemes can still yield substantial gains over traditional benchmarks.
- **Aggregation-Based Generalized** closely matches Constant Rebalancing (CRP) in both final wealth (approximately 12.16) and Sharpe ratio (0.7552 vs. 0.7553). Although this strategy benefits from aggregating many experts, it does not adapt as

aggressively as gradient-based methods.

- **Follow the Leading History (FTLH)** lags behind the top-tier meta-strategies, finishing with a final wealth of 12.5197 and a Sharpe ratio of 0.6134, suggesting that overreacting to short-term trends may hurt performance in this dataset.

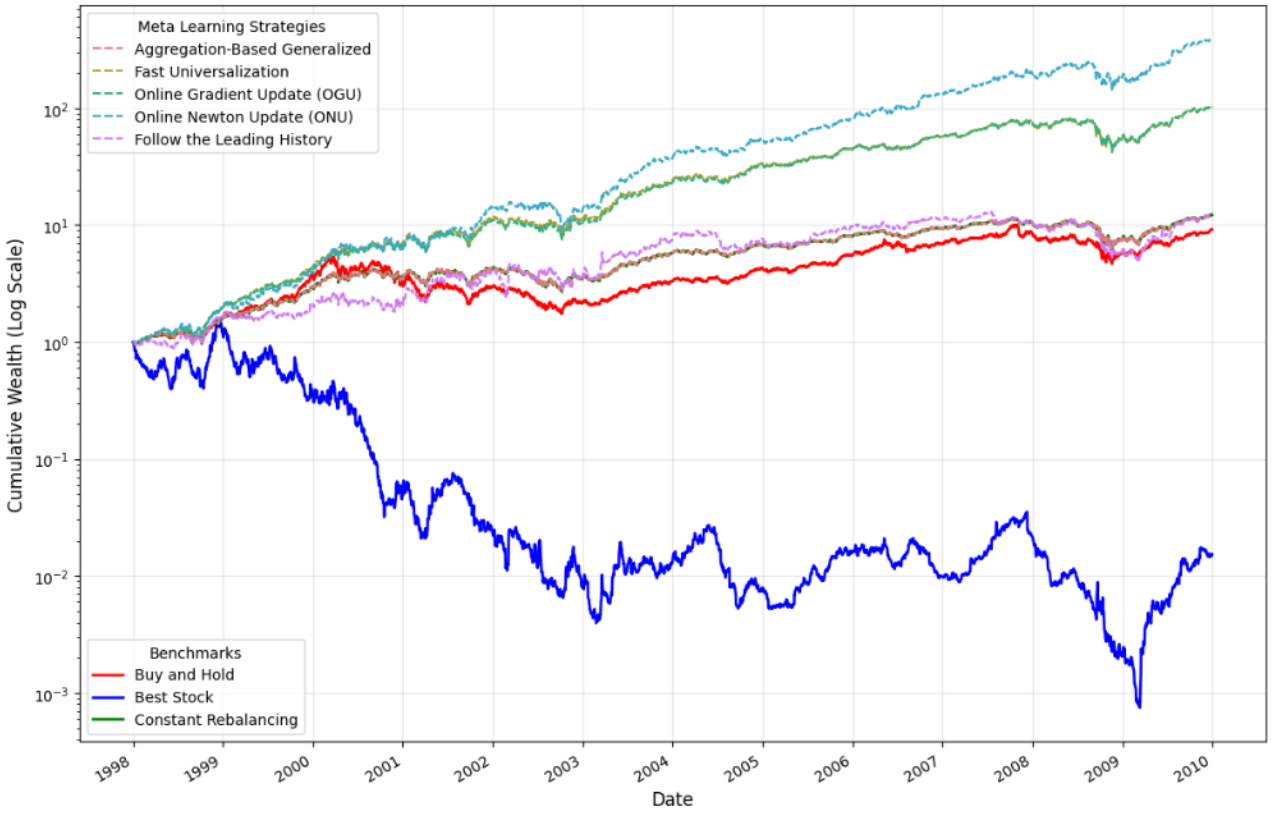

Figure 8. Cumulative Wealth of Meta-Learning Strategies vs. Benchmarks (Log Scale)

Table 6 summarizes the performance metrics for the meta-learning strategies, sorted by Sharpe ratio. Notably, the best-performing approaches (ONU, OGU, and Fast Universalization) all rely on adaptively shifting capital among high-performing base portfolios (CWMR, FTRL, PAMR), demonstrating the effectiveness of ensemble methods in capturing diverse market behaviors. It is worth mentioning that although these ensemble methods exhibit attractive Sharpe ratios, they are computationally intensive compared to some of the more individual algorithms.

