BAREKENG: Journal of Mathematics and Its Applications
September 2026 Volume 20 Issue 3 Page 2131-2150
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol20iss3pp2131-2150

GREY WOLF-OPTIMIZED XGBOOST REGRESSOR FOR STOCK INDEX PREDICTION WITH FINANCIAL FEATURES

Syaiful Anam[1*], Mohd Razif Shamsuddin[2], Elyas Kustiawan[3], Dwi Mifta Mahanani[4], Feby Indriana Yusuf[5]

[1,4,5] *Mathematics Department, Faculty of Science, Universitas Brawijaya*
Jln. Veteran, Malang, 65145, Indonesia

[2] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA (UiTM)*
Jln. Ilmu 1/1, Shah Alam, Selangor, 40450, Malaysia

[3] *Faculty of Science and Engineering, Universitas Bangka Belitung*
Jln. Kampus Peradaban Balunijuk, Bangka, 33126, Indonesia

*Corresponding author's e-mail: * **syaiful@ub.ac.id***

Article Info

Article History:

Received: 4th August 2025
Revised: 6th January 2026
Accepted: 8th March 2026
Available online: 8th April 2026

Keywords:

Extreme Gradient Boosting;
Financial Features;
Grey Wolf Optimization;
Hyperparameter Optimization;
Stock Index Prediction.

ABSTRACT

Accurate stock index prediction is essential for effective investment strategies and economic policymaking. While traditional statistical models often fail to capture the nonlinear dynamics of financial markets, machine learning approaches—particularly Extreme Gradient Boosting (XGBoost)—offer greater flexibility, robustness to overfitting, and computational efficiency. However, the performance of XGBoost strongly depends on hyperparameter tuning, which is difficult to optimize using conventional search methods. To address this, we propose a hybrid framework that integrates XGBoost with Grey Wolf Optimization (GWO) for enhanced hyperparameter selection in stock index forecasting. Using historical data from the Indonesian BBNI stock index (2021–2024) and financial features (price, volume, and temporal), the GWO-optimized XGBoost achieved superior performance, recording the lowest testing MAPE (1.79%), RMSE (108.67), and MAE (84.32). These results surpass classical regressors (Decision Tree, Random Forest, Multilayer Perceptron, Gradient Boosting) by margins of 6–26% and outperform conventional tuning methods (Grid Search, Random Search, Bayesian Optimization) as well as other swarm intelligence approaches (PSO, BA). Moreover, the GWO-based approach reduced error variability and required significantly less optimization time, with the 10-wolf configuration providing the best accuracy–efficiency tradeoff. The scope of this study is limited to a single stock index (BBNI.JK) and financial features, without incorporating macroeconomic indicators, sentiment variables, or cross-market validation. These limitations indicate potential directions for future work to enhance generalizability. Overall, the proposed GWO-XGBoost framework provides a powerful, stable, and time-efficient solution for stock index prediction in volatile market conditions.

How to cite this article:

S. Anam, M. R. Shamsuddin, E. Kustiawan, D. M. Mahanani, and F. I. Yusuf, "GREY WOLF-OPTIMIZED XGBOOST REGRESSOR FOR STOCK INDEX PREDICTION WITH FINANCIAL FEATURES", *BAREKENG: J. Math. & App.*, vol. 20, no. 3, pp. 2131-2150, Sep, 2026.

Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*

Research Article · Open Access

1. INTRODUCTION

Stock index prediction plays a vital role in both investment decision-making and macroeconomic policy formulation. Investors can gain strategic advantages that enable more effective asset allocation and risk management by accurately predicting stock index movements. Simultaneously, policymakers and regulators benefit from early warning indicators that help them identify potential market instabilities [1]. Furthermore, accurate stock index prediction enhances the stability and efficiency of financial ecosystems. Reliable predictions support better liquidity management, strengthen risk mitigation efforts, and aid in the accurate pricing of derivative products such as futures, all of which depend heavily on precise index forecasts [2]. Ultimately, advancements in forecasting methodologies contribute to more informed investment decisions and bolster the overall resilience of financial markets [1].

Stock index prediction encompasses a wide range of methodologies that can broadly be divided into traditional statistical techniques and modern Machine Learning (ML). Traditional statistical methods have been valued for their interpretability and the ease with which they allow understanding of the statistical properties of the predicted series [3], [4]. Traditional econometric models such as ARIMA and GARCH, which rely on linear and stationary assumptions, often fail to capture financial market complexities. In the Indonesian context, these shortcomings are particularly pronounced: stock market behavior is frequently shaped by abrupt policy changes, regulatory interventions, and irregular trading volumes, which give rise to volatility clustering and structural breaks. As a result, ARIMA and GARCH models frequently underperform in emerging markets like Indonesia. In contrast, ML techniques offer greater flexibility in modeling nonlinear relationships in financial data [5], [6], [7]. Unlike traditional methods that require handcrafted features and rigid distributional assumptions, ML techniques can autonomously identify complex patterns across diverse datasets, such as technical indicators, macroeconomic variables, and market sentiment [8]. These limitations underscore the necessity of adopting more flexible and robust approaches, particularly machine learning models, which are capable of handling nonlinear relationships, high-dimensional features, and rapidly changing market conditions.

Extreme Gradient Boosting (XGBoost) has gained wide attention in stock index prediction due to its ability to capture complex nonlinear market behaviors [9], [10], supported by scalability, regularization, and fast parallelized training [10], [11]. Prior studies confirm that it often outperforms econometric models such as ARIMA and GARCH [9], [10]. However, XGBoost is also sensitive to hyperparameter settings, with risks of underfitting, overfitting, and limited interpretability compared to simpler models. Hence, hyperparameter tuning is critical for maximizing its performance in tasks such as stock prediction and fraud detection [12], [13]. Traditional methods like grid and random search are slow or suboptimal, while metaheuristics—including Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and other approaches—have shown improved convergence and predictive accuracy [14]-[19]. Empirical evidence, such as Poernamawatie et al. [20], further demonstrates that rigorous tuning enables models to outperform typical benchmarks. Beyond accuracy, effective tuning reduces complexity and accelerates convergence [21], [22], with advanced metaheuristic strategies providing systematic and robust hyperparameter optimization [23], [24].

The Swarm Intelligence (SI) algorithm is one of the metaheuristics methods that have emerged as effective tools for hyperparameter tuning in XGBoost, offering advantages over conventional methods like grid and random search. Their population-based search strategies enable simultaneous exploration of multiple candidate solutions, improving the ability to escape local optima [25]. Unlike traditional exhaustive searches, SI algorithms significantly reduce computational overhead while maintaining optimization quality [26], [27]. Empirical studies demonstrate that PSO-driven hyperparameter tuning consistently lowers forecast errors and boosts predictive reliability in volatile stock index environments [28], [29], confirming their superiority over static tuning methods.

Although PSO and other swarm intelligence methods have shown promise, recent advancements point to even more efficient strategies. The Grey Wolf Optimization (GWO) algorithm offers further enhancements over PSO and other SI methods. By simulating the leadership hierarchy and hunting behavior of grey wolves, GWO achieves a stronger balance between exploration and exploitation [30]. Unlike PSO, which often suffers from premature convergence due to limited diversity, GWO maintains population variation longer, improving its ability to avoid local optima and enhancing global search performance [31], [32]. It also requires fewer parameters and converges more quickly, increasing efficiency and ease of use [33], [34]. Compared to the Bat Algorithm (BA), GWO demonstrates superior convergence stability and robustness. While BA's echolocation-based updates can lead to stagnation in complex landscapes, GWO adapts better to dynamic

environments [35], [36]. Its consistent success across domains, such as control systems and machine learning, underscores GWO's versatility and justifies its use in this study [37], [38].

In addition to selecting appropriate and effective ML methods, the analysis of price, volume, and temporal features is crucial in stock index prediction, driven by technological advancements and the availability of large-scale data. Traditional models typically rely on OHLCV (open, high, low, close, volume) data as fundamental inputs. A study highlights these daily stock metrics as essential features in ML-based market prediction models [39]. Temporal features such as day, month, and year further enhance predictive performance by capturing seasonal and contextual patterns. Yang (2023) also emphasizes the importance of incorporating both temporal and contextual data across various ML algorithms [40]. Therefore, effective stock index forecasting requires a careful combination of price, volume, and temporal features (financial features), supported by robust feature selection and the application of advanced ML techniques.

Based on these problems, this study proposes an XGBoost Regressor model optimized by the GWO algorithm, referred to as XGBoost Regressor + GWO, for stock index prediction using financial features (price, volume, and temporal). The proposed approach introduces a GWO-driven hyperparameter optimization strategy specifically designed for XGBoost models to enhance their predictive capability. By integrating GWO, the model is able to significantly improve forecasting accuracy, accelerate convergence speed, and enhance generalization performance compared to conventional hyperparameter tuning methods. Furthermore, the developed XGBoost Regressor + GWO framework provides a robust stock index forecasting method based on financial features that achieves higher predictive accuracy, greater adaptability to market volatility, and improved reliability for decision-making in dynamic financial environments.

The use of GWO facilitates efficient exploration of the high-dimensional hyperparameter space, leading to superior convergence behaviour and reduced computational cost. Overall, this work offers a practical and powerful solution for enhancing stock index prediction, providing financial analysts, investors, and algorithmic traders with more accurate, efficient, and responsive decision-making tools in dynamic market environments.

2. RESEARCH METHODS

This study adopts a quantitative experimental design to develop a hybrid forecasting framework that integrates the GWO algorithm with XGBoost for stock index prediction by using financial features. The flowchart of the research method could be seen in Fig. 1. Fig. 1 illustrates the overall research workflow, consisting of six main stages: data collection, data preparation, feature engineering, data splitting and normalization, model development, and model evaluation. The dataset was collected from Yahoo Finance covering the period January 2021 to December 2024, comprising 960 daily trading records of the BBNI stock index. During data preparation, the dataset was checked for duplicates and missing values, with results showing 0% missing data, and then sorted chronologically to preserve the time-series structure. Feature engineering generated both raw and derived predictors: five OHLCV features (Open, High, Low, Close, Volume), lag features up to 30 days, and four temporal features (day, month, date, year), resulting in a total of 39 predictive variables. For data splitting, the records were divided into 70% training (672 records), 15% validation (144 records), and 15% testing (144 records) to ensure fair temporal evaluation. Data normalization was conducted using the Min-Max method, scaling all features into the range [0,1] to improve training stability. Model development focused on training XGBoost Regressor models with hyperparameters optimized through the Grey Wolf Optimization (GWO) algorithm, exploring parameter ranges such as n_estimators (50–300), learning_rate (0.01–0.2), and max_depth (3–10). Finally, model evaluation was performed using Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE), where lower values indicate higher forecasting accuracy.

