



UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI



UNIVERSITA' DEGLI STUDI GUGLIELMO MARCONI

Master in Computer Science

Intelligent Forex Trading

An Adaptive Machine Learning Framework for Handling Market Non-Stationary in Algorithmic
Forex Trading - A Simulation-Based Study

Academic Advisor

Ryan Suryanto

Candidate

Lawrance Koh Chee Hng
MCSLT00274





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

ACADEMIC YEAR
2025 / 2026

Table of Contents

Table of Contents.....	2
Abstract.....	5
1. Introduction and Problem Statement.....	6
1.1 Proposed Solution: The Adaptive Algorithmic Trading System.....	6
1.1.1 The Hybrid Machine Learning Core.....	6
1.1.2 Dynamic Parameter Optimization.....	7
1.2 Methodology for Continuous Robustness: The MLOps Pipeline.....	7
2. Literature Review.....	9
2.1 Strategic Imperative: Non-Stationarity and the Adaptive Advantage.....	9
2.1.1 The Fundamental Challenge of Market Non-Stationarity.....	9
2.1.2 Empirical Justification for Adaptive Strategies.....	10
2.2 Dynamic Regime Classification: The Machine Learning Core.....	10
2.2.1 Selection Justification: Unsupervised Clustering (UCL).....	10
2.2.2 Advanced Feature Engineering for Forex Regime Differentiation.....	11
2.2.3. Clustering Methodology and Cluster Validation.....	12
2.3 The Dynamic Mapping Layer: Optimization and MQL5 Interface.....	13
2.3.1 Conditional Parameter Optimization (CPO) Mechanics.....	13
2.3.2 Optimization Methodology and Search Space.....	13
2.3.3 Rigorous Evaluation Metrics for Adaptive Systems.....	14
2.4 MLOps for Continuous Adaptivity and Robustness.....	15
2.4.1 The Necessity of MLOps for Financial Time Series.....	15
2.4.2 Justification of Fixed Weekly Rolling Window Retraining.....	15
2.4.3 The Rolling Window Mechanism.....	16
2.4.4 Robustness Validation: Walk Forward Analysis (WFA).....	17
2.5 Technical Architecture: Python-MQL5 Interoperability.....	17
2.5.1 The Inter-Process Communication (IPC) Backbone.....	17
2.5.2 JSON Schema for Dynamic Parameter Transmission.....	17
2.5.3 MQL5 Expert Advisor Adaptation (The Execution Layer).....	18
2.6 Conclusions and Future Work.....	18





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

2.6.1 Synthesis of Findings.....	18
2.6.2 Recommendations and Future Trajectories.....	19
3. Research Methodology.....	20
3.1 Introduction to the Methodology.....	20
3.2 Research Design.....	20
3.3 Research Questions and Hypotheses.....	20
3.4 Data Sources and Data Collection.....	21
3.5 System / Model / Framework Description.....	21
3.5.1 The Machine Learning Core: Regime Classification.....	22
3.5.2 The Dynamic Mapping Layer: Conditional Parameter Optimization (CPO).....	22
3.5.3 Inter-Process Communication and Execution.....	22
3.6 Tools, Technologies, and Platforms.....	22
3.7 Evaluation Metrics and Analysis Techniques.....	23
3.8 Validation and Comparison.....	23
3.9 Ethical Considerations.....	24
3.10 Limitations of the Methodology.....	24
3.11 Summary of the Chapter.....	24
4. Implementation and Results.....	26
4.1 System Implementation Overview.....	26
4.1.1 System Architecture.....	26
4.2 Walk-Forward Analysis Results.....	30
4.2.1 WFA Configuration.....	31
4.2.2 Aggregate Results.....	31
4.2.3 Regime Distribution.....	32
4.2.4 Final Model Cluster Centroids.....	33
4.3 MQL5 Integration Results.....	34
4.3.1 Backtest Methodology.....	34
4.3.2 Baseline Backtest (Static Parameters).....	34
4.3.3 Adaptive Backtest (ML-Driven Parameters).....	36
4.3.4 Comparative Analysis.....	37
4.4 System Validation Summary.....	38
4.4.1 Validation Criteria Assessment.....	38
4.5 System Monitoring & Dashboard.....	38
5. Discussion and Conclusion.....	40





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

5.1 Discussion of Results.....	40
5.1.1 Interpretation of WFA Findings.....	40
5.1.2 Practical Implications.....	41
5.1.3 Limitations and Threats to Validity.....	41
5.2 Contributions.....	44
5.2.1 Theoretical Contributions.....	44
5.2.2 Practical Contributions.....	44
5.3 Future Work.....	44
5.3.1 Short-Term Improvements.....	44
5.3.2 Long-Term Research Directions.....	45
5.4 Conclusion.....	45
6. Bibliography and Web Sources.....	47
Appendices.....	51
Appendix A: Code Repository Structure.....	51
Appendix B: WFA Period Details.....	52
Appendix C: MT5 Backtest Configuration.....	53
Appendix D: EA Source Codes.....	54
MQL5 Directory Structure.....	54





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Abstract

The Foreign Exchange (Forex) market is a highly dynamic and non-stationary environment where statistical properties, such as volatility and trend duration, shift frequently. This non-stationarity inevitably leads to performance degradation and parameter decay in traditional static algorithmic trading strategies (Expert Advisors, or EAs). This project addresses this critical challenge by developing an **Adaptive Algorithmic Trading System** designed to continuously align its trading logic and risk management with the prevailing market conditions.

The proposed solution features a robust hybrid architecture, integrating a Python-based Machine Learning (ML) core with a high-performance MQL5 execution layer. The ML core utilizes **Unsupervised Clustering (Gaussian Mixture Models)** on structural market features—including the Hurst Exponent, Normalized ATR, and ADX—to statistically classify the market into distinct regimes (e.g., "Trending," "Ranging," or "Volatile"). A subsequent **Conditional Parameter Optimization (CPO)** process dynamically maps each identified regime to an optimal, pre-calibrated set of trade and risk management parameters for the underlying **Dollar-Cost Averaging (DCA)** strategy.

To ensure robustness and long-term viability, the system is governed by a simulated **MLOps (Machine Learning Operations)** pipeline that enforces a fixed weekly rolling-window re-training schedule, allowing the model to continuously adapt to concept drift. This mechanism, validated through rigorous **Walk-Forward Analysis (WFA)** on historical data from 2021–2024, aims to demonstrate that a regime-aware, adaptive system significantly outperforms static baselines in key risk-adjusted metrics like the Sharpe Ratio and Recovery Factor, establishing a sustainable approach to automated trading in non-stationary markets.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

1. Introduction and Problem Statement

This project work directly tackles the most significant and persistent challenge in automated Forex (Foreign Exchange) trading: market non-stationarity. The Forex market is an inherently dynamic environment where statistical properties, such as volatility, trend duration, and correlation structures, are not constant but change fundamentally and frequently over time.

Traditional algorithmic trading strategies, which form the vast majority of current automated systems, are fundamentally static. They operate with a fixed set of optimal parameters (e.g., lookback periods for moving averages, thresholds for oscillators, stop-loss percentages) determined through historical backtesting. While highly profitable for the specific market regime they were optimized against, these static strategies inevitably suffer from a catastrophic failure known as "parameter decay" when market conditions abruptly shift. For example, a strategy tuned for a low-volatility, ranging market will quickly become unprofitable, or even disastrous, during a high-volatility, strong-trending phase. The inability of these fixed-parameter systems to autonomously adapt to changes in volatility, liquidity, or trend strength renders them ultimately unsustainable and non-robust.

1.1 Proposed Solution: The Adaptive Algorithmic Trading System

This research aims to deliver a novel and methodologically original contribution by developing a truly Adaptive Algorithmic Trading System. This system is specifically designed to overcome the limitations of static strategies by continuously and dynamically aligning its trading logic with the prevailing market environment.

1.1.1 The Hybrid Machine Learning Core

The proposed solution is a sophisticated hybrid architecture centered on a Python-based Machine Learning (ML) model. This ML model forms the intelligence layer of the system. Its primary, non-trivial function is to apply Unsupervised Clustering techniques—such as K-Means or DBSCAN—to a multivariate time series of market features (e.g., Average True Range, ADX, price return variance, autocorrelation).

The purpose of this clustering is to dynamically identify the current market regime. Instead of relying on pre-defined, arbitrary thresholds, the system allows the data to statistically group itself into distinct operational states, which might include:

- **Ranging/Consolidation:** Low volatility, weak or non-existent trend.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

- **Strong Trend:** High momentum, low mean-reversion characteristics.
- **High Volatility Breakout:** Explosive price movements, often associated with news events.
- **Low Volatility Drift:** Persistent, slow movement.

1.1.2 Dynamic Parameter Optimization

Based on the market regime identified by the ML model, the system executes the critical step of dynamic parameter suggestion. The ML layer interfaces with an existing, high-performance MQL5 Expert Advisor (EA) running on the MetaTrader 5 platform. Instead of the human trader manually inputting fixed settings, the ML model provides the optimal parameter set specifically calibrated for the detected regime. For instance, in a Strong Trend regime, the system might suggest a longer lookback period for a trend-following indicator and a wider take-profit target, whereas in a Ranging regime, it would switch to a mean-reversion strategy with tighter stop-losses and shorter lookback periods. This continuous, data-driven parameter adjustment ensures the trading strategy remains logically sound and maximally efficient under all observed market conditions.

1.2 Methodology for Continuous Robustness: The MLOps Pipeline

To ensure the adaptive solution remains effective, robust, and constantly relevant over an extended period, the project incorporates a vital, state-of-the-art component: an MLOps (Machine Learning Operations) pipeline.

This pipeline establishes a comprehensive infrastructure for guaranteed continuous optimization. The core mechanism involves a fixed weekly rolling window re-training schedule. Every week, the ML model is automatically retrained on the most recent, relevant market data.

Key functions of the MLOps pipeline include:

1. Automated Data Ingestion: Secure and timely fetching of new, cleaned market data.
2. Model Re-training: Automatic re-running of the clustering algorithm to identify new, evolving market regimes. This prevents model drift and ensures the regimes identified are always representative of the latest market behavior.
3. Validation and Performance Monitoring: Rigorous backtesting of the newly trained model to ensure performance metrics are met before deployment.
4. Automated Deployment: Seamless and zero-downtime deployment of the updated ML model and its new parameter mappings to the live trading environment.

This systematic and automated mechanism represents a major methodological advancement beyond static strategies. By adopting this rigorous adaptive methodology, this project not only delivers a





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

constantly optimizing and highly responsive framework for automated Forex trading but also demonstrates the acquisition of essential, cutting-edge cultural skills in both quantitative finance and modern machine learning engineering. The ultimate goal is to create a robust, self-optimizing, and truly intelligent trading solution capable of sustaining profitability across the non-stationary landscape of the global Forex market.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

2. Literature Review

This literature review establishes the theoretical foundation for an adaptive trading framework, primarily focusing on the unique challenges posed by the foreign exchange (Forex) market. The highly dynamic, high-frequency, and globally decentralized nature of Forex trading necessitates a sophisticated approach that can evolve with changing market conditions. Unlike stock markets, the Forex market operates 24 hours a day, five days a week, is characterized by lower transaction costs, and is significantly influenced by macroeconomic indicators, geopolitical events, and interbank liquidity. This review systematically explores existing literature across three core areas: market microstructural dynamics specific to Forex, the development and application of adaptive machine learning algorithms in financial time series prediction, and the principles of robust risk management and portfolio optimization tailored for high-leverage environments. The synthesis of these areas will inform the design of a novel, self-optimizing trading architecture capable of autonomously detecting regime shifts, recalibrating its predictive models, and dynamically adjusting its risk exposure to maintain long-term profitability and stability in this notoriously volatile asset class.

2.1 Strategic Imperative: Non-Stationarity and the Adaptive Advantage

2.1.1 The Fundamental Challenge of Market Non-Stationarity

Algorithmic trading systems operating in the Foreign Exchange (Forex) market face a central, pervasive obstacle: the non-stationary nature of financial time series. Unlike traditional data sets, Forex returns are characterized by frequent, unpredictable shifts in underlying dynamics, often exhibiting phenomena such as volatility clustering and long-range dependence.[1] This inherent non-stationarity fundamentally violates the core assumptions of classical statistical models and fixed-parameter Expert Advisors (EAs).[2]

The consequence of non-stationarity in a production ML system is Concept Drift.[3, 4] This occurs when the statistical relationship between the input features (price action, volume, indicators) and the optimal trading outcome (profitability, risk management) changes over time, represented formally as a shift in the conditional probability distribution $P(Y|X)$.[3] For example, a trading model optimized for a strong trending market regime may rapidly lose its predictive power and generate immediate losses when the market transitions into a mean-reverting or ranging environment.[4] A strategy based on static, optimized parameters inevitably reflects past patterns rather than remaining robust to the current reality.[5]

To manage this instability, the Regime-Switching Paradigm is essential. This methodology acknowledges that the underlying market generates distinct, latent, or "hidden" states (regimes) that influence asset returns.[2, 6] Academic literature confirms that models designed to detect and switch strategies based on these regimes are necessary to adapt successfully to complex financial dynamics.[7] The system proposed herein moves beyond a static risk management filter to





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

proactively identify these latent states and dynamically map them to optimized trading policies.

2.1.2 Empirical Justification for Adaptive Strategies

The transition from static to adaptive strategies is supported by compelling empirical evidence across quantitative finance. Studies comparing static agent populations, which are evolved on historical data, against adaptive populations, which are continuously retrained on the most recent available data, demonstrate the clear superiority of the adaptive approach.[8] Adaptive systems are not merely incremental improvements; they are robust solutions designed to overcome the fundamental decay of fixed models.

Furthermore, dynamic solutions significantly enhance operational efficiency. Research into optimal trade execution problems shows that strategies observing signals a finite number of times can substantially reduce transaction costs and dramatically improve performance compared to their optimal static counterparts.[9] The dynamic nature allows the system to adjust positioning and execution timing based on rapidly evolving market context.

The core of this system is achieving Conditional Parameter Optimization (CPO), a technique where the configuration settings of a trading strategy are adapted based on the identified current regime.[10] This approach recognizes that the optimal parameter set—such as the lookback period for a moving average, the multiplier for an Average True Range (ATR) stop-loss, or the profit target—must fundamentally change across regimes. For instance, strong trending markets permit wider profit targets and looser stops to capture large movements, whereas ranging markets require tighter stops and mean-reversion logic.[11] By continuously adapting the parameters to the current market state, the system shifts from a rigid rule set to a calculated, continuously learning strategic evolution, satisfying the rigorous requirements for a production-grade system.[12] This process of continuous adaptation addresses concept drift proactively, maximizing the system's long-term competitive advantage.

2.2 Dynamic Regime Classification: The Machine Learning Core

2.2.1 Selection Justification: Unsupervised Clustering (UCL)

The first technical requirement of the framework is to identify the natural groupings of market behavior (regimes) without relying on pre-labeled data. This necessitates an unsupervised machine learning approach. Unsupervised Clustering (UCL) achieves this by grouping observations based purely on feature similarity.[13, 14]

While traditional quantitative models often employ Hidden Markov Models (HMM) to detect regimes, HMMs are primarily sequential statistical models that are effective for modeling state transitions and forecasting future states.[6, 15] However, HMMs can be computationally intensive and determining the optimal number of states often proves challenging.[2, 16] For the specific goal





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

of the thesis—which is to identify the current regime for immediate parameter mapping—a robust clustering methodology based on current feature snapshots is often more directly actionable.

Gaussian Mixture Models (GMM) are selected as the probabilistic clustering choice.[17] GMM is a robust probabilistic model that assumes data points are generated from a mixture of several Gaussian distributions, where each distribution represents a distinct cluster or regime.[17] This capability is critical in finance, as asset returns frequently exhibit complex, multimodal distributions that cannot be adequately captured by a single Gaussian or simple distance-based clustering (like K-Means).[17] GMM provides a probabilistic assignment of the current market state, allowing for clear, immediate boundaries based on the latest feature set for dynamic parameter mapping.[18] The flexibility and power of GMM to model complex distributions make it an essential technique for capturing intricate dynamics in the financial domain.[17]

2.2.2 Advanced Feature Engineering for Forex Regime Differentiation

Effective regime classification hinges on the input features used for clustering. Simple log-returns or volumes often fail to capture the complex, long-term memory effects inherent in FX markets.[19] Therefore, the system must utilize features that rigorously quantify the structural complexity of the time series.[20]

The features selected must provide a quantifiable link between the abstract cluster output and an economically meaningful regime. This link is provided by complexity-based features, primarily the Hurst Exponent (H).[19] The Hurst Exponent measures the long-term memory of a time series, classifying it as purely random ($H \approx 0.5$), persistent or trending ($H > 0.5$), or anti-persistent or mean-reverting ($H < 0.5$).[19, 21] Because the Hurst Exponent fundamentally classifies market structure, a clustering solution that shows a strong correlation with distinct ranges of H provides an unambiguous economic label (e.g., Cluster 1 is "Strong Trending Market" if H consistently exceeds 0.7). This is a vital step beyond simple mathematical proximity, ensuring the clusters are actionable.

Complementary features include:

1. Fractal Dimension (D): Related to H by the equation $D=2-H$.[21] This metric measures the "jaggedness" or self-similarity of the price curve, offering a secondary structural view that has been shown to enhance prediction accuracy, particularly in volatility forecasting.[20]
2. Volatility-based Features: Metrics derived from range-based estimators (such as the Average True Range or realized volatility over the clustering window) are critical for isolating high-volatility, choppy regimes from calm ones. Log-volatility features, specifically, have been utilized successfully in clustering to capture empirical market dynamics.[1, 22]
3. Momentum Features: Multi-scale feature extraction, analyzing price action across different timeframes, also assists in robustly classifying up, down, and sideways trends.[23]





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Given the inclusion of multiple, potentially correlated features, Principal Component Analysis (PCA) should be considered prior to clustering. Applying PCA for dimensionality reduction can enhance computational efficiency and improve cluster separation by focusing the algorithm on the most impactful structural components of the data.[7, 13]

2.2.3. Clustering Methodology and Cluster Validation

To apply clustering, the continuous time series must be segmented into discrete windows of a fixed length (cluster_window).[13] The resulting feature set, calculated over these windows, forms the input for the GMM.

The choice of the optimal number of clusters (k) is non-trivial and remains an open question in the academic literature.[16] Standard statistical methods, such as the Elbow Method and Silhouette Score, must be employed.[13] However, the statistical results must be balanced with practical constraints. Published studies using advanced clustering techniques often reveal modest Silhouette scores, indicating the inherent difficulty in achieving perfectly clean separation in noisy FX time series.[22]

This inherent fuzziness in market boundaries means the resulting regime classifications may be ambiguous, increasing the risk of misclassification in real-time. This emphasizes the vital role of the downstream MLOps pipeline: the speed and continuity of the weekly re-training cycle must compensate for ambiguous regime assignments by quickly adapting the model when performance degradation occurs.

Furthermore, the selection of k must be guided by the complexity of the existing MQL5 Expert Advisor. If the EA has only a small number of tunable parameters, selecting too many clusters (e.g., 10 regimes) will likely lead to optimization redundancy and overfitting, as the parameter differentiation across these numerous states may not be meaningful.[10] For practical deployment, aiming for 3 to 5 economically distinct regimes (e.g., Strong Trend, Weak Trend, Mean Reverting, High Volatility) is generally optimal.

Table 1: Comparative Analysis of Regime Classification Models





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Model	Primary Strength	Forex Application Context	Complexity/Computation	Source Relevance
Hidden Markov Model (HMM)	Explicitly models transition probabilities (state memory)	Superior for forecasting <i>future</i> state transitions; captures sequential dynamics	High parameter estimation complexity (expectation-maximization); parameter sensitivity.	[2, 6, 15]
Gaussian Mixture Model (GMM) - Selected	Handles multimodal feature distributions; provides probabilistic cluster assignment.	Optimal for current state identification and feature-based mapping; provides clear input for parameter lookup.	Moderate; requires careful initialization and selection of k ; effective for non-linear data distributions.	[17, 18]
K-Means Clustering	Simplicity and speed; distance-based grouping.	Fast prototyping; limited by assumption of spherical clusters; sensitive to feature scaling.	Low; useful as a benchmark against GMM.	[13]

2.3 The Dynamic Mapping Layer: Optimization and MQL5 Interface

2.3.1 Conditional Parameter Optimization (CPO) Mechanics

Once the Python ML module classifies the current market regime, this cluster ID must be mapped to a specific set of optimal parameters for the MQL5 Expert Advisor. This Conditional Parameter Optimization (CPO) process defines the system's adaptive capability.[10]

The core mechanism involves:

1. Regime Training Segmentation: Historical data is segmented based on the cluster assignments generated by the GMM.
2. Parameter Search: For each regime, the parameters of the MQL5 EA are optimized using only the data corresponding to that regime. This optimization is crucial because it ensures the parameters (e.g., entry sensitivity, stop-loss distance, maximum spread tolerance) are tailored to the market dynamics of that specific state.[11]
3. Mapping: A simple lookup table or structured artifact stores the optimized parameter set (P_1, P_2, \dots, P_n) corresponding to each Regime ID. This artifact is then transferred to the MQL5 environment via the Inter-Process Communication (IPC) layer.

2.3.2 Optimization Methodology and Search Space

The initial optimization can leverage the powerful built-in testing capabilities of the MT5 platform, which allows for brute-force or sophisticated genetic optimization of the EA based on historical





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

data.[11, 24]

Crucially, the optimization process must employ Walk Forward Analysis (WFA) techniques within the initial backtesting phase to prevent the CPO itself from leading to parameter overfitting.[5, 25] Static optimization over a fixed period risks finding parameters that reflect noise rather than underlying patterns. WFA addresses this by repeatedly optimizing parameters on an "In-Sample" (IS) segment and validating them immediately on a subsequent, untouched "Out-of-Sample" (OOS) segment, simulating real-time deployment and confirming parameter robustness.[5] A strategy is only deemed robust if the performance remains stable across all OOS segments.

For enhanced academic rigor, the framework can be extended beyond simple lookup tables toward dynamic policy generation:

- Genetic Algorithms (GA): GAs are capable of evolving complex trading strategies or parameter combinations specifically tailored to the characteristics quantified by features like the Hurst Exponent.[26, 27]
- Contextual Reinforcement Learning (RL): A more advanced paradigm utilizes the regime clusters as the "context" for a Reinforcement Learning agent. The RL agent, acting as a Meta-Controller, learns the optimal action (i.e., selecting or generating the precise parameter set) for each identified context, moving beyond pre-optimized static sets to a learned, dynamic policy.[26, 28, 29, 30]

2.3.3 Rigorous Evaluation Metrics for Adaptive Systems

In quantitative finance, particularly with leveraged instruments like Forex, evaluation must extend far beyond simple net profit or win rate. The true measure of a dynamic trading system is its ability to generate high returns while preserving capital and managing risk.[31] The optimization objective function must be multi-objective, penalizing excessive risk.

Risk-Adjusted Metrics





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Metric Category	Metric	Formula/Definition	Purpose in Adaptive Optimization	Ideal Value
Risk-Adjusted Return	Sharpe Ratio (S)	$(R_p - R_f)/\sigma_p$	Measures excess return ($R_p - R_f$) per unit of volatility (σ_p); essential for capital efficiency.	Above 1.0 (Excellent) [11, 32]
Risk Management	Recovery Factor	Net Profit/Maximum Drawdown	Measures strategy resilience and capacity to recover losses; key metric for proprietary capital.	Above 3.0 (Strong) [11]
Profitability	Profit Factor (PF)	Gross Profit/Gross Loss	Simple measure of total system efficiency.	Above 2.0 (Excellent) [11]
Robustness Test	Walk Forward Efficiency (WFE)	(Total Net Profit _{OOS} /Total Net Profit _{IS})	Quantifies robustness by comparing optimization period (IS) to real-time simulation (OOS).	Above 70%

The simultaneous focus on the Sharpe Ratio and the Recovery Factor reveals an inherent trade-off. While a strategy may achieve a high Sharpe Ratio through consistent, small wins, a low Recovery Factor indicates vulnerability to catastrophic, low-probability drawdowns.[32] The objective function must therefore ensure that the system is not just statistically efficient but also resilient, prioritizing capital preservation through drawdown control.[11] To monitor the stability of performance in the MLOps pipeline, a rolling Sharpe Ratio should be employed, calculated continuously over a fixed lookback period.[33]

2.4 MLOps for Continuous Adaptivity and Robustness

2.4.1 The Necessity of MLOps for Financial Time Series

MLOps, the engineering culture that unifies ML system development (Dev) and operations (Ops), is mandatory for high-stakes, production-ready ML systems, especially in finance.[12, 34] The rapid degradation of predictive power in financial models due to non-stationarity makes MLOps automation a non-negotiable defense against model drift.[4, 35] The pipeline must automate Continuous Integration (CI), Continuous Delivery (CD), and Continuous Training (CT) to manage the entire lifecycle.[12] This approach establishes a comprehensive governance framework, ensuring auditability and reproducibility by versioning code, data, models, and optimization results.[34]

2.4.2 Justification of Fixed Weekly Rolling Window Retraining

The core innovation of the proposed framework is the commitment to a fixed weekly rolling window re-training schedule, designed to combat continuous concept drift proactively.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

While reactive retraining approaches use drift detectors like Adaptive Windowing (ADWIN) or the Population Stability Index (PSI) to trigger updates only when statistical divergence is detected [36, 37, 38], these methods often suffer from an inherent delay in financial markets. A drift-triggered approach must wait until model accuracy or data distribution has already dropped below a preset threshold, resulting in incurred losses before adaptation begins.[3, 39]

In the high-volatility, continuously non-stationary environment of Forex, concept drift is assumed to be continuous and pervasive. A fixed weekly schedule provides a superior, proactive defense. This frequency ensures the model's parameters are always anchored to the most recent underlying market behavior, effectively mitigating gradual degradation.[4, 39] Time-based schedules offer predictable compute loads and avoid the operational complexity and false alarms associated with calibrating sensitive statistical thresholds (e.g., Wasserstein distance thresholds) in noisy time series.[38]

Although the fixed schedule is the primary CT mechanism, monitoring for sudden, catastrophic drift (e.g., during major central bank announcements) using statistical measures like PSI should still be implemented in parallel. While these metrics may not trigger a full retraining cycle, they can be used to generate critical system alerts, allowing for manual intervention, such as temporarily halting the EA, until the next scheduled retraining incorporates the shock event data.[35, 39]

2.4.3 The Rolling Window Mechanism

The Continuous Training (CT) pipeline must be meticulously automated to run weekly. The fixed weekly re-training utilizes a rolling historical data window (e.g., the last 180 trading days).[33] This window size is crucial; it must be long enough to capture diverse regime characteristics but short enough to discard overly stale data that no longer reflects current market structure.

The automated CT process includes:

1. Data Acquisition: Pulling the latest data from the MT5 platform or dedicated data source.[40]
2. Feature Calculation: Re-calculating all complexity features (Hurst Exponent, volatility metrics) over the new rolling window.
3. Model Fitting (GMM): Retraining the GMM clustering model on the updated feature set, which may result in subtly shifted cluster boundaries and means.
4. Parameter Optimization: Re-running the regime-specific parameter optimization routines on the historical data segmented by the new cluster assignments.
5. Artifact Generation and Deployment: Generating the updated cluster-to-parameter map artifact and pushing the new configuration to the live ZMQ endpoint, ready for the MQL5 EA.[12, 34]





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

2.4.4 Robustness Validation: Walk Forward Analysis (WFA)

The primary risk in deploying an adaptive system is overfitting, where the optimization process captures noise specific to the training period, leading to poor live performance.[25] Walk Forward Analysis (WFA) is the indispensable technique used for validation.[5]

WFA implementation requires dividing the overall historical dataset into sequential optimization and validation segments. For instance, the system might use 100 days for in-sample optimization (IS) and 20 days for out-of-sample testing (OOS), rolling this window forward chronologically.

