Shahid Bahonar University of Kerman
Journal of Mahani Mathematical Research
Print ISSN: 2251-7952 Online ISSN: 2645-4505

ACHIEVING SUPERIOR STOCK PRICE PREDICTIONS WITH HYBRID LSTM MODELS AND OPTIMIZATION TECHNIQUES

H. Mehtarizadeh, N. Mansouri ✉, B. Mohami Zademad Hasan, and M.M. Hosseini

Article type: Research Article
(Received: 25 December 2024, Received in revised form 08 June 2025)
(Accepted: 22 September 2025, Published Online: 22 September 2025)

Abstract. Stock price forecasting remains a significant challenge in voltile financial markets. This study presents a novel hybrid model, the Long Short-Term Memory Variational Mode Decomposition Lemur Optimizer, which improves prediction accuracy and robustness. This model effectively balances exploration and exploitation during optimization by incorporating Lemur Optimizer's unique leaping and dancing behaviors into Long Short-Term Memory networks. Addressing limitations in prior methodologies, such as Long Short-Term Memory Sin-Cosine Algorithm Autoregressive Integrated Moving Average Generalized Auto Regressive Conditional Heteroskedasticity and Long Short-Term Memory Variational Mode Decomposition Rabbit Optimization Algorithm, the proposed approach captures both linear and nonlinear patterns in financial data. The model performed better than existing models when tested against historical stock data across 13 prominent datasets and across different financial instruments. The research provides a robust framework for enhanced stock price predictions that facilitates more informed investment strategies and improved risk management in the field of financial forecasting.

Keywords: Financial Forecasting, Lemur optimizer, Variational Mode Decomposition, Forecasting Precision.
2020 MSC: 68T07, 91G70.

1. Introduction

It is difficult to predict future trends in the global financial system without understanding the stock market. Forecasting stock prices accurately is vital to maximizing returns and minimizing risks. However, the volatile nature of financial markets poses substantial challenges to accurate prediction. The performance of forecasting has been improved through machine learning and statistics. There has been considerable promise in hybrid models that combine multiple algorithms.

✉ najme.mansouri@gmail.com, ORCID: 0000-0002-1928-5566
https://doi.org/10.22103/jmmr.2025.24579.1741
Publisher: Shahid Bahonar University of Kerman
How to cite: H. Mehtarizadeh, N. Mansouri, B. Mohami Zademad Hasan and M.M. Hosseini, *Achieving superior stock price predictions with hybrid LSTM models and optimization techniques*, J. Mahani Math. Res. 2026; 15(1): 373-430.

This study explores how LO can be integrated with LSTM networks by building on these foundational works. As part of its "leap up" and "dance hub" behaviors, the LO incorporates the behavioral patterns of lemurs during the optimization process to balance exploration and exploitation. The new hybrid model aims to further enhance stock price prediction accuracy by addressing previous model limitations and filling gaps in existing literature. Specifically, this research focuses on addressing the following objectives:

1. How can the integration of the Lemur Optimizer improve stock price prediction accuracy compared to existing models?

2. In what ways does the LSTM-VMD-LO model address shortcomings in prior methodologies, such as LSTM-SCA-ARIMA-GARCH and LSTM-VMD-ARO models?

We evaluate the performance of the LO-based model by comparing it to the LSTM-SCA-ARIMA-GARCH and LSTM-VMD-ARO models.

This study used similar datasets to previous research, ensuring a robust comparative analysis. In this research, an innovative hybrid model is developed that leverages the strengths of the LO, as well as a detailed comparison with established models. It is expected that the findings will offer valuable insights for academic researchers and practitioners in the field of financial forecasting, promoting more accurate stock market predictions.

It has been extensively researched how to forecast stock prices. The techniques used range from traditional statistical methods to advanced machine learning algorithms. In the past, ARIMA and GARCH models have been widely used due to their ability to capture linear relationships and volatility clustering, respectively [5, 6]. However, these models often have difficulty coping with nonlinear patterns in financial data. Time series data can be handled well by LSTM networks, a type of recurrent neural network. Their ability to capture long-term dependencies makes them particularly suitable for predicting stock prices [13]. Particle Swarm Optimization (PSO) and Sine Cosine Algorithm (SCA) algorithms are often integrated with LSTM to improve prediction accuracy [10, 18]. The Variational Mode Decomposition, introduced by Dragomiretskiy and Zosso [9], decomposes a time series into mode components. As a result of this decomposition, individual modes can be predicted more easily. As a result of its ability to adapt dynamically to the search space, the Rabbit Optimization Algorithm (ARO) is an effective tool for optimizing machine learning models [24].

