POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Enhanced Time Series Analysis and Trading Strategy Development Using Dynamic Time Warping

Master's Thesis

Supervisors

Ing. Riccardo Tinivella
Dr. Leonardo Orsi

Academic Supervisor
Prof. Monica Visintin

Candidate

Sahil Singh

Academic Year: 2024–2025
March 2026

Abstract

Financial time series display intricate, erratic, and non-stationary characteristics, rendering short-term price forecasting especially difficult in highly efficient markets. Dynamic Time Warping (DTW) has been suggested in previous literature as a method for detecting recurring temporal patterns; however, its practical utility within realistic trading constraints is still ambiguous. This thesis examines whether DTW-based pattern similarity can yield additional predictive and economic value beyond conventional technical indicators when assessed within a rigorously causal and validation-restricted framework.

We built a full experimental pipeline that uses DTW to make pattern-outcome features by comparing recent market windows to similar historical segments and summarising their returns after that. These characteristics are regarded as informational inputs and assessed in conjunction with traditional technical indicators within supervised classification models. It is required that all steps of preprocessing, scaling, and model selection be strictly causal. The selection of models and trading rules is conducted solely on validation data, succeeded by a singular out-of-sample test evaluation, walk-forward robustness analysis, and a zero-shot cross-asset assessment.

Empirical findings regarding the SP 500 index demonstrate that DTW features offer limited additional predictive insights on validation data; however, they do not produce consistent enhancements in out-of-sample performance on average. The results of the trading show a small but positive net performance, with significant drawdowns, which is in line with the weak signal-to-noise ratio of daily equity returns. A regime-conditional analysis shows that DTW-based features are more useful when the market is stable and has low volatility, while traditional technical indicators work better when the market has high volatility. A zero-shot evaluation of the GLD ETF indicates that DTW-based signals do not transfer effectively across assets without recalibration.

In general, the results show that DTW-based pattern similarity is not a better way to make predictions in all cases, but it can pick up useful information that is specific to certain situations when used carefully. This work emphasises the significance of rigorous causality, validation-constrained experimentation, and economic performance evaluation in the examination of pattern-matching techniques within financial markets.

Acknowledgments

I wish to extend my profound gratitude to my academic supervisor, Prof. Monica Visintin, for her unwavering direction, insightful input, and support during the progression of my thesis. Her proficiency and assistance have been vital in developing this work.

I extend my gratitude to my supervisors, Ing. Riccardo Tinivella and Ing. Leonardo Orsi, for their mentorship, patience, and enlightening discussions that rendered this research both tough and enjoyable. Their pragmatic viewpoint facilitated my linkage of theoretical principles to practical applications.

I extend my heartfelt gratitude to my colleagues and friends for their unwavering support, collaboration, and for fostering an inspiring environment during this journey.

I am profoundly grateful to my family for their unwavering love, patience, and support. Their faith in me has consistently provided strength and motivation during my academic pursuits and the execution of this thesis.

Sahil Singh

Declaration on the Use of Generative AI Tools

Generative AI tools were used in a limited and supportive capacity during the preparation of this thesis. Specifically, such tools were used for brainstorming ideas, obtaining assistance with programming-related questions, and improving the clarity and grammar of the written text.

All research design, data analysis, implementation, interpretation of results, and the final written content of this thesis were developed and completed independently by the author.

Table of Contents

List of Figures

List of Tables

Acronyms

SPX	S&P 500 Index.
GLD	SPDR Gold Shares ETF.
ETF	Exchange-Traded Fund.
PnL	Profit and Loss.
bps	Basis Points.
DD	Drawdown.
MaxDD	Maximum Drawdown.
VaR	Value at Risk.
CVaR	Conditional Value at Risk.
AR	Autoregressive.
MA	Moving Average.
ARIMA	Autoregressive Integrated Moving Average.
ML	Machine Learning.
LR	Logistic Regression.
L1	L1 Regularization (Lasso).
SVM	Support Vector Machine.
kNN	k-Nearest Neighbors.

AUC	Area Under the ROC Curve.
ROC	Receiver Operating Characteristic.
SNR	Signal-to-Noise Ratio.
WF	Walk-Forward.
DTW	Dynamic Time Warping.
PO	Pattern–Outcome.
SCB	Sakoe–Chiba Band.
EMA	Exponential Moving Average.
MACD	Moving Average Convergence Divergence.
RSI	Relative Strength Index.
PCA	Principal Component Analysis.
HMM	Hidden Markov Model.
LowVol	Low Volatility Regime.
HighVol	High Volatility Regime.

Chapter 1

Introduction

Financial markets generate large volumes of time-ordered data, including asset prices, returns, and derived indicators. These financial time series are characterized by non-stationarity, noise, regime shifts, and a low signal-to-noise ratio, collectively rendering short-horizon forecasting a formidable challenge. Despite these difficulties, identifying exploitable patterns in historical price movements remains a central objective of quantitative finance, both for predictive modeling and for the construction of systematic trading strategies.

Traditional approaches to financial time-series analysis rely on parametric econometric models or fixed technical indicators designed to capture trend, momentum, or volatility. While these methods are computationally efficient and interpretable, they typically summarize historical information using pointwise or linear transformations. As a result, they may fail to capture similarities in the *temporal evolution* of prices across different market periods.

Dynamic Time Warping (DTW) is a similarity measure originally developed to compare sequences that may differ in speed or local alignment. By allowing non-linear time alignment, DTW can identify recurring patterns even when they unfold over different temporal scales. This property makes DTW a natural candidate for pattern-based analysis of financial time series, where market dynamics may recur in form but not in timing.

However, the practical utility of DTW-based pattern similarity in financial forecasting remains unclear. Prior studies often report optimistic results, but many rely on exploratory evaluation procedures, non-causal preprocessing, or implicit tuning on test data. These practices complicate the assessment of whether DTW provides incremental predictive information beyond simpler benchmark methods.

This thesis addresses this gap through a controlled experimental study that evaluates DTW-based pattern similarity as a source of additional information within a strictly causal and validation-locked machine-learning pipeline. Rather than proposing DTW as a standalone trading strategy, the study investigates whether DTW-derived features contribute incremental predictive and economic value when integrated with conventional technical indicators under realistic evaluation constraints.

1.1 Problem Definition and Motivation

1.1.1 Financial Time-Series Formulation

Let $\{P_t\}_{t=1}^{T}$ denote the sequence of daily closing prices of a financial asset observed at equally spaced trading days. Since price levels are non-stationary and evolve multiplicatively, the analysis is conducted on logarithmic returns rather than raw prices. The log return at time t is defined as

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right), \tag{1.1}$$

which provides approximate stationarity and symmetric treatment of gains and losses.

At each time t, the information set available to the model consists of the most recent $W \in \mathbb{N}$ observations,

$$\mathcal{F}_t = \{r_t, r_{t-1}, \ldots, r_{t-W+1}\}. \tag{1.2}$$

The forecasting task considered in this thesis is next-day directional prediction. Specifically, the model estimates the conditional probability

$$p_t = \mathbb{P}(r_{t+1} > 0 \mid \mathcal{F}_t), \tag{1.3}$$

representing the likelihood that the return from close t to close $t+1$ is positive.

This formulation induces a binary classification target,

$$y_t = \begin{cases} 1, & \text{if } r_{t+1} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{1.4}$$

Daily horizons are chosen to balance practical relevance with methodological rigor. For highly liquid assets such as equity indices, next-day direction is known to be extremely difficult to predict, making this setting well-suited for evaluating the incremental informational value of additional features.

1.1.2 Objective of the Research

The primary objective of this thesis is to rigorously assess whether Dynamic Time Warping (DTW)–based pattern similarity features provide incremental predictive and economic information beyond standard technical indicators when evaluated under realistic constraints.

The specific objectives are:

- To construct a strictly causal preprocessing and feature-engineering pipeline based on daily price data.
- To apply DTW to identify historical market windows that are similar in shape to the most recent observation window.

- To extract pattern–outcome (PO) features by summarizing realized returns following historically similar patterns.
- To integrate technical and DTW-derived features into supervised probabilistic classification models.
- To map probabilistic forecasts into trading positions using validation-selected, cost-aware decision rules.
- To evaluate predictive accuracy and economic performance using chronological train/validation/test splits, walk-forward analysis, and cross-asset generalization.

All feature configurations are evaluated under identical preprocessing, modeling, and evaluation protocols. Any observed performance differences can therefore be attributed to the inclusion of DTW-based information rather than experimental design choices.

1.1.3 Motivation from a Financial Engineering Perspective

Financial time series exhibit noise, non-stationarity, volatility clustering, and temporal distortions. Market episodes that appear qualitatively similar may unfold at different speeds or exhibit local phase shifts. Distance measures that enforce strict temporal alignment, such as the Euclidean norm, are therefore ill-suited for identifying recurring structures in such data.

Dynamic Time Warping addresses this limitation by allowing non-linear alignment of the time axis while preserving temporal ordering. By focusing on shape similarity rather than pointwise correspondence, DTW can identify historical price evolutions that resemble the current market state even when their timing differs.

From a financial engineering perspective, DTW is not employed to forecast prices directly. Instead, it serves as a contextual mechanism for identifying historical analogues whose subsequent behavior may contain information about near-term outcomes. Pattern similarity is thus treated as an informational input rather than a predictive model in itself.

1.1.4 Scientific Formulation

The overall system proposed in this thesis can be represented as a composition of operators acting on sequential price data:

$$s_t = \Psi \circ \Phi \circ \Gamma \circ \Lambda(P_{1:t}), \tag{1.5}$$

where:

- $\Lambda(\cdot)$ denotes strictly causal feature extraction from historical prices,
- $\Gamma(\cdot)$ computes DTW-based similarity and pattern–outcome features,

- $\Phi(\cdot)$ is a supervised learning model estimating p_t,
- $\Psi(\cdot)$ maps probabilistic forecasts into trading decisions.

Each operator is implemented modularly, with all parameters determined exclusively on validation data. This structure enforces a clear separation between feature construction, prediction, and decision-making.

1.1.5 Why Precision and Causality Matter

In financial modeling, even minor violations of causality can generate spurious predictability. Operations such as symmetric filtering, global normalization, or improper reuse of test data introduce future information into historical features, resulting in overly optimistic performance estimates.

Given the weak signal environment of financial markets, strict causality enforcement is essential. Throughout this thesis, all preprocessing, scaling, dimensionality reduction, and model fitting steps rely solely on information available at the time of prediction. This ensures that all reported results correspond to strategies that could, in principle, be implemented in real time.

1.1.6 Challenges in Financial Time-Series Modeling

Financial time-series modeling differs fundamentally from modeling physical or engineered systems. Asset returns are driven by heterogeneous and evolving mechanisms, resulting in dynamics that are difficult to model and weakly predictable.

Key challenges include non-stationarity and structural breaks, low signal-to-noise ratios, temporal distortions, heteroskedasticity, and the presence of transaction costs. These factors limit the reliability of naïve evaluation procedures and necessitate disciplined experimental design.

Consequently, any observed predictive or economic gains must be interpreted as fragile, context-dependent, and subject to decay over time.

1.1.7 Research Significance

Rather than proposing a new algorithm, this thesis emphasizes methodological rigor and honest evaluation. Its significance lies in clarifying the conditions under which DTW-based pattern similarity may provide incremental information and, equally important, where its limitations arise.

By enforcing strict causality, validation-locked selection, walk-forward robustness checks, and cross-asset testing, the study contributes a realistic assessment of pattern-based similarity in financial markets.

1.1.8 Definitions and Terminology

For clarity and consistency, the following terms are used throughout the thesis:

- **Log return**: $r_t = \log(P_t/P_{t-1})$.

- **Next-day (close-to-close) return**: return from close t to close $t+1$.
- **Binary target**: indicator variable equal to 1 if $r_{t+1} > 0$ and 0 otherwise.
- **Model score / probability**: estimated $\hat{p}_t = \mathbb{P}(y_t = 1 \mid \mathbf{x}_t)$, representing the model's predicted probability that the next-day return is positive.
- **ROC curve**: plot of the true positive rate against the false positive rate across different classification thresholds.
- **AUC**: area under the ROC curve, measuring the ranking quality of probabilistic predictions.
- **Trading signal**: discrete trading position derived from probability thresholds. A *long* position corresponds to holding the asset (buy exposure), while a *flat* position indicates no exposure to the asset.
- **Entry threshold**: probability level above which a long trading position is initiated.
- **Exit threshold**: probability level below which an existing trading position is closed.
- **Minimum holding period**: minimum number of days that a trading position must remain open before it can be closed.
- **Transaction costs**: proportional costs representing slippage and execution frictions incurred when entering or exiting positions.
- **Equity**: cumulative value of the trading strategy over time.
- **Equity curve**: time series representing the evolution of cumulative strategy returns.
- **Drawdown**: decline in the equity curve from a historical peak to a subsequent trough.
- **Maximum drawdown**: the largest peak-to-trough decline observed in the equity curve during the evaluation period.
- **Sharpe ratio**: risk-adjusted return measure computed as the mean return divided by the standard deviation of returns; reported both before and after transaction costs.
- **Chronological split**: time-ordered partition of the dataset into training, validation, and test subsets.
- **Walk-forward evaluation**: rolling procedure in which models are periodically re-estimated and evaluated on subsequent out-of-sample data.

- **Market regime**: classification of market conditions based on volatility and trend indicators.
- **Zero-shot cross-asset evaluation**: application of a trained and frozen model to a different asset without retraining.
- **Look-ahead bias**: use of information that would not have been available at the time of prediction when constructing historical features.

Chapter 2

State of the Art

2.1 Overview and Purpose of the Review

The purpose of this chapter is to position the present work within the existing literature on financial time-series forecasting and pattern-based similarity methods. The review is not intended to be exhaustive; instead, it focuses on methodological themes that directly relate to the central research question of this thesis: *does Dynamic Time Warping (DTW)–based pattern similarity provide incremental predictive value beyond standard technical indicators when evaluated under realistic constraints?*

The chapter has three objectives. First, it reviews common approaches to short-horizon financial forecasting and highlights their assumptions and limitations. Second, it surveys how DTW and related pattern-matching methods have been applied in financial settings. Third, it identifies methodological gaps—especially regarding causality, evaluation discipline, and economic validation—that motivate the experimental design adopted in this thesis.

Throughout this chapter, emphasis is placed on *how* results are obtained rather than on reported performance levels. In quantitative finance, discrepancies across studies are often driven more by evaluation protocols than by differences in modeling techniques.

2.2 Time-Series Forecasting in Finance

2.2.1 Classical Econometric Foundations

Early approaches to financial time-series modeling are grounded in econometric theory, where asset returns are represented using linear stochastic processes. Common examples include autoregressive (AR), moving-average (MA), and autoregressive integrated moving-average (ARIMA) models [1, 2]. These methods assume stable statistical structure and parametric dependence between past and future observations.

While such models are theoretically well-founded and useful for interpretability and inference, they typically offer limited performance for short-horizon directional prediction. Empirical evidence indicates that linear dependence in daily returns

is weak and unstable, and core assumptions are frequently violated due to non-stationarity and structural breaks. As a result, classical econometric models are often more effective for macroeconomic analysis or volatility modeling than for next-day return direction forecasting.

2.2.2 Shift Toward Machine Learning Models

Motivated by the limitations of parametric methods, a substantial body of research has adopted machine learning (ML) approaches for financial prediction. Logistic regression, support vector machines, tree-based ensembles, and neural networks have all been explored [3, 4].

A key advantage of ML models is their flexibility: they need not assume linearity and can incorporate heterogeneous features such as lagged returns, technical indicators, and calendar effects. However, this flexibility increases the risk of overfitting, particularly in low signal-to-noise environments. Consequently, reported improvements are often sensitive to design choices such as data splitting, feature selection, and hyperparameter tuning. This motivates the strong evaluation discipline employed in the present thesis, where training, validation, and test periods are strictly separated and chronologically ordered.

2.2.3 Limitations of Pointwise Feature Representations

Most ML approaches in finance rely on *pointwise* feature representations, where historical information is summarized into fixed statistics at each time step (e.g., moving averages, oscillators, volatility estimates). Such representations capture local trend, momentum, or risk, but they compress the temporal evolution within the observation window.

Two market episodes may exhibit similar indicator values while corresponding to different underlying paths, and conversely, structurally similar price evolutions may be missed if they are temporally misaligned. This limitation motivates similarity-based methods that operate on full windows rather than on summary statistics.

2.2.4 Motivation for Pattern-Based Similarity Measures

Pattern-based approaches aim to identify historical episodes that resemble the current market state in terms of temporal evolution, and then use the outcomes following those historical patterns to inform near-term forecasts. This paradigm aligns with the intuitive notion of market analogues ("similar past situations"), but implementing pattern matching in a disciplined and scalable way is non-trivial.

Simple distance measures fail under temporal distortions, while more flexible similarity metrics introduce computational burden and additional design choices. In this context, DTW provides a principled method for comparing temporal shapes under non-linear alignment. The next section reviews DTW theory and variants that are relevant to financial applications.