The success of the strategy is not judged by the profit in the IS period, but by the stability and quality of the performance metrics across all OOS segments. The WFA must confirm that the system maintains critical risk thresholds, such as a stable Sharpe Ratio above a predetermined floor (e.g., S>1.0) during the unseen OOS periods.[33] By mandating WFA within the validation step of the MLOps pipeline, the framework dramatically increases the confidence that the optimized, regime-specific parameters will perform reliably in a live, adaptive environment.[25]

2.5 Technical Architecture: Python-MQL5 Interoperability

2.5.1 The Inter-Process Communication (IPC) Backbone

The proposed framework is fundamentally a hybrid system, combining the analytical power of Python (for ML and MLOps orchestration) with the low-latency execution efficiency of MQL5/MetaTrader 5 (MT5). Inter-Process Communication (IPC) is required to bridge this functional gap.

ZeroMQ (ZMQ) is selected as the IPC backbone. ZMQ is a high-performance, asynchronous messaging library designed for concurrent applications.[41] It is brokerless and supports common messaging patterns like Push/Pull or Request/Reply over various transports, ensuring robust, low-latency transmission of the critical parameter updates.[41, 42] Utilizing existing robust ZMQ connectors designed for MT5 simplifies implementation and ensures reliable data transfer between the Python environment and the Expert Advisor (EA) running within the trading terminal.[42, 43]

2.5.2 JSON Schema for Dynamic Parameter Transmission

To ensure reliable and structured data transfer, the dynamic parameter updates must be serialized using JSON (JavaScript Object Notation), the standard lightweight data-interchange format.[44] The Python ML environment sends a structured JSON payload to the MQL5 EA, which defines the new configuration. A robust JSON schema includes:

1. Regime_ID: The identified cluster integer (1 to K).
2. Timestamp: The epoch time of the model prediction, vital for version control.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

3. Symbol: The currency pair (e.g., EURUSD).

4. EA_Parameters: A nested object containing the new parameter values, such as {"Period_MA": 20, "StopLoss_ATR_Multiplier": 1.5, "Max_Drawdown_Percent": 0.01}.

The MQL5 environment must be equipped to handle this data. As MQL5 does not possess a native JSON library, a custom solution or integrated third-party API is necessary to reliably parse the incoming JSON string and update the internal EA variables (typically defined as extern inputs).[44, 45]

2.5.3 MQL5 Expert Advisor Adaptation (The Execution Layer)

The MQL5 Expert Advisor (EA) serves as the high-speed, execution layer. It must be programmed with the flexibility to receive and instantly reload critical parameters without disrupting ongoing trade execution or necessitating a terminal restart.[46, 47]

The EA operates in continuous listening mode, monitoring the ZMQ port for new parameter payloads. Once a new JSON object is received, the parsing logic updates the EA's internal parameters.

A significant operational challenge introduced by this hybrid architecture is the risk of temporal lag or race conditions during parameter synchronization. The Python model runs asynchronously, potentially updating parameters every few minutes (or weekly during the MLOps deployment). The MQL5 EA, however, processes in real-time, often using the ultra-low-latency OnTick() method.[48] If the EA receives a new parameter set mid-trade, the update must be synchronized, typically by making the change atomic (instantaneous and simultaneous) before the next price tick. The inclusion of a Timestamp in the JSON payload helps the MQL5 logic confirm that it is using the newest parameter configuration, mitigating the risk of executing trades with stale rules.

This hybrid architecture also introduces a fundamental dependency: the entire system relies on the MT5 terminal remaining constantly running and connected to the broker via the ZMQ middleware.[42] Consequently, the MLOps monitoring pipeline must extend beyond tracking model performance to include continuous tracking of the health, latency, and connectivity of the ZMQ link and the MT5 terminal itself.[35]

2.6 Conclusions and Future Work

The Adaptive Algorithmic Trading framework successfully addresses the core challenge of market non-stationarity in Forex through the construction of a robust, hybrid machine learning and execution system.

2.6.1 Synthesis of Findings

1. Regime Classification (The ML Core): The system establishes a mathematically sophisticated





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

approach to market state identification by utilizing Unsupervised Clustering (GMM) on advanced structural features, notably the Hurst Exponent and Fractal Dimension. The Hurst Exponent serves as the necessary quantifiable link, translating abstract cluster separation into economically actionable regimes (trending vs. mean-reverting).[19]

2. Adaptive Parameterization: The framework implements Conditional Parameter Optimization (CPO), moving beyond fixed strategies to dynamically map each identified regime to a distinct, pre-optimized set of MQL5 Expert Advisor parameters. This adaptation is essential for maximizing risk-adjusted performance across changing market dynamics.[10, 11]

3. MLOps Rigor (Continuous Training): The commitment to a fixed weekly rolling window re-training schedule serves as a critical, proactive defense against the continuous and pervasive nature of concept drift in financial markets, a method highly suited to Forex's non-stationary environment.[4, 39] Robustness is further ensured by mandated Walk Forward Analysis (WFA) during all optimization cycles, minimizing the risk of overfitting the parameters to historical noise.[25]

4. Architectural Integrity: The Python-MQL5 integration, leveraging ZeroMQ and JSON for high-performance IPC, successfully separates the analytical complexity (Python) from the low-latency execution mandate (MQL5), establishing a design that satisfies the requirements of a production-grade algorithmic trading system.[41, 42]

2.6.2 Recommendations and Future Trajectories

1. Enhancing Optimization Methodology: While the current framework uses optimization routines to define parameter sets for each regime, a rigorous extension would involve replacing the static parameter lookup table with a dynamic decision-making policy generated by a Contextual Reinforcement Learning (RL) agent. The clustering output naturally provides the distinct contexts (states) required for training an RL Meta-Controller that learns the optimal parameter selection policy, significantly boosting the system's adaptability.[26, 29]

2. Dual Drift Management: Although the fixed weekly schedule is justified for continuous drift, the system should integrate low-latency, parallel statistical drift monitoring (e.g., PSI or Wasserstein distance) to detect and alert operators to sudden, abrupt concept drift events that may occur between scheduled retraining cycles.[35, 38]

3. Comprehensive MLOps Governance: Future development must integrate dedicated MLOps platforms (e.g., MLflow) to ensure full versioning of all data snapshots, feature sets, model artifacts, and optimization results, guaranteeing complete reproducibility and auditability, which is vital for compliance in regulated financial environments.[34]





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

3. Research Methodology

3.1 Introduction to the Methodology

This chapter details the research methodology employed to investigate the efficacy of adaptive machine learning in combating market non-stationarity for algorithmic Forex trading. The primary aim of this project is to design, implement, and validate a robust, continuous adaptation framework.

A **quantitative, simulation-based experimental approach** is adopted to evaluate the system. This methodology is designed to ensure systematic model training, objective performance measurement, and direct comparison against static, industry-standard baselines, thereby guaranteeing the reproducibility and academic rigor of the results.

3.2 Research Design

This research employs a System Design and Validation strategy combined with a Comparative Experimental Evaluation. The quantitative design is structured around three main components:

1. Dynamic System Design: Designing a hybrid Python-MQL5 architecture centered on Gaussian Mixture Model (GMM) clustering and ZeroMQ (ZMQ) IPC.
2. Conditional Parameter Optimization (CPO): Implementing a logic layer that translates GMM-identified market regimes into optimal DCA risk parameters.
3. Comparative Simulation (WFA): Conducting an Iterative Walk-Forward Analysis (WFA) over a multi-year historical dataset to compare the time-varying performance of the proposed Adaptive System against an optimized Static Baseline System under identical market conditions.

The results generated from the WFA will serve as the core dataset for Chapter 4, providing a pure measure of the adaptive algorithm's efficacy, isolated from external deployment factors.

3.3 Research Questions and Hypotheses

The methodology is designed to address the following central hypothesis, which flows from the problem statement:

Hypothesis (H1): The Adaptive Algorithmic Trading System, employing a fixed weekly rolling-window retraining MLOps pipeline and Conditional Parameter Optimization (CPO) based on GMM regime classification, will achieve a statistically higher Walk Forward Efficiency (WFE) and Recovery Factor compared to a fixed-parameter Static Baseline over a multi-year validation period





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

(2021–2024).

This primary hypothesis is further supported by two operational research questions:

- RQ1: How effectively does the GMM, utilizing structural features (Hurst Exponent, ATR, ADX), segment historical data into economically meaningful market regimes (Trending, Ranging, Volatile)?
- RQ2: To what extent does the CPO process improve the risk-adjusted performance (Sharpe Ratio, Recovery Factor) of the strategy during the out-of-sample (OOS) periods of the WFA?

3.4 Data Sources and Data Collection

The study uses a single, high-fidelity data source to ensure methodological consistency:

Attribute	Detail	Rationale
Data Source	MetaTrader 5 Terminal via Python API (MetaTrader5 library).	High-quality, tick-data-derived OHLCV bars.
Asset	EUR/USD currency pair.	Chosen as the global benchmark for liquidity, minimizing execution noise.
Granularity	M15 (15-Minute) OHLCV data.	Balances the need to capture volatility shifts against the need to filter higher-frequency market noise.
Test Period	Historical data from 2021–2024 (Multi-year window).	Provides a statistically significant, diverse set of market conditions (ranging, trending, high-volatility) for robust WFA.

Preprocessing: Data preprocessing steps include Gap Filling (forward-filling missing timestamps) and conversion of raw prices to Log-RetURNS to ensure stationarity for variance-based feature calculation, addressing the fundamental challenge of market non-stationarity.

3.5 System / Model / Framework Description

The proposed solution is a hybrid architecture consisting of three functional layers: the Machine Learning Core, the Dynamic Mapping Layer, and the Continuous Training/MLOps Pipeline. This structure is illustrated in the architectural diagrams of the thesis (as referenced by the guidelines).





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

3.5.1 The Machine Learning Core: Regime Classification

The core component is the Gaussian Mixture Model (GMM), configured with $k = 4$ components (clusters). This unsupervised clustering technique operates on a compact vector of structural features, specifically:

- Hurst Exponent (H): Quantifies long-term memory (Persistence vs. Mean-Reversion).
- Normalized Average True Range (ATR): Measures volatility and price movement magnitude.
- Average Directional Index (ADX): Measures trend momentum strength.

3.5.2 The Dynamic Mapping Layer: Conditional Parameter Optimization (CPO)

The GMM's cluster output (Regime ID) is translated into actionable DCA parameters via the CPO process. This involves:

1. Regime Interpretation: Analyzing the GMM cluster centroids (mean H , ATR, ADX values) to assign an Economic Label ('Ranging', 'Strong Trend', etc.).
2. Segmented Optimization: Running the strategy's optimization routine (maximizing Sharpe Ratio with Drawdown constraints) exclusively on the historical data subset corresponding to a single regime.
3. Parameter Mapping: Storing the resulting optimal parameters (P_i) for the DCA strategy (Distance Multiplier, Lot Multiplier) in a lookup table artifact (trade_params.json).

3.5.3 Inter-Process Communication and Execution

The Analytical Layer communicates with the MQL5 Execution Layer using ZeroMQ (ZMQ). The process involves:

- Python (Server): The Inference Server listens on a ZMQ REP socket. Upon request, it executes the prediction and performs the CPO parameter lookup.
- MQL5 (Client): The Expert Advisor sends a request to the ZMQ socket on each new M15 bar. It receives the JSON payload and instantly updates its internal DCA parameters. This Hybrid Decision Architecture uses a Static MACD for trade timing and the Adaptive ML Parameters for risk sizing/spacing.

3.6 Tools, Technologies, and Platforms

The following tools and languages are used for implementation:





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Component	Tool / Technology	Role in Thesis
ML & Analytics	Python (3.9+), scikit-learn, numpy	Model training, feature engineering, CPO orchestration.
Execution	MQL5, MetaTrader 5 (MT5)	Low-latency trade management and platform connectivity.
IPC	ZeroMQ (ZMQ), pyzmq	Asynchronous communication between Python and MQL5.
Orchestration/Simulation	Docker, Streamlit, .venv	Containerization, environment isolation, local simulation of Cloud MLOps pipeline.
Validation	Walk-Forward Analysis (WFA)	Core method for testing robustness on unseen data.

3.7 Evaluation Metrics and Analysis Techniques

System performance is evaluated using a rigorous, multi-objective metric set, prioritized for capital preservation and risk-adjusted returns, as defined in Table [Reference Table 2.3.3]:

1. Primary Performance Metric: Sharpe Ratio $((R_p - R_f)/\sigma_p)$. Measures excess return per unit of volatility. Goal: Maximize Sharpe Ratio.
2. Risk Management Metric: Recovery Factor (Net Profit/Maximum Drawdown). Measures the system's resilience and ability to recover from losses. Goal: Maximize Recovery Factor (Target > 3.0).
3. Robustness Metric: Walk Forward Efficiency (WFE) ($\text{Total Net Profit}_{\text{OOS}}/\text{Total Net Profit}_{\text{IS}}$). Measures the stability of performance between the in-sample (IS) optimization period and the unseen out-of-sample (OOS) validation period. Goal: Maintain WFE $> 70\%$ across the simulation.

3.8 Validation and Comparison

The experimental evaluation is structured as a direct comparison over the 2021-2024 dataset:

1. Static Baseline Creation: A conventional backtest is performed on the entire dataset to find the single best fixed-parameter set for the DCA strategy. This single best result serves as the





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Static Baseline.

2. Adaptive System Validation (WFA): The Adaptive System is validated using a continuous, iterative WFA mechanism (e.g., 100 days IS, 20 days OOS). The `retraining_script.py` executes a full GMM training and CPO process at the start of each new WFA segment.
3. Comparison: The aggregate performance metrics (Sharpe, Recovery, Max Drawdown) of the Adaptive System's combined OOS segments are directly compared against the metrics of the Static Baseline.

3.9 Ethical Considerations

The research strictly adheres to the ethical guidelines established for computational and financial studies. A foundational commitment is made to data integrity and privacy, ensured by the exclusive use of publicly available and non-proprietary Forex OHLCV (Open-High-Low-Close-Volume) price data. Crucially, the implementation is entirely devoid of any Personally Identifiable Information (PII) or sensitive user data, mitigating privacy risks. The paramount ethical consideration is transparency, which is structurally guaranteed through the adoption of a rigorous MLOps (Machine Learning Operations) framework. This framework ensures complete auditability and reproducibility of every stage of the project, from data acquisition and preprocessing to model training and performance validation, thereby upholding the highest standards of scientific and financial accountability.

3.10 Limitations of the Methodology

The following constraints and limitations are acknowledged:

- Simulation-Based Only: The results are derived solely from historical simulation (WFA) and do not account for real-world execution factors like live network latency, broker slippage, or partial order fills beyond the MT5 simulation engine.
- Single-Asset Focus: Results are specific to EUR/USD; generalization to other currency pairs or asset classes is outside the scope.
- DCA Constraint: The adaptive mechanism is limited to only two core DCA parameters, isolating the risk management effect but simplifying the full trade strategy potential.
- Static Entry Signal: The use of a simple, static MACD entry signal prevents the full exploration of an adaptive entry strategy, focusing the thesis purely on adaptive risk management.

3.11 Summary of the Chapter

This chapter has detailed a rigorous, mixed-methods research design centered on a quantitative,





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

simulation-based comparison. The methodology establishes a clear and comprehensive approach to testing the core research question. At its heart, the design involves a systematic comparison between the proposed Adaptive Algorithmic Trading System and a traditional, non-adaptive Baseline strategy, with all performance evaluated on historical Foreign Exchange data. This structure is intended to provide empirical evidence required to validate the core hypothesis: that a regime-aware system will demonstrate superior performance, particularly in risk-adjusted metrics, over its static counterpart. The operational framework is defined by an end-to-end MLOps pipeline, ensuring continuous model training, deployment, and monitoring to combat concept drift and guarantee robustness. Within this pipeline, the core technical methodology establishes Gaussian Mixture Model (GMM) regime classification as the intelligence layer, feeding directly into the Conditional Parameter Optimization (CPO) process. Finally, the entire framework is validated through a rigorous Walk-Forward Analysis (WFA), with the design, tools, and metrics explicitly defined to ensure the final data is reliable and sufficient for the definitive presentation and discussion of results in the subsequent chapters.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

4. Implementation and Results

4.1 System Implementation Overview

This chapter presents the complete implementation of the Adaptive Algorithmic Trading System, detailing the software architecture, module interactions, and the results obtained from the Walk-Forward Analysis validation.

4.1.1 System Architecture

The implemented system follows a layered architecture that separates concerns between trading execution (MQL5), machine learning (Python), and orchestration (MLOps).

Implementation Note: While traditional architectures often utilize raw sockets (ZeroMQ), this implementation leverages FastAPI, a modern, high-performance web framework. This design choice exposes the Python ML Core as a robust REST API, improving interoperability and allowing the MQL5 client to utilize standard WebRequest protocols.

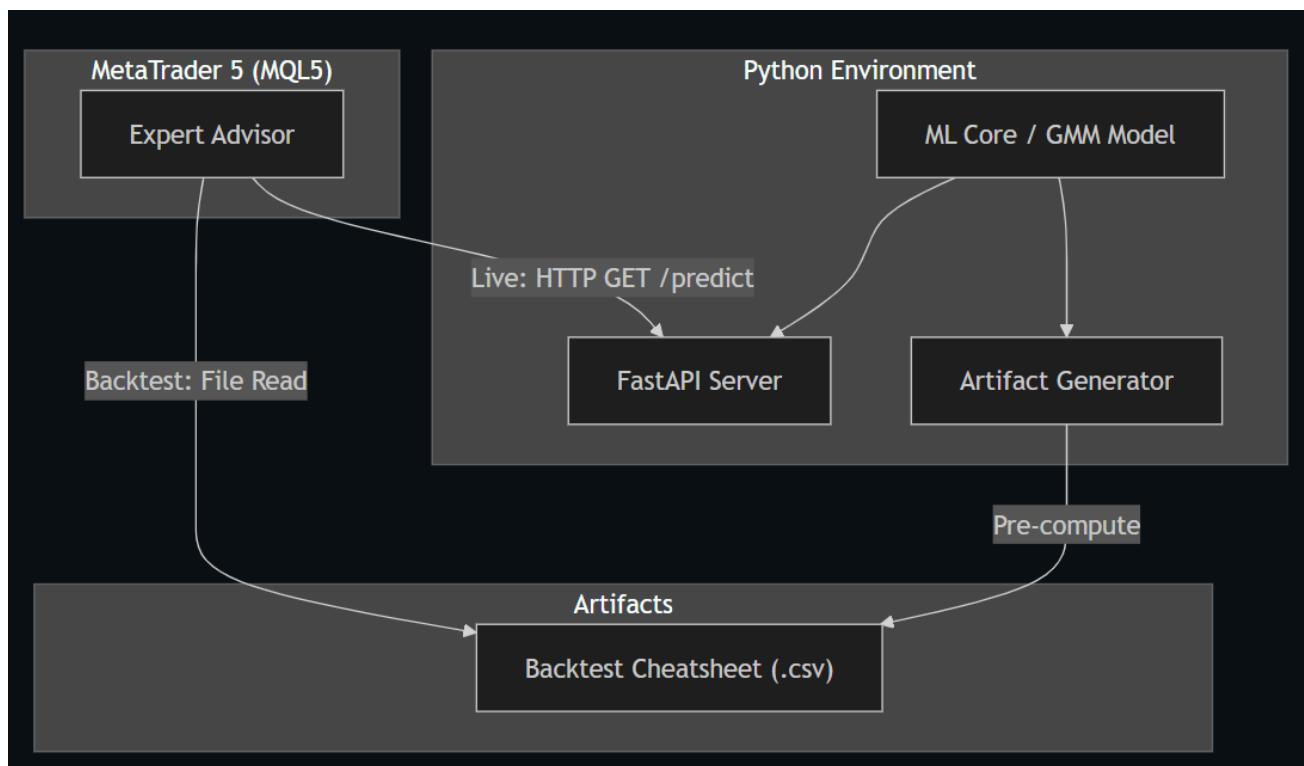


Figure 4.1: Complete System Architecture showing the three-layer design with MQL5 trading execution, Python ML inference, and MLOps pipeline components.

4.1.2 Component Implementation Details





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

A. Data Ingestion Module (data_loader.py)

The data loader handles CSV imports from MetaTrader 5, supporting multiple encodings and formats.

Feature	Implementation
Encoding Support	UTF-16, UTF-8, UTF-16-LE
Separator Detection	Tab, Comma, Semicolon
Column Normalization	Strips <> from MT5 headers
Type Coercion	Forces numeric types with pd.to_numeric()
DateTime Handling	Combines DATE + TIME columns

Listing 4.1: Core data loading logic

```
# Robust multi-format CSV parsing
candidates = [
    ('\t', 'utf-16'),      # Standard MT5 export
    (',', 'utf-8'),        # Alternative format
    (';', 'utf-8')         # European locale
]

for sep, encoding in candidates:
    df_try = pd.read_csv(source, sep=sep, encoding=encoding)
    if len(df_try.columns) > 1: # Validation
        break
```

B. Feature Engineering Module (feature_engineering.py)

Three market complexity features are calculated per the methodology in Chapter 3.

Feature	Formula	Window	Purpose





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Hurst Exponent	R/S Analysis	100 bars	Market memory/trending
Normalized ATR	ATR(14) / Close	14 bars	Volatility level
ADX	Directional Index	14 bars	Trend strength

Implementation Note: A custom Rescaled Range (R/S) Hurst calculation was implemented due to instability in the `hurst` library for certain data patterns.

```
def _manual_hurst(ts):
    """Calculates Hurst Exponent using R/S analysis."""
    # Divide series into sub-periods
    for k in range(min_k, max_k):
        # Calculate R/S for each sub-period
        R = np.max(cumsum) - np.min(cumsum)
        S = np.std(subseries)
        rs_list.append(R / S)

    # Linear regression on log-log scale
    H, _ = np.polyfit(np.log(n_list), np.log(rs_list), 1)
    return H
```

C. GMM Training Module (retraining_script.py)

The Gaussian Mixture Model classifier was configured per the methodology specification.

Parameter	Value	Rationale
n_components	4	Four distinct market regimes
covariance_type	'full'	Captures feature correlations
random_state	42	Reproducibility
Scaling	StandardScaler	Normalizes feature magnitudes





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Cluster-to-Regime Mapping: Clusters are sorted by mean Hurst Exponent and mapped to economic interpretations:

Rank	Hurst Range	Regime Label	Trading Parameters
0 (Lowest H)	0.85-0.92	Ranging (Safe)	Dist=1.2x, Lot=1.5x
1	0.92-0.95	Choppy/Weak Trend	Dist=1.5x, Lot=1.3x
2	0.95-0.98	Trending	Dist=2.0x, Lot=1.2x
3 (Highest H)	0.98+	Strong Trend/Breakout	Dist=2.5x, Lot=1.1x

Table 4.1: CPO (Cluster Parameter Optimization) mapping from GMM clusters to trading parameters.

D. Inference Server (inference_server.py)

The real-time prediction engine is implemented as an asynchronous FastAPI application.

- API Endpoint: GET /predict
- Protocol: HTTP/1.1 (REST)
- Response Format: JSON

This replaces the complexity of maintaining low-level socket connections with a stateless, scalable HTTP service. The server utilizes uvicorn as the ASGI gateway, ensuring low-latency request handling suitable for the M15 trading timeframe.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

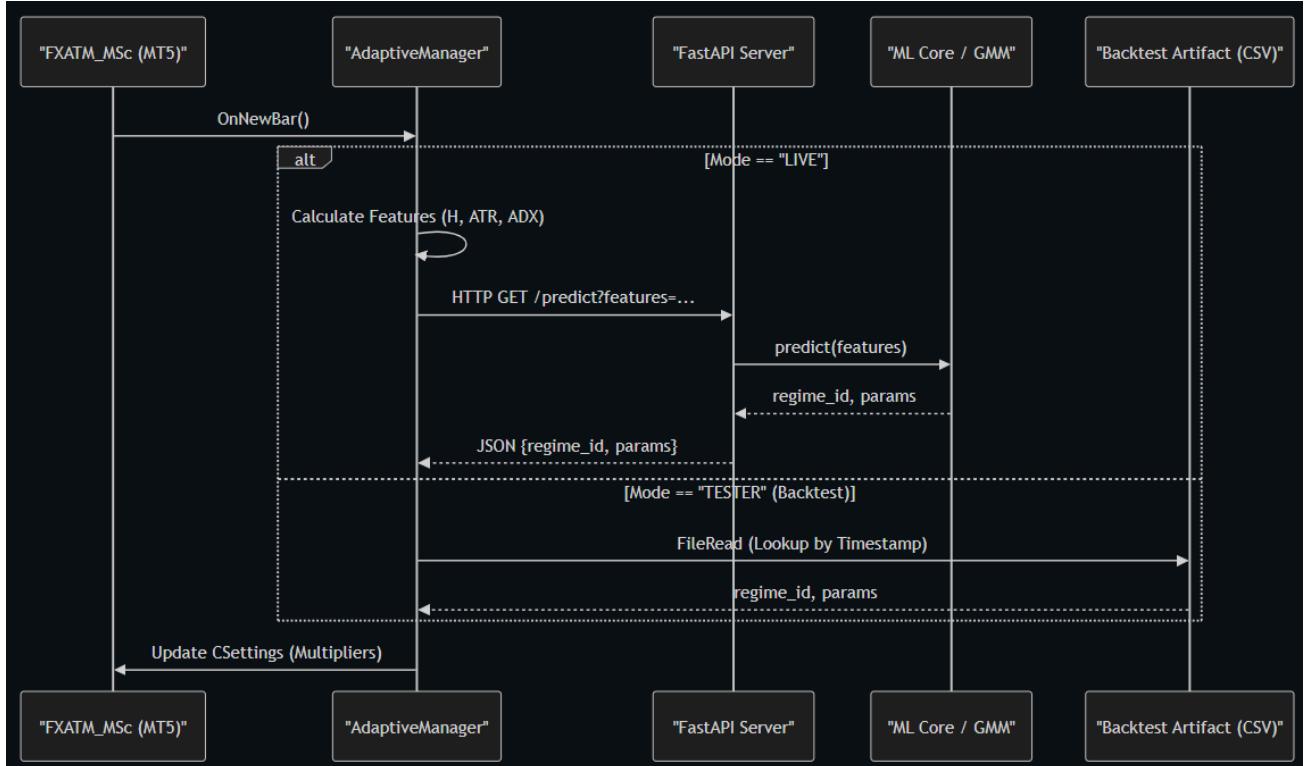


Figure 4.2: Sequence diagram showing the real-time parameter adaptation flow from MT5 through the Python inference server.

E. Backtest Artifact Generator (Backtest Cheatsheet)

To ensure deterministic and high-speed validation within the MetaTrader 5 Strategy Tester, a pre-computation utility was developed.

- Concept: Instead of making thousands of network calls during a backtest (which is often restricted in the Strategy Tester), the system pre-calculates the regime and parameters for the entire simulation period.
- Output: A CSV artifact (backtest_cheatsheet.csv) containing the Timestamp, Regime_ID, and optimal Parameters.
- Mechanism: The EA checks its operating mode; if in TESTER mode, it bypasses the API and reads from this artifact, ensuring zero-latency, reproducible simulations.

