Enhancing High-Frequency Trading Strategies with Hybrid AI Models for Predictive Market Forecasting

Marco Di Maio

Abstract

This paper presents the development and evaluation of a hybrid trading strategy that integrates rule-based logic with AI-driven predictive modeling for high-frequency trading applications. The core framework operates as a trend-following system, using exponential moving average (EMA) crossovers to identify entry points, validated through the Average Directional Index (ADX) and volume-based confirmation. Exit logic is handled via a differential analysis of ADX variations, enabling early exits from weakening trends.

A key innovation is the incorporation of a predictive module based on machine learning (Random Forest, XGBoost) and deep learning (LSTM) models, trained on higher timeframe data to generate directional forecasts. These forecasts are integrated in real time via a TCP socket connection with a MetaTrader 5 Expert Advisor, influencing the trade sizing logic according to agreement or disagreement with technical signals.

The system supports both live trading and backtesting scenarios, allowing performance comparisons across different assets and timeframes. While the strategy is designed to be market-agnostic, extensive testing highlights particular synergies on specific symbols and configurations. Results indicate a notable improvement in profitability and risk-adjusted metrics when AI-based predictions are incorporated.

Keywords: algorithmic trading, expert advisor, machine learning, deep learning, ADX, EMA, multi-timeframe prediction, high-frequency trading, forex, MetaTrader 5

1. Introduction

The growth of algorithmic and high-frequency trading (HFT) has profoundly reshaped financial markets, introducing speed, automation, and data-centric strategies to trading operations. HFT systems today rely heavily on both reactive technical indicators and proactive prediction layers to optimize entry and exit precision (Ansary, 2023).

While pure rule-based systems are interpretable and straightforward to implement, they often struggle to adapt to volatile or shifting market conditions. Conversely, predictive models based on artificial intelligence (AI) can capture hidden patterns in time series and offer directional insights, especially when using historical features or derived indicators (Ballings et al., 2015). However, these models lack transparency and often require stabilization through complementary logic.

As a result, recent research has focused on hybrid approaches that merge technical signals with predictive layers based on machine learning (ML) or deep learning (DL) (Ouf et al., 2025). Long Short-Term Memory (LSTM) networks, in particular, have been successfully used in financial applications for capturing sequential dependencies in price movements (Wu, 2024).

In this work, we propose a hybrid trading framework that combines a carefully designed trend-following strategy with AI-based forecasting to enhance decision-making. The core trading logic, implemented as a MetaTrader 5 Expert Advisor (EA), operates on a single optimized timeframe and is based on the crossover of two exponential moving averages (EMA), filtered by trend strength (via the Average Directional Index, ADX) and confirmed through volume analysis. An original exit mechanism based on a differential analysis of the ADX trend allows early detection of trend weakening and mitigates prolonged exposure during trend reversals.

To assess the viability of the base strategy, a preliminary large-scale backtesting campaign was conducted. The EA was tested on 24 of the most liquid Forex and commodity symbols across lower timeframes (M1 to M15), using six months of historical data. This selection process aimed to identify the most promising symbol-timeframe combinations where the rule-based system could already demonstrate consistent profitability.

Following this selection phase, a predictive AI module was integrated into the EA. The forecasts, generated using models such as Random Forest, XGBoost, and Long Short-Term Memory (LSTM), were trained exclusively on higher timeframes (H1), to offer a broader market perspective while preserving the responsiveness of lower-timeframe trading. Communication between the EA and the AI module is handled via TCP socket in live trading or via CSV file reading in backtesting mode.

The results show that introducing predictive signals into the decision pipeline—especially when aligned with technical signals—can enhance both profitability and stability. This work contributes a practical and extensible example of hybrid AI integration in financial computing, validated through rigorous

testing and real-time deployment architecture.

The paper is organized as follows. Section 2 reviews related approaches in financial prediction and algorithmic trading. Section 3 details the design of the rule-based strategy, the integration of AI models, and the experimental setup. Section 4 discusses the implications of the results, the performance comparison of the AI models, and potential directions for future work. Finally, Section 5 provides a summary of the key findings and concludes the study.

2. Related Work

The intersection of artificial intelligence (AI) and algorithmic trading has garnered extensive interest in recent years. As financial markets grow increasingly complex and data-rich, conventional rule-based trading systems—often constructed from heuristics and technical indicators—face limitations in their ability to adapt dynamically to non-stationary conditions and structural regime changes. These limitations have prompted a shift toward data-driven and hybrid decision-making systems (Ballings et al., 2015; Wu, 2024).

In the realm of financial time-series forecasting, machine learning (ML) techniques such as Random Forest (RF), Gradient Boosting Machines (e.g., XGBoost), and Support Vector Machines (SVMs) have shown promising performance in classification tasks like directional movement prediction (Ansary, 2023; Ouf et al., 2025). These models are often favored for their interpretability and low computational overhead compared to deep learning methods, making them well-suited for latency-sensitive applications.

Deep learning (DL) architectures—particularly Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs)—have also gained traction in recent financial research. Fischer and Krauss (Fischer and Krauss, 2018) applied LSTM networks to S&P 500 data, demonstrating that these models can outperform classical approaches such as Random Forests and logistic regression, especially in capturing non-linear dependencies over time. Ailyn (Ailyn, 2024) further explored deep hybrid models for high-frequency trading (HFT), highlighting the benefits of sequence-aware architectures when combined with technical features and well-engineered time-window inputs.

Another important trend is the use of **multi-timeframe architectures**, where predictive models operate on higher timeframes while execution logic remains on lower timeframes. This design balances *strategic perspective* and *tactical reactivity*, as noted in works on multi-horizon ensemble systems and rolling forecast windows (Wu, 2024). Such an approach helps reduce noise sensitivity while leveraging broader market structure, which is particularly relevant in volatile or thinly traded instruments.

Hybrid systems have emerged as a particularly compelling direction. These integrate interpretable, rule-based frameworks (e.g., moving averages, trend filters) with the adaptive capabilities of ML models. Hybrid EAs often use AI-generated forecasts as *confirmation or veto signals*, rather than sole decision-makers, enhancing robustness without compromising real-time performance. This hybrid paradigm has been applied across asset classes, including Forex, commodities, and indices, and is shown to outperform both purely manual and purely AI-driven approaches in multiple studies (Ansary, 2023; Ailyn, 2024).

This work contributes to the hybrid AI-for-trading literature by implementing a rule-based Expert Advisor (EA) enhanced with higher-timeframe XGBoost predictions. In contrast to end-to-end AI strategies, our architecture uses ML as a macro-filter to validate short-term entry signals. The EA combines trend-following logic based on EMA crossovers, volume confirmation, and ADX-based exit conditions, and integrates ML forecasts to modulate position sizing. This design not only improves interpretability and stability, but also allows decoupling of predictive and execution layers—an architecture rarely studied in prior empirical work.

3. Methodology

This section outlines the step-by-step pipeline adopted to design, implement, and evaluate a hybrid trading system that combines rule-based logic with machine learning forecasts. The methodological workflow consists of seven main stages:

First, a deterministic Expert Advisor (EA) was developed and benchmarked across multiple symbols and timeframes to identify a profitable baseline strategy. Through this preliminary testing, the most robust configuration was found to be the **XAGUSD** symbol on the **M12** timeframe, which was selected for all subsequent experiments.

Second, an initial backtesting phase (pre-optimization) was conducted using the default technical parameters (`EMA 7/21` and `ADX threshold 30`). This allowed for a baseline evaluation of the AI-enhanced system across three different models: **Random Forest**, **XGBoost**, and **LSTM**.

Third, a full optimization process was carried out to improve the technical components of the strategy. Specifically, the EMA crossover periods were shortened and the ADX threshold was lowered to increase responsiveness to market conditions. This led to an optimized configuration with `EMA 6/24` and `ADX threshold 22`.

Fourth, the AI-enhanced system was re-evaluated through post-optimization backtesting using the updated configuration, to assess how the strategy improvements affected model performance in terms of profitability, consistency, and drawdown.

Fifth, AI-generated forecasts were integrated into the EA logic as a *lot size modulation mechanism*: when the strategy and AI agree, the system increases position size; when they disagree, it reduces exposure. This preserves control within the rule-based logic while leveraging the AI's predictive signal for risk-adjusted refinement.

Finally, a real-time communication mechanism was implemented via **TCP/IP sockets** to allow the EA to send live feature data to a Python-based microservice and receive predictions in real time. During backtests, this was replaced by reading from a pre-generated CSV containing historical model predictions, allowing full integration with MetaTrader's testing environment.

3.1. Baseline Strategy Design and Evaluation

The foundational trading logic implemented in the Expert Advisor (EA) follows a deterministic trend-following approach with volume confirmation and algorithmic exit conditions. No machine learning components were active during this initial phase. The system was designed to be robust, rule-based, and interpretable, serving as a performance baseline before introducing predictive models.

3.1.1. Technical Logic and Entry Rules

The EA opens positions based on the crossover of two Exponential Moving Averages (EMAs), evaluated in combination with the Average Directional Index (ADX) and volume trends. Specifically:

- **Trend Signal:** EMA(7) crossing EMA(21);
- **Trend Strength:** ADX (period 14) must exceed 30;
- **Volume Confirmation:** Current volume must be greater than the previous bar's volume.

A long position is opened when the fast EMA crosses above the slow EMA, with strong ADX and increasing volume. A symmetric condition applies for short entries. Stop Loss (SL) and Take Profit (TP) levels are fixed at 100 and 300 points, respectively, enforcing a 1:3 risk-reward ratio.

3.1.2. Exit Logic: Differential ADX Algorithm

The exit strategy is based on a trend deterioration detection mechanism:

- **Forced Exit:** If 3 out of the last 5 first-order ADX differences are negative, the system exits the position, even if ADX >30;
- **Prudent Exit:** If ADX drops below 30 but remains above 25 and volume does not increase, the system exits cautiously;
- **Computation Scope:** ADX first-order differences are calculated over the last 6 bars within the same timeframe as the Expert Advisor, ensuring temporal coherence in trend weakening detection.

This differential logic helps anticipate trend weakening, reducing the probability of holding through reversals or ranging periods.

3.1.3. Backtesting Campaign Setup

The EA was subjected to a large-scale backtesting campaign across a diverse set of 24 highly liquid symbols, including major currency pairs, commodities, indices, and cryptocurrencies:
`EURUSD, GBPUSD, USDJPY, XAUUSD.cy, US30, US100, USDCHF, USDCAD, AUDUSD, NZDUSD, EURJPY, GBPJPY, EURNZD, GBPAUD, AURCAD, AUDNZD, US500, UK100, XAGUSD.cy, WTI, BRENT, BTCUSD, ETHUSD, GER40`

Each symbol was evaluated independently on multiple granular timeframes (M1, M2, M3, M4, M5, M6, M10, M12, M15), using historical data from **January to June, 2025**.