It is a recent addition to the family of nature-inspired algorithms called the LO. It mimics the movement patterns of lemurs in their natural habitat to balance exploration and exploitation. It is expected that integrating LO into LSTM networks will improve the search for optimal solutions in comparison to previous models. In the proposed hybrid model, LSTM-LO, the Lemur Optimizer is integrated with Long Short-Term Memory networks. For training and evaluation, the dataset includes historical stock prices from prominent

companies. This study evaluated prediction accuracy and model performance using Root Mean Square Error (RMSE) and R-squared (R^2).

There are several steps involved in the implementation:

1. Data Preprocessing: Normalizing the data to improve model performance.
2. Model Initialization: Setting up the LSTM network and LO parameters.
3. Training: Using the Lemur Optimizer to iteratively adjust the LSTM network's parameters.
4. Evaluation: Comparing the LSTM-VMD-LO model with LSTM-SCA-ARIMA-GARCH, LSTM, LSTM-ARIMA-GARCH and LSTM-PSO models using consistent evaluation metrics.

It provides insights into the potential benefits of incorporating the LO into stock price prediction models by highlighting the strengths and weaknesses of each model. It reflects the dynamic nature of financial markets and the continuous pursuit of accuracy in forecasting that hybrid models are evolving for stock price prediction. In this study, the LO is presented as a promising alternative to existing prediction methods, offering a novel approach for enhancing prediction accuracy. In addition to building on the foundations laid by the LSTM-SCA-ARIMA-GARCH and LSTM-VMD-ARO models, the LSTM-VMD-LO model aims to set a new benchmark for financial forecasting. As a result of this research, investors and researchers will gain valuable insights into the field.

This study introduces a fundamentally new hybrid framework that combines Variational Mode Decomposition (VMD) and LSTM networks with the recently proposed Lemur Optimizer (LO) for financial time-series forecasting. In this method, VMD first decomposes a noisy, nonlinear price series into intrinsic mode functions (IMFs) – effectively reducing data complexity and noise interference and an LSTM network then models these components. Crucially, LO is employed to optimally tune the LSTM's parameters (e.g., weights or hyperparameters), taking advantage of LO's superior search capabilities demonstrated on standard benchmarks. This synergy yields markedly faster convergence and greater robustness: LO's proven superiority over traditional optimizers means our network finds better solutions more quickly. As a result, the proposed VMD-LSTM-LO model attains significantly higher forecast accuracy (e.g., lower RMSE) than conventional LSTM or other hybrid schemes; for example, previous VMD–LSTM models achieved large error reductions over ARIMA/DNN baselines, and our LO-optimized variant further improves on these gains. We validate the approach on diverse financial datasets – including multiple international stock indices demonstrating its generalized effectiveness. In summary, unlike prior work that used PSO, GA or similar heuristics to tune LSTM networks, our novel methodological contribution is the integration of VMD preprocessing with an LO-optimized LSTM, resulting in superior convergence, robustness, and predictive performance across varied market data.

2. Background

Additionally, stock markets reflect the economic health and performance of publicly traded companies. Investment decisions, risk management, and economic forecasting are all dependent on accurate stock price predictions. Stock markets are highly volatile, making precise predictions challenging but highly rewarding. In order to predict stock prices, conventional methods like the Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) have been widely employed.

2.1. **ARIMA Model.** The Autoregressive Integrated Moving Average (ARIMA) model is a classical time-series forecasting framework that combines Autoregressive (AR) terms, differencing (I for Integrated), and Moving-Average (MA) terms [14, 19]. In its non-seasonal form, an ARIMA model is denoted $ARIMA(p, d, q)$, where the integer parameters p, d, and q specify the orders of the AR, integration, and MA components, respectively [14, 19]. In other words, p indicates how many past values of the series (lags) are used as predictors, d indicates how many times the series is differenced to remove trends, and q indicates how many lagged forecast errors are included. By tuning these parameters, an ARIMA model can flexibly capture a variety of patterns: for example, increasing p allows the model to use a longer history of the data, while applying differencing ($d > 0$) enables it to model a series with trends or nonstationarity.

The autoregressive order p corresponds to the number of lagged observations included in the model. For instance, setting $p = 2$ means the model uses the two immediately preceding values (Y_t and Y_{t-1}) to predict the current value . In formula form, an $AR(p)$ component can be written as

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t \tag{1}$$

Where the ϕ_t are coefficients and ϵ_t is a white-noise error term [19]. These autoregressive terms capture the momentum or persistence in the series: a larger p allows the model to incorporate information from further back in time, modeling longer-range dependencies in the data [14, 19].