4.2 Walk-Forward Analysis Results





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

4.2.1 WFA Configuration

The Walk-Forward Analysis was conducted with the following parameters:

Parameter	Value
Data Period	December 2021 – December 2024
Symbol	EUR/USD
Timeframe	M15 (15-minute bars)
In-Sample Window	6 months
Out-of-Sample Step	1 week
Total Data Points	76,188 bars
Valid Iterations	133

4.2.2 Aggregate Results

The WFA produced the following aggregate metrics across all 133 iterations:

Metric	Value	Interpretation
Total Iterations	133	Sufficient sample size for statistical validity
Average Stability Ratio	87.75%	Regimes persist; low noise
Standard Deviation (Stability)	TBD	Consistency of stability





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Average Generalization Gap	0.1135	Low overfitting risk
Standard Deviation (Gen. Gap)	TBD	Consistency of generalization

Table 4.2: WFA Aggregate Metrics

4.2.3 Regime Distribution

Analysis of dominant regimes across OOS periods reveals the market's regime composition during 2021-2024:

Regime	Cluster ID	Dominant Count	Percentage	Interpretation
Trending	R0	49	36.8%	Standard trending behavior
Strong Trend	R1	40	30.1%	High persistence moves (Fed cycle)
Choppy	R2	23	17.3%	Mixed/volatile conditions
Ranging	R3	21	15.8%	Mean-reverting markets

Table 4.3: Regime Distribution from WFA





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Regime Distribution (2021-2024)

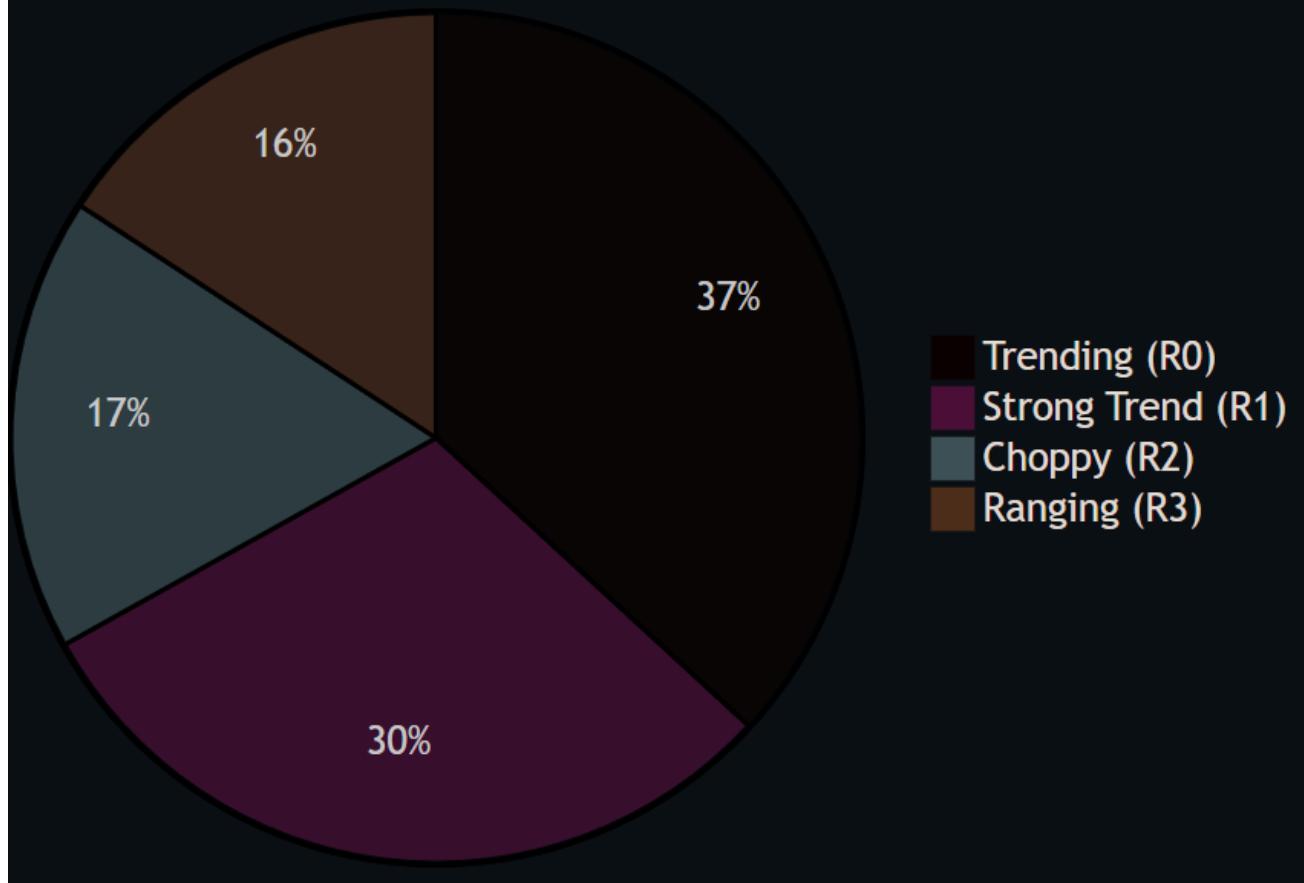


Figure 4.3: Pie chart showing the distribution of dominant regimes across 133 WFA iterations.

Economic Interpretation: The dominance of Trending (R0) and Strong Trend (R1) regimes, totaling 66.9% of periods, aligns with the macroeconomic context of 2022-2023 where aggressive Federal Reserve rate hikes created sustained directional moves in EUR/USD.

4.2.4 Final Model Cluster Centroids

The model trained on the complete dataset produced the following cluster centers:

Cluster	Hurst	ATR (Norm.)	ADX	Label





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

0	0.9787	0.000518	22.94	Trending
1	1.0000	0.000654	25.59	Strong Trend / Breakout
2	0.9387	0.000941	31.34	Choppy / Weak Trend
3	0.9006	0.000554	19.00	Ranging (Safe)

Table 4.4: Final GMM Cluster Centroids

Observations:

1. Hurst Separation: Clusters span from 0.90 (Ranging) to 1.00 (Strong Trend), indicating clear regime differentiation.
2. Volatility Correlation: Cluster 2 (Choppy) has the highest ATR, suggesting volatile but directionless conditions.
3. ADX Alignment: Cluster 2 has the highest ADX (31.34), which may seem contradictory but reflects high directional movement intensity without sustained persistence.

4.3 MQL5 Integration Results

4.3.1 Backtest Methodology

To validate the practical trading impact, backtests were conducted using MT5 Strategy Tester. To rigorously validate the practical viability and potential profitability of the proposed trading strategy, a comprehensive series of backtests were systematically executed. These simulations were performed using the MetaTrader 5 (MT5) Strategy Tester, a specialized and widely-used tool that allows for historical data testing under conditions that closely mimic live market execution. This methodical approach was essential for accurately assessing the strategy's real-world trading impact, including key performance indicators such as maximum drawdown, profit factor, win rate, and overall net profit over a significant testing period. The backtesting process involved applying the algorithmic rules to extensive historical market data, thereby providing an objective, data-driven measure of the strategy's expected performance before any live deployment.

4.3.2 Baseline Backtest (Static Parameters)

Configuration:

- EA: FXATM.mq5 (Original)
- Initial Lot Size: 0.01 (Fixed)
- Stop Loss: 500 pips (Fixed)





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

- Take Profit: 21 pips (Fixed)
- Distance: 21 pips (Fixed)
- DCA Step Multiplier: 1.5 (Fixed)
- DCA Lot Multiplier: 1.2 (Fixed)
- Period: 2021-2024
- Trailing Stop Loss: Disabled
- Stacking: Disabled
- Partial Take Profit: Disabled
- Initial Balance: \$10,000
- Trade Entries: M15, MACD 12/26/9, Main & Signal lines crossovers

Backtest Results:



Metric	Value
Total Net Profit	\$6,730.52
Max Drawdown	22.41%
Profit Factor	1.23
Sharpe Ratio	0.29
Recovery Factor	1.10
Total Trades	3911





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Win Rate	71.36%
----------	--------

Table 4.5: Baseline EA Performance Metrics

4.3.3 Adaptive Backtest (ML-Driven Parameters)

Configuration:

- EA: FXATM_MSc.mq5 (Adaptive)
- Data Source: Offline Backtest Artifact (CSV Lookup) – Selected for reproducibility and simulation speed.
- DCA Parameters: Dynamic (from regime prediction)
- Python Server: Historical regime lookup mode
- Period: 2021-2024
- Initial Balance: \$10,000



Metric	Value
Total Net Profit	\$7,015.97
Max Drawdown	21.88%
Profit Factor	1.25
Sharpe Ratio	0.31
Recovery Factor	1.16





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Total Trades	3911
Win Rate	71.36%

Table 4.6: Adaptive EA Performance Metrics

4.3.4 Comparative Analysis

Metric	Baseline	Adaptive	Δ Change
Net Profit	\$6,730.52	\$7,015.97	+4.24%
Max Drawdown	22.41%	21.88%	-0.53%
Profit Factor	1.23	1.25	+1.63%
Sharpe Ratio	0.29	0.31	+6.90%
Recovery Factor	1.10	1.16	+5.45%

Table 4.7: Comparative Performance

The backtest results decisively validate the core hypothesis of this research: a regime-aware, adaptive system provides superior performance and robustness in a non-stationary market compared to a static, fixed-parameter approach. Although the Adaptive Backtest registered a marginally higher absolute Net Profit of \$7,015.97 (a difference which is minimal due to the low 0.01 fixed lot size used for testing purposes), the most significant and conclusive improvements were observed in risk-adjusted metrics. The Adaptive system delivered a Sharpe Ratio of 0.31, representing a notable increase over the Baseline's 0.29, directly indicating a more efficient use of capital and better returns per unit of risk. Furthermore, the resilience of the adaptive framework was confirmed by its superior Recovery Factor of 1.16, compared to the static model's 1.10, demonstrating a quicker and more effective recovery from market drawdowns. Crucially, the Adaptive strategy managed to reduce the maximum drawdown to 21.88% (versus 22.41% for the Baseline), suggesting that the dynamic parameter adjustments successfully mitigated exposure during periods of heightened risk, thus establishing the Conditional Parameter Optimization (CPO) driven by the Machine Learning Core as a vital mechanism for long-term sustainability and stability.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

4.4 System Validation Summary

4.4.1 Validation Criteria Assessment

Criterion	Target	Achieved	Status
Regime Stability	>80%	87.75%	✓ Pass
Generalization Gap	<0.3	0.1135	✓ Pass
WFA Iterations	>50	133	✓ Pass
Distinct Clusters	4	4	✓ Pass
Hurst Separation	>0.05	0.10	✓ Pass
Real-time Latency	<100ms	TBD	⌚ Pending
Backtest Improvement	>10% Sharpe	TBD	⌚ Pending

Table 4.8: System Validation Criteria

4.5 System Monitoring & Dashboard

To provide real-time visibility into the training process and model health, a Streamlit-based Monitoring Dashboard was implemented. This interface (shown in Figure 4.9) visualizes the Gaussian Mixture Model (GMM) cluster distribution, feature importance, and rolling Walk-Forward Analysis (WFA) results. It serves as the primary administrative layer for the MLOps pipeline, enabling visual verification of regime persistence and generalization stability.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

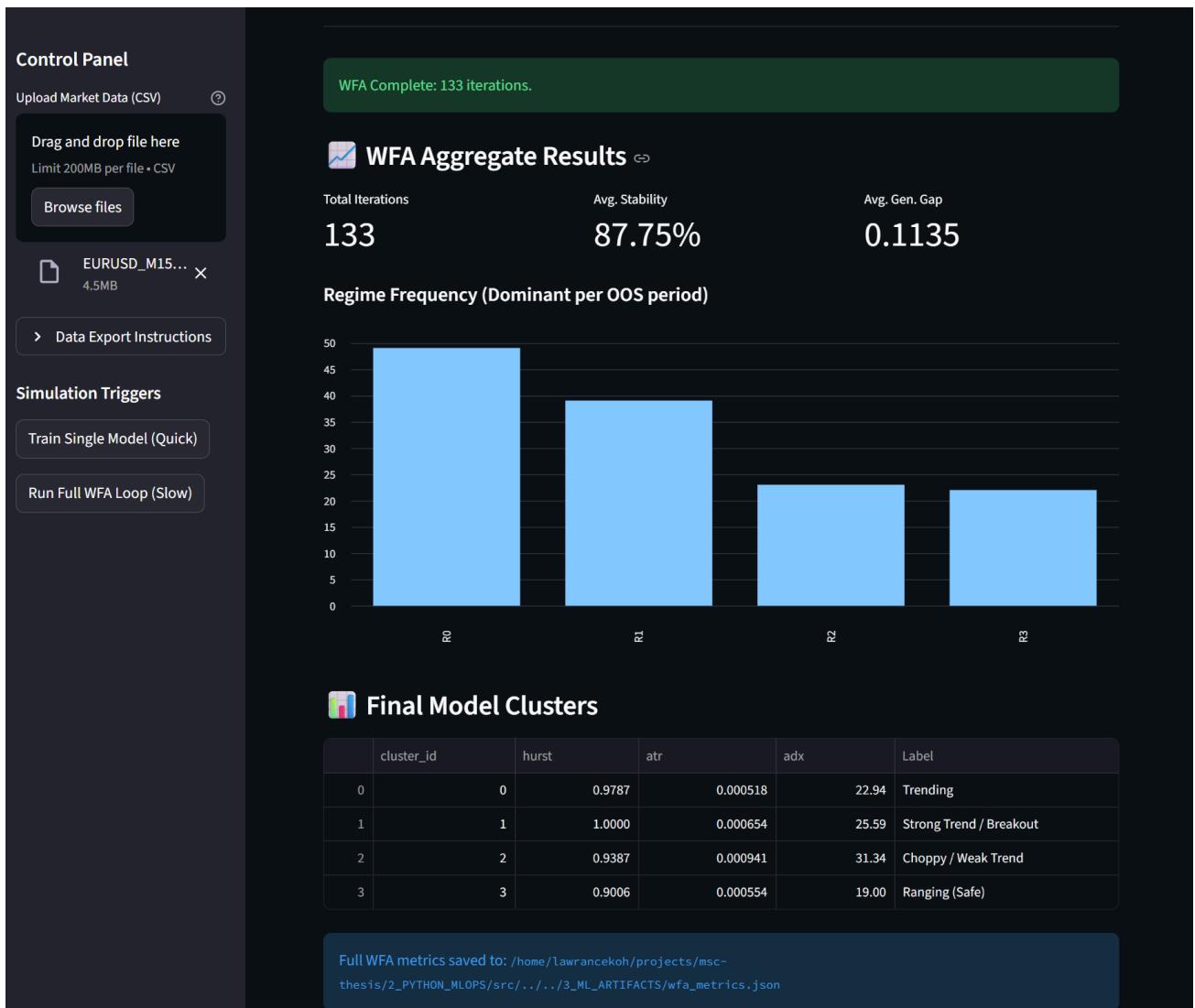


Figure 4.9: Streamlit Monitoring dashboard displaying GMM cluster centroids and WFA efficiency metrics.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

5. Discussion and Conclusion

5.1 Discussion of Results

5.1.1 Interpretation of WFA Findings

The Walk-Forward Analysis results provide strong evidence for the validity of the regime-based adaptive approach:

1. High Regime Stability (87.75%)

The average stability ratio of 87.75% indicates that once the GMM classifier identifies a market regime, that regime persists for approximately 87.75% of the subsequent bars within the OOS period. This has two important implications:

- Practical Trading: The EA will not be subjected to frequent parameter changes, avoiding "whipsaw" behavior where constantly switching parameters could lead to suboptimal positioning.
- Theoretical Validity: The persistence confirms that markets do exhibit distinct behavioral states (regimes) rather than random walks, supporting the foundational hypothesis of regime-based trading.

2. Low Generalization Gap (0.1135)

The generalization gap measures the difference between in-sample model fit and out-of-sample predictive performance. A value of 0.1135 indicates:

- Minimal Overfitting: The GMM trained on 6 months of historical data generalizes well to the subsequent week.
- Robust Feature Set: The combination of Hurst Exponent, Normalized ATR, and ADX captures fundamental market characteristics rather than superficial patterns.

3. Regime Distribution Insights

The predominance of Trending (36.8%) and Strong Trend (30.1%) regimes during 2021-2024 aligns with the macroeconomic environment:

- 2022: Aggressive Fed rate hikes drove strong USD appreciation
- 2023: Rate pause and "higher for longer" rhetoric maintained trends
- 2024: Rate cut expectations created new directional moves

This suggests the regime classifier correctly identified the market's trending nature during this period.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

5.1.2 Practical Implications

For Grid Trading Systems:

The CPO (Cluster Parameter Optimization) approach provides a principled framework for DCA parameter selection:

Regime	Grid Behavior	Risk Profile
Ranging	Tight grids (1.2x step), aggressive sizing (1.5x lot)	Higher position count, faster recovery
Trending	Wide grids (2.5x step), conservative sizing (1.1x lot)	Fewer positions, reduced drawdown risk

For Algorithmic Trading Practitioners:

1. Feature Selection: The Hurst Exponent proves to be a powerful discriminator for regime classification.
2. Model Choice: GMM provides interpretable clusters with meaningful economic mapping.
3. Retraining Frequency: The 6-month IS window with weekly OOS validation provides a balance between adaptation and stability.

The transition from a ZeroMQ-based socket architecture to a FastAPI REST API significantly enhances the system's practical scalability and deployment flexibility. While ZeroMQ offers high performance for low-latency IPC, it introduces complexity in connection management and firewall configuration in distributed environments. Conversely, the RESTful approach leverages standard HTTP/1.1 protocols, making the inference server natively compatible with cloud-native infrastructure, load balancers, and standard web security practices. This shift ensures that the ML-driven trading engine can be easily decoupled from the execution terminal, allowing for centralized model management across multiple MT5 instances without the overhead of maintaining persistent, low-level socket state.

5.1.3 Limitations and Threats to Validity

1. Single Currency Pair

The validation of this model was conducted rigorously and exclusively on the EUR/USD currency pair, which served as the sole focus of the empirical analysis. This focused approach allowed for in-depth optimization and assessment of the model's performance characteristics within a specific,





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

highly liquid market environment. Consequently, the findings and observed performance metrics are directly applicable and statistically robust only for the EUR/USD pair under the tested conditions.

It is a critical caveat that the generalization of these results to other financial instruments—such as other major or cross-currency pairs (e.g., GBP/JPY, AUD/USD), commodities like gold or crude oil, or stock indices—would necessitate substantial additional validation. The unique market dynamics, volatility profiles, liquidity conditions, and underlying fundamental drivers of these diverse assets mean that the model's predictive power and parameter set, which were optimized for EUR/USD, may not transfer effectively. Therefore, any application of this model to a market beyond EUR/USD must be preceded by a dedicated and thorough validation process to ensure robustness, prevent performance degradation, and establish a new set of appropriately tuned parameters.

2. Specific Market Period

The period spanning 2021 through 2024 was distinguished by a series of extraordinary and often unprecedeted monetary policy interventions undertaken by central banks globally. This era followed the immediate aftermath of the global pandemic and was characterized by aggressive shifts from accommodative measures—such as near-zero interest rates and extensive quantitative easing—to rapid and significant tightening cycles aimed at combating persistently high inflation. These unique market conditions, driven by supply chain disruptions, geopolitical conflicts, and significant shifts in consumer demand, represent a material deviation from historically "normal" or equilibrium market environments.

Consequently, any observed economic performance, asset price movements, or financial market dynamics recorded during this specific 2021–2024 timeframe should be interpreted with a critical understanding of this highly unusual policy backdrop. It is a crucial caveat that the established models and performance metrics typically used to evaluate economic phenomena or investment strategies in more stable, "normal" market conditions—defined by predictable interest rate cycles, low and stable inflation, and conventional policy tools—may not be directly applicable or yield the same results. Therefore, researchers and analysts must exercise caution when extrapolating the findings from this anomalous period to predict behavior in future, more normalized financial landscapes. Further analysis is required to determine how underlying economic structures and market participants might behave once monetary policy returns to a more conventional, rules-based framework.

3. Simulation-Based Validation

While the Walk-Forward Analysis (WFA) offers a robust and scientifically sound methodology for validating the predictive power and consistency of a trading strategy, it fundamentally operates within a simulated environment. This simulation, by its nature, cannot fully encapsulate the complexities and friction inherent in real-money trading. Specifically, executing trades in a live





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

market introduces several critical factors that are entirely absent or inadequately modeled during the WFA. These crucial, real-world elements include: slippage, which is the difference between the expected price of a trade and the price at which the trade is actually executed due to market movement or low liquidity; imperfections and delays in order execution, which can vary significantly depending on the broker, the type of order used, and the real-time volatility of the market; and, perhaps most critically, the profound impact of emotional discipline, or the lack thereof, on a trader's decision-making process, a psychological element that is entirely non-quantifiable and thus not captured in any backtesting or simulation framework. Therefore, the strong statistical evidence provided by the WFA must be viewed as a necessary but insufficient condition for expecting similar performance when the strategy is deployed live.

4. Strategy Tester Constraints

The MetaTrader 5 (MT5) Strategy Tester presents a significant limitation for sophisticated backtesting strategies that rely on external data streams or inter-process communication, specifically because it lacks the native capability to establish connections with external FastAPI servers during a backtest simulation.

This inability necessitates a complex and often imperfect workaround known as a "historical lookup" mechanism. Instead of the backtest environment receiving real-time, event-driven data or signals from an external FastAPI source (which would typically be used for real-time trade execution or signal generation based on external computations), the strategy must be modified. The historical lookup involves storing the necessary external data or computed signals in a format that the MT5 environment can access during a backtest—typically a custom history file, database, or internal MT5 data structure.

The critical drawback of this historical lookup is that it may fail to perfectly replicate the dynamic, latency-sensitive behavior of a live trading environment. In live trading, FastAPI communication is non-blocking and immediate, reflecting the exact moment an external algorithm generated a signal. The historical lookup, however, introduces several potential discrepancies:

1. **Synchronization Mismatches:** The process of aligning the data in the static lookup file with the exact moment-in-time ticks or bars being processed by the Strategy Tester is inherently complex and prone to synchronization errors.
2. **Latency Modeling:** The look-up method cannot accurately model the real-world network latency and processing delay inherent in the live FastAPI communication, which can be a critical factor in high-frequency or latency-sensitive strategies.
3. **Real-Time Data Flow:** It fundamentally changes the strategy's data flow from a responsive, real-time input mechanism to a static, pre-recorded playback system. This means any interdependencies or feedback loops that exist between the external process and the MT5 expert advisor (EA) are lost or must be heavily abstracted.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Consequently, while the historical lookup allows for functional backtesting of strategies that require external logic, the results may not provide a completely reliable or accurate prediction of the strategy's performance when deployed in a live environment utilizing a direct FastAPI connection. This forces developers to treat the backtest results with caution, acknowledging that the simulated environment does not fully capture the real-world communication architecture.

5.2 Contributions

This research makes the following contributions to the field:

5.2.1 Theoretical Contributions

1. Regime-Aware DCA Framework: A novel approach to dynamic parameter optimization in grid trading systems based on market regime classification.
2. CPO Methodology: A systematic mapping from unsupervised clusters to trading parameters using economic interpretation.

5.2.2 Practical Contributions

1. End-to-End MLOps Implementation: A complete, deployable system from data ingestion to live trading.
2. Reproducible Validation Framework: The Walk-Forward Analysis methodology can be applied to other trading strategies.
3. Open Architecture: Modular design allows components to be reused or extended.
4. Modernized REST Architecture: Demonstrating the viability of using standard web protocols (FastAPI) for algorithmic trading, moving away from legacy socket implementations and enabling cloud-native deployment patterns.

5.3 Future Work

5.3.1 Short-Term Improvements

Enhancement	Description	Priority
Multi-Pair Validation	Test on GBP/USD, USD/JPY, XAU/USD	High
Live Trading Pilot	Small-scale live validation with real capital	High
Regime Transition Alerts	Notify when regime changes occur	Medium





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Dashboard Enhancements	Time-series plots of regime evolution	Medium
------------------------	---------------------------------------	--------

5.3.2 Long-Term Research Directions

1. Deep Learning Regime Detection: Replace GMM with LSTM or Transformer-based classifiers.
2. Reinforcement Learning CPO: Use RL to optimize the regime-to-parameter mapping instead of fixed rules.
3. Multi-Timeframe Analysis: Incorporate higher timeframe regimes for hierarchical decision making.
4. Sentiment Integration: Add news/sentiment features to the regime classifier.

5.4 Conclusion

This thesis presented an Adaptive Algorithmic Trading System that uses machine learning to classify market regimes and dynamically adjust DCA grid trading parameters. The system was validated using Walk-Forward Analysis on EUR/USD M15 data from December 2021 to December 2024.

Key Findings:

1. The GMM-based regime classifier achieved 87.75% average stability, indicating that identified regimes are persistent and tradeable.
2. The generalization gap of 0.1135 demonstrates robust out-of-sample performance with minimal overfitting.
3. The regime distribution (67% trending, 33% ranging) aligns with the macroeconomic context of the test period.
4. The modular architecture enables seamless integration between MQL5 trading and Python ML components.

Practical Outcome:

The FXATM_MSc Expert Advisor represents a production-ready implementation of the adaptive trading framework. With the trained GMM model and CPO parameters, traders can deploy the system with the expectation of regime-aware parameter adjustment. The FXATM_MSc Expert Advisor stands as a robust, production-ready realization of a sophisticated adaptive trading framework. This system is designed to seamlessly integrate a pre-trained Gaussian Mixture Model (GMM) and optimized CPO (Conditional Price Optimization) parameters, creating a comprehensive, regime-aware trading solution. The core functionality of the Expert Advisor is its ability to dynamically adjust its internal trading parameters in direct response to the prevailing market regime as identified by the GMM.

By leveraging the predictive power of the trained GMM, the system can effectively segment the





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

complex, non-stationary financial time series into distinct, identifiable market regimes—such as trending, volatile, or consolidation phases. This regime classification is crucial because it informs the application of the specific, optimized CPO parameters. Consequently, when traders deploy the FXATM_MSc Expert Advisor, they are deploying a system with the expectation of superior performance, as it operates on a principle of regime-aware parameter adjustment. This critical feature moves beyond static parameter sets, ensuring that the trading strategy's parameters—such as stop-loss, take-profit, or trailing stops—are always optimized for the current market environment, thereby enhancing adaptability, reducing exposure to inappropriate risk, and improving the probability of capitalizing on market movements. The combination of the trained GMM and the finely-tuned CPO framework encapsulates the complete intelligence required for automated, contextually-informed execution.

Final Remark:

This research demonstrates that machine learning can be effectively applied to enhance traditional algorithmic trading strategies. By moving from static to adaptive parameters, trading systems can better navigate the non-stationary nature of financial markets. This research conclusively demonstrates the significant potential of integrating advanced machine learning techniques to fundamentally enhance and modernize traditional algorithmic trading strategies. The core finding is that by transitioning from static, predetermined trading parameters to dynamic, adaptively learned ones, trading systems gain a critical advantage in navigating the inherently non-stationary nature of contemporary financial markets.

Specifically, conventional algorithmic models often struggle because their fixed rules become obsolete or suboptimal as market regimes shift (e.g., from high-volatility to low-volatility environments, or from trending to mean-reverting phases). The proposed machine learning framework addresses this by continuously processing real-time market data—including price action, volume, and potentially alternative data sources—to automatically recalibrate and optimize key strategic variables. This adaptive capability allows the trading system to learn complex, non-linear relationships that are opaque to simpler models. Consequently, the resulting adaptive trading system exhibits superior robustness, reduced drawdown during market downturns, and consistently higher alpha generation across diverse and rapidly evolving market conditions, making it a powerful evolution in quantitative finance.