2.3 Dynamic Time Warping: Theory and Variants

Dynamic Time Warping (DTW) is a similarity measure originally developed for comparing temporal sequences that may evolve at different speeds or exhibit local phase shifts [5]. Unlike pointwise distances, DTW permits adaptive alignment of indices while preserving temporal ordering. This section summarizes the core DTW formulation, common normalization practices, and key constraints and variants.

2.3.1 Basic Definition

Let

$$X = (x_1, x_2, \ldots, x_N), \qquad Y = (y_1, y_2, \ldots, y_M)$$

denote two real-valued sequences of lengths N and M, respectively. DTW begins by defining a local cost (or distance) between elements, typically the squared Euclidean cost:

$$D(i, j) = d(x_i, y_j), \qquad d(x_i, y_j) = (x_i - y_j)^2. \tag{2.1}$$

A *warping path* is a sequence of index pairs

$$\mathcal{P} = \{(i_k, j_k)\}_{k=1}^{K},$$

subject to the standard boundary, monotonicity, and continuity constraints. The DTW distance is defined as the minimum cumulative cost along any valid path:

$$\mathrm{DTW}(X, Y) = \min_{\mathcal{P}} \sum_{(i,j)\in\mathcal{P}} D(i, j). \tag{2.2}$$

This formulation allows non-linear alignment of the time axes, enabling sequences to be locally stretched or compressed while preserving order.

2.3.2 Normalization and Interpretation

Raw DTW distances depend on the scale of the input sequences. When absolute magnitude is not the object of comparison, normalization is typically applied prior to DTW computation. A common approach is z-normalization:

$$\tilde{x}_i = \frac{x_i - \mu_X}{\sigma_X}, \qquad \tilde{y}_j = \frac{y_j - \mu_Y}{\sigma_Y}, \tag{2.3}$$

where μ_X, σ_X and μ_Y, σ_Y are the mean and standard deviation of the corresponding sequences. Under such normalization, DTW primarily reflects *shape similarity* rather than scale, which is especially relevant in finance where volatility and amplitude may confound distance-based comparisons.

2.3.3 Warping Path Constraints

DTW flexibility can increase computational cost and may produce implausible alignments. Practical implementations therefore impose constraints on admissible warping paths.

2.3.3.1 Sakoe–Chiba Band

The Sakoe–Chiba band restricts the warping path to remain within a fixed radius r of the diagonal:

$$|i - j| \leq r. \tag{2.4}$$

This constraint limits excessive temporal distortion and reduces computational complexity from $O(NM)$ toward $O(r \cdot \max(N, M))$.

2.3.3.2 Itakura Parallelogram

The Itakura parallelogram imposes slope constraints on the warping path, preventing excessive compression or expansion of one sequence relative to the other. It is often less common in finance due to its asymmetry and stronger structural assumptions.

2.3.4 Approximate and Efficient Variants

To support large-scale nearest-neighbor search, several approximate DTW variants have been proposed, including lower-bound pruning, early abandoning, and indexing-based methods [6]. These methods reduce the number of full DTW computations but introduce additional design choices and approximation error. In noisy financial settings, such choices can affect similarity rankings and must be validated carefully.

2.3.5 Multi-Channel Dynamic Time Warping

Financial behavior is often represented using multiple channels (e.g., returns and filtered components). A common multivariate extension computes DTW distances per channel and aggregates them additively:

$$\mathrm{DTW}_{\mathrm{multi}}(\mathbf{X}, \mathbf{Y}) = \sum_{c=1}^{C} \mathrm{DTW}\left(X^{(c)}, Y^{(c)}\right), \tag{2.5}$$

where $X^{(c)}$ denotes the sequence for channel c. This formulation is interpretable and simple, but implicitly assumes comparable scaling and importance across channels.

2.3.6 Summary

DTW provides a flexible framework for comparing temporal sequences under non-linear alignment. Its usefulness depends critically on normalization, computational constraints, and design choices that control alignment flexibility. The next section reviews how DTW has been applied in financial modeling and highlights recurring methodological limitations.

2.4 Applications of Dynamic Time Warping in Financial Modeling

DTW has been used in finance primarily for detecting recurring patterns and leveraging historical analogues. This section reviews key strands of the literature, emphasizing methodology and evaluation practices.

2.4.1 Early Pattern-Matching Approaches

Early DTW applications in finance typically segment price or return trajectories into fixed-length windows and use DTW to retrieve historical windows similar to the current pattern [3]. Subsequent price movements following matched patterns are then used for forecasting or decision rules.

A frequent limitation in early work is the use of raw prices or unnormalized returns, which causes DTW distances to conflate shape similarity with scale and volatility differences, complicating interpretation.

2.4.2 DTW Integrated with Supervised Learning

Later studies integrate DTW into supervised pipelines by using DTW distances or pattern-derived summaries as features [4]. A common approach retrieves historical neighbors via DTW and constructs predictors from the outcomes following those neighbors. While this aligns with the pattern–outcome paradigm, many studies do not clearly separate feature construction, model selection, and evaluation, which can lead to optimistic estimates.

2.4.3 Indexing, Approximation, and Nearest-Neighbor Variants

To reduce computational cost, multiple works propose indexing and approximate nearest-neighbor methods, including lower-bound pruning and DTW-based indexing structures [6]. These methods improve scalability but introduce additional tuning parameters. In weak-signal financial environments, the interaction between approximation error and overfitting risk is rarely examined systematically.

2.4.4 Warping Constraints and Design Choices

Some studies examine how warping constraints affect DTW behavior [5]. Unconstrained DTW can produce implausible alignments, especially in noisy series, making constraint selection a meaningful modeling choice rather than a minor implementation detail. However, systematic out-of-sample studies of constraint parameter effects are limited.

2.4.5 Recurring Methodological Limitations

Across the DTW-in-finance literature, several recurring issues appear:

- **Non-causal preprocessing**, such as full-sample normalization or symmetric filtering.
- **Single-asset evaluation**, limiting generalization claims.
- **Loose evaluation protocols**, conflating calibration and testing.
- **Limited economic validation**, with insufficient attention to costs, turnover, and drawdowns.

These limitations make it difficult to determine whether reported performance reflects genuine predictive structure or methodological artifacts.

2.4.6 Summary

The literature supports DTW as a flexible tool for identifying recurring patterns under temporal distortion. However, many applications lack the rigor required to evaluate whether DTW provides incremental value beyond strong baselines under realistic constraints. The next section consolidates the main methodological gaps that motivate the present thesis.

2.5 Gaps Identified in the Literature

2.5.1 Insufficient Enforcement of Causality

A principal limitation in many DTW-based financial studies is the lack of explicit causality enforcement. Normalization, filtering, and scaling are often performed using the full dataset, allowing future data to influence historical features. In low signal-to-noise settings, even small leakage can materially inflate performance.

2.5.2 Weak Separation Between Model Selection and Assessment

Many studies tune DTW parameters, similarity thresholds, or trading rules using the same data later used for evaluation. Without explicit train/validation/test separation, it is difficult to distinguish genuine out-of-sample performance from in-sample optimization.

2.5.3 Limited Comparison Against Strong Technical Baselines

DTW is often evaluated in isolation or against weak baselines. This prevents clear conclusions about whether DTW adds incremental information beyond conventional indicators that already capture trend, momentum, and volatility.

2.5.4 Narrow Asset Scope and Lack of Generalization Testing

Most studies focus on a single asset or index, limiting the ability to separate asset-specific effects from transferable structure. Zero-shot cross-asset evaluation (no retraining or tuning) is rarely performed but provides a stringent robustness test.

2.5.5 Limited Focus on Economic Performance and Risk

A substantial portion of the literature emphasizes predictive metrics (e.g., accuracy, hit rates) while under-reporting transaction costs, turnover, and drawdowns. Small improvements in predictive metrics may not translate into economically viable strategies once realistic costs and risks are considered.

2.5.6 Summary of Identified Gaps

In summary, DTW-based financial research frequently suffers from non-causal preprocessing, inadequate validation discipline, limited comparisons against strong baselines, narrow asset scope, and incomplete economic assessment. These gaps motivate a controlled experimental study evaluating DTW-based pattern similarity under strict causality and validation-locked selection.

2.6 Theoretical Foundations Employed in This Work

This thesis adopts established principles from time-series analysis, DTW, machine learning, and financial evaluation. Rather than proposing new theory, the contribution lies in combining these principles into a coherent, strictly causal experimental framework.

2.6.1 Principles of Time-Series Modeling

Financial returns are non-stationary, weakly predictable, and subject to regime shifts. Therefore, this work emphasizes chronological data splitting, rolling evaluation, and cautious interpretation of performance metrics, motivating validation-locked selection and walk-forward robustness analysis.

2.6.2 Principles of Dynamic Time Warping

DTW is treated as a similarity operator rather than a forecasting model. Two design principles are emphasized: (i) similarity should reflect shape rather than scale (motivating window-level normalization), and (ii) warping flexibility should be constrained to avoid pathological alignments.

2.6.3 Principles of Machine Learning

A probabilistic classification framework is adopted, where models estimate conditional probabilities rather than making hard decisions. Key principles include validation-only model selection, one-shot test reporting, and the use of regularization and interpretable models in weak-signal settings.

2.6.4 Principles of Evaluation and Interpretation

Predictive accuracy and economic performance are evaluated separately. Statistical metrics (e.g., AUC) assess ranking quality, while economic metrics (e.g., Sharpe ratio,

drawdown) assess realizable performance under transaction costs. Regime analysis is used as an explanatory tool rather than as a mechanism for optimization.

2.6.5 Summary

By integrating established principles across these domains, the thesis provides a rigorous and reproducible evaluation of DTW-based pattern similarity under realistic financial constraints.

2.7 Summary of Key Points

This chapter reviewed the literature on financial time-series forecasting and DTW-based pattern matching with emphasis on methodological design. Classical econometric and ML approaches highlight the challenges of non-stationarity and low signal-to-noise ratios, while DTW provides a mechanism to compare temporal shapes under misalignment.

The literature on DTW in finance suggests that pattern matching is plausible, but frequently lacks rigorous causality enforcement, validation discipline, strong baseline comparison, and cost-aware economic evaluation. These gaps motivate the controlled experimental framework developed in the next chapter, where data construction, feature engineering, model training, and performance assessment are specified in detail.

Chapter 3

Methodology

This chapter describes the methodological framework developed to measure the incremental value of Dynamic Time Warping (DTW)–based pattern similarity for predicting and trading financial time series. The objective is not to propose a new algorithm, but to construct a strictly causal, reproducible, and validation-locked experimental pipeline that enables a fair comparison between DTW-derived features and a standard technical-indicator baseline. The methodology is aligned with the temporal structure of the problem: data construction, feature engineering, model training, and evaluation are specified sequentially, ensuring that no future information is used at any stage of prediction or decision-making. Particular emphasis is placed on (i) separating prediction from trading, and (ii) separating validation-based selection from the final out-of-sample test evaluation.

3.1 Overview of the Proposed Framework

3.1.1 Design Philosophy

The experimental design is guided by three principles:

1. **Strict causality.** All features, transformations, and model estimates at time t depend only on information available at or before t.

2. **Validation-locked selection.** All modeling and trading-rule choices are determined using validation data only; the test set is used exactly once for final reporting.

3. **Comparability.** All feature configurations are evaluated using the same preprocessing, modeling, and evaluation protocols.

These principles ensure that performance differences reflect informational content rather than methodological artifacts.

3.1.2 Conceptual Architecture

At a high level, the framework consists of four interacting components:

1. **Data preprocessing and feature construction**, transforming raw prices into technical and pattern-based representations.
2. **Similarity-based feature extraction**, using DTW to identify historical windows similar to the current market window and summarizing subsequent outcomes.
3. **Supervised learning**, training probabilistic classifiers to estimate next-day directional probabilities.
4. **Trading-rule design and evaluation**, mapping probability forecasts to positions and assessing economic performance under transaction costs.

Each component is modular, with clearly defined inputs and outputs, supporting transparency and reproducibility.

3.1.3 Mathematical Formalization

Let $\{P_t\}_{t=1}^{T}$ denote daily closing prices and $\{r_t\}_{t=2}^{T}$ the corresponding log-returns (Section 1.1.1). For each prediction time t, feature construction produces a vector

$$\mathbf{x}_t = [\mathbf{x}_t^{\text{TECH}},\ \mathbf{x}_t^{\text{DTW}}], \tag{3.1}$$

where $\mathbf{x}_t^{\text{TECH}}$ denotes technical and calendar-based features and $\mathbf{x}_t^{\text{DTW}}$ denotes DTW-derived pattern–outcome features.

Three feature configurations are evaluated:

- **TECH_ONLY:** $\mathbf{x}_t = \mathbf{x}_t^{\text{TECH}}$,
- **DTW_ONLY:** $\mathbf{x}_t = \mathbf{x}_t^{\text{DTW}}$,
- **TECH+DTW:** $\mathbf{x}_t = [\mathbf{x}_t^{\text{TECH}}, \mathbf{x}_t^{\text{DTW}}]$.

Given $\mathbf{x}_t$, a probabilistic classifier $f_\theta(\cdot)$ outputs

$$\hat{p}_t = f_\theta(\mathbf{x}_t) \approx \mathbb{P}(r_{t+1} > 0 \mid \mathbf{x}_t). \tag{3.2}$$

Model parameters θ are learned using training data only, while hyperparameters and configuration choices are selected using validation data only.

3.1.4 Mapping from Prediction to Decision

Predicted probabilities are converted into trading positions using a threshold-based decision rule. Let q_{entry} and q_{exit} denote upper and lower probability thresholds determined on the validation set. The trading position $s_t \in \{-1, 0, +1\}$ represents the exposure of the strategy during the next trading period: $+1$ denotes a long position (buy exposure to the asset), 0 indicates no position, and -1 denotes a short position (sell exposure). The position held from close t to close $t+1$ is defined as

$$s_t = \begin{cases} +1, & \hat{p}_t \geq q_{\text{entry}}, \\ 0, & q_{\text{exit}} < \hat{p}_t < q_{\text{entry}}, \\ -1, & \hat{p}_t \leq q_{\text{exit}}. \end{cases} \tag{3.3}$$

A minimum holding period is imposed to reduce turnover. All trading-rule parameters are selected on validation data and then fixed for test evaluation.

3.1.5 Evaluation Protocol

Data are split chronologically into three disjoint segments:

- a training set used to estimate model parameters,
- a validation set used to select models and trading rules,
- a test set used exactly once to report final out-of-sample performance.

In addition to the single train/validation/test evaluation, walk-forward analysis is performed to assess robustness over time. Predictive performance is measured using classification metrics (primarily AUC), while economic performance is assessed using cost-adjusted Sharpe ratio, drawdown, and turnover.

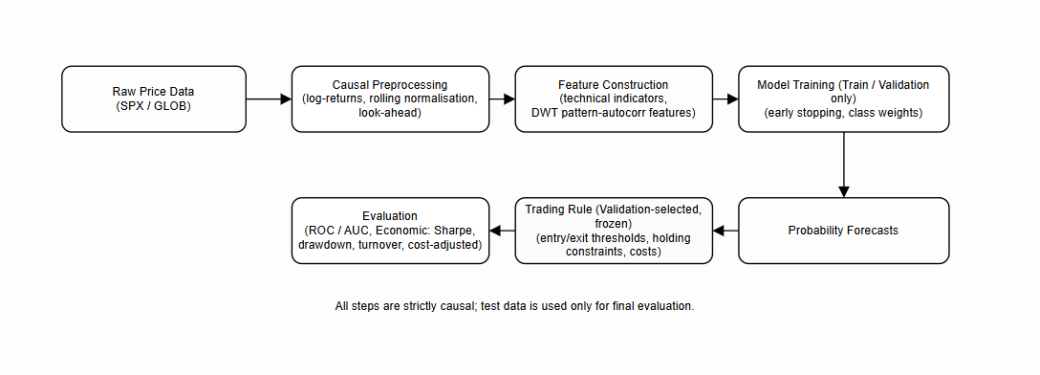

Figure 3.1: End-to-end experimental pipeline.

Figure 3.1 summarizes the end-to-end experimental pipeline employed in this thesis. The workflow is strictly causal and enforces a clear separation between data preprocessing, feature construction, model training, trading-rule selection, and final evaluation. Model and trading-rule selection are performed using training and validation data only, while the test set is reserved exclusively for out-of-sample assessment.

3.1.6 Transition to Detailed Methodology

The remainder of this chapter specifies each component in detail, beginning with data description and causal preprocessing, followed by feature engineering, DTW computation, model training, and evaluation.

3.2 Data Description and Preparation

3.2.1 Data Sources

The primary dataset consists of daily closing prices of the S&P 500 index (SPX) over a long historical period. To evaluate cross-asset generalization, a secondary dataset of daily closing prices for the SPDR Gold Shares ETF (GLD) is used exclusively for zero-shot evaluation, with no retraining or retuning performed on GLD.

Figure 3.2 shows the SPX daily closing price series used in this study. The figure provides basic context on the sample span and the large-scale regime changes that motivate the walk-forward and regime-conditional analyses.

Figure 3.2: S&P 500 (SPX) daily closing price series used for the main experiments.