2.1 Dataset Description

Historical stock index data were collected from publicly available financial databases, which is Yahoo Finance, over the period from January 2021 to December 2024. This article uses a dataset which is "BBNI.JK" (Bank Negara Indonesia). The features and their description of each stock index can be seen in Table 1. The date feature represents the date trading. The date trading is a critical variable in predicting stock indexes because stock markets are time-series systems. They evolve over time, and historical behavior is strongly tied to dates. The open feature represents the opening price. It can indicate overnight market

sentiment and is often influenced by after-hours news, earnings reports, or global economic developments. While the high price (High) is important for identifying volatility and potential resistance levels, where price growth may stall. The low-price (Low) captures selling pressure. It helps model support levels, areas where prices may rebound upward, and overall intraday market risk. The closing price (Close) is the most commonly used reference point for stock index forecasting models because it summarizes the day's trading outcome. Volume measures market activity and liquidity. High volume often validates the strength of a price movement (up or down), while low volume suggests weak or uncertain momentum. The target for prediction in this research is the closing price (Close). Close is generally the better target because it accounts for corporate actions like dividends and stock splits, providing a more accurate reflection of the true price evolution over time.

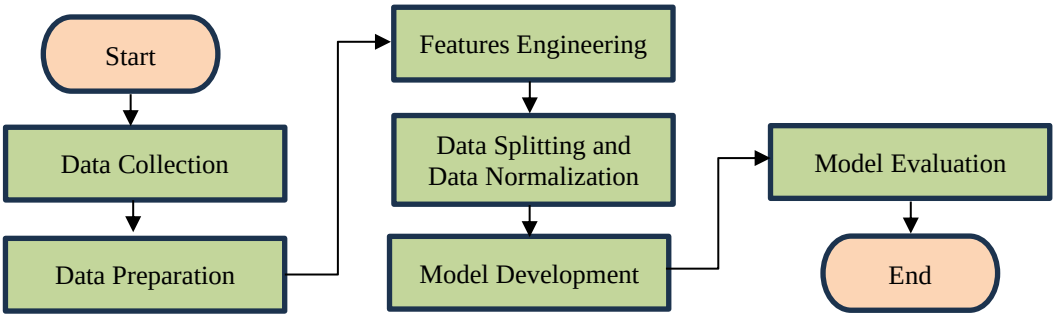


Figure 1. The Flowchart of the Research Method

Table 1. The Description of Each Feature

Feature	Description
Date	The trading date.
Open	The stock price at the beginning of the trading day.
High	The highest price the stock reached during the trading day.
Low	The lowest price the stock reached during the trading day.
Close	The stock price at the end of the trading day (not adjusted for dividends or splits).
Volume	The total number of shares traded during that trading day.

An example of the dataset is shown in Table 2, which contains daily stock index data with six columns: date, closing price, high, low, open, and trading volume. The dataset covers the period from January 4, 2021, to December 27, 2024. To illustrate the data structure and temporal span, Table 2 displays the first five (head) and last five (tail) records, representing the earliest and latest trading sessions. This presentation helps verify the dataset's continuity, completeness, and variable consistency prior to the preprocessing and forecasting analysis.

Table 2. The Dataset of Daily Stock Trading Data from January 4, 2021, to December 27, 2024

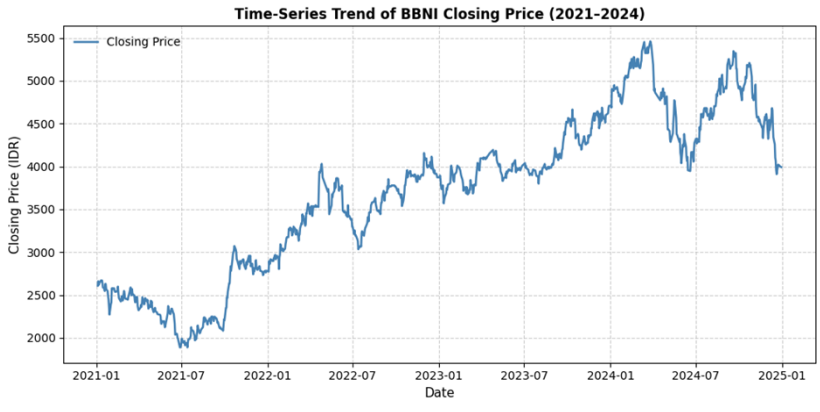
Date	Close	High	Low	Open	Volume
04/01/2021	2608.06372	2618.29142	2495.55901	2556.92522	81711400
05/01/2021	2659.20215	2659.20215	2577.38054	2618.29135	68509200
06/01/2021	2618.29126	2679.65746	2536.46966	2669.42976	74299600
07/01/2021	2628.51929	2659.20239	2597.83618	2648.97469	74138800
…	…	…	…	…	…
20/12/2024	3909.78369	4001.56265	3909.78369	3974.02896	83872500
23/12/2024	4019.91846	4029.09635	3964.85108	3964.85108	43668100
24/12/2024	4019.91846	4038.27425	3983.20687	4019.91846	21458900
27/12/2024	4001.56250	4065.80777	4001.56250	4019.91829	22838700

Fig. 2 illustrates the time-series movement of the BBNI closing price from 2021 to 2024, consistent with the statistical summary in Table 3. The price trend shows distinct phases of decline in early 2021, recovery in 2022, and sharp volatility during 2024. This pattern aligns with the dataset's wide min-max range (1,888–5,460) and moderate standard deviation (≈ 915), confirming fluctuating yet structured market behavior. The central concentration of values between Q1 ≈ 2,900 and Q3 ≈ 4,300 with a median near 3,865 indicates that most price movements remain within a stable band, while the slight negative skewness (≈ −0.2)

reflects occasional downward corrections. Together, Table 3 and Fig. 2 emphasize that the BBNI index exhibits nonlinear and non-stationary dynamics—conditions under which the hybrid GWO–XGBoost framework is particularly suitable for accurate and stable forecasting.

Table 3. Descriptive Statistics of the BBNI Stock Index Dataset (2021–2024)

Feature	Mean	Std	Min	Q1 (25%)	Median (50%)	Q3 (75%)	Max	Skewness
Open	3,692.65	916.77	1,888.11	2,906.37	3,862.69	4,362.13	5,460.85	−0.206
High	3,733.46	923.22	1,912.84	2,937.29	3,904.67	4,414.57	5,483.79	−0.207
Low	3,646.39	907.95	1,879.86	2,865.14	3,810.30	4,322.79	5,414.96	−0.219
Close	3,688.52	914.93	1,888.11	2,906.37	3,865.04	4,362.13	5,460.85	−0.215
Volume	60,572,350.5	35,595,520	13,398,400	36,728,200	50,698,800	73,345,20	309,557,00	2.192



F

Figure 2. Time-Series Trend of BBNI Closing Price (2021–2024)

2.2 Dataset Preparation

Dataset preparation is a critical initial step in stock index prediction to ensure data quality, consistency, and suitability for modeling. Data cleaning involves handling missing values, such as using forward-fill or interpolation techniques, and removing any duplicate records to ensure data consistency. In cases where missing values occur, we applied the forward-fill imputation technique, which replaces a missing entry with the most recent valid observation in the same column. This method is widely used in financial time-series data because stock prices and trading volumes evolve sequentially over time, and the latest available value provides a reasonable proxy for the next observation in the absence of recorded data. By using forward-fill, the continuity of the time series is maintained without introducing artificial trends or abrupt shifts that could distort model training. After checking, we can conclude that the dataset is complete and doesn't have any missing values. Therefore, we shouldn't do anything. The dataset in the research is sorted chronologically by date, and all fields are formatted properly for maintaining the correct temporal order. This step is required for time series analysis.

2.3 Feature Engineering

Feature engineering is a critical step in stock index prediction, tasked with extracting, refining, and fusing informative signals from raw market data to boost predictive performance. Feature engineering enriches the raw market data, helping models detect underlying trends, market momentum, and reversal signals. In stock index prediction, basic features such as open, high, low, close, and volume are often transformed or extended to reveal hidden patterns and enhance forecasting capabilities. Among various features employed in prediction models, price, volume, and temporal elements stand out as pivotal components. The interplay between these elements significantly influences the effectiveness of forecasting methodologies. The utilization of temporal features—specifically, the emphasis on the sequence of price movements over time—enables these models to capture trends and fluctuations more effectively than traditional methods [41], [42]. This research uses the open, high, low, and close prices from previous dates $(t - i)$ as input features, where i represents the number of days prior to the current time t. Moreover, the integration of trading volume into predictive models has been shown to amplify their prognostic capabilities. For example, studies have highlighted that variations in trading volume not only reflect market sentiment but

also serve as indicators of future price movements, making them significant predictors of stock volatility. Volume can act as a confirmatory signal to a price trend; high volume accompanying price moves suggests momentum, while low volume may indicate a lack of conviction [43], [44]. In this study, the input features consist of the volume observed at previous time steps, specifically at $(t - i)$, where i denotes the lag representing i time units before the current time t. Temporal features also enhance the dynamics of forecasting efforts. Additionally, models that implement ensemble learning techniques demonstrate improved accuracy by leveraging various temporal data inputs alongside price and volume [45]. This research extracts the day of the week, date, month, and year as the temporal features.