To automate large-scale testing, the MetaTrader 5 Genetic Optimization engine was used to run simulations in parallel. Although the underlying parameters of the strategy were fixed, the optimization allowed efficient execution and comparison of results across assets and timeframes.

3.1.4. Evaluation Metrics and Selection Criteria

For each symbol-timeframe pair, the following metrics were collected:

- Net Profit
- Number of Trades
- Profit Factor
- Expected Payoff
- Maximum Drawdown (absolute and percentage)
- Recovery Factor
- Sharpe Ratio

The selection of the final configuration was based on a trade-off between profitability and operational stability. The goal was to identify a setup that produced:

- A substantial number of trades (to ensure statistical relevance);
- A high profit factor and Sharpe Ratio;
- Acceptable drawdown levels relative to returns.

3.2. Selection of Target Symbol and Timeframe

Following an extensive backtesting phase involving 24 symbols across granular timeframes from M1 to M15, the configuration that consistently demonstrated superior performance was identified as **XAGUSD.cy on the M12 timeframe**.

This selection was based on multiple criteria:

- **Net profitability**: Total net profit reached **610.45 USD** over the six-month test period (January–June 2025), with a gross profit of 1,359.45 USD and a gross loss of -749.00 USD;
- **Risk-adjusted returns**: A solid *Profit Factor* of 1.82 and a *Sharpe Ratio* of 10.20 indicated favorable return-to-risk characteristics;
- **Trade volume and stability**: 34 trades were executed with a win rate of 61.76%, and the system showed stable performance across various times of the day, week, and month;
- **Drawdown control**: Maximum relative drawdown remained low at 2.64% in balance and 3.29% in equity.

Table 1: Performance summary of the baseline strategy on **XAGUSD.cy @ M12**.

Metric	Value	Notes
Net Profit	610.45 USD	Starting balance: 10,000 USD
Profit Factor	1.82	Ratio of gross profit to gross loss
Sharpe Ratio	10.20	Indicates risk-adjusted return
Max Drawdown (Balance)	287.50 USD (2.64%)	Absolute and relative
Max Drawdown (Equity)	360.50 USD (3.29%)	Absolute and relative
Number of Trades	68	34 completed cycles
Win Rate	61.76%	21 winning vs 13 losing trades
Best Trade	148.50 USD	Max single gain
Worst Trade	-65.00 USD	Max single loss

This configuration, as summarized in Table 1, served as the benchmark scenario for the integration of predictive AI models in the subsequent methodology phase.

The following visual reports, exported from the MQL5 Strategy Tester, provide insights into temporal behavior, entry distributions, and performance metrics:

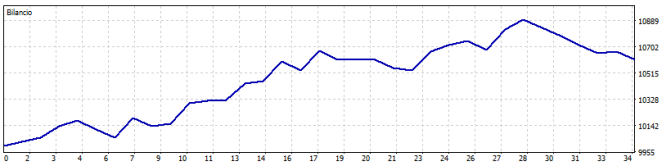

Figure 1: Equity curve over the testing period.

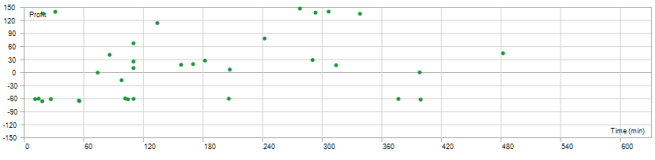

Figure 2: Holding time vs. profitability.

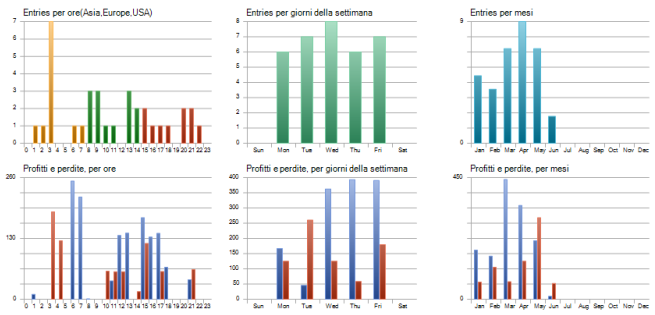

Figure 3: Entries and profitability by time and date.

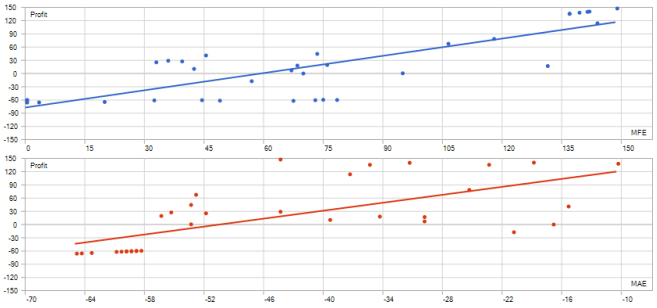

Figure 4: Profit correlation with MFE and MAE.

This thorough evaluation provided a robust and stable baseline strategy against which future enhancements via AI-based trend forecasting (described in the following subsections) could be measured.

3.3. Data Extraction and Feature Engineering

To train AI models capable of short-term directional forecasting, a dedicated MQL5 extraction pipeline was implemented for the selected instrument **XAGUSD.cy**. The script operated on the **H1 timeframe** and retrieved all available historical data up to a predefined upper limit of 100,000 bars. In practice, the extracted dataset contained **30,108 hourly bars**, covering the entire available period from **May 19, 2020, to June 20, 2025**.

Script-Based Data Collection

A standalone MQL5 script was developed to automate the export process. For each hourly bar, the following attributes were computed and saved to the output file `XAG_H1_exp.csv`:

- `Time`: Bar timestamp
- `Open, High, Low, Close`: OHLC prices
- `Volume`: Tick volume
- `EMA(7), EMA(21)`: Fast and slow exponential moving averages
- `ADX(14)`: Average Directional Index for trend strength

The indicators were computed using MQL5 native functions (`iMA`, `iADX`) and populated via `CopyBuffer()`. The resulting CSV was formatted with ANSI encoding and ordered chronologically from most recent to oldest. No real-time updates were included—the dataset was static and intended for offline processing.

Feature Vector Structure

The exported dataset consists of **30,108 data points**, each with 9 attributes. The structure of each row is shown in Table 2, and includes both raw price inputs and technical indicators.

No transformations, windowing, or labeling were performed at this stage. The file serves as the raw input to subsequent modeling or preprocessing steps.

Table 2: Excerpt of exported dataset (`XAG_H1_exp.csv`).

Time	Open	High	Low	Close	Volume	EMA_Fast	EMA_Slow	ADX
2025.06.20 23:00	36.008	36.032	35.977	36.008	1352	36.012	36.053	20.03
2025.06.20 22:00	36.039	36.078	35.996	36.008	2549	36.013	36.058	22.08
2025.06.20 21:00	36.057	36.072	36.010	36.038	1988	36.015	36.063	23.86

3.4. Model Training and Evaluation

This section introduces the machine learning models adopted to enhance the baseline strategy through predictive analytics. The primary objective was to develop models capable of learning from recent market structure, using engineered features derived from historical price action and technical indicators.

Three families of models were selected for evaluation, each representing a distinct approach to forecasting:

- **Random Forest (RF)**: a robust ensemble method based on decision trees;
- **XGBoost (XGB)**: a gradient boosting algorithm optimized for tabular data;
- **Long Short-Term Memory (LSTM)**: a recurrent neural network architecture well-suited for time-series input.

All models were trained using the dataset previously exported from MQL5 for the **XAGUSD.cy** symbol on the H1 timeframe (see Section 3.3). The input data consisted of 30,108 hourly bars, each including 9 features: Open, High, Low, Close, Volume, EMA(7), EMA(21), ADX(14), and Timestamp. At this stage, no labels or target variables were introduced, as labeling was handled in the subsequent modeling-specific pipelines.

A unified Python environment was used for model development, leveraging standard libraries such as `pandas`, `scikit-learn`, `xgboost`, and `TensorFlow/Keras`. Each model pipeline included tailored preprocessing, scaling, and validation procedures as appropriate for its architecture.

The following subsections describe the training workflow, feature handling, and configuration adopted for each model.

Exploratory Analysis and Visualization

A preliminary exploratory analysis was performed to assess data structure, distribution, and feature interactions. This analysis was shared across all model pipelines.

The correlation heatmap in Figure 5 confirms strong positive relationships among OHLC features and between the Close price and both EMA values. Volume and ADX displayed more independent behavior, suggesting potentially useful variation.

In addition, a time series plot of the Close price overlaid with EMA(7) and EMA(21) was produced to visualize the interaction between price and trend-following indicators. This is shown in Figure 6.

This initial analysis helped confirm that the extracted indicators and raw market data were suitable inputs for learning algorithms targeting short-term price behavior.

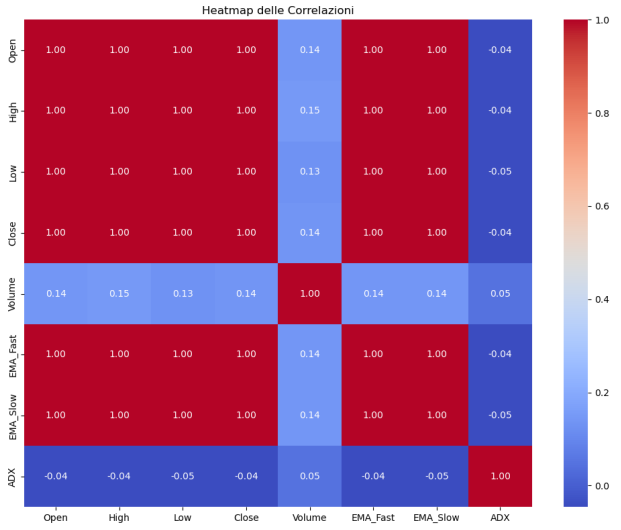

Figure 5: Correlation heatmap of all numeric features extracted from H1 historical data.

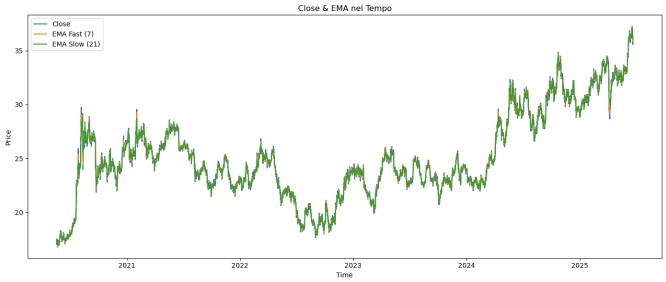

Figure 6: Close price with EMA(7) and EMA(21) overlays on XAGUSD.cy (H1 timeframe).

3.4.1. Random Forest (RF)

The first predictive model evaluated was a **Random Forest (RF)** classifier, chosen for its robustness, low variance, and interpretability. The model aimed to anticipate short-term upward movements in the XAGUSD.cy price series by analyzing engineered features extracted from historical hourly data.

Label Construction.. A binary classification target was defined to indicate whether the Close price, shifted by $n = 3$ future bars, exceeded the current Close by more than a fixed threshold ($\delta = 0.0003$). Formally, the target variable was set as:

$$\texttt{target} = \begin{cases} 1 & \text{if } \texttt{Close}_{t+3} > \texttt{Close}_t + \delta \\ 0 & \text{otherwise} \end{cases}$$

This approach yielded a balanced class distribution: 11,649 zeros and 10,930 ones in the training set, and 3,883 zeros vs. 3,644 ones in the test set.

Feature Engineering.. In addition to the original features (OHLC, Volume, EMA, ADX), several derived variables were computed to better capture price dynamics and volatility:

- `EMA_diff` = EMA(7) - EMA(21)

- `Close_pct_change`: percentage change in Close
- `Volume_pct_change`: percentage change in Volume
- `ADX_delta`, `Volume_delta`: first differences
- `ADX_roll_std3`, `Volume_roll_std3`: 3-bar rolling standard deviations

After removing NaNs caused by rolling computations, the resulting dataset contained 30,106 rows and 17 columns.

Preprocessing and Scaling.. All numerical features were standardized using `StandardScaler`, which was serialized to disk (`scaler_randomforest.save`) to ensure consistency during deployment or future prediction stages. The processed dataset was also exported to CSV for reproducibility.

Training Procedure.. The model was trained using `RandomForestClassifier` from `scikit-learn`, with the following configuration:

- `n_estimators` = 100
- `random_state` = 42
- `n_jobs` = -1 (parallel execution)

The data was split into training and test sets with a 75/25 ratio using stratified sampling to preserve class balance.

Performance Results.. The trained model achieved an accuracy of **70%** on the test set, with relatively balanced precision and recall across both classes. Full evaluation metrics are reported below:

Table 3: Classification report — Random Forest on test set

Class	Precision	Recall	F1-score	Support
0 (No Uptrend)	0.69	0.74	0.71	3,883
1 (Uptrend)	0.70	0.65	0.67	3,644
Accuracy		**0.70**		

The confusion matrix is provided in Figure 7, illustrating the distribution of correct and incorrect predictions.

Model Serialization.. The trained model was saved as `random_forest_trend_model.pkl`, ready for use in real-time prediction or batch forecasting environments.

3.4.2. XGBoost (XGB)

XGBoost (Extreme Gradient Boosting) was chosen as a high-performance gradient-boosted tree algorithm, particularly effective on structured tabular data. The model was trained using the same feature-engineered dataset constructed from the historical H1 series of **XAGUSD.cy**.

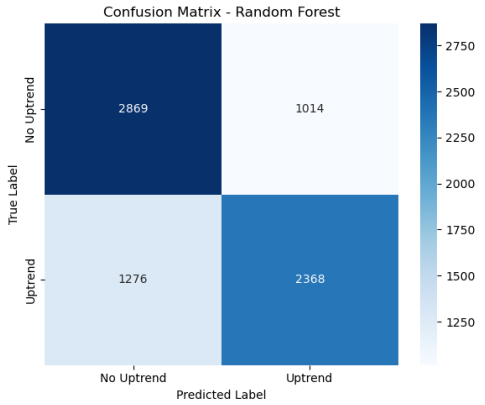

Figure 7: Confusion matrix — Random Forest predictions on the test set.

Label Construction.. As with the Random Forest model, the prediction task was framed as a binary classification problem. The target variable was set to `1` if the closing price shifted by $t + 3$ exceeded the current Close price plus a threshold of $\delta = 0.0003$, and `0` otherwise. Formally, the target variable was defined as:

$$\texttt{target} = \begin{cases} 1 & \text{if } \texttt{Close}_{t+3} > \texttt{Close}_t + \delta \\ 0 & \text{otherwise} \end{cases}$$

This approach resulted in a balanced class distribution: 11,649 zeros and 10,930 ones in the training set, and 3,883 zeros vs. 3,644 ones in the test set.

Feature Engineering.. In addition to the original features (OHLC, Volume, EMA, ADX), the following derived variables were computed to enhance the model's ability to capture price trends and volatility:

- `EMA_diff` = EMA(7) - EMA(21)
- `Close_pct_change`: percentage change in Close
- `Volume_pct_change`: percentage change in Volume
- `ADX_delta`, `Volume_delta`: first differences
- `ADX_roll_std3`, `Volume_roll_std3`: 3-bar rolling standard deviations

After removing NaNs caused by rolling computations, the final dataset consisted of **30,106 samples**, each with **17 features**.

Preprocessing and Scaling.. As with Random Forest, all numerical features were standardized using `StandardScaler`. The scaler object was serialized to disk as `scaler_xgboost.save` to ensure consistency during future inference stages. The processed dataset was also saved to a CSV file for reproducibility.

Model Configuration and Class Imbalance Handling.. To address slight class imbalance in the dataset, the `scale_pos_weight` parameter was dynamically computed as the ratio of negative to positive training samples. This yielded a value of approximately `1.07`, which was used to penalize the model for misclassifying the minority class. The model was configured with the following parameters:

- `n_estimators = 100`
- `eval_metric = 'logloss'`
- `random_state = 42`
- `scale_pos_weight = 1.07`

The model was trained using `XGBClassifier` from the `xgboost` library, and the trained model was saved as `xgboost_trend_model.pkl` for future use.

Performance Evaluation.. XGBoost achieved a test set accuracy of **73.77%**, outperforming Random Forest slightly. The confusion matrix and detailed classification report are provided below.

Table 4: Classification Report — XGBoost on Test Set

Class	Precision	Recall	F1-score	Support
0 (No Uptrend)	0.7470	0.7435	0.7452	3,883
1 (Uptrend)	0.7280	0.7316	0.7298	3,644
Accuracy		**0.7377**		

The confusion matrix is illustrated in Figure 8, which shows the distribution of correct and incorrect predictions.

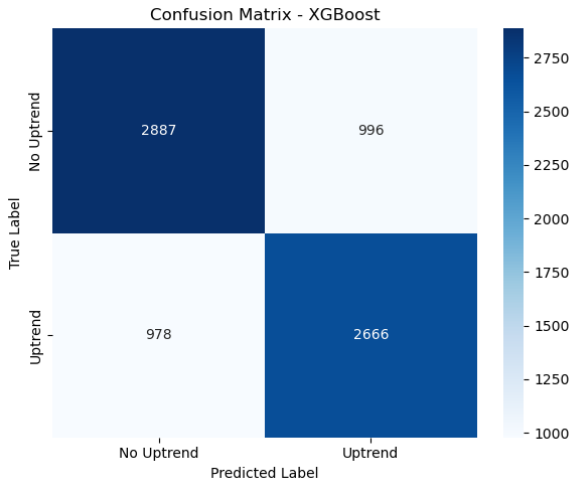

Figure 8: Confusion matrix — XGBoost predictions on the test set.

Model Serialization.. The trained XGBoost model was serialized and saved as `xgboost_trend_model.pkl`, ready for deployment in real-time prediction or batch forecasting environments.

3.4.3. Long Short-Term Memory (LSTM)

The third model evaluated was a **Long Short-Term Memory (LSTM)** network, a type of recurrent neural network (RNN) designed to capture temporal dependencies in sequential data. LSTM is particularly effective for time series forecasting, as it is capable of learning long-range patterns and relationships.

Label Construction.. Similar to the other models, the prediction task was framed as a binary classification problem. The target variable was defined as:

$$\texttt{trend} = \begin{cases} 1 & \text{if } \texttt{Close}_{t+1} > \texttt{Close}_t + \delta \\ -1 & \text{if } \texttt{Close}_{t+1} < \texttt{Close}_t - \delta \\ 0 & \text{otherwise} \end{cases}$$

This labeling approach ensured that both upward and downward trends were represented, and the distribution of the target variable was balanced, with 50.73% of samples labeled as -1 and 49.27% labeled as 1.

Feature Engineering.. In addition to the raw features (OHLC, Volume, EMA, ADX), several derived features were computed to capture price dynamics, volatility, and market activity:

- `EMA_diff` = EMA(7) - EMA(21)
- `Close_pct_change`: percentage change in Close price
- `Volume_pct_change`: percentage change in trading volume
- `ADX_delta`, `Volume_delta`: first differences of ADX and Volume
- `ADX_roll_std3`, `Volume_roll_std3`: 3-bar rolling standard deviations for ADX and Volume

After handling NaNs caused by rolling calculations, the final dataset contained **30,106 samples** and **17 features**, with the target variable being the only categorical column.

Preprocessing and Scaling.. All numerical features were standardized using `StandardScaler`, and the scaler was saved as `scaler_lstm.save` for future inference consistency. The data was then split into training, validation, and test sets. The sequence length was set to 20, ensuring that each input sample represented a window of 20 consecutive bars.

Model Configuration.. The LSTM model was constructed using the Keras framework with the following architecture:

- The first LSTM layer had 64 units with `return_sequences=True` to retain the full sequence for the next LSTM layer.
- A Dropout layer with a rate of 0.2 was added to prevent overfitting.
- The second LSTM layer had 32 units.
- A second Dropout layer with a rate of 0.2 was added.

- The final layers included a Dense layer with 32 units and a ReLU activation function, followed by a Dense layer with 1 unit and a `tanh` activation function, to output values between -1 and 1.

The model was compiled using the `adam` optimizer, the `hinge` loss function, and accuracy as the evaluation metric. Early stopping was employed to prevent overfitting by monitoring the validation loss and restoring the best weights.

Training Procedure.. The model was trained for up to 50 epochs, with a batch size of 64. The training process was monitored using validation data, and the training stopped early if the validation loss did not improve for 5 consecutive epochs. The model achieved a test set accuracy of **62.23%**.

Performance Evaluation.. The trained LSTM model achieved an accuracy of **62.23%** on the test set. The confusion matrix and detailed classification report are shown below:

Table 5: Classification Report — LSTM on Test Set

Class	Precision	Recall	F1-score	Support
-1.0	0.6242	0.6423	0.6331	2273
1.0	0.6203	0.6017	0.6109	2207
Accuracy		**0.6223**		

The confusion matrix is provided in Figure 9, which shows the distribution of correct and incorrect predictions.

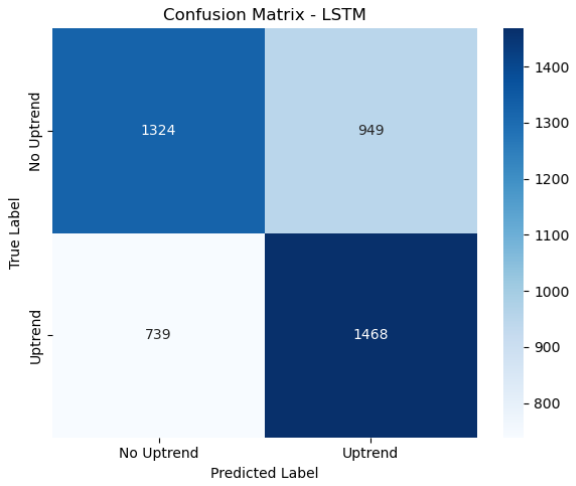

Figure 9: Confusion matrix — LSTM predictions on the test set.