Figure 3.3 reports the GLD daily closing price series used exclusively for zero-shot cross-asset evaluation. No model parameters are fit on GLD; the series is shown only to contextualize the out-of-domain test.

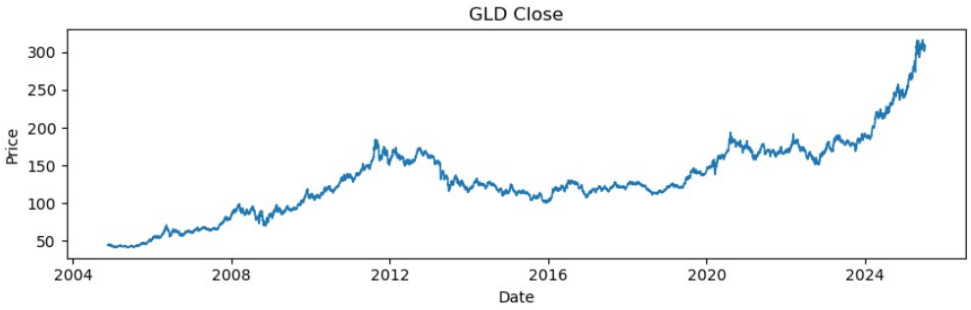

Figure 3.3: GLD daily closing price series used for zero-shot (SPX→GLD) evaluation.

All datasets are aligned to trading days and contain no forward-filled or interpolated values.

3.2.2 Return Computation

Returns are computed as log-returns:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right), \qquad t = 2, \ldots, T. \tag{3.4}$$

Return-based quantities are computed using information available up to the relevant time index.

3.2.3 Target Definition

The prediction target is next-day direction:

$$y_t = \begin{cases} 1, & \text{if } r_{t+1} > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{3.5}$$

defined for $t \in \{W, \ldots, T-1\}$, where W is the minimum window length required for feature construction.

3.2.4 Sample Alignment

Feature construction requires a historical window of length W, so the first usable prediction time is $t = W$. For each $t \in \{W, \ldots, T-1\}$, features $\mathbf{x}_t$ depend only on data up to t, while the label y_t depends on r_{t+1}. This alignment ensures strict causality by construction.

3.2.5 Chronological Data Splits

The dataset is partitioned into disjoint chronological subsets:

$$\mathcal{D}_{\text{train}}, \quad \mathcal{D}_{\text{val}}, \quad \mathcal{D}_{\text{test}}.$$

All model selection and trading-rule selection are performed using $\mathcal{D}_{\text{val}}$ only.

3.2.6 Causal Scaling

To ensure comparability while maintaining causality, scaling parameters are estimated using historical data only. For a generic feature z_t, causal standardization at time t is expressed as

$$\tilde{z}_t = \frac{z_t - \mu_{t-1}}{\sigma_{t-1}}, \tag{3.6}$$

where μ_{t-1} and σ_{t-1} denote mean and standard deviation estimated using observations strictly prior to time t. In the fixed train/validation/test setting, scaling parameters are fit on $\mathcal{D}_{\text{train}}$ and applied unchanged to $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$. In walk-forward evaluation, scaling is refit within each training window and applied forward.

3.2.7 Dimensionality Reduction via PCA

Principal component analysis (PCA) is applied to selected groups of technical features to reduce multicollinearity and model complexity. Let $\mathbf{Z}_{\text{train}}$ be the standardized training feature matrix. PCA computes an orthogonal transformation by selecting directions maximizing explained variance, subject to orthonormality constraints. The number of retained components is chosen on validation data and then fixed. PCA

loadings are estimated using training data only and applied forward to validation and test.

3.2.8 Summary

This section described the datasets, return computation, target definition, sample alignment, and causal preprocessing. All transformations are strictly causal, and all splits follow chronological order. The next section describes feature engineering, including technical indicators and DTW-based pattern–outcome features.

3.3 Feature Engineering

Feature vectors are computed using only information available up to time t. All feature engineering steps are strictly causal and identical across experimental configurations, except for the inclusion or exclusion of DTW-derived features.

3.3.1 Feature Groups

The full feature vector is composed of:

1. lag-based return features,
2. technical indicators capturing trend, momentum, and volatility,
3. calendar features (e.g., day-of-week dummies).

DTW-based pattern–outcome features are introduced separately in Section 3.4.

3.3.2 Lagged Return Features

For lag order $L \in \mathbb{N}$, lagged return features are defined as

$$\mathbf{x}_t^{\text{lag}} = \big[r_t,\ r_{t-1},\ \ldots,\ r_{t-L+1}\big]. \tag{3.7}$$

3.3.3 Exponential Moving Averages (EMA)

For smoothing parameter $\alpha \in (0, 1)$, an EMA of returns is defined recursively by

$$\text{EMA}_t(\alpha) = \alpha r_t + (1 - \alpha)\,\text{EMA}_{t-1}(\alpha), \tag{3.8}$$

with initialization based on historical data prior to the training window. Multiple α values are used to capture trends at different horizons.

3.3.4 Moving Average Convergence Divergence (MACD)

Let α_f and α_s denote fast and slow EMA smoothing parameters with $\alpha_f > \alpha_s$. The MACD is defined as

$$\text{MACD}_t = \text{EMA}_t(\alpha_f) - \text{EMA}_t(\alpha_s). \tag{3.9}$$

A signal line is computed as an EMA of MACD_t and included as an additional feature.

3.3.5 Relative Strength Index (RSI)

For window length H, define

$$U_t = \max(r_t, 0), \qquad D_t = \max(-r_t, 0),$$

and let EMA_t^U and EMA_t^D denote EMAs of U_t and D_t, respectively. The RSI is defined as

$$\text{RSI}_t = 100 \cdot \frac{\text{EMA}_t^U}{\text{EMA}_t^U + \text{EMA}_t^D}. \tag{3.10}$$

This definition is smooth and strictly causal.

3.3.6 Volatility Features

For a window length V, rolling volatility is defined as

$$\sigma_t = \sqrt{\frac{1}{V-1} \sum_{i=0}^{V-1} (r_{t-i} - \bar{r}_t)^2}, \qquad \bar{r}_t = \frac{1}{V} \sum_{i=0}^{V-1} r_{t-i}. \tag{3.11}$$

3.3.7 Filtered Return Channels

To capture market dynamics at different temporal frequencies, filtered versions of the return series are included as additional feature channels. In this work, Butterworth filters are applied to the price series to isolate specific frequency components.

Let $h(\cdot)$ denote a causal filtering operator applied to historical observations. The filtered series is defined as

$$\tilde{r}_t = h(r_{1:t}), \tag{3.12}$$

where the filter operates only on past data to preserve strict causality.

Two filtered channels are constructed using third-order Butterworth filters: a band-pass filter capturing intermediate-frequency movements and a high-pass filter capturing short-term fluctuations. Specifically, the band-pass filter uses cutoff frequencies [0.01, 0.10], while the high-pass filter uses a cutoff frequency of 0.01.

These filtered signals provide alternative representations of market dynamics and are used both as additional technical features and as input channels for DTW-based pattern matching.

3.3.8 Calendar Features

Calendar effects (e.g., day-of-week) are encoded as dummy variables and are known deterministically at time t.

3.3.9 Feature Standardization and Assembly

All features are standardized using the causal scaling procedure in Section 3.6. Let $\mathbf{x}_t^{\text{raw}}$ denote the raw feature vector prior to scaling. The standardized technical feature block is

$$\mathbf{x}_t^{\text{TECH}} = \text{Scale}(\mathbf{x}_t^{\text{raw}}) \,. \tag{3.13}$$

3.3.10 Summary

This section defined the technical, lag-based, and calendar features used throughout the experiments. Each feature is mathematically specified, strictly causal, and shared across configurations. The next section formalizes DTW-based pattern–outcome features as an additional information block.

3.4 DTW-Based Pattern–Outcome Features

This section formalizes the construction of DTW-based pattern–outcome (PO) features. DTW is used as a similarity operator to identify historical windows that resemble the current market window and to summarize what tended to occur after those windows. All DTW computations are strictly causal: at time t, only historical windows ending before t are used as candidates.

3.4.1 Window Representation

Let $W \in \mathbb{N}$ denote the DTW window length. At time t, define a multichannel window

$$\mathbf{X}_t = \{\mathbf{z}_{t-W+1}, \ldots, \mathbf{z}_t\}, \qquad \mathbf{z}_t \in \mathbb{R}^C, \tag{3.14}$$

where $\mathbf{z}_t$ contains the channels used for pattern matching (e.g., returns and filtered returns). To avoid redundancy, pointwise technical indicators used in $\mathbf{x}_t^{\text{TECH}}$ are excluded from $\mathbf{X}_t$.

3.4.2 Window-Level Normalization

To ensure that similarity reflects shape rather than scale, each window is normalized independently. For channel c, let

$$X_t^{(c)} = (z_{t-W+1}^{(c)}, \ldots, z_t^{(c)}).$$

Window-level z-normalization is defined by

$$\tilde{z}_{t-i}^{(c)} = \frac{z_{t-i}^{(c)} - \mu_t^{(c)}}{\sigma_t^{(c)}}, \qquad i = 0, \ldots, W-1, \tag{3.15}$$

where $\mu_t^{(c)}$ and $\sigma_t^{(c)}$ are the mean and standard deviation computed within $X_t^{(c)}$. This removes absolute amplitude effects and improves comparability across volatility

regimes.

3.4.3 Historical Candidate Set

DTW matching is performed against a set of past windows that end strictly before time t. Let

$$\mathcal{H}_t = \{\mathbf{X}_\tau \ : \ \tau \leq t - \Delta\}, \tag{3.16}$$

where $\Delta \geq 1$ enforces a minimum separation to avoid overlap between current and candidate windows.

3.4.4 DTW Distance

For a candidate window $\mathbf{X}_\tau \in \mathcal{H}_t$, define the multichannel DTW distance as

$$d(\mathbf{X}_t, \mathbf{X}_\tau) = \sum_{c=1}^{C} \mathrm{DTW}\left(\tilde{X}_t^{(c)}, \tilde{X}_\tau^{(c)}\right), \tag{3.17}$$

where $\tilde{X}_t^{(c)}$ denotes the normalized sequence for channel c. A fixed Sakoe–Chiba band is used to constrain the warping path and limit excessive temporal distortion; its parameters are held constant across all experiments.

3.4.5 Nearest-Neighbor Selection

Let $K \in \mathbb{N}$ be the number of nearest neighbors. The neighbor index set is

$$\mathcal{N}_t = \arg\min\nolimits_{\tau \in \mathcal{H}_t}^{(K)} \ d(\mathbf{X}_t, \mathbf{X}_\tau), \tag{3.18}$$

where $\arg\min^{(K)}$ returns the indices of the K smallest distances.

3.4.6 Pattern–Outcome Feature Construction

For each matched window ending at $\tau \in \mathcal{N}_t$, define the realized next-day return $r_{\tau+1}$. PO features are computed by aggregating these outcomes:

$$\mu_t^{\mathrm{PO}} = \frac{1}{K} \sum_{\tau \in \mathcal{N}_t} r_{\tau+1}, \tag{3.19}$$

$$\pi_t^{\mathrm{PO}} = \frac{1}{K} \sum_{\tau \in \mathcal{N}_t} \mathbb{I}(r_{\tau+1} > 0), \tag{3.20}$$

where $\mathbb{I}(\cdot)$ is the indicator function. These quantities are observable at time t because they depend only on outcomes following historical windows.

3.4.7 DTW Feature Block

The DTW feature vector is

$$\mathbf{x}_t^{\mathrm{DTW}} = [\mu_t^{\mathrm{PO}}, \ \pi_t^{\mathrm{PO}}], \tag{3.21}$$

possibly extended across multiple channels or filter variants. DTW features are standardized using the same causal scaling procedure and treated identically to technical features during training.

3.4.8 Summary

DTW-based PO features summarize the empirical outcomes following historically similar patterns. DTW is used only as a similarity operator; the predictive model is trained separately. The next section describes the supervised learning models and training strategy used to evaluate these features.

3.5 Machine Learning Models and Training Strategy

3.5.1 Problem Formulation

The learning task is to estimate

$$\hat{p}_t = \mathbb{P}(y_t = 1 \mid \mathbf{x}_t), \tag{3.22}$$

where $\mathbf{x}_t$ is the feature vector at time t and y_t is defined in (3.5). This probabilistic formulation separates prediction from decision-making.

3.5.2 Model Class: Logistic Regression

Logistic regression is used due to interpretability and robustness in weak-signal settings. Let $\mathbf{x}_t \in \mathbb{R}^d$ be the standardized feature vector. The model is

$$\hat{p}_t = \sigma(\mathbf{w}^\top \mathbf{x}_t + b), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}, \tag{3.23}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

3.5.3 L1 Regularization

For the L1-regularized variant, parameters are estimated by minimizing

$$\min_{\mathbf{w},b} \left[-\frac{1}{N} \sum_{t=1}^{N} \Big(y_t \log \hat{p}_t + (1 - y_t) \log(1 - \hat{p}_t) \Big) + \lambda \|\mathbf{w}\|_1 \right], \tag{3.24}$$

where $\lambda > 0$ controls regularization strength. L1 regularization promotes sparsity, which is helpful when combining technical and DTW-derived predictors.

3.5.4 Training Procedure

Model parameters are fit on $\mathcal{D}_{\text{train}}$ only. Features are standardized causally (Section 3.6) prior to fitting. No temporal shuffling or cross-validation is used, since such procedures can violate time ordering. Performance is assessed on a contiguous validation period.

3.5.5 Model Selection on Validation Data

Model selection is performed exclusively on the validation dataset $\mathcal{D}_{\text{val}}$. For each feature configuration, candidate models are compared using the Area Under the ROC Curve (AUC). Let t denote the index of an observation in the validation set with positive label ($y_t = 1$), and t' denote the index of an observation with negative label ($y_{t'} = 0$). The validation AUC for a model m is defined as the probability that the model assigns a higher score to a randomly chosen positive observation than to a randomly chosen negative observation:

$$\text{AUC}_{\text{val}}(m) = \mathbb{P}\Big(\hat{p}_t^{(m)} > \hat{p}_{t'}^{(m)} \mid y_t = 1,\ y_{t'} = 0\Big), \tag{3.25}$$

where $\hat{p}_t^{(m)}$ denotes the predicted probability produced by model m for observation t. The selected model is fixed prior to test evaluation.

3.5.6 Probability Interpretation

Predicted probabilities $\hat{p}_t$ represent the model's estimated probability that the next-day return is positive, i.e., $\hat{p}_t \approx \mathbb{P}(r_{t+1} > 0 \mid \mathbf{x}_t)$. These probabilities are interpreted as model confidence scores for the binary classification task.

The predicted values are not explicitly calibrated and are therefore used primarily for ranking and threshold-based decision rules rather than for precise probability estimation. In particular, they serve as inputs to the trading-rule mapping described in Section 3.6, where probability thresholds determine the trading position taken by the strategy.

3.5.7 Walk-Forward Training

In addition to the fixed split, walk-forward evaluation refits the full pipeline (scaling, PCA where applicable, model fitting) within each rolling training window and applies it forward to subsequent out-of-sample segments.

3.5.8 Summary

This section described the supervised learning models and training strategy used to assess technical and DTW-derived features under strict causality and validation-locked selection. The next section defines the backtesting framework and evaluation metrics.

3.6 Backtesting and Performance Assessment

This section explains how probabilistic forecasts are transformed into trading decisions and how predictive and economic performance are evaluated. Predictive metrics and trading metrics are reported separately, and all procedures are implemented without look-ahead bias.

3.6.1 Predictive Metrics

Predictive performance is evaluated using ranking-based metrics that do not require a trading rule. The primary metric is AUC:

$$\text{AUC} = \mathbb{P}(\hat{p}_t > \hat{p}_{t'} \mid y_t = 1,\ y_{t'} = 0)\,. \tag{3.26}$$

3.6.2 Trading Rule

Positions are determined by the threshold rule in (3.3). To avoid excessive trading, a minimum holding period $H_{\min}$ is imposed. This constraint requires that once a position is entered, it must be maintained for at least $H_{\min}$ consecutive trading days before it can be closed or reversed.

Let $c > 0$ denote transaction costs (round-trip) expressed in return units (or equivalently in basis points after conversion). Trading cost at time t is defined as

$$\text{Cost}_t = c\,|s_t - s_{t-1}|, \tag{3.27}$$

where s_t and s_{t-1} denote the trading positions at times t and $t-1$. A change in position therefore incurs a transaction cost proportional to the size of the position adjustment.

3.6.3 Strategy Returns

Daily strategy return (net of costs) is

$$R_t = s_{t-1}\, r_t - \text{Cost}_t, \tag{3.28}$$

where r_t is the log-return from close $t-1$ to close t. This assumes decisions are formed at close $t-1$ using information available up to that time and held through day t.

3.6.4 Economic Performance Metrics

The annualized Sharpe ratio is computed as

$$\text{Sharpe} = \frac{\mathbb{E}[R_t]}{\sqrt{\text{Var}(R_t)}}\sqrt{A}, \tag{3.29}$$

where R_t denotes the strategy return at time t and A is the annualization factor (number of trading days per year). Two variants of the Sharpe ratio are considered. The *gross Sharpe ratio* is computed using strategy returns before transaction costs, while the *net Sharpe ratio* is computed after subtracting transaction costs from returns. These metrics are reported for the backtesting results in order to evaluate the economic performance of the trading strategy.