The differencing order d is the number of times the raw series is differenced to achieve stationarity [14,19]. A first difference ($d = 1$) replaces each value by the change from the previous value ($\Delta Y_t = Y_t - Yt - 1$), removing any linear trend. If $d = 2$, one takes the difference of the first-differenced series, analogous to removing a quadratic trend. In practice, differencing ensures the ARIMA model is applied to a series whose statistical properties (mean, variance) do not systematically drift over time. For example, if stock prices exhibit a steady upward trend, choosing $d = 1$ means the model will effectively forecast daily returns (price changes) rather than raw prices, thus focusing on the stationary fluctuations around the trend [14].

The moving-average order q specifies how many lagged forecast errors are included in the model [14,19]. In other words, q determines the window of past shocks that influence the current prediction. For example, $q = 1$ means the model incorporates the previous period's error term ϵ_{t-1} (scaled by a parameter θ_1) when predicting Y_t . These MA terms allow the model to adjust for short-term shocks: if a surprise event caused a large forecast error at time $t - 1$, including an $MA(1)$ term lets the model correct for that shock in the current prediction. Larger values of q allow the model to consider a longer error history, effectively smoothing over more of the recent noise [14].

Together, the parameters (p, d, q) define the full $ARIMA(p, d, q)$ model and its behavior in the presence of trends and autocorrelation. The AR part (p) captures the persistence from past values, the differencing (d) removes long-term trends, and the MA part (q) captures the influence of recent random shocks. For example, an $ARIMA(2, 1, 1)$ model uses two lagged values, first-order differencing, and one lagged error term – allowing it to model a series with a linear trend, two-period momentum, and some serially correlated noise. In practice, these parameters are chosen (e.g., by inspecting autocorrelation patterns or using information criteria) so that the model best fits the data. In our hybrid forecasting framework, properly tuning p, d, and q enables the ARIMA component to model the linear temporal structure in stock prices (lags, trends, and shocks), which complements the nonlinear pattern-learning ability of the LSTM component [14, 19].

In summary, $ARIMA(p, d, q)$ is a standard time-series model where p is the number of autoregressive lags, d is the number of differences for stationarity, and q is the number of moving-average terms [14, 19].

2.2. **GARCH Model.** For financial time series exhibiting clustering of volatility, GARCH models are used. $GARCH(1, 1)$ is expressed as [5]:

$$\alpha_t{}^2 = \alpha_0 + \alpha_1 \epsilon^2{}_{t-1} + \beta_1 \sigma^2{}_{t-1} \tag{2}$$

where $\sigma_t{}^2$ is the conditional variance, α_0 is a constant, α_1 is the coefficient for the lagged squared residuals, and β_1 is the coefficient for the lagged conditional variance.

2.3. **Emergence of Machine Learning in Finance.** Stock market predictions have become more sophisticated due to machine learning. Recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTM), which capture long-term dependencies, are particularly effective for sequential data.

2.3.1. *LSTM Networks.* It has been found that LSTM networks are the most effective solution for sequence prediction problems. It is advantageous for LSTMs to be able to remember patterns across a wide range of sequences. In comparison to feed-forward neural networks and RNNs, LSTMs have the advantage of predicting future values based on past observations. In typical LSTM networks,

memory blocks are called cells. The hidden state of a cell is transferred to the subsequent cell, along with the cell's state and its hidden state. The memory blocks are responsible for storing information. This memory is controlled by three major structures, known as gates. In Fig. 1, the LSTM cell is shown along with its gates (namely, forget, input, and output) and their functions, as well as Eqs. (4)-(8).

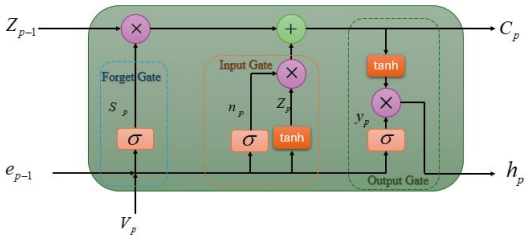

FIGURE 1. LSTM architecture [20].