Strategy	Final Wealth	Exp. Growth	Sharpe Ratio	Time (sec)
Online Newton Update (ONU)	387.4096	0.0020	1.5562	2483.3489
Online Gradient Update (OGU)	102.6580	0.0015	1.3648	2360.4510
Fast Universalization	101.7371	0.0015	1.3490	3805.3808
Constant Rebalancing (CRP)	12.1584	0.0008	0.7553	0.0038
Aggregation-Based Generalized	12.1570	0.0008	0.7552	0.0434
Follow the Leading History (FTLH)	12.5197	0.0008	0.6134	1.8361
Buy and Hold	9.1742	0.0007	0.5797	0.0173
Best Stock	0.0153	-0.0014	0.0905	0.0084

Table 6. Performance Summary – Meta-Learning Strategies (Ordered by Sharpe Ratio)

5.3. Performance Summary

Across the algorithm groups, our results reveal clear trends regarding performance and computational efficiency:

- **Follow-the-Winner:** The adaptive method *Follow-the-Regularized Leader (FTRL)* consistently delivers higher risk-adjusted returns, underscoring the benefits of incorporating regularization into the rebalancing process.
- **Follow-the-Loser:** Mean-reversion strategies that incorporate adaptive or confidence-based adjustments—especially *CWMR* and *PAMR*—achieve outstanding final wealth and Sharpe ratios by effectively exploiting mean-reversion opportunities.

- **Pattern Matching:** Histogram-Based Selection methods significantly outperform other pattern-matching variants in terms of returns and risk-adjusted performance, albeit at a substantially higher computational cost.
- **Meta-Learning:** Ensemble methods that combine high-performing base portfolios (CWMR, FTRL, PAMR) using gradient- or Newton-based updates deliver strong risk-adjusted performance. However, these approaches are considerably more computationally intensive than many individual algorithms.

5.4. Comparison with Existing Literature

Our empirical results generally align with the established literature on online portfolio selection and quantitative trading. In particular, mean-reversion strategies—such as Confidence Weighted Mean Reversion (CWMR) and Passive-Aggressive Mean Reversion (PAMR)—consistently achieve high Sharpe ratios and cumulative wealth in mean-reverting market regimes, confirming earlier findings [7]. Likewise, momentum-based approaches like Follow-the-Regularized-Leader (FTRL) demonstrate robust performance when appropriate regularization is applied, in agreement with both academic studies and industry insights [19]. Although pattern-matching methods yield exceptional risk-adjusted returns, their high computational cost has been documented in previous work [24].

Our investigation into meta-learning ensembles—where base strategies such as CWMR, PAMR, and FTRL are combined using online gradient or Newton updates—indicates that adaptive weighting can enhance performance by capturing both momentum and mean-reversion effects. While these ensemble methods require considerable computational resources, their ability to dynamically adjust to changing market conditions confirms and extends prior work on expert aggregation techniques [27, 11]. Notably, this work integrates these diverse approaches into a unified framework that bridges theoretical performance with real-world financial applications.

6. Experience

The following sections provide a brief overview of the qualitative and technical challenges encountered during this study.

6.1. Challenges

- **Limited Computing Resources:** Reliance on local computing resources posed significant challenges for processing large datasets and performing computationally intensive tasks.
- **Data Storage:** Efficient storage and rapid access to extensive historical datasets were challenging due to their size and the frequency of access required.

6.2. Approach Changes

- **Parallelization:** We adopted the joblib library [25] to parallelize hyperparameter tuning, dramatically reducing computation time.
- **Modularization:** Code was modularized to separate data preprocessing, algorithm implementations, and performance evaluations, greatly improving maintainability and flexibility.
- **Vectorization:** We leveraged `NumPy` and `pandas` for efficient, vectorized operations in data manipulation and calculations.

6.3. Successes

- **Repeatable Infrastructure:** We established a robust, repeatable infrastructure that enables consistent and systematic evaluations of multiple portfolio strategies.
- **Extensive Hyperparameter Tuning:** A comprehensive grid search ensured that each algorithm was optimized under comparable conditions, leading to reliable performance comparisons.
- **Uniform Comparisons:** By standardizing evaluation metrics and experimental protocols, the study provides a fair and uniform comparison of diverse online portfolio selection methods.