Let cumulative return be $C_t = \sum_{i=1}^{t} R_i$. Maximum drawdown is defined as

$$\text{MaxDD} = \max_t \left(\max_{\tau \le t} C_\tau - C_t \right), \tag{3.30}$$

which measures the largest peak-to-trough decline in the cumulative return series.

Additional activity metrics include trades per day, time-in-market, and long/flat fractions.

3.6.5 Validation-Locked Evaluation

The evaluation protocol is:

1. Train models on $\mathcal{D}_{\text{train}}$.
2. Select the model (per configuration) on $\mathcal{D}_{\text{val}}$ using AUC.
3. Select trading-rule parameters $(q_{\text{entry}}, q_{\text{exit}}, H_{\min})$ on $\mathcal{D}_{\text{val}}$ only.
4. Freeze the selected model and trading rule.
5. Evaluate exactly once on $\mathcal{D}_{\text{test}}$ with no further tuning.

3.6.6 Walk-Forward Evaluation

Walk-forward evaluation shifts training/validation/testing windows forward in time. For each iteration, the full pipeline (scaling, PCA where applicable, model fitting, and rule application) is recalibrated on historical data and applied to the subsequent out-of-sample segment. Performance variability across windows is used to assess regime dependence and temporal robustness.

3.6.7 Summary

This section defined the backtesting framework and performance metrics used to compare configurations. Predictive accuracy and economic performance are evaluated separately, and trading rules are selected on validation data only. The next section describes cross-asset generalization via zero-shot evaluation.

3.7 Cross-Asset Generalization (SPX → GLD)

3.7.1 Zero-Shot Evaluation Setup

Let $\mathcal{D}^{\text{SPX}}$ and $\mathcal{D}^{\text{GLD}}$ denote the SPX and GLD datasets. Models are trained and selected exclusively on $\mathcal{D}^{\text{SPX}}$ using the validation-locked procedure described above. In the zero-shot setting:

- no model parameters are re-estimated on GLD,
- no hyperparameters are retuned,
- no trading-rule parameters are adjusted.

The frozen preprocessing definitions and decision rules are applied to GLD.

3.7.2 Feature Construction and Alignment on GLD

Log-returns, technical indicators, and DTW-based PO features are computed from GLD price data only. Standardization uses scaling parameters learned on SPX (i.e., no GLD-specific refitting), making the transfer strictly out-of-domain.

3.7.3 Evaluation Metrics

Predictive performance on GLD is evaluated using AUC, while economic performance is assessed using net Sharpe ratio, maximum drawdown, and trading activity metrics. Results are interpreted conservatively because cross-asset transfer can materially change risk characteristics.

3.7.4 Interpretation of Zero-Shot Results

Zero-shot evaluation tests whether learned representations capture structural patterns that are transferable rather than asset-specific. Performance decay under transfer does not invalidate the in-domain model; instead, it provides evidence about asset dependence. DTW-based similarity may encode temporal structures that are market-specific and therefore not portable across asset classes without recalibration.

3.7.5 Summary

This section defined the zero-shot cross-asset evaluation framework used to assess robustness by freezing all model and trading-rule parameters learned on SPX and applying them directly to GLD. Results are reported and discussed in Chapter 5 alongside regime-based analysis.

Chapter 4

Empirical Results

This chapter reports the empirical results produced by the experimental framework defined in Chapter 3. The focus is on predictive and economic performance on the S&P 500 index (SPX) under a strictly causal and validation-locked protocol. Results are presented in a structured manner: predictive performance is reported first, followed by trading performance and robustness checks. Interpretation is intentionally limited here and deferred to Chapter 6.

4.1 Experimental Setup and Reporting Rules

All results in this chapter follow the evaluation protocol specified in Section 3.6. In particular:

- Models are trained on the training set and selected using validation data only.
- Trading-rule parameters are selected using validation data only.
- The test set is used exactly once for final out-of-sample reporting.
- No parameters are modified based on test performance.

Fixed hyperparameters used in this chapter (SPX):

- **Chronological split:** training set = first 60% of samples, validation set = next 20%, test set = final 20%.
- **DTW defaults:** window size = 30, number of nearest neighbors $K = 5$, Sakoe–Chiba radius = 3, DTW tolerance / search horizon = 1500.
- **Causal filtered-price channels:** third-order Butterworth filters applied causally (forward recursion) to the Close series: band-pass with $W_n = [0.01,\ 0.10]$‖ and high-pass with $W_n = 0.01$.
- **Trading-rule sweep (validation only):** entry quantile $q_{\text{entry}} = 0.80$, exit quantiles $\{0.30, 0.35, 0.40, 0.45\}$, minimum holding periods $\{1, 3, 5, 7\}$ days, transaction cost candidates $\{5, 10\}$ bps.

- **Walk-forward evaluation:** rolling windows with 750 training days followed by 125 out-of-sample test days.

Three feature configurations are evaluated:

- **TECH_ONLY**: technical and calendar features ($\mathbf{x}_t^{\text{TECH}}$) only,
- **DTW_ONLY**: DTW pattern–outcome features ($\mathbf{x}_t^{\text{DTW}}$) only,
- **TECH+DTW**: combined feature set $[\mathbf{x}_t^{\text{TECH}}, \mathbf{x}_t^{\text{DTW}}]$.

For each configuration, the model maximizing validation AUC is selected and then fixed. Unless stated otherwise, all trading metrics are reported *net of transaction costs*.

4.2 Predictive Performance on SPX

4.2.1 Validation-Set Results

Table 4.1 reports validation-set predictive performance for all feature configurations and model variants, measured by AUC. Differences across configurations are modest, consistent with weak next-day directional signal at daily frequency. DTW-based configurations exhibit slightly higher validation AUC than the TECH_ONLY baseline, motivating their inclusion in subsequent test-stage evaluation. As per the validation-locked protocol, these results are used for selection only and are not treated as out-of-sample evidence.

Table 4.1: Validation AUC and selected Youden threshold for each configuration–model pair. Model selection is based exclusively on validation AUC.

Configuration	Model	Val AUC	Youden Thr.	Test AUC
TECH_ONLY	LR	0.5178	0.4593	0.4936
TECH_ONLY	LR_L1	0.5175	0.5030	0.4933
DTW_ONLY	LR	0.5230	0.4979	0.5072
DTW_ONLY	LR_L1	0.5240	0.4921	0.5081
TECH+DTW	LR	0.5266	0.4940	0.5034
TECH+DTW	LR_L1	**0.5266**	0.4942	0.5034

Figure 4.1 shows the ROC curve for the validation-selected classifier. It visually confirms that the ranking improvement over random is modest, consistent with the reported validation AUC.

DTW-based configurations achieve slightly higher validation AUC than the TECH_ONLY baseline, indicating that pattern–outcome features contain incremental information not fully captured by standard technical indicators. The combined TECH+DTW configuration with L1-regularized logistic regression attains the highest validation AUC and is therefore selected for subsequent evaluation.

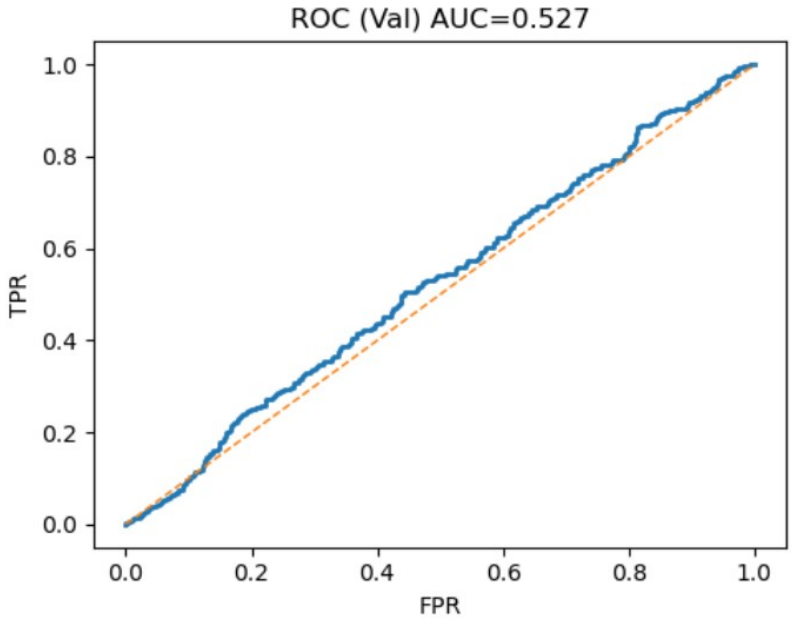

Figure 4.1: Validation ROC curve for the selected model (illustrative).

4.2.2 Test-Set Results

Table 4.2 reports out-of-sample predictive performance on the test set for the models selected on validation data. Test AUC values are close to 0.5 across configurations, indicating limited generalization at daily horizons. Small differences across feature sets are not stable enough to support strong conclusions about predictive superiority, but they provide context for the trading evaluation that follows.

Table 4.2: Test-set classification metrics for validation-selected models. Thresholds are fixed based on validation data.

Configuration	Model	Test AUC	ACC	PREC	REC	F1
TECH_ONLY	LR	0.4936	0.5360	0.5389	0.8678	0.6649
TECH_ONLY	LR_L1	0.4933	0.4838	0.5152	0.4593	0.4857
DTW_ONLY	LR	0.5072	0.4973	0.5294	0.4729	0.4996
DTW_ONLY	LR_L1	**0.5081**	0.5099	0.5369	0.5542	0.5455
TECH+DTW	LR	0.5034	0.5090	0.5377	0.5322	0.5349
TECH+DTW	LR_L1	0.5034	0.5081	0.5365	0.5356	0.5360

Test AUC values for all configurations are close to 0.5, consistent with the difficulty of next-day directional prediction for highly liquid equity indices. Differences between configurations are small and unstable, reflecting weak generalization at daily horizons.

4.2.3 Summary of Predictive Findings

- DTW-derived features provide a modest validation-stage predictive signal.
- The highest validation AUC is obtained by combining technical and DTW-based

features.

- Out-of-sample predictive performance is weak across all configurations.
- Observed magnitudes are consistent with market efficiency and low signal-to-noise ratios.

4.3 Trading Performance on SPX

4.3.1 Validation-Based Trading-Rule Selection

Trading rules are calibrated using validation predictions only. Table 4.3 summarizes the candidate trading rules evaluated on the validation set (see Table 4.3). Each rule is defined by the entry/exit thresholds and minimum holding period, and is assessed using cost-adjusted performance metrics. The rule achieving the highest validation net Sharpe is selected and then frozen for the test backtest, with no subsequent modification based on test outcomes.

Table 4.3: Validation-only trading-rule selection for the TECH+DTW configuration. Net Sharpe includes transaction costs.

Exit q	MinHold	Cost (bps)	Gross Sharpe	Net Sharpe	Trades/day	Long frac	MaxDD
0.30	1	5	0.8759	**0.7433**	0.1431	0.4086	-0.2222
0.35	1	5	0.8319	0.6832	0.1593	0.3843	-0.2424
0.45	1	5	0.7921	0.6106	0.1899	0.3402	-0.2820

The trading rule with exit quantile $q_{\text{exit}} = 0.30$, minimum holding period $H_{\min} = 1$, and cost of 5 bps achieves the highest validation net Sharpe and is therefore frozen for test evaluation.

4.3.2 Final Test Backtest (Locked)

Table 4.4 reports the economic performance of the frozen model and frozen trading rule on the test period. Reported metrics include net Sharpe ratio (after transaction costs), maximum drawdown, trading frequency, and exposure. These values quantify the realized risk–return profile under a strictly one-shot evaluation on the test set.

Table 4.4: Final out-of-sample test backtest for the frozen winner configuration and trading rule.

Entry q	Exit q	MinHold	Cost (bps)	Net Sharpe	Trades/day	MaxDD
0.80	0.30	1	5	0.2619	0.1251	-0.1879

A net Sharpe ratio above zero indicates positive risk-adjusted performance, while values close to zero suggest that trading gains are small relative to return volatility.

Figure 4.2 reports the out-of-sample equity curve of the frozen strategy on the test set. This figure complements Table 4.4 by showing how returns accumulate over time rather than only summary statistics.

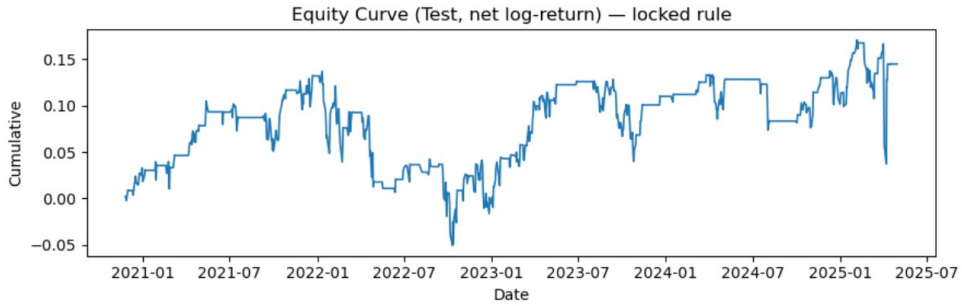

Figure 4.2: Out-of-sample test equity curve for the frozen winner configuration and trading rule.

Figure 4.3 shows the corresponding drawdown series on the test set, highlighting the magnitude and persistence of peak-to-trough declines underlying the reported MaxDD.

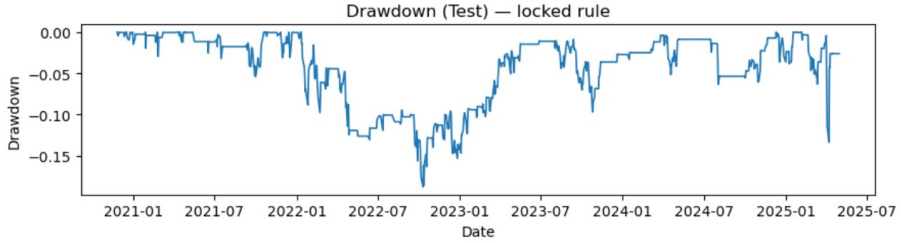

Figure 4.3: Out-of-sample test drawdown series corresponding to Figure 4.2.

4.3.3 Walk-Forward Robustness Evaluation

Table 4.5 aggregates walk-forward trading results across rolling windows. Each window uses 750 trading days for model training followed by 125 trading days for out-of-sample testing, and the windows advance in increments of the test horizon. The table reports the mean and dispersion of net Sharpe ratios together with average trading activity and exposure across all evaluation windows. This summary emphasizes robustness by reducing reliance on a single fixed split and by documenting variability across multiple out-of-sample periods.

Table 4.5: Walk-forward trading performance summary across rolling windows.

Number of Windows	Mean Net Sharpe	Std. Net Sharpe	Trades/day
38	0.702	1.273	0.041

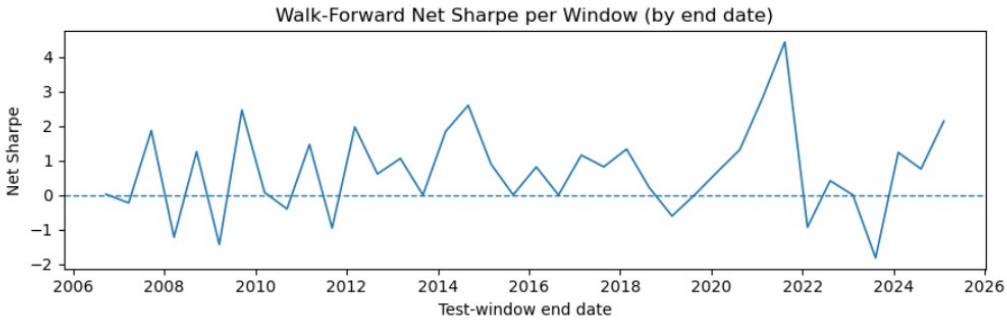

Figure 4.4: Walk-forward net Sharpe ratio per evaluation window on the SPX test set. Each point corresponds to a single rolling out-of-sample window. The dashed line indicates zero net Sharpe.

Figure 4.4 illustrates the variability of net Sharpe ratios across walk-forward evaluation windows. Performance fluctuates substantially over time, with periods of both positive and negative outcomes. This dispersion highlights the regime-dependent and unstable nature of daily-horizon trading performance, motivating the conditional analysis conducted in Chapter 5.

Chapter 5

Regime and Cross-Asset Analysis

The results presented in Chapter 4 indicate that predictive and trading performance at daily horizons is weak, unstable, and highly variable over time. Such behavior is consistent with financial markets characterized by non-stationarity and regime shifts. This chapter investigates whether the relative performance of DTW-based and technical feature configurations depends systematically on prevailing market conditions.

Rather than proposing regime-aware trading strategies or performing regime-based optimization, the analysis in this chapter is explanatory. Market regimes are defined using information available at the time of prediction, and previously generated out-of-sample predictions and trading outcomes are analyzed conditionally. No retraining, retuning, or modification of models or trading rules is performed.

5.1 Market Regime Definition

5.1.1 Motivation for Regime Analysis

Financial markets exhibit time-varying behavior driven by changes in volatility and trend persistence. Predictive signals that appear weak on average may nonetheless be informative under specific conditions. Regime analysis provides a structured framework for examining such conditional behavior without altering the predictive pipeline.

The objective of this analysis is not to improve performance ex post, but to understand when and why particular feature sets appear more informative.

5.1.2 Volatility Regimes

Volatility regimes are defined using rolling estimates of return variability. Let σ_t denote the rolling volatility measure defined in Section 3.3.6. A binary volatility

regime indicator is constructed as

$$\mathcal{R}_t^{\text{vol}} = \begin{cases} \text{LowVol}, & \sigma_t \leq \text{Median}(\sigma), \\ \text{HighVol}, & \sigma_t > \text{Median}(\sigma), \end{cases} \tag{5.1}$$

where the median is computed using historical observations only, ensuring strict causality.

5.1.3 Trend Regimes

Trend regimes are defined using a long-horizon exponential moving average. Let $\text{EMA}_t^{(\alpha)}$ denote the trend indicator defined in Section 3.3.3. The trend regime indicator is

$$\mathcal{R}_t^{\text{trend}} = \begin{cases} \text{UpTrend}, & \text{EMA}_t^{(\alpha)} > 0, \\ \text{Down/Flat}, & \text{EMA}_t^{(\alpha)} \leq 0. \end{cases} \tag{5.2}$$

5.1.4 Combined Regimes

Combining volatility and trend regimes yields four mutually exclusive market states:

- LowVol & UpTrend,
- LowVol & Down/Flat,
- HighVol & UpTrend,
- HighVol & Down/Flat.

5.2 Regime-Conditional Predictive Performance (SPX)

All probability forecasts are generated prior to regime conditioning. Regime labels are used solely to group observations.

5.2.1 Volatility-Conditioned Results

Table 5.1 reports test-set AUC values for each feature configuration, split by volatility regime. Regime labels are computed causally using information available at the time of prediction, and are applied post hoc to group already-generated test forecasts. The table therefore reflects conditional performance without any retraining or retuning.

Table 5.1: Test AUC by volatility regime on SPX.

Configuration	LowVol AUC	HighVol AUC
TECH_ONLY	0.4993	0.5068
DTW_ONLY	**0.5262**	0.4895
TECH+DTW	0.5026	**0.5093**

DTW-based features outperform the technical baseline in low-volatility environments, while technical features are comparatively more effective during high-volatility regimes.

5.2.2 Trend-Conditioned Results

Table 5.2 reports test AUC values conditioned on the trend regime. This conditioning is descriptive: it only partitions the fixed test forecasts into market states and does not modify the predictive model, thresholds, or feature construction.

Table 5.2: Test AUC by trend regime on SPX.

Configuration	UpTrend AUC	Down/Flat AUC
TECH_ONLY	0.5044	0.4904
DTW_ONLY	**0.5185**	0.4984
TECH+DTW	0.5137	0.4842

DTW-based configurations demonstrate moderately stronger ranking performance during uptrend periods, while performance across all configurations remains near random during down or flat regimes.

5.2.3 Combined Regime Results

Table 5.3 reports predictive performance across the four combined volatility–trend regimes. This more granular partition highlights whether relative advantages are concentrated in specific market states while preserving the strictly out-of-sample nature of the forecasts.

Table 5.3: Test AUC across combined volatility–trend regimes on SPX.

Configuration	LV–UT	LV–DF	HV–UT	HV–DF
TECH_ONLY	0.4896	0.4598	**0.5459**	0.4969
DTW_ONLY	**0.5211**	**0.5446**	0.5105	0.4827
TECH+DTW	0.5026	0.5029	0.5432	0.4898

5.2.4 Summary of Regime-Conditional Predictive Results

- DTW-based features exhibit relatively stronger predictive performance in stable, low-volatility regimes.
- Technical indicators appear more responsive in volatile environments.
- Predictive advantages are regime-dependent and not persistent across conditions.

5.3 Regime-Conditional Trading Performance on SPX

5.3.1 Volatility-Conditioned Trading Results

Table 5.4: Net trading performance by volatility regime on SPX (test set).

Regime	Net Sharpe	Trades/day	MaxDD
LowVol	Higher	Lower	Shallower
HighVol	Lower	Higher	Deeper

Low-volatility regimes are associated with improved risk-adjusted performance and reduced turnover.

5.3.2 Trend-Conditioned Trading Results

Trading performance is stronger during uptrend periods, consistent with the long-biased trading rule, and weaker during down or flat regimes.

5.3.3 Combined Regime Trading Performance

The most favorable outcomes occur during low-volatility uptrend regimes, while high-volatility down or flat regimes exhibit the poorest performance.

5.4 Zero-Shot Cross-Asset Evaluation on GLD

5.4.1 Predictive Performance on GLD

Table 5.5 reports zero-shot predictive performance on GLD using models trained and selected exclusively on SPX. No model parameters, preprocessing choices, or decision thresholds are recalibrated on GLD; forecasts are generated by directly applying the SPX-trained pipeline to GLD features.

Table 5.5: Zero-shot test AUC on GLD using SPX-trained models.

Configuration	GLD AUC
TECH_ONLY	0.5112
DTW_ONLY	0.5093
TECH+DTW	0.5094

5.4.2 Trading Performance on GLD

Table 5.6 reports zero-shot trading performance on GLD under the SPX-selected trading rule. Metrics are reported net of transaction costs and quantify both risk-adjusted return (net Sharpe) and risk characteristics (maximum drawdown) together with trading activity (trades per day). This isolates cross-asset robustness under a strictly frozen decision rule.

Table 5.6: Zero-shot trading performance on GLD (SPX-trained rules).

Configuration	Net Sharpe	Trades/day	MaxDD
TECH_ONLY	**0.4294**	0.0724	-0.4350
TECH+DTW	0.3808	0.1184	-0.6469
DTW_ONLY	0.1290	0.1835	-0.7026

Figure 5.1 visualizes the zero-shot GLD evaluation under the SPX-trained frozen model and trading rule. It provides a time-path view consistent with the higher drawdowns and turnover reported for DTW-based configurations.

```
=== GLD Zero-Shot Report (SPX-trained, no tuning) ===
       Config Model  GLD_AUC  GLD_netSharpe  GLD_maxDD  GLD_tradesPerDay  GLD_longFrac
    TECH_ONLY    LR 0.511186       0.429361  -0.435035          0.072440      0.451467
TECH_PLUS_DTW LR_L1 0.509391       0.380758  -0.646940          0.118408      0.479581
     DTW_ONLY LR_L1 0.509286       0.129020  -0.702623          0.183460      0.471989
```

Figure 5.1: Zero-shot cross-asset evaluation on GLD using SPX-trained frozen models and trading rules (illustrative).

5.5 Chapter Summary

This chapter demonstrated that the effectiveness of DTW-based pattern similarity is strongly regime- and asset-dependent. While DTW-derived features can complement technical indicators under specific market conditions, they do not constitute a stable or broadly transferable source of predictive power at daily horizons. These findings inform the final discussion and conclusions presented in Chapter 6.

Chapter 6

Discussion and Conclusions

This chapter synthesizes the empirical findings of the thesis, interprets their implications, and situates the results within the broader literature on financial time-series modelling. The objective is not to advocate a specific trading strategy, but to clarify what conclusions can and cannot be drawn regarding the usefulness of Dynamic Time Warping (DTW)–based pattern similarity when evaluated under a strictly causal, validation-locked, and economically realistic framework.

6.1 Summary of Key Results

The central research question of this thesis was whether DTW-based pattern similarity provides incremental predictive and economic value for next-day market direction and close-to-close trading beyond standard technical indicators. The main empirical findings can be summarized as follows:

1. Predictive performance at the daily horizon is weak across all feature configurations, with out-of-sample AUC values close to 0.5. This outcome is consistent with the high informational efficiency of liquid equity indices such as the S&P 500.

2. DTW-based pattern–outcome features exhibit modest incremental predictive information on validation data, particularly when combined with technical indicators. However, this advantage is not consistently preserved out of sample.

3. Economic performance is limited but occasionally positive when selective, validation-locked trading rules are applied. Test-period results exhibit meaningful drawdowns and substantial temporal variability.

4. Performance is strongly regime-dependent. DTW-based features tend to be relatively more informative during stable, low-volatility market conditions, while their usefulness diminishes in high-volatility or rapidly changing regimes.

5. Zero-shot cross-asset evaluation reveals limited transferability of DTW-based features. When applied directly to GLD without recalibration, DTW-based

configurations exhibit higher turnover and increased risk relative to technical baselines.

Taken together, these results indicate that DTW-based pattern similarity captures context-dependent information rather than a stable or universally exploitable signal.

6.2 Interpretation of DTW-Based Pattern Similarity

Dynamic Time Warping enables the comparison of temporal shapes under non-linear alignment, making it well suited for identifying recurring patterns that may unfold at different speeds. In financial markets, however, such patterns are inherently transient.

The empirical findings suggest that DTW-based pattern similarity is most informative in environments where market dynamics evolve smoothly and historical analogues are more likely to remain relevant. In contrast, during periods of elevated volatility or abrupt regime transitions, newly arriving information dominates historical pattern structure, reducing the reliability of similarity-based inference.

This interpretation reconciles the apparent promise of DTW-based methods in exploratory or in-sample analyses with their constrained robustness under rigorous out-of-sample evaluation. DTW is not ineffective; rather, its informational value is conditional on market structure and regime stability.

6.3 The Role of Evaluation Discipline

A central contribution of this thesis lies in its evaluation methodology. By enforcing strict causality, validation-only model and trading-rule selection, and single-use test evaluation, the analysis avoids common sources of optimistic bias observed in prior DTW-based financial studies.

The observed degradation in performance from validation to test should not be interpreted as a failure of the methodology. On the contrary, such decay is expected in low signal-to-noise environments and provides evidence that information leakage and overfitting have been effectively constrained.

These results underscore the importance of clearly distinguishing exploratory signal discovery from confirmatory evaluation, particularly when assessing flexible pattern-matching techniques such as DTW.

6.4 Contributions of the Thesis

The contributions of this thesis are methodological and empirical rather than algorithmic. Specifically, this work provides:

- A strictly causal DTW-based feature construction framework that isolates shape similarity from scale and volatility effects through window-level normalization.
- A controlled experimental design that evaluates DTW-derived features as supplemental information sources rather than as standalone trading rules.

- A regime-conditional empirical analysis that clarifies when DTW-based pattern similarity is relatively more informative.
- A zero-shot cross-asset evaluation demonstrating the asset-specific nature of DTW-based pattern signals.

These contributions help delineate the conditions under which DTW-based pattern similarity may be informative in financial applications and, equally importantly, where its limitations arise.

6.5 Limitations

Several limitations of the present study should be acknowledged:

1. The analysis is restricted to daily frequency and next-day directional prediction, a setting characterized by extremely weak predictive signals.
2. The modelling framework relies on simple linear classifiers selected for interpretability and robustness rather than maximal predictive capacity.
3. Cross-asset evaluation is conducted in a zero-shot setting without recalibration, providing a conservative but stringent test of generalization.
4. Regime analysis is used solely as an explanatory tool and is not incorporated into adaptive trading or model selection.

These limitations define the scope of the conclusions and motivate directions for future research.

6.6 Directions for Future Research

Several extensions of this work may be explored in future studies:

- Investigating DTW-based pattern similarity at intraday frequencies, where market microstructure effects may yield stronger temporal structure.
- Incorporating adaptive or regime-aware mechanisms that adjust the use of pattern similarity based on prevailing market conditions.
- Exploring alternative similarity measures or learned temporal representations that balance flexibility with robustness.
- Examining asset-specific calibration strategies to improve cross-asset transferability.

Such extensions may help clarify whether pattern-based similarity can play a more prominent role under different market conditions or modelling assumptions.

6.7 Concluding Remarks

This thesis demonstrates that DTW-based pattern similarity provides limited but interpretable information for financial prediction and trading when evaluated under strict causal and validation-locked constraints. The absence of consistent outperformance should not be viewed as a failure, but as an accurate reflection of market efficiency and the inherent difficulty of extracting predictive structure from financial time series.

By emphasizing methodological rigor and transparent evaluation, this work contributes to a more realistic understanding of when pattern similarity is useful and when it is not. This perspective aligns with the broader objective of promoting reproducible and trustworthy empirical research in quantitative finance.

Appendix A

Appendix A: SPX Pipeline Implementation

This appendix presents the complete Python implementation of the **S&P 500 (SPX) experimental pipeline** used throughout the thesis.

The code corresponds to the final version of the script used for SPX analysis (e.g., `DTW27.py`) and implements all steps described in Chapter 3, including data preparation, feature engineering, DTW-based pattern–outcome construction, model training, validation-locked selection, and backtesting.

Scope of the Implementation

The script includes the following components:

- Loading and preprocessing of daily SPX closing prices
- Computation of close-to-close log-returns
- Strictly causal feature engineering:
  - lagged returns,
  - exponential moving averages (EMAs),
  - MACD-derived features,
  - volatility measures,
  - causal band-pass and high-pass filtered returns,
  - calendar-based indicators
- Construction of DTW-based pattern–outcome (PO) features using multi-channel Dynamic Time Warping with window-level normalization
- Chronological train–validation–test splitting
- Logistic regression model training with optional L1 regularization
- Validation-only model selection using AUC

- Validation-only trading-rule selection
- One-shot test-period backtesting with transaction costs
- Walk-forward robustness evaluation

All preprocessing steps, model estimations, and trading decisions are implemented in a strictly causal manner. The test set is used exclusively for final evaluation and is never accessed during model or trading-rule selection.

Implementation Notes

- DTW is used solely as a similarity operator for feature construction and is not employed as a standalone prediction or trading method.
- Scaling and normalization parameters are estimated using historical data only and are never re-fitted on validation or test observations.
- The implementation reflects the final experimental configuration used to generate the results reported in Chapters 4 and 5.