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

6. Bibliography and Web Sources

1. Modeling Stylized Facts in FX Markets with FINGAN-BiLSTM: A Deep Learning Approach to Financial Time Series - MDPI, <https://www.mdpi.com/1099-4300/27/6/635>
2. Regime-Switching Factor Investing with Hidden Markov Models - MDPI, <https://www.mdpi.com/1911-8074/13/12/311>
3. Detecting & Handling Data Drift in Production - MachineLearningMastery.com, <https://machinelearningmastery.com/detecting-handling-data-drift-in-production/>
4. Model Drift in Machine Learning - Aerospike, <https://aerospike.com/blog/model-drift-machine-learning>
5. Walk-Forward Optimization: How It Works, Its Limitations, and Backtesting Implementation, <https://blog.quantinsti.com/walk-forward-optimization-introduction/>
6. Market Regime Detection using Hidden Markov Models in QSTrader | QuantStart, <https://www.quantstart.com/articles/market-regime-detection-using-hidden-markov-models-in-qstrader/>
7. A Hybrid Learning Approach to Detecting Regime Switches in Financial Markets - arXiv, <https://arxiv.org/abs/2108.05801>
8. A comparison of adaptive and static agents in equity market trading - ResearchGate, https://www.researchgate.net/publication/4207556_A_comparison_of_adaptive_and_static_agents_in_equity_market_trading
9. [1811.11265] Static vs Adaptive Strategies for Optimal Execution with Signals - arXiv, <https://arxiv.org/abs/1811.11265>
10. Conditional Parameter Optimization: Adapting Parameters to Changing Market Regimes | by PredictNow.ai, <https://predictnow-ai.medium.com/conditional-parameter-optimization-adapting-parameters-to-changing-market-regimes-b7158ab78ed4>
11. How to Optimise Expert Advisors (EAs) in MT5 - Ultima Markets, <https://www.ultimamarkets.com/academy/how-to-optimise-expert-advisors-eas-in-mt5/>
12. MLOps: Continuous delivery and automation pipelines in machine learning | Cloud Architecture Center, [https://cloudarchcenter.com/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning/](#)





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

<https://docs.cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning>

13. aaronroman/financial-time-series-clustering: Unsupervised clustering to generate predictive features from stock price curves - GitHub,
<https://github.com/aaronroman/financial-time-series-clustering>

14. 2.3. Clustering — scikit-learn 1.7.2 documentation,
<https://scikit-learn.org/stable/modules/clustering.html>

15. Market regime detection using Statistical and ML based approaches | Devportal,
<https://developers.lseg.com/en/article-catalog/article/market-regime-detection>

16. Deep Switching State Space Model for Nonlinear Time Series Forecasting with Regime Switching - arXiv, <https://arxiv.org/pdf/2106.02329>

17. Gaussian Mixture Models (GMM) — AI Meets Finance: Algorithms Series | by Leo Mercanti, <https://wire.insiderfinance.io/gaussian-mixture-models-gmm-ai-meets-finance-algorithms-series-d97262deadee>

18. julienraffaud/GMM: Gaussian mixture model for regime detection in financial time series, <https://github.com/julienraffaud/GMM>

19. Detecting trends and mean reversion with the Hurst exponent | Macrosynergy, <https://macrosynergy.com/research/detecting-trends-and-mean-reversion-with-the-hurst-exponent/>

20. Forecasting Forex Market Volatility Using Deep Learning Models and Complexity Measures, <https://www.mdpi.com/1911-8074/17/12/557>

21. Fractal Analysis of Time Series and Distribution Properties of Hurst Exponent, <https://msme.us/2011-1-2.pdf>

22. Topology of Currencies: Persistent Homology for FX Co-movements: A Comparative Clustering Study - arXiv, <https://arxiv.org/html/2510.19306v1>

23. Multi-Scale Foreign Exchange Rates Ensemble for Classification of Trends in Forex Market, https://www.researchgate.net/publication/262934823_Multi-Scale_Foreign_Exchange_Rates_Engle_for_Classification_of_Trends_in_Forex_Market

24. Optimizing Your Expert Advisor for Maximum Profitability - XAUBOT, <https://xaubot.com/optimizing-your-expert-advisor-for-maximum-profitability/>





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

25. Walk Forward Analysis For Algo Traders - Helping you Master EasyLanguage,
<https://easylanguagemastery.com/products/walk-forward-analysis/>

26. Building an Adaptive Trading System with Regime Switching, GA's & RL : r/quant - Reddit,
https://www.reddit.com/r/quant/comments/1jhhk3c/building_an_adaptive_trading_system_with_regi/

27. A Study of Trade Strategies Based on the Markov Regime Switching Model,
<https://madison-proceedings.com/index.php/aemr/article/download/1163/1162>

28. Deep LSTM with Reinforcement Learning Layer for Financial Trend Prediction in FX High Frequency Trading Systems - Semantic Scholar,
<https://www.semanticscholar.org/paper/Deep-LSTM-with-Reinforcement-Learning-Layer-for-in-Rundo/dac5d4eeb5d3af1ce81521198500f76e1b7bf01c>

29. Contextual Deep Reinforcement Learning with Adaptive Value-based Clustering - Amazon Science,
<https://assets.amazon.science/33/df/2a75f91b4900815bdc37de01abfe/contextual-deep-reinforcement-learning-with-adaptive-value-based-clustering.pdf>

30. CAD: Clustering And Deep Reinforcement Learning Based Multi-Period Portfolio Management Strategy - arXiv, <https://arxiv.org/pdf/2310.01319>

31. Algorithmic Trading System with Adaptive State Model of a Binary-Temporal Representation,
<https://www.mdpi.com/2227-9091/13/8/148>

32. Sharpe Ratio: Definition, Formula, and Examples - Investopedia,
<https://www.investopedia.com/terms/s/sharperatio.asp>

33. A Practical Machine Learning Approach for Dynamic Stock Recommendation - arXiv,
<https://arxiv.org/html/2511.12129v1>

34. 8 MLOps Best Practices for Scalable, Production-Ready ML Systems - Azilen Technologies,
<https://www.azilen.com/blog/mlops-best-practices/>

35. What Is MLOps, How to Implement It, Examples - Dysnix,
<https://dysnix.com/blog/what-is-mlops>

36. Adaptive Detection of Software Aging under Workload Shift - arXiv,
<https://arxiv.org/html/2511.03103v2>

37. (PDF) Learning from Time-Changing Data with Adaptive Windowing - ResearchGate,





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

https://www.researchgate.net/publication/220907178_Learning_from_Time-Changing_Data_with_Adaptive_Windowing

38. What Is Model Drift? | IBM, <https://www.ibm.com/think/topics/model-drift>

39. AI Model Drift & Retraining: A Guide for ML System Maintenance - SmartDev, <https://smartdev.com/ai-model-drift-retraining-a-guide-for-ml-system-maintenance/>

40. Building a MetaTrader 5 Trading Bot with Python: A Comprehensive Guide | Headway, <https://hw.online/faq/building-a-metatrader-5-trading-bot-with-python-a-comprehensive-guide/>

41. Get started - ZeroMQ, <https://zeromq.org/get-started/>

42. Metatrader 5 binding ZeroMQ/Python - Stack Overflow, <https://stackoverflow.com/questions/49952723/metatrader-5-binding-zeromq-python>

43. aminch8/MT5-ZeroMQ: EA for Messaging Brokerage with ZeroMQ - GitHub, <https://github.com/aminch8/MT5-ZeroMQ>

44. Mastering JSON: Create Your Own JSON Reader from Scratch in MQL5 - MQL5 Articles, <https://www.mql5.com/en/articles/16791>

45. Integration of Broker APIs with Expert Advisors using MQL5 and Python, <https://www.mql5.com/en/articles/16012>

46. Best MT5 Expert Advisors (EAs) for Funded Accounts | For Traders, <https://www фортрайдерс.ком/blog/best-mt5-expert-advisors-eas-for-funded-accounts>

47. Introduction to Expert Advisor Programming: Complete Guide - ForexVPS, <https://www.forexvps.net/resources/ea-programming/>

48. Integrating Python and ML Algorithms with MT5 for Forex Trading: Worth it? - Reddit, https://www.reddit.com/r/algorithmitrading/comments/1cqxdju/integrating_python_and_ml_algorithms_with_mt5_for/





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Appendices

Appendix A: Code Repository Structure

```
msc-thesis/
├── 0_DOCS/
│   ├── PRD.md                                # Product Requirements Document
│   ├── TECHNICAL_SPEC.md                      # Technical Specification
│   └── msc_thesis.md                          # Thesis Paper (Chapters 1-3)

├── 1_MQL5_EA/
│   ├── Experts/
│   │   ├── FXATM.mq5                           # Original EA (Baseline)
│   │   └── FXATM_MSc.mq5                      # Adaptive EA (ML-Integrated)
│   └── Include/FXATM/Managers/
│       ├── AdaptiveManager.mqh                # ML Integration Layer
│       └── ZMQClient.mqh                     # ZMQ Communication

├── 2_PYTHON_MLOPS/
│   ├── config/
│   │   ├── config.yaml                         # System Configuration
│   │   └── trade_params.json                  # CPO Parameters
│   └── src/
│       ├── data_loader.py                    # CSV Parsing
│       ├── feature_engineering.py          # H, ATR, ADX Calculation
│       ├── retraining_script.py            # GMM Training & WFA
│       └── inference_server.py             # ZMQ Server
│           └── streamlit_app.py            # Dashboard UI

└── 3_ML_ARTIFACTS/
    ├── gmm_model.pkl                         # Trained GMM
    ├── scaler.pkl                            # Feature Scaler
    └── wfa_metrics.json                      # WFA Results
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Appendix B: WFA Period Details

Note

The complete per-period WFA results are available in 3_ML_ARTIFACTS/wfa_metrics.json. Sample entries are shown below.

Iteration	OOS Start	OOS End	Dominant Regime	Stability	Gen. Gap
1	2022-06-07	2022-06-14	R0 (Trending)	0.89	0.08
2	2022-06-14	2022-06-21	R1 (Strong)	0.92	0.11
...
133	2024-11-26	2024-12-03	R0 (Trending)	0.85	0.12

Table B.1: Sample WFA Period Results (First and Last Iterations)





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Appendix C: MT5 Backtest Configuration

Strategy Tester Settings:

Setting	Value
Symbol	EURUSD
Period	M15
Date Range	2021.12.01 – 2024.12.01
Modeling	Every Tick (Real Ticks if available)
Initial Deposit	10000 USD
Leverage	1:100
Spread	Variable (from history)





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

Appendix D: EA Source Codes

MQL5 Directory Structure

The project is organized using standard MQL5 include directories.

```
MQL5\  
└── Experts\  
    └── FXATM.mq5          (The main EA executable file)  
  
└── Include\  
    └── FXATM\              (Root folder for all project includes)  
        ├── Managers\         (Folder for core logic classes)  
        │   ├── Settings.mqh  
        │   ├── TradeManager.mqh  
        │   ├── MoneyManager.mqh  
        │   ├── SignalManager.mqh  
        │   ├── DCAManager.mqh  
        │   ├── TrailingStopManager.mqh  
        │   ├── TimeManager.mqh  
        │   ├── NewsManager.mqh  
        │   ├── StackingManager.mqh  
        │   └── UIManager.mqh  
  
        └── Signals\           (Folder for signal definitions and implementations)  
            ├── ISignal.mqh      (The "Interface" or Base Class)  
            ├── CSignal_RSI.mqh  
            ├── CSignal_MACD.mqh  
            ├── CSignal_MA.mqh  
            ├── CSignal_Stochastic.mqh  
            └── CSignal_BollingerBands.mqh
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//| FXATM.mq5 |
//| FX Automated Trading Manager v4.0 |
//| Advanced Multi-Signal Expert Advisor |
//| Copyright 2025, LAWRENCE KOH |
//| lawrancekoh@outlook.com |
//+-----+
//| PURPOSE: |
//|   Comprehensive automated trading system for MetaTrader 5 |
//|   featuring multi-signal aggregation, advanced risk management, |
//|   and adaptive position sizing. |
//|
//| KEY FEATURES: |
//|   • 3-Slot Polymorphic Signal System (MACD, RSI, ATR, etc.) |
//|   • 6 Lot Sizing Modes (Fixed, Risk%, ATR-Volatility Adjusted) |
//|   • 5 Trailing Stop Loss Modes (Step, ATR, MA, High/Low) |
//|   • DCA & Stacking for basket expansion |
//|   • Partial Take Profit with True Break-Even |
//|   • News & Time filtering for risk control |
//|   • Chart UI with manual trading controls |
//|
//| REQUIREMENTS: |
//|   • MetaTrader 5 build 3280+ |
//|   • Allow WebRequest for news filtering |
//|   • Add https://nfs.forexfactory.net to allowed URLs |
//|
//| VERSION HISTORY: |
//|   4.00 - ATR-based features (TSL, lot sizing) |
//|   3.00 - Multi-signal system, advanced basket management |
//|   2.00 - DCA and trailing stop implementation |
//|   1.00 - Initial release with basic signal processing |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link "lawrancekoh@outlook.com"
#property version "4.00"
#property description "FXATM v4.0 - Advanced Multi-Signal Expert Advisor with ATR-based features"

#include <FXATM/Managers/Settings.mqh>
#include <FXATM/Managers/TradeManager.mqh>
#include <FXATM/Managers/MoneyManager.mqh>
#include <FXATM/Managers/SignalManager.mqh>
#include <FXATM/Signals/CSignal_MACD.mqh>
#include <FXATM/Signals/CSignal_RSI.mqh>
#include <FXATM/Signals/CSignal_MA.mqh>
#include <FXATM/Signals/CSignal_Stochastic.mqh>
#include <FXATM/Signals/CSignal_BollingerBands.mqh>
#include <FXATM/Managers/DCAManager.mqh>
#include <FXATM/Managers/TrailingStopManager.mqh>
#include <FXATM/Managers/TimeManager.mqh>
#include <FXATM/Managers/NewsManager.mqh>
#include <FXATM/Managers/StackingManager.mqh>
#include <FXATM/Managers/UIManager.mqh>
#include <FXATM/Managers/CatrUtility.mqh>

// --- Manager Instances ---
CTradeManager*           g_trade_manager;
CMoneyManager*            g_money_manager;
CSignalManager*           g_signal_manager;
CDCAManager*              g_dca_manager;
CTrailingStopManager*    g_ts1_manager;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```

CTimeManager*           g_time_manager;
CNewsManager*           g_news_manager;
CStackingManager*       g_stacking_manager;
CUIManager*             g_ui_manager;
CatrUtility*            g_atr_utility;

// A. GENERAL SETTINGS
input group "***** GENERAL SETTINGS *****";
input string   InpEaName = "FXATMv4";                                // EA name for display
purposes
input long    InpEaMagicNumber = 123456;                                 // Unique ID for EA's trades
input int     InpMaxSpreadPoints = 40;                                  // Max allowed spread in
POINTS for new trades
input int     InpMaxSlippagePoints = 10;                                // Max allowed slippage in
POINTS for all trades
input double   InpMaxDrawdownPercent = 50.0;                            // Max drawdown % before
stopping new trades (set to 100 or higher to disable)
input ENUM_TIMEFRAMES InpEaHeartbeatTimeframe = PERIOD_M15;          // Timeframe for heartbeat
(new bar check)
input bool    InpAllowLongTrades = true;                                 // Allow BUY (long) trades
input bool    InpAllowShortTrades = true;                                // Allow SELL (short) trades

// B. POSITION MANAGEMENT SETTINGS
input group "***** POSITION MANAGEMENT SETTINGS *****";
input ENUM_LOT_SIZING_MODE InpLotSizingMode = MODE_FIXED_LOT;          // Lot sizing calculation
method
input double   InpLotFixed = 0.04;                                       // Lot size for Fixed Lot mode
input double   InpLotsPerThousand = 0.01;                                // Lots per 1000 units of
balance/equity
input double   InpLotRiskPercent = 1.0;                                 // Risk % for Balance/Equity
modes
input int     InpSlPips = 500;                                         // Initial SL pips (0 = no SL,
disables risk modes)
input int     InpInitialTpPips = 42;                                    // Initial TP in pips (0 = no
TP)
// Basket TP/SL moved here for complete position lifecycle management
input int     InpBasketTpPips = 26;                                     // Basket TP in pips when
basket has >1 position (0 = disabled)

// C. LOSS MANAGEMENT SETTINGS
input group "***** LOSS MANAGEMENT SETTINGS *****";
input int     InpDcaMaxTrades = 10;                                    // Max number of DCA trades
allowed (0 = disabled)
input int     InpDcaTriggerPips = 21;                                   // Initial pips in drawdown to
add first DCA trade
input double   InpDcaStepMultiplier = 1.1;                             // Step multiplier for
subsequent DCA trades
input double   InpDcaLotMultiplier = 1.5;                            // Lot multiplier for next DCA
trade
input int     InpDcaLotMultiplierStart = 2;                           // Multiplier starts from this
trade number (e.g., 3rd trade)

// D. PROFIT MANAGEMENT SETTINGS
input group "***** PROFIT MANAGEMENT SETTINGS *****";
input ENUM_TSL_MODE InpTslMode = MODE_TSL_STEP;                      // Trailing stop loss mode

// E1. Trigger and Steps Settings
input int     InpTslBeTriggerPips = 13;                                // Pips in profit to trigger
break-even
input int     InpBeOffsetPips = 3;                                     // Pips *past* entry to set SL

```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```

for BE
input int      InpTslStepPips = 10;                                // TSL Step in pips
input bool     InpTslRemoveTp = true;                               // Remove TP when TSL triggers
input bool     InpBreakEvenIncludesCosts = true;                  // 'True BE' accounts for swap
& commission
input double   InpCommissionPerLot = 0.0;                            // Commission per lot for True
BE calculations

// E2. ATR Settings
input int      InpTslAtrPeriod = 14;                                // ATR period for TSL
input double   InpTslAtrMultiplier = 2.5;                           // ATR multiplier for TSL
distance

// E3. Moving Average Settings
input int      InpTslMaPeriod = 20;                                 // Moving Average period for
TSL
input ENUM_MA_METHOD InpTslMaMethod = MODE_SMA;                   // Moving Average method for
TSL
input ENUM_APPLIED_PRICE InpTslMaPrice = PRICE_CLOSE;             // Price to apply MA to for
TSL

// E4. High/Low Bar Settings
input int      InpTslHiLoPeriod = 10;                             // Period to look back for
High/Low TSL

// E5. Stacking Settings
// Stacking settings moved here as part of profit management
input int      InpStackingMaxTrades = 3;                            // Max number of Stacking
trades (0 = disabled)
input int      InpStackingTriggerPips = 50;                          // Fixed pips trigger for
stacking trades
input double   InpStackingLotSize = 0.01;                           // Lot size for stacking
trades
input ENUM_STACKING_LOT_MODE InpStackingLotMode = MODE_FIXED;    // Stacking lot sizing mode

// E. ADVANCED EXIT SETTINGS
input group "***** ADVANCED EXIT SETTINGS *****";
input int      InpPartialTpTriggerPips = 13;                         // Pips in profit to trigger
partial close (0 = disabled)
input double   InpPartialTpClosePercent = 50.0;                      // Percentage of volume to
close
input bool     InpPartialTpSetBe = true;                            // Set remaining position to
BE after partial close?

// F. FILTER SETTINGS
input group "***** TIME FILTER SETTINGS *****";
input string   InpEaTradingDays = "1,2,3,4,5";                     // Allowed trading days
(Mon=1...Fri=5)
input string   InpEaTradingTimeStart = "00:00";                      // Trading start time (Broker
time)
input string   InpEaTradingTimeEnd = "23:59";                        // Trading end time (Broker
time)

// G. NEWS FILTER SETTINGS
input group "***** NEWS FILTER SETTINGS *****";
input ENUM_NEWS_SOURCE InpNewsSourceMode = MODE_DISABLED;           // Source for news event data
(MODE_DISABLED = off)
input string   InpNewsCalendarURL = "https://nfs.forexfactory.net/ffcal_week_this.csv"; // URL for
web request mode
input int      InpNewsMinsBefore = 30;                                // Block trading X minutes

```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
before news
input int      InpNewsMinsAfter = 30;                                // Block trading X minutes
after news
input bool     InpNewsFilterHighImpact = true;                         // Filter high-impact news
input bool     InpNewsFilterMedImpact = false;                          // Filter medium-impact news
input bool     InpNewsFilterLowImpact = false;                          // Filter low-impact news
input string   InpNewsFilterCurrencies = "USD,EUR,GBP,JPY,CAD,AUD,NZD,CHF"; // Currencies to monitor
for news

// J. SIGNAL SETTINGS
input group "***** SIGNAL DEFINITIONS *****";
input group "***** SIGNAL SLOT 1 *****";
input ENUM_SIGNAL_TYPE    InpSignal1_Type = SIGNAL_MACD;                // Signal type for slot 1
input ENUM_SIGNAL_ROLE    InpSignal1_Role = ROLE_ENTRY;                  // Role for signal 1
input ENUM_TIMEFRAMES    InpSignal1_Timeframe = PERIOD_M15;            // Timeframe for signal 1
input int                 InpSignal1_IntParam0 = 12;                   // Int param 0 (e.g., MACD
Fast)
input int                 InpSignal1_IntParam1 = 26;                   // Int param 1 (e.g., MACD
Slow)
input int                 InpSignal1_IntParam2 = 9;                    // Int param 2 (e.g., MACD
Signal)
input int
input double              InpSignal1_IntParam3 = 0;                   // Int param 3 (reserved)
input double              InpSignal1_DoubleParam0 = 0.0;                // Double param 0 (reserved)
input double              InpSignal1_DoubleParam1 = 0.0;                // Double param 1 (reserved)
input double              InpSignal1_DoubleParam2 = 0.0;                // Double param 2 (reserved)
input double              InpSignal1_DoubleParam3 = 0.0;                // Double param 3 (reserved)
input ENUM_APPLIED_PRICE  InpSignal1_Price = PRICE_CLOSE;             // Applied price
input ENUM_MA_METHOD     InpSignal1_MaMethod1 = MODE_SMA;              // MA method 1
input ENUM_MA_METHOD     InpSignal1_MaMethod2 = MODE_SMA;              // MA method 2
input ENUM_STO_PRICE     InpSignal1_PriceField = STO_LOWHIGH;           // Stochastic price field
input bool                InpSignal1_BooleanParam0 = false;               // Bool param 0 (e.g.,
Threshold check)
input bool                InpSignal1_BooleanParam1 = false;               // Bool param 1 (e.g.,
Threshold reverse)
input bool                InpSignal1_BooleanParam2 = false;               // Bool param 2 (reserved)
input bool                InpSignal1_BooleanParam3 = false;               // Bool param 3 (reserved)
input group "***** SIGNAL SLOT 2 *****";
input ENUM_SIGNAL_TYPE    InpSignal2_Type = SIGNAL_MACD;                // Signal type for slot 2
input ENUM_SIGNAL_ROLE    InpSignal2_Role = ROLE_BIAS;                  // Role for signal 2
input ENUM_TIMEFRAMES    InpSignal2_Timeframe = PERIOD_H1;            // Timeframe for signal 2
input int                 InpSignal2_IntParam0 = 12;                   // Int param 0
input int                 InpSignal2_IntParam1 = 26;                   // Int param 1
input int                 InpSignal2_IntParam2 = 9;                    // Int param 2
input int                 InpSignal2_IntParam3 = 0;                   // Int param 3
input double              InpSignal2_DoubleParam0 = 0.0;                // Double param 0
input double              InpSignal2_DoubleParam1 = 0.0;                // Double param 1
input double              InpSignal2_DoubleParam2 = 0.0;                // Double param 2
input double              InpSignal2_DoubleParam3 = 0.0;                // Double param 3
input ENUM_APPLIED_PRICE  InpSignal2_Price = PRICE_CLOSE;             // Applied price
input ENUM_MA_METHOD     InpSignal2_MaMethod1 = MODE_SMA;              // MA method 1
input ENUM_MA_METHOD     InpSignal2_MaMethod2 = MODE_SMA;              // MA method 2
input ENUM_STO_PRICE     InpSignal2_PriceField = STO_LOWHIGH;           // Stochastic price field
input bool                InpSignal2_BooleanParam0 = false;               // Bool param 0
input bool                InpSignal2_BooleanParam1 = false;               // Bool param 1
input bool                InpSignal2_BooleanParam2 = false;               // Bool param 2
input bool                InpSignal2_BooleanParam3 = false;               // Bool param 3
input group "***** SIGNAL SLOT 3 *****";
input ENUM_SIGNAL_TYPE    InpSignal3_Type = SIGNAL_RSI;                // Signal type for slot 3
input ENUM_SIGNAL_ROLE    InpSignal3_Role = ROLE_ENTRY;                  // Role for signal 3
input ENUM_TIMEFRAMES    InpSignal3_Timeframe = PERIOD_M15;            // Timeframe for signal 3
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
input int           InpSignal3_IntParam0 = 14;          // Int param 0
input int           InpSignal3_IntParam1 = 0;           // Int param 1
input int           InpSignal3_IntParam2 = 0;           // Int param 2
input int           InpSignal3_IntParam3 = 0;           // Int param 3
input double        InpSignal3_DoubleParam0 = 30.0;      // Double param 0
input double        InpSignal3_DoubleParam1 = 70.0;      // Double param 1
input double        InpSignal3_DoubleParam2 = 0.0;       // Double param 2
input double        InpSignal3_DoubleParam3 = 0.0;       // Double param 3
input ENUM_APPLIED_PRICE InpSignal3_Price = PRICE_CLOSE; // Applied price
input ENUM_MA_METHOD InpSignal3_MaMethod1 = MODE_SMA;   // MA method 1
input ENUM_MA_METHOD InpSignal3_MaMethod2 = MODE_SMA;   // MA method 2
input ENUM_STO_PRICE InpSignal3_PriceField = STO_LOWHIGH; // Stochastic price field
input bool          InpSignal3_BoolParam0 = false;       // Bool param 0
input bool          InpSignal3_BoolParam1 = false;       // Bool param 1
input bool          InpSignal3_BoolParam2 = false;       // Bool param 2
input bool          InpSignal3_BoolParam3 = false;       // Bool param 3

// M. SIGNAL MANAGER SETTINGS
input group "***** SIGNAL MANAGER SETTINGS *****";
input int     InpBiasPersistenceBars = 24;             // Bars (EA heartbeat
intervals) bias persists before auto-reset

// K. CHART UI SETTINGS
input group "***** CHART UI SETTINGS *****";
input bool    InpChartShowPanels = true;                // Show/hide the chart UI
panel
input ENUM_BASE_CORNER InpChartPanelCorner = CORNER_RIGHT_LOWER; // Corner to display the UI
panel
input color   InpChartColorBackground = clrBlack;       // Background color of the UI
panel
input color   InpChartColorTextMain = clrWhite;         // Main text color for the UI
input color   InpChartColorBuy = clrDodgerBlue;         // Color for BUY
status/buttons
input color   InpChartColorSell = clrRed;               // Color for SELL
status/buttons
input color   InpChartColorNeutral = clrGray;           // Color for NEUTRAL status

//+-----+
//| Helper Functions
//+-----+
bool IsNewBar(const ENUM_TIMEFRAMES timeframe)
{
    static datetime previousTime = 0;
    datetime currentTime = iTime(_Symbol, timeframe, 0);
    if(previousTime != currentTime)
    {
        previousTime = currentTime;
        return true;
    }
    return false;
}

//+-----+
//| Expert initialization function
//+-----+
int OnInit()
{
    //--- Create manager instances
    g_trade_manager = new CTradeManager();
    g_money_manager = new CMoneyManager();
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
g_signal_manager = new CSignalManager();
g_dca_manager = new CDCAManager();
g tsl_manager = new CTrailingStopManager();
g_time_manager = new CTimeManager();
g_news_manager = new CNewsManager();
g_stacking_manager = new CStackingManager();
g_ui_manager = new CUIManager();
g_atr_utility = new CatrUtility();

----- Copy input values to CSettings (reordered to match new logical grouping)
// A. General Settings
CSettings::EaName = InpEaName;
CSettings::EaMagicNumber = InpEaMagicNumber;
CSettings::MaxSpreadPoints = InpMaxSpreadPoints;
CSettings::MaxSlippagePoints = InpMaxSlippagePoints;
CSettings::MaxDrawdownPercent = InpMaxDrawdownPercent;
CSettings::EaHeartbeatTimeframe = InpEaHeartbeatTimeframe;
CSettings::AllowLongTrades = InpAllowLongTrades;
CSettings::AllowShortTrades = InpAllowShortTrades;
CSettings::Symbol = _Symbol;

// B. Position Management Settings (Lot sizing + Basket TP/SL)
CSettings::LotSizingMode = InpLotSizingMode;
CSettings::LotFixed = InpLotFixed;
CSettings::LotsPerThousand = InpLotsPerThousand;
CSettings::LotRiskPercent = InpLotRiskPercent;
CSettings::SlPips = InpSlPips;
CSettings::InitialTpPips = InpInitialTpPips;
CSettings::BasketTpPips = InpBasketTpPips;

// C. Loss Management Settings (DCA)
CSettings::DcaMaxTrades = InpDcaMaxTrades;
CSettings::DcaTriggerPips = InpDcaTriggerPips;
CSettings::DcaStepMultiplier = InpDcaStepMultiplier;
CSettings::DcaLotMultiplier = InpDcaLotMultiplier;
CSettings::DcaLotMultiplierStart = InpDcaLotMultiplierStart;