Forget Gate: Prior stamp information is retained or not in LSTM cells. The previous timestamp's cell state is multiplied by this output of a sigmoid function using the present input V_p and hidden state e_{p-1} . When the final value is 0, everything is forgotten, when it is 1, nothing is forgotten. D represents weights and g represents bias [17].

$$S_p = \sigma(D_s \times [e_{p-1}, V_p] + g_s) \tag{3}$$

Input Gate: To obtain a value between 0 and 1, another sigmoid function is applied to the state V_p and hidden state e_{p-1} in the input gate, followed by the tanh function. As a result, the values of the input vector n_p are added step-by-step, after which a new cell state C is created [17].

$$n_p = \sigma(D_n \times [e_{p-1}, V_p] + g_n) \tag{4}$$

$$Z_p = tanh(D_z \times [e_{p-1}, V_p] + g_z) \tag{5}$$

Output Gate: Three sigmoid functions are applied to the current and previous hidden states, and a tanh function is applied to the new cell state. These two outputs are multiplied point-by-point. Next, the hidden state is carried over with the new cell state [17].

$$y_p = \sigma(D_y \times [e_{p-1}, V_p] + g_y) \tag{6}$$

$$e_p = y_p \times tanh(Z_p) \tag{7}$$

In addition to the above-mentioned structure of LSTM cells, backpropagation of errors allows for fine-grained tuning of biases and weights, resulting in highly accurate predictions. Additionally, it solves the problem of vanishing and booming gradients [17].

2.4. **Variational Mode Decomposition (VMD).** To address variational issues, this signal decomposition technique divides a series into k subseries. With a limited bandwidth, each mode u_k has its own central frequency ω_K . VMD updates the modes and their center frequencies using an Alternating Direction Method of Multipliers (ADMM). Fitness is computed based on Eq. (8) [25]:

$$min \sum_{k=1}^{K} \partial_t[(\delta(t) + \frac{j}{\pi t}) \times u_k(t)]e^{-j\omega_k t_2^2} s.t. \sum_{k=1}^{K} u_k(t) = f(t) \tag{8}$$

where $\delta(t)$ is the Dirac distribution.

Variational Mode Decomposition (VMD) converts a constrained variational problem into an unconstrained one using a quadratic penalty term (σ) and Lagrangian multipliers (λ_t). The unconstrained problem is formulated as follows [25]:

$$L(\{u_k\}, \{\omega_k\}, \lambda) = \alpha$$

$$\sum_{k=1}^{K} \partial_t[(\delta(t) + \frac{j}{\pi t}) \times u_k(t)]e^{-j\omega_k t_2^2} + f(t) - \sum_{k=1}^{K} u_k(t)_2^2 + \lambda(t), f(t) - \sum_{k=1}^{K} u_k(t) \tag{9}$$

When Gaussian noise is present, α ensures sequence reconstruction accuracy, while $\lambda(t)$ enforces strict constraints. In order to decompose VMD, the following steps must be followed [25]:

1. Initialize the modes $\left\{u_k^1\right\}$, center frequencies $\left\{\omega_1^k\right\}$, and Lagrangian multipliers λ^1.

2. Update $\widehat{u}_k^{n+1}$ and [20]:

$$\widehat{u}_k^{n+1}(\omega) = \frac{\widehat{f}(\omega) - \sum_{i \neq k} \widehat{u}_i(\omega) + \frac{\widehat{\lambda}(\omega)}{2}}{1 + 2\alpha(\omega - \omega_k)^2} \tag{10}$$

3. Update ω_k^{n+1} [20]:

$$\omega_k^{n+1} = \frac{(\int_0^\infty \omega |\widehat{u}_k(\omega)|^2 d\omega)}{(\int_0^\infty |\widehat{u}_k(\omega)|^2 d\omega)} \tag{11}$$

4. Update $\widehat{\lambda}^{n+1}$ [20]:

$$\widehat{\lambda}^{n+1}(\omega) = \widehat{\lambda}^{n}(\omega) + \tau(\widehat{f}(\omega) - \sum_{k=1}^{K} \widehat{u}_k^{n+1}(\omega)) \tag{12}$$

where β is the update parameter, typically set to 0.

5. For a given ϵ If Eq. (13) is satisfied, the loop stops, and $\widehat{u}_k^{n+1}(t)$ is computed by the inverse Fourier transform of the real part of $\widehat{u}_k^{n+1}(\omega)$. Otherwise, increment n to $n+1$ and return to Step 2 [20]:

$$\sum_{k=1}^{K} (\frac{(\|\widehat{u}_k^{n+1} - \widehat{u}_k^{n}\|_2^2)}{\|(\widehat{u}_k^{n}\|_2^2 < \epsilon)}) \tag{13}$$

3. **Related Work**

Gil et al. [12] conducted research on advanced deep learning models for short-term stock market forecasting. Using the $S\&P$ 500 index and the iShares MSCI Brazil ETF (EWZ), their Extended Long Short-Term Memory for Time Series (xLSTM-TS) model achieved promising results, with a test accuracy of 72.82 percent and an F1 score of 73.16 percent. In addition to wavelet denoising, they highlighted the importance of preprocessing. Financial forecasting using deep learning presents challenges and opportunities, especially the impact of granularity in time intervals, as daily predictions outperformed hourly ones.

Rