Proceedings of ICFTBA 2025 Symposium: Financial Framework's Role in Economics and Management of Human-Centered Development
DOI: 10.54254/2754-1169/2025.GL30493

Adaptive Long–Short Equity Strategies with Salience Theory and Hidden Markov Regimes

Xin Lin[1*], Qichen Huang[2], Zhuocheng Ni[3], Zhanrong Li[4], Yuer Zhang[5]

[1]*School of Business and Management, Shanghai International Studies University, Shanghai, China*
[2]*College of Arts & Sciences, University of Washington, Seattle, USA*
[3]*Faculty of Science Division I, Tokyo University of Science, Tokyo, Japan*
[4]*School of Economics, Shenzhen University, Shenzhen, China*
[5]*Jinan Xinhang Experimental Foreign Language School, Jinan, China*
**Corresponding Author. Email: 0211149066@alumni.shisu.edu.cn*

Abstract. This study bridges a gap in behavioral finance by integrating Salience Theory with Hidden Markov Models to develop adaptive long-short strategies. Existing salience-based approaches, while effective predictors of mispricing, remain static and vulnerable to shifting market regimes. The hybrid framework dynamically adjusts signals between momentum (in Bull states) and reversal (in Bear states), while applying a refined salience metric. The results demonstrate that this synthesis significantly improves performance stability and reduces drawdowns during volatile periods, but it does not consistently outperform the CRSP benchmark after accounting for transaction costs. The hybrid strategy provides a distinct risk-return profile, making it suitable for certain market conditions but not universally superior. The study confirms the cross-market applicability of Salience Theory and suggests a framework for future adaptive models combining behavioral insights with dynamic market timing.

Keywords: Salience Theory, Hidden Markov Models, Behavioral Finance, Adaptive Strategies, Market Regimes

1. Introduction

1.1. Introduction to Salience Theory (ST)

Highlight: ST systematically converts attention-driven mispricing into tradable signals, generating cross-sectional alpha.

The strategy leverages ST to quantify behavioral mispricing by measuring the distortion between salience-weighted returns and objective returns (ST value). It follows the approach of Cosemans et al. (2021) to construct a portfolio by going long on undervalued stocks while shorting bubbles.[1]

1.2. Introduction to Hidden Markov Models (HMM) Regime-Switching

Highlight: HMM introduce regime-awareness, switching alpha logic across states to mitigate drawdowns and improve return stability.

DOI: 10.54254/2754-1169/2025.GL30493

This study proposes an adaptive long–short equity framework that employs HMM to infer market regimes and select the appropriate alpha signal. This perspective follows the approach of Wang, Lin, and Mikhelson, who demonstrate that HMM can successfully time factor exposures by switching between strategies conditioned on the inferred market state. [2]

2. Specifications

2.1. Economic Intuition

Financial markets exhibit persistent behavioral biases and regime-dependent dynamics.

ST: Traditional finance assumes that investors are fully rational and process all information objectively. Behavioral finance, however, recognizes that attention is limited and subject to systematic biases. Investors tend to overweight salient features such as extreme returns or unusually high trading volumes, while neglecting less visible information [3].

ST formalizes this bias by assuming that investors distort the weights applied to past returns, placing excessive emphasis on salient outcomes. This distortion results in mispricing: stocks with saliently high past returns or volumes become overpriced, while those with saliently low features are undervalued [1].

The strategy implements this concept through the ST value, defined as the difference between salience-weighted returns and objective equal-weighted returns. A high ST value signals overvaluation, while a low ST value suggests undervaluation. Portfolios are constructed by taking long positions in low-ST stocks (undervalued) and short positions in high-ST stocks (overvalued).[4].

HMM: Financial markets often alternate between stable and turbulent periods. In stable conditions, herding and trend-following behavior strengthen return persistence, making momentum profitable. In turbulent conditions, heightened volatility reflects investor overreaction and liquidity stress, creating short-term reversals that can be exploited by contrarian trades. Empirical research shows that momentum crashes are concentrated in high-volatility episodes, while contrarian strategies tend to perform better in these environments [5,**?**].

The HMM framework captures these dynamics by treating regime states as latent variables inferred from market-level features, including turbulence, realized volatility, and aggregate returns. By classifying the market probabilistically into Bull (stable) or Bear (turbulent) regimes, the model allows the trading rule to switch adaptively between momentum and reversal signals. This regime-sensitive structure provides a mechanism to align trading logic with the underlying state of the market.

2.2. Performance Metrics

We evaluate strategies using five core metrics that jointly capture profitability, risk-adjusted performance, downside risk, consistency, and implementability: Annualized Return, Sharpe Ratio, Maximum Drawdown, Win Rate, and Annualized Turnover.

2.3. Universe

The investment universe consists of the largest 3,000 U.S. equities by market capitalization. Stocks priced below $5 are excluded to avoid noise and illiquidity. This set ensures sufficient liquidity, cross-sectional variation, and implementability at scale.

2.4. Data Requirements and Sources

Daily adjusted price and return data for U.S. equities are required for both strategies.

DOI: 10.54254/2754-1169/2025.GL30493

ST: Requires stock-level daily returns, market capitalization, and trading volume to compute salience weights and liquidity filters.

HMM: Requires aggregate market returns, realized volatility, and turbulence measures (computed using Mahalanobis distance of cross-sectional returns).

Data are sourced from the CRSP database, which provides survivorship-bias-free coverage of both active and delisted U.S. securities.

2.5. Date Range and Data Splits

The sample spans 2015–2024 at daily frequency, and the data are divided into two periods: the In-Sample (Train + Validation) period, which covers 2015–2022 and is used for model estimation and refinement, and the Out-of-Sample (Test) period, which spans 2023–2024 and is reserved strictly for forward-looking evaluation. This setup ensures that all reported test results are based on unseen data, consistent with standard practices in empirical asset pricing and systematic strategy evaluation.

2.6. Raw Alpha

2.6.1. ST Raw Alpha.

In month t, we first calculate the salience score of the return $r_{i,s}$ of the stock i on each trading day s as follows[1]:

$$\sigma(r_{i,s;t}, \bar{r}_{s;t}) = \frac{|r_{i,s;t} - \bar{r}_{s;t}|}{|r_{i,s;t}| + |\bar{r}_{s;t}| + \theta},$$

where $\bar{r}_{s;t}$ is the average return of all available stocks in the market on day s and $\theta = 0.1$ to avoid a zero denominator. Then we rank $\sigma(r_{i,s;t})$ from 1 (most salient) to S_t (least salient). S_t denotes the total number of trading days in month t. Next, we calculate the salience weights for the returns of stock i within the month:

$$\omega_{i,s;t} = \frac{\delta^{\text{rank}_{i,s;t}}}{\sum_{s'} \delta^{\text{rank}_{i,s';t}} \cdot \pi_{i,s';t}}, \quad \delta \in (0, 1],$$

where $\text{rank}_{i,s;t}$ and $\pi_{i,s';t}$ are the salience rank and objective probability of $r_{i,s;t}$, respectively, and $\pi_{i,s';t}$ is equal to the inverse of S_t. δ captures the degree to which salience distorts the decision weights. If $\delta = 1$, then $\omega_{i,s;t}$ equals 1, which means that the weight for every trading day is equal, and there is no distortion. If $\delta < 1$, the investor is a salient thinker who will overweight salient returns (for stocks with salient score σ whose rank is close to 1, they will have a higher weight than 1) and underweigh non-salient ones (for stocks with non-salient score σ whose rank is close to S_t, they will have lower weight than 1). Here, we assume $\delta = 0.7$ following Cosemans et al. (2021).[1] It is worth noticing that the $E[\omega_{i,s;t}]$ equals 1. Salient theory value for stock i in month t is defined as follows:

$$Alpha_{i,t} = \text{cov}(\omega_{i,s;t}, r_{i,s;t}) = \sum_{s}^{S_t} \pi_{i,s;t}\omega_{i,s;t}r_{i,s;t} - \sum_{s}^{S_t} \pi_{i,s;t}r_{i,s;t} = E^{ST}[r_{i,s;t}] - \bar{r}_{i,s;t},$$

where the second equality follows from $E[\omega_{i,s;t}] = 1$, and the last equality follows from $\pi_{i,s;t} = \frac{1}{S_t}$, where S_t is equal to the number of trading days in month t. The construction of Alpha starts from the covariance between salience weights and daily returns. Intuitively, if higher salience weights are assigned to days with higher daily returns, the covariance increases, indicating that the stock tends

DOI: 10.54254/2754-1169/2025.GL30493

to be overvalued. Conversely, if higher salience weights are placed on days with lower daily returns, the covariance decreases, implying that the stock is undervalued. The equation further demonstrates that ST is equal to the difference between subjective salience-weighted past returns and objective equal-weighted past returns. ST thus measures the distortion in return expectations caused by salient thinking.

2.6.2. HMM Raw Alpha.

The strategy's core mechanics are defined by a set of formulas that govern regime detection, turbulence measurement, and alpha signal generation.

The market regime at time t, denoted as Regime_t, is classified by a function f whose inputs are key market state indicators. This function is implemented by a HMM trained to probabilistically infer the latent market state, consistent with applications in regime-aware factor investing [2,**?**,**?**]. The inputs are the turbulence index T_t, the market return Return_t, and the market volatility Volatility_t, yielding the classification:

$$\text{Regime}_t = f(T_t, \text{Return}_t, \text{Volatility}_t).$$

The turbulence index T_t itself is calculated using the Mahalanobis distance, which requires a vector of asset returns. Here, r_t represents an $N \times 1$ vector of cross-sectional returns for the N assets in the universe at time t. This measure evaluates how much the current return vector $\mathbf{r_t}$ deviates from its recent historical distribution, characterized by the mean return vector $\bar{r}$ (calculated over the past 60 days) and the covariance matrix Σ, following the systemic risk approach of [8] and subsequent financial applications [9]:

$$T_t = (r_t - \bar{r})^\top \Sigma^{-1} (r_t - \bar{r}).$$

Once the regime is determined, the alpha signal for each asset i is constructed by calculating cumulative returns over specific lookback windows and applying a ranking logic. The notation $r_{i,[t-A:t]}$ represents the cumulative return for asset i from time $t - A$ to t.

– In stable regimes, the strategy employs a momentum signal, consistent with evidence that return persistence dominates in calm markets [6]. This is computed by ranking assets based on their cumulative return over a one-month window, specifically from $t - 20$ to t (i.e., 20 trading days).

– In turbulent regimes, the strategy switches to a mean-reversion signal, in line with findings that volatility shocks often induce short-term reversals [5]. This is computed by applying a reverse ranking to assets based on their cumulative return over a much shorter, one-week window from $t - 5$ to t (i.e., 5 trading days).

This adaptive logic is formalized as:

$$\text{Alpha}_{i,t} = \begin{cases} \text{Rank}_i\left(r_{i,[t-21:t]}\right), & \text{if the regime is stable,} \\ -\text{Rank}_i\left(r_{i,[t-5:t]}\right), & \text{if the regime is turbulent.} \end{cases}$$

The dynamic switching between these two signals based on the inferred regime $f(\cdot)$ is the fundamental adaptive mechanism of the strategy.

DOI: 10.54254/2754-1169/2025.GL30493

2.7. Operations

2.7.1. ST Portfolio.

The ST alpha signal is computed monthly for each stock in the U.S. Top 3000 universe by quantifying the distortion between salience-weighted and objective returns. Implementation follows two steps:

1. **Ranking.** At the end of each month, stocks are sorted by their ST values:

$$f^{\text{rank}} : \{ST_{i,t}\} \;\mapsto\; \text{rank}(ST_{i,t}).$$

2. **Portfolio Construction.** A dollar-neutral long–short portfolio is formed by taking equal-weighted long positions in the 10% of stocks with the lowest ST values (undervalued) and short positions in the 10% with the highest ST values (overvalued):

$$f^{\text{port}} : \text{rank}(ST_{i,t}) \;\mapsto\; w_{i,t}.$$

Positions are capped at 5% of capital per stock to avoid concentration. Rebalancing occurs monthly at the close.

2.7.2. HMM Portfolio.

The HMM strategy generates regime-dependent alpha by classifying the market state into Bull (stable) or Bear (turbulent). In stable periods, alpha is defined as one-month momentum; in turbulent periods, it switches to one-week reversal. Operations proceed in two steps:

1. **Ranking.** At each rebalance, stocks are ranked by their regime-specific alpha values:

$$f^{\text{rank}} : \{\alpha_{i,t}\} \;\mapsto\; \text{rank}(\alpha_{i,t}).$$

2. **Portfolio Construction.** A dollar-neutral portfolio is formed by taking equal-weighted long positions in the top 10% of ranked stocks and equal-weighted short positions in the bottom 10%:

$$f^{\text{port}} : \text{rank}(\alpha_{i,t}) \;\mapsto\; w_{i,t}.$$

Rebalancing cadence adapts to regime: monthly in stable states to capture persistent trends, and weekly in turbulent states to react to reversals. Position caps of 5% are enforced.

2.8. Constraints

Portfolio construction for both ST and HMM strategies is subject to constraints that ensure neutrality, diversification, and implementability:

2.8.1. Dollar Neutrality

$$\sum_{i=1}^{N} w_{i,t} = 0 \quad \forall t$$

2.8.2. Position Limits

$$|w_{i,t}| \le w_{\max} \quad \forall i, t$$

with $w_{\max} = 5\%$.

DOI: 10.54254/2754-1169/2025.GL30493

2.8.3. Gross Leverage

$$\sum_{i=1}^{N} |w_{i,t}| \leq L_{\max}$$

where $L_{\max} = 1$, corresponding to 100% long + 100% short exposure.

2.8.4. Turnover Constraint

$$\text{Turnover}_t = \sum_{i=1}^{N} |w_{i,t} - w_{i,t-1}|$$

Transaction costs are directly deducted from PnL in proportion to turnover, capturing both trading fees and market frictions.

2.9. Trade Execution

All trades are executed at closing prices on scheduled rebalance days, with portfolio weights held constant between rebalances. In the original and refined in-sample tests, transaction costs were excluded in order to evaluate the raw signal performance. This provides a clear view of the intrinsic strength and stability of the alpha without the influence of implementation frictions.

Turnover is measured as the sum of absolute changes in portfolio weights between consecutive rebalance dates. Although costs are not deducted in these in-sample results, turnover remains an important diagnostic, since high turnover implies greater exposure to slippage, market impact, and borrow expenses in live trading. For instance, short positions in less liquid stocks may carry higher financing costs, and rapid position changes can amplify execution risk.

Cumulative performance is computed by aggregating daily gross simple returns of the rebalanced portfolios. Evaluation metrics such as annualized return, Sharpe ratio, win rate, and maximum drawdown are calculated from these gross series, while turnover is reported separately to assess implementability. This separation allows for consistent comparison across model variants, with the understanding that the more realistic assessment of tradability comes from the out-of-sample results where explicit transaction costs are included.

3. Implementation

3.1. ST

3.1.1. Settings.

The ST strategy is implemented on the Top 3000 U.S. equities with monthly rebalancing. The training window covers 2015–2022, while the out-of-sample test runs from 2023–2024. Transaction costs are excluded at this stage to focus on the raw in-sample profitability of the alpha.

DOI: 10.54254/2754-1169/2025.GL30493

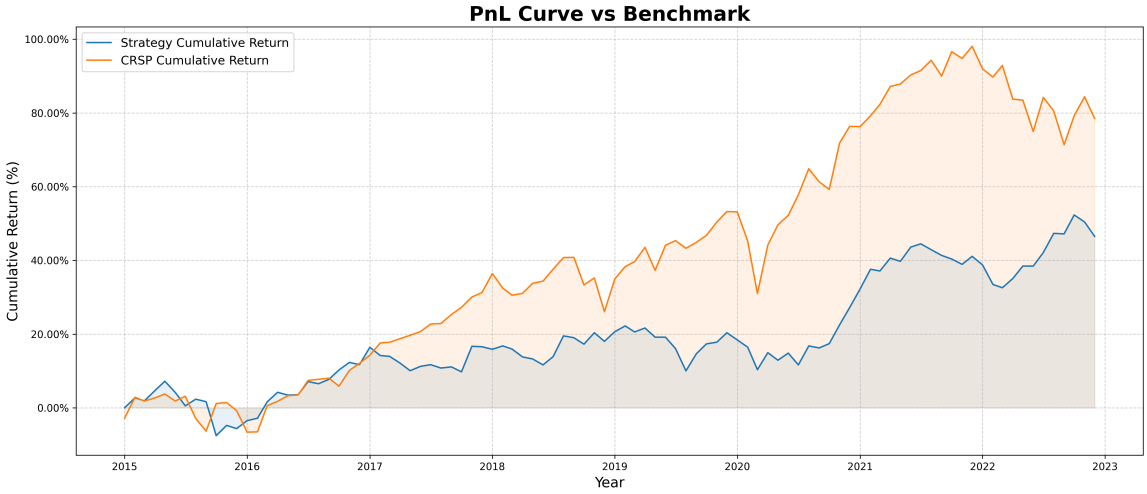

Figure 1: Cumulative Returns: ST Strategy vs. CRSP Benchmark (In-Sample)

3.1.2. Graphical Results.

Figures 1, 2, and 3 show cumulative performance of the ST portfolio relative to the CRSP benchmark, 1-year rolling Sharpe ratio, and turnover dynamics. The results show that the cumulative return trend of the strategy is similar to that of the benchmark, but the strategy's return is lower than the benchmark's. Although the strategy underperforms the benchmark in terms of returns, it demonstrates stronger risk resistance, particularly in 2020 and 2022.

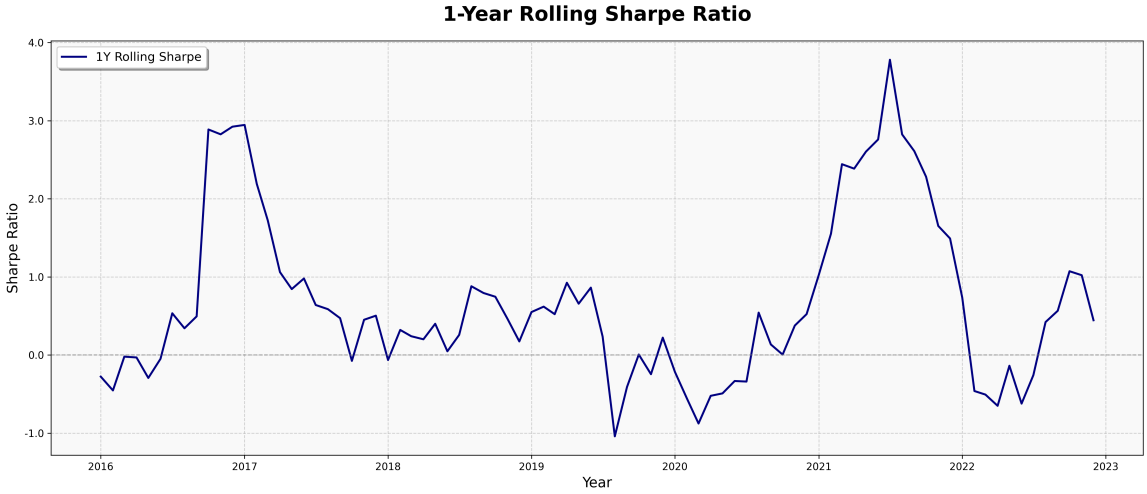

Figure 2: 1-Year Rolling Sharpe Ratio (In-Sample)

DOI: 10.54254/2754-1169/2025.GL30493

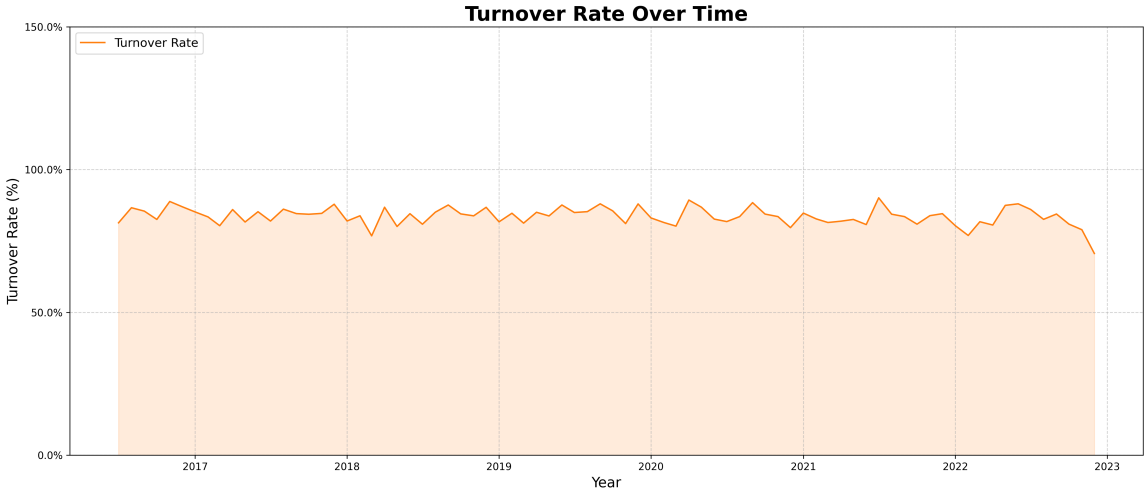

Figure 3: Turnover Rate Over Time (In-Sample)

3.1.3. Graph Analysis.

The 1-year rolling Sharpe ratio is generally below 1, except in 2016 and 2021 when it was relatively high, peaking above 3.0. Turnover typically fluctuates between 150% and 175%, primarily due to monthly rebalancing. The relatively long interval between rebalancing periods leads to significant changes in alpha signals, resulting in high turnover.

3.1.4. Performance Metrics.

Table 1 reports in-sample metrics for the ST strategy versus the CRSP benchmark.

Table 1: Performance Metrics

Metric	ST Strategy	CRSP Benchmark	Difference
Annualized Return	5.81%	9.73%	-3.92%
Sharpe Ratio	0.57	0.53	**0.04**
Max Drawdown	-13.79%	-34.98%	**21.19%**
Win Rate	48.96%	54.10%	-5.14%
Annualized Turnover	990.0%	N/A	N/A

3.1.5. Table Analysis.

The ST strategy delivers an average annualized return of 5.81%, which is lower than that of traditional long-only benchmarks. However, the strategy demonstrates better drawdown control, with a maximum drawdown of only -13.79%, contributing to a Sharpe ratio of 0.57, slightly higher than that of the benchmark. And the win rate hovers around 50%, consistent with a cross-sectional arbitrage profile.

3.2. HMM

DOI: 10.54254/2754-1169/2025.GL30493

3.2.1. Settings.

The HMM strategy classifies regimes using three market-level features: turbulence (PCA–Mahalanobis distance), volatility, and daily market returns. A two-state Gaussian HMM is estimated, with states mapped ex post to Bull (stable) and Bear (turbulent) regimes based on realized market drift. The strategy applies momentum signals in Bull states and short-term reversal signals in Bear states. Rebalancing occurs monthly in stable regimes and weekly in turbulent regimes. Transaction costs are excluded at this stage to focus on raw model behavior.

3.2.2. Graphs.

Figure 4 reports cumulative returns of the HMM strategy against the CRSP benchmark from 2015–2022, with regime overlays. The lower panels display turnover dynamics and a 63-day rolling Sharpe ratio.

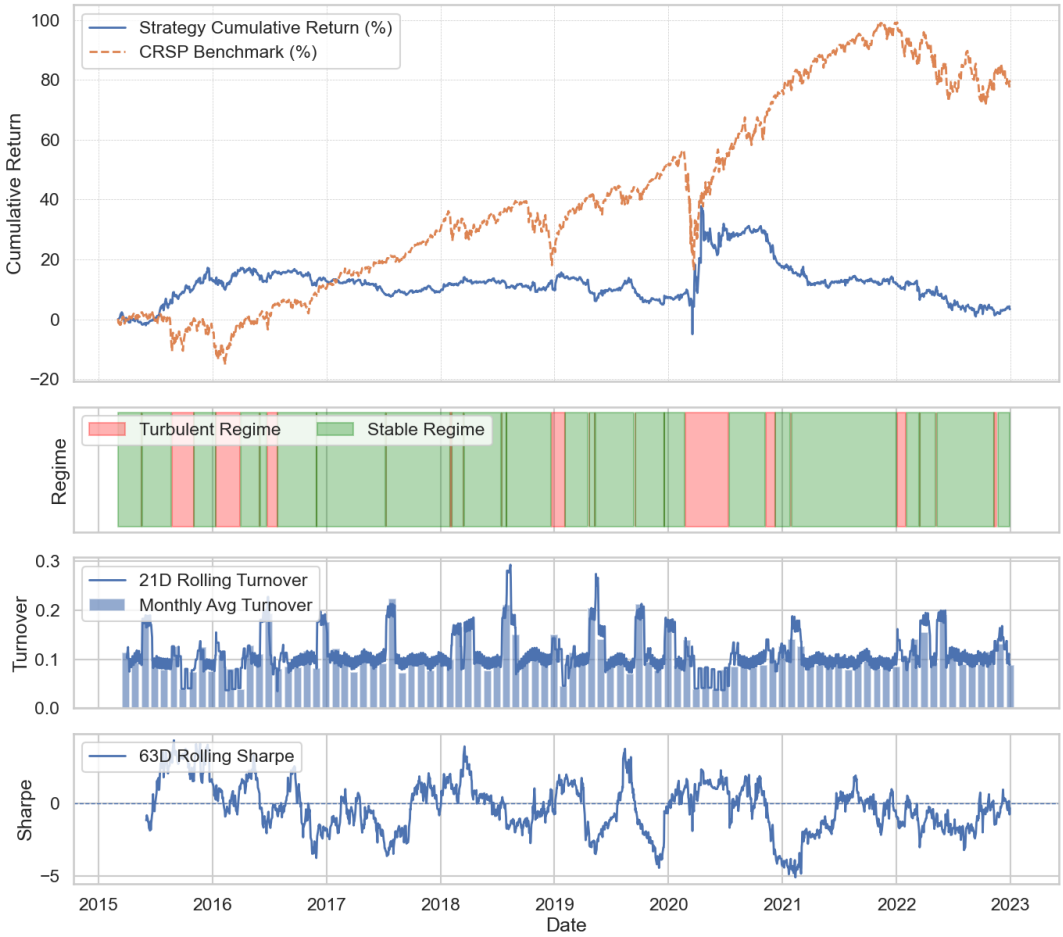

Figure 4: In-Sample Performance (Cumulative Returns vs. CRSP Benchmark, Regime Overlay, Turnover, and Rolling Sharpe)

DOI: 10.54254/2754-1169/2025.GL30493

3.2.3. Graph Analysis.

The HMM segments markets into long stable spans punctuated by distinct turbulent clusters, including the COVID shock and the 2022 volatility spike. Classification is directionally sensible, with only brief boundary noise around state changes. The portfolio switches to reversal in turbulent windows and momentum in stable ones, which trims some drawdowns but exhibits lower upside capture during prolonged bull markets. Returns remain modest because momentum signals decay near regime transitions and the model reacts with a short lag. Rolling Sharpe oscillates around zero, improving in clearly stable stretches and fading during choppy handoffs. Turnover rises around transitions and during late-sample turbulence, indicating that execution risk concentrates when the regime signal is least persistent.

3.2.4. Performance Metrics.

Table 2: Original HMM: Performance Metrics (In-Sample, No Cost)

Metric	HMM Strategy	CRSP Benchmark	Difference
Annualized Return	0.41%	9.73%	-9.32%
Sharpe Ratio	-0.04	0.53	-0.57
Max Drawdown	-26.75%	-34.98%	**8.23%**
Win Rate	52.47%	54.10%	-1.63%
Annualized Turnover	2702.9%	N/A	N/A

3.2.5. Table Analysis.

In-sample, the HMM strategy posts a near-zero annualized return (0.41%) with a slightly negative Sharpe (-0.04). Drawdowns are smaller than the CRSP benchmark (-26.75% vs. -34.98%), and the win rate is only marginally above 50% (52.47%). The chief concern remains implementability: annualized turnover is extremely high at roughly 2703%.

4. Refinement

4.1. ST Refinement

4.1.1. New Settings / Proposal.

We strengthen robustness and implementability through five changes:
(i) *Universe expansion*: Considering that mispricing is less likely to occur among large-cap stocks, we broaden the investment universe to include all CRSP/Compustat-linked securities listed on the NYSE, NASDAQ, and AMEX, including delisted firms;
(ii) *Stability*: Due to the inclusion of delisted firms and the expanded investment universe, extreme outliers become more prevalent, and the presence of thinly traded stocks—whose transaction costs are difficult to estimate—also increases. To address these issues, in addition to the original filter that excludes stocks priced below $5, we winsorize key input variables at the 1% tails and exclude stocks whose 60-day rolling average daily dollar volume ranks below the 0.1 threshold each day, effectively filtering out illiquid securities. Additionally, to enhance stability, we apply a 120-day rolling mean to

DOI: 10.54254/2754-1169/2025.GL30493

smooth the ST signal;
(iii) *Implementation frictions*: We adopt tiered transaction costs by market-cap bucket (0.001 / 0.002 / 0.005 for large-, mid-, and small-cap stocks, respectively);
(iv) *Risk controls*: We enforce dollar neutrality in the portfolio and cap individual position weights at 5%.
(v) *Implementation*: We construct a daily rebalanced long-short portfolio by taking long positions in the bottom 1% of stocks and short positions in the top 1%, with equal weighting assigned to each position.

4.1.2. In-sample Results.

Figure 5 displays the cumulative return of the refined, dollar-neutral ST strategy against the CRSP benchmark over the training window (2015–2022), while Figure 6 and Figure 7 report the rolling Sharpe and turnover.

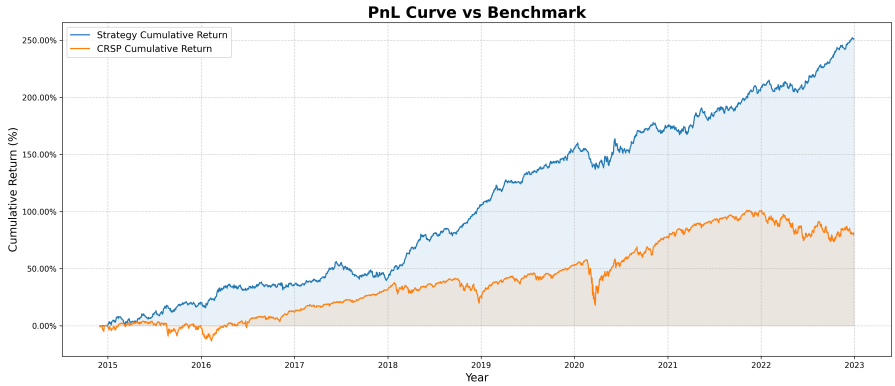

Figure 5: Refined ST: Cumulative Returns vs. CRSP Benchmark

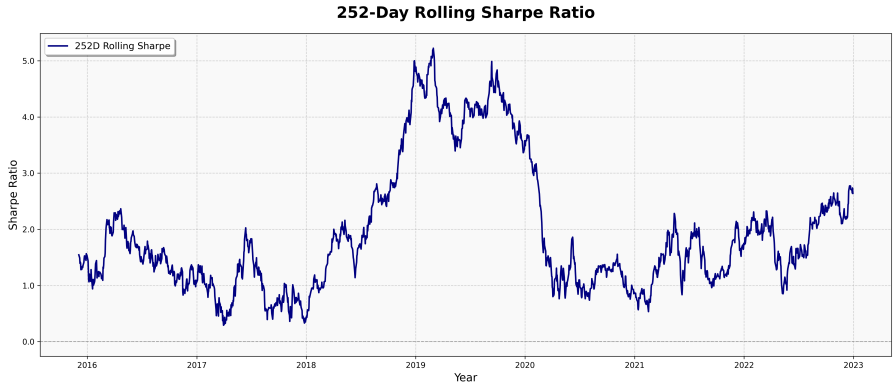

Figure 6: Refined ST: Rolling Sharpe

DOI: 10.54254/2754-1169/2025.GL30493

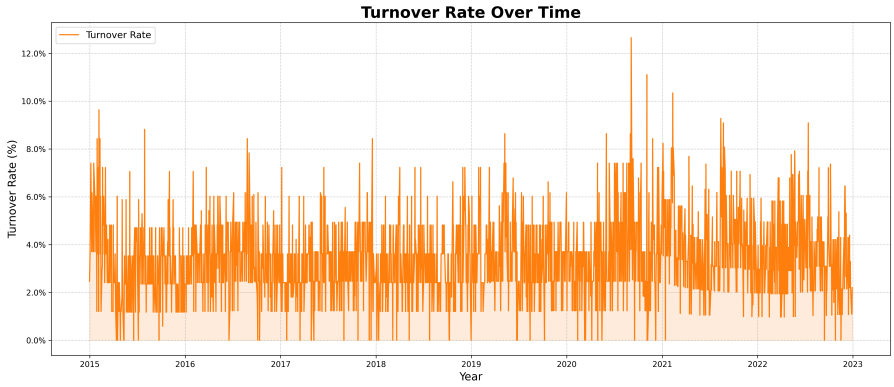

Figure 7: Refined ST: Turnover

Table 3: Refined ST: Performance Metrics (In-Sample)

Metric	**ST Strategy**	**CRSP Benchmark**	**Difference**
Annualized Return	31.15%	9.73%	**21.42%**
Sharpe Ratio	1.92	0.53	**1.39**
Max Drawdown	-10.63%	-34.98%	**24.35%**
Win Rate	54.75%	54.10%	**0.65%**
Annualized Turnover	858.06%	N/A	N/A

4.1.3. Table Analysis.

Relative to the baseline ST, the refined setup achieves a higher Sharpe ratio and reduced drawdowns, primarily due to improved outlier control, an expanded investment universe, rolling alpha smoothing, and a more selective signal application—restricting positions to only the top and bottom 1% of the signal distribution, thereby enhancing alpha exposure. Turnover is reduced as the original signal is smoothed using a rolling window, which increases its stability and reduces sensitivity to short-term fluctuations. Finally, the introduction of tiered transaction costs adds greater realism and rigor to the strategy's implementation.

4.1.4. Out-of-Sample Results (Without Cost).

Figure 8 shows the out-of-sample cumulative return profile (2023–2024). The refined ST strategy outperformed the CRSP benchmark in terms of raw returns, while maintaining a Sharpe ratio comparable to the benchmark, despite experiencing increased volatility toward the end of the test window.

DOI: 10.54254/2754-1169/2025.GL30493

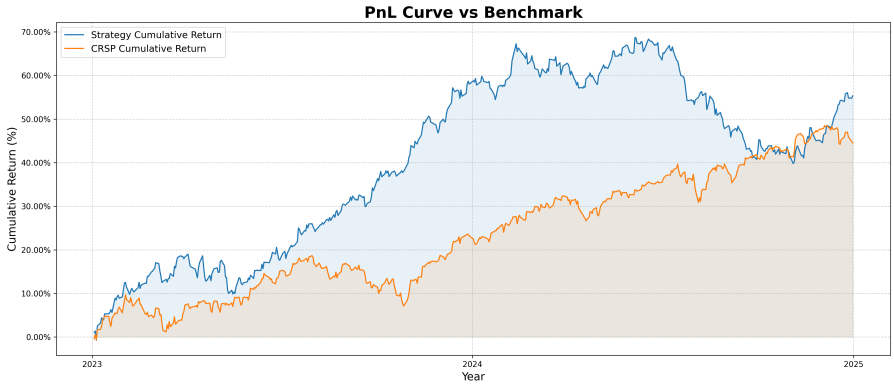

Figure 8: Refined ST: Cumulative Returns vs. CRSP Benchmark (Out-of-Sample, No Cost)

Table 4: Refined ST: Performance Metrics (Out-of-Sample)

Metric	**ST Strategy**	**CRSP Benchmark**	**Difference**
Annualized Return	27.77%	23.98%	**3.79%**
Sharpe Ratio	1.60	1.71	-0.11
Max Drawdown	-17.14%	-11.08%	-6.06%
Win Rate	52.59%	55.18%	-2.59%
Annualized Turnover	703.08%	N/A	N/A

4.1.5. Out-of-Sample Results (With Cost).

Once tiered transaction costs are applied, net performance remains positive but materially weaker. Figure 9 shows that the strategy's advantage over CRSP is largely eroded after accounting for execution frictions. The Sharpe ratio declines while drawdowns increase slightly, highlighting the importance of turnover control.

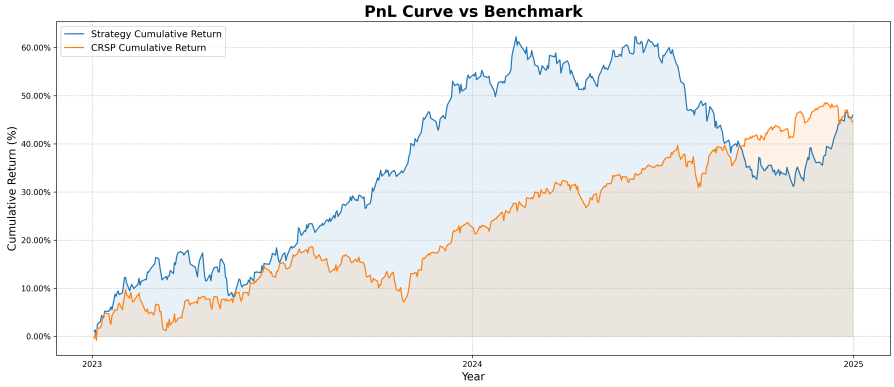

Figure 9: Refined ST: Cumulative Returns vs. CRSP Benchmark (Out-of-Sample, With Cost)

DOI: 10.54254/2754-1169/2025.GL30493

Table 5: Refined ST: Performance Metrics (Out-of-Sample, With Cost)

Metric	ST Strategy	CRSP Benchmark	Difference
Annualized Return	23.05%	23.98%	-0.93%
Sharpe Ratio	1.33	1.71	-0.38
Max Drawdown	-19.16%	-11.08%	-8.08%
Win Rate	52.19%	55.18%	-2.99%
Annualized Turnover	703.08%	N/A	N/A

4.1.6. Analysis.

The refined ST strategy produces robust in-sample results and demonstrates out-of-sample outperformance before accounting for costs. However, realistic frictions reduce net profitability and diminish its edge over the CRSP benchmark. Nonetheless, turnover remains low relative to the original strategy, and the smoothed ST signal provides more stable performance. Future refinements should focus on adaptive cost modeling and cross-factor integration to strengthen live implementability.

4.2. HMM Refinement

4.2.1. New Settings / Proposal.

The refinement introduced targeted changes to enhance regime detection and reduce spurious switches. First, the turbulence measure was extended to a 252-day horizon and computed via PCA–Mahalanobis distance, allowing the model to capture prolonged market stress more effectively. Second, the HMM feature window was lengthened to 63 days, incorporating both trailing mean and volatility of market returns. Third, all features were standardized prior to estimation, improving stability of the Gaussian HMM. Finally, hidden states were mapped to stable and turbulent regimes using realized market drift, ensuring clearer interpretability and a closer alignment between regime classification and observed market conditions.

4.2.2. In-Sample Results.

Figure 10 reports the in-sample performance of the refined HMM strategy. Relative to the original specification, regime classification is more coherent: volatile drawdowns such as the 2020 crash and the 2022 spike are now captured as turbulent states, while long advances are assigned to stable regimes. This reduces the number of spurious flips and improves continuity across market cycles. The refined strategy exhibits tighter drawdown control and modest cumulative gains, particularly by limiting losses during stress periods. While it still trails the CRSP benchmark in extended bull markets, the improved mapping between stable and turbulent phases results in a clearer and more interpretable regime structure, offering more reliable downside protection.

DOI: 10.54254/2754-1169/2025.GL30493

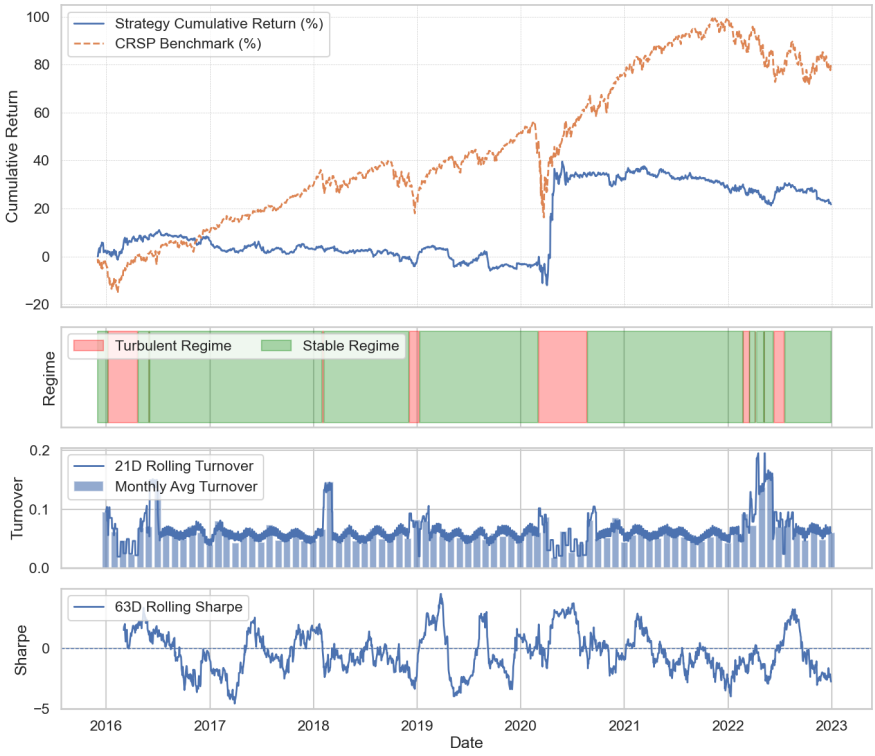

Figure 10: Refined HMM Strategy

Table 6: Refined HMM: Performance Metrics (In-Sample)

Metric	HMM Strategy	CRSP Benchmark	Difference
Annualized Return	2.80%	9.73%	-6.93%
Sharpe Ratio	0.16	0.53	-0.37
Max Drawdown	-20.77%	-34.98%	**14.21%**
Win Rate	50.81%	54.10%	-3.29%
Annualized Turnover	15.60%	N/A	N/A

4.2.3. Out-of-Sample Results.

Figure 11 reports the test-period performance without costs. The refined HMM delivered an annualized return of 11.72%, compared to 20.22% for the CRSP benchmark. Sharpe improved to 0.85, indicating reasonable risk-adjusted performance, though the strategy continued to underperform in absolute return terms.

DOI: 10.54254/2754-1169/2025.GL30493

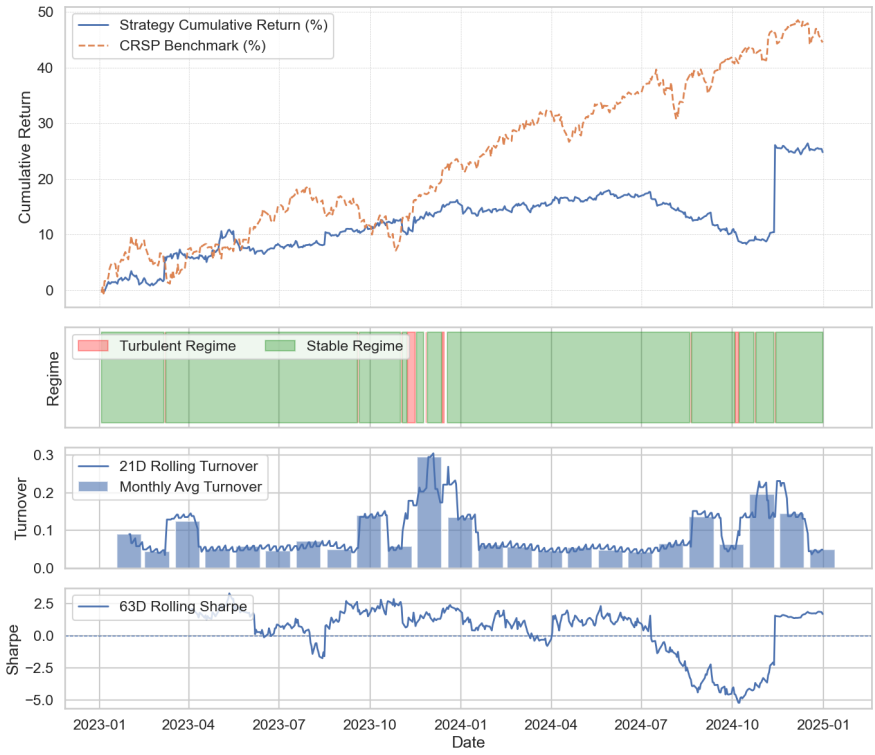

Figure 11: Refined HMM Strategy: Out-of-Sample

Table 7: Refined HMM: Performance Metrics (Out-of-Sample)

Metric	HMM Strategy	CRSP Benchmark	Difference
Annualized Return	11.72%	23.98%	-12.26%
Sharpe Ratio	0.85	1.71	-0.86
Max Drawdown	-8.22%	-11.08%	**2.86%**
Win Rate	52.98%	55.18%	-2.2%
Annualized Turnover	22.32%	N/A	N/A

4.2.4. Out-of-Sample Results (With Cost).

Including baseline transaction costs shifts the outcome materially. Figure 12 shows that the strategy's return falls to 7.65%, with Sharpe declining to 0.52. The benchmark outperforms decisively, underscoring that turnover and execution costs remain the main bottlenecks.

DOI: 10.54254/2754-1169/2025.GL30493

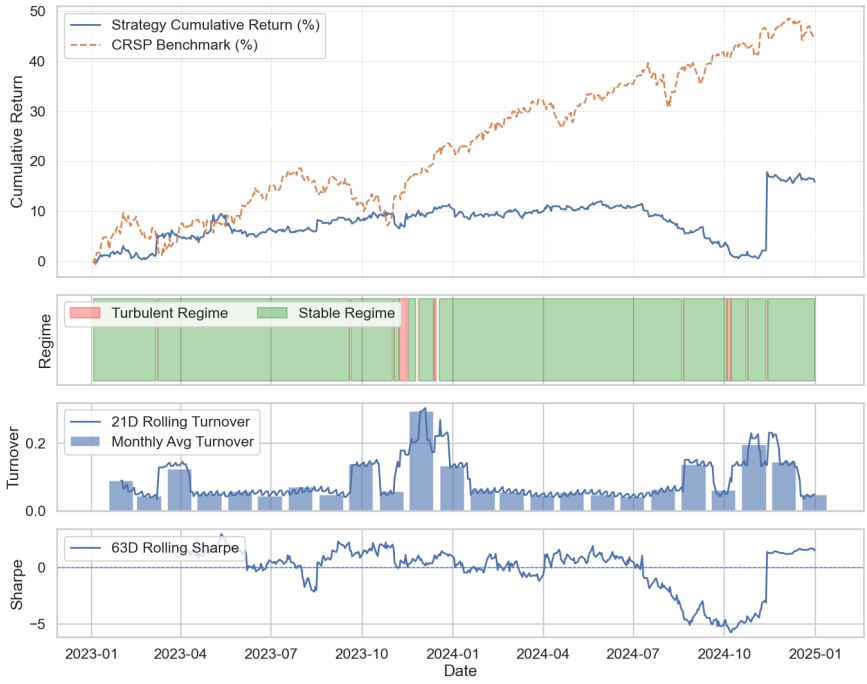

Figure 12: Refined HMM Strategy: Out-of-Sample (With Cost)

Table 8: Refined HMM: Performance Metrics (Out-of-Sample, With Cost)

Metric	HMM Strategy	CRSP Benchmark	Difference
Annualized Return	7.65%	23.98%	-16.33%
Sharpe Ratio	0.52	1.71	-1.19
Max Drawdown	-10.23%	-11.08%	**0.85%**
Win Rate	51.98%	55.18%	-3.2%
Annualized Turnover	22.32%	N/A	N/A

4.2.5. Analysis.

The refined HMM stabilized regime classification and improved resilience, particularly by reducing drawdowns both in- and out-of-sample. While the no-cost out-of-sample profile delivered reasonable Sharpe ratios, once transaction costs were incorporated the model lost competitiveness to the benchmark. This underscores that the design is better suited as a risk-management overlay than as a primary source of alpha, with future improvements needed in turnover control and cost-aware execution.

5. Conclusion

DOI: 10.54254/2754-1169/2025.GL30493

5.1. Final Selection of Refinement

The empirical results highlight that neither ST nor HMM in isolation provided a fully satisfactory outcome. ST consistently delivered positive cross-sectional alpha with low turnover but was sensitive to transaction costs and suffered from limited robustness across market regimes. In contrast, HMM provided regime awareness and reduced drawdowns in turbulent markets but underperformed on absolute returns and generated excessive turnover. The refined designs—smoothing and tiered cost adjustment for ST, and extended turbulence horizons with standardized features for HMM—improved both stability and interpretability. Nevertheless, the most effective configuration combined the strengths of both strategies.