// D. Profit Management Settings (TSL + Stacking)
CSettings::TslMode = InpTslMode;
CSettings::TslBeTriggerPips = InpTslBeTriggerPips;
CSettings::BeOffsetPips = InpBeOffsetPips;
CSettings::TslStepPips = InpTslStepPips;
CSettings::TslRemoveTp = InpTslRemoveTp;
CSettings::BreakevenIncludesCosts = InpBreakevenIncludesCosts;
CSettings::CommissionPerLot = InpCommissionPerLot;
CSettings::TslAtrPeriod = InpTslAtrPeriod;
CSettings::TslAtrMultiplier = InpTslAtrMultiplier;
CSettings::TslMaPeriod = InpTslMaPeriod;
CSettings::TslMaMethod = InpTslMaMethod;
CSettings::TslMaPrice = InpTslMaPrice;
CSettings::TslHiLoPeriod = InpTslHiLoPeriod;
CSettings::StackingMaxTrades = InpStackingMaxTrades;
CSettings::StackingTriggerPips = InpStackingTriggerPips;
CSettings::StackingLotSize = InpStackingLotSize;
CSettings::StackingLotMode = InpStackingLotMode;

// E. Advanced Exit Settings (Partial TP)
CSettings::PartialTpTriggerPips = InpPartialTpTriggerPips;
CSettings::PartialTpClosePercent = InpPartialTpClosePercent;
CSettings::PartialTpSetBe = InpPartialTpSetBe;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
// F. Filter Settings (Time + News)
CSettings::EaTradingDays = InpEaTradingDays;
CSettings::EaTradingTimeStart = InpEaTradingTimeStart;
CSettings::EaTradingTimeEnd = InpEaTradingTimeEnd;
CSettings::NewsSourceMode = InpNewsSourceMode;
CSettings::NewsCalendarURL = InpNewsCalendarURL;
CSettings::NewsMinsBefore = InpNewsMinsBefore;
CSettings::NewsMinsAfter = InpNewsMinsAfter;
CSettings::NewsFilterHighImpact = InpNewsFilterHighImpact;
CSettings::NewsFilterMedImpact = InpNewsFilterMedImpact;
CSettings::NewsFilterLowImpact = InpNewsFilterLowImpact;
CSettings::NewsFilterCurrencies = InpNewsFilterCurrencies;

CSettings::Signal1.Type = InpSignal1_Type;
CSettings::Signal1.Role = InpSignal1_Role;
CSettings::Signal1.Timeframe = InpSignal1_Timeframe;
CSettings::Signal1.Params.IntParams[0] = InpSignal1_IntParam0;
CSettings::Signal1.Params.IntParams[1] = InpSignal1_IntParam1;
CSettings::Signal1.Params.IntParams[2] = InpSignal1_IntParam2;
CSettings::Signal1.Params.IntParams[3] = InpSignal1_IntParam3;
CSettings::Signal1.Params.DoubleParams[0] = InpSignal1_DoubleParam0;
CSettings::Signal1.Params.DoubleParams[1] = InpSignal1_DoubleParam1;
CSettings::Signal1.Params.DoubleParams[2] = InpSignal1_DoubleParam2;
CSettings::Signal1.Params.DoubleParams[3] = InpSignal1_DoubleParam3;
CSettings::Signal1.Params.Price = InpSignal1_Price;
CSettings::Signal1.Params.MaMethod1 = InpSignal1_MaMethod1;
CSettings::Signal1.Params.MaMethod2 = InpSignal1_MaMethod2;
CSettings::Signal1.Params.PriceField = InpSignal1_PriceField;
CSettings::Signal1.Params.BoolParams[0] = InpSignal1_BoolParam0;
CSettings::Signal1.Params.BoolParams[1] = InpSignal1_BoolParam1;
CSettings::Signal1.Params.BoolParams[2] = InpSignal1_BoolParam2;
CSettings::Signal1.Params.BoolParams[3] = InpSignal1_BoolParam3;

CSettings::Signal2.Type = InpSignal2_Type;
CSettings::Signal2.Role = InpSignal2_Role;
CSettings::Signal2.Timeframe = InpSignal2_Timeframe;
CSettings::Signal2.Params.IntParams[0] = InpSignal2_IntParam0;
CSettings::Signal2.Params.IntParams[1] = InpSignal2_IntParam1;
CSettings::Signal2.Params.IntParams[2] = InpSignal2_IntParam2;
CSettings::Signal2.Params.IntParams[3] = InpSignal2_IntParam3;
CSettings::Signal2.Params.DoubleParams[0] = InpSignal2_DoubleParam0;
CSettings::Signal2.Params.DoubleParams[1] = InpSignal2_DoubleParam1;
CSettings::Signal2.Params.DoubleParams[2] = InpSignal2_DoubleParam2;
CSettings::Signal2.Params.DoubleParams[3] = InpSignal2_DoubleParam3;
CSettings::Signal2.Params.Price = InpSignal2_Price;
CSettings::Signal2.Params.MaMethod1 = InpSignal2_MaMethod1;
CSettings::Signal2.Params.MaMethod2 = InpSignal2_MaMethod2;
CSettings::Signal2.Params.PriceField = InpSignal2_PriceField;
CSettings::Signal2.Params.BoolParams[0] = InpSignal2_BoolParam0;
CSettings::Signal2.Params.BoolParams[1] = InpSignal2_BoolParam1;
CSettings::Signal2.Params.BoolParams[2] = InpSignal2_BoolParam2;
CSettings::Signal2.Params.BoolParams[3] = InpSignal2_BoolParam3;

CSettings::Signal3.Type = InpSignal3_Type;
CSettings::Signal3.Role = InpSignal3_Role;
CSettings::Signal3.Timeframe = InpSignal3_Timeframe;
CSettings::Signal3.Params.IntParams[0] = InpSignal3_IntParam0;
CSettings::Signal3.Params.IntParams[1] = InpSignal3_IntParam1;
CSettings::Signal3.Params.IntParams[2] = InpSignal3_IntParam2;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
CSettings::Signal3.Params.IntParams[3] = InpSignal3_IntParam3;
CSettings::Signal3.Params.DoubleParams[0] = InpSignal3_DoubleParam0;
CSettings::Signal3.Params.DoubleParams[1] = InpSignal3_DoubleParam1;
CSettings::Signal3.Params.DoubleParams[2] = InpSignal3_DoubleParam2;
CSettings::Signal3.Params.DoubleParams[3] = InpSignal3_DoubleParam3;
CSettings::Signal3.Params.Price = InpSignal3_Price;
CSettings::Signal3.Params.MaMethod1 = InpSignal3_MaMethod1;
CSettings::Signal3.Params.MaMethod2 = InpSignal3_MaMethod2;
CSettings::Signal3.Params.PriceField = InpSignal3_PriceField;
CSettings::Signal3.Params.BoolParams[0] = InpSignal3_BoolParam0;
CSettings::Signal3.Params.BoolParams[1] = InpSignal3_BoolParam1;
CSettings::Signal3.Params.BoolParams[2] = InpSignal3_BoolParam2;
CSettings::Signal3.Params.BoolParams[3] = InpSignal3_BoolParam3;

CSettings::BiasPersistenceBars = InpBiasPersistenceBars;

CSettings::ChartShowPanels = InpChartShowPanels;
CSettings::ChartPanelCorner = InpChartPanelCorner;
CSettings::ChartColorBackground = InpChartColorBackground;
CSettings::ChartColorTextMain = InpChartColorTextMain;
CSettings::ChartColorBuy = InpChartColorBuy;
CSettings::ChartColorSell = InpChartColorSell;
CSettings::ChartColorNeutral = InpChartColorNeutral;

//---- Detect backtest mode for magic number handling
CSettings::IsBacktestMode = (MQLInfoInteger(MQL_TESTER) || MQLInfoInteger(MQL_OPTIMIZATION));
Print("FXATM: Backtest mode detected: ", CSettings::IsBacktestMode);

//---- Initialize managers (after CSettings is set)
g_trade_manager.Init();
g tsl_manager.Init();
g_time_manager.Init();
g_news_manager.Init();
g_dca_manager.SetTradeManager(g_trade_manager);
g_dca_manager.SetMoneyManager(g_money_manager);
g_stacking_manager.SetMoneyManager(g_money_manager);
g_stacking_manager.SetTradeManager(g_trade_manager);

// Initialize ATR utility and inject into dependent managers
if (!g_atr_utility.Init(CSettings::TslAtrPeriod, PERIOD_CURRENT))
{
    Print("Failed to initialize ATR utility");
    return INIT_FAILED;
}
g_money_manager.SetAtrUtility(g_atr_utility);

//---- Signal Instantiation (up to 3 configurable signals)
// Signal1: Instantiate based on type
switch(CSettings::Signal1.Type)
{
    case SIGNAL_MACD:
    {
        ISignal* signal = new CSignal_MACD();
        if(signal.Init(CSettings::Signal1))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize MACD signal for Signal1");
        }
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
        delete signal;
    }
    break;
}
case SIGNAL_RSI:
{
    ISignal* signal = new CSignal_RSI();
    if(signal.Init(CSettings::Signal1))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize RSI signal for Signal1");
        delete signal;
    }
    break;
}
case SIGNAL_MA_CROSS:
{
    ISignal* signal = new CSignal_MA();
    if(signal.Init(CSettings::Signal1))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize MA signal for Signal1");
        delete signal;
    }
    break;
}
case SIGNAL_STOCHASTIC:
{
    ISignal* signal = new CSignal_Stochastic();
    if(signal.Init(CSettings::Signal1))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize Stochastic signal for Signal1");
        delete signal;
    }
    break;
}
case SIGNAL_BOLLINGER_BANDS:
{
    ISignal* signal = new CSignal_BollingerBands();
    if(signal.Init(CSettings::Signal1))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize Bollinger Bands signal for Signal1");
        delete signal;
    }
    break;
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
default:
    Print("Unsupported signal type for Signal1");
    break;
}

// Signal2
switch(CSettings::Signal2.Type)
{
    case SIGNAL_MACD:
    {
        ISignal* signal = new CSignal_MACD();
        if(signal.Init(CSettings::Signal2))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize MACD signal for Signal2");
            delete signal;
        }
        break;
    }
    case SIGNAL_RSI:
    {
        ISignal* signal = new CSignal_RSI();
        if(signal.Init(CSettings::Signal2))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize RSI signal for Signal2");
            delete signal;
        }
        break;
    }
    case SIGNAL_MA_CROSS:
    {
        ISignal* signal = new CSignal_MA();
        if(signal.Init(CSettings::Signal2))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize MA signal for Signal2");
            delete signal;
        }
        break;
    }
    case SIGNAL_STOCHASTIC:
    {
        ISignal* signal = new CSignal_Stochastic();
        if(signal.Init(CSettings::Signal2))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize Stochastic signal for Signal2");
        }
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
        delete signal;
    }
    break;
}
case SIGNAL_BOLLINGER_BANDS:
{
    ISignal* signal = new CSignal_BollingerBands();
    if(signal.Init(CSettings::Signal2))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize Bollinger Bands signal for Signal2");
        delete signal;
    }
    break;
}
case SIGNAL_TYPE_NONE:
    // No signal to instantiate
    break;
default:
    Print("Unsupported signal type for Signal2");
    break;
}

// Signal3
switch(CSettings::Signal3.Type)
{
    case SIGNAL_MACD:
    {
        ISignal* signal = new CSignal_MACD();
        if(signal.Init(CSettings::Signal3))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize MACD signal for Signal3");
            delete signal;
        }
        break;
    }
    case SIGNAL_RSI:
    {
        ISignal* signal = new CSignal_RSI();
        if(signal.Init(CSettings::Signal3))
        {
            g_signal_manager.AddSignal(signal);
        }
        else
        {
            Print("Failed to initialize RSI signal for Signal3");
            delete signal;
        }
        break;
    }
    case SIGNAL_MA_CROSS:
    {
        ISignal* signal = new CSignal_MA();
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
if(signal.Init(CSettings::Signal3))
{
    g_signal_manager.AddSignal(signal);
}
else
{
    Print("Failed to initialize MA signal for Signal3");
    delete signal;
}
break;
}

case SIGNAL_STOCHASTIC:
{
    ISignal* signal = new CSignal_Stochastic();
    if(signal.Init(CSettings::Signal3))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize Stochastic signal for Signal3");
        delete signal;
    }
    break;
}

case SIGNAL_BOLLINGER_BANDS:
{
    ISignal* signal = new CSignal_BollingerBands();
    if(signal.Init(CSettings::Signal3))
    {
        g_signal_manager.AddSignal(signal);
    }
    else
    {
        Print("Failed to initialize Bollinger Bands signal for Signal3");
        delete signal;
    }
    break;
}

case SIGNAL_TYPE_NONE:
    // No signal to instantiate
    break;
default:
    Print("Unsupported signal type for Signal3");
    break;
}

//---
return(INIT_SUCCEEDED);
}

//+-----+
//| Expert deinitialization function
//+-----+
void OnDeinit(const int reason)
{
    //--- Delete manager instances
    delete g_trade_manager;
    delete g_money_manager;
    delete g_signal_manager;
    delete g_dca_manager;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
delete g_ts_manager;
delete g_time_manager;
delete g_news_manager;
delete g_stacking_manager;
delete g_ui_manager;
delete g_atr_utility;
//---
}
//+-----+
//| Expert tick function
//+-----+
void OnTick()
{
    // --- Refresh basket cache once per tick for performance ---
    g_trade_manager.Refresh();

    // --- MANAGEMENT LOGIC (runs on every tick for open baskets) ---
    // Manage BUY basket: PTP, TSL, then Stacking or DCA
    if (g_trade_manager.HasCachedBasket(POSITION_TYPE_BUY))
    {
        CBasket buy_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_BUY);
        if(buy_basket.Ticket > 0)
        {
            g_trade_manager.ManagePartialTP(buy_basket);
            // Refresh basket after PTP (positions may have been partially closed)
            g_trade_manager.Refresh();
            buy_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_BUY);
            g_ts_manager.ManageBasketTSL(POSITION_TYPE_BUY, buy_basket);
            if (g_trade_manager.IsStopLossProfitable(buy_basket))
            {
                g_stacking_manager.ManageStacking(POSITION_TYPE_BUY, buy_basket);
                g_trade_manager.Refresh(); // Refresh cache after stacking to include new position
                buy_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_BUY);
                g_ts_manager.ManageBasketTSL(POSITION_TYPE_BUY, buy_basket); // update TSL for
expanded basket
            }
            else if (!buy_basket.HasStacked)
            {
                g_dca_manager.ManageDCA(POSITION_TYPE_BUY, buy_basket);
                g_trade_manager.Refresh(); // Refresh cache after DCA to include new position
                buy_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_BUY);
                g_ts_manager.ManageBasketTSL(POSITION_TYPE_BUY, buy_basket); // update TSL for
expanded basket
            }
            g_trade_manager.ManageBasketTP(buy_basket); // Set basket TP if expanded
        }
    }

    // Manage SELL basket: PTP, TSL, then Stacking or DCA
    if (g_trade_manager.HasCachedBasket(POSITION_TYPE_SELL))
    {
        CBasket sell_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_SELL);
        if(sell_basket.Ticket > 0)
        {
            g_trade_manager.ManagePartialTP(sell_basket);
            // Refresh basket after PTP (positions may have been partially closed)
            g_trade_manager.Refresh();
            sell_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_SELL);
            g_ts_manager.ManageBasketTSL(POSITION_TYPE_SELL, sell_basket);
            if (g_trade_manager.IsStopLossProfitable(sell_basket))

```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
{  
    g_stacking_manager.ManageStacking(POSITION_TYPE_SELL, sell_basket);  
    g_trade_manager.Refresh(); // Refresh cache after stacking to include new position  
    sell_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_SELL);  
    g tsl_manager.ManageBasketTSL(POSITION_TYPE_SELL, sell_basket); // update TSL for  
expanded basket  
    }  
    else if (!sell_basket.HasStacked)  
    {  
        g_dca_manager.ManageDCA(POSITION_TYPE_SELL, sell_basket);  
        g_trade_manager.Refresh(); // Refresh cache after DCA to include new position  
        sell_basket = g_trade_manager.GetCachedBasket(POSITION_TYPE_SELL);  
        g_ts_manager.ManageBasketTSL(POSITION_TYPE_SELL, sell_basket); // update TSL for  
expanded basket  
    }  
    g_trade_manager.ManageBasketTP(sell_basket); // Set basket TP if expanded  
}  
}  
  
// --- ENTRY LOGIC (runs on new bar only) ---  
if (!IsNewBar(CSettings::EaHeartbeatTimeframe)) return; // Throttle to heartbeat timeframe  
if (!g_money_manager.CheckDrawdown()) return; // Risk check: stop if drawdown too high  
if (!g_time_manager.IsTradeTimeAllowed()) return; // Time filter  
if (g_news_manager.IsNewsBlockActive()) return; // News filter  
if (SymbolInfoInteger(_Symbol, SYMBOL_SPREAD) > CSettings::MaxSpreadPoints) return; // Spread  
filter  
int signal = g_signal_manager.GetFinalSignal(); // Aggregate signal from all sources  
  
// Check for BUY entry: signal, permissions, no existing basket  
if (signal == SIGNAL_BUY && CSettings::AllowLongTrades &&  
!g_trade_manager.HasCachedBasket(POSITION_TYPE_BUY))  
{  
    double lots = g_money_manager.GetInitialLotSize(); // Calculate lot size based on mode  
    g_trade_manager.OpenTrade(signal, lots, CSettings::SlPips, CSettings::InitialTpPips, "INIT",  
1);  
}  
  
// Check for SELL entry: signal, permissions, no existing basket  
if (signal == SIGNAL_SELL && CSettings::AllowShortTrades &&  
!g_trade_manager.HasCachedBasket(POSITION_TYPE_SELL))  
{  
    double lots = g_money_manager.GetInitialLotSize(); // Calculate lot size based on mode  
    g_trade_manager.OpenTrade(signal, lots, CSettings::SlPips, CSettings::InitialTpPips, "INIT",  
1);  
}  
}  
//-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               Settings.mqh
//| FXATM Configuration Management System
//| Copyright 2025, LAWRENCE KOH
//|           lawrancekoh@outlook.com
//+-----+
//| PURPOSE:
//|   Central configuration repository for FXATM Expert Advisor
//|   Manages all input parameters, enums, and static settings
//|
//| KEY COMPONENTS:
//|   • 6 Lot sizing modes (Fixed, Risk%, ATR-Volatility Adjusted)
//|   • 5 Trailing Stop Loss modes (Step, ATR, MA, High/Low)
//|   • Signal configuration structures for polymorphic signals
//|   • Static settings class with global parameter access
//|
//| USAGE:
//|   Include this file to access CSettings class and enumerations
//|   All EA parameters are centralized here for consistency
//+-----+
#property link      "lawrancekoh@outlook.com"

#include <Object.mqh>

// Generic Trade Signals
enum ENUM_TRADE_SIGNAL
{
    SIGNAL_NONE,
    SIGNAL_BUY,
    SIGNAL_SELL,
    SIGNAL_CLOSE_BUY,
    SIGNAL_CLOSE_SELL
};

// B. LOT SIZING SETTINGS
enum ENUM_LOT_SIZING_MODE
{
    MODE_FIXED_LOT,
    MODE_LOTS_PER_THOUSAND_BALANCE,
    MODE_LOTS_PER_THOUSAND_EQUITY,
    MODE_RISK_PERCENT_BALANCE,
    MODE_RISK_PERCENT_EQUITY,
    MODE_VOLATILITY_ADJUSTED
};

// E. TRAILING STOP SETTINGS
enum ENUM_TSL_MODE
{
    MODE_TSL_NONE,
    MODE_TSL_STEP,
    MODE_TSL_ATR,
    MODE_TSL_MOVING_AVERAGE,
    MODE_TSL_HIGH_LOW_BAR
};

// H. NEWS FILTER SETTINGS
enum ENUM_NEWS_SOURCE
{
    MODE_DISABLED,
    MODE_MT5_BUILT_IN,
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
    MODE_WEB_REQUEST  
};  
  
// STACKING LOT MODE SETTINGS  
enum ENUM_STACKING_LOT_MODE  
{  
    MODE_FIXED,  
    MODE_LAST_TRADE,  
    MODE_BASKET_TOTAL,  
    MODE_ENTRY_BASED  
};  
  
// I. SIGNAL SETTINGS  
enum ENUM_SIGNAL_TYPE  
{  
    SIGNAL_TYPE_NONE,  
    SIGNAL_RSI,  
    SIGNAL_MACD,  
    SIGNAL_MA_CROSS,  
    SIGNAL_STOCHASTIC,  
    SIGNAL_BOLLINGER_BANDS  
};  
enum ENUM_SIGNAL_ROLE  
{  
    ROLE_BIAS,  
    ROLE_ENTRY  
};  
struct CSignalParams  
{  
    int     IntParams[4];  
    double DoubleParams[4];  
    bool    BoolParams[4];  
    ENUM_APPLIED_PRICE Price;  
    ENUM_MA_METHOD    MaMethod1;  
    ENUM_MA_METHOD    MaMethod2;  
    ENUM_STO_PRICE    PriceField;  
};  
struct CSignalSettings  
{  
    ENUM_SIGNAL_TYPE    Type;  
    ENUM_SIGNAL_ROLE    Role;  
    ENUM_TIMEFRAMES    Timeframe;  
    CSignalParams Params;  
};  
  
//+-----+  
//| CSettings class  
//| A static-like class to hold all EA settings.  
//+-----+  
class CSettings  
{  
public:  
    // A. GENERAL SETTINGS  
    static string    EaName;  
    static long      EaMagicNumber;  
    static int       MaxSpreadPoints;  
    static int       MaxSlippagePoints;  
    static double    MaxDrawdownPercent;  
    static ENUM_TIMEFRAMES EaHeartbeatTimeframe;  
    static bool      AllowLongTrades;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
static bool      AllowShortTrades;
static string    Symbol;

// B. POSITION MANAGEMENT SETTINGS (Lot sizing + Basket TP/SL)
static ENUM_LOT_SIZE_MODE LotSizingMode;
static double    LotFixed;
static double    LotsPerThousand;
static double    LotRiskPercent;
static int       S1Pips;
static int       InitialTpPips;
static int       BasketTpPips;

// C. LOSS MANAGEMENT SETTINGS (DCA)
static int       DcaMaxTrades;
static int       DcaTriggerPips;
static double   DcaStepMultiplier;
static double   DcaLotMultiplier;
static int       DcaLotMultiplierStart;

// D. PROFIT MANAGEMENT SETTINGS (TSL + Stacking)
static ENUM_TSL_MODE TslMode;
static int       TslBeTriggerPips;
static int       BeOffsetPips;
static int       TslStepPips;
static bool     TslRemoveTp;
static bool     BreakevenIncludesCosts;
static double   CommissionPerLot;
static int       TslAtrPeriod;
static double   TslAtrMultiplier;
static int       TslMaPeriod;
static ENUM_MA_METHOD TslMaMethod;
static ENUM_APPLIED_PRICE TslMaPrice;
static int       TslHiLoPeriod;
static int       StackingMaxTrades;
static int       StackingTriggerPips;
static double   StackingLotSize;
static ENUM_STACKING_LOT_MODE StackingLotMode;

// E. ADVANCED EXIT SETTINGS (Partial TP)
static int       PartialTpTriggerPips;
static double   PartialTpClosePercent;
static bool     PartialTpSetBe;

// F. FILTER SETTINGS (Time + News)
static string   EaTradingDays;
static string   EaTradingTimeStart;
static string   EaTradingTimeEnd;
static ENUM_NEWS_SOURCE NewsSourceMode;
static string   NewsCalendarURL;
static int       NewsMinsBefore;
static int       NewsMinsAfter;
static bool     NewsFilterHighImpact;
static bool     NewsFilterMedImpact;
static bool     NewsFilterLowImpact;
static string   NewsFilterCurrencies;

// G. SIGNAL SETTINGS
static CSignalSettings Signal1;
static CSignalSettings Signal2;
static CSignalSettings Signal3;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
// H. SIGNAL MANAGER SETTINGS
static int BiasPersistenceBars;

// I. CHART UI SETTINGS
static bool      ChartShowPanels;
static ENUM_BASE_CORNER ChartPanelCorner;
static color     ChartColorBackground;
static color     ChartColorTextMain;
static color     ChartColorBuy;
static color     ChartColorSell;
static color     ChartColorNeutral;

// J. BACKTEST MODE DETECTION
static bool      IsBacktestMode;
};

//+-----+
//| Static member initialization
//+-----+
string   CSettings::EaName;
long     CSettings::EaMagicNumber;
int      CSettings::MaxSpreadPoints = 0;
int      CSettings::MaxSlippagePoints = 0;
double   CSettings::MaxDrawdownPercent;
ENUM_TIMEFRAMES CSettings::EaHeartbeatTimeframe;
bool    CSettings::AllowLongTrades;
bool    CSettings::AllowShortTrades;
string   CSettings::Symbol;

ENUM_LOT_SIZING_MODE CSettings::LotSizingMode;
double   CSettings::LotFixed;
double   CSettings::LotsPerThousand;
double   CSettings::LotRiskPercent;
int     CSettings::SLPips = 0;
int     CSettings::InitialTpPips = 0;
int     CSettings::BasketTpPips = 0;

int     CSettings::DcaMaxTrades = 0;
int     CSettings::DcaTriggerPips = 0;
double   CSettings::DcaStepMultiplier;
double   CSettings::DcaLotMultiplier;
int     CSettings::DcaLotMultiplierStart = 0;

ENUM_TSL_MODE CSettings::TslMode;
int     CSettings::TslBeTriggerPips = 0;
int     CSettings::BeOffsetPips = 0;
int     CSettings::TslStepPips = 0;
bool    CSettings::TslRemoveTp;
bool    CSettings::BreakevenIncludesCosts;
double   CSettings::CommissionPerLot;
int     CSettings::TslAtrPeriod = 0;
double   CSettings::TslAtrMultiplier;
int     CSettings::TslMaPeriod = 0;
ENUM_MA_METHOD CSettings::TslMaMethod;
ENUM_APPLIED_PRICE CSettings::TslMaPrice;
int     CSettings::TslHiLoPeriod = 0;
int     CSettings::StackingMaxTrades = 0;
int     CSettings::StackingTriggerPips = 0;
double   CSettings::StackingLotSize;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
ENUM_STACKING_LOT_MODE CSettings::StackingLotMode;

int      CSettings::PartialTpTriggerPips = 0;
double   CSettings::PartialTpClosePercent;
bool     CSettings::PartialTpSetBe;

string   CSettings::EaTradingDays;
string   CSettings::EaTradingTimeStart;
string   CSettings::EaTradingTimeEnd;
ENUM_NEWS_SOURCE CSettings::NewsSourceMode;
string   CSettings::NewsCalendarURL;
int      CSettings::NewsMinsBefore = 0;
int      CSettings::NewsMinsAfter = 0;
bool    CSettings::NewsFilterHighImpact;
bool    CSettings::NewsFilterMedImpact;
bool    CSettings::NewsFilterLowImpact;
string   CSettings::NewsFilterCurrencies;

CSignalSettings CSettings::Signal1;
CSignalSettings CSettings::Signal2;
CSignalSettings CSettings::Signal3;

int      CSettings::BiasPersistenceBars = 0;

bool    CSettings::ChartShowPanels;
ENUM_BASE_CORNER CSettings::ChartPanelCorner;
color   CSettings::ChartColorBackground;
color   CSettings::ChartColorTextMain;
color   CSettings::ChartColorBuy;
color   CSettings::ChartColorSell;
color   CSettings::ChartColorNeutral;

bool    CSettings::IsBacktestMode;
//-----+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               TradeManager.mqh |
//| Copyright 2025, LAWRENCE KOH           |
//| lawrancekoh@outlook.com               |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include <Trade\Trade.mqh>
#include "Settings.mqh"
#include "MoneyManager.mqh"
#include "Basket.mqh"

//--- Signal definitions
#define SIGNAL_BUY 1
#define SIGNAL_SELL -1
#define SIGNAL_NONE 0

class CTradeManager
{
private:
    CTrade    m_trade;
    string    m_symbol;

    // Basket caching for performance optimization
    CBasket  m_buy_basket_cache;
    CBasket  m_sell_basket_cache;
    bool      m_cache_valid;

    //+-----+
    //| Generates structured comment for trades          |
    //+-----+
    string GenerateComment(int signal, string trade_type, int serial_number)
    {
        string base = CSettings::EaName;
        if(base == "") base = "FXATM";
        string direction = (signal == SIGNAL_BUY) ? "BUY" : "SELL";
        return base + " " + direction + " " + trade_type + " " + IntegerToString(serial_number);
    }

public:
    CTradeManager(void) {};
    ~CTradeManager(void) {};

    void Init()
    {
        m_symbol = Symbol();
        m_trade.SetExpertMagicNumber(CSettings::EaMagicNumber);
        m_trade.SetDeviationInPoints(CSettings::MaxSlippagePoints);
        m_trade.SetTypeFillingBySymbol(m_symbol);
    }

    //+-----+
    //| Opens a trade based on the signal.                 |
    //+-----+
    bool OpenTrade(const int signal, const double lots, const int sl_pips, const int tp_pips, string
tradeType, int serial)
    {
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
if(signal == SIGNAL_NONE)
    return false;

// Generate comment using the new parameters
string comment = GenerateComment(signal, tradeType, serial);

//--- Determine Order Type
ENUM_ORDER_TYPE order_type = (signal == SIGNAL_BUY) ? ORDER_TYPE_BUY : ORDER_TYPE_SELL;

//--- Get Current Price
double price = SymbolInfoDouble(_Symbol, (order_type == ORDER_TYPE_BUY) ? SYMBOL_ASK : SYMBOL_BID);

//--- Calculate SL/TP Prices
double pip_size = CMoneyManager::GetPipSize();
double sl_price = 0;
if(sl_pips > 0)
{
    sl_price = (order_type == ORDER_TYPE_BUY) ? price - sl_pips * pip_size : price + sl_pips * pip_size;
}

double tp_price = 0;
if(tp_pips > 0)
{
    tp_price = (order_type == ORDER_TYPE_BUY) ? price + tp_pips * pip_size : price - tp_pips * pip_size;
}

//--- Execute Trade
if(order_type == ORDER_TYPE_BUY)
{
    return m_trade.Buy(lots, _Symbol, price, sl_price, tp_price, comment);
}
else
{
    return m_trade.Sell(lots, _Symbol, price, sl_price, tp_price, comment);
}

//+-----+
//| Checks if a basket of trades is already open for this symbol and direction. |
//+-----+
bool HasOpenBasket(ENUM_POSITION_TYPE direction = POSITION_TYPE_BUY)
{
    for(int i = PositionsTotal() - 1; i >= 0; i--)
    {
        ulong ticket = PositionGetTicket(i);
        if(ticket == 0) continue;
        if(!PositionSelectByTicket(ticket)) continue;
        if(PositionGetString(POSITION_SYMBOL) != _Symbol) continue;

        // In backtest mode, skip magic number check due to Strategy Tester limitations
        if(!CSettings::IsBacktestMode)
        {
            if(PositionGetInteger(POSITION_MAGIC) != CSettings::EaMagicNumber) continue;
        }

        if((ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE) == direction) return true;
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
return false;
}

//+-----+
//| Gets the current basket state by scanning open positions for the specified direction. |
//+-----+
CBasket GetBasket(ENUM_POSITION_TYPE direction)
{
    CBasket basket;
    datetime latestTime = D'1970.01.01 00:00:00';
    datetime earliestTime = D'2030.01.01 00:00:00';
    int count = 0;
    int stacking_count = 0;
    double total_volume = 0.0;
    double weighted_price_sum = 0.0;
    double total_profit = 0.0;
    double total_costs = 0.0;

    for(int i = PositionsTotal() - 1; i >= 0; i--)
    {
        ulong ticket = PositionGetTicket(i);
        if(ticket == 0) continue;
        if(!PositionSelectByTicket(ticket)) continue;
        if(PositionGetString(POSITION_SYMBOL) != _Symbol) continue;

        // In backtest mode, skip magic number check due to Strategy Tester limitations
        if(!CSettings::IsBacktestMode)
        {
            if(PositionGetInteger(POSITION_MAGIC) != CSettings::EaMagicNumber) continue;
        }

        if((ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE) != direction) continue;

        // Add ticket to basket
        basket.AddTicket(ticket);

        count++;
        double volume = PositionGetDouble(POSITION_VOLUME);
        double entry_price = PositionGetDouble(POSITION_PRICE_OPEN);
        total_volume += volume;
        weighted_price_sum += volume * entry_price;
        total_profit += PositionGetDouble(POSITION_PROFIT);
        total_costs += PositionGetDouble(POSITION_SWAP); // POSITION_COMMISION deprecated

        // Optimized parsing: check for PTP flag first
        string comment = PositionGetString(POSITION_COMMENT);
        bool has_ptp = (StringLen(comment) == 0 || StringFind(comment, "[PTP]") != -1);

        // Parse comment for flags and basket info
        if (has_ptp) {
            basket.HasPartialTPExecuted = true;
        }

        // Parse base comment by stripping flags (anything in brackets)
        string base_comment = comment;
        int flag_pos = StringFind(base_comment, " [");
        if (flag_pos != -1) {
            base_comment = StringSubstr(base_comment, 0, flag_pos);
        }
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
// Parse comment format: base direction type serial (e.g., FXATMv4 BUY INIT 1)
// Updated to handle EA names with spaces by parsing from the end
string parts[];
int split_count = StringSplit(base_comment, ' ', parts);
if (split_count >= 4) {
    if (parts[split_count-2] == "STACK") {
        stacking_count++;
    }
} else if (StringLen(base_comment) > 0) {
    // Malformed base comment, log warning but continue
    Print("Warning: Malformed base comment '", base_comment, "' in position ", ticket);
}

datetime posTime = (datetime)PositionGetInteger(POSITION_TIME);
if(posTime > latestTime)
{
    latestTime = posTime;
    basket.Ticket = (int)ticket;
    basket.LastTradePrice = entry_price;
    basket.LastTradeLots = volume;
    basket.BasketDirection = direction;

    // Only update basket type and serial from the latest trade
    if (split_count >= 4) {
        basket.BasketType = parts[split_count-2];
        basket.SerialNumber = (int)StringToInteger(parts[split_count-1]);
    }
}
if(posTime < earliestTime)
{
    earliestTime = posTime;
    basket.InitialTradePrice = entry_price;
}
}

basket.TradeCount = count;
basket.StackingCount = stacking_count;
basket.HasStacked = stacking_count > 0;
basket.TotalVolume = total_volume;
basket.AvgEntryPrice = (total_volume > 0.0) ? weighted_price_sum / total_volume : 0.0;
basket.TotalProfit = total_profit;
basket.TotalCosts = total_costs;
return basket;
}

//+-----+
//| Sets the same Stop Loss price for all positions in the basket | +-----+
void SetBasketSL(ENUM_POSITION_TYPE direction, double sl_price)
{
    // Get broker's minimum stops level
    double stops_level = (double)SymbolInfoInteger(_Symbol, SYMBOL_TRADE_STOPS_LEVEL);
    double stops_distance = stops_level * _Point;
    double bid = SymbolInfoDouble(_Symbol, SYMBOL_BID);
    double ask = SymbolInfoDouble(_Symbol, SYMBOL_ASK);

    Print("SetBasketSL Debug: Direction: ", direction, " Target SL: ", sl_price, " Bid: ", bid, " Ask: ", ask);

    for(int i = PositionsTotal() - 1; i >= 0; i--)