Model Serialization.. The trained LSTM model was saved as `lstm_trend_model.h5`, ready for use in real-time prediction or batch forecasting environments.

3.5. Forecast Generation and Integration in EA

In this section, we describe the process of generating forecast files from the AI models (Random Forest, XGBoost, and LSTM) and their subsequent integration into the Expert Advisor (EA) for backtesting. The forecasts are used as confirmation or negation of trading signals provided by the baseline indicators, influencing the lot size for each trade.

Forecast File Generation

For each of the three models, the forecast file is generated in a consistent manner, with the following general steps:

- **Loading the Data:** The preprocessed dataset, which contains the engineered features and trend labels from historical data, is loaded from a CSV file. The filenames for each model are:
  - `randomforest_ready_dataset.csv` for the Random Forest model,
  - `xgboost_ready_dataset.csv` for the XGBoost model,
  - `lstm_ready_dataset.csv` for the LSTM model.
- **Loading the Scaler and Model:** The scaler and the trained model are loaded from disk. The scaler is used to standardize the features, ensuring they match the preprocessing used during training.
- **Preprocessing and Scaling:** The necessary features (e.g., Open, High, Low, Close, Volume, EMA, ADX) are scaled using the scaler that was saved during model training. This ensures consistent feature scaling for prediction.
- **Prediction:** The model makes predictions for each time step in the dataset. The output is a binary prediction (1 for an uptrend, -1 for a downtrend) indicating the expected market movement in the next time interval (H1).
- **Saving the Predictions:** The predictions and their corresponding timestamps are saved in a CSV file in the format required by the Expert Advisor (EA). The generated CSV files are named as follows:
  - `trend_forecast_RF.csv` for the Random Forest model,
  - `trend_forecast_XGBoost.csv` for the XGBoost model,
  - `trend_forecast_lstm.csv` for the LSTM model.

  These files contain two columns: `time` and `trend`, where the `trend` column indicates the predicted market trend.

The forecast files are stored in the MetaTrader 5 file directory (`...\MetaQuotes\Terminal\Common\Files`) to be read by the EA during backtesting.

Integration with the Expert Advisor (EA)

During backtesting, the Expert Advisor (EA) retrieves the forecast file for the respective model (e.g., XGBoost) and reads the trend prediction for the relevant time step. The key point is that the EA does not make trading decisions solely based on the forecast. Instead, the AI-generated prediction serves as a supplementary confirmation or negation of the trade signals generated by the baseline indicators (EMA, ADX).

- **Backtest Setup:** When the EA detects a potential trading opportunity based on its baseline indicators (EMA crossover, ADX threshold, etc.), it checks the prediction from the relevant forecast file.
- **Trade Confirmation or Negation:** If the AI model's prediction agrees with the signal from the baseline indicators (i.e., both signal a potential uptrend or downtrend), the EA will enter the trade with an increased lot size. If the model's prediction contradicts the baseline indicators, the EA may decrease the lot size.
- **Lot Size Adjustment:** The lot size for each trade is adjusted based on the AI model's prediction. For example, if the prediction indicates a strong uptrend and the baseline indicators also suggest an uptrend, the EA might increase the lot size to maximize profit potential. Conversely, if there is a disagreement between the two, the EA reduces the lot size to minimize risk.

This integration allows the EA to leverage the predictive power of AI models to enhance decision-making during backtesting and real-time trading. The AI models provide valuable supplementary signals, which, when combined with the baseline strategy's indicators, improve trade accuracy and risk management.

In the following sections, we will analyze the backtest results to evaluate how the integration of AI-driven forecasts—using Random Forest, XGBoost, and LSTM models—affects the performance of the baseline strategy, comparing results with and without the use of AI predictions.

3.6. Backtest Settings – Pre-Optimization Baseline

Before conducting the final optimization steps described in the following section, an initial round of backtesting was performed using standard indicator parameters, in order to establish a consistent benchmark across all AI models. These settings represent the baseline version of the strategy, without custom tuning of EMA or ADX components, and they were used uniformly across Random Forest, XGBoost, and LSTM models.

The goal of this baseline backtest was to evaluate the relative impact of AI model predictions on a common, fixed strategy configuration. Later, this will be compared with the results obtained after the strategy optimization phase.

All three AI models—Random Forest, XGBoost, and LSTM—were tested using the same set of parameters to ensure a fair comparison.

The backtest was carried out on the **XAGUSD.cy** symbol using the **M12** timeframe from January 2025 to June 2025 for all three AI models (Random Forest, XGBoost, and LSTM). The expert advisor (EA) used the pre-trained models to predict market trends, with the predictions stored in CSV files (generated by the model) and loaded during the backtesting process.

The backtest for all models was configured with the following parameters:

- EMA Fast Period: 7
- EMA Slow Period: 21
- ADX Period: 14
- ADX Threshold: 30.0
- ADX Exit Threshold: 25.0
- Stop Loss: 100.0 pips
- Take Profit: 300.0 pips
- Lot Size: 0.1
- Lot Multiplier Agreement: 1.2
- Lot Multiplier Disagreement: 0.5

With these common settings established, the specific backtest results for each AI model are analyzed in the following subsections.

3.6.1. Random Forest (RF) Backtest Results

Key Results:. During the backtesting period, the expert advisor was able to achieve a total net profit of **660.89 USD**, with a gross profit of **1,300.34 USD** and a gross loss of **-639.45 USD**. The highest drawdown reached **310.70 USD** (2.83%), and the maximum equity drawdown was **398.30 USD** (3.60%).

Other key metrics include:

- **Profit Factor**: 2.03
- **Expected Payoff**: 19.44 USD
- **Recovery Factor**: 1.66
- **Sharpe Ratio**: 11.46
- **Number of Trades**: 34
- **Winning Trades**: 21 (61.76%)
- **Losing Trades**: 13 (38.24%)
- **Max Consecutive Wins**: 7 (444.30 USD)
- **Max Consecutive Losses**: 4 (-245.65 USD)

Graphical Analysis:. The following figures provide a comprehensive overview of the backtest results:

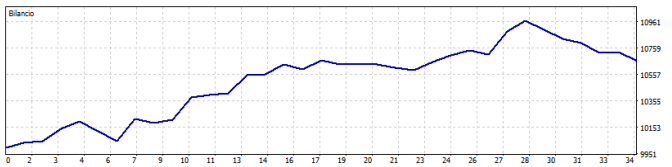

Figure 10: Equity curve for the Random Forest backtest on XAGUSD.cy (M12).

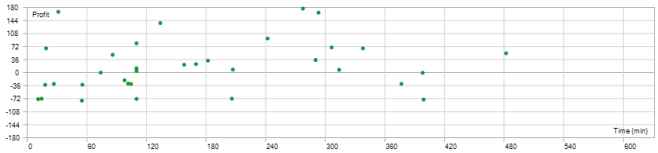

Figure 11: Profit vs Holding time during Random Forest backtest on XAGUSD.cy.

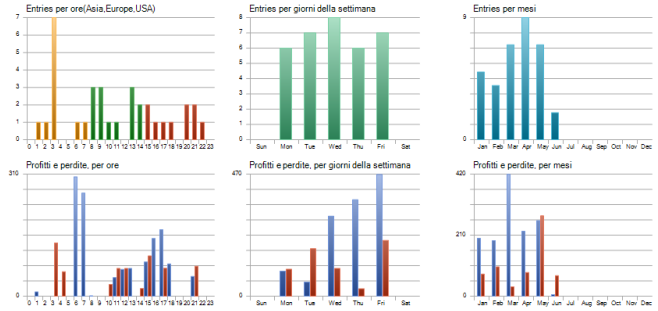

Figure 12: Entries and profitability by time and date for Random Forest backtest.

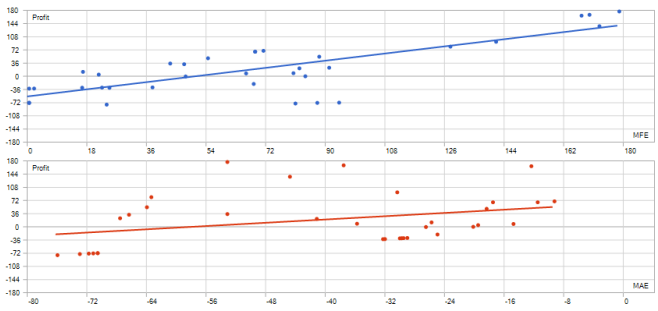

Figure 13: Profit correlation with MFE and MAE during Random Forest backtest.

Conclusion:. In conclusion, the Random Forest model demonstrated strong performance with a high Profit Factor of 2.03 and an impressive Sharpe Ratio of 11.46, indicating both profitability and effective risk management. These results show that the Random Forest model significantly enhanced the trading strategy, providing a solid foundation for AI-driven trend forecasting in real-world trading environments.

3.6.2. XGBoost Backtest Results

Key Results:. During the backtesting period, the expert advisor was able to achieve a total net profit of **683.99 USD**, with a gross profit of **1,411.99 USD** and a gross loss of **-728.00 USD**. The highest drawdown reached **310.70 USD** (2.83%), and the maximum equity drawdown was **398.30 USD** (3.59%).

Other key metrics include:

- **Profit Factor**: 1.94
- **Expected Payoff**: 20.12 USD
- **Recovery Factor**: 1.72
- **Sharpe Ratio**: 11.25
- **Number of Trades**: 34
- **Winning Trades**: 21 (61.76%)
- **Losing Trades**: 13 (38.24%)
- **Max Consecutive Wins**: 7 (555.95 USD)
- **Max Consecutive Losses**: 4 (-245.65 USD)

Graphical Analysis:. The following figures provide a comprehensive overview of the backtest results:

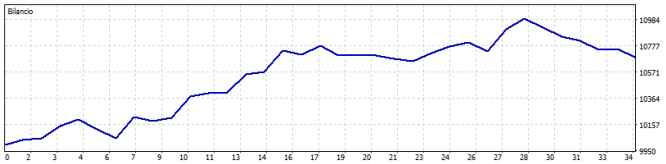

Figure 14: Equity curve for the XGBoost backtest on XAGUSD.cy (M12).

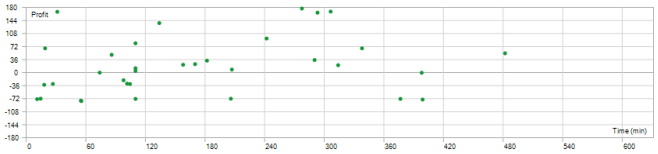

Figure 15: Profit vs Holding time during XGBoost backtest on XAGUSD.cy.