5.2. The Combined Alpha

The final portfolio allocates 80% of capital to the refined ST strategy and 20% to the refined HMM overlay:

$$W_t^{\text{Hybrid}} = 0.8\, W_t^{\text{ST}} + 0.2\, W_t^{\text{HMM}}.$$

This weighting was chosen based on sensitivity analysis (as detailed below): ST generated superior standalone returns, while HMM contributed regime-conditioned downside protection. A heavier ST allocation maximizes return potential, while a smaller HMM allocation tempers drawdowns and regime whipsaws without overwhelming portfolio turnover. The result is a hybrid that balances profitability with robustness.

To be more precise, the daily returns of the HMM strategy are calculated, while rebalancing occurs on a weekly or monthly basis. The performance of the hybrid strategy is then derived by combining the daily returns of both strategies with a ratio of 80/20, resulting in a unified performance metric that reflects the contribution of both ST and HMM strategies.

5.3. Refined Performance (In-Sample and Out-of-Sample)

5.3.1. Out-of-Sample (No Cost).

Figure 13 plots cumulative returns of the hybrid versus the CRSP value-weighted benchmark; Figures 14 and 15 report the 252-day rolling Sharpe and daily turnover (with monthly bars), respectively. The hybrid outperformed during the first three quarters of the test window, then trailed as the benchmark rallied late in the period; the rolling Sharpe decayed from ≈ 3 to near zero by year-end, consistent with the visible catch-up of CRSP.

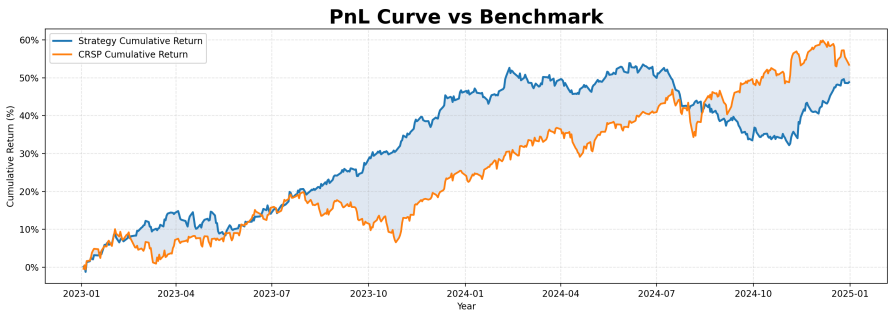

Figure 13: Hybrid vs. CRSP VW Benchmark: Cumulative Return (Out-of-Sample, No Cost)

DOI: 10.54254/2754-1169/2025.GL30493

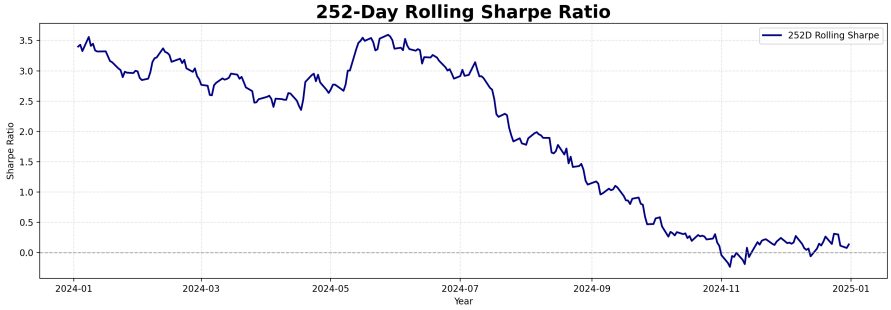

Figure 14: Hybrid: 252-Day Rolling Sharpe (Out-of-Sample, No Cost)

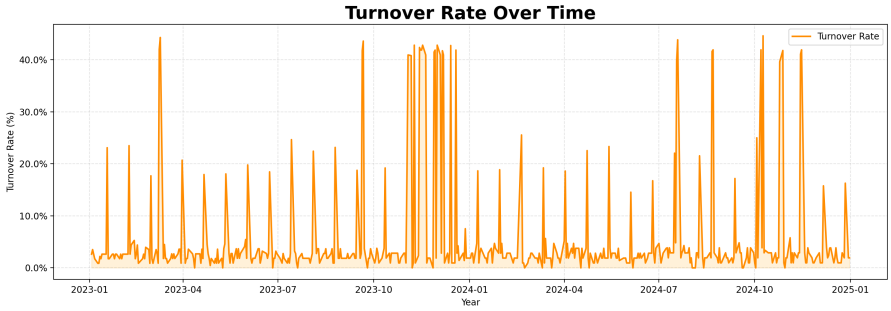

Figure 15: Hybrid: Turnover Over Time (Out-of-Sample, No Cost)

Table 9: Hybrid vs. CRSP Benchmark Performance (Out-of-Sample, No Cost)

Metric	**Hybrid Strategy**	**CRSP Benchmark**	**Difference**
Annualized Return	22.10%	23.98%	-1.88%
Sharpe Ratio	1.95	1.71	**0.24**
Max Drawdown	-14.09%	-11.08%	-3.01%
Win Rate	51.99%	55.18%	-3.19%
Annualized Turnover	1420.20%	N/A	N/A

5.3.2. Out-of-Sample (With Cost).

Figures 16–18 repeat the panels after applying execution costs. Net performance remains positive but the gap versus CRSP widens during late-year strength, and the rolling Sharpe dips below zero briefly during the same span. Turnover spikes line up with regime transitions and short rebalancing intervals, highlighting the cost sensitivity of the overlay.

DOI: 10.54254/2754-1169/2025.GL30493

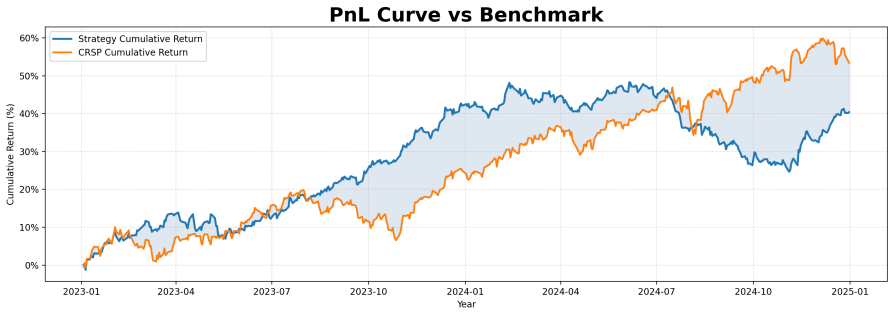

Figure 16: Hybrid vs. CRSP VW Benchmark: Cumulative Return (Out-of-Sample, With Cost)

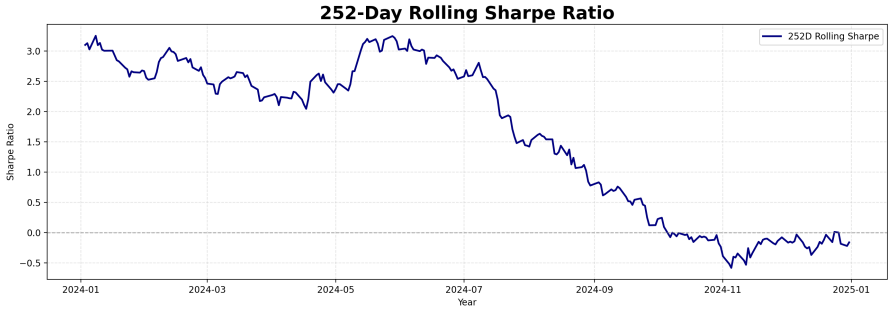

Figure 17: Hybrid: 252-Day Rolling Sharpe (Out-of-Sample, With Cost)

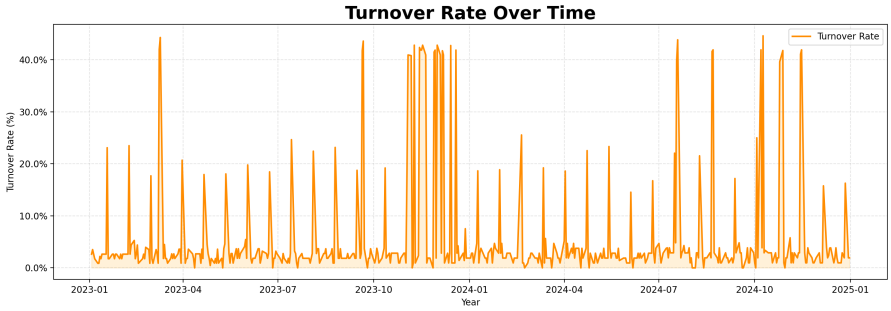

Figure 18: Hybrid: Turnover Over Time (Out-of-Sample, With Cost)

DOI: 10.54254/2754-1169/2025.GL30493

Table 10: Hybrid vs. CRSP Benchmark Performance (Out-of-Sample, With Cost)

Metric	Hybrid Strategy	CRSP Benchmark	Difference
Annualized Return	18.57%	23.98%	-5.41%
Sharpe Ratio	1.63	1.71	-0.08
Max Drawdown	-15.95%	-11.08%	-4.87%
Win Rate	51.79%	55.18%	-3.39%
Annualized Turnover	1420.20%	N/A	N/A

5.4. Sensitivity Analysis

The sensitivity analysis tests various allocation ratios (20/80, 50/50, 80/20) between the ST and HMM strategies. The results in Table 11 show that the 80/20 allocation provides the highest annualized return (22.10%) and Sharpe ratio (1.95), indicating superior risk-adjusted performance compared to other ratios. However, it also experiences the largest maximum drawdown (-14.09%). The 80/20 allocation was chosen primarily for its higher Sharpe ratio, reflecting a better balance of return and risk compared to other configurations.

Table 11: Sensitivity Analysis for the Allocation Ratio (Out-of-Sample, No Cost)

Ratio(ST/HMM)	Annualized Return	Sharpe ratio	Max Drawdown
20/80	15.40%	1.23	-5.33%
50/50	18.88%	1.86	-9.37%
80/20	22.10%	1.95	-14.09%

5.5. Analysis

From the perspective of ST, the refined ST strategy demonstrated substantial performance improvement in the in-sample period. Annualized returns increased by approximately 20%, maximum drawdown decreased by 3 percentage points, and the Sharpe ratio improved significantly from 0.57 to 1.92. Additionally, annualized turnover dropped by nearly 130%, reflecting a significant enhancement in cost efficiency and implementation stability.

However, the out-of-sample results show a noticeable decline in performance. Without accounting for transaction costs, the strategy achieved an annualized return of 27.77%, a Sharpe ratio of 1.60, and a maximum drawdown of 17%. After incorporating tiered transaction costs, performance declined to an annualized return of 23%, a Sharpe ratio of 1.33, and a slightly higher maximum drawdown of 19%. In summary, the refined ST strategy is a promising source of behavioral alpha, particularly under in-sample conditions. Yet, its out-of-sample robustness is limited, highlighting potential challenges in generalizability.

The HMM refinements improved stability and downside protection, but still underperformed the CRSP benchmark in total returns. This structural challenge reflects the fact that a dollar-neutral, regime-conditioned long–short strategy must compete with a long-only benchmark that consistently earns the equity risk premium. Although refined HMM showed resilience during turbulence and stayed profitable after costs, they could not match the sustained upside of CRSP.

DOI: 10.54254/2754-1169/2025.GL30493

The literature confirms this pattern. HMM added the most value when deployed as overlays for asset allocation rather than as direct stock-selection engines. Studies show that regime-switching between equities, bonds, or cash during volatility spikes reduces drawdowns and smooths return paths [8,**?**,**?**]. Hybrid machine-learning approaches, such as Tanaka's [11] regime-conditioned Random Forests, further improve predictive power by exploiting nonlinear regime-specific dynamics. Other research applies HMM at the factor level: Feature Saliency HMM improved smart beta allocations by 50–60% [12], while [13,**?**] show that regime detection stabilizes multi-factor strategies and prevents momentum crashes. Taken together, these comparisons suggest that the weakness of our HMM results lies not in the methodology itself, but in applying it to a highly competitive cross-sectional equity context.

5.6. Trading Recommendation and Considerations

While the refined hybrid demonstrates improved efficiency and stability, several risks remain. First, high turnover in HMM regimes can erode net returns once realistic slippage and borrow costs are included. Second, regime classification depends on features such as turbulence and volatility that may be unstable out-of-sample. Third, the market-neutral design shields against systematic beta exposure but introduces shorting costs and liquidity risks. Finally, survivorship-bias-free data and careful treatment of delistings are critical, and rolling HMM recalibrations risk look-ahead bias if not carefully implemented.

Practical verdict: the hybrid is promising as a research alpha factor or as part of a multi-factor allocation, but not yet strong enough for standalone live deployment. The 80–20 allocation reflects the relative strengths of each component: overweighting ST for consistent alpha, while retaining HMM exposure for resilience during turbulent regimes.

5.7. Limitation and Further Research

Limitation:
One limitation of this study is the relatively short out-of-sample period (2023-2024), which primarily represents a bull market recovery. This may not fully capture the model's performance across different market conditions, such as during periods of high volatility or bear markets. While this out-of-sample period provides valuable insights, future research could extend the out-of-sample testing period to include a more diverse range of market conditions. This would help assess the robustness of the strategy under different economic environments and improve the model's generalizability.
A second limitation is that while the hybrid strategy outperforms the benchmark in terms of risk-adjusted returns before accounting for transaction costs, it does not offer a strictly superior performance once these costs are included. This suggests that the hybrid strategy provides a different risk/return profile rather than being strictly better than the benchmark. The trade-off between return potential and transaction costs must be carefully considered, and future research could explore ways to reduce turnover and enhance cost efficiency to make the strategy more practically implementable.
Future refinements could explore:

1. integrating ST with other behavioral or fundamental signals (e.g., earnings drift, analyst revisions);
2. enhancing HMM regime classification with nonlinear ML methods (Random Forests, LSTMs) [11];
3. stress-testing turnover with explicit execution models including slippage and market impact;
4. applying the hybrid to other asset classes (bonds, futures, credit) where regime overlays add more value;

DOI: 10.54254/2754-1169/2025.GL30493

5. moving beyond a static 80–20 split toward dynamic weighting linked to regime confidence.
6. exploring more contextually relevant benchmarks, such as market-neutral factor models or peer groups of long-short equity hedge funds.

Overall, the refinements highlight the relative strengths of each approach: ST as a behavioral alpha engine, and HMM as a regime-aware overlay for risk management. Extending this framework to factor allocation or asset-class rotation [10,**?**,**?**,**?**] would provide a more realistic test of HMM's potential in live portfolio management.

6. Acknowledgement

Xin Lin, Qichen Huang, Zhuocheng Ni, Zhanrong Li, and Yuer Zhang contributed equally to this work and should be considered co-first authors.

References

[1] Cosemans, M., & Frehen, R. (2021). *Salience theory and stock prices: Empirical evidence*. Journal of Financial Economics, 140(2), 460–483.
[2] Wang, M., Lin, Y. H., & Mikhelson, I. (2020). *Regime-switching factor investing with hidden Markov models*. Journal of Risk and Financial Management, 13(12), 311.
[3] Bordalo, P., Gennaioli, N., & Shleifer, A. (2012). *Salience theory of choice under risk*. The Quarterly Journal of Economics, 127(3), 1243–1285.
[4] Sun, K., Wang, H., & Zhu, Y. (2023). *Salience theory in price and trading volume: Evidence from China*. Journal of Empirical Finance, 70, 38–61.
[5] Butt, H. A., Kolari, J. W., & Sadaqat, M. (2020). *Market volatility, momentum, and reversal*. SSRN Working Paper.
[6] Daniel, K., Jagannathan, R., & Kim, S. (2019). *A hidden Markov model of momentum*. Working paper.
[7] Kim, E., Lee, H., & Kim, S. (2019). *Global asset allocation strategy using a hidden Markov model*. Journal of Risk and Financial Management, 12(4), 187.
[8] Kritzman, M., Li, Y., Page, S., & Rigobon, R. (2010). *Principal components as a measure of systemic risk*. Journal of Portfolio Management, 37(4), 112–126.
[9] Stöckl, S., Hanke, M., & Angerer, M. (2014). *Financial applications of the Mahalanobis distance*. Applied Economics and Finance, 1(2), 79–89.
[10] Ibáñez, F. (2024). *Incorporating market regimes into large-scale stock allocation*. MPRA Paper No. 121552.
[11] Gonzáles Tanaka, J. C. (2025). *Market regime using hidden Markov model*. QuantInsti Blog.
[12] Fons, E., Dawson, P., Yau, J., Zeng, X., & Keane, J. (2019). *Dynamic asset allocation using feature saliency HMM for smart beta investing*. arXiv:1902.10849.
[13] Gu, J., & Mulvey, J. M. (2021). *Factor momentum and a regime-switching overlay strategy*. Journal of Financial Data Science, 3(4), 101–129.
[14] Shu, Y., & Mulvey, J. M. (2024). *Dynamic factor allocation leveraging regime-switching signals*. Quantitative Finance, forthcoming.

Appendix

Work Allocation

- Xin Lin (Phoenix) — Developed the Salience Theory proposal, carried out its coding, implementation, and refinement, technical research.

DOI: 10.54254/2754-1169/2025.GL30493

- Qichen Huang (Xavier) — HMM coding, implementation, and refinement, conducted the associated research, and finalized the paper by editing and assembling the complete version.
- Zhuocheng Ni (Taku) — Handled data cleaning, integration, and processing, and coded the combined strategies.
- Zhanrong Li (Morris) — Documentation, written assignments, drafted the initial paper sections, took detailed notes during class discussions, and ensured requirements were addressed.
- Yuer Zhang (Ella) - Documentation, written assignments, initial paper drafting, supported detail checks, and helped manage submission logistics.

ST Code

```
import pandas as pd
import numpy as np
import polars as pl
from tqdm import tqdm
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mtick
import statsmodels.api as sm


class Backtester:
    def __init__(self, df_path, beta_path, crsp_path, initial_capital=1e7):
        """
        Initialize Backtester with dataset paths.
        """
        self.df_path = df_path
        self.beta_path = beta_path
        self.crsp_path = crsp_path
        self.initial_capital = initial_capital
        self.df = None
        self.strategy = None
        self.result_df = None

    # ==================== Factor calculation ====================
    def calculate_ST(self, stock: pl.DataFrame) -> pl.DataFrame:
        """
        Calculate ST factor for each stock using a 20-day rolling window.
        """
        stock = stock.sort('date')
        stock = stock.with_columns(pl.col("date").cast(pl.Datetime("us")))

        if stock.height < 20:
            return stock.with_columns(pl.lit(np.nan).alias('ST'))[[
                'GVKEY','ticker','date','close','adjclose','return_d','ST','vol',
                'mv','adjfac','shrout','exch','status','date_delta'
            ]]

        results = []
        for i in range(19, stock.height):
            window = stock.slice(i-19,20)
            if window['close'].min() < 5:
                ST = np.nan
            elif stock[i,'vol_perc'] < 0.1:
                ST = np.nan
            elif window.slice(1,19)['date_delta'].max() > 20:
                ST = np.nan
            else:
```

DOI: 10.54254/2754-1169/2025.GL30493

```
                window = window.with_columns([
                    pl.col('sigma').rank('ordinal',descending=True).alias('rank')
                ]).with_columns([
                    (0.7 ** pl.col('rank')).alias('exp')
                ])
                w = window['exp'] / window['exp'].sum()
                ST = ((w - 1 / 20) * window['return_d']).sum()

            results.append({
                'GVKEY': stock[i,'GVKEY'],
                'ticker': stock[i,'ticker'],
                'date': stock[i,'date'],
                'close': stock[i,'close'],
                'adjclose': stock[i,'adjclose'],
                'return_d': stock[i,'return_d'],
                'ST': ST,
                'vol': stock[i,'vol'],
                'mv': stock[i,'mv'],
                'adjfac': stock[i,'adjfac'],
                'shrout': stock[i,'shrout'],
                'exch': stock[i,'exch'],
                'status': stock[i,'status'],
                'date_delta': stock[i,'date_delta']
            })
        return pl.DataFrame(results)

    def winsorize_series(self, series, lower=0.01, upper=0.99):
        """
        Winsorize extreme values in a series.
        """
        low = series.quantile(lower)
        up = series.quantile(upper)
        return np.clip(series, low, up)

    def labeling(self, group):
        """
        Assign long/short/middle labels based on ST_rolling ranks.
        """
        group = group.copy()
        group['ST_rank'] = group['ST_rolling'].rank(method='first', ascending=True)
        n = len(group[~group['ST_rolling'].isna()])
        group['label'] = 'middle'
        group.loc[group['ST_rank'] <= n*0.01, 'label'] = 'long'
        group.loc[group['ST_rank'] >= n*0.99, 'label'] = 'short'
        return group

    def assign_cost_rate(self, g):
        """
        Assign transaction cost rate by market cap quantiles.
        """
        g['cost_rate'] = pd.qcut(
            g['mv'], q=[0, 0.3, 0.7, 1.0],
            labels=[0.005, 0.001, 0.0005]
        ).astype(float)
        return g

    def period_sharpe(self, df):
        """
        Calculate annualized Sharpe ratio within a rolling window.
        """
        final_value = self.initial_capital + df['pnl'].sum()
```

DOI: 10.54254/2754-1169/2025.GL30493

```
        total_periods = 252
        re = final_value / self.initial_capital - 1
        annual_return = re * (252 / total_periods)
        vol = df['net_return'].std()
        annual_vol = vol * np.sqrt(252)
        return annual_return / annual_vol if annual_vol > 0 else np.nan

    # ==================== Data Preparation ====================
    def load_and_prepare_data(self):
        """
        Load raw data, compute ST, merge beta and CRSP.
        """
        df = pd.read_parquet(self.df_path)
        df['date'] = pd.to_datetime(df['date'])
        date_diff = df.groupby('ticker')['date'].diff()
        df['date_delta'] = date_diff.dt.days.fillna(0)
        df['avg_return'] = df.groupby('date')['return_d'].transform('mean')
        df['sigma'] = abs(df['return_d'] - df['avg_return'])/(abs(df['return_d']) + abs(df['avg_return']) + 0.1)
        df = pl.from_pandas(df).sort(['ticker','date'])

        results = []
        for group in tqdm(df.partition_by('ticker', as_dict=False), desc='Calculate ST'):
            results.append(self.calculate_ST(group))
        df = pl.concat(results).to_pandas()
        df = df.to_pandas()

        # Rolling ST
        df['ST_rolling'] = df.groupby('ticker')['ST'].transform(lambda x: x.rolling(window=120,min_periods=1).mean())

        # Winsorize
        for v in ['close','adjclose','return_d','ST','vol','mv','adjfac','shrout','ST_rolling']:
            df[v] = self.winsorize_series(df[v])

        # Normalize
        df['ST'] = (df['ST'] - df['ST'].mean())/df['ST'].std()
        df['ST_rolling'] = (df['ST_rolling'] - df['ST_rolling'].mean())/df['ST_rolling'].std()
        df['year'] = df['date'].dt.year
        df = df[df['date'] <= '2024-12-31']

        # Merge beta
        df_beta = pd.read_parquet(self.beta_path)
        df = df.merge(df_beta, on=['date','ticker'], how='left')
        df['beta'] = df.groupby('ticker')['beta'].ffill().bfill()
        self.df = df

    # ==================== Backtest ====================
    def run_backtest(self):
        """
        Run daily backtest with cost and turnover.
        """
        df = self.df.copy()
        df = df.groupby('date', group_keys=False).apply(self.labeling)
        df['label_next'] = df.groupby('ticker')['label'].shift(1).fillna('middle')

        label_map = {'long': 1.0, 'short': -1.0, 'middle': 0.0}
        df['raw_weight'] = df['label_next'].map(label_map)
        df['abs_weight'] = df['raw_weight'].abs()
```