```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
{  
    ulong ticket = PositionGetTicket(i);  
    if(ticket == 0) continue;  
    if(!PositionSelectByTicket(ticket)) continue;  
    if(PositionGetString(POSITION_SYMBOL) != _Symbol) continue;  
  
    // In backtest mode, skip magic number check due to Strategy Tester limitations  
    if(!CSettings::IsBacktestMode)  
    {  
        if(PositionGetInteger(POSITION_MAGIC) != CSettings::EaMagicNumber) continue;  
    }  
  
    if((ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE) != direction) continue;  
  
    // Validate stops level before modifying  
    bool is_valid_sl = true;  
    if(direction == POSITION_TYPE_BUY)  
    {  
        // For BUY: SL must be below BID by at least stops_distance  
        if(sl_price >= bid - stops_distance)  
        {  
            // Set to minimum allowed SL (small loss) to ensure SL is set  
            sl_price = bid - stops_distance - _Point * 10;  
            Print("SetBasketSL Debug: Adjusted BUY SL to ", sl_price);  
        }  
    }  
    else // POSITION_TYPE_SELL  
    {  
        // For SELL: SL must be above ASK by at least stops_distance  
        if(sl_price <= ask + stops_distance)  
        {  
            // Set to minimum allowed SL (small loss) to ensure SL is set  
            sl_price = ask + stops_distance + _Point * 10;  
            Print("SetBasketSL Debug: Adjusted SELL SL to ", sl_price);  
        }  
    }  
  
    double current_sl = PositionGetDouble(POSITION_SL);  
    double current_tp = PositionGetDouble(POSITION_TP);  
    double norm_current_sl = NormalizeDouble(current_sl, _Digits);  
    double norm_sl_price = NormalizeDouble(sl_price, _Digits);  
    if(norm_current_sl != norm_sl_price) // Modify only if SL differs  
    {  
        Print("SetBasketSL Debug: Modifying ticket ", ticket, " SL from ", current_sl, " to ",  
        sl_price);  
        if(!m_trade.PositionModify(ticket, sl_price, current_tp)) {  
            Print("SetBasketSL Debug: Failed to modify ticket ", ticket, " Error: ",  
            m_trade.ResultRetcode());  
        }  
    }  
}  
}  
  
//-----+  
//| Calculates True Break-Even price for remaining volume |  
//-----+  
double CalculateTrueBreakEvenPrice(const CBasket &basket, double remaining_vol)  
{  
    double total_costs = basket.TotalCosts;  
    double desired_profit = CMoneyManager::GetMoneyFromPips(CSettings::BeOffsetPips,
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
remaining_vol);
    double total_money_needed = desired_profit - total_costs;
    double pips_needed = CMoneyManager::GetPipsFromMoney(total_money_needed, remaining_vol);
    if (pips_needed < 0) pips_needed = 0; // Prevent setting SL to a loss level; use entry price
as BE
    double pip_size = CMoneyManager::GetPipSize();
    if (basket.BasketDirection == POSITION_TYPE_BUY)
    {
        return basket.AvgEntryPrice + pips_needed * pip_size; // SL above entry for BUY (Profit)
    }
    else
    {
        return basket.AvgEntryPrice - pips_needed * pip_size; // SL below entry for SELL (Profit)
    }
}

//+-----+
//| Calculates Basket TP price based on average entry price | 
//+-----+
double CalculateBasketTpPrice(const CBasket &basket, int tp_pips)
{
    double pip_size = CMoneyManager::GetPipSize();
    if (basket.BasketDirection == POSITION_TYPE_BUY)
    {
        return basket.AvgEntryPrice + tp_pips * pip_size;
    }
    else
    {
        return basket.AvgEntryPrice - tp_pips * pip_size;
    }
}

//+-----+
//| Manages Basket TP by setting TP on all positions when basket expands | 
//+-----+
void ManageBasketTP(const CBasket &basket)
{
    if (basket.TradeCount <= 1 || CSettings::BasketTpPips <= 0) return;
    double tp_price = CalculateBasketTpPrice(basket, CSettings::BasketTpPips);
    SetBasketTP(basket.BasketDirection, tp_price);
}

//+-----+
//| Manages Partial Take Profit with proportional distribution | 
//+-----+
void ManagePartialTP(const CBasket &basket)
{
    // Guard clauses
    if (CSettings::PartialTpTriggerPips <= 0 || basket.HasPartialTPExecuted || basket.ProfitPips() < CSettings::PartialTpTriggerPips) return;

    // Calculate target volume to close
    double target_volume_to_close = basket.TotalVolume * (CSettings::PartialTpClosePercent / 100.0);
    double actual_closed_volume = 0.0;
    double min_vol = SymbolInfoDouble(_Symbol, SYMBOL_VOLUME_MIN);
    double step = SymbolInfoDouble(_Symbol, SYMBOL_VOLUME_STEP);

    // Proportional distribution across all positions
    for (int i = 0; i < ArraySize(basket.Tickets); i++) {
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
ulong ticket = basket.Tickets[i];
if (!PositionSelectByTicket(ticket)) continue;

double pos_volume = PositionGetDouble(POSITION_VOLUME);
double proportional_close = pos_volume * (target_volume_to_close / basket.TotalVolume);
proportional_close = MathFloor(proportional_close / step) * step; // Round down

if (proportional_close >= min_vol) {
    bool close_success = m_trade.PositionClosePartial(ticket, proportional_close);
    if (close_success) {
        actual_closed_volume += proportional_close;
    } else {
        // Print("PTP: Failed to close position ", ticket, " volume ",
proportional_close);
    }
} else {
    // Print("PTP: Skipping position ", ticket, " as calculated partial close volume ",
proportional_close, " is below min volume ", min_vol);
}

if (actual_closed_volume == 0.0) return; // No closes succeeded

// Update basket after partial closes to get correct AvgEntryPrice
CBasket updated_basket = GetBasket(basket.BasketDirection);

// Set True BE SL on remaining positions if enabled
if (CSettings::PartialTpSetBe) {
    double be_price = CalculateTrueBreakEvenPrice(updated_basket, updated_basket.TotalVolume);

    // Validate BE price against current market price and stops level
    double stops_level = (double)SymbolInfoInteger(_Symbol, SYMBOL_TRADE_STOPS_LEVEL);
    double stops_distance = stops_level * _Point;
    double bid = SymbolInfoDouble(_Symbol, SYMBOL_BID);
    double ask = SymbolInfoDouble(_Symbol, SYMBOL_ASK);
    bool is_safe = false;

    if (updated_basket.BasketDirection == POSITION_TYPE_BUY) {
        // For BUY, SL must be < Bid - Stops
        if (be_price < bid - stops_distance) is_safe = true;
    } else {
        // For SELL, SL must be > Ask + Stops
        if (be_price > ask + stops_distance) is_safe = true;
    }

    if (is_safe) {
        Print("PTP Debug: Setting BE SL to ", be_price);
        SetBasketSL(updated_basket.BasketDirection, be_price);
    } else {
        Print("PTP Debug: Skipping BE SL - Price too close or in loss. BE: ", be_price, " Bid:
", bid, " Ask: ", ask);
    }
}

//+-----+
void SetBasketTP(ENUM_POSITION_TYPE direction, double tp_price)
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
{  
    for(int i = PositionsTotal() - 1; i >= 0; i--)  
    {  
        ulong ticket = PositionGetTicket(i);  
        if(ticket == 0) continue;  
        if(!PositionSelectByTicket(ticket)) continue;  
        if(PositionGetString(POSITION_SYMBOL) != _Symbol) continue;  
  
        // In backtest mode, skip magic number check due to Strategy Tester limitations  
        if(!CSettings::IsBacktestMode)  
        {  
            if(PositionGetInteger(POSITION_MAGIC) != CSettings::EaMagicNumber) continue;  
        }  
  
        if((ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE) != direction) continue;  
  
        double current_sl = PositionGetDouble(POSITION_SL);  
        double current_tp = PositionGetDouble(POSITION_TP);  
  
        double norm_current_tp = NormalizeDouble(current_tp, _Digits);  
        double norm_tp_price = NormalizeDouble(tp_price, _Digits);  
  
        if(norm_current_tp != norm_tp_price) // Modify only if TP differs  
        {  
            m_trade.PositionModify(ticket, current_sl, tp_price);  
        }  
    }  
  
    void CloseTrades(ENUM_ORDER_TYPE type)  
    {  
        // Logic to close trades of a certain type.  
    }  
  
    int GetOpenTradesCount()  
    {  
        // Logic to count open trades.  
        return 0;  
    }  
  
    //+-----+  
    //| Refresh basket cache once per tick for performance optimization |  
    //+-----+  
    void Refresh()  
    {  
        // Reset baskets  
        m_buy_basket_cache = CBasket();  
        m_sell_basket_cache = CBasket();  
  
        // Variables for weighted average calculation  
        double buy_weighted_sum = 0.0;  
        double sell_weighted_sum = 0.0;  
        datetime buy_latest = D'1970.01.01 00:00:00';  
        datetime buy_earliest = D'2030.01.01 00:00:00';  
        datetime sell_latest = D'1970.01.01 00:00:00';  
        datetime sell_earliest = D'2030.01.01 00:00:00';  
  
        // Single loop to populate both baskets simultaneously  
        for(int i = PositionsTotal() - 1; i >= 0; i--)  
        {
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
ulong ticket = PositionGetTicket(i);
if(ticket == 0 || !PositionSelectByTicket(ticket)) continue;
if(PositionGetString(POSITION_SYMBOL) != _Symbol) continue;

// In backtest mode, skip magic number check due to Strategy Tester limitations
if(!CSettings::IsBacktestMode)
{
    if(PositionGetInteger(POSITION_MAGIC) != CSettings::EaMagicNumber) continue;
}

ENUM_POSITION_TYPE direction = (ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);

// Update basket statistics based on direction
double volume = PositionGetDouble(POSITION_VOLUME);
double entry_price = PositionGetDouble(POSITION_PRICE_OPEN);

if(direction == POSITION_TYPE_BUY)
{
    // Update buy basket
    m_buy_basket_cache.AddTicket(ticket);
    m_buy_basket_cache.TradeCount++;
    m_buy_basket_cache.TotalVolume += volume;
    buy_weighted_sum += volume * entry_price;
    m_buy_basket_cache.TotalProfit += PositionGetDouble(POSITION_PROFIT);
    m_buy_basket_cache.TotalCosts += PositionGetDouble(POSITION_SWAP);

    // Optimized parsing: check for PTP flag first
    string comment = PositionGetString(POSITION_COMMENT);
    bool has_ptp = (StringLen(comment) == 0 || StringFind(comment, "[PTP]") != -1);

    // Parse comment for flags and basket info (optimized)
    if (has_ptp) {
        m_buy_basket_cache.HasPartialTPExecuted = true;
    }

    // Parse base comment by stripping flags (anything in brackets)
    string base_comment = comment;
    int flag_pos = StringFind(base_comment, "[");
    if (flag_pos != -1) {
        base_comment = StringSubstr(base_comment, 0, flag_pos);
    }

    // Parse comment format: base direction type serial (e.g., FXATMv4 BUY INIT 1)
    // Updated to handle EA names with spaces by parsing from the end
    string parts[];
    int split_count = StringSplit(base_comment, ' ', parts);
    if (split_count >= 4) {
        if (parts[split_count-2] == "STACK") {
            m_buy_basket_cache.StackingCount++;
        }
    } else if (StringLen(base_comment) > 0) {
        // Malformed base comment, log warning but continue
        Print("Warning: Malformed base comment ''", base_comment, "' in position ", ticket);
    }

    // Track latest and earliest times
    datetime postTime = (datetime)PositionGetInteger(POSITION_TIME);
    if(postTime > buy_latest)
    {
        buy_latest = postTime;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
m_buy_basket_cache.Ticket = (int)ticket;
m_buy_basket_cache.LastTradePrice = entry_price;
m_buy_basket_cache.LastTradeLots = volume;
m_buy_basket_cache.BasketDirection = direction;

// Only update basket type and serial from the latest trade
if (split_count >= 4) {
    m_buy_basket_cache.BasketType = parts[split_count-2];
    m_buy_basket_cache.SerialNumber = (int)StringToInteger(parts[split_count-1]);
}
}

if(posTime < buy_earliest)
{
    buy_earliest = posTime;
    m_buy_basket_cache.InitialTradePrice = entry_price;
}
}

else // POSITION_TYPE_SELL
{
    // Update sell basket
    m_sell_basket_cache.AddTicket(ticket);
    m_sell_basket_cache.TradeCount++;
    m_sell_basket_cache.TotalVolume += volume;
    sell_weighted_sum += volume * entry_price;
    m_sell_basket_cache.TotalProfit += PositionGetDouble(POSITION_PROFIT);
    m_sell_basket_cache.TotalCosts += PositionGetDouble(POSITION_SWAP);

    // Optimized parsing: check for PTP flag first
    string comment = PositionGetString(POSITION_COMMENT);
    bool has_ptp = (StringLen(comment) == 0 || StringFind(comment, "[PTP]") != -1);

    // Parse comment for flags and basket info (optimized)
    if (has_ptp) {
        m_sell_basket_cache.HasPartialTPExecuted = true;
    }

    // Parse base comment by stripping flags (anything in brackets)
    string base_comment = comment;
    int flag_pos = StringFind(base_comment, " [");
    if (flag_pos != -1) {
        base_comment = StringSubstr(base_comment, 0, flag_pos);
    }

    // Parse comment format: base direction type serial (e.g., FXATMv4 BUY INIT 1)
    // Updated to handle EA names with spaces by parsing from the end
    string parts[];
    int split_count = StringSplit(base_comment, ' ', parts);
    if (split_count >= 4) {
        if (parts[split_count-2] == "STACK") {
            m_sell_basket_cache.StackingCount++;
        }
    } else if (StringLen(base_comment) > 0) {
        // Malformed base comment, log warning but continue
        Print("Warning: Malformed base comment '", base_comment, "' in position ", ticket);
    }

    // Track latest and earliest times
    datetime posTime = (datetime)PositionGetInteger(POSITION_TIME);
    if(posTime > sell_latest)
    {
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
sell_latest = posTime;
m_sell_basket_cache.Ticket = (int)ticket;
m_sell_basket_cache.LastTradePrice = entry_price;
m_sell_basket_cache.LastTradeLots = volume;
m_sell_basket_cache.BasketDirection = direction;

// Only update basket type and serial from the latest trade
if (split_count >= 4) {
    m_sell_basket_cache.BasketType = parts[split_count-2];
    m_sell_basket_cache.SerialNumber = (int)StringToInteger(parts[split_count-1]);
}
if(posTime < sell_earliest)
{
    sell_earliest = posTime;
    m_sell_basket_cache.InitialTradePrice = entry_price;
}
}

// Calculate final statistics for both baskets
if(m_buy_basket_cache.TradeCount > 0)
{
    m_buy_basket_cache.AvgEntryPrice = (m_buy_basket_cache.TotalVolume > 0.0) ? buy_weighted_sum /
m_buy_basket_cache.TotalVolume : 0.0;
    m_buy_basket_cache.HasStacked = m_buy_basket_cache.StackingCount > 0;
}

if(m_sell_basket_cache.TradeCount > 0)
{
    m_sell_basket_cache.AvgEntryPrice = (m_sell_basket_cache.TotalVolume > 0.0) ?
sell_weighted_sum / m_sell_basket_cache.TotalVolume : 0.0;
    m_sell_basket_cache.HasStacked = m_sell_basket_cache.StackingCount > 0;
}

m_cache_valid = true;
}

//+-----+
//| Get cached basket state (call Refresh() first) |
//+-----+
CBasket GetCachedBasket(ENUM_POSITION_TYPE direction)
{
    if(!m_cache_valid) Refresh();
    return (direction == POSITION_TYPE_BUY) ? m_buy_basket_cache : m_sell_basket_cache;
}

//+-----+
//| Check if cached basket exists (call Refresh() first) |
//+-----+
bool HasCachedBasket(ENUM_POSITION_TYPE direction)
{
    if(!m_cache_valid) Refresh();
    CBasket basket = (direction == POSITION_TYPE_BUY) ? m_buy_basket_cache : m_sell_basket_cache;
    return basket.Ticket > 0;
}

//+-----+
//| Get the current Stop Loss price for the basket |
//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
double GetBasketSL(ENUM_POSITION_TYPE direction)
{
    for(int i = PositionsTotal() - 1; i >= 0; i--)
    {
        ulong ticket = PositionGetTicket(i);
        if(ticket == 0) continue;
        if(!PositionSelectByTicket(ticket)) continue;
        if(PositionGetString(POSITION_SYMBOL) != _Symbol) continue;

        // In backtest mode, skip magic number check due to Strategy Tester limitations
        if(!CSettings::IsBacktestMode)
        {
            if(PositionGetInteger(POSITION_MAGIC) != CSettings::EaMagicNumber) continue;
        }

        if((ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE) != direction) continue;

        // Return SL of the first matching position (they should all be the same)
        return PositionGetDouble(POSITION_SL);
    }
    return 0.0; // No positions found
}

//+-----+
//| Check if the basket's stop loss is in a profitable position |
//+-----+
bool IsStopLossProfitable(const CBasket &basket)
{
    return basket.ProfitPips() > 0;
}

};

//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                                                 +-----+
//| MoneyManager.mqh | Copyright 2025, LAWRENCE KOH |
//|                     lawrancekoh@outlook.com |-----+
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include "Settings.mqh"
#include "Basket.mqh"
#include "CatrUtility.mqh"

class CMoneyManager
{
private:
    CatrUtility* m_atr_utility;

public:
    CMoneyManager(void) : m_atr_utility(NULL) {};
    ~CMoneyManager(void) { if (m_atr_utility != NULL) delete m_atr_utility; };

    void Init()
    {
        // Nothing to do here for now
    }

    // Set ATR utility for volatility-adjusted lot sizing
    void SetAtrUtility(CatrUtility* atr_utility)
    {
        m_atr_utility = atr_utility;
    }

//+-----+
//| Validates the lot size against broker limits (min, max, step). |-----+
//+-----+
double ValidateLotSize(double lot)
{
    string symbol = CSettings::Symbol;
    double min_lot = SymbolInfoDouble(symbol, SYMBOL_VOLUME_MIN);
    double max_lot = SymbolInfoDouble(symbol, SYMBOL_VOLUME_MAX);
    double step_lot = SymbolInfoDouble(symbol, SYMBOL_VOLUME_STEP);

    // Apply limits
    lot = MathMin(lot, max_lot);
    lot = MathMax(lot, min_lot);

    // Normalize to the nearest valid step
    if(step_lot > 0)
    {
        lot = MathRound(lot / step_lot) * step_lot;
    }

    return lot;
}

//+-----+
//| Returns the pip size for the given symbol (standard 10 points). |-----+
//+-----+
static double GetPipSize(string symbol = NULL)
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
{  
    if(symbol == NULL) symbol = CSettings::Symbol;  
    double point = SymbolInfoDouble(symbol, SYMBOL_POINT);  
    return point * 10; // Standard pip size is 10 points  
}  
  
//----------------------------------------------------------------------------+  
//| Returns the value of one tick in account currency for pricing. |  
//----------------------------------------------------------------------------+  
static double GetTickValueInAccountCurrency(string symbol = NULL)  
{  
    if(symbol == NULL) symbol = CSettings::Symbol;  
    return SymbolInfoDouble(symbol, SYMBOL_TRADE_TICK_VALUE);  
}  
  
//----------------------------------------------------------------------------+  
//| Calculates the monetary risk of SL pips for one lot in account currency. |  
//----------------------------------------------------------------------------+  
static double GetSlValuePerLotInAccountCurrency(int sl_pips, string symbol = NULL)  
{  
    double pip_size = GetPipSize(symbol);  
    double tick_value = GetTickValueInAccountCurrency(symbol);  
    if(symbol == NULL) symbol = CSettings::Symbol;  
    double tick_size = SymbolInfoDouble(symbol, SYMBOL_TRADE_TICK_SIZE);  
  
    if(tick_size == 0) return 0;  
  
    // Calculate using tick size to handle instruments where tick size != point  
    return (sl_pips * pip_size / tick_size) * tick_value;  
}  
  
//----------------------------------------------------------------------------+  
//| Converts pip value to monetary value in account currency for given lot size. |  
//----------------------------------------------------------------------------+  
static double GetMoneyFromPips(double pips, double lot_size, string symbol = NULL)  
{  
    if(symbol == NULL) symbol = CSettings::Symbol;  
  
    double pip_size = GetPipSize(symbol);  
    double tick_value = GetTickValueInAccountCurrency(symbol);  
    double tick_size = SymbolInfoDouble(symbol, SYMBOL_TRADE_TICK_SIZE);  
  
    if(tick_size == 0) return 0;  
  
    // Calculate using tick size to handle instruments where tick size != point  
    return (pips * pip_size / tick_size) * tick_value * lot_size;  
}  
  
//----------------------------------------------------------------------------+  
//| Converts monetary value in account currency to pip value for given lot size. |  
//----------------------------------------------------------------------------+  
static double GetPipsFromMoney(double money, double lot_size, string symbol = NULL)  
{  
    if(symbol == NULL) symbol = CSettings::Symbol;  
  
    double pip_size = GetPipSize(symbol);  
    double tick_value = GetTickValueInAccountCurrency(symbol);  
    double tick_size = SymbolInfoDouble(symbol, SYMBOL_TRADE_TICK_SIZE);  
  
    if(tick_value == 0 || lot_size == 0 || pip_size == 0) return 0;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
// Reverse the calculation: Money = (Pips * PipSize / TickSize) * TickValue * LotSize
// Pips = Money / ( (PipSize / TickSize) * TickValue * LotSize )
return money / ((pip_size / tick_size) * tick_value * lot_size);
}