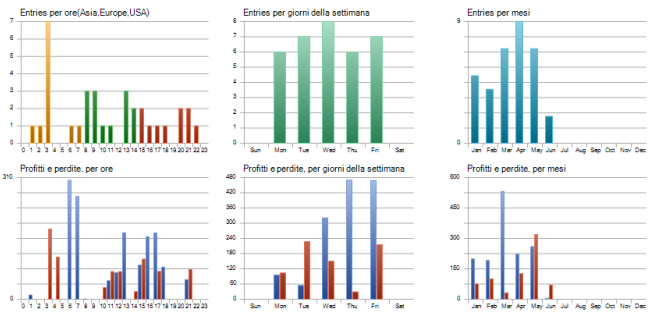

Figure 16: Entries and profitability by time and date for XGBoost backtest.

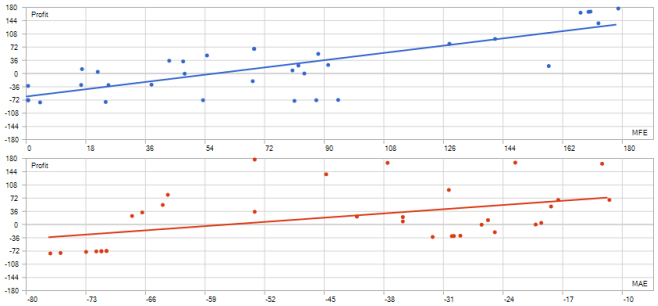

Figure 17: Profit correlation with MFE and MAE during XGBoost backtest.

Conclusion:. The XGBoost model achieved notable results, with a Profit Factor of 1.94 and a Sharpe Ratio of 11.25, demonstrating consistent profitability and effective risk-adjusted returns. These outcomes confirm that XGBoost can provide valuable insights for enhancing trading strategies, improving both decision-making and risk control in real-time trading.

3.6.3. LSTM Backtest Results

Key Results:. During the backtesting period, the expert advisor was able to achieve a total net profit of **153.44 USD**, with a gross profit of **752.99 USD** and a gross loss of **-599.55 USD**. The highest drawdown reached **270.10 USD** (2.59%), and the maximum equity drawdown was **306.60 USD** (2.93%).

Other key metrics include:

- **Profit Factor**: 1.26
- **Expected Payoff**: 4.51 USD
- **Recovery Factor**: 0.50
- **Sharpe Ratio**: 3.32
- **Number of Trades**: 34
- **Winning Trades**: 21 (61.76%)
- **Losing Trades**: 13 (38.24%)
- **Max Consecutive Wins**: 7 (231.65 USD)
- **Max Consecutive Losses**: 4 (-205.05 USD)

Graphical Analysis:. The following figures provide a comprehensive overview of the backtest results:

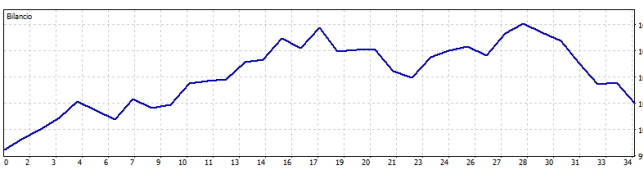

Figure 18: Equity curve for the LSTM backtest on XAGUSD.cy (M12).

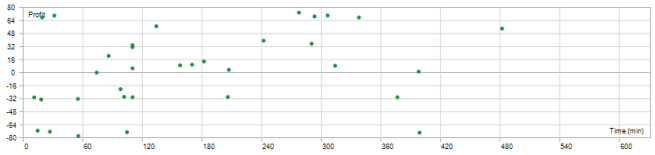

Figure 19: Profit vs Holding time during LSTM backtest on XAGUSD.cy.

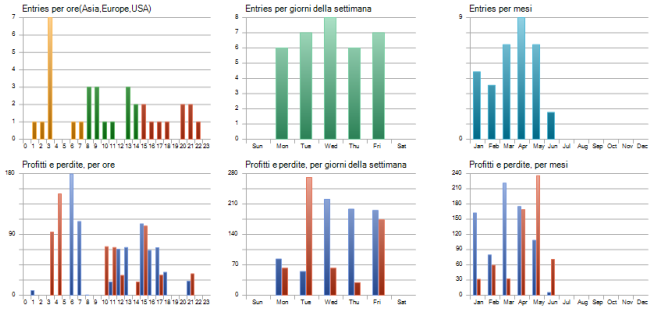

Figure 20: Entries and profitability by time and date for LSTM backtest.

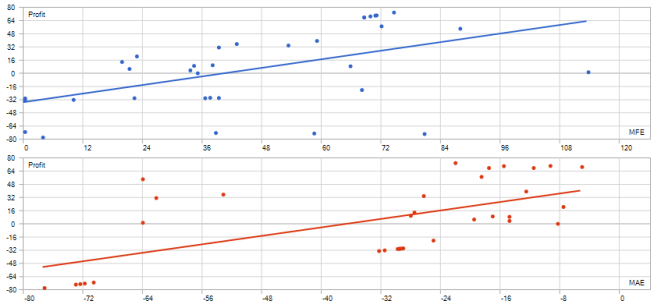

Figure 21: Profit correlation with MFE and MAE during LSTM backtest.

Conclusion:. The LSTM model showed decent performance with a Profit Factor of 1.26 and a Sharpe Ratio of 3.32, indicating a moderate but positive impact on profitability. Despite these factors, the LSTM model still contributed positively to the overall trading strategy and holds promise for future performance optimization in AI-driven trend forecasting.

3.7. Summary of Key Backtest Results

In this section, we summarize the key backtest results for all three AI models: Random Forest (RF), XGBoost, and LSTM. The table below presents a comparison of the net profit, maximum drawdown, profit factor, and Sharpe ratio for each model.

Table 6: Summary of Key Backtest Results

Model	Net Profit (USD)	Max Drawdown (USD)	Profit Factor	Sharpe Ratio
Random Forest	660.89	310.70	2.03	11.46
XGBoost	683.99	310.70	1.94	11.25
LSTM	153.44	270.10	1.26	3.32

This table provides a high-level comparison of the results from the backtests of the three AI models. It highlights the varying performance levels across the models, with XGBoost

achieving the highest net profit (683.99 USD), while Random Forest achieved the highest Profit Factor (2.03) and Sharpe Ratio (11.46). This means that, although XGBoost made the highest profit, Random Forest performed better in terms of risk management, delivering more favorable returns per unit of risk taken. The LSTM model, while showing positive results, had a more modest impact on profitability compared to the other models.

3.8. Strategy Optimization and Parameter Tuning

During the initial backtesting phase (see Section 3.6), the equity curves of the Expert Advisor exhibited irregular stair-like patterns and prolonged plateaus across all tested models (baseline, Random Forest, XGBoost, and LSTM). These artifacts suggested potential inefficiencies in the underlying entry/exit logic of the base strategy.

To improve signal responsiveness and reduce latency, a first round of optimization focused on the exponential moving average (EMA) parameters responsible for trend detection. The goal was to refine crossover accuracy and improve trade timing.

EMA Parameter Grid Search

A full grid search was performed using the MetaTrader 5 Strategy Tester with the following parameter ranges:

- **EMA Fast Period**: 3 to 15 (step = 1)
- **EMA Slow Period**: 10 to 30 (step = 1)

Each combination was evaluated on the non-AI version of the strategy, using metrics such as total net profit, maximum drawdown, and profit factor. The optimal configuration identified was:

- **Fast EMA:** 6
- **Slow EMA:** 24

These values were subsequently adopted across all models in further testing phases.

Backtest Results without AI

- **Total Net Profit:** 636.20 USD
- **Max Drawdown (Balance):** 294.50 USD (2.70%)
- **Profit Factor:** 1.83
- **Total Trades:** 35
- **Winning Trades:** 60.00%
- **Max Consecutive Wins:** 7 (483.85 USD)
- **Sharpe Ratio:** 11.50

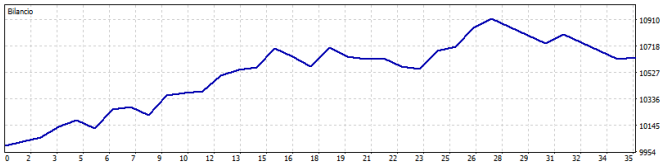

Figure 22: Equity curve — Baseline strategy with optimized EMA parameters.

Backtest Results with Random Forest

- **Total Net Profit:** 659.47 USD
- **Max Drawdown (Balance):** 313.50 USD (2.86%)
- **Profit Factor:** 2.01
- **Total Trades:** 35
- **Winning Trades:** 60.00%
- **Max Consecutive Wins:** 7 (468.96 USD)
- **Sharpe Ratio:** 12.00

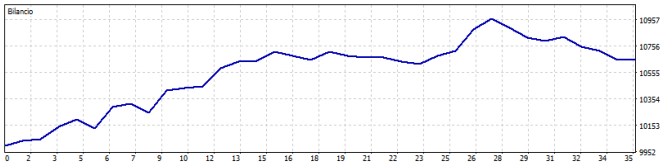

Figure 23: Equity curve — Random Forest strategy with optimized EMA parameters.

Backtest Results with XGBoost

- **Total Net Profit:** 821.87 USD
- **Max Drawdown (Balance):** 269.40 USD (2.43%)
- **Profit Factor:** 2.27
- **Total Trades:** 35
- **Winning Trades:** 60.00%
- **Max Consecutive Wins:** 7 (580.61 USD)
- **Sharpe Ratio:** 14.07

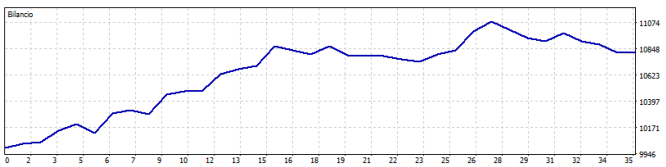

Figure 24: Equity curve — XGBoost strategy with optimized EMA parameters.

Backtest Results with LSTM

- **Total Net Profit:** 74.03 USD
- **Max Drawdown (Balance):** 315.15 USD (3.04%)
- **Profit Factor:** 1.11
- **Total Trades:** 35
- **Winning Trades:** 60.00%
- **Max Consecutive Wins:** 7 (241.92 USD)
- **Sharpe Ratio:** 1.91

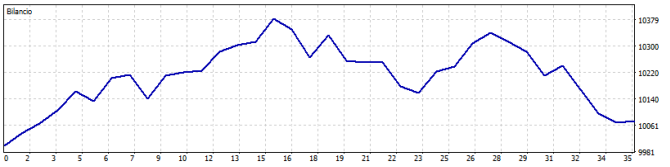

Figure 25: Equity curve — LSTM strategy with optimized EMA parameters.

Summary of EMA Optimization Results

The EMA optimization yielded positive outcomes for the baseline and XGBoost strategies, both surpassing pre-optimization performance. The Random Forest model maintained similar results, while the LSTM model underperformed, though still ending in profit. These findings validate the benefit of preliminary parameter tuning prior to introducing predictive components.