7. Conclusion and Future Work

This study introduced a comprehensive Python-based infrastructure designed to evaluate a diverse range of online quantitative trading strategies. Through systematic implementation and extensive hyperparameter optimization, we benchmarked four main families of algorithms—including momentum-driven Follow-the-Winner, mean-reversion-based Follow-the-Loser, pattern-matching techniques, and meta-learning ensembles—against conventional benchmarks like Buy-and-Hold and Constant Rebalancing.

Our experiments, conducted on daily historical stock data, demonstrated that adaptive approaches (particularly those incorporating regularization, such as Follow-the-Regularized Leader, or confidence-based adjustments, such as CWMR and PAMR) consistently achieve superior risk-adjusted performance. Additionally, ensemble methods that combine multiple strong base strategies via gradient- or Newton-based updates emerged as robust solutions for dynamically capturing diverse market behaviors.

However, the study also uncovered a practical trade-off: the substantial computational overhead of pattern-matching and meta-learning strategies may limit their application in high-frequency or intraday trading. The modular infrastructure developed in this work offers a repeatable approach to online portfolio selection and effectively bridges theoretical insights with practical financial applications.

Future Work – The challenges and results presented in this study provide several promising directions for further research:

- **Computational Efficiency:** Leverage GPU acceleration, distributed computing, and more efficient algorithmic techniques to significantly reduce runtime, especially for computationally intensive strategies.
- **Cloud Deployment:** Transition the framework to a cloud-native environment using containerization, CI/CD pipelines, and restful API frameworks to enable scalable, resilient, and collaborative system development.
- **Additional Algorithms and Metrics:** Expand the framework to include advanced techniques—such as deep reinforcement learning and evolutionary algorithms—and incorporate additional performance metrics (e.g., maximum drawdown, Conditional Value at Risk).
- **Broader Market Data and Asset Classes:** Evaluate strategies across alternative asset classes (cryptocurrencies, commodities, ESG-focused portfolios) and different time scales (intraday, weekly, monthly) to improve generalization and robustness.
- **Real-Time Integration:** Integrate the framework with real-time data feeds and simulate live trading conditions by accounting for transaction costs, order-book dynamics, latency, and slippage.
- **Explainability and Interpretability:** Enhance transparency and user trust by incorporating interpretability tools (such as SHAP, LIME, or attention mechanisms) to clarify the factors influencing algorithmic decisions.

Overall, our findings contribute to a deeper understanding of online portfolio selection while emphasizing the promise of adaptive, ensemble-based approaches in quantitative trading. The proposed future work also provides a roadmap for further refining these strategies and bridging the gap between theoretical performance and real-world applicability.

References

[1] Amit Agarwal, Elad Hazan, Satyen Kale, and Robert E. Schapire. Algorithms for portfolio management based on the newton method. *ICML*, 2006. 7

[2] M. Akcoglu et al. Fast universalization algorithm. *Journal of Financial Optimization*, 2002. 10

[3] Steven C. H. Hoi Bin Li and Peilin Zhao. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Intelligent Systems and Technology*, 3(2):1–28, 2011. 9

[4] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock? *Journal of Artificial Intelligence Research*, 18:221–287, 2003. 8

[5] Thomas M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991. 5, 7

[6] P. Das and A. Banerjee. Online gradient and newton updates for portfolio selection. *Journal of Machine Learning Research*, 2011. 10

[7] Ali Fereydooni and Others. Pattern-matching approaches in portfolio optimization: A comparative study. *Journal of Financial Data Science*, 5(1):34–56, 2023. 16

[8] Alexei Gaivoronski and Francesco Stella. Adaptive portfolio optimization. *Annals of Operations Research*, 99(1-4):165–187, 2000. 7

[9] László Györfi, Gábor Ottucsák, and Harro Walk. *Machine Learning for Financial Engineering*. World Scientific, 2006. 9

[10] László Györfi and Harro Walk. Empirical portfolio selection based on log-optimal strategy. *Mathematical Finance*, 17(3):493–514, 2007. 9

[11] E. Hazan and C. Seshadhri. Follow the leading history for online portfolio selection. *Mathematical Finance*, 2009. 10, 16