DOI: 10.54254/2754-1169/2025.GL30493

```
        df['weight'] = df.groupby('date').apply(
            lambda g: g['raw_weight'] / g['abs_weight'].sum()
        ).reset_index(level=0, drop=True).fillna(0)

        df = df.sort_values(['ticker','date']).copy()
        df['prev_weight'] = df.groupby('ticker')['weight'].shift(1).fillna(0)
        df['weight_change'] = (df['weight'] - df['prev_weight']).abs()

        turnover_df = df.groupby('date')['weight_change'].sum().rename("turnover").
    reset_index()

        # Transaction cost
        df = df.groupby('date', group_keys=False).apply(self.assign_cost_rate)
        df['trade_cost'] = df['weight_change'] * df['cost_rate']
        cost_df = df.groupby('date')['trade_cost'].sum().rename("cost").reset_index()

        # Strategy return
        strategy = df.groupby(['date','label_next'])['return_d'].mean().unstack()
        strategy['return_d'] = strategy['long'] - strategy['short']
        strategy = strategy.reset_index()
        strategy = strategy.merge(cost_df,on='date',how='left').merge(turnover_df,on='
    date',how='left')
        strategy['net_return'] = strategy['return_d'] - strategy['cost']
        strategy = strategy.fillna(0)

        # PnL
        strategy['pnl'] = self.initial_capital * strategy['net_return']
        strategy['cum_pnl'] = strategy['pnl'].cumsum()
        strategy['date'] = pd.to_datetime(strategy['date'])
        strategy['year'] = strategy['date'].dt.year
        self.strategy = strategy

    # ==================== Performance ====================
    def summarize_results(self):
        """
        Summarize yearly and total performance metrics.
        """
        results = []
        for year, group in self.strategy.groupby('year'):
            if len(group) < 10: continue
            final_value = self.initial_capital + group['pnl'].sum()
            re = final_value / self.initial_capital - 1
            annual_return = re * (252 / len(group))

            group['c_pnl'] = group['pnl'].cumsum()
            portfolio_value = self.initial_capital + group['c_pnl']
            peak = portfolio_value.cummax()
            max_drawdown = (1 - portfolio_value / (peak+1e-12)).max()

            vol = group['net_return'].std()
            annual_vol = vol * np.sqrt(252)
            sharpe = annual_return / annual_vol if annual_vol > 0 else np.nan
            win_rate = (group['net_return'] > 0).mean()
            avg_turnover = group['turnover'].mean()

            results.append({
                'Year': year,
                'Annual return': annual_return,
                'Max drawdown': max_drawdown,
                'Sharpe': sharpe,
                'Win rate': win_rate,
```

DOI: 10.54254/2754-1169/2025.GL30493

```
                'Turnover': avg_turnover
            })

        result_df = pd.DataFrame(results).set_index('Year')
        self.result_df = result_df
        return result_df

    # ==================== Visualization ====================
    def plot_cum_pnl(self):
        """
        Plot cumulative returns vs CRSP benchmark.
        """
        strategy = self.strategy.copy()
        df_crsp = pd.read_csv(self.crsp_path).rename(columns={'caldt':'date','vwretd':'v_crsp'})
        df_crsp['date'] = pd.to_datetime(df_crsp['date'])
        strategy = strategy.merge(df_crsp[['date','v_crsp']], on='date', how='left')
        strategy['pnl_crsp'] = self.initial_capital * strategy['v_crsp']
        strategy['cum_pnl_crsp'] = strategy['pnl_crsp'].cumsum()
        strategy['cum_return'] = strategy['cum_pnl']/self.initial_capital*100
        strategy['cum_return_crsp'] = strategy['cum_pnl_crsp']/self.initial_capital*100

        plt.figure(figsize=(16,7))
        plt.plot(strategy['date'], strategy['cum_return'], color='#1f77b4', linewidth=1.5, label='Strategy')
        plt.fill_between(strategy['date'], strategy['cum_return'], color='#1f77b4', alpha=0.1)
        plt.plot(strategy['date'], strategy['cum_return_crsp'], color='#ff7f0e', linewidth=1.5, label='CRSP')
        plt.fill_between(strategy['date'], strategy['cum_return_crsp'], color='#ff7f0e', alpha=0.1)
        plt.gca().xaxis.set_major_locator(mdates.YearLocator())
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter(decimals=2))
        plt.title('Cumulative Return vs Benchmark')
        plt.legend(); plt.tight_layout(); plt.show()

    def plot_rolling_sharpe(self, window=252):
        """
        Plot rolling Sharpe ratio.
        """
        rolling_sharpe = []
        for i in range(len(self.strategy)):
            if i < window: rolling_sharpe.append(np.nan)
            else: rolling_sharpe.append(self.period_sharpe(self.strategy.iloc[i-window+1:i+1]))
        self.strategy['rolling_sharpe'] = rolling_sharpe

        plt.figure(figsize=(16,7))
        plt.plot(self.strategy['date'], self.strategy['rolling_sharpe'], color='navy', linewidth=2, label='Rolling Sharpe')
        plt.axhline(0, color='gray', linestyle='--')
        plt.title(f'{window}-Day Rolling Sharpe Ratio')
        plt.legend(); plt.tight_layout(); plt.show()

    def plot_turnover(self):
        """
        Plot turnover rate over time.
        """
        strat = self.strategy.iloc[18:].copy()
        plt.figure(figsize=(16,7))
```

DOI: 10.54254/2754-1169/2025.GL30493

```
        plt.plot(strat['date'], strat['turnover'], color='#ff7f0e', linewidth=1.5, label='Turnover')
        plt.fill_between(strat['date'], strat['turnover'], color='#ff7f0e', alpha=0.15)
        plt.gca().yaxis.set_major_formatter(mtick.PercentFormatter(xmax=1, decimals=1))
        plt.gca().xaxis.set_major_locator(mdates.YearLocator())
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
        plt.title('Turnover Rate Over Time')
        plt.legend(); plt.tight_layout(); plt.show()


# ====================== Main Execution ======================
if __name__ == "__main__":
    bt = Backtester(
        df_path=r".\DATA\cleaned.parquet",
        beta_path=r".\DATA\df_beta.parquet",
        crsp_path=r".\DATA\return_on_crsp_index.csv"
    )
    bt.load_and_prepare_data()
    bt.run_backtest()
    result = bt.summarize_results()
    print(result)
    bt.plot_cum_pnl()
    bt.plot_rolling_sharpe()
    bt.plot_turnover()
```

Listing 1: HMM-Based Adaptive Alpha Strategy (Excerpt)

HMM Code

```
import os
import time
import warnings
from datetime import timedelta

import numpy as np
import pandas as pd
import pyarrow.parquet as pq
import matplotlib.pyplot as plt
import seaborn as sns
from joblib import Parallel, delayed
from hmmlearn import hmm
from sklearn.preprocessing import StandardScaler
from tqdm import tqdm

warnings.filterwarnings("ignore")

DATA_PATH = "top3000data.parquet"
DATE_COL, TICKER_COL, CLOSE_COL = "datadate", "tic", "prccd"
SHARES_COL = "cshoc"
CRSP_CSV_PATH = "return_on_crsp_index.csv"

IN_SAMPLE_START  = "2014-01-01"
IN_SAMPLE_END    = "2023-01-01"
OUT_SAMPLE_START = "2023-01-01"
END_DATE         = "2025-01-01"

ROLLING_WINDOW_TURBULENCE    = 252
ROLLING_WINDOW_HMM_FEATURES  = 63
TURB_METHOD                  = "pca"
TURB_PCA_COMPONENTS          = 10
```

DOI: 10.54254/2754-1169/2025.GL30493

```
HMM_STATES = 2
HMM_ITER   = 200
RANDOM_STATE = 42
INITIAL_CAPITAL = 100_000
TRANSACTION_COST_PER_TRADE = 0.001
PERCENTAGE_TOP_BOTTOM = 0.1
REBALANCE_BULL_DAYS = 21
REBALANCE_BEAR_DAYS = 5
RISK_FREE_RATE = 0.01 / 252

ROLLING_SHARPE_WINDOW = 63
ROLLING_TURNOVER_WINDOW = 21

def log(msg: str):
    print(f"[{time.strftime('%H:%M:%S')}] {msg}")

def compute_crsp_vw_returns(rets: pd.DataFrame, market_caps: pd.DataFrame) -> pd.Series:
    rets = rets.copy()
    lag_caps = market_caps.shift(1)
    valid = rets.replace([np.inf, -np.inf], np.nan).notna() & (lag_caps > 0)
    lag_caps_valid = lag_caps.where(valid, other=0.0)
    rets_valid = rets.where(valid, other=0.0)
    cap_sum = lag_caps_valid.sum(axis=1)
    cap_sum = cap_sum.where(cap_sum > 0, other=np.nan)
    weights = lag_caps_valid.div(cap_sum, axis=0)
    vw_ret = (weights * rets_valid).sum(axis=1)
    vw_ret.name = "CRSP_VW_Return"
    return vw_ret.fillna(0.0)

# Data Preparation
def prepare_data(path: str) -> tuple[pd.DataFrame, pd.DataFrame]:
    if not os.path.exists(path):
        raise FileNotFoundError(f"Data file not found: {path}")
    log("Loading price and shares data...")
    df = pq.read_table(path, columns=[DATE_COL, TICKER_COL, CLOSE_COL, SHARES_COL]).
    to_pandas()
    df[DATE_COL] = pd.to_datetime(df[DATE_COL], errors="coerce")
    df["tic"] = df["tic"].str.strip().str.upper()
    df = df.dropna(subset=[DATE_COL, "tic"])
    df = df.drop_duplicates(subset=[DATE_COL, "tic"], keep="first")

    log("Pivoting data to wide format...")
    prices = df.pivot(index=DATE_COL, columns=TICKER_COL, values=CLOSE_COL)
    shares = df.pivot(index=DATE_COL, columns=TICKER_COL, values=SHARES_COL)

    prices = prices.ffill()
    shares = shares.ffill()

    common_idx = prices.index.intersection(shares.index)
    common_cols = prices.columns.intersection(shares.columns)
    prices = prices.loc[common_idx, common_cols]
    shares = shares.loc[common_idx, common_cols]

    market_caps = prices * shares

    prices = prices.loc[IN_SAMPLE_START:END_DATE]
    market_caps = market_caps.loc[IN_SAMPLE_START:END_DATE]
    return prices, market_caps

# Turbulence
def _md2_pca(window_mat: np.ndarray, x_now: np.ndarray, k: int) -> float:
```

DOI: 10.54254/2754-1169/2025.GL30493

```
    mean_vec = window_mat.mean(axis=0, keepdims=True)
    Z = window_mat - mean_vec
    x0 = x_now - mean_vec.ravel()
    U, S, Vt = np.linalg.svd(Z, full_matrices=False)
    var = (S**2) / max(1, (Z.shape[0] - 1))
    k = min(k, len(var))
    if k == 0:
        return np.nan
    Vt_k = Vt[:k, :]
    var_k = np.where(var[:k] <= 1e-12, 1e-12, var[:k])
    proj = Vt_k @ x0
    return float((proj**2 / var_k).sum())

def compute_turbulence(
    returns_df: pd.DataFrame, window: int = 60,
    method: str = "pca", pca_components: int = 10, n_jobs: int = -1
) -> pd.Series:
    idx, n = returns_df.index, len(returns_df)
    out = np.full(n, np.nan, dtype=float)
    R = returns_df.to_numpy(dtype=float)

    def _worker(i):
        W, x = R[i - window:i, :], R[i, :]
        mask = np.isfinite(W).all(axis=0) & np.isfinite(x)
        if mask.sum() < 5:
            return (i, np.nan)
        Wc, xc = W[:, mask], x[mask]
        k = min(pca_components, min(Wc.shape[0] - 1, Wc.shape[1]))
        md2 = _md2_pca(Wc, xc, k)
        return (i, np.sqrt(md2) if np.isfinite(md2) else np.nan)

    log(f"Computing turbulence (method={method}, window={window}, k={pca_components}) ...
    ")
    t0 = time.time()
    results = Parallel(n_jobs=n_jobs, backend="loky")(delayed(_worker)(i) for i in range(
    window, n))
    for i, val in results:
        out[i] = val
    log(f"Turbulence done in {time.time() - t0:.1f}s")
    return pd.Series(out, index=idx, name="Turbulence")

# HMM
class HMMRegimeDetector:
    def __init__(self, n_components=HMM_STATES, n_iter=HMM_ITER, random_state=
    RANDOM_STATE):
        self.scaler = StandardScaler()
        self.model = hmm.GaussianHMM(
            n_components=n_components,
            covariance_type="full",
            n_iter=n_iter,
            tol=1e-4,
            random_state=random_state,
            min_covar=1e-6,
        )
    def train(self, X: pd.DataFrame):
        self.model.fit(self.scaler.fit_transform(X))
    def predict(self, X: pd.DataFrame) -> np.ndarray:
        return self.model.predict(self.scaler.transform(X))

def map_regimes(daily_returns_df: pd.DataFrame, regimes_series: pd.Series, n_states: int)
    :
```