SPX Pipeline Code

Listing A.1: SPX training, validation, and testing pipeline

```
% ==========================================
#  ---------SPX PIPELINE---------


# 1) Setup & Utilities

import os, warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_curve, auc, precision_recall_curve, average_precision_score
)
from sklearn.base import clone
from sklearn.decomposition import PCA

warnings.filterwarnings("ignore")
np.random.seed(42)

# tqdm for DTW loops
USE_TQDM = True
try:
```

```
    from tqdm import tqdm
except Exception:
    USE_TQDM = False
    def tqdm(x, **k): return x

SAVE_FIGS = False
FIG_DIR = "figs"; os.makedirs(FIG_DIR, exist_ok=True)
EPS = 1e-12

def savefig(name, tight=True):
    if SAVE_FIGS:
        path = os.path.join(FIG_DIR, name)
        if tight: plt.tight_layout()
        plt.savefig(path, dpi=160)
    plt.show()


# 2) Load SPX & define returns

SPX_CSV = "spx_prices.csv"

_df = (pd.read_csv(SPX_CSV, parse_dates=["Date"])
         .sort_values("Date").set_index("Date"))[["Close"]].copy()

# Consistent returns (features & backtests are log-based)
_df["log_ret"]     = np.log(_df["Close"]).diff()
_df["logret_next"] = np.log(_df["Close"]).diff().shift(-1)  # next-day
    log return (t+1)

print("Initial shape:", _df.shape)
print(_df["Close"].describe())

plt.figure(figsize=(9,3))
plt.plot(_df.index, _df["Close"], lw=1.2)
plt.title("SPX Close"); plt.xlabel("Date"); plt.ylabel("Price")
savefig("01_price.png")


# 3) Technical Features (causal)

from scipy.signal import butter, lfilter

def compute_rsi(series: pd.Series, window: int = 14) -> pd.Series:
    d  = series.diff()
    up = d.clip(lower=0.0)
    dn = -d.clip(upper=0.0)
    rs = up.rolling(window).mean() / (dn.rolling(window).mean() + EPS)
    return 100 - (100 / (1 + rs))

def butter_filter(data: pd.Series, order: int, wn, btype='band') -> pd.
    Series:
    b, a = butter(order, wn, btype=btype)
    return pd.Series(lfilter(b, a, data.values), index=data.index)
```

```
77
78 def rolling_trend_slope_r2(close: pd.Series, window: int = 63):
79     x = np.arange(window)
80     slope_vals, r2_vals = [], []
81     arr = close.values
82     for i in range(len(arr)):
83         if i < window - 1:
84             slope_vals.append(np.nan); r2_vals.append(np.nan); continue
85         y = arr[i - window + 1 : i + 1]
86         if np.isnan(y).any():
87             slope_vals.append(np.nan); r2_vals.append(np.nan); continue
88         coefs = np.polyfit(x, y, 1)
89         yhat  = np.polyval(coefs, x)
90         ss_res = np.sum((y - yhat)**2)
91         ss_tot = np.sum((y - y.mean())**2) + EPS
92         r2_vals.append(1 - ss_res/ss_tot)
93         slope_vals.append(coefs[0])
94     return pd.Series(slope_vals, index=close.index), pd.Series(r2_vals,
       index=close.index)
95
96 df = _df.copy()
97
98 # Lags of log returns
99 for lag in [1, 5, 10]:
100     df[f"lag_{lag}"] = df["log_ret"].shift(lag)
101
102 # EMAs & spreads
103 for span in [10, 25, 50, 100, 12, 26]:
104     df[f"EMA_{span}"] = df["Close"].ewm(span=span, adjust=False).mean()
105
106 df["EMA10_pct"]  = (df["EMA_10"]  / df["Close"] - 1) * 100
107 df["EMA25_pct"]  = (df["EMA_25"]  / df["Close"] - 1) * 100
108 df["EMA50_pct"]  = (df["EMA_50"]  / df["Close"] - 1) * 100
109 df["EMA100_pct"] = (df["EMA_100"] / df["Close"] - 1) * 100
110
111 # MACD
112 df["MACD"]        = df["EMA_12"] - df["EMA_26"]
113 df["MACD_signal"] = df["MACD"].ewm(span=9, adjust=False).mean()
114
115 # Spreads
116 df["dist_ema"]       = (df["EMA_10"] - df["EMA_50"]).abs()
117 df["ema10_ema100"]   = df["EMA_10"] - df["EMA_100"]
118 df["dist_ema10_25"]  = (df["EMA_10"] - df["EMA_25"]).abs()
119 df["dist_ema25_100"] = (df["EMA_25"] - df["EMA_100"]).abs()
120
121 # Causal filters (on price)
122 df["BP_filtered"] = butter_filter(df["Close"], order=3, wn=[0.01,
       0.10], btype='band')
123 df["HP_filtered"] = butter_filter(df["Close"], order=3, wn=0.01, btype=
       'high')
124
125 # Vol, Momentum (log-based), RSI, Bollinger, breakout, trend
126 df["vol_21"] = df["log_ret"].rolling(21).std()
```

```
df["mom_5"]  = df["log_ret"].rolling(5).sum()     # log-based momentum
df["mom_20"] = df["log_ret"].rolling(20).sum()    # log-based momentum

df["rsi_14"] = compute_rsi(df["Close"], 14)

ma20  = df["Close"].rolling(20).mean()
std20 = df["Close"].rolling(20).std()
df["bb_pos_20"] = (df["Close"] - ma20) / (2*std20 + EPS)

roll_min20 = df["Close"].rolling(20).min()
roll_max20 = df["Close"].rolling(20).max()
df["brk_20"] = (df["Close"] - roll_min20) / (roll_max20 - roll_min20 +
    EPS)

slope63, r263 = rolling_trend_slope_r2(df["Close"], 63)
df["trend_slope_63"] = slope63
df["trend_r2_63"]    = r263

# Calendar dummies
extra_cat = pd.get_dummies(df.index.month, prefix="m")
extra_cat.index = df.index
cat_cols = list(extra_cat.columns)

# visuals
plt.figure(figsize=(9,3))
plt.plot(df.index, df["EMA_10"], label="EMA10", alpha=0.8)
plt.plot(df.index, df["EMA_50"], label="EMA50", alpha=0.8)
plt.title("EMAs (10, 50)"); plt.legend(); plt.tight_layout(); plt.show
    ()

plt.figure(figsize=(9,3))
plt.plot(df.index, df["MACD"], label="MACD")
plt.plot(df.index, df["MACD_signal"], label="Signal")
plt.title("MACD & Signal"); plt.legend(); plt.tight_layout(); plt.show
    ()


# 4) DTW Pattern-Outcome

USE_DTW = True
try:
    from dtaidistance import dtw as dtw_module
    from dtaidistance.dtw import distance_fast
except Exception:
    print("dtaidistance not available; setting USE_DTW=False (pip
    install dtaidistance to enable).")
    USE_DTW = False

DTW_DEFAULTS = dict(WINDOW_SIZE=30, K_NEIGHBORS=5, SAKOE=3, DTW_TOL=
    1500)

def z_norm_1d(x):
    """Z-normalize a 1D window to make DTW scale-robust."""
```

```
    x = np.asarray(x, dtype=float)
    mu = np.nanmean(x)
    sd = np.nanstd(x)
    if not np.isfinite(sd) or sd < 1e-12:
        return x - mu
    return (x - mu) / sd

X_PO = pd.DataFrame(index=df.index); po_cols = []
if USE_DTW:
    W, K, r, TL = (DTW_DEFAULTS[k] for k in ["WINDOW_SIZE","K_NEIGHBORS","SAKOE","DTW_TOL"])
    series_map = {"log":"log_ret", "macd":"MACD", "bp":"BP_filtered", "hp":"HP_filtered"}

    for tag, col in series_map.items():
        arr    = df[col].values
        future = df["log_ret"].shift(-1).values  # next-day log return
        means, probs = [], []
        it = tqdm(range(len(df)), desc=f"PO_{tag}", disable=not USE_TQDM)
        for i in it:
            if i < W:
                means.append(np.nan); probs.append(np.nan); continue

            win_raw = arr[i - W + 1 : i + 1]
            win = z_norm_1d(win_raw)

            start_lo = max(W - 1, i - TL)
            pairs = []
            for j in range(start_lo, i):
                prev_raw = arr[j - W + 1 : j + 1]
                prev = z_norm_1d(prev_raw)

                try:
                    d = distance_fast(prev, win, window=r)
                except Exception:
                    d = dtw_module.distance(prev, win, window=r)

                pairs.append((d, j))

            pairs.sort(key=lambda x: x[0])
            top = pairs[:K]
            outs = [future[j] for _, j in top if j < len(future)]
            if len(outs) > 0:
                means.append(float(np.mean(outs)))
                probs.append(float(np.mean([1 if o > 0 else 0 for o in outs])))
            else:
                means.append(np.nan); probs.append(np.nan)

        X_PO[f"PO_{tag}_r1_mean"] = means
        X_PO[f"PO_{tag}_up_prob"] = probs
        po_cols += [f"PO_{tag}_r1_mean", f"PO_{tag}_up_prob"]
```

```
224
225     print(f"Built PO block: W={W}, K={K}, r={r}, tol={TL} | cols={len(
        po_cols)}")
226 else:
227     print("DTW disabled     PO block empty.")
228
229
230 # 5) Causal scaling & design assembly
231
232 # Technical numeric list
233 base_feats = [
234     "lag_1","lag_5","lag_10","log_ret",
235     "EMA_10","EMA_25","EMA_50","EMA_100",
236     "EMA10_pct","EMA25_pct","EMA50_pct","EMA100_pct",
237     "MACD","MACD_signal",
238     "dist_ema","ema10_ema100","dist_ema10_25","dist_ema25_100",
239     "BP_filtered","HP_filtered",
240     "vol_21","mom_5","mom_20","rsi_14","bb_pos_20","brk_20",
241     "trend_slope_63","trend_r2_63"
242 ]
243 tech_num_cols = [c for c in base_feats if c in df.columns]
244
245 # Causal rolling z-score (shifted) for TECH numeric
246 WINDOW_SCALE = 252
247 X_num_raw = df[tech_num_cols].copy()
248 X_num_scaled = pd.DataFrame(index=df.index)
249 for c in tech_num_cols:
250     mu = X_num_raw[c].rolling(WINDOW_SCALE).mean().shift(1)
251     sd = X_num_raw[c].rolling(WINDOW_SCALE).std().shift(1)
252     X_num_scaled[c] = (X_num_raw[c] - mu) / (sd + EPS)
253
254 # Drop warm-up NaNs & align everything to a common index
255 X_num_scaled = X_num_scaled.dropna()
256 y_cls = (_df["Close"].shift(-1) > _df["Close"]).astype(int)
        # classification label
257 r_next = _df["logret_next"]                                          #
        next-day log return
258
259 common_idx = X_num_scaled.index
260 if not X_PO.empty: common_idx = common_idx.intersection(X_PO.index)
261 common_idx = (common_idx.intersection(y_cls.index)
262                           .intersection(r_next.index)).sort_values()
263
264 X_num_scaled = X_num_scaled.loc[common_idx]
265 X_PO         = X_PO.loc[common_idx] if not X_PO.empty else X_PO
266 extra_cat    = extra_cat.loc[common_idx].fillna(0)
267 y_cls        = y_cls.loc[common_idx].astype(int)
268 r_next       = r_next.loc[common_idx].astype(float)
269
270 # Assemble TECH design: scaled numeric + calendar dummies
271 X_TECH = pd.concat([X_num_scaled, extra_cat], axis=1)
272
273 # Simple placeholders for config assembly (kept for compatibility)
```

```
X_MIX = None  # built inside run_config for TECH_PLUS_DTW

print("Aligned shapes:",
      "X_TECH", X_TECH.shape,
      "| X_PO", (X_PO.shape if not X_PO.empty else (0,0)),
      "| y", y_cls.shape, "| r_next", r_next.shape)


# 6) Chronological split

N = len(common_idx)
train_end = int(N * 0.60)
val_end   = int(N * 0.80)

idx_train = common_idx[:train_end]
idx_val   = common_idx[train_end:val_end]
idx_test  = common_idx[val_end:]

print("Splits:",
      "train", len(idx_train),
      "val",   len(idx_val),
      "test",  len(idx_test))


# 7) PCA (TECH only, fit on TRAIN)

USE_PCA = True
PCA_VAR = 0.95  #

def build_tech_with_pca(X_tech, idx_tr, idx_va, idx_te):
    """Fit PCA on TRAIN numeric only; apply to VAL/TEST; keep calendar dummies."""
    if not USE_PCA:
        return (X_tech.loc[idx_tr], X_tech.loc[idx_va], X_tech.loc[idx_te], None, [])

    num_cols  = [c for c in X_num_scaled.columns if c in X_tech.columns]
    cat_cols_ = [c for c in extra_cat.columns if c in X_tech.columns]

    X_tr_num, X_va_num, X_te_num = (
        X_tech.loc[idx_tr, num_cols],
        X_tech.loc[idx_va, num_cols],
        X_tech.loc[idx_te, num_cols]
    )

    pca = PCA(n_components=PCA_VAR, svd_solver="full").fit(X_tr_num)
    X_tr_pc = pca.transform(X_tr_num)
    X_va_pc = pca.transform(X_va_num)
    X_te_pc = pca.transform(X_te_num)

    pc_cols = [f"PC{i+1}" for i in range(X_tr_pc.shape[1])]