[12] David P. Helmbold, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998. 7

[13] Steven C. H. Hoi Jiangang Huang and Bin Li. Robust median reversion strategy for online portfolio selection. *ACM Transactions on Intelligent Systems and Technology*, 5(1), 2013. 8

[14] John L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4):917–926, 1956. 5

[15] György Ottucsák László Györfi and Harro Walk. Nearest neighbor based portfolio selection strategies. *Statistics and Decisions*, 26:145–157, 2008. 9

[16] Bin Li and Steven C.H. Hoi. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Intelligent Systems and Technology*, 4(4), 2013. 8

[17] Bin Li, Steven C.H. Hoi, and Doyen Sahoo. Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 77–92, 2012. 8

[18] Bin Li and Steven C. H. Hoi. Online portfolio selection: A survey. *ACM Computing Surveys*, 46(3):1–36, 2013. Available at: http://arxiv.org/abs/1212.2129v2. 4, 5, 8

[19] Tobias J. Moskowitz, Yongheng Ooi, and Lasse H. Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012. 16

[20] nglahani. Online quantitative trading strategies. https://github.com/nglahani/Online-Quantitative-Trading-Strategies, 2020. GitHub repository (accessed: 2025-02-06). 4

[21] György Ottucsák and István Vajda. An efficient implementation of markowitz portfolio selection. *Annals of Operations Research*, 151(1):181–192, 2007. 9

[22] Inc. QuantQuote. Quantquote historical market data, 2024. Accessed: January 28, 2025. 4

[23] William F. Sharpe. The sharpe ratio. *The Journal of Portfolio Management*, 21(1):49–58, 1994. 5

[24] Tim van Erven, Wouter Koolen, et al. *Online Learning: Theory, Algorithms, and Applications*. Cambridge University Press, 2020. 16

[25] Gaël Varoquaux. Joblib: a set of tools to provide lightweight pipelining in python. https://joblib.readthedocs.io/, 2013. 11, 16

[26] V. Vovk. Aggregating strategies. *Proceedings of COLT*, 1990. 10

[27] Vladimir Vovk and Chris Watkins. Universal portfolio selection. *Theoretical Computer Science*, 244(1-2):37–58, 1998. 7, 10, 16

8. Appendix

Table Exhibit A. Optimal Hyperparameters for All Algorithms

Category	Algorithm	Optimal Parameters
Follow-the-Loser		
	Anticorrelation (Anticor)	window_size = 3, α = 2.0, corr_threshold = 0.5
	PAMR	ϵ = 0.9, C = 10.0
	CWMR	ϵ = 0.91, θ = 0.96, η = 0.95
	OLMAR	window_size = 2, ϵ = 1.0, η = 25
	RMR	window_size = 7, ϵ = 1.0, η = 20
Follow-the-Winner		
	Universal Portfolios	num_portfolios = 3, τ = 0.4
	Exponential Gradient (EG)	learning_rate = 0.1, smoothing = 0.0
	Follow-the-Leader (FTL)	γ = 1.0, α = 2.0
	Follow-the-Regularized-Leader (FTRL)	β = 0.09, δ = 0.925, ridge_const = 0.0075
	Aggregation-Based Simple	learning_rate = 0.4, num_base_portfolios = 1
Meta-Learning		
	Aggregation Algorithm (Generalized)	learning_rate = 0.001, γ = 0.3
	Fast Universalization	learning_rate = 0.1, base_experts = {CWMR, FTRL, PAMR}
	Online Gradient Update Meta	learning_rate = 0.01, base_experts = {CWMR, FTRL, PAMR}
	Online Newton Update Meta	learning_rate = 0.01, base_experts = {CWMR, FTRL, PAMR}
	Follow-the-Leading-History (FTLH)	η = 0.4, learning_rate = 0.05, drop_threshold = 0.5
Pattern Matching		
	Histogram-Based Selection	w = 4, bins = {(0,0.5), (0.5,1), (1,1.5)}
	Kernel-Based Selection	w = 4, threshold = 0.1
	Nearest Neighbor Selection	w = 4, num_neighbors = 3
	Correlation-Based Selection	w = 4, ρ = 0.7
	Log-Optimal Portfolio	(No additional parameters)
	Semi-Log-Optimal Portfolio	(No additional parameters)
	Markowitz Portfolio	λ = 0.7
	Pattern-Matching Portfolio Master	w = 4, methods as defined