DOI: 10.54254/2754-1169/2025.GL30493

```
    mkt = daily_returns_df.mean(axis=1)
    regime_ret = {s: mkt.loc[regimes_series[regimes_series == s].index].mean() for s in range(n_states)}
    order = sorted(regime_ret, key=regime_ret.get, reverse=True)
    return {order[0]: "Bull", order[1]: "Bear"}

def dollar_neutral_momentum(signal: pd.Series, pct: float) -> pd.Series:
    k = max(1, int(len(signal) * pct))
    top_idx, bot_idx = signal.nlargest(k).index, signal.nsmallest(k).index
    w = pd.Series(0.0, index=signal.index)
    w.loc[top_idx] = 0.5 / k
    w.loc[bot_idx] = -0.5 / k
    return w

def dollar_neutral_meanrev(signal: pd.Series, pct: float) -> pd.Series:
    k = max(1, int(len(signal) * pct))
    long_idx, short_idx = signal.nsmallest(k).index, signal.nlargest(k).index
    w = pd.Series(0.0, index=signal.index)
    w.loc[long_idx] = 0.5 / k
    w.loc[short_idx] = -0.5 / k
    return w

# Portfolio Construction
def construct_portfolio(daily_returns: pd.DataFrame, regimes: pd.Series, regime_map: dict) -> pd.DataFrame:
    weights = pd.DataFrame(0.0, index=daily_returns.index, columns=daily_returns.columns)
    last_reb, last_type = None, None
    for i in tqdm(range(1, len(daily_returns)), desc="Constructing Portfolio", leave=False):
        d, d_prev = daily_returns.index[i], daily_returns.index[i - 1]
        if d_prev not in regimes.index:
            weights.iloc[i] = weights.iloc[i - 1]
            continue
        rtype = regime_map.get(regimes.loc[d_prev], "Unknown")
        need_reb = (
            last_reb is None
            or (rtype == "Bull" and (d - last_reb).days >= REBALANCE_BULL_DAYS)
            or (rtype == "Bear" and (d - last_reb).days >= REBALANCE_BEAR_DAYS)
            or (rtype != last_type)
        )
        if need_reb:
            start = d_prev - pd.DateOffset(days=ROLLING_WINDOW_HMM_FEATURES)
            signal = daily_returns.loc[start:d_prev].mean(skipna=True).dropna()
            if signal.empty:
                weights.iloc[i] = weights.iloc[i - 1]
                continue
            if rtype == "Bull":
                w_today = dollar_neutral_meanrev(signal, PERCENTAGE_TOP_BOTTOM)
            elif rtype == "Bear":
                w_today = dollar_neutral_momentum(signal, PERCENTAGE_TOP_BOTTOM)
            else:
                w_today = pd.Series(0.0, index=daily_returns.columns)
            weights.iloc[i] = w_today.reindex(weights.columns, fill_value=0.0)
            last_reb, last_type = d, rtype
        else:
            weights.iloc[i] = weights.iloc[i - 1]
    return weights.apply(lambda s: s - s.mean(), axis=1)

def backtest(daily_returns: pd.DataFrame, weights: pd.DataFrame):
    """
    Aggregate daily gross simple returns (no compounding).
```

DOI: 10.54254/2754-1169/2025.GL30493

```
    """
    idx = daily_returns.index.intersection(weights.index)
    R, W = daily_returns.loc[idx], weights.loc[idx]
    port_r = (W.shift(1) * R).sum(axis=1)

    port_r_net = pd.Series(0.0, index=port_r.index)
    turnover = pd.Series(0.0, index=port_r.index)
    eq = pd.Series(INITIAL_CAPITAL, index=port_r.index)

    cum_simple = 0.0
    for i in range(1, len(port_r)):
        daily_turnover = (W.iloc[i] - W.iloc[i - 1]).abs().sum() / 2
        turnover.iloc[i] = daily_turnover
        tc = daily_turnover * 2 * TRANSACTION_COST_PER_TRADE
        r = port_r.iloc[i] - tc
        port_r_net.iloc[i] = r


        cum_simple += r
        eq.iloc[i] = INITIAL_CAPITAL * (1.0 + cum_simple)

    cum = port_r_net.cumsum()
    return port_r_net, eq, cum, turnover

def perf(daily_r: pd.Series, turnover: pd.Series, rf=RISK_FREE_RATE):
    """
    Performance using simple-aggregated returns for HPR and drawdowns.
    """
    r = daily_r.dropna()
    if r.empty:
        return {
            k: np.nan
            for k in [
                "HoldingPeriodReturn",
                "AnnualizedReturn",
                "Sharpe",
                "MaxDrawdown",
                "WinRate",
                "AnnualizedTurnover",
            ]
        }

    hpr = r.sum()

    ann = (1 + hpr) ** (252 / len(r)) - 1
    sharpe = np.sqrt(252) * (r - rf).mean() / ((r - rf).std() + 1e-12)

    wealth = 1.0 + r.cumsum()
    mdd = (wealth / wealth.cummax() - 1.0).min()

    win = (r > 0).mean()
    ann_turnover = turnover.mean() * 252
    return dict(
        HoldingPeriodReturn=hpr,
        AnnualizedReturn=ann,
        Sharpe=sharpe,
        MaxDrawdown=mdd,
        WinRate=win,
        AnnualizedTurnover=ann_turnover,
    )

```

DOI: 10.54254/2754-1169/2025.GL30493

```
def rolling_sharpe(daily_r: pd.Series, window: int = ROLLING_SHARPE_WINDOW, rf_daily: float = RISK_FREE_RATE) -> pd.Series:
    r = daily_r - rf_daily
    mu = r.rolling(window).mean()
    sd = r.rolling(window).std()
    rs = np.sqrt(252) * (mu / (sd + 1e-12))
    rs.name = f"RollingSharpe{window}"
    return rs

def rolling_turnover(turnover: pd.Series, window: int = ROLLING_TURNOVER_WINDOW) -> pd.Series:
    rt = turnover.rolling(window).mean()
    rt.name = f"RollingTurnover{window}"
    return rt

# Plotting
def plot_performance(
    cum_ret: pd.Series,
    regimes: pd.Series,
    benchmark_cum: pd.Series,
    turnover: pd.Series,
    daily_r: pd.Series,
    title: str,
    regime_map: dict,
):
    sns.set_theme(style="whitegrid", context="talk")
    fig, axes = plt.subplots(
        4, 1, figsize=(14, 16), sharex=True, gridspec_kw={"height_ratios": [3, 1, 1.2, 1.2]}
    )
    ax1, ax2, ax3, ax4 = axes
    fig.suptitle(title if title.strip() else " ", fontsize=18, fontweight="bold")

    ax1.plot(cum_ret.index, cum_ret * 100, lw=2, label="Strategy Cumulative Return (%)")
    if not benchmark_cum.empty:
        ax1.plot(benchmark_cum.index, benchmark_cum * 100, lw=2, linestyle="--", label="CRSP Benchmark (%)")
    ax1.set_ylabel("Cumulative Return")
    ax1.grid(True, which="both", linestyle="--", linewidth=0.5)
    ax1.legend(loc="upper left")

    name_to_num = {v: k for k, v in regime_map.items()}
    bull_state, bear_state = name_to_num.get("Bull"), name_to_num.get("Bear")
    ax2.fill_between(regimes.index, 0, 1, where=regimes == bull_state, color="red", alpha=0.3, label="Turbulent Regime")
    ax2.fill_between(regimes.index, 0, 1, where=regimes == bear_state, color="Green", alpha=0.3, label="Stable Regime")
    ax2.set_yticks([])
    ax2.set_ylabel("Regime")
    ax2.legend(ncol=2, loc="upper left")

    monthly_turnover = turnover.resample("M").mean()
    ax3.bar(monthly_turnover.index, monthly_turnover, width=25, alpha=0.6, label="Monthly Avg Turnover")
    rt = rolling_turnover(turnover, ROLLING_TURNOVER_WINDOW)
    ax3.plot(rt.index, rt, lw=2, label=f"{ROLLING_TURNOVER_WINDOW}D Rolling Turnover")
    ax3.set_ylabel("Turnover")
    ax3.legend(loc="upper left")

    rs = rolling_sharpe(daily_r, ROLLING_SHARPE_WINDOW, RISK_FREE_RATE)
    ax4.plot(rs.index, rs, lw=2, label=f"{ROLLING_SHARPE_WINDOW}D Rolling Sharpe")
```

DOI: 10.54254/2754-1169/2025.GL30493

```
    ax4.axhline(0, lw=1, linestyle="--")
    ax4.set_ylabel("Sharpe")
    ax4.set_xlabel("Date")
    ax4.legend(loc="upper left")

    plt.tight_layout(rect=[0, 0, 1, 0.96])
    plt.show()

# Main
def main():
    log("Starting main process...")
    prices, market_caps = prepare_data(DATA_PATH)
    rets = prices.pct_change()

    keep_cols = rets.notna().mean()[lambda x: x >= 0.90].index
    if len(keep_cols) >= 100:
        rets = rets[keep_cols]
        market_caps = market_caps[keep_cols]

    log("Loading CRSP benchmark from CSV...")
    df_crsp = pd.read_csv(CRSP_CSV_PATH).rename(columns={"caldt": "date", "vwretd": "
    CRSP_Return"})
    df_crsp["date"] = pd.to_datetime(df_crsp["date"])
    df_crsp = df_crsp.set_index("date").sort_index()
    crsp_benchmark_returns = df_crsp["CRSP_Return"]

    turb = compute_turbulence(rets, window=ROLLING_WINDOW_TURBULENCE, method=TURB_METHOD)
    mkt = rets.mean(axis=1)
    feats = pd.DataFrame(
        {
            "Turbulence": turb,
            "MarketAvgReturn": mkt.rolling(ROLLING_WINDOW_HMM_FEATURES).mean(),
            "MarketVolatility": mkt.rolling(ROLLING_WINDOW_HMM_FEATURES).std(),
        }
    ).dropna()

    idx = rets.index.intersection(feats.index)
    rets, feats = rets.loc[idx], feats.loc[idx]

    in_sample_set = feats.loc[IN_SAMPLE_START:IN_SAMPLE_END]
    out_sample_set = feats.loc[OUT_SAMPLE_START:END_DATE]
    if in_sample_set.empty:
        raise ValueError("Training (In-Sample) window is empty.")

    log(f"Training HMM (2-state) on In-Sample data from {IN_SAMPLE_START} to {
    IN_SAMPLE_END}...")
    det = HMMRegimeDetector(n_components=HMM_STATES)
    det.train(in_sample_set)

    in_sample_reg = pd.Series(det.predict(in_sample_set), index=in_sample_set.index)
    regime_map = map_regimes(rets.loc[in_sample_set.index], in_sample_reg, HMM_STATES)
    log(f"Regime mapping: {regime_map}")

    def run_period(name: str, Xp: pd.DataFrame):
        if Xp.empty:
            return log(f"\{name\} set is empty - skipping.")
        log(f"Scoring \{name\}...")
        reg = pd.Series(det.predict(Xp), index=Xp.index)

        rets_p = rets.loc[Xp.index]
        weights = construct_portfolio(rets_p, reg, regime_map)
```

DOI: 10.54254/2754-1169/2025.GL30493

```
        daily_r, _, cum, turnover = backtest(rets_p, weights)

        benchmark_r = crsp_benchmark_returns.reindex(cum.index).fillna(0.0)
        benchmark_cum = benchmark_r.cumsum()

        ms = perf(daily_r, turnover)
        mb = perf(benchmark_r, pd.Series(0.0, index=benchmark_r.index))

        summ = pd.DataFrame(
            {
                f"HMM Strategy ({name})": ms,
                f"CRSP Benchmark ({name})": mb,
            }
        )
        print(f"\n--- Performance Summary ({name}) ---")
        print(summ.round(4).to_markdown())

        plot_performance(
            cum, reg, benchmark_cum, turnover, daily_r,
            title=f" ",
            regime_map=regime_map,
        )

    run_period("In-Sample", in_sample_set)
    run_period("Out-of-Sample", out_sample_set)

if __name__ == "__main__":
    t0 = time.time()
    main()
    log(f"Total runtime: {time.strftime('%H:%M:%S', time.gmtime(time.time() - t0))}")
```

Listing 2: HMM-Based Adaptive Alpha Strategy Implementation