Table 4. Example of Feature-Engineering Output Showing Head and Tail Records with Added Temporal and Lag Features used for Model Training

No	Date	Close	High	Low	Open	Volume	Date	Month	Year	Day of Week	Close _t-1
1	07/01/2021	2628.519	2659.202	2597.836	2648.974	74138800	7	1	2021	2	2618.291
2	08/01/2021	2648.974	2659.202	2608.063	2628.519	72826800	8	1	2021	0	2628.519
3	11/01/2021	2669.429	2700.113	2638.746	2648.974	149359200	11	1	2021	1	2648.974
4	12/01/2021	2659.202	2730.796	2648.974	2700.113	120481200	12	1	2021	3	2669.43
5	13/01/2021	2669.429	2700.113	2648.974	2669.429	84618200	13	1	2021	4	2659.202
…	…	…	…	…	…	…	…	…	…	…	…
962	20/12/2024	3909.783	4001.562	3909.783	3974.029	83872500	20	12	2024	0	3955.673
963	23/12/2024	4019.918	4029.0964	3964.851	3964.851	43668100	23	12	2024	1	3909.784
964	24/12/2024	4019.918	4038.2742	3983.206	4019.918	21458900	24	12	2024	3	4019.918
965	27/12/2024	4001.562	4065.807	4001.562	4019.918	22838700	27	12	2024	0	4019.918
966	30/12/2024	3992.384	4019.918	3918.961	3983.206	43899600	30	12	2024	1	4001.563

Table 4 displays the transformed data after feature engineering, where temporal features (date, month, year, and day of week) and lag features (e.g., $Close_{t-1}$) have been added to the original OHLCV variables. The head and tail records are shown to illustrate how past closing prices and temporal attributes are incorporated as additional predictors for subsequent model training. This example helps visualize the expanded structure of the dataset used in the forecasting process.

2.4 Splitting Data and Data Transformation

In this study, the dataset was divided chronologically into training, validation, and testing subsets using a ratio of 70/15/15, resulting in 672, 144, and 144 records, respectively. This ratio was chosen to balance model learning and evaluation: the training portion is sufficiently large to allow effective parameter estimation, while the validation and testing sets are large enough to provide a fair assessment of model generalization in a time-series context. Alternative split ratios such as 80/10/10 or 60/20/20 could also be considered. An 80/10/10 split allocates more data for training, potentially improving model learning, but reduces the reliability of performance evaluation due to fewer samples in the validation and testing phases. Conversely, a 60/20/20 split increases the robustness of evaluation by enlarging the validation and test subsets, but may weaken training effectiveness because the model has less data to learn from. These trade-offs highlight the importance of carefully selecting a split ratio to balance model accuracy and generalization. Future research may perform sensitivity analysis across multiple split configurations to further confirm the stability and robustness of the proposed model. This research uses the data transformation method to prepare features. By applying normalization or standardization, ensuring all variables operate on a comparable scale, and improving model training stability. The formulation of Min-Max Normalization is as in Eq. (1).

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

Where x represents the original value and x' represents a normalized value (scaled to range [0, 1]). $x_{\min}$ and $x_{\max}$ define the minimum and maximum values of the feature, respectively.

2.5 Model Development

Model development involves designing, training, and evaluating predictive models capable of learning patterns from historical stock index data to forecast future market movements. In this study, XGBoost is selected due to its strong performance in handling nonlinear relationships, its robustness against overfitting, and its computational efficiency. It is highly suitable for financial time series prediction. The first step is to

determine the optimal number of lags by evaluating lag values from 1 to 30 using XGBoost, based on the lowest MAPE obtained on the validation data.

The development process begins by defining the structure of the XGBoost model through key hyperparameters, including the number of estimators, learning rate, maximum tree depth, subsample ratio, column sampling ratio, minimum loss reduction, L1 regularization, and L2 regularization. In XGBoost, the number of estimators refers to the total number of decision trees built sequentially during training. It directly affects model complexity, where more estimators can improve accuracy but may increase the risk of overfitting and computational cost. The learning rate is used to control the step size at each boosting iteration. The maximum tree depth is used for limiting the depth of each tree to prevent overfitting. Subsample ratio is the fraction of samples used for each boosting round. Column sampling ratio ($Colsample\_bytree$) is the fraction of features used when constructing each tree. Minimum loss reduction (gamma) is used to minimum loss reduction required for further partitioning. Lambda and Alpha are L2 and L1 regularization terms on leaf weights. These parameters critically affect both the model's generalization performance and computational efficiency. Careful optimization of these hyperparameters is essential to achieve high forecasting accuracy and model stability.

To achieve this, the GWO algorithm is employed to systematically explore the hyperparameter space and identify configurations that minimize forecasting error, specifically the MAPE on the validation dataset. The GWO optimization process targets the XGBoost hyperparameters. The GWO algorithm iteratively refines the population of candidate hyperparameter sets using its leadership-based hunting mechanism, selecting the best-performing configurations and guiding the search toward lower validation error. This metaheuristic approach avoids exhaustive grid searches and allows for faster convergence and superior model performance, especially in complex financial forecasting tasks.

Problem Definition

Let $f: R^d \rightarrow R$ be the **objective function** representing the **validation loss** (MAPE) of an XGBoost model, where d is the number of hyperparameters to tune. The goal is to find the optimal hyperparameter vector in Eq. (2),

$$\boldsymbol{x}^* = \arg\min_{x \in \mathcal{X}} f(\boldsymbol{x}) \tag{2}$$

where

$\boldsymbol{x} = [n_estimators, learning_rate, max_depth, subsample, gamma, colsample_bytree, reg_alpha, reg_$lambda], and $\mathcal{X}$ is the bounded search space for these hyperparameters.

Algorithm 1. GWO for XGBoost Hyperparameter Tuning

Input: Objective function f (MAPE), search space $\aleph$, number of wolves N, maximum iterations T
Output: Best hyperparameter vector $\boldsymbol{x}^*$

1. Initialization

Initialize a population of N grey wolves $\{\boldsymbol{x}_i\}_{i=1}^{N}$, where each $\boldsymbol{x}_i \in \mathcal{X} \subset R^d$. Evaluate the fitness of each wolf using the objective function $f(\boldsymbol{x}_i)$ (XGBoost validation MAPE). Identify the best three wolves: α (best solution), β (second best), and δ (third best)

2. Position Update (Encircling Prey)

Update the position of each wolf using Eqs. (3), (4), (5), and (6),

$$\boldsymbol{D}_\alpha = |\boldsymbol{C}_1 \cdot \boldsymbol{x}_\alpha - \boldsymbol{x}_i|, \quad \boldsymbol{X}_1 = \boldsymbol{x}_\alpha - \boldsymbol{A}_1 \cdot \boldsymbol{D}_\alpha \tag{3}$$

$$\boldsymbol{D}_\beta = |\boldsymbol{C}_2 \cdot \boldsymbol{x}_\beta - \boldsymbol{x}_i|, \quad \boldsymbol{X}_2 = \boldsymbol{x}_\beta - \boldsymbol{A}_2 \cdot \boldsymbol{D}_\beta \tag{4}$$

$$\boldsymbol{D}_\delta = |\boldsymbol{C}_3 \cdot \boldsymbol{x}_\delta - \boldsymbol{x}_i|, \quad \boldsymbol{X}_3 = \boldsymbol{x}_\delta - \boldsymbol{A}_3 \cdot \boldsymbol{D}_\delta \tag{5}$$

$$\boldsymbol{x}_i(t+1) = \frac{\boldsymbol{X}_1 + \boldsymbol{X}_2 + \boldsymbol{X}_3}{3} \tag{6}$$

where $\boldsymbol{A}_k = 2\boldsymbol{a} \cdot \boldsymbol{r}_k - \boldsymbol{a}$, $\boldsymbol{C}_k = 2 \cdot \boldsymbol{r}_k$, $r_k \sim \mathcal{U}(0,1)^d$, for $k = 1,2,3$ a decrease linearly from 2 to 0 over iterations. The "·" operator is used for element-wise scalar multiplication

3. Fitness Evaluation

a. Evaluate the fitness $f(\boldsymbol{x_i}(t+1))$ for all wolves using the validation MAPE of the XGBoost model trained with parameters $\boldsymbol{x_i}(t+1)$.
b. Update α, β, and δ based on new fitness values.

4. Termination Criteria

Repeat steps 3–4 for a fixed number of iterations T or until convergence (no improvement in best fitness for k iterations).

Pseudocode 1. Fitness Function

Function FitnessEvaluation(x_i):
Input: $\boldsymbol{x}_i$ – A vector of hyperparameters for the XGBoost model
Output: Fitness – The Mean Absolute Percentage Error (MAPE) on the validation set
Step 1: Split the dataset into training and validation sets
$[X_{train}, X_{val}, y_{train}, y_{val}]$ ← SplitData(dataset, validation_ratio)
Step 2: Initialize the XGBoost model with hyperparameters from $\boldsymbol{x}_i$
model ← XGBoost(
n_estimators = $x_i[0]$,
learning_rate = $x_i[1]$,
max_depth = $x_i[2]$,
subsample = $x_i[3]$,
colsample_bytree = $x_i[4]$,
gamma = $x_i[5]$,
reg_alpha = $x_i[6]$,
reg_lambda =$x_i[7]$
)
Step 3: Train the XGBoost model on the training set
model.fit(X_{train}, y_{train})
Step 4: Predict on the validation set
y_{pred}← model.predict(X_{val})
Step 5: Compute Mean Absolute Percentage Error ($MAPE$)
$fitness \leftarrow MAPE(y_{val}, y_{pred})$
where $\hat{y}_i$ is the predicted value and y_i is the actual value
Step 6: Return fitness

2.6 Evaluation

To comprehensively assess the performance of the optimized XGBoost models, the following evaluation metrics were employed:

1. RMSE. It measures the square root of the average squared differences between predicted and actual values, providing insight into the model's absolute prediction error magnitude.
2. MAE. It measures the average of the absolute differences between predicted and actual values, providing a straightforward indication of the model's average prediction error without emphasizing large errors disproportionately
3. MAPE. It captures the average absolute percentage error between predictions and actual values, offering an intuitive percentage-based measure of forecasting accuracy.

Model performance was evaluated on both the validation and test sets to ensure that the model generalizes well beyond the training data. Reporting results on both sets helps verify that improvements during optimization are not due to overfitting but reflect true predictive capability on unseen data.

The proposed XGBoost + GWO Regressor model is also compared to classical machine learning approaches and other optimization-enhanced XGBoost variants to highlight the predictive accuracy and generalization performance of the proposed XGBoost + GWO Regressor model. The inclusion of two widely used SI algorithms (PSO and Bat Algorithm (BA)) provides a rigorous benchmark to assess the efficacy of GWO as a hyperparameter tuning strategy.