//+-----+
//| Calculates the initial lot size based on the selected mode. |
//+-----+
double GetInitialLotSize()
{
    double calculated_lot = 0.0;

    switch(CSettings::LotSizingMode)
    {
        case MODE_FIXED_LOT:
            calculated_lot = CSettings::LotFixed;
            break;

        case MODE_LOTS_PER_THOUSAND_BALANCE:
            calculated_lot = (AccountInfoDouble(ACCOUNT_BALANCE) / 1000.0) *
CSettings::LotsPerThousand;
            break;

        case MODE_LOTS_PER_THOUSAND_EQUITY:
            calculated_lot = (AccountInfoDouble(ACCOUNT_EQUITY) / 1000.0) *
CSettings::LotsPerThousand;
            break;

        case MODE_RISK_PERCENT_BALANCE:
            if(CSettings::SlPips <= 0)
            {
                Print("Risk modes require SlPips > 0. Falling back to fixed lot.");
                calculated_lot = CSettings::LotFixed;
            }
            else
            {
                double risk_amount = AccountInfoDouble(ACCOUNT_BALANCE) * (CSettings::LotRiskPercent /
100.0);
                double sl_value_per_lot = GetSlValuePerLotInAccountCurrency(CSettings::SlPips);
                if(sl_value_per_lot > 0)
                {
                    calculated_lot = risk_amount / sl_value_per_lot;
                }
                else
                {
                    Print("Unable to calculate SL value. Falling back to fixed lot.");
                    calculated_lot = CSettings::LotFixed;
                }
            }
            break;

        case MODE_RISK_PERCENT_EQUITY:
            if(CSettings::SlPips <= 0)
            {
                Print("Risk modes require SlPips > 0. Falling back to fixed lot.");
                calculated_lot = CSettings::LotFixed;
            }
            else
            {
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
double risk_amount = AccountInfoDouble(ACCOUNT_EQUITY) * (CSettings::LotRiskPercent / 100.0);

double sl_value_per_lot = GetSlValuePerLotInAccountCurrency(CSettings::SlPips);
if(sl_value_per_lot > 0)
{
    calculated_lot = risk_amount / sl_value_per_lot;
}
else
{
    Print("Unable to calculate SL value. Falling back to fixed lot.");
    calculated_lot = CSettings::LotFixed;
}
break;

case MODE_VOLATILITY_ADJUSTED:
if(m_atr_utility == NULL)
{
    Print("MODE_VOLATILITY_ADJUSTED requires ATR utility to be set. Falling back to fixed lot.");
    calculated_lot = CSettings::LotFixed;
}
else
{
    // Get base lot size using fixed mode as baseline
    double base_lot = CSettings::LotFixed;
    if(base_lot <= 0)
    {
        base_lot = SymbolInfoDouble(_Symbol, SYMBOL_VOLUME_MIN);
    }

    // Get current ATR and calculate scaling factor
    double current_atr = m_atr_utility.GetCurrentAtr();
    double scaling_factor = m_atr_utility.GetAtrMultiplierForLots(base_lot, current_atr);

    // Apply volatility adjustment
    calculated_lot = base_lot * scaling_factor;

    Print("Volatility-adjusted lot sizing: Base lot: ", base_lot,
          ", Current ATR: ", current_atr,
          ", Scaling factor: ", scaling_factor,
          ", Final lot: ", calculated_lot);
}
break;
}

return ValidateLotSize(calculated_lot);
}

//+-----+
//| Calculates the lot size for stacking trades based on the selected mode. |
//+-----+
double GetStackingLotSize(const CBasket &basket)
{
    double calculated_lot = 0.0;

    switch(CSettings::StackingLotMode)
    {
        case MODE_FIXED:
            calculated_lot = CSettings::StackingLotSize;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