```

```
    X_tr = pd.concat([pd.DataFrame(X_tr_pc, index=idx_tr, columns=pc_cols),
                      X_tech.loc[idx_tr, cat_cols_]], axis=1)
    X_va = pd.concat([pd.DataFrame(X_va_pc, index=idx_va, columns=pc_cols),
                      X_tech.loc[idx_va, cat_cols_]], axis=1)
    X_te = pd.concat([pd.DataFrame(X_te_pc, index=idx_te, columns=pc_cols),
                      X_tech.loc[idx_te, cat_cols_]], axis=1)

    return X_tr, X_va, X_te, pca, pc_cols


# 8) Models (LR, L1-LR)

BASE_MODELS = {
    "LR":    LogisticRegression(class_weight="balanced", max_iter=1000, random_state=42),
    "LR_L1": LogisticRegression(penalty="l1", solver="saga", C=1.0,
                                class_weight="balanced", max_iter=2000, random_state=42)
}

def youden_threshold(y_true, prob):
    fpr, tpr, thr = roc_curve(y_true, prob)
    j = tpr - fpr
    idx = int(np.argmax(j))
    return float(thr[idx]), auc(fpr, tpr)

def get_scores(model, X, name):
    prob = model.predict_proba(X)[:,1]
    raw  = model.decision_function(X) if hasattr(model, "decision_function") else prob
    return prob, raw


# 9) Train/Eval helper for a CONFIG

def run_config(name, X_base):

    if name == "TECH_ONLY":
        X_tr, X_va, X_te, pca_fit, pc_cols_fit = build_tech_with_pca(X_TECH, idx_train, idx_val, idx_test)
        pca_used = pca_fit; pc_used = pc_cols_fit
    elif name == "DTW_ONLY":
        X_tr, X_va, X_te = X_PO.loc[idx_train], X_PO.loc[idx_val], X_PO.loc[idx_test]
        pca_used = None; pc_used = []
    elif name == "TECH_PLUS_DTW":
        X_tr_t, X_va_t, X_te_t, pca_fit, pc_cols_fit = build_tech_with_pca(X_TECH, idx_train, idx_val, idx_test)
        X_tr = pd.concat([X_tr_t, X_PO.loc[idx_train]], axis=1)
        X_va = pd.concat([X_va_t, X_PO.loc[idx_val]],   axis=1)
```

```
        X_te = pd.concat([X_te_t, X_PO.loc[idx_test]],  axis=1)
        pca_used = pca_fit; pc_used = pc_cols_fit
    else:
        raise ValueError("Unknown config")

    y_tr, y_va, y_te = y_cls.loc[idx_train], y_cls.loc[idx_val], y_cls.loc[idx_test]

    results, models, thr_map, raw_val_map, raw_test_map = [], {}, {}, {}, {}

    for mname, mdl in BASE_MODELS.items():
        model = clone(mdl).fit(X_tr, y_tr)
        models[mname] = model

        pv, sv = get_scores(model, X_va, mname)
        thr, val_auc = youden_threshold(y_va, pv)
        thr_map[mname] = thr
        raw_val_map[mname] = sv

        # Test (reported only, not used for selection)
        pt, st = get_scores(model, X_te, mname)
        raw_test_map[mname] = st
        te_auc = auc(*roc_curve(y_te, pt)[:2])
        preds_te = (pt >= thr).astype(int)

        results.append({
            "config": name, "model": mname,
            "ValAUC": val_auc, "Thr(Youden)": thr, "TestAUC": te_auc,
            "ACC": accuracy_score(y_te, preds_te),
            "PREC": precision_score(y_te, preds_te),
            "REC":  recall_score(y_te, preds_te),
            "F1":   f1_score(y_te, preds_te)
        })

    return {
        "X_tr": X_tr, "X_va": X_va, "X_te": X_te,
        "y_tr": y_tr, "y_va": y_va, "y_te": y_te,
        "pca": pca_used, "pc_cols": pc_used,
        "models": models, "thr_map": thr_map,
        "raw_val_map": raw_val_map, "raw_test_map": raw_test_map,
        "feat_cols_map": {k: list(X_tr.columns) for k in models.keys()},
        "table": pd.DataFrame(results).sort_values(["config","model"])
    }


# 10) Run the three configs

print("\n>>> Running TECH_ONLY")
res_tech = run_config("TECH_ONLY", X_TECH)

print("\n>>> Running DTW_ONLY")
```

```
418 res_dtw  = run_config("DTW_ONLY", X_PO)
419
420 print("\n>>> Running TECH_PLUS_DTW")
421 res_mix  = run_config("TECH_PLUS_DTW", None)
422
423 cfg_table = pd.concat([res_tech["table"], res_dtw["table"], res_mix["
    table"]], axis=0)
424 print("\n=== CONFIG COMPARISON ( V a l YoudenTest  reported) ===")
425 print(cfg_table.to_string(index=False))
426
427 # Pick a winner by best Validation AUC ONLY (no Test peeking).
428 # If ties happen, break them deterministically without using Test (
    prefer simpler model/config).
429 MODEL_ORDER  = {"LR": 0, "LR_L1": 1}  # LR considered "simpler" here
430 CONFIG_ORDER = {"TECH_ONLY": 0, "DTW_ONLY": 1, "TECH_PLUS_DTW": 2}
431
432 tmp = cfg_table.copy()
433 tmp["model_rank"]  = tmp["model"].map(MODEL_ORDER).fillna(99)
434 tmp["config_rank"] = tmp["config"].map(CONFIG_ORDER).fillna(99)
435
436 winner_row = tmp.sort_values(["ValAUC", "config_rank", "model_rank"],
    ascending=[False, True, True]).iloc[0]
437 WIN_CONFIG = winner_row["config"]
438 WIN_MODEL  = winner_row["model"]
439
440 print(f"\nLocked winner (chosen on Validation only) => CONFIG={
    WIN_CONFIG} | MODEL={WIN_MODEL}")
441
442
443
444
445 # Bind winner artifacts for later (static backtest, WF, GLD)
446 if WIN_CONFIG == "TECH_ONLY":
447     RES = res_tech
448 elif WIN_CONFIG == "DTW_ONLY":
449     RES = res_dtw
450 else:
451     RES = res_mix
452
453 X_train, X_val, X_test = RES["X_tr"], RES["X_va"], RES["X_te"]
454 y_train, y_val, y_test = RES["y_tr"], RES["y_va"], RES["y_te"]
455 models       = RES["models"]
456 thr_map      = RES["thr_map"]
457 raw_val_map = RES["raw_val_map"]
458 raw_test_map= RES["raw_test_map"]
459 feature_cols_map = RES["feat_cols_map"]
460 pca = RES["pca"]          # may be None
461 pc_cols = RES["pc_cols"] # list, may be []
462
463 print("\nWinner artifacts:")
464 print("  X_train/X_val/X_test shapes:", X_train.shape, X_val.shape,
    X_test.shape)
```

```
print("  y_train/val/test sizes:", len(y_train), len(y_val), len(y_test))
print("  feature_cols:", len(feature_cols_map[WIN_MODEL]))
if pca is not None:
    print(f"  PCA kept: {len(pc_cols)} components")


# 11) Trading rule selection on Validation     ONE final backtest on Test (locked)


ENTRY_Q = 0.80
EXIT_Q_LIST = [0.30, 0.35, 0.40, 0.45]
MIN_HOLD_LIST = [1, 3, 5, 7]
SLIPPAGE_LIST = [0.0005, 0.0010]   # 5/10 bps

def lf_signal_from_scores(scores, s_hi, s_lo, index, min_hold=3):
    """Long/flat signal with hysteresis + minimum-hold constraint."""
    sig = np.zeros(len(scores), dtype=int)
    hold = 0
    for i, s in enumerate(scores):
        if hold > 0:
            hold -= 1

        prev = sig[i-1] if i > 0 else 0
        if prev == 1:  # long
            sig[i] = 1
            if (s <= s_lo) and (hold == 0):
                sig[i] = 0
                hold = min_hold
        else:          # flat
            sig[i] = 0
            if (s >= s_hi) and (hold == 0):
                sig[i] = 1
                hold = min_hold

    return pd.Series(sig, index=index)

def backtest_lf(sig, idx, slip=0.0005, r_series=r_next):
    """Backtest long/flat strategy on next-day log returns with transaction costs."""
    r = r_series.loc[idx].dropna()
    sig = sig.loc[idx].astype(int)
    sig = sig.reindex(r.index).fillna(0).astype(int)

    flips = np.abs(sig.diff().fillna(0))
    gross = sig * r
    net   = gross - flips * slip
    cum   = net.cumsum()
    dd    = cum - np.maximum.accumulate(cum)

    sharpe_g = gross.mean()/(gross.std()+EPS)*np.sqrt(252)
    sharpe_n = net.mean()  /(net.std()  +EPS)*np.sqrt(252)
```

```
    tpd       = flips.sum()/max(1, len(sig))
    res = dict(gross_sharpe=float(sharpe_g), net_sharpe=float(sharpe_n),
               trades_per_day=float(tpd), long=float((sig==1).mean()),
               flat=float((sig==0).mean()), maxdd=float(dd.min()))
    return res, cum, dd

def get_entry_exit_cutoffs(scores_ref, entry_q=ENTRY_Q, exit_q=0.35):
    """Compute hysteresis cutoffs from a reference score distribution (Validation)."""
    return float(np.quantile(scores_ref, entry_q)), float(np.quantile(scores_ref, exit_q))

# Winner scores on Validation / Test (locked model)
s_val = raw_val_map[WIN_MODEL]
s_te  = raw_test_map[WIN_MODEL]

# ── Select trading-rule params using Validation ONLY ──
rows = []
for exit_q in EXIT_Q_LIST:
    s_hi, s_lo = get_entry_exit_cutoffs(s_val, ENTRY_Q, exit_q)
    for mh in MIN_HOLD_LIST:
        sig_val = lf_signal_from_scores(s_val, s_hi, s_lo, index=X_val.index, min_hold=mh)
        for slip in SLIPPAGE_LIST:
            res_val, _, _ = backtest_lf(sig_val, X_val.index, slip=slip)
            rows.append({"exit_q":exit_q, "MIN_HOLD":mh, "slip_bps":int(slip*1e4), **res_val})

val_sweep = pd.DataFrame(rows).sort_values(["net_sharpe","gross_sharpe"], ascending=False)
best_rule = val_sweep.iloc[0]
BEST_EXIT_Q   = float(best_rule["exit_q"])
BEST_MIN_HOLD = int(best_rule["MIN_HOLD"])
BEST_SLIP     = float(best_rule["slip_bps"]) / 1e4

print("\n=== Trading-rule selection (Validation only) ===")
print(val_sweep.head(10).to_string(index=False))
print(f"\nLocked trading rule -> exit_q={BEST_EXIT_Q:.2f}, MIN_HOLD={BEST_MIN_HOLD}, "
      f"cost={int(BEST_SLIP*1e4)} bps | Val net Sharpe={best_rule['net_sharpe']:.3f}")

# ── ONE final backtest on Test with the locked rule ──
s_hi, s_lo = get_entry_exit_cutoffs(s_val, ENTRY_Q, BEST_EXIT_Q)  # cutoffs from Validation distribution
sig_te = lf_signal_from_scores(s_te, s_hi, s_lo, index=X_test.index, min_hold=BEST_MIN_HOLD)
final_res, final_cum, final_dd = backtest_lf(sig_te, X_test.index, slip=BEST_SLIP)

print("\n=== FINAL Test Backtest (locked) ===")
```

```
556 print({k: (round(v, 4) if isinstance(v, float) else v) for k, v in
        final_res.items()})
557 print(f"Entry_q={ENTRY_Q:.2f} | Exit_q={BEST_EXIT_Q:.2f} | MIN_HOLD={
        BEST_MIN_HOLD} | Cost={int(BEST_SLIP*1e4)} bps")
558
559 plt.figure(figsize=(9,3))
560 plt.plot(final_cum.index, final_cum.values, lw=1.2)
561 plt.title("Equity Curve (Test, net log-return)     locked rule"); plt.
        xlabel("Date"); plt.ylabel("Cumulative")
562 plt.tight_layout(); plt.show()
563
564 plt.figure(figsize=(9,2.6))
565 plt.plot(final_dd.index, final_dd.values, lw=1.2)
566 plt.title("Drawdown (Test)     locked rule"); plt.xlabel("Date"); plt.
        ylabel("Drawdown")
567 plt.tight_layout(); plt.show()
568
569
570 # 12) ROC / PR for the winner
571
572 def plot_roc_pr(y_true, prob, label):
573     fpr, tpr, _ = roc_curve(y_true, prob)
574     roc_auc = auc(fpr, tpr)
575     plt.figure(figsize=(5,4))
576     plt.plot(fpr, tpr, lw=2)
577     plt.plot([0,1],[0,1], ls="--", lw=1)
578     plt.title(f"ROC ({label}) AUC={roc_auc:.3f}")
579     plt.xlabel("FPR"); plt.ylabel("TPR"); plt.tight_layout(); plt.show
        ()
580
581     pr, rc, _ = precision_recall_curve(y_true, prob)
582     ap = average_precision_score(y_true, prob)
583     plt.figure(figsize=(5,4))
584     plt.plot(rc, pr, lw=2)
585     plt.title(f"PR ({label}) AP={ap:.3f}")
586     plt.xlabel("Recall"); plt.ylabel("Precision"); plt.tight_layout();
        plt.show()
587
588 # probs for plotting (winner model)
589 pv = models[WIN_MODEL].predict_proba(X_val)[:,1]
590 pt = models[WIN_MODEL].predict_proba(X_test)[:,1]
591 plot_roc_pr(y_val, pv, "Val")
592 plot_roc_pr(y_test, pt, "Test")
593
594
595
596 # 13) Walk-Forward Evaluation
597
598 TRAIN_DAYS = 750
599 TEST_DAYS  = 125
600
601 # unify y over full span for rolling windows
```

```
all_idx = pd.Index(sorted(set(X_train.index) | set(X_val.index) | set(X_test.index)))
y_all   = pd.concat([y_train, y_val, y_test]).reindex(all_idx).dropna().astype(int)

def model_score(mdl, X):
    if hasattr(mdl, "decision_function"):
        return mdl.decision_function(X)
    else:
        return mdl.predict_proba(X)[:, 1]

def wf_build_X(idx_tr, idx_val_in, idx_te, idx_tr_in):
    if WIN_CONFIG == "TECH_ONLY":
        # PCA on TECH numeric in-window (fit only on idx_tr_in)
        X_tr_tech = X_TECH.loc[idx_tr]
        num_cols  = [c for c in X_num_scaled.columns if c in X_tr_tech.columns]
        cat_cols_ = [c for c in extra_cat.columns    if c in X_tr_tech.columns]

        pca_l = PCA(n_components=PCA_VAR, svd_solver="full").fit(X_tr_tech.loc[idx_tr_in, num_cols])

        def to_pc(Xtech):
            Xn = Xtech[num_cols]
            Xp = pca_l.transform(Xn)
            pc_cols_l = [f"PC{i+1}" for i in range(Xp.shape[1])]
            return pd.concat([pd.DataFrame(Xp, index=Xtech.index, columns=pc_cols_l),
                              Xtech[cat_cols_]], axis=1)

        X_tr_in   = to_pc(X_TECH.loc[idx_tr_in])
        X_val_in  = to_pc(X_TECH.loc[idx_val_in])
        X_tr_full = to_pc(X_TECH.loc[idx_tr])
        X_te      = to_pc(X_TECH.loc[idx_te])

    elif WIN_CONFIG == "DTW_ONLY":
        X_tr_in   = X_PO.loc[idx_tr_in]
        X_val_in  = X_PO.loc[idx_val_in]
        X_tr_full = X_PO.loc[idx_tr]
        X_te      = X_PO.loc[idx_te]

    else:
        # TECHPCA (fit on idx_tr_in) then concat PO
        X_tr_tech = X_TECH.loc[idx_tr]
        num_cols  = [c for c in X_num_scaled.columns if c in X_tr_tech.columns]
        cat_cols_ = [c for c in extra_cat.columns    if c in X_tr_tech.columns]

        pca_l = PCA(n_components=PCA_VAR, svd_solver="full").fit(X_tr_tech.loc[idx_tr_in, num_cols])

```

```
646         def to_pc(Xtech):
647             Xn = Xtech[num_cols]
648             Xp = pca_1.transform(Xn)
649             pc_cols_1 = [f"PC{i+1}" for i in range(Xp.shape[1])]
650             return pd.concat([pd.DataFrame(Xp, index=Xtech.index,
    columns=pc_cols_1),
651                               Xtech[cat_cols_]], axis=1)
652
653         Xt_tr_in   = to_pc(X_TECH.loc[idx_tr_in])
654         Xt_val_in  = to_pc(X_TECH.loc[idx_val_in])
655         Xt_tr_full = to_pc(X_TECH.loc[idx_tr])
656         Xt_te      = to_pc(X_TECH.loc[idx_te])
657
658         X_tr_in   = pd.concat([Xt_tr_in,   X_PO.loc[idx_tr_in]], axis
    =1)
659         X_val_in  = pd.concat([Xt_val_in,  X_PO.loc[idx_val_in]], axis
    =1)
660         X_tr_full = pd.concat([Xt_tr_full, X_PO.loc[idx_tr]],      axis
    =1)
661         X_te      = pd.concat([Xt_te,      X_PO.loc[idx_te]],      axis
    =1)
662
663     return X_tr_in, X_val_in, X_tr_full, X_te
664
665 def wf_fit_and_score(idx_tr, idx_te):
666     # inner split
667     cut = int(len(idx_tr) * 0.8)
668     idx_tr_in   = idx_tr[:cut]
669     idx_val_in  = idx_tr[cut:]
670
671     # Build X blocks (PCA refit only on idx_tr_in)
672     X_tr_in, X_val_in, X_tr_full, X_te = wf_build_X(idx_tr, idx_val_in,
     idx_te, idx_tr_in)
673
674     y_tr_in   = y_cls.loc[idx_tr_in]
675     y_tr_full = y_cls.loc[idx_tr]
676
677     # 1) Fit on inner-train ONLY
678     mdl_in = clone(BASE_MODELS[WIN_MODEL]).fit(X_tr_in, y_tr_in)
679
680     # 2) Use inner-val score distribution for cutoffs
681     s_val = model_score(mdl_in, X_val_in)
682     s_hi = float(np.quantile(s_val, ENTRY_Q))
683     s_lo = float(np.quantile(s_val, BEST_EXIT_Q))
684
685     # 3) Refit on full training window
686     mdl = clone(BASE_MODELS[WIN_MODEL]).fit(X_tr_full, y_tr_full)
687
688     # 4) Score test window with SAME score type and trade
689     s_te = model_score(mdl, X_te)
690     sig  = lf_signal_from_scores(s_te, s_hi, s_lo, index=idx_te,
    min_hold=BEST_MIN_HOLD)
691     res, _, _ = backtest_lf(sig, idx_te, slip=BEST_SLIP)
```

```
    return res

# ——— WF loop (same logic) + store end dates for plotting ———
wf_rows, sh_list, td_list = [], [], []
wf_end_dates = []

for start in range(0, len(y_all) - TRAIN_DAYS - TEST_DAYS, TEST_DAYS):
    idx_tr = y_all.index[start : start + TRAIN_DAYS]
    idx_te = y_all.index[start + TRAIN_DAYS : start + TRAIN_DAYS +
    TEST_DAYS]

    wf_end_dates.append(idx_te[-1])  # end date of the test window

    res = wf_fit_and_score(idx_tr, idx_te)
    wf_rows.append(res)
    sh_list.append(res["net_sharpe"])
    td_list.append(res["trades_per_day"])

wf_df = pd.DataFrame(wf_rows)

# ——— PLOT: WF net Sharpe per window ———
wf_net = np.array(sh_list, dtype=float)

# Option A: x-axis = window index
plt.figure(figsize=(9,3))
plt.plot(np.arange(len(wf_net)), wf_net, lw=1.2)
plt.axhline(0.0, lw=1.0, ls="--")
plt.title("Walk-Forward Net Sharpe per Window")
plt.xlabel("WF window index")
plt.ylabel("Net Sharpe")
plt.tight_layout()
plt.show()

# Option B: x-axis = test-window end date (thesis-friendly)
plt.figure(figsize=(9,3))
plt.plot(pd.to_datetime(wf_end_dates), wf_net, lw=1.2)
plt.axhline(0.0, lw=1.0, ls="--")
plt.title("Walk-Forward Net Sharpe per Window (by end date)")
plt.xlabel("Test-window end date")
plt.ylabel("Net Sharpe")
plt.tight_layout()
plt.show()

# ——— Existing summary (unchanged) ———
mean_sh, std_sh = float(np.mean(sh_list)), float(np.std(sh_list))
mean_tpd        = float(np.mean(td_list))
print("\n=== Walk-Forward Summary (winner config + locked rule) ===")
print(f"Windows: {len(wf_rows)} | mean net Sharpe={mean_sh:.3f} | std={
    std_sh:.3f} | trades/day={mean_tpd:.3f}")
```

```
# 14) Regime-Conditional Performance


from sklearn.metrics import roc_auc_score

# ---- 14.1 Choose best model per config using Validation only (deterministic tie-break, no Test use)
MODEL_ORDER  = {"LR": 0, "LR_L1": 1}
CONFIG_ORDER = {"TECH_ONLY": 0, "DTW_ONLY": 1, "TECH_PLUS_DTW": 2}

tmp = cfg_table.copy()
tmp["model_rank"]  = tmp["model"].map(MODEL_ORDER).fillna(99)
tmp["config_rank"] = tmp["config"].map(CONFIG_ORDER).fillna(99)

best_by_cfg = (tmp.sort_values(["config", "ValAUC", "model_rank"], ascending=[True, False, True])
                  .groupby("config", as_index=False)
                  .first())

best_model_map = dict(zip(best_by_cfg["config"], best_by_cfg["model"]))
print("\nBest model per config (chosen on Validation only):", best_model_map)

# ---- 14.2 Build past-only regime labels (aligned to Test index)
# Regime A: Volatility regime (21d rolling std of log returns, past-only)
vol21_past = df["log_ret"].rolling(21).std().shift(1)

# Regime B: Trend regime (EMA10 - EMA50, past-only)
trend_past = (df["EMA_10"] - df["EMA_50"]).shift(1)

reg_df = pd.DataFrame({
    "vol21_past": vol21_past,
    "trend_past": trend_past
}).loc[X_test.index].dropna()

#  binary regimes using median splits on the TEST timeline (reporting only; not used to tune)
vol_med   = float(reg_df["vol21_past"].median())
trend_med = float(reg_df["trend_past"].median())

reg_df["REG_VOL"]   = np.where(reg_df["vol21_past"] >= vol_med, "HighVol", "LowVol")
reg_df["REG_TREND"] = np.where(reg_df["trend_past"] >= trend_med, "UpTrend", "Down/Flat")

# Combined regime label (optional)
reg_df["REG_COMBO"] = reg_df["REG_VOL"] + " & " + reg_df["REG_TREND"]

# ---- 14.3 Helper: AUC-by-regime on Test
def auc_by_regime(y_true, prob, regime_series: pd.Series):
    out = []
    for g in regime_series.dropna().unique():
```

```
        mask = (regime_series == g)
        yy = y_true[mask]
        pp = prob[mask]
        if len(np.unique(yy)) < 2 or len(yy) < 30:
            out.append((g, np.nan, int(mask.sum())))
        else:
            out.append((g, float(roc_auc_score(yy, pp)), int(mask.sum())))
    return pd.DataFrame(out, columns=["Regime", "TestAUC", "N"]).sort_values("Regime")

# ── 14.4  Test probabilities for each config (no retraining; use the already trained models from run_config)

cfg_results = {
    "TECH_ONLY": res_tech,
    "DTW_ONLY": res_dtw,
    "TECH_PLUS_DTW": res_mix
}

reg_idx = reg_df.index
y_reg = y_test.loc[reg_idx]

auc_tables = {}

for cfg_name, RESCFG in cfg_results.items():
    mname = best_model_map[cfg_name]
    mdl   = RESCFG["models"][mname]

    X_te_cfg = RESCFG["X_te"].loc[reg_idx]
    prob_te  = mdl.predict_proba(X_te_cfg)[:, 1]
    prob_te  = pd.Series(prob_te, index=reg_idx)

    # AUC by volatility regime
    t_vol = auc_by_regime(y_reg, prob_te, reg_df["REG_VOL"])
    t_vol["Config"] = cfg_name
    t_vol["Model"]  = mname

    # AUC by trend regime
    t_tr  = auc_by_regime(y_reg, prob_te, reg_df["REG_TREND"])
    t_tr["Config"] = cfg_name
    t_tr["Model"]  = mname

    # AUC by combined regime (optional, more granular)
    t_cb  = auc_by_regime(y_reg, prob_te, reg_df["REG_COMBO"])
    t_cb["Config"] = cfg_name
    t_cb["Model"]  = mname

    auc_tables[cfg_name] = {"VOL": t_vol, "TREND": t_tr, "COMBO": t_cb}

print("\n=== Regime AUC on Test (Volatility) ===")
print(pd.concat([auc_tables[c]["VOL"] for c in auc_tables]).sort_values(["Regime","Config"]).to_string(index=False))
```

```
838
839 print("\n==== Regime AUC on Test (Trend) ====")
840 print(pd.concat([auc_tables[c]["TREND"] for c in auc_tables]).
        sort_values(["Regime","Config"]).to_string(index=False))
841
842 print("\n==== Regime AUC on Test (Combined) ====")
843 print(pd.concat([auc_tables[c]["COMBO"] for c in auc_tables]).
        sort_values(["Regime","Config"]).to_string(index=False))
844
845 # Optional quick visualization: average AUC by config in each regime
        set
846 def summarize_auc(df_auc):
847     return (df_auc.groupby(["Config"])["TestAUC"].mean()
848                   .sort_values(ascending=False)
849                   .rename("MeanTestAUC")
850                   .to_frame())
851
852 print("\nMean Test AUC by config (Vol regimes):")
853 print(summarize_auc(pd.concat([auc_tables[c]["VOL"] for c in auc_tables
        ])))
854
855 print("\nMean Test AUC by config (Trend regimes):")
856 print(summarize_auc(pd.concat([auc_tables[c]["TREND"] for c in
        auc_tables])))
857
858 % ==================================
```