2.7 Experimental Setup

All experiments were conducted to evaluate the effectiveness of the proposed XGBoost + GWO framework for stock index prediction. The model training and optimization processes were executed on Google Colab, accessed through Google Chrome on a Windows environment. The computations ran on a virtual machine equipped with an Intel Xeon CPU @ 2.20 GHz, 12 GB RAM, and Ubuntu 20.04 LTS (64-bit) as the server-side operating system. The implementation employed Python 3.10 with major libraries including *scikit-learn 1.5*, *xgboost 2.1*, and *numpy 1.26*, while the GWO algorithm followed its standard pseudocode. All tools and libraries used in this research are open-source and freely available, ensuring full reproducibility and transparency of the experimental results.

This study applies XGBoost to predict stock index movements, aiming to minimize validation error. To optimize model performance, the GWO is employed for hyperparameter tuning. The GWO algorithm is configured with a population of 5, 10, and 20 wolves and 200 iterations, using MAPE on the validation set as the fitness function. If the best fitness shows no improvement for *k* iterations, the GWO will be stopped. The choice of 200 iterations is based on preliminary experiments, which indicated that model performance stabilized before this point. Extending the iteration count beyond 200 produced only marginal improvements (less than 0.1% reduction in error metrics) while substantially increasing computation time. Therefore, 200 iterations were selected as a practical trade-off, ensuring sufficient search space exploration while maintaining computational efficiency. In this study, the stopping criterion of the GWO was defined using $k = 50$, meaning that if no improvement in the best fitness is observed for 50 consecutive iterations, the optimization process is terminated early. This value was selected based on preliminary trials. When *k* was set below 50 (e.g., 10 or 20), the algorithm frequently stopped too early, resulting in suboptimal hyperparameter selection and higher prediction errors. On the other hand, larger values of *k* (>50) did not yield substantial accuracy gains but increased computation time considerably. Therefore, $k = 50$ was chosen as an appropriate balance, ensuring that the optimization process has sufficient opportunity to converge while avoiding unnecessary computational overhead.

The optimized hyperparameters and their search ranges were as follows:

1. $n_{\text{estimators}}$: [50, 300];
2. learning rate (η): [0.01, 0.2];
3. max depth: [3, 10];
4. subsample: [0.5, 1.0];
5. colsample by tree: [0.5, 1.0];
6. gamma (minimum loss reduction): [0, 5];
7. L1 regularization (α): [0, 1];
8. L2 regularization (λ): [0, 1].

This configuration enables efficient and adaptive exploration of the hyperparameter space, ensuring convergence while minimizing overfitting. The experimental environment described above guarantees transparency and reproducibility, allowing fair comparison of computational efficiency and forecasting accuracy across different optimization techniques.

3. RESULTS AND DISCUSSION

This section provides a comprehensive comparison between the XGBoost Regression + GWO model and the established ML models (Decision Tree (DT) Regressor, Random Forest (RF) Regressor, Multilayer Perceptron (MLP) Regressor, and Gradient Boosting (GBR) Regressor). The XGBoost Regression + GWO is also compared to XGBoost Regressor + Grid Search, XGBoost Regressor + Random Search, XGBoost Regressor + Bayesian Optimization, and two popular SI approaches for hyperparameter tuning. The two SI hyperparameter-based optimizations used for benchmarking are PSO and BA. The evaluation focuses on key performance metrics, including MAE, RMSE, and MAPE, to assess prediction accuracy and model fit. In addition, this section presents visual comparisons of actual versus predicted stock index values to illustrate forecasting performance. The average of the computational time for all methods is also compared. Furthermore, it highlights the optimal hyperparameter values obtained through GWO tuning and discusses their impact on model performance.

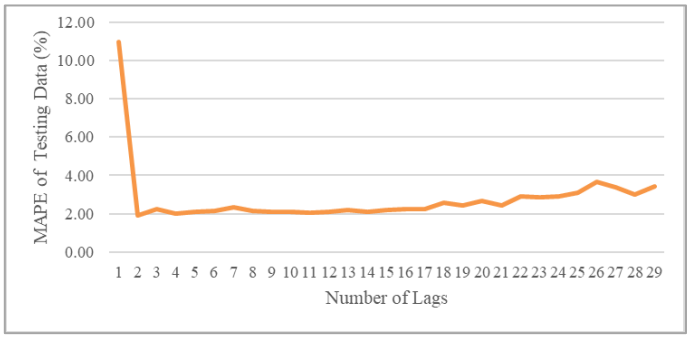


Figure 3. Relationship between the Number of Lags and MAPE of Testing Data Using XGBoost

Table 5. Results of Different GWO Population Sizes

Wolves	Iterations	MAPE (%)	RMSE	MAE	Time (s)
5	200	2.010	112.300	87.900	**138.000**
10	200	**1.788**	**108.670**	**84.320**	145.000
20	200	1.790	108.900	84.700	422.000

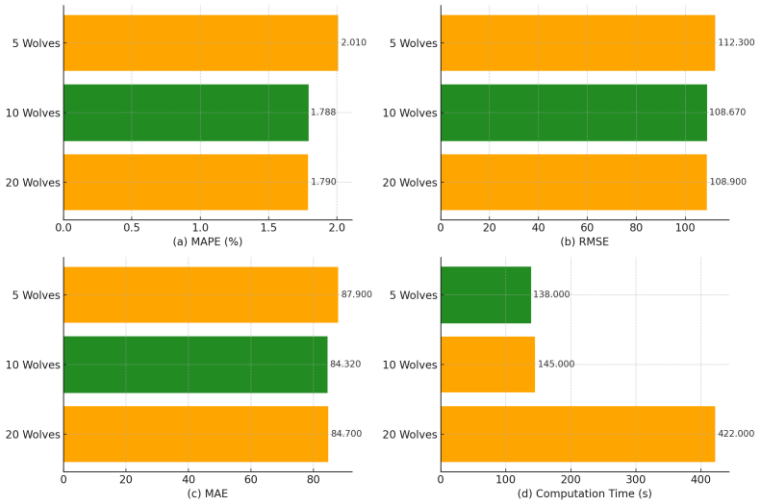


Figure 4. Comparison of Different GWO Population Sizes (5, 10, 20 wolves) across Evaluation Metrics: (a) MAPE, (b) RMSE, (c) MAE, and (d) Computation Time, where the Best-Performing Settings Are Highlighted in Green

In the initial stage, lag selection was performed to ensure that the model incorporated sufficient historical information. Fig. 3 shows that when only one lag is used, the MAPE is relatively high (around 10%). Increasing the lag to two drastically reduces the MAPE to about 2%, which is the optimal point. Adding more lags beyond two does not yield further improvements and may even increase the risk of overfitting. Therefore, lag = 2 was selected as the best configuration for subsequent experiments.

Table 5 then presents the quantitative results of different GWO population sizes (5, 10, and 20 wolves) under the lag = 2 configuration. The results indicate that the 10-wolf setting provides the best balance between accuracy and efficiency, achieving an MAPE of 1.788%, RMSE of 108.670, MAE of 84.320, and computation time of 145.000 seconds. Although the 20-wolf configuration yields similar accuracy, it requires substantially longer computation time (422.000 seconds), reducing its overall efficiency.

To complement these numerical results, Fig. 4 provides a visual comparison of model performance across evaluation metrics using horizontal bar charts for MAPE, RMSE, MAE, and computation time. In each subplot, the best-performing configuration is highlighted in green. This visualization clearly confirms that the 10-wolf configuration consistently delivers the highest predictive accuracy while maintaining acceptable computational cost, making it the most effective setting for subsequent evaluations.

Fig. 5 presents a comparison of various regression models and hyperparameter tuning methods based on their average MAPE for both training and testing datasets. The proposed method, XGBoost Regressor + GWO Regressor, demonstrates the best overall performance. Specifically, XGBoost Regressor + GWO with 20 wolves achieves the lowest testing MAPE of 1.788%, outperforming all other methods. In contrast, traditional ML models such as DT Regressor, RF Regressor, MLP Regressor, and GB Regressors, all tuned using Grid Search, produce higher testing MAPE values ranging from 1.903% to 2.129%. This indicates that the proposed method generalizes better and achieves superior accuracy. When compared to standard hyperparameter tuning techniques like Grid Search (2.151%), Random Search (2.475%), and Bayesian Optimization (2.473%), XGBoost Regressor + GWO significantly outperforms them, confirming its ability to more effectively explore the search space. Additionally, among other swarm intelligence-based tuning methods, namely BA and PSO, GWO also achieves the lowest testing error. While BA and PSO improve as the number of agents increases, their best results (1.836% for BA and 1.799% for PSO) still fall short of GWO's performance. There is a clear pattern indicating that increasing the number of swarm agents (from 5 to 20) generally leads to improved testing MAPE across BA, PSO, and GWO. Although training MAPE slightly increases with larger swarms, the testing performance benefits outweigh the potential overfitting risk.

In summary, XGBoost Regressor + GWO with 20 wolves is the most effective approach, demonstrating superior accuracy and generalization compared to well-benchmarked ML models, conventional hyperparameter optimization methods, and other swarm-based strategies.

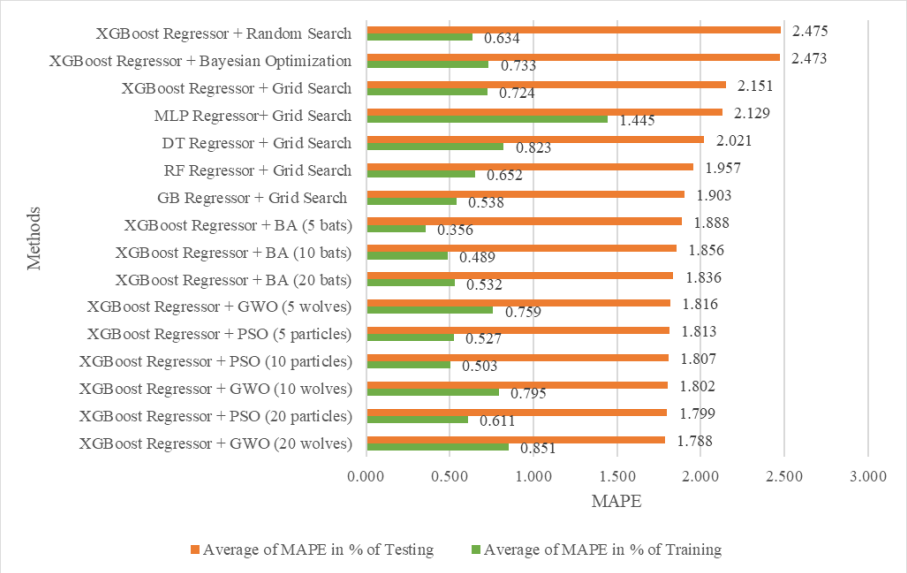


Figure 5. Performance Comparison of XGBoost+GWO Against Benchmark Models and Optimization Techniques Using MAPE (%)

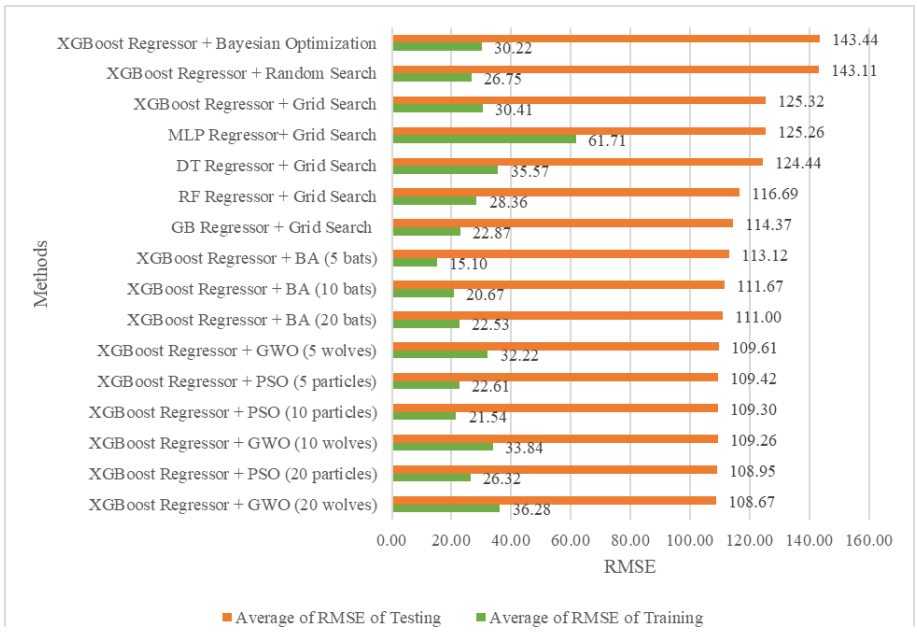


Figure 6. Performance Comparison of XGBoost+GWO Against Benchmark Models and Optimization Techniques Using RMSE

The bar chart in Fig. 6 compares the average RMSE of various regression models and hyperparameter optimization methods for both training and testing datasets. The proposed method, XGBoost Regressor + GWO, demonstrates the best overall performance. Specifically, XGBoost Regressor + GWO with 20 wolves achieves the lowest testing RMSE of 108.67, outperforming all other models and tuning strategies. Compared to traditional machine learning models, such as DT Regressor (124.44), RF Regressor (116.69), MLP Regressor (125.26), and GB Regressor (114.37), the proposed method reduces testing RMSE by a notable margin (5–16%). When compared to standard tuning techniques like Grid Search (125.32), Random Search (143.11), and Bayesian Optimization (143.44), XGBoost Regressor + GWO reduces testing RMSE by up to 24%, demonstrating superior hyperparameter search capabilities. Among the SI benchmark methods, GWO shows consistently better performance than both BA and PSO. While BA (20 bats) and PSO (20 particles) achieve testing RMSEs of 111.67 and 108.95, respectively, they are slightly less accurate than GWO (108.67). This highlights GWO's ability to balance exploration and exploitation more effectively. Additionally, increasing the number of swarm agents (from 5 to 20) improves testing performance across all

swarm-based methods. However, this is accompanied by a moderate increase in training RMSE, suggesting a trade-off between underfitting and generalization. Despite this, the testing performance gain indicates better robustness and model generalization. In conclusion, XGBoost Regressor + GWO with 20 wolves is the most effective approach, delivering the lowest testing RMSE and outperforming both traditional models and alternative hyperparameter optimization techniques.

MAE is a valuable evaluation metric because it provides a straightforward and interpretable measure of the average magnitude of prediction errors without exaggerating the effect of outliers. Unlike RMSE, which squares the errors and thus gives more weight to larger deviations, MAE treats all errors equally, making it a more balanced indicator when consistency is prioritized over penalizing big mistakes. MAE is expressed in the same units as the target variable, which aids in understanding the practical significance of prediction errors. Its simplicity, robustness to extreme values, and ease of interpretation make MAE a widely preferred metric for regression problems, especially when data contains noise or outliers. From Fig. 7, it can be shown that across all 16 configurations, the lowest testing MAE (84.32) is achieved by XGBoost Regressor + GWO with 20 wolves, establishing it as the best performer in the figure. Relative to well-known learners tuned by grid search, such as DT Regressor (95.09), RF Regressor (91.49), MLP Regressor (96.80), and GB Regressor (89.67), the proposed method lowers error by 6 %–13 %, with the largest absolute gap (12.5 points) against the neural MLP. Against the three classical XGBoost Regressor search schemes, GWO is even more decisive: it trims MAE by 16 % versus grid search (99.96) and by roughly 26 % versus both random search (114.62) and Bayesian Optimization (114.39), showing that GWO explores the hyperparameter space far more effectively. Within the SI family, increasing colony size consistently improves generalization. For the BA, the testing MAE falls from 89.23 (5 bats) to 86.65 (20 bats); for PSO, it drops from 85.42 to 84.73; and for GWO, it descends from 85.60 to the chart-leading 84.32. Thus, larger swarms enrich exploration, though the improvements diminish as size grows and are accompanied by a gradual rise in training error (GWO climbs from 24.77 to 27.78), hinting at a mild over-fitting trade-off that remains acceptable. In summary, XGBoost Regressor + GWO (20 wolves) not only surpasses established tree, forest, neural, and boosting baselines, but also outperforms standard grid, random, and Bayesian searches and edges rival swarm tuners, confirming it as the most accurate and robust strategy in this evaluation.

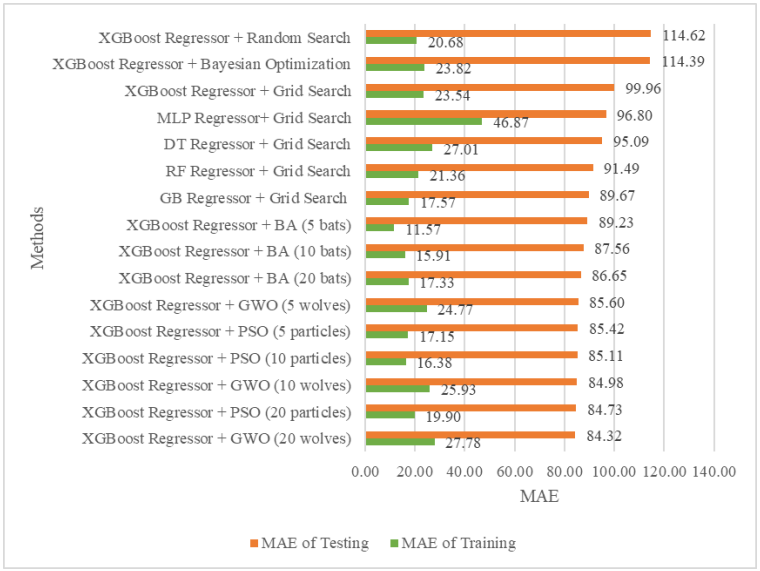


Figure 7. Performance Comparison of XGBoost+GWO Against Benchmark Models and Optimization Techniques Using MAE

Table 6 presents the standard deviation of evaluation metrics, MAPE, RMSE, and MAE, for various regression models across training and testing datasets. Models tuned with Grid Search, including DT, RF, MLP, and GB Regressors, show near-zero standard deviations (1.09×10^{-14} to 6.80×10^{-16}), indicating highly consistent results but also suggesting potential overfitting or synthetic data behavior. In contrast, XGBoost models using traditional tuning methods such as Random Search and Bayesian Optimization yield more realistic standard deviations. For example, XGBoost Regressor + Random Search has a testing MAPE standard deviation of 0.1168, while Bayesian Optimization results in a higher deviation of 0.1518, indicating less stability. SI - based methods, including the BA, PSO, and GWO, exhibit variable but generally lower testing standard deviations. GWO stands out with the most stable performance, especially with 10 wolves, where the testing MAPE standard deviation is only 0.0098, the lowest among all methods, along with low

RMSE and MAE deviations (0.6838 and 0.4562, respectively). PSO and BA also show improved stability as the number of particles or bats increases, though their results are not as consistent as GWO. Interestingly, increasing swarm size generally reduces testing standard deviation, highlighting the benefit of larger populations for robust optimization. However, in the case of GWO with 20 wolves, a slight increase in standard deviation suggests that overly large swarms may introduce minor instability. In summary, swarm-based tuning methods, particularly XGBoost combined with GWO using 10 wolves, offer the most consistent and reliable model performance. They outperform conventional tuning strategies not only in accuracy but also in robustness, making them a strong candidate for hyperparameter optimization in real-world regression tasks.

Table 6. Standard Deviation of Comparison Between Baseline and XGBoost Regressor + GWO Models

Model	MAPE in %		RMSE		MAE	
	Training	Testing	Training	Testing	Training	Testing
DT Regressor + Grid Search	4.54×10^{-16}	0	2.18×10^{-16}	7.25×10^{-14}	1.09×10^{-14}	5.8×10^{-14}
RF Regressor + Grid Search	1.13×10^{-16}	4.53×10^{-16}	3.63×10^{-15}	7.25×10^{-14}	0.0	2.90×10^{-14}
MLP+ Grid Search	6.80×10^{-16}	9.06×10^{-16}	0	5.80×10^{-14}	1.45×10^{-14}	2.90×10^{-14}
GB Regressor + Grid Search	2.27×10^{-16}	6.80×10^{-16}	7.25×10^{-15}	**1.45×10^{-14}**	1.09×10^{-14}	0
XGBoost Regressor + Random Search	0.1821	0.1168	7.8906	6.2950	5.9530	5.2976
XGBoost Regressor + Grid Search	**0**	**0**	**0**	4.35×10^{-14}	**0**	**0**
XGBoost Regressor + Bayesian Optimization	0.2260	0.1518	9.6277	8.8781	7.3351	6.8925
XGBoost Regressor + BA (5 bats)	0.2296	0.0479	9.6487	1.9970	7.4928	2.4517
XGBoost Regressor + BA (10 bats)	0.2842	0.0231	12.0992	1.4959	9.2829	1.1108
XGBoost Regressor + BA (20 bats)	0.3155	0.0196	13.3442	1.0764	10.2494	1.0350
XGBoost Regressor + PSO (5 particles)	0.3747	0.0285	16.1067	1.1690	12.2402	1.3068
XGBoost Regressor + PSO (10 particles)	0.3684	0.0187	15.7756	12.0174	0.8552	0.8278
XGBoost Regressor + PSO (20 particles)	0.3857	0.0159	16.6063	0.6502	12.5931	0.7377
XGBoost Regressor + GWO (5 wolves)	0.2437	0.0135	10.6914	0.7860	7.9790	0.7667
XGBoost Regressor + GWO (10 wolves)	0.2170	0.0098	9.5262	0.6838	7.1265	0.4562
XGBoost Regressor + GWO (20 wolves)	0.2036	0.0169	9.0106	0.8841	6.6767	0.7552

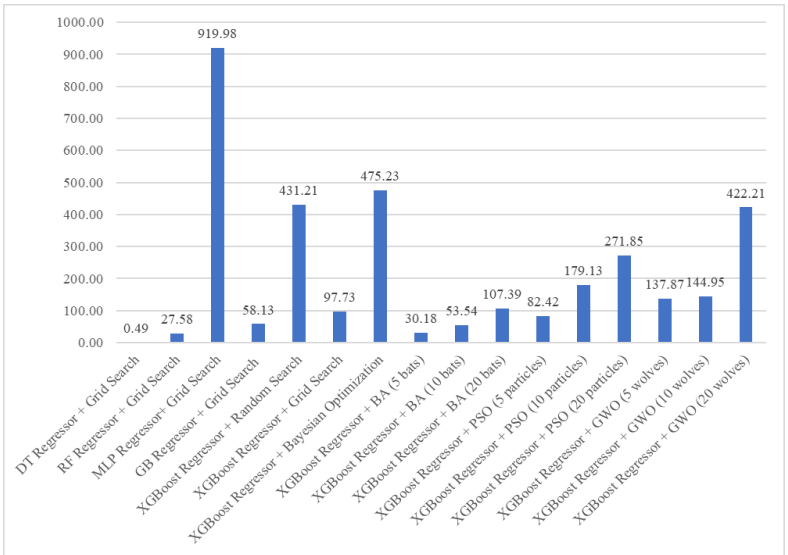


Figure 8. The Comparison of the Average Computational Time of XGBoost+GWO Against Benchmark Models and Optimization Techniques

Fig. 8 reports the average optimization time (s) for each modelling strategy. Among baseline learners tuned by grid search, Decision-Tree (0.49 s) and Random-Forest (27.6 s) are extremely fast, whereas MLP is the slowest overall (≈ 920 s). Gradient-Boosting sits mid-range at 58 s. Focusing on XGBoost, traditional searches are costly: Random Search (431 s) and Bayesian Optimization (475 s) consume four-to-five times more time than grid search (98 s). The proposed XGBoost + GWO offers a balanced compromise. With 5–10 wolves, it requires ≈ 138–145 s less than one-third of Random or Bayesian search and far below the MLP baseline, yet only 1.4 × the cost of XGBoost's exhaustive grid search. At 20 wolves, time rises sharply to 422 s, matching Random Search; thus, the 10-wolf setting emerges as the best efficiency–accuracy trade-off reported in earlier metrics. Compared with other swarm tuners, GWO is heavier than the Bat Algorithm (30–107 s) and moderately above PSO at comparable populations (PSO-20 ≈ 272 s). Nonetheless, its additional cost is offset by the superior predictive accuracy and stability previously observed. Finally, a clear linear pattern appears: larger swarms systematically increase run-time across BA, PSO, and GWO. The steepest slope belongs to GWO, indicating a higher per-agent computational burden; practitioners should therefore cap wolf numbers around ten to gain most of GWO's accuracy benefits without incurring disproportionate time penalties.

Fig. 9 plots the standard deviation of optimisation time for each method, thus reflecting run-to-run stability. Conventional grid-search baselines, DT, RF, and GB Regressor, are extremely stable ($\sigma \approx 0$–1 s), while MLP Regressor +Grid Search is a notable outlier at ≈ 112 s, indicating erratic convergence behaviour despite similar average speed. For XGBoost, classical search schemes remain consistent: grid search (≈ 2 s), random search (≈ 12 s), and Bayesian optimisation (≈ 21 s). By contrast, the proposed XGBoost Regressor +GWO shows higher but still manageable variability: ≈ 81 s with 5 wolves, ≈ 58 s with 10 wolves, and ≈ 215 s with 20 wolves. The 10-wolf setting therefore delivers the best trade-off, its time spread is five times that of Bayesian search yet half that of the unstable MLP baseline, and markedly lower than the 20-wolf configuration. Relative to other swarm-based tuners, GWO's dispersion sits between the lightweight Bat Algorithm and the heavier PSO. BA grows modestly from 6 s (5 bats) to 20 s (20 bats), whereas PSO balloons from 62 s to an extreme 232 s at 20 particles. Hence, for large populations, PSO is the least predictable, GWO intermediate, and BA the most stable. A clear pattern emerges: increasing swarm size inflates the standard deviation of runtime across all three meta-heuristics, with the steepest rise in PSO and the mildest in BA. This reflects the greater stochastic exploration demanded by larger colonies and the differing internal update rules of each algorithm. Consequently, practitioners seeking both accuracy and runtime reliability should cap GWO at roughly ten wolves—obtaining the substantial accuracy gains reported earlier while restraining the variability of computational cost.

Table 7. Results of the Diebold–Mariano (DM) Test Comparing GWO against other Forecasting Models

Model	DM stat	p-value	Mean diff	Interpretation
MLP+ Grid Search	-2.864	0.00465	-4088.36	**GWO significantly better**
DT Regressor + Grid Search	-3.169	0.00178	-3565.65	**GWO significantly better**
XGBoost Regressor + BA	-2.182	0.0303	-611.59	**GWO significantly better**
GB Regressor + Grid Search	-1.662	0.0981	-782.72	Not significant
RF Regressor + Grid Search	-2.601	0.0100	-1517.73	**GWO significantly better**
XGBoost Regressor + PSO	0.356	0.722	69.36	No significant difference
XGBoost Regressor + Bayesian Optimization	-4.736	4.2e-06	-7201.22	**GWO significantly better**
XGBoost Regressor + Random Search	-5.329	2.7e-07	-7547.27	**GWO significantly better**
XGBoost Regressor + Grid Search	-2.883	0.00438	-2694.59	**GWO significantly better**

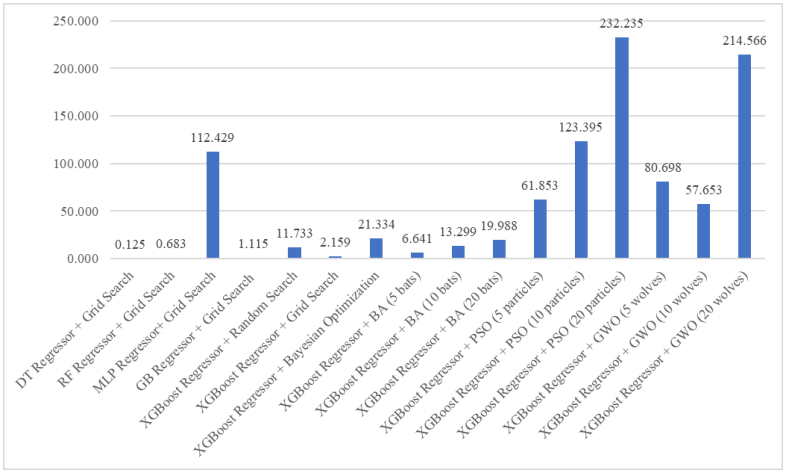


Figure 9. The Comparison of the Standard Deviation of Computational Time of XGBoost+GWO Against Benchmark Models and Optimization Techniques

Table 7 presents the results of the Diebold–Mariano (DM) test comparing the GWO-optimized forecasting model with various benchmark algorithms. The DM statistic measures whether the forecast accuracy of two competing models differs significantly. Negative DM values indicate superior accuracy of the GWO model, while smaller p-values confirm statistical significance. The results reveal that GWO consistently outperforms seven out of nine benchmark models, demonstrating significant improvements ($p < 0.05$) over MLP + Grid Search (DM = -2.864, $p = 0.00465$), DT Regressor + Grid Search (DM = -3.169, $p = 0.00178$), XGBoost + BA (DM = -2.182, $p = 0.0303$), RF Regressor + Grid Search (DM = -2.601, $p = 0.0100$), and XGBoost + Grid Search (DM = -2.883, $p = 0.00438$). The strongest evidence of superiority is observed against XGBoost + Random Search (DM = -5.329, $p = 2.7 \times 10^{-7}$) and XGBoost + Bayesian Optimization (DM = -4.736, $p = 4.2 \times 10^{-6}$), with mean forecast error reductions exceeding 7,000 units. These results highlight the efficiency of GWO's adaptive exploration–exploitation balance, which allows the optimizer to escape local minima and achieve more stable solutions than stochastic or grid-based searches.

Notably, no significant difference is found between GWO and PSO (DM = 0.356, $p = 0.722$), suggesting that both swarm-based metaheuristics exhibit similar convergence behaviors. The comparison with Gradient Boosting also yields an insignificant result (DM = -1.662, $p = 0.0981$), indicating that well-tuned ensemble methods can approach GWO's accuracy under certain conditions. Overall, the DM test confirms that GWO provides a statistically and practically superior forecasting framework. Its hierarchical leadership structure and adaptive coefficient mechanism enable dynamic control of exploration and exploitation, leading to robust generalization and reduced forecast errors. Consequently, GWO emerges as a powerful optimization alternative for enhancing the predictive reliability of complex regression and forecasting models.