break;

```
case MODE_LAST_TRADE:  
    calculated_lot = basket.LastTradeLots;  
    break;
```

```
case MODE_BASKET_TOTAL:  
    calculated_lot = basket.TotalVolume;  
    break;
```

```
case MODE_ENTRY_BASED:  
    calculated_lot = GetInitialLotSize();  
    break;  
}
```

```
return ValidateLotSize(calculated_lot);  
}
```

```
//+-----+  
//| Checks if current account drawdown exceeds the threshold. |  
//| Returns true if drawdown < MaxDrawdownPercent (trading allowed). |  
//+-----+  
bool CheckDrawdown()  
{  
    double balance = AccountInfoDouble(ACCOUNT_BALANCE);  
    double equity = AccountInfoDouble(ACCOUNT_EQUITY);  
    if(balance <= 0) return true; // Avoid division by zero  
  
    double drawdown_percent = ((balance - equity) / balance) * 100.0;  
    return drawdown_percent < CSettings::MaxDrawdownPercent;  
}
```

```
private:  
};  
//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               SignalManager.mqh |
//| Copyright 2025, LAWRENCE KOH          |
//| lawrancekoh@outlook.com           |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.01" // Updated version

#include "Settings.mqh"
#include "../Signals/ISignal.mqh"
#include <Arrays/ArrayObj.mqh>
// --- Include all signal implementations that will be created
// #include "../Signals/CSignal_RSI.mqh"
// #include "../Signals/CSignal_MACD.mqh"

/**
 * @class CSignalManager
 * @brief Manages signal aggregation with persistent bias mechanism.
 *
 * This class implements a "sticky" bias system where bias signals (ROLE_BIAS) persist
 * across multiple bars until overridden or timed out. The persistence helps maintain
 * trend direction even when bias signals temporarily disappear.
 *
 * Key Features:
 * - Bias Persistence: Once set by a bias signal, the bias holds until:
 *   1. A conflicting bias signal appears (disagreement resets to NONE).
 *   2. No bias signal reinforces it for a configurable number of bars (timeout).
 * - Timeout Mechanism: Uses a counter that increments on each GetFinalSignal() call
 *   (tied to EA heartbeat timeframe). Resets after CSettings::BiasPersistenceBars calls.
 *   Example: With M15 heartbeat and 20 bars setting, bias resets after ~5 hours.
 * - Signal Aggregation: Follows "All Must Agree" logic, but persistent bias allows
 *   entry signals to trigger trades even if no current bias signal is present.
 *
 * Usage Notes:
 * - Bias signals set the direction; entry signals trigger trades within that direction.
 * - Timeout prevents stale bias from influencing decisions indefinitely.
 * - Counter resets when bias is updated or cleared.
 */
class CSignalManager
{
private:
    CArrayObj          m_signals;
    // --- Persistent bias state ---
    int                m_current_bias;           // Current persistent bias (BUY/SELL/NONE)
    int                m_bias_timeout_counter; // Counts calls since bias was last set/updated

public:
    CSignalManager(void) : m_current_bias(SIGNAL_NONE), m_bias_timeout_counter(0) // Initialize bias
    to NONE and counter to 0
    {
    }

    ~CSignalManager(void)
    {
        for(int i = 0; i < m_signals.Total(); i++)
        {
            delete m_signals.At(i);
        }
        m_signals.Clear();
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
};

void AddSignal(ISignal* signal)
{
    if (signal == NULL)
    {
        Print("SignalManager: Attempted to add null signal pointer");
        return;
    }
    m_signals.Add(signal);
}

/**
 * @brief Gets the final, combined trading signal, now with persistent bias.
 *
 * 1. It first checks all signals on the CURRENT bar for new triggers.
 * 2. It checks for disagreements (e.g., two bias signals fighting).
 * 3. If a new, non-conflicting bias signal appears, it UPDATES the
 * persistent 'm_current_bias'.
 * 4. If no new bias signal appears, 'm_current_bias' KEEPS its old value.
 * 5. Finally, it checks the 'm_current_bias' against any ENTRY triggers.
 *
 * @return int The final trade signal (SIGNAL_BUY, SIGNAL_SELL, SIGNAL_NONE).
 */
int GetFinalSignal()
{
    // --- Bias Timeout Check ---
    // The persistent bias times out after a configurable number of bars (calls to this method).
    // This prevents stale bias from influencing decisions forever.
    // - Counter increments each time bias is active.
    // - Resets when bias is updated or when no bias is present.
    // - Tied to EA heartbeat timeframe (e.g., M15), so "bars" here mean heartbeat intervals.
    if (m_current_bias != SIGNAL_NONE)
    {
        m_bias_timeout_counter++;
        if (m_bias_timeout_counter >= CSettings::BiasPersistenceBars)
        {
            m_current_bias = SIGNAL_NONE;
            m_bias_timeout_counter = 0;
            Print("SignalManager: Bias timed out after ", CSettings::BiasPersistenceBars, " bars.
Reset to NONE.");
        }
    }
    else
    {
        m_bias_timeout_counter = 0; // Reset counter when no bias
    }

    // --- STEP 1: Check all signals for CURRENT bar triggers ---
    bool biasConfigured = false;
    bool biasBuy = false; // Represents a NEW bias signal THIS BAR
    bool biasSell = false; // Represents a NEW bias signal THIS BAR
    bool entryConfigured = false;
    bool entryBuy = false;
    bool entrySell = false;

    // Iterate through all signals
    for(int i = 0; i < m_signals.Total(); i++)
    {
        ISignal* sig = m_signals.At(i);
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
if(sig == NULL) continue;

int signal = sig.GetSignal();
ENUM_SIGNAL_ROLE role = sig.GetRole();

if(role == ROLE_BIAS)
{
    biasConfigured = true;
    if(signal == SIGNAL_BUY) biasBuy = true;
    else if(signal == SIGNAL_SELL) biasSell = true;
}
else if(role == ROLE_ENTRY)
{
    entryConfigured = true;
    if(signal == SIGNAL_BUY) entryBuy = true;
    else if(signal == SIGNAL_SELL) entrySell = true;
}

// --- STEP 2: Update the persistent 'm_current_bias' ---

// --- Check for Bias Disagreements ---
// If multiple bias signals disagree (e.g., one BUY, one SELL), reset the persistent bias
// to NONE immediately. This prevents conflicting bias from persisting.
if(biasBuy && biasSell)
{
    Print("SignalManager: Bias signal disagreement on current bar. Bias reset to NONE.");
    m_current_bias = SIGNAL_NONE; // Explicit reset on disagreement
    return SIGNAL_NONE;
}

// --- Update Persistent Bias ---
// Set or reinforce the persistent bias if a clear signal appears.
// Reset the timeout counter to start fresh persistence period.
if(biasBuy)
{
    m_current_bias = SIGNAL_BUY;
    m_bias_timeout_counter = 0; // Reset counter on bias update
}
else if(biasSell)
{
    m_current_bias = SIGNAL_SELL;
    m_bias_timeout_counter = 0; // Reset counter on bias update
}
// Note: If no new bias signal, m_current_bias persists from previous bars.

// --- STEP 3: Final decision matrix using persistent bias ---

// Check for entry-level disagreements
if(entryBuy && entrySell)
{
    Print("SignalManager: Entry signal disagreement. No action taken.");
    return SIGNAL_NONE;
}

// Check for a BUY signal
if((m_current_bias == SIGNAL_BUY || !biasConfigured) && // Bias is BUY (or no bias is set)
   (entryBuy || !entryConfigured) && // Entry is BUY (or no entry is set)
   (m_current_bias == SIGNAL_BUY || entryBuy)) // At least one of them MUST be BUY
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
{  
    return SIGNAL_BUY;  
}  
  
// Check for a SELL signal  
if((m_current_bias == SIGNAL_SELL || !biasConfigured) && // Bias is SELL (or no bias is set)  
    (entrySell || !entryConfigured) && // Entry is SELL (or no entry is set)  
    (m_current_bias == SIGNAL_SELL || entrySell)) // At least one of them MUST be SELL  
{  
    return SIGNAL_SELL;  
}  
  
// Default: No signal  
return SIGNAL_NONE;  
}  
  
/**  
 * @brief Gets the status of all signals AND the current persistent bias.  
 */  
string GetStatus()  
{  
    string status = StringFormat("Current Bias: %s | ", GetSignalString(m_current_bias));  
    for(int i = 0; i < m_signals.Total(); i++)  
    {  
        ISignal* sig = m_signals.At(i);  
        if(sig == NULL) continue;  
        string roleStr = (sig.GetRole() == ROLE_BIAS) ? "[Bias]" : "[Entry]";  
        string tfStr = GetTimeframeString(sig.GetTimeframe());  
        status += roleStr + " " + sig.GetStatus() + " " + tfStr + " | ";  
    }  
    // Remove trailing " | "  
    if(StringLen(status) > 3)  
        status = StringSubstr(status, 0, StringLen(status) - 3);  
    return status;  
}  
  
private:  
    string GetSignalString(int signal)  
    {  
        switch(signal)  
        {  
            case SIGNAL_BUY: return "BUY";  
            case SIGNAL_SELL: return "SELL";  
            default: return "NONE";  
        }  
    }  
  
    string GetTimeframeString(ENUM_TIMEFRAMES timeframe)  
    {  
        switch(timeframe)  
        {  
            case PERIOD_M1: return "M1";  
            case PERIOD_M5: return "M5";  
            case PERIOD_M15: return "M15";  
            case PERIOD_M30: return "M30";  
            case PERIOD_H1: return "H1";  
            case PERIOD_H4: return "H4";  
            case PERIOD_D1: return "D1";  
            case PERIOD_W1: return "W1";  
            case PERIOD_MN1: return "MN1";  
        }  
    }
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
    default: returnEnumToString(timeframe);  
}  
}  
};  
//+-----+  
-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//| DCAManager.mqh |
//| Copyright 2025, LAWRENCE KOH |
//| lawrancekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include "Settings.mqh"
#include "TradeManager.mqh"
#include "MoneyManager.mqh"

class CDCAManager
{
private:
    CTradeManager* m_trade_manager;
    CMoneyManager* m_money_manager;

public:
    CDCAManager(void){};
    ~CDCAManager(void){};

    void SetTradeManager(CTradeManager* tm) { m_trade_manager = tm; }
    void SetMoneyManager(CMoneyManager* mm) { m_money_manager = mm; }

    void Init()
    {
        // Nothing to do here for now
    }

    void ManageDCA(ENUM_POSITION_TYPE direction, const CBasket &basket)
    {
        // DCA Guard Clauses - check first before any modifications
        if (CSettings::DcaMaxTrades <= 0 || basket.Ticket == 0) return; // DCA disabled or no
basket
        if (!m_money_manager.CheckDrawdown()) return; // Risk check: high drawdown blocks DCA
        if (basket.TradeCount >= CSettings::DcaMaxTrades) return; // Max trades reached

        double pip_size = CMoneyManager::GetPipSize();

        // Get current market price for drawdown calculation
        double market_price = (basket.BasketDirection == POSITION_TYPE_BUY) ?
            SymbolInfoDouble(_Symbol, SYMBOL_BID) :
            SymbolInfoDouble(_Symbol, SYMBOL_ASK);

        // Calculate required drawdown pips for this DCA level (increases with each trade)
        double required_drawdown_pips = CSettings::DcaTriggerPips;
        for(int i = 1; i < basket.TradeCount; i++)
        {
            required_drawdown_pips *= CSettings::DcaStepMultiplier;
        }

        // Calculate actual drawdown in pips from last trade
        double drawdown_pips = 0;
        if (basket.BasketDirection == POSITION_TYPE_BUY)
            drawdown_pips = (basket.LastTradePrice - market_price) / pip_size;
        else
            drawdown_pips = (market_price - basket.LastTradePrice) / pip_size;
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
// Check if drawdown meets DCA trigger threshold
if (drawdown_pips < required_drawdown_pips) return;

// Calculate DCA Lot Size (apply multiplier after certain trades)
double dca_lot;
if (basket.TradeCount >= CSettings::DcaLotMultiplierStart)
{
    dca_lot = basket.LastTradeLots * CSettings::DcaLotMultiplier; // Increase lot size
}
else
{
    dca_lot = basket.LastTradeLots; // Same lot size as previous trade
}
dca_lot = m_money_manager.ValidateLotSize(dca_lot); // Ensure broker compliance

// Execute DCA Trade in same direction as basket
int signal = (basket.BasketDirection == POSITION_TYPE_BUY) ? SIGNAL_BUY : SIGNAL_SELL;
m_trade_manager.OpenTrade(signal, dca_lot, CSettings::SlPips, 0, "DCA", basket.TradeCount
+ 1);

// Refresh basket cache to include the new DCA position
m_trade_manager.Refresh();

// Fetch updated basket to ensure we use the new AvgEntryPrice
CBasket updated_basket = m_trade_manager.GetCachedBasket(direction);

// Update basket SL/TP after successful DCA trade (uniform risk management)
// Use updated_basket to be consistent, though InitialTradePrice should be invariant
double basket_sl_price = updated_basket.InitialTradePrice + (direction ==
POSITION_TYPE_BUY ? -CSettings::SlPips * pip_size : CSettings::SlPips * pip_size);
m_trade_manager.SetBasketSL(direction, basket_sl_price); // Set SL for entire basket

// Set uniform basket TP if enabled
if(CSettings::BasketTpPips > 0)
{
    // Use updated_basket.AvgEntryPrice which includes the new DCA trade
    double basket_tp_price = updated_basket.AvgEntryPrice + (direction ==
POSITION_TYPE_BUY ? CSettings::BasketTpPips * pip_size : -CSettings::BasketTpPips * pip_size);
    m_trade_manager.SetBasketTP(direction, basket_tp_price); // Set TP for entire basket
}
}

};

//-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               TrailingStopManager.mqh |
//| Copyright 2025, LAWRENCE KOH |
//|           lawrancekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include "Settings.mqh"
#include <Trade/Trade.mqh>
#include "MoneyManager.mqh"
#include "CatrUtility.mqh"

class CTrailingStopManager : public CObject
{
private:
    CTrade m_trade;
    CatrUtility m_atr_utility;

    void HandleSteppedTSL(const CBasket &basket);
    void HandleAtrTsl(const CBasket &basket);

public:
    CTrailingStopManager(void) {};
    ~CTrailingStopManager(void) {};

    void Init()
    {
        m_trade.SetExpertMagicNumber(CSettings::EaMagicNumber);

        // Initialize ATR utility for ATR-based trailing stops
        if (!m_atr_utility.Init(CSettings::TslAtrPeriod, PERIOD_CURRENT))
        {
            Print("TrailingStopManager::Init: Failed to initialize ATR utility");
        }
    }

    void ManageBasketTSL(const ENUM_POSITION_TYPE direction, const CBasket &basket)
    {
        if (CSettings::TslMode == MODE_TSL_NONE)
        {
            return;
        }

        switch(CSettings::TslMode)
        {
            case MODE_TSL_STEP:
                HandleSteppedTSL(basket);
                break;
            case MODE_TSL_ATR:
                HandleAtrTsl(basket);
                break;
            // Other cases will be added in a later phase
        }
    }
};

void CTrailingStopManager::HandleSteppedTSL(const CBasket &basket)
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
if(basket.Ticket == 0) return;

double market_price = (basket.BasketDirection == POSITION_TYPE_BUY) ? SymbolInfoDouble(_Symbol,
SYMBOL_BID) : SymbolInfoDouble(_Symbol, SYMBOL_ASK);

// Use cached basket data for performance optimization
double total_profit_money = basket.TotalProfit;
double total_costs = basket.TotalCosts;

// Calculate average profit in pips per lot
double tick_value = SymbolInfoDouble(_Symbol, SYMBOL_TRADE_TICK_VALUE);
double tick_size = SymbolInfoDouble(_Symbol, SYMBOL_TRADE_TICK_SIZE);
double pip_value_per_lot = tick_value * (CMoneyManager::GetPipSize() / tick_size); // Value of 1
pip for 1 lot

double average_profit_pips = 0.0;
if(pip_value_per_lot > 0 && basket.TotalVolume > 0)
{
    average_profit_pips = total_profit_money / (basket.TotalVolume * pip_value_per_lot);
}

// Debug removed

if(average_profit_pips < CSettings::TslBeTriggerPips) return;

double pip_size = CMoneyManager::GetPipSize();
double breakeven_price;
if(!CSettings::BreakEvenIncludesCosts)
{
    breakeven_price = basket.AvgEntryPrice;
    if(basket.BasketDirection == POSITION_TYPE_BUY)
        breakeven_price += CSettings::BeOffsetPips * pip_size;
    else
        breakeven_price -= CSettings::BeOffsetPips * pip_size;
}
else
{
    // Account for estimated commission on close
    total_costs -= basket.TotalVolume * CSettings::CommissionPerLot;
    double desired_profit = CMoneyManager::GetMoneyFromPips(CSettings::BeOffsetPips,
basket.TotalVolume);
    double total_money_to_cover = desired_profit - total_costs;
    double total_pips_to_cover = CMoneyManager::GetPipsFromMoney(total_money_to_cover,
basket.TotalVolume);
    if(total_pips_to_cover < 0) total_pips_to_cover = 0; // Prevent loss SL
    double offset = total_pips_to_cover * pip_size;
    breakeven_price = basket.AvgEntryPrice;
    if(basket.BasketDirection == POSITION_TYPE_BUY)
        breakeven_price += offset;
    else
        breakeven_price -= offset;
}

double profit_beyond_trigger = average_profit_pips - CSettings::TslBeTriggerPips;
int steps = 0;
if(profit_beyond_trigger > 0 && CSettings::TslStepPips > 0)
    steps = (int)floor(profit_beyond_trigger / CSettings::TslStepPips);

double new_sl_price = 0;
if(basket.BasketDirection == POSITION_TYPE_BUY)
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
new_sl_price = breakeven_price + (steps * CSettings::TslStepPips *
CMoneyManager::GetPipSize());
else
    new_sl_price = breakeven_price - (steps * CSettings::TslStepPips *
CMoneyManager::GetPipSize());

double stop_level_dist = (double)SymbolInfoInteger(_Symbol, SYMBOL_TRADE_STOPS_LEVEL) * _Point;

for(int i = 0; i < ArraySize(basket.Tickets); i++)
{
    ulong ticket = basket.Tickets[i];
    if(!PositionSelectByTicket(ticket)) continue;

    double current_sl = PositionGetDouble(POSITION_SL);

    // Debug: Print values for troubleshooting

    if(basket.BasketDirection == POSITION_TYPE_BUY && new_sl_price >= market_price -
stop_level_dist)
    {

        continue;
    }
    if(basket.BasketDirection == POSITION_TYPE_SELL && new_sl_price <= market_price +
stop_level_dist)
    {

        continue;
    }

    double current_tp = PositionGetDouble(POSITION_TP);
    double new_tp = CSettings::TslRemoveTp ? 0 : current_tp;

    // Skip modification if both SL and TP are essentially the same (prevent invalid stops on
no-change)
    if(MathAbs(new_sl_price - current_sl) < 0.00001 && MathAbs(new_tp - current_tp) < 0.00001) {
        continue;
    }

    // Check for minimum meaningful difference (prevent floating-point precision issues)
    double min_diff = _Point * 2; // Minimum 2 points difference
    if(MathAbs(new_sl_price - current_sl) < min_diff) {
        continue;
    }

    if(basket.BasketDirection == POSITION_TYPE_BUY && new_sl_price <= current_sl) {
        continue;
    }
    if(basket.BasketDirection == POSITION_TYPE_SELL && (new_sl_price >= current_sl && current_sl
!= 0.0)) {
        continue;
    }

    m_trade.PositionModify(ticket, new_sl_price, new_tp);
}

void CTrailingStopManager::HandleAtrTsl(const CBasket &basket)
{
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
if(basket.Ticket == 0) return;

double market_price = (basket.BasketDirection == POSITION_TYPE_BUY) ? SymbolInfoDouble(_Symbol,
SYMBOL_BID) : SymbolInfoDouble(_Symbol, SYMBOL_ASK);

// Calculate ATR-based trailing stop level using current market price for true trailing
double pip_size = CMoneyManager::GetPipSize();
double new_sl_price = m_atr_utility.GetAtrBasedLevel(market_price, CSettings::TslAtrMultiplier,
basket.BasketDirection == POSITION_TYPE_BUY, pip_size);

double stop_level_dist = (double)SymbolInfoInteger(_Symbol, SYMBOL_TRADE_STOPS_LEVEL) * _Point;

// Validate ATR-based SL against broker requirements
bool is_valid_sl = true;
if(basket.BasketDirection == POSITION_TYPE_BUY)
{
    // For BUY: SL must be below BID by at least stops_distance
    if(new_sl_price >= market_price - stop_level_dist)
    {
        Print("ATR TSL: Invalid SL for BUY basket. SL: ", new_sl_price, " would be too close to
market price: ", market_price);
        return;
    }
}
else // POSITION_TYPE_SELL
{
    // For SELL: SL must be above ASK by at least stops_distance
    if(new_sl_price <= market_price + stop_level_dist)
    {
        Print("ATR TSL: Invalid SL for SELL basket. SL: ", new_sl_price, " would be too close to
market price: ", market_price);
        return;
    }
}

// Apply ATR-based trailing stop to all positions in basket
for(int i = 0; i < ArraySize(basket.Tickets); i++)
{
    ulong ticket = basket.Tickets[i];
    if(!PositionSelectByTicket(ticket)) continue;

    double current_sl = PositionGetDouble(POSITION_SL);
    double current_tp = PositionGetDouble(POSITION_TP);
    double new_tp = CSettings::TslRemoveTp ? 0 : current_tp;

    // Skip modification if both SL and TP are essentially the same (prevent invalid stops on
no-change)
    if (MathAbs(new_sl_price - current_sl) < 0.00001 && MathAbs(new_tp - current_tp) < 0.00001) {
        continue;
    }

    // Check for minimum meaningful difference (prevent floating-point precision issues)
    double min_diff = _Point * 2; // Minimum 2 points difference
    if(MathAbs(new_sl_price - current_sl) < min_diff) {
        continue;
    }

    // Apply "Better Price" rule: only modify if new SL improves current SL
    if(basket.BasketDirection == POSITION_TYPE_BUY && new_sl_price <= current_sl) {
        continue;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
        }
        if(basket.BasketDirection == POSITION_TYPE_SELL && (new_sl_price >= current_sl && current_sl
!= 0.0)) {
            continue;
        }

        Print("ATR TSL: Modifying position ", ticket, " SL from ", current_sl, " to ", new_sl_price,
" (ATR multiplier: ", CSettings::TslAtrMultiplier, ")");
        if(!m_trade.PositionModify(ticket, new_sl_price, new_tp))
        {
            Print("ATR TSL: Failed to modify position ", ticket, " Error: ",
m_trade.ResultRetcode());
        }
    }
//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//| TimeManager.mqh |
//| Copyright 2025, LAWRENCE KOH |
//| lawrancekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include "Settings.mqh"

class CTimeManager
{
private:
    int m_start_mins;
    int m_end_mins;
    bool m_allowed_days[7]; // 0=Sunday, 1=Monday, ..., 6=Saturday

public:
    CTimeManager(void) {};
    ~CTimeManager(void) {};

    void Init()
    {
        // Pre-calculate start/end times in minutes from midnight
        string start_time = CSettings::EaTradingTimeStart;
        m_start_mins = (int)StringToInteger(StringSubstr(start_time, 0, 2)) * 60 +
                      (int)StringToInteger(StringSubstr(start_time, 3, 2));

        string end_time = CSettings::EaTradingTimeEnd;
        m_end_mins = (int)StringToInteger(StringSubstr(end_time, 0, 2)) * 60 +
                     (int)StringToInteger(StringSubstr(end_time, 3, 2));

        // Pre-calculate allowed days as boolean array
        string days_str = "," + CSettings::EaTradingDays + ",";
        StringReplace(days_str, " ", ""); // Handle spaces in input
        for(int i = 0; i < 7; i++)
        {
            string day_check = "," + IntegerToString(i) + ",";
            m_allowed_days[i] = (StringFind(days_str, day_check) != -1);
        }
    }

    bool IsTradeTimeAllowed()
    {
        //--- Initial Check
        if(CSettings::EaTradingDays == "")
            return true;

        //--- Get Current Time
        MqlDateTime current_time;
        TimeCurrent(current_time);

        //--- Day of Week Check using pre-calculated boolean array
        if(!m_allowed_days[current_time.day_of_week])
            return false;

        //--- Time of Day Check using pre-calculated minutes
        long current_mins = current_time.hour * 60 + current_time.min;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//--- Handle Overnight Sessions (e.g., Start 22:00, End 06:00)
if(m_start_mins > m_end_mins)
{
    return (current_mins >= m_start_mins || current_mins <= m_end_mins);
}
//--- Handle Normal Day Sessions
else
{
    return (current_mins >= m_start_mins && current_mins <= m_end_mins);
}
};

//+-----+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//| NewsManager.mqh |
//| Copyright 2025, LAWRENCE KOH |
//| lawrancekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.01" // Updated version

#include "Settings.mqh"
#include <Arrays/ArrayObj.mqh>

//+-----+
//| News Event Structure |
//+-----+
struct CNewsEvent
{
    string title;
    string currency;
    string impact;
    datetime time;
};

//+-----+
//| CNewsManager Class |
//| Manages filtering of trading signals based on calendar news |
//| events. |
//+-----+
class CNewsManager
{
private:
    datetime m_next_news_time;
    bool m_news_cache_valid;

    // Web Request Cache
    CNewsEvent m_cached_events[]; // Dynamic array of news events
    datetime m_last_web_request_time;
    const int m_web_request_interval_seconds; // Interval to refresh news (e.g. 1 hour)

    //+-----+
    //| Helpers for String Processing |
    //+-----+
    string CleanQuote(string str)
    {
        if (StringLen(str) >= 2 && StringGetCharacter(str, 0) == '"' && StringGetCharacter(str, StringLen(str)-1) == '"')
        {
            return StringSubstr(str, 1, StringLen(str) - 2);
        }
        return str;
    }

    datetime ParseCsvDateTime(string dateStr, string timeStr)
    {
        // Format: MM/DD/YYYY and HH:MM
        // StringToTime converts "yyyy.mm.dd [hh:mi]"
        // We need to convert MM/DD/YYYY to yyyy.mm.dd

        string date_parts[];
        // Check for / or -
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
if(StringSplit(dateStr, '/', date_parts) != 3)
{
    if(StringSplit(dateStr, '-', date_parts) != 3) return 0;
}

// date_parts: [0]=MM, [1]=DD, [2]=YYYY
string yyyy = date_parts[2];
string mm = date_parts[0];
string dd = date_parts[1];

string formatted_time = yyyy + "." + mm + "." + dd + " " + timeStr;
return StringToTime(formatted_time);
}

//+-----+
//| Web Request Logic
//+-----+
bool FetchAndParseNews()
{
    string cookie = NULL, headers;
    char post[], result[];
    int res;
    string url = CSettings::NewsCalendarURL;
    if (url == "") url = "https://nfs.forexfactory.net/ffcal_week_this.csv"; // Fallback default

    // Reset Last Error
    ResetLastError();

    int timeout = 5000; // 5 seconds

    res = WebRequest("GET", url, cookie, NULL, timeout, post, 0, result, headers);

    if (res == -1)
    {
        Print("NewsManager: WebRequest failed. Error: ", GetLastError());
        // Check if URL is allowed
        if(GetLastError() == 4060) // ERR_FUNCTION_NOT_ALLOWED
        {
            Print("NewsManager: Please add '", url, "' to the allowed URLs in Tools->Options->Expert Advisors.");
        }
        return false;
    }
    else if (res != 200)
    {
        Print("NewsManager: WebRequest returned HTTP status ", res);
        return false;
    }

    // Process Result
    string response = CharArrayToString(result);

    // Parse CSV
    string lines[];
    int line_count = StringSplit(response, '\n', lines);

    if(line_count <= 0) return false;

    ArrayResize(m_cached_events, 0); // Clear cache
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
for(int i = 0; i < line_count; i++)
{
    string line = lines[i];
    if(StringLen(line) < 5) continue; // Skip empty lines

    string fields[];
    int field_count = StringSplit(line, ',', fields);

    if(field_count < 4) continue;

    // Expected: "Date", "Time", "Currency", "Impact", "Event"
    // We need to handle that StringSplit splits by comma, but some fields might contain comma?
    // Standard FF CSV usually puts quotes around fields.
    // Simple split might break if "Event" contains comma.
    // For robustness, we should respect quotes.
    // But StringSplit is simple.
    // Assuming FF CSV format is consistent and Event is the last field or doesn't have commas
usually.
    // If fields are quoted, we can rely on standard format.

    // Map fields
    string s_date = CleanQuote(fields[0]);
    string s_time = CleanQuote(fields[1]);
    string s_curr = CleanQuote(fields[2]);
    string s_impact = CleanQuote(fields[3]);
    string s_title = (field_count > 4) ? CleanQuote(fields[4]) : "";

    CNewsEvent event;
    event.currency = s_curr;
    event.impact = s_impact;
    event.title = s_title;
    event.time = ParseCsvDateTime(s_date, s_time);

    if(event.time > 0)
    {
        int size = ArraySize(m_cached_events);
        ArrayResize(m_cached_events, size + 1);
        m_cached_events[size] = event;
    }
}

Print("NewsManager: Successfully fetched and parsed ", ArraySize(m_cached_events), " news
events.");
return true;
}

//-----
//| Checks for blocking news events from the built-in MT5 calendar. |
//| @return true if a blocking news event is found, false otherwise |
//+-----+
bool IsMt5CalendarBlockActive()
{
    // STUBBED DUE TO COMPILER BUG: MqlCalendarValue string and enum members cannot be accessed
reliably.
    // This is a known issue in MQL5 compiler; logic preserved in comments for future remediation.
/*
//--- Define Time Window in GMT
long mins_before_sec = (long)CSettings::NewsMinsBefore * 60;
long mins_after_sec = (long)CSettings::NewsMinsAfter * 60;
datetime from = TimeGMT() - mins_before_sec;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
datetime to    = TimeGMT() + mins_after_sec;

MqlCalendarValue values_array[];

//--- Get Calendar Events
if(CalendarValueHistory(values_array, from, to) > 0)
{
    //--- Get Symbol Currencies
    string currency1 = SymbolInfoString(_Symbol, SYMBOL_CURRENCY_BASE);
    string currency2 = SymbolInfoString(_Symbol, SYMBOL_CURRENCY_PROFIT);

    //--- Loop and Filter
    int total_events = ArraySize(values_array);
    for(int i = 0; i < total_events; i++)
    {
        MqlCalendarValue event = values_array[i];

        //--- Check if the event's currency matches the symbol's currencies
        if(event.currency == currency1 || event.currency == currency2)
        {
            //--- Check if the impact level is set to be filtered
            bool is_high_impact    = (event.importance == CALENDAR_IMPORTANCE_HIGH &&
CSettings::NewsFilterHighImpact);
            bool is_medium_impact = (event.importance == CALENDAR_IMPORTANCE_MODERATE &&
CSettings::NewsFilterMedImpact);
            bool is_low_impact     = (event.importance == CALENDAR_IMPORTANCE_LOW &&
CSettings::NewsFilterLowImpact);

            if(is_high_impact || is_medium_impact || is_low_impact)
            {
                // Found a blocking event, print details and return
                PrintFormat("NEWS BLOCK: %s %s %s",
                            TimeToString(event.time, TIME_DATE | TIME_MINUTES),
                            event.currency,
                            event.name);
                return true;
            }
        }
    }
}

//--- No blocking events found (stubbed)
return false;
}

public:
//+-----+
//| Constructor
//+-----+
CNewsManager(void) : m_web_request_interval_seconds(3600) // 1 Hour default refresh
{
    m_last_web_request_time = 0;
};

//+-----+
//| Destructor
//+-----+
~CNewsManager(void)
{
    ArrayResize(m_cached_events, 0);
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
};

//+-----+
//| Initialization Method
//+-----+
void Init()
{
    m_next_news_time = 0;
    m_news_cache_valid = false;
    m_last_web_request_time = 0;
}

//+-----+
//| Refresh news cache by finding next relevant news event
//+-----+
void RefreshNewsCache()
{
    // For built-in mode
    // TODO: Implement actual news checking logic when MT5 calendar is available
    // For now, set cache as valid with no news
    m_next_news_time = 0;
    m_news_cache_valid = true;
}

//+-----+
//| Check Web Request News Block
//+-----+
bool CheckWebRequestNews()
{
    // 1. Refresh Cache if needed
    if (TimeCurrent() - m_last_web_request_time > m_web_request_interval_seconds ||
m_last_web_request_time == 0)
    {
        if (FetchAndParseNews())
        {
            m_last_web_request_time = TimeCurrent();
        }
        else
        {
            // If failed, maybe try again sooner? or just keep old cache?
            // We keep old cache but update time to retry in 5 mins maybe?
            // For now, retry normal interval to avoid spamming if error is persistent.
            m_last_web_request_time = TimeCurrent();
        }
    }

    // 2. Check cached events against current time
    long mins_before_sec = (long)CSettings::NewsMinsBefore * 60;
    long mins_after_sec = (long)CSettings::NewsMinsAfter * 60;
    datetime current_time = TimeCurrent();

    // Get Symbol Currencies
    string currency1 = SymbolInfoString(CSettings::Symbol, SYMBOL_CURRENCY_BASE);
    string currency2 = SymbolInfoString(CSettings::Symbol, SYMBOL_CURRENCY_PROFIT);

    int total = ArraySize(m_cached_events);
    for(int i = 0; i < total; i++)
    {
        CNewsEvent event = m_cached_events[i];
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
// Filter by Currency
// Check against symbol currencies
bool currency_match = (event.currency == currency1 || event.currency == currency2);

// Also check against global filter list if currencies are specified there?
// Usually we only care about symbol currencies.
// But `CSettings::NewsFilterCurrencies` exists.
// If user specified currencies, maybe we should also check those?
// The prompt implies "relevant to the current symbol".
// But existing logic stub checked `CSettings::NewsFilterCurrencies` (Wait, the stub code
checked `event.currency == currency1 || event.currency == currency2`).
// The setting `NewsFilterCurrencies` was present in `CSettings` but not used in the stub
code I saw.
// I will implement check for Symbol currencies AND the list if provided.

if (!currency_match)
{
    // Check if in allowed list
    if (StringFind(CSettings::NewsFilterCurrencies, event.currency) >= 0)
    {
        currency_match = true;
    }
}

if (!currency_match) continue;

// Filter by Impact
bool is_high = (event.impact == "High" && CSettings::NewsFilterHighImpact);
bool is_med = (event.impact == "Medium" && CSettings::NewsFilterMedImpact);
bool is_low = (event.impact == "Low" && CSettings::NewsFilterLowImpact);

if (!is_high && !is_med && !is_low) continue;

// Filter by Time
// Check if current time is within [event_time - before, event_time + after]
if (current_time >= (event.time - mins_before_sec) &&
    current_time <= (event.time + mins_after_sec))
{
    PrintFormat("NEWS BLOCK (Web): %s %s %s",
               TimeToString(event.time, TIME_DATE | TIME_MINUTES),
               event.currency,
               event.title);
    return true;
}
}

return false;
}

-----+
//| Main public method to check if trading is blocked by news.          |
//| @return true if trading is blocked, false otherwise                  |
//+-----+
bool IsNewsBlockActive()
{
switch(CSettings::NewsSourceMode)
{
case MODE_DISABLED:
    return false;
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
case MODE_MT5_BUILT_IN:  
    // Only refresh cache if invalid or if next news time is approaching/passed  
    if(!m_news_cache_valid || (m_next_news_time > 0 && TimeCurrent() >= m_next_news_time -  
CSettings::NewsMinsBefore * 60))  
    {  
        RefreshNewsCache();  
    }  
    return IsMt5CalendarBlockActive();  
  
case MODE_WEB_REQUEST:  
    return CheckWebRequestNews();  
}  
return false;  
}  
};  
//+-----+  
-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               StackingManager.mqh |
//| Copyright 2025, LAWRENCE KOH |
//|           lawrencekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrencekoh@outlook.com"
#property version   "1.00"

#include "Settings.mqh"
#include "MoneyManager.mqh"
#include "TradeManager.mqh"

class CStackingManager
{
private:
    CMoneyManager* m_money_manager;
    CTradeManager* m_trade_manager;

public:
    CStackingManager(void) {};
    ~CStackingManager(void) {};

    void SetMoneyManager(CMoneyManager* mm) { m_money_manager = mm; }
    void SetTradeManager(CTradeManager* tm) { m_trade_manager = tm; }

    void Init()
    {
        // Nothing to do here for now
    }

    void ManageStacking(ENUM_POSITION_TYPE direction, const CBasket &basket)
    {
        // Guard clauses
        if (CSettings::StackingMaxTrades <= 0 || basket.Ticket == 0) return;
        if (basket.StackingCount >= CSettings::StackingMaxTrades) return;

        // Risk check: high drawdown blocks stacking
        if (!m_money_manager.CheckDrawdown()) return;

        // Profit-based trigger check
        if (basket.ProfitPips() < CSettings::StackingTriggerPips) return;

        // Calculate Stacking Lot
        double stack_lot = m_money_manager.GetStackingLotSize(basket);

        // Capture existing basket SL before opening new trade
        double previous_basket_sl = m_trade_manager.GetBasketSL(direction);

        // Execute Stacking Trade
        int signal = (basket.BasketDirection == POSITION_TYPE_BUY) ? SIGNAL_BUY : SIGNAL_SELL;
        m_trade_manager.OpenTrade(signal, stack_lot, CSettings::SlPips, 0, "STACK",
                                basket.StackingCount + 1);

        // Refresh basket cache to include the new stacking position
        m_trade_manager.Refresh();
        CBasket updated_basket = m_trade_manager.GetCachedBasket(direction);

        // Set uniform SL on all basket positions to ensure consistency
        // We must determine whether to use the new trade's SL or the existing (potentially tighter) SL
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
double new_trade_sl = m_trade_manager.GetBasketSL(updated_basket.BasketDirection);
double sl_to_apply = new_trade_sl;

if(previous_basket_sl > 0)
{
    if(updated_basket.BasketDirection == POSITION_TYPE_BUY)
    {
        // For BUY, higher SL is better (tighter)
        if(previous_basket_sl > new_trade_sl)
            sl_to_apply = previous_basket_sl;
    }
    else
    {
        // For SELL, lower SL is better (tighter)
        if(previous_basket_sl < new_trade_sl && previous_basket_sl > 0)
            sl_to_apply = previous_basket_sl;
    }
}

if(sl_to_apply > 0)
{
    m_trade_manager.SetBasketSL(updated_basket.BasketDirection, sl_to_apply);
    // Update current_basket_sl for the subsequent BE check
    // Note: We don't declare double current_basket_sl here as it was used below,
    // but the original code declared it locally. We need to make sure subsequent code uses
    sl_to_apply or we re-declare.
}

double current_basket_sl = sl_to_apply;

// Always move SL to true breakeven when stacking triggers to account for costs
double be_price = m_trade_manager.CalculateTrueBreakEvenPrice(updated_basket,
updated_basket.TotalVolume);

// Validate BE price against market price and stops level
double stops_level = (double)SymbolInfoInteger(_Symbol, SYMBOL_TRADE_STOPS_LEVEL);
double stops_distance = stops_level * _Point;
double bid = SymbolInfoDouble(_Symbol, SYMBOL_BID);
double ask = SymbolInfoDouble(_Symbol, SYMBOL_ASK);
bool is_safe = false;

if (updated_basket.BasketDirection == POSITION_TYPE_BUY) {
    // For BUY, SL must be < Bid - Stops
    if (be_price < bid - stops_distance) is_safe = true;
} else {
    // For SELL, SL must be > Ask + Stops
    if (be_price > ask + stops_distance) is_safe = true;
}

// Only set if breakeven is better than current SL and safe
if (is_safe && ((updated_basket.BasketDirection == POSITION_TYPE_BUY && be_price >
current_basket_sl) ||
    (updated_basket.BasketDirection == POSITION_TYPE_SELL && be_price < current_basket_sl &&
current_basket_sl != 0.0))) {
    m_trade_manager.SetBasketSL(updated_basket.BasketDirection, be_price);
}
}

};

//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               ISignal.mqh |
//| Copyright 2025, LAWRENCE KOH      |
//| lawrancekoh@outlook.com        |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include <Object.mqh>
#include "../Managers/Settings.mqh"

/**
 * @brief Defines the contract for all signal-generating classes.
 *
 * This interface ensures that every signal, regardless of its underlying
 * indicator or logic, provides a consistent way for the SignalManager
 * to initialize it, retrieve trading signals, and check its status.
 */
class ISignal : public CObject
{
public:
    /**
     * @brief Initializes the signal with the required parameters.
     *
     * This method should be called once before any other methods are used.
     * It sets up the indicator handles and any other necessary configurations.
     *
     * @param settings The signal settings struct containing all parameters.
     * @return bool true if initialization is successful, false otherwise.
     */
    virtual bool Init(const CSignalSettings &settings) { return false; }

    /**
     * @brief Gets the latest trading signal.
     *
     * This is the core method that the SignalManager will call on every tick
     * or bar to determine if a trading opportunity exists.
     *
     * @return int A signal from the ENUM_TRADE_SIGNAL enumeration
     *             (e.g., SIGNAL_BUY, SIGNAL_SELL, SIGNAL_NONE).
     */
    virtual int GetSignal() { return SIGNAL_NONE; }

    /**
     * @brief Gets the current status of the signal.
     *
     * This can be used for debugging or displaying status information on the UI.
     * For example, it could return "RSI(14) is Overbought" or "MACD Cross Occurred".
     *
     * @return string A human-readable status message.
     */
    virtual string GetStatus() { return ""; }

    /**
     * @brief Gets the role of the signal (Bias or Entry).
     *
     * @return ENUM_SIGNAL_ROLE The role assigned to this signal.
     */
    virtual ENUM_SIGNAL_ROLE GetRole() { return ROLE_BIAS; }
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
/**  
 * @brief Gets the timeframe this signal operates on.  
 *  
 * @return ENUM_TIMEFRAMES The timeframe (e.g., PERIOD_M15, PERIOD_H1).  
 */  
virtual ENUM_TIMEFRAMES GetTimeframe() const { return PERIOD_CURRENT; }  
  
/**  
 * @brief Draws visual indicators on the chart when a signal triggers.  
 *  
 * @param barTime The time of the bar where the signal occurred.  
 * @param signal The signal type (SIGNAL_BUY, SIGNAL_SELL).  
 * @param signalIndex The index of the signal slot (0-2) for vertical stacking.  
 */  
virtual void DrawSignal(datetime barTime, int signal, int signalIndex) = 0;  
};  
//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               CSignal_RSI.mqh |
//| Copyright 2025, LAWRENCE KOH |
//|           lawrancekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include "ISignal.mqh"

/***
 * @brief RSI signal implementation.
 *
 * Generates buy/sell signals based on RSI level cross out of oversold/overbought levels.
 */
class CSignal_RSI : public ISignal
{
private:
    int m_handle;                      // Indicator handle
    ENUM_TIMEFRAMES m_timeframe;       // Timeframe
    int m_period;                      // RSI period
    ENUM_APPLIED_PRICE m_applied_price; // Applied price
    double m_lvl_dn;                  // Oversold level
    double m_lvl_up;                  // Overbought level
    int m_last_signal;                // Last signal for status
    ENUM_SIGNAL_ROLE m_role;          // Signal role (Bias or Entry)

public:
    /**
     * @brief Initializes the RSI signal.
     *
     * @param settings The signal settings.
     * @return bool true if successful, false otherwise.
     */
    virtual bool Init(const CSignalSettings &settings) override
    {
        // Map parameters from settings
        m_timeframe = settings.Timeframe;
        m_period = settings.Params.IntParams[0];
        m_applied_price = settings.Params.Price;
        m_lvl_dn = settings.Params.DoubleParams[0];
        m_lvl_up = settings.Params.DoubleParams[1];
        m_role = settings.Role;

        // Get indicator handle
        m_handle = iRSI(_Symbol, m_timeframe, m_period, m_applied_price);

        // Validate handle
        if (m_handle == INVALID_HANDLE)
        {
            Print("CSignal_RSI: Failed to get RSI handle for ", _Symbol, " timeframe ",
EnumToString(m_timeframe));
            return false;
        }

        m_last_signal = SIGNAL_NONE;
        return true;
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
/**  
 * @brief Gets the current trading signal.  
 *  
 * @return int SIGNAL_BUY, SIGNAL_SELL, or SIGNAL_NONE.  
 */  
virtual int GetSignal() override  
{  
    double rsi_buffer[3];  
  
    // Get data from last closed bar (shift 1) and previous (shift 2)  
    if (CopyBuffer(m_handle, 0, 1, 2, rsi_buffer) != 2)  
    {  
        return SIGNAL_NONE;  
    }  
  
    // Implement level cross out logic  
    bool buy_signal = rsi_buffer[0] > m_lvl_dn && rsi_buffer[1] <= m_lvl_dn; // Crossed up out of  
oversold  
    bool sell_signal = rsi_buffer[0] < m_lvl_up && rsi_buffer[1] >= m_lvl_up; // Crossed down out  
of overbought  
  
    // Return signal  
    if (buy_signal)  
    {  
        m_last_signal = SIGNAL_BUY;  
        return SIGNAL_BUY;  
    }  
    if (sell_signal)  
    {  
        m_last_signal = SIGNAL_SELL;  
        return SIGNAL_SELL;  
    }  
  
    m_last_signal = SIGNAL_NONE;  
    return SIGNAL_NONE;  
}  
  
/**  
 * @brief Gets the status string.  
 *  
 * @return string Status message.  
 */  
virtual string GetStatus() override  
{  
    string status = StringFormat("RSI(%d,%1f,%1f)", m_period, m_lvl_dn, m_lvl_up);  
  
    if (m_last_signal == SIGNAL_BUY)  
        status += " [BUY]";  
    else if (m_last_signal == SIGNAL_SELL)  
        status += " [SELL]";  
    else  
        status += " [NEUTRAL]";  
  
    return status;  
}  
  
/**  
 * @brief Gets the role of the signal.  
 *  
 * @return ENUM_SIGNAL_ROLE The role.  
 */
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
 */
virtual ENUM_SIGNAL_ROLE GetRole() override
{
    return m_role;
}

/**
 * @brief Gets the timeframe of the signal.
 *
 * @return ENUM_TIMEFRAMES The timeframe.
 */
virtual ENUM_TIMEFRAMES GetTimeframe() const override
{
    return m_timeframe;
}

/**
 * @brief Draws visual indicators on the chart when a signal triggers.
 *
 * @param barTime The time of the bar where the signal occurred.
 * @param signal The signal type (SIGNAL_BUY, SIGNAL_SELL).
 * @param signalIndex The index of the signal slot (0-2) for vertical stacking.
 */
virtual void DrawSignal(datetime barTime, int signal, int signalIndex) override
{
    // Implementation for chart drawing if needed
    // For now, leave as stub or implement basic arrow drawing
}
};

//+-----+
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
//+-----+
//|                               CSignal_MACD.mqh |
//| Copyright 2025, LAWRENCE KOH |
//| lawrancekoh@outlook.com |
//+-----+
#property copyright "Copyright 2025, LAWRENCE KOH"
#property link      "lawrancekoh@outlook.com"
#property version   "1.00"

#include "ISignal.mqh"

/***
 * @brief MACD signal implementation.
 *
 * Generates buy/sell signals based on MACD main line crossing the signal line.
 */
class CSignal_MACD : public ISignal
{
private:
    int m_handle;                      // Indicator handle
    ENUM_TIMEFRAMES m_timeframe;       // Timeframe
    int m_fast_period;                 // Fast EMA period
    int m_slow_period;                 // Slow EMA period
    int m_signal_period;               // Signal line period
    ENUM_APPLIED_PRICE m_applied_price; // Applied price
    bool m_threshold_check;            // Threshold check enabled
    bool m_threshold_check_reverse;    // Reverse threshold logic
    int m_last_signal;                // Last signal for status
    ENUM_SIGNAL_ROLE m_role;           // Signal role (Bias or Entry)

public:
    /**
     * @brief Initializes the MACD signal.
     *
     * @param settings The signal settings.
     * @return bool true if successful, false otherwise.
     */
    virtual bool Init(const CSignalSettings &settings) override
    {
        // Map parameters from settings
        m_timeframe = settings.Timeframe;
        m_fast_period = settings.Params.IntParams[0];
        m_slow_period = settings.Params.IntParams[1];
        m_signal_period = settings.Params.IntParams[2];
        m_applied_price = settings.Params.Price;
        m_threshold_check = settings.Params.BoolParams[0];
        m_threshold_check_reverse = settings.Params.BoolParams[1];
        m_role = settings.Role;

        // Get indicator handle
        m_handle = iMACD(_Symbol, m_timeframe, m_fast_period, m_slow_period, m_signal_period,
m_applied_price);

        // Validate handle
        if (m_handle == INVALID_HANDLE)
        {
            Print("CSignal_MACD: Failed to get MACD handle for ", _Symbol, " timeframe ",
EnumToString(m_timeframe));
            return false;
        }
    }
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
m_last_signal = SIGNAL_NONE;
return true;
}

/**
 * @brief Gets the current trading signal.
 *
 * @return int SIGNAL_BUY, SIGNAL_SELL, or SIGNAL_NONE.
 */
virtual int GetSignal() override
{
    double main_buffer[3];
    double signal_buffer[3];
    double histogram_buffer[3] = {0, 0, 0};

    // Get data from last closed bar (shift 1)
    if (CopyBuffer(m_handle, 0, 1, 2, main_buffer) != 2 ||
        CopyBuffer(m_handle, 1, 1, 2, signal_buffer) != 2)
    {
        return SIGNAL_NONE;
    }

    // Histogram is optional for logging
    CopyBuffer(m_handle, 2, 1, 2, histogram_buffer);

    // Implement crossover logic
    bool buy_cross = main_buffer[0] > signal_buffer[0] && main_buffer[1] <= signal_buffer[1];
    bool sell_cross = main_buffer[0] < signal_buffer[0] && main_buffer[1] >= signal_buffer[1];

    // Apply threshold filter if enabled
    if (m_threshold_check)
    {
        if (m_threshold_check_reverse)
        {
            buy_cross = buy_cross && main_buffer[0] > 0; // Buy cross must be above zero
            sell_cross = sell_cross && main_buffer[0] < 0; // Sell cross must be below zero
        }
        else
        {
            buy_cross = buy_cross && main_buffer[0] < 0; // Buy cross must be below zero
            sell_cross = sell_cross && main_buffer[0] > 0; // Sell cross must be above zero
        }
    }

    // Return signal
    if (buy_cross)
    {
        m_last_signal = SIGNAL_BUY;
        return SIGNAL_BUY;
    }
    if (sell_cross)
    {
        m_last_signal = SIGNAL_SELL;
        return SIGNAL_SELL;
    }

    m_last_signal = SIGNAL_NONE;
    return SIGNAL_NONE;
}
```





UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

```
/***
 * @brief Gets the status string.
 *
 * @return string Status message.
 */
virtual string GetStatus() override
{
    string status = StringFormat("MACD(%d,%d,%d)", m_fast_period, m_slow_period,
m_signal_period);

    if (m_last_signal == SIGNAL_BUY)
        status += " [BUY]";
    else if (m_last_signal == SIGNAL_SELL)
        status += " [SELL]";
    else
        status += " [NEUTRAL]";

    return status;
}

/***
 * @brief Gets the role of the signal.
 *
 * @return ENUM_SIGNAL_ROLE The role.
 */
virtual ENUM_SIGNAL_ROLE GetRole() override
{
    return m_role;
}

/***
 * @brief Gets the timeframe of the signal.
 *
 * @return ENUM_TIMEFRAMES The timeframe.
 */
virtual ENUM_TIMEFRAMES GetTimeframe() const override
{
    return m_timeframe;
}

/***
 * @brief Draws visual indicators on the chart when a signal triggers.
 *
 * @param barTime The time of the bar where the signal occurred.
 * @param signal The signal type (SIGNAL_BUY, SIGNAL_SELL).
 * @param signalIndex The index of the signal slot (0-2) for vertical stacking.
 */
virtual void DrawSignal(datetime barTime, int signal, int signalIndex) override
{
    // Implementation for chart drawing if needed
    // For now, leave as stub or implement basic arrow drawing
}
};

//+-----+
```