ADX Filter Optimization

Subsequent optimization efforts targeted the Average Directional Index (ADX) threshold, originally set to 30. A range between 20 and 35 was tested to identify the most effective setting for filtering non-trending markets.

Lowering the threshold to 22 resulted in more frequent entries without significantly increasing risk exposure.

Backtest Results without AI Using Optimized ADX Threshold

- **Total Net Profit:** 1,559.29 USD
- **Max Drawdown (Balance):** 647.65 USD (5.97%)
- **Profit Factor:** 1.40
- **Total Trades:** 121
- **Winning Trades:** 47.11%
- **Max Consecutive Wins:** 6 (495.85 USD)
- **Sharpe Ratio:** 7.01

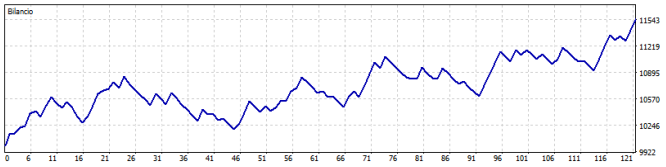

Figure 26: Equity curve — Baseline strategy with EMA(6,24) and ADX threshold 22.

Backtest Results with Random Forest and Optimized ADX Threshold

- **Total Net Profit:** 1,757.78 USD
- **Max Drawdown (Balance):** 559.54 USD (4.89%)
- **Profit Factor:** 1.53
- **Total Trades:** 121
- **Winning Trades:** 57.00%
- **Max Consecutive Wins:** 6 (537.96 USD)
- **Sharpe Ratio:** 8.56

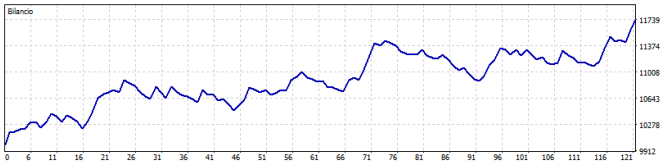

Figure 27: Equity curve — Random Forest strategy with EMA(6,24) and ADX threshold 22.

Backtest Results with XGBoost and Optimized ADX Threshold

- **Total Net Profit:** 2,053.93 USD
- **Max Drawdown (Balance):** 543.09 USD (4.71%)
- **Profit Factor:** 1.61
- **Total Trades:** 121
- **Winning Trades:** 47.11%
- **Max Consecutive Wins:** 6 (595.01 USD)
- **Sharpe Ratio:** 9.24

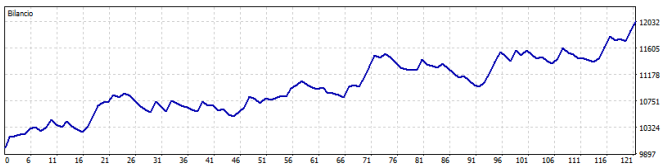

Figure 28: Equity curve — XGBoost strategy with EMA(6,24) and ADX threshold 22.

Backtest Results with LSTM and Optimized ADX Threshold

- **Total Net Profit:** -356.78 USD
- **Max Drawdown (Balance):** 1,358.86 USD (13.04%)
- **Profit Factor:** 0.91
- **Total Trades:** 121
- **Winning Trades:** 47.11%
- **Max Consecutive Wins:** 6 (280.36 USD)
- **Sharpe Ratio:** -2.03

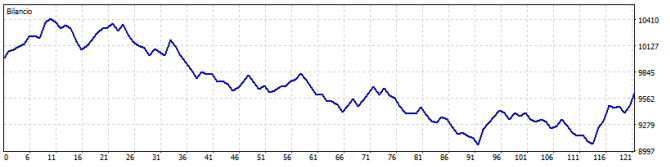

Figure 29: Equity curve — LSTM strategy with EMA(6,24) and ADX threshold 22.

Summary of ADX Optimization Results

Introducing the optimized ADX threshold of 22 led to a substantial increase in trade frequency and total profitability for the baseline, Random Forest, and XGBoost strategies. Notably, the XGBoost model achieved the highest net profit and Sharpe ratio, confirming its superior adaptability under the revised entry conditions. In contrast, the LSTM-based approach deteriorated in performance, incurring a net loss and elevated drawdown, thereby indicating instability under the more reactive filtering conditions introduced by the lower ADX threshold.

Volume Indicator Replacement: From iVolume to Accumulation/Distribution (iAD)

In the final phase of the strategy refinement process, we investigated the potential impact of replacing the standard volume filter based on tick volume (`iVolume`) with a more sophisticated measure of market participation, the Accumulation/Distribution indicator (`iAD`).

iVolume is a straightforward indicator that returns the number of price changes (ticks) within a given candle. It serves as a proxy for market activity but does not account for price directionality or the distribution of traded volumes.

iAD, on the other hand, attempts to capture cumulative buying and selling pressure by factoring in both price movements and volume across bars. It increases when the close is near the high (suggesting accumulation) and decreases when the close is near the low (suggesting distribution), thus providing a more nuanced view of underlying market sentiment.

The rationale behind testing `iAD` was to explore whether its directional volume logic could improve the robustness of trend confirmation within the strategy, especially in filtering out false signals in low-momentum phases.

Backtest Results without AI Using iAD Indicator

To assess the impact of replacing the standard volume filter with the Accumulation/Distribution (iAD) indicator, the baseline strategy was re-tested using EMA parameters (6,24) and an ADX threshold of 22. The results are summarized below:

- **Total Net Profit:** 236.26 USD
- **Maximum Drawdown (Balance):** 1,497.80 USD (14.11%)
- **Profit Factor:** 1.06
- **Total Trades:** 116
- **Winning Trades Percentage:** 41.38%
- **Max Consecutive Wins:** 6 (536.00 USD)
- **Sharpe Ratio:** 1.07

The strategy experienced a substantial decline in performance, with negative profitability and significantly higher drawdown compared to the version employing the standard volume filter. The sharp drop in the profit factor and Sharpe ratio highlights a marked deterioration in both absolute and risk-adjusted returns.

Backtest Results with Random Forest Using iAD Indicator

In this configuration, the Random Forest-enhanced strategy was tested with the iAD indicator, while retaining the same EMA and ADX settings. Results are as follows:

- **Total Net Profit:** 341.31 USD
- **Maximum Drawdown (Balance):** 1,195.93 USD (11.23%)
- **Profit Factor:** 1.12
- **Total Trades:** 116
- **Winning Trades Percentage:** 41.38%
- **Max Consecutive Wins:** 6 (475.90 USD)
- **Sharpe Ratio:** 1.90

Relative to the same setup using `iVolume`, this configuration produced significantly worse results across all metrics, indicating a substantial degradation in predictive reliability and risk control.

Backtest Results with XGBoost Using iAD Indicator

The XGBoost-based strategy was evaluated using the iAD indicator under identical EMA and ADX conditions. The backtest outcomes were as follows:

- **Total Net Profit:** 893.66 USD
- **Maximum Drawdown (Balance):** 1,178.43 USD (11.15%)

- **Profit Factor:** 1.29
- **Total Trades:** 116
- **Winning Trades Percentage:** 41.38%
- **Max Consecutive Wins:** 6 (475.90 USD)
- **Sharpe Ratio:** 4.25

Despite the use of a more advanced prediction model, the integration of iAD failed to deliver meaningful benefits. Profitability and drawdown worsened, confirming the incompatibility of this indicator with the overall strategy design.

Backtest Results with LSTM Using iAD Indicator

The LSTM-enhanced strategy also underwent evaluation using the iAD indicator, with consistent EMA and ADX parameters. Key metrics are listed below:

- **Total Net Profit:** -1,306.78 USD
- **Maximum Drawdown (Balance):** 1,686.54 USD (16.37%)
- **Profit Factor:** 0.66
- **Total Trades:** 116
- **Winning Trades Percentage:** 41.38%
- **Max Consecutive Wins:** 6 (302.30 USD)
- **Sharpe Ratio:** -5.00

This variant produced the worst performance among all models, with negative returns, high volatility, and low statistical confidence. The iAD indicator clearly failed to enhance the LSTM's ability to identify profitable trading opportunities.

Note on Equity Curves

Due to the consistently negative outcomes obtained across all models when employing the iAD indicator, the associated equity curves have been deliberately excluded. The numerical results alone sufficiently illustrate the degradation in performance, and their omission contributes to a clearer and more concise presentation.

Summary of iAD Indicator Evaluation

The substitution of the `iVolume` filter with the `iAD` (Accumulation/Distribution) indicator was intended to provide richer context on market participation by combining price and volume data. However, experimental results demonstrated a universal decline in strategy effectiveness across all tested models.

In every configuration, the iAD-based filter produced lower net profit, increased drawdown, and significantly worse Sharpe ratios. These outcomes suggest that, within this system's framework, raw volume data remains a more robust and reliable signal. As a result, the original volume-based filter was retained for all subsequent development and evaluation stages.

3.9. Backtest Results – Post-Optimization Performance

After identifying suboptimal performance patterns during the initial backtests, a comprehensive optimization process was conducted (see Section 3.8) to fine-tune the strategy's key technical parameters. Specifically, the EMA crossover configuration was adjusted to **(6, 24)** and the ADX threshold was lowered to **22**, as these settings offered the best trade-off between profitability, drawdown control, and trade frequency.

This section presents the backtest results obtained using the optimized strategy configuration applied uniformly across all AI models: Random Forest, XGBoost, and LSTM. These results allow for a direct comparison of model performance under improved trading conditions and highlight the impact of strategy optimization on overall profitability and risk metrics.

All tests were conducted on the **XAGUSD.cy** instrument using the **M12** timeframe over the period from **January 2025 to June 2025**. The expert advisor operated with the updated indicators and maintained the use of `iVolume` as a filter, which had previously demonstrated superior results compared to the alternative `iAD` indicator.

Each of the following subsections details the specific backtest performance metrics and includes visual analyses such as equity curves, trade distribution, profit correlation with MFE/MAE, and holding time patterns, to offer a holistic view of each model's behavior under optimized conditions.

3.9.1. Random Forest (RF) – Post-Optimization Results

Key Results:. The Random Forest-enhanced strategy, tested under the optimized configuration (EMA 6/24 and ADX threshold 22), delivered positive outcomes with a total net profit of **1,757.78 USD**. The system recorded a gross profit of **5,069.00 USD** against gross losses totaling **-3,311.22 USD**. The maximum drawdown reached **559.54 USD** (4.89%) on balance, and **579.79 USD** (5.07%) on equity.

Key performance metrics include:

- **Profit Factor**: 1.53
- **Expected Payoff**: 14.53 USD
- **Recovery Factor**: 3.03
- **Sharpe Ratio**: 8.56
- **Number of Trades**: 121
- **Winning Trades**: 57 (47.11%)
- **Losing Trades**: 64 (52.89%)
- **Max Consecutive Wins**: 6 (537.96 USD)
- **Max Consecutive Losses**: 5 (-264.75 USD)

Graphical Analysis:. The following figures illustrate the performance and behavioral patterns of the Random Forest strategy under the optimized parameter configuration:

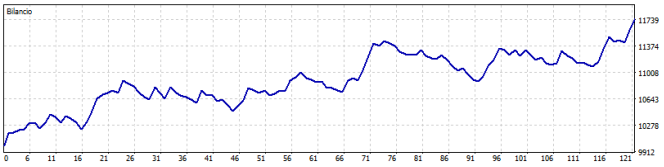

Figure 30: Equity curve for the Random Forest model after optimization.