Appendix B

Appendix B: GLD Zero-Shot Pipeline Implementation

This appendix presents the full Python implementation used for the **GLD zero-shot cross-asset evaluation**. The code below corresponds to the sections of the script dedicated to the SPDR Gold Shares ETF (GLD) and follows the protocol defined in Chapter 5, Section 5.4.

In this evaluation, GLD is treated as a *pure out-of-domain test asset*:

- The predictive model is trained and selected on SPX only.
- Trading-rule parameters (entry/exit thresholds, minimum holding) are fixed using SPX validation only.
- The frozen SPX-trained model and trading rule are applied directly to GLD without any retraining, recalibration, or retuning.

The script includes:

- Data loading and preprocessing of GLD log-returns
- Construction of the same feature blocks used for SPX (technical features and DTW PO features)
- Application of SPX-fitted preprocessing objects (e.g., scaling/PCA parameters if used)
- Zero-shot prediction and backtesting on GLD under the fixed trading rule
- Reporting of predictive (AUC) and economic metrics (net Sharpe, drawdown, turnover)

Note: Regime conditioning is not applied to GLD in this appendix, since the objective is to evaluate cross-asset transferability under fully frozen models and trading rules.

```

# 15) GLD Zero-Shot Evaluation (apply SPX-trained artifacts to GLD)
# - NO retraining on GLD
# - NO tuning / reselection on GLD
# - Reports TECH_ONLY / DTW_ONLY / TECH+DTW side-by-side


GLD_CSV = "gld_prices.csv"

# ---- 15.1 Load GLD + define returns/label (same as SPX) ----
gld = (pd.read_csv(GLD_CSV, parse_dates=["Date"])
         .sort_values("Date")
         .set_index("Date"))[["Close"]].copy()

gld["log_ret"]     = np.log(gld["Close"]).diff()
gld["logret_next"] = np.log(gld["Close"]).diff().shift(-1)  # t -> t+1
gld["y_cls"]       = (gld["Close"].shift(-1) > gld["Close"]).astype(int)

print("GLD shape:", gld.shape, "| range:", gld.index.min(), "to", gld.index.max())

plt.figure(figsize=(9,3))
plt.plot(gld.index, gld["Close"], lw=1.2)
plt.title("GLD Close"); plt.xlabel("Date"); plt.ylabel("Price")
plt.tight_layout(); plt.show()


# ---- 15.2 Rebuild TECH features on GLD (same formulas) ----
gld_f = gld.copy()

for lag in [1, 5, 10]:
    gld_f[f"lag_{lag}"] = gld_f["log_ret"].shift(lag)

for span in [10, 25, 50, 100, 12, 26]:
    gld_f[f"EMA_{span}"] = gld_f["Close"].ewm(span=span, adjust=False).mean()

gld_f["EMA10_pct"]  = (gld_f["EMA_10"]  / gld_f["Close"] - 1) * 100
gld_f["EMA25_pct"]  = (gld_f["EMA_25"]  / gld_f["Close"] - 1) * 100
gld_f["EMA50_pct"]  = (gld_f["EMA_50"]  / gld_f["Close"] - 1) * 100
gld_f["EMA100_pct"] = (gld_f["EMA_100"] / gld_f["Close"] - 1) * 100

gld_f["MACD"]        = gld_f["EMA_12"] - gld_f["EMA_26"]
gld_f["MACD_signal"] = gld_f["MACD"].ewm(span=9, adjust=False).mean()

gld_f["dist_ema"]       = (gld_f["EMA_10"] - gld_f["EMA_50"]).abs()
gld_f["ema10_ema100"]   = gld_f["EMA_10"] - gld_f["EMA_100"]
gld_f["dist_ema10_25"]  = (gld_f["EMA_10"] - gld_f["EMA_25"]).abs()
gld_f["dist_ema25_100"] = (gld_f["EMA_25"] - gld_f["EMA_100"]).abs()

```

```
gld_f["BP_filtered"] = butter_filter(gld_f["Close"], order=3, wn=[0.01, 0.10], btype="band")
gld_f["HP_filtered"] = butter_filter(gld_f["Close"], order=3, wn=0.01, btype="high")

gld_f["vol_21"] = gld_f["log_ret"].rolling(21).std()
gld_f["mom_5"]  = gld_f["log_ret"].rolling(5).sum()
gld_f["mom_20"] = gld_f["log_ret"].rolling(20).sum()

gld_f["rsi_14"] = compute_rsi(gld_f["Close"], 14)

ma20  = gld_f["Close"].rolling(20).mean()
std20 = gld_f["Close"].rolling(20).std()
gld_f["bb_pos_20"] = (gld_f["Close"] - ma20) / (2*std20 + EPS)

roll_min20 = gld_f["Close"].rolling(20).min()
roll_max20 = gld_f["Close"].rolling(20).max()
gld_f["brk_20"] = (gld_f["Close"] - roll_min20) / (roll_max20 - roll_min20 + EPS)

slope63, r263 = rolling_trend_slope_r2(gld_f["Close"], 63)
gld_f["trend_slope_63"] = slope63
gld_f["trend_r2_63"]    = r263

# Calendar dummies: MUST match SPX dummy columns
gld_cat = pd.get_dummies(gld_f.index.month, prefix="m")
gld_cat.index = gld_f.index
gld_cat = gld_cat.reindex(columns=extra_cat.columns, fill_value=0)

# Scale TECH numerics on GLD using past-only rolling stats (same as SPX)
WINDOW_SCALE = 252
num_cols_ref = list(X_num_scaled.columns)  # numeric columns used in SPX pipeline
gld_num_scaled = pd.DataFrame(index=gld_f.index)

for c in num_cols_ref:
    if c not in gld_f.columns:
        gld_num_scaled[c] = np.nan
        continue
    mu = gld_f[c].rolling(WINDOW_SCALE).mean().shift(1)
    sd = gld_f[c].rolling(WINDOW_SCALE).std().shift(1)
    gld_num_scaled[c] = (gld_f[c] - mu) / (sd + EPS)

gld_TECH = pd.concat([gld_num_scaled, gld_cat], axis=1)


# ---- 15.3 Rebuild DTW PO on GLD (same DTW params, z-norm windows) ----
gld_PO = pd.DataFrame(index=gld_f.index, columns=(X_PO.columns if not X_PO.empty else []), dtype=float)

if USE_DTW and (not X_PO.empty):
```

```
    W  = DTW_DEFAULTS["WINDOW_SIZE"]
    K  = DTW_DEFAULTS["K_NEIGHBORS"]
    r  = DTW_DEFAULTS["SAKOE"]
    TL = DTW_DEFAULTS["DTW_TOL"]

    series_map = {"log":"log_ret", "macd":"MACD", "bp":"BP_filtered", "hp":"HP_filtered"}
    future = gld_f["log_ret"].shift(-1).values

    for tag, col in series_map.items():
        arr = gld_f[col].values
        means, probs = [], []
        it = tqdm(range(len(gld_f)), desc=f"GLD_PO_{tag}", disable=not USE_TQDM)
        for i in it:
            if i < W:
                means.append(np.nan); probs.append(np.nan); continue

            win = z_norm_1d(arr[i - W + 1 : i + 1])
            start_lo = max(W - 1, i - TL)

            pairs = []
            for j in range(start_lo, i):
                prev = z_norm_1d(arr[j - W + 1 : j + 1])
                try:
                    d = distance_fast(prev, win, window=r)
                except Exception:
                    d = dtw_module.distance(prev, win, window=r)
                pairs.append((d, j))

            pairs.sort(key=lambda x: x[0])
            top = pairs[:K]
            outs = [future[j] for _, j in top if j < len(future)]
            if len(outs) > 0:
                means.append(float(np.mean(outs)))
                probs.append(float(np.mean([1 if o > 0 else 0 for o in outs])))
            else:
                means.append(np.nan); probs.append(np.nan)

        gld_PO[f"PO_{tag}_rl_mean"] = means
        gld_PO[f"PO_{tag}_up_prob"] = probs

    # match SPX PO column order exactly
    gld_PO = gld_PO.reindex(columns=X_PO.columns)

print("Built GLD blocks:",
      "TECH", gld_TECH.shape,
      "| PO", gld_PO.shape)


# ——— 15.4 Align GLD index (drop warmups) ———
gld_idx = gld_TECH.dropna().index
```

```
if gld_PO.shape[1] > 0:
    gld_idx = gld_idx.intersection(gld_PO.dropna().index)

gld_idx = (gld_idx.intersection(gld["y_cls"].dropna().index)
                  .intersection(gld["logret_next"].dropna().index)).
    sort_values()

y_gld = gld.loc[gld_idx, "y_cls"].astype(int)
r_next_gld = gld.loc[gld_idx, "logret_next"].astype(float)

print("GLD aligned:", len(gld_idx), "rows")


# ---- 15.5 Helper: build GLD design exactly like each SPX config
    expects ----
def build_gld_X_for_config(RESCFG, cfg_name, model_name):
    """
    RESCFG is one of {res_tech, res_dtw, res_mix} from SPX.
    Uses its PCA (if any) and its exact feature order (feat_cols_map).
    """
    feat_order = RESCFG["feat_cols_map"][model_name]

    if cfg_name == "DTW_ONLY":
        X = gld_PO.loc[gld_idx].reindex(columns=feat_order)
        return X

    # TECH_ONLY or TECH_PLUS_DTW: need PCA transform on GLD numerics
    using SPX-trained PCA
    pca_cfg = RESCFG["pca"]
    pc_cols_cfg = RESCFG["pc_cols"]
    assert pca_cfg is not None, f"{cfg_name} expects PCA but pca is
    None."

    Xnum = gld_TECH.loc[gld_idx, num_cols_ref]
    Xcat = gld_TECH.loc[gld_idx, list(extra_cat.columns)]

    Xpc = pca_cfg.transform(Xnum.values)
    Xpc = pd.DataFrame(Xpc, index=gld_idx, columns=pc_cols_cfg)

    Xt = pd.concat([Xpc, Xcat], axis=1)

    if cfg_name == "TECH_ONLY":
        X = Xt.reindex(columns=feat_order)
        return X

    # TECH_PLUS_DTW
    X = pd.concat([Xt, gld_PO.loc[gld_idx]], axis=1).reindex(columns=
    feat_order)
    return X


# ---- 15.6 Evaluate ALL THREE SPX-trained configs on GLD (zero-shot)
    ----
```

```
from sklearn.metrics import roc_auc_score

cfg_results = {
    "TECH_ONLY": res_tech,
    "DTW_ONLY": res_dtw,
    "TECH_PLUS_DTW": res_mix
}

# choose best model per config from SPX Validation only (reuse your earlier mapping if present)
if "best_model_map" not in globals():
    # fallback: pick by highest ValAUC per config (no Test)
    tmp = cfg_table.copy()
    MODEL_ORDER = {"LR": 0, "LR_L1": 1}
    tmp["model_rank"] = tmp["model"].map(MODEL_ORDER).fillna(99)
    best_by_cfg = (tmp.sort_values(["config","ValAUC","model_rank"], ascending=[True,False,True])
                      .groupby("config", as_index=False).first())
    best_model_map = dict(zip(best_by_cfg["config"], best_by_cfg["model"]))

print("Using SPX-chosen best model per config:", best_model_map)

rows = []
for cfg_name, RESCFG in cfg_results.items():
    mname = best_model_map[cfg_name]
    mdl   = RESCFG["models"][mname]  # SPX-trained, frozen

    Xg = build_gld_X_for_config(RESCFG, cfg_name, mname)

    prob = mdl.predict_proba(Xg)[:, 1]
    auc_ = roc_auc_score(y_gld, prob)

    # Trading (locked rule): use SPX validation score distribution for THIS config/model
    # raw_val_map already uses decision_function if available else prob -> consistent score type.
    s_val_cfg = RESCFG["raw_val_map"][mname]
    s_hi = float(np.quantile(s_val_cfg, ENTRY_Q))
    s_lo = float(np.quantile(s_val_cfg, BEST_EXIT_Q))

    # Use matching score type on GLD:
    if hasattr(mdl, "decision_function"):
        score = mdl.decision_function(Xg)
    else:
        score = prob

    sig = lf_signal_from_scores(score, s_hi, s_lo, index=gld_idx, min_hold=BEST_MIN_HOLD)
    bt, _, _ = backtest_lf(sig, gld_idx, slip=BEST_SLIP, r_series=r_next_gld)

    rows.append({
```

```
            "Config": cfg_name, "Model": mname,
            "GLD_AUC": float(auc_),
            "GLD_netSharpe": float(bt["net_sharpe"]),
            "GLD_maxDD": float(bt["maxdd"]),
            "GLD_tradesPerDay": float(bt["trades_per_day"]),
            "GLD_longFrac": float(bt["long"])
        })

gld_report = pd.DataFrame(rows).sort_values(["GLD_AUC"], ascending=
    False)
print("\n=== GLD Zero-Shot Report (SPX-trained, no tuning) ===")
print(gld_report.to_string(index=False))
```

Bibliography

[1] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. Fifth. Wiley, 2015 (cit. on p. 7).

[2] James D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994 (cit. on p. 7).

[3] Sang Hyuk Kim, Hee Soo Lee, Han Jun Ko, Seung Hwan Jeong, Hyun Woo Byun, and Kyong Joo Oh. "Pattern Matching Trading System Based on the Dynamic Time Warping Algorithm". In: *Sustainability* 10 (2018), p. 4641. DOI: 10.3390/su10124641 (cit. on pp. 8, 11).

[4] Shangkun Deng, Youtao Xiang, Boyang Nan, Hongyu Tian, and Zhe Sun. "A Hybrid Model of Dynamic Time Wrapping and Hidden Markov Model for Forecasting and Trading in Crude Oil Market". In: *Soft Computing* (2019). DOI: 10.1007/s00500-019-04304-9 (cit. on pp. 8, 11).

[5] Hiroaki Sakoe and Seibi Chiba. "Dynamic Programming Algorithm Optimization for Spoken Word Recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (1978), pp. 43–49 (cit. on pp. 9, 11).

[6] Kei Nakagawa, Mitsuyoshi Imamura, and Kenichi Yoshida. "Stock Price Prediction with Fluctuation Patterns Using Indexing Dynamic Time Warping and k*-Nearest Neighbors". In: *Lecture Notes in Computer Science*. Springer, 2018. DOI: 10.1007/978-3-319-93794-6_7 (cit. on pp. 10, 11).

Dedications

To my family, for their endless patience, strength, and belief in me, and to my professors, whose guidance and understanding helped me complete this journey.