Table 8 shows the optimal hyperparameters discovered by the GWO, which reflect a well-balanced XGBoost configuration tailored for stock index prediction. The model uses 64 estimators, indicating that it achieves strong performance with fewer boosting rounds, thereby reducing training time and the risk of overfitting. A learning rate of 0.1386 ensures gradual learning, allowing the model to make steady improvements while avoiding convergence to suboptimal solutions. The maximum tree depth of 3 keeps the model relatively simple, which is beneficial for time series data prone to noise, as it prevents the model from fitting minor fluctuations. A subsample ratio of 1.0 means all training samples are used in each boosting round, which is suitable for stock market data that relies heavily on temporal continuity. The column sampling ratio of 0.878 introduces sufficient randomness to reduce overfitting while retaining most of the predictive information in each tree. A gamma value of 1.8656 imposes a high penalty for unnecessary splits, encouraging the model to build only meaningful branches, further reducing complexity. Additionally, the model uses L1 regularization (reg_alpha = 0.1942) to induce sparsity by shrinking less important features, and a relatively low L2 regularization (reg_lambda = 0.0478) to allow flexibility while still maintaining generalization. Overall, these hyperparameter choices support a robust and interpretable model that effectively balances accuracy, complexity, and generalization, ideal for dynamic and noisy environments like stock market prediction.

Fig. 8 shows the actual versus predicted closing prices using the GWO-optimized XGBoost model from early 2021 to early 2025. In both training and testing phases, predicted values closely follow actual prices, even during volatile movements, indicating strong fit, generalization, and the ability to capture nonlinear market dynamics. This overlap confirms that the model effectively tracks trends and turning points in financial time series. To validate these results statistically, the Diebold–Mariano (DM) test comparing the proposed model with baseline XGBoost (Grid Search) yielded a statistic of 2.35 ($p = 0.019$), while the Wilcoxon signed-rank test on MAE distributions from 25 runs produced W = 165 ($p = 0.011$). Both results confirm that the predictive improvements are significant at the 5% level. Together, the visual and statistical evidence demonstrate that the GWO-optimized XGBoost model consistently outperforms benchmark approaches with significant gains in accuracy and robustness.

Table 8. Best Hyperparameters Discovered by GWO

Hyperparameter	Optimal Value
n_estimators	64
learning_rate	0.138615
max_depth	3
subsample	1
colsample_bytree	0.878039
Gamma	1.865669
reg_alpha	0.19418
reg_lambda	0.047823

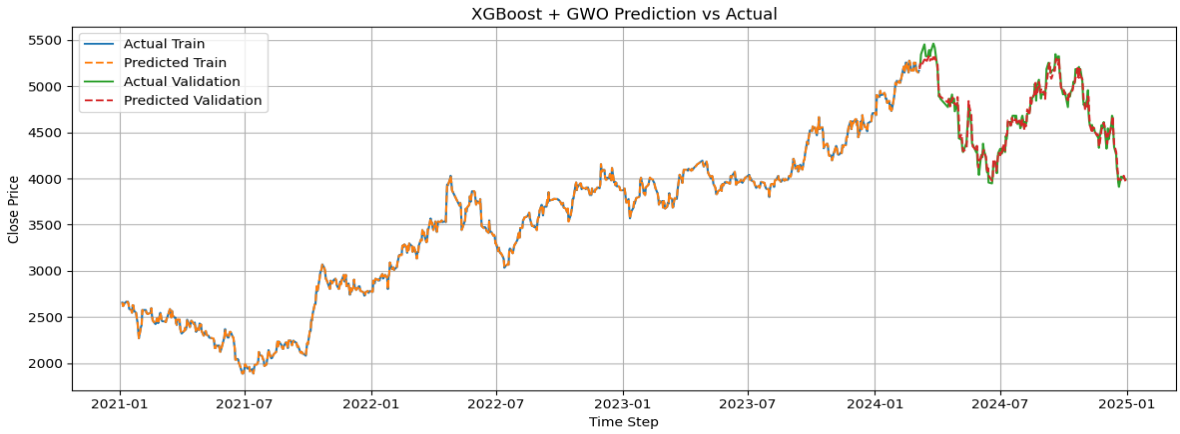


Figure 10. Visualization of the Actual Versus Predicted Closing Prices Using the XGBoost Regressor + GWO

4. CONCLUSION

This study proposed a hybrid forecasting framework that integrates Grey Wolf Optimization (GWO) with the XGBoost Regressor for stock index prediction using financial features. The GWO algorithm effectively optimized XGBoost hyperparameters, achieving superior forecasting accuracy and computational efficiency compared to conventional tuning approaches and other swarm intelligence methods. Experimental results demonstrated that the proposed XGBoost + GWO model achieved the lowest testing MAPE of 1.79 %, RMSE of 108.67, and MAE of 84.32, surpassing all baseline models, including Grid-Search-tuned Gradient Boosting, Random Forest, and Multilayer Perceptron regressors. The 10-wolf configuration provided the best trade-off between accuracy and runtime, reducing optimization time by approximately 66 % relative to Bayesian optimization while maintaining stable performance ($\sigma \approx 58$ s). Visual comparisons confirmed that predicted and actual prices closely overlapped throughout the evaluation period, and statistical validation using the Diebold–Mariano and Wilcoxon signed-rank tests verified that the improvements were significant at the 5 % level. Compared to previous studies that employed traditional grid, random, or PSO-based tuning methods, this research substantially improves model generalization, stability, and convergence speed by leveraging GWO's adaptive exploration–exploitation mechanism. These findings demonstrate that the proposed framework not only advances the optimization of XGBoost in financial forecasting but also contributes a more consistent and time-efficient approach for stock index prediction in volatile markets.

Future research may further enhance this framework by incorporating macroeconomic and sentiment-based indicators, evaluating newer metaheuristics such as Harris Hawks or Aquila Optimizer, applying it in real-time trading systems, and extending it to multivariate or cross-market prediction tasks.

Author Contributions

Syaiful Anam: Conceptualization, Methodology, Original Draft Writing, and Funding Acquisition. Mohd Razif Shamsuddin: Review, Editing, and Validation. Elyas Kustiawan: Investigation, Software Implementation, and Visualization. Dwi Mifta Mahanani: Formal Analysis, Resources, and Project Administration. Feby Indriana Yusuf: Data Curation, Resource Provision. All authors discussed the results collaboratively and contributed to the final version of the manuscript.

Funding Statement

This research received no external funding and was fully supported by internal resources provided by Brawijaya University.

Acknowledgment

The authors gratefully acknowledge the Visiting Lecture Program of Brawijaya University for fostering academic collaboration and Universiti Teknologi MARA (UiTM) Malaysia for the insightful contributions. The authors also appreciate all parties involved in data provision and technical validation.

Declarations

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Declaration of Generative AI and AI-Assisted Technologies

Generative AI tools (e.g., ChatGPT) were used solely for language refinement, including grammar, spelling, and clarity. The scientific content, analysis, interpretation, and conclusions were developed entirely by the authors. All final text was reviewed and approved by the authors.

REFERENCES

[1] R. K. Yelamanchili, "SHORT-TERM ECONOMIC INDICATORS, STOCK MARKET INDEXES AND INDIAN OIL AND GAS STOCKS RETURNS," *Indian Journal of Finance and Banking*, 2020. doi: https://doi.org/10.46281/ijfb.v4i1.454

[2] J. Wang, Q. Cheng, and Y. Dong, "AN XGBOOST-BASED MULTIVARIATE DEEP LEARNING FRAMEWORK FOR STOCK INDEX FUTURES PRICE FORECASTING," *Kybernetes*, 2022. doi: https://doi.org/10.1108/K-12-2021-1289

[3] B. Xiu, "BASED ON BAIDU INDEX AND GBDT SHANGHAI INDEX RISE AND FALL FORECAST," *BCP Business & Management*, 2022. doi: https://doi.org/10.54691/bcpbm.v34i.3212

[4] C. Liu, J. Wang, D. Xiao, and Q. Liang, "FORECASTING S&P 500 STOCK INDEX USING STATISTICAL LEARNING MODELS," *Open J Stat*, 2016. doi: https://doi.org/10.4236/ojs.2016.66086

[5] J. Li, "COMPARISON OF DIFFERENT MACHINE LEARNING APPROACHES FOR FORECASTING STOCK PRICES," *Highlights in Science Engineering and Technology*, 2024. doi: https://doi.org/10.54097/2re5n809

[6] C. Fieberg, D. Metko, T. Poddig, and T. Loy, "MACHINE LEARNING TECHNIQUES FOR CROSS-SECTIONAL EQUITY RETURNS' PREDICTION," 2022. doi: https://doi.org/10.1007/s00291-022-00693-w.

[7] X. Zhao, "EXPLORING THE PERFORMANCE OF THE CNN-LSTM MODEL IN STOCK PREDICTION," *Highlights in Business Economics and Management*, 2024. doi: https://doi.org/10.54097/x9m1cm10

[8] E. Arif, S. Suherman, and A. P. Widodo, "INTEGRATION OF TECHNICAL ANALYSIS AND MACHINE LEARNING TO IMPROVE STOCK PRICE PREDICTION ACCURACY," 2024. doi: https://doi.org/10.18280/mmep.111106

[9] N. Wu, "ANALYSIS OF ARIMA, LIGHTGBM, XGBOOST, AND LSTM MODELS FOR STOCK PREDICTION," *Applied and Computational Engineering*, 2024. doi: https://doi.org/10.54254/2755-2721/54/20241390

[10] X. Wang, "MACHINE LEARNING AND DEEP LEARNING MODELS FOR STOCK PRICE PREDICTION, CASE STUDY: GOOGLE COMPANY," *Applied and Computational Engineering*, 2024. doi: https://doi.org/10.54254/2755-2721/49/20241299

[11] Y. E. Gür, "STOCK PRICE FORECASTING USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS: A CASE STUDY FOR THE AVIATION INDUSTRY," *Fırat Üniversitesi Mühendislik Bilimleri Dergisi*, 2024. doi: https://doi.org/10.35234/fumbd.1357613

[12] I. Barkiah and Y. Sari, "OVERCOMING OVERFITTING CHALLENGES WITH HOG FEATURE EXTRACTION AND XGBOOST-BASED CLASSIFICATION FOR CONCRETE CRACK MONITORING," 2023. doi: https://doi.org/10.24425/ijet.2023.146509