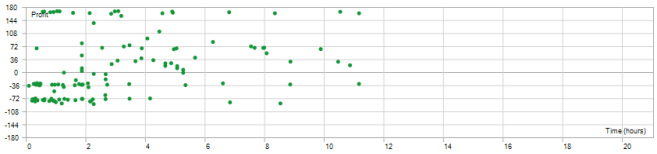

Figure 31: Profit vs. holding time during RF post-optimization backtest.

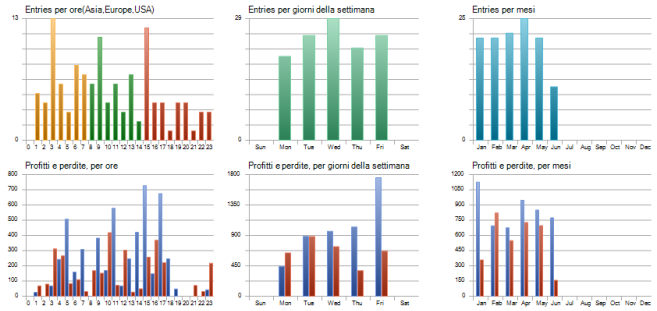

Figure 32: Entry distribution and profitability by time for RF post-optimization.

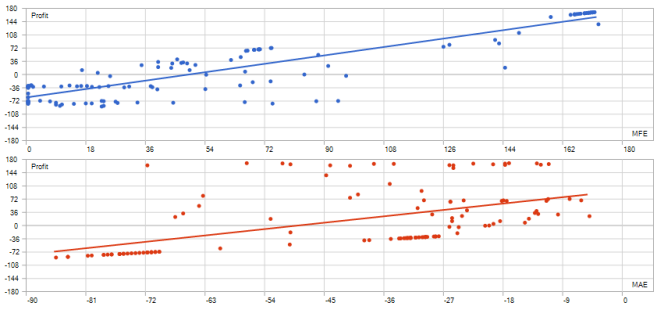

Figure 33: Profit correlation with MFE and MAE for RF post-optimization.

Conclusion:. The post-optimization results confirm that Random Forest remained a robust and profitable forecasting model. It maintained a favorable risk-to-reward ratio and showed reliable trade behavior, especially under the refined EMA and ADX settings.

3.9.2. XGBoost Backtest Results – Post-Optimization

Key Results:. The backtest using the XGBoost-enhanced strategy under the optimized settings (EMA 6/24, ADX threshold 22) produced robust results, achieving a total net profit of **2,053.93 USD**. The system recorded a gross profit of **5,441.75 USD** against a gross loss of **-3,387.82 USD**, with a maximum balance drawdown of **543.09 USD** (4.71%) and an equity drawdown of **563.69 USD** (4.89%).

Key performance metrics include:

- **Profit Factor**: 1.61
- **Expected Payoff**: 16.97 USD
- **Recovery Factor**: 3.64
- **Sharpe Ratio**: 9.24
- **Number of Trades**: 121
- **Winning Trades**: 57 (47.11%)
- **Losing Trades**: 64 (52.89%)
- **Max Consecutive Wins**: 6 (595.01 USD)
- **Max Consecutive Losses**: 5 (-308.85 USD)

Graphical Analysis:. The following visualizations illustrate the XGBoost model's behavior under the optimized trading regime:

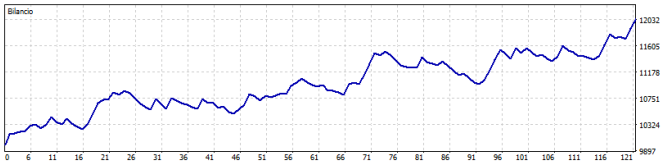

Figure 34: Equity curve for the XGBoost strategy post-optimization on XAGUSD.cy (M12).

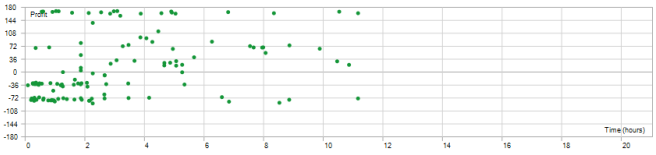

Figure 35: Profit vs. Holding time for the XGBoost model post-optimization.

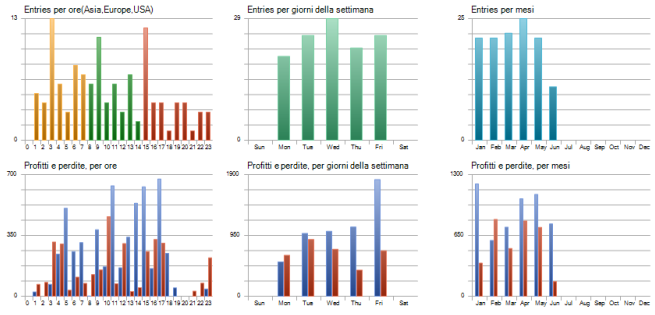

Figure 36: Temporal and categorical analysis of trade entries and performance for XGBoost post-optimization.

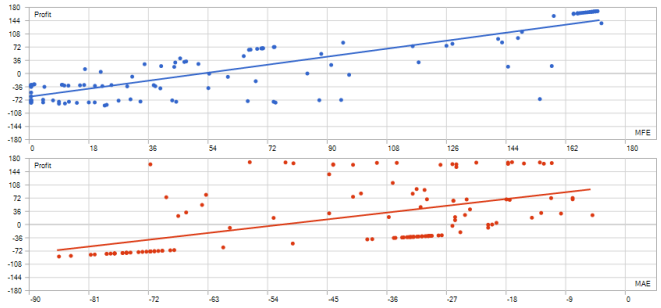

Figure 37: Profit correlation with MFE and MAE for the XGBoost model.

Conclusion:. The XGBoost model delivered the best overall profitability among the three AI approaches when deployed under the optimized conditions. With a strong balance between return and risk—as evidenced by the high Sharpe Ratio and recovery factor—the model proved adept at identifying profitable trades while controlling drawdowns. The high correlation with favorable price excursions further reinforces its utility in capturing extended market trends. Overall, XGBoost emerged as a highly effective component of the optimized strategy.

3.9.3. LSTM – Post-Optimization Results

Key Results:. The performance of the LSTM-based strategy under optimized conditions (EMA 6/24 and ADX threshold 22) was considerably weaker than its tree-based counterparts. The system concluded with a total net loss of **-356.78 USD**, derived from a gross profit of **3,436.76 USD** and a gross loss of **-3,793.54 USD**. The maximum drawdown reached **1,358.86 USD** on balance (13.04%) and **1,419.86 USD** on equity (13.60%).

Key performance metrics include:

- **Profit Factor**: 0.91
- **Expected Payoff**: -2.95 USD
- **Recovery Factor**: -0.25
- **Sharpe Ratio**: -2.03
- **Number of Trades**: 121
- **Winning Trades**: 57 (47.11%)
- **Losing Trades**: 64 (52.89%)
- **Max Consecutive Wins**: 6 (280.36 USD)
- **Max Consecutive Losses**: 5 (-414.00 USD)

Graphical Analysis:. To further understand the model's behavior, a series of visualizations are presented below.

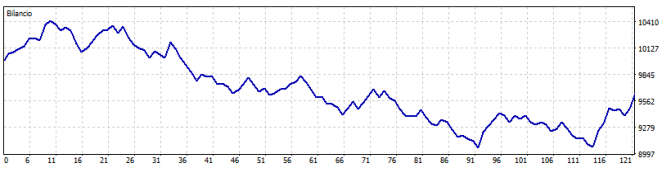

Figure 38: Equity curve for the LSTM model with post-optimization parameters.

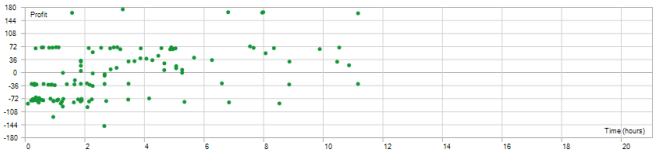

Figure 39: Profit by holding time (LSTM, post-optimization).

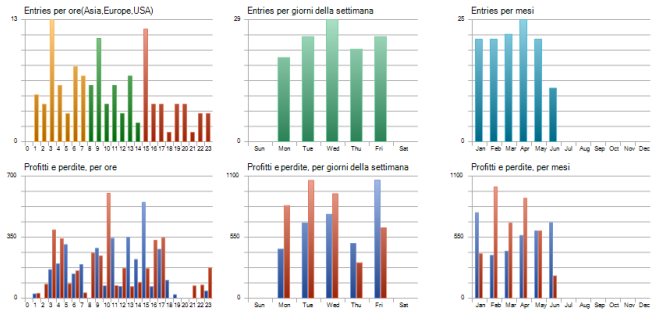

Figure 40: Trade frequency and profitability distribution by hour, weekday, and month.

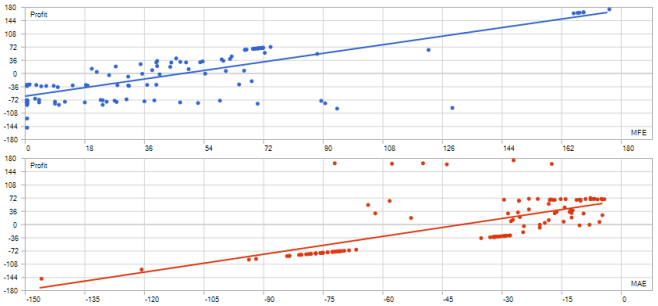

Figure 41: Profit correlation with MFE and MAE (LSTM model).

Conclusion:. Overall, the LSTM model failed to deliver positive returns under the optimized trading strategy. The elevated drawdowns, combined with negative risk-adjusted metrics and a downward-sloping equity curve, suggest that this model architecture may require further tuning or alternative data preprocessing to perform competitively in this context.

3.10. Summary of Post-Optimization Backtest Results

This section provides a comparative summary of the post-optimization backtest performance across all three AI models: Random Forest (RF), XGBoost, and LSTM. The table below reports key performance indicators such as net profit, maximum drawdown, profit factor, and Sharpe ratio for each model under the optimized configuration.

Table 7: Summary of Post-Optimization Results by Model

Model	Net Profit (USD)	Max Drawdown (USD)	Profit Factor	Sharpe Ratio
XGBoost	2,053.93	563.69	1.61	9.24
Random Forest	1,757.78	579.79	1.53	8.56
LSTM	-356.78	1,419.86	0.91	-2.03

This comparative overview reveals that **XGBoost** achieved the highest net profit ($2,053.93), closely followed by **Random Forest** ($1,757.78). In terms of risk-adjusted performance, XGBoost also posted the best **Sharpe Ratio** (9.24) and **Profit Factor** (1.61), making it the most effective model in the post-optimization phase. While Random Forest demonstrated slightly lower profitability, it still maintained a strong balance between gains and drawdowns, confirming its robustness.

Conversely, the **LSTM model** underperformed significantly, closing the test period with a net loss of **-$356.78** and the highest drawdown ($1,419.86). These results suggest that, at least within the tested timeframe and configuration, recurrent neural network architectures such as LSTM may be less suited for the current strategy compared to tree-based models.

3.11. Live Prediction via Socket Communication

In this section, we describe the integration of real-time AI predictions into the trading strategy through socket communication. The Expert Advisor (EA) is designed to communicate with a Python server that hosts the pre-trained AI models. This enables the EA to send live market data and receive predictions on market trends (uptrend or downtrend) to adjust trading decisions in real time.

The communication flow between the EA and the Python server is as follows:

- The EA collects the current market features (such as EMA, ADX, price, and volume) and sends this data to the Python server via a TCP socket.
- The Python server, which hosts the pre-trained AI model (e.g., XGBoost), receives the market data, processes it, and predicts the market trend.
- The predicted trend (uptrend or downtrend) is sent back to the EA, where it is used to adjust the lot size.

The Python server listens for incoming data from the EA, processes the received market features, and returns a prediction based on the pre-trained model. The server is capable of handling incoming connections, performing preprocessing, scaling the input data, and returning predictions in real-time.

The Expert Advisor, operating in MetaTrader, uses the received predictions to adjust its decision-making process. If the prediction from the AI model agrees with the technical indicators (such as EMA crossovers and ADX thresholds), the EA may enter a trade with an increased lot size. Conversely, if there is disagreement, the EA may reduce the lot size.

This integration allows the EA to leverage the power of the AI model, enhancing the decision-making process by incorporating real-time trend predictions. The use of socket communication ensures that the system remains flexible and efficient, allowing for continuous interaction between the EA and the Python server.

4. Discussion

In this section, we analyze the post-optimization performance of the three AI-driven models—Random Forest, XGBoost, and LSTM—and reflect on their respective behaviors under improved trading conditions. By comparing these results with the earlier pre-optimization phase, we draw insights into the impact of strategy refinement and the broader implications of deploying AI in real-time trading systems.

4.1. Performance Comparison

Following the optimization process, the XGBoost model emerged as the top-performing strategy across all major performance metrics. It achieved the highest net profit of **2,053.93 USD**, as well as the best risk-adjusted returns with a **Sharpe Ratio of 9.24** and a **Profit Factor of 1.61**. Additionally, XGBoost maintained a relatively low drawdown (**563.69 USD**), suggesting effective risk containment alongside strong profitability. These results indicate that, under the refined parameter settings, XGBoost offered the most favorable balance between return and risk.

The Random Forest model also performed well, generating a net profit of **1,757.78 USD** with a **Sharpe Ratio of 8.56** and a **Profit Factor of 1.53**. While slightly less profitable than XGBoost, Random Forest demonstrated strong consistency and solid risk control. Its comparable drawdown (**579.79 USD**) further reinforces its robustness as a model suited for stable, lower-volatility strategies.

In contrast, the LSTM model failed to improve under the optimized configuration. It recorded a net loss of **-356.78 USD** with a **Profit Factor of 0.91** and a **Sharpe Ratio of -2.03**. These metrics suggest the LSTM model was unable to effectively capture profitable trading patterns and suffered from elevated drawdowns. This underperformance may reflect limitations in its ability to generalize on non-stationary financial time series or a mismatch between the model architecture and the strategy's decision-making requirements.

Overall, the results confirm that tree-based models, particularly XGBoost, are better suited to the given trading framework than recurrent neural network architectures like LSTM. XGBoost, in particular, demonstrated a compelling combination of profitability, risk-adjusted return, and drawdown control, establishing itself as the most effective model in the post-optimization analysis.

4.2. Implications of AI Integration in Real-Time Trading

The integration of AI models into a real-time trading system—via dynamic socket-based communication with the Expert Advisor (EA)—proved to be a practical and technically viable solution for enhancing trade decisions. By leveraging real-time predictions, the EA was able to modulate trade execution parameters such as volume and entry timing based on the current state of the market and the AI model's outlook.

Post-optimization results suggest that incorporating machine learning models, particularly tree-based architectures like XGBoost and Random Forest, can significantly improve the risk-adjusted returns of rule-based strategies. XGBoost, in par-

ticular, demonstrated an excellent balance between profitability and drawdown control, making it a compelling candidate for live deployment. Random Forest, while slightly less profitable, maintained strong consistency and robust risk management—further validating the utility of ensemble learning methods in financial forecasting.

Despite these encouraging results, several practical challenges remain when transitioning from backtesting to real-time trading. A key concern is model adaptability. While the models were effective on historical data, their continued success depends on their ability to generalize in the face of evolving market dynamics. Financial markets are inherently non-stationary, and models may degrade over time unless they are retrained regularly on updated datasets.

Overfitting also remains a risk—especially when model complexity increases or optimization is too tightly coupled to past market conditions. To mitigate this, a systematic approach to model validation, cross-market testing, and periodic retraining should be part of any robust deployment framework.

Furthermore, latency and synchronization between the EA and AI prediction server must be tightly managed. Any delays in prediction delivery or discrepancies in data alignment can lead to suboptimal trade execution. These aspects become particularly critical in lower-timeframe strategies where execution precision is paramount.

Future iterations of this framework could explore adaptive or online learning techniques to allow continuous retraining as new data becomes available. Incorporating more advanced ensemble methods, or hybrid systems combining rule-based logic with reinforcement learning agents, may also yield improvements in adaptability and robustness across diverse market conditions.

4.3. Future Directions and Limitations

While the post-optimization results substantially improved the overall performance of the trading system, several limitations remain that merit further investigation. Notably, the underperformance of the LSTM model demonstrates that deep learning architectures do not inherently guarantee superior outcomes in financial time series forecasting. This may be due to the high noise, non-stationarity, and limited dataset size typical of financial data. Future research could explore alternative architectures such as Temporal Convolutional Networks (TCNs) or Transformer-based models, which have shown promise in capturing temporal dependencies with greater stability.

Another important limitation of the current framework is its reliance on static, offline training. Models are trained once and remain unchanged during live operation, which can lead to a gradual loss of accuracy as market conditions evolve. Implementing online learning or adaptive retraining routines—where the model continuously updates its parameters based on new incoming data—could help maintain predictive relevance and responsiveness, especially in high-volatility environments.

Additionally, although the socket-based architecture successfully enabled real-time communication between the Expert Advisor and the AI models, it introduces a layer of complexity that must be carefully managed in live deployments. Latency, synchronization, and failover resilience become critical considerations when deploying such systems in production environments.

In summary, the post-optimization phase confirmed the effectiveness of tree-based models—particularly XGBoost and Random Forest—in enhancing the baseline strategy. While XGBoost achieved the highest profitability, Random Forest offered a better balance between return and risk. These results underscore the importance of selecting models that align well with both the trading objectives and the operational constraints of the deployment environment. Continued experimentation with alternative architectures and adaptive training paradigms will be key to further improving robustness and performance in live financial applications.

5. Conclusion

This study investigated the integration of machine learning models—Random Forest, XGBoost, and LSTM—into a baseline Expert Advisor (EA) to enhance trend-following strategies in algorithmic trading. While early backtesting indicated mixed performance across the models, a subsequent optimization of the underlying strategy parameters (EMA and ADX thresholds) significantly improved the outcome and allowed for a more meaningful comparison of each model's effectiveness.

Post-optimization results revealed that the **XGBoost** model delivered the most favorable overall performance, achieving the highest net profit (**2,053.93 USD**), strongest risk-adjusted return (**Sharpe Ratio: 9.24**), and best profitability-to-risk ratio (**Profit Factor: 1.61**). These findings highlight XGBoost's strong capacity to capture favorable market trends while containing drawdowns, making it a compelling candidate for real-time deployment in systematic trading strategies.

The **Random Forest** model also performed well, offering a strong balance between return and risk. Although its net profit (**1,757.78 USD**) was slightly lower than XGBoost's, it maintained high consistency with a Sharpe Ratio of **8.56** and a Profit Factor of **1.53**. These results suggest that Random Forest remains a reliable choice, especially for strategies prioritizing stability and robustness over aggressive profit maximization.

In contrast, the **LSTM** model failed to deliver competitive performance despite the improved strategy settings. It concluded with a net loss of **-356.78 USD** and negative risk-adjusted metrics. This underperformance may stem from LSTM's sensitivity to non-stationarity, model complexity, or suboptimal data representation. While LSTM holds promise in broader time series forecasting contexts, its application in this specific financial setup may require more advanced tuning or alternative architectures.

Overall, the post-optimization phase confirmed the value of AI integration in trading systems—particularly with tree-based models such as XGBoost and Random Forest. These models enhanced the profitability and risk management capabilities of the baseline EA, demonstrating that machine learning can provide substantial improvements when aligned with a well-calibrated rule-based framework.

Looking ahead, future work should focus on extending this framework through online learning or adaptive retraining, allowing models to evolve in response to changing market conditions. Additionally, exploring hybrid approaches—such as combining ensemble methods with reinforcement learning or transformer-based architectures—may unlock further potential in capturing complex market dynamics.

In conclusion, this research affirms that machine learning, when effectively optimized and integrated, can significantly elevate the performance of algorithmic trading systems. The hybrid rule-AI architecture presented in this study offers a flexible, modular foundation for further innovation in financial machine learning applications.

References

Ailyn, D., 2024. Deep learning for high-frequency trading: Predicting market movements with time-series data .

Ansary, M.S., 2023. Machine learning for predicting the stock price direction with trading indicators. ENP Engineering Science Journal 3, 34–40. doi:10.53907/enpesj.v3i2.178.

Ballings, M., Van den Poel, D., Hespeels, N., Gryp, R., 2015. Evaluating multiple classifiers for stock price direction prediction. Expert Systems with Applications 42, 7046–7056. URL: https://www.sciencedirect.com/science/article/pii/S0957417415003334, doi:https://doi.org/10.1016/j.eswa.2015.05.013.

Fischer, T., Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research 270, 654–669. URL: https://www.sciencedirect.com/science/article/pii/S0377221717310652, doi:https://doi.org/10.1016/j.ejor.2017.11.054.

Ouf, S., El Hawary, M., Aboutabl, A., Adel, S., 2025. A comparative analysis of the performance of machine learning and deep learning techniques in predicting stock prices. Journal of Computer and Communications 13, 1–17.

Wu, H., 2024. Comparison of random forest and lstm in stock prediction. Advances in Economics, Management and Political Sciences 86, 28–34. doi:10.54254/2754-1169/86/20240936.