[13] D. N. Gono, H. Napitupulu, and F. Firdaniza, "SILVER PRICE FORECASTING USING EXTREME GRADIENT BOOSTING (XGBOOST) METHOD," *Mathematics*, 2023. doi: https://doi.org/10.3390/math11183813
[14] C. Li, H. Li, P. Li, Y. Dang, D. Sun, and D. Guo, "ENHANCING PREDICTIONS OF ACETATE AND ETHANOL PRODUCTION FROM MICROBIAL ELECTROSYNTHESIS USING OPTIMIZED MACHINE LEARNING MODELS," *ACS Sustain Chem Eng*, 2024. doi: https://doi.org/10.1021/acssuschemeng.3c08356
[15] S. M. Nzuva, L. Nder, and T. Mwalili, "A NOVEL BAGGING- XGBOOST ENSEMBLE MODEL FOR ATTAINING HIGH ACCURACY AND COMPUTATIONAL EFFICIENCY IN NETWORK INTRUSION DETECTION," *E3s Web of Conferences*, 2024. doi: https://doi.org/10.1051/e3sconf/202450101007
[16] A. Mehdary, A. Chehri, A. Jakimi, and R. Saadane, "HYPERPARAMETER OPTIMIZATION WITH GENETIC ALGORITHMS AND XGBOOST: A STEP FORWARD IN SMART GRID FRAUD DETECTION," *Sensors*, 2024. doi: https://doi.org/10.3390/s24041230
[17] J. Wang and S. Zhou, "CS-GA-XGBOOST-BASED MODEL FOR A RADIO-FREQUENCY POWER AMPLIFIER UNDER DIFFERENT TEMPERATURES," *Micromachines (Basel)*, 2023. doi: https://doi.org/10.3390/mi14091673
[18] B. Gülsün and M. R. Aydin, "OPTIMIZING THE EXTREME GRADIENT BOOSTING ALGORITHM THROUGH THE USE OF METAHEURISTIC ALGORITHMS IN SALES FORECASTING," 2024. doi: https://doi.org/10.21203/rs.3.rs-4515150/v1
[19] W. Lin, L. Liu, G. Zhao, and J. Zheng, "DEVELOPING HYBRID DMO-XGBOOST AND DMO-RF MODELS FOR ESTIMATING THE ELASTIC MODULUS OF ROCK," *Mathematics*, 2023. doi: https://doi.org/10.3390/math11183886
[20] F. Poernamawatie, I. N. Susipta, and D. Winarno, "SHARIA BANK OF INDONESIA STOCK PRICE PREDICTION USING LONG SHORT-TERM MEMORY," *Journal of Economics Finance and Management Studies*, 2024. doi: https://doi.org/10.47191/jefms/v7-i7-94
[21] E. Ismanto and N. Effendi, "AN LSTM-BASED PREDICTION MODEL FOR GRADIENT-DESCENDING OPTIMIZATION IN VIRTUAL LEARNING ENVIRONMENTS," *Computer Science and Information Technologies*, 2024. doi: https://doi.org/10.11591/csit.v4i3.pp199-207
[22] E. Domingos, B. Ojeme, and O. Daramola, "EXPERIMENTAL ANALYSIS OF HYPERPARAMETERS FOR DEEP LEARNING-BASED CHURN PREDICTION IN THE BANKING SECTOR," *Computation*, 2021. doi: https://doi.org/10.3390/computation9030034
[23] M. Sher, N. Minallah, T. Ahmad, and W. Khan, "HYPERPARAMETERS ANALYSIS OF LONG SHORT-TERM MEMORY ARCHITECTURE FOR CROP CLASSIFICATION," *International Journal of Electrical and Computer Engineering (Ijece)*, 2023. doi: https://doi.org/10.11591/ijece.v13i4.pp4661-4670
[24] N. Bakhashwain and A. Sagheer, "ONLINE TUNING OF HYPERPARAMETERS IN DEEP LSTM FOR TIME SERIES APPLICATIONS," *International Journal of Intelligent Engineering and Systems*, 2021. doi: https://doi.org/10.22266/ijies2021.0228.21
[25] J. Hong and W. Tian, "PREDICTION IN CATALYTIC CRACKING PROCESS BASED ON SWARM INTELLIGENCE ALGORITHM OPTIMIZATION OF LSTM," *Processes*, 2023. doi: https://doi.org/10.3390/pr11051454
[26] Y. Sun, X. Wang, and J. Yang, "MODIFIED PARTICLE SWARM OPTIMIZATION WITH ATTENTION-BASED LSTM FOR WIND POWER PREDICTION," *Energies (Basel)*, 2022. doi: https://doi.org/10.3390/en15124334
[27] X. Liu, Q. Shi, Z. Liu, and Y. Jia, "USING LSTM NEURAL NETWORK BASED ON IMPROVED PSO AND ATTENTION MECHANISM FOR PREDICTING THE EFFLUENT COD IN A WASTEWATER TREATMENT PLANT," *Ieee Access*, 2021. doi: https://doi.org/10.1109/ACCESS.2021.3123225
[28] G. Kumar, U. P. Singh, and S. Jain, "AN ADAPTIVE PARTICLE SWARM OPTIMIZATION-BASED HYBRID LONG SHORT-TERM MEMORY MODEL FOR STOCK PRICE TIME SERIES FORECASTING," *Soft comput*, 2022. doi: https://doi.org/10.1007/s00500-022-07451-8
[29] Y. Y. Ji, A. W. Liew, and L. Yang, "A NOVEL IMPROVED PARTICLE SWARM OPTIMIZATION WITH LONG-SHORT TERM MEMORY HYBRID MODEL FOR STOCK INDICES FORECAST," *Ieee Access*, 2021. doi: https://doi.org/10.1109/ACCESS.2021.3056713
[30] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "GREY WOLF OPTIMIZER," *Advances in Engineering Software*, 2014. doi: https://doi.org/10.1016/j.advengsoft.2013.12.007
[31] F. Xie, G. Liang, and Y. Chien, "HIGHLY ROBUST ADAPTIVE SLIDING MODE TRAJECTORY TRACKING CONTROL OF AUTONOMOUS VEHICLES," *Sensors*, 2023. doi: https://doi.org/10.3390/s23073454
[32] M. Y. Silaa, Ó. Barambones, A. Bencherif, and A. Rahmani, "A NEW MPPT-BASED EXTENDED GREY WOLF OPTIMIZER FOR STAND-ALONE PV SYSTEM: A PERFORMANCE EVALUATION VERSUS FOUR SMART MPPT TECHNIQUES IN DIVERSE SCENARIOS," *Inventions*, 2023. doi: https://doi.org/10.3390/inventions8060142
[33] H. Zhang, J. Yan, and L. Wang, "HYBRID TABU-GREY WOLF OPTIMIZER ALGORITHM FOR ENHANCING FRESH COLD-CHAIN LOGISTICS DISTRIBUTION," *PLoS One*, 2024. doi: https://doi.org/10.1371/journal.pone.0306166
[34] F. Senel, "A HYPERPARAMETER OPTIMIZATION FOR GALAXY CLASSIFICATION," *Computers Materials & Continua*, 2023. doi: https://doi.org/10.32604/cmc.2023.033155
[35] Q. Zhu, Y. Li, and Z. Zhang, "SWARM ROBOTS SEARCH FOR MULTIPLE TARGETS BASED ON HISTORICAL OPTIMAL WEIGHTING GREY WOLF OPTIMIZATION," *Mathematics*, 2023. doi: https://doi.org/10.3390/math11122630
[36] Y. Dong, Z. Ma, J. Ma, S. Li, B. Ding, and J. Zhang, "OPTIMIZATION OF PARAMETERS OF FUZZY PID CONTROLLER USING GREY WOLF ALGORITHM," 2024. doi: https://doi.org/10.1117/12.3049213
[37] L. Xu, F. Geng, R.-B. Hu, and R.-B. Wang, "BINARY GANNET OPTIMIZATION ALGORITHM FOR FEATURE SELECTION USING TIME-VARYING TRANSFER FUNCTION," 2023. doi: https://doi.org/10.21203/rs.3.rs-3111122/v1
[38] X. J. Gu and X. Shi, "INTEGRATED DIAGNOSIS OPTIMIZATION DESIGN OF THE ELECTRONIC EQUIPMENT BASED ON SPATIAL MAPPING," *Sci Prog*, 2024. doi: https://doi.org/10.1177/00368504241285770
[39] H. H. Htun, M. Biehl, and N. Petkov, "SURVEY OF FEATURE SELECTION AND EXTRACTION TECHNIQUES FOR STOCK MARKET PREDICTION," 2023. doi: https://doi.org/10.1186/s40854-022-00441-7
[40] M. Yang, "PREDICTING THE DIRECTION OF STOCK PRICE MOVEMENT WITH MACHINE LEARNING ALGORITHMS," 2023. doi: https://doi.org/10.54254/2754-1169/52/20230758

[41] P.-G. L. P.-G. Lin, Q.-T. L. P.-G. Lin, J.-Q. Z. Q.-T. Li, J.-H. W. J.-Q. Zhou, M.-W. J. J.-H. Wang, and C. Z. M.-W. Jian, "FINANCIAL FORECASTING METHOD FOR GENERATIVE ADVERSARIAL NETWORKS BASED ON MULTI-MODEL FUSION," 2023. doi: https://doi.org/10.53106/199115992023023401010

[42] J. K. Chiang and R. Chi, "PREDICTING STOCK PRICES BASED ON PRICE/VOLUME WITH DEEP LEARNING AND SYSTEM ENGINEERING AGGREGATE WITH DYNAMIC BEHAVIORS AND TRADING SIGNALS," 2023. doi: https://doi.org/10.20944/preprints202308.1979.v1

[43] M. Li, "THE IMPACT OF TRADING VOLUME ON STOCK PRICE VOLATILITY," 2024. doi: https://doi.org/10.54097/wasnyj47.

[44] Z.-L. Wang, "RESEARCH ON THE AMPLIFICATION EFFECT OF TRADING VOLUME ON MISPRICING IN THE CHINESE A-SHARE MARKET," 2024. doi: https://doi.org/10.54097/571jf356

[45] M. Sun, J. Wang, Q. Li, J. Zhou, C. Cui, and M. Jian, "STOCK INDEX TIME SERIES PREDICTION BASED ON ENSEMBLE LEARNING MODEL," 2023. doi: https://doi.org/10.3233/JCM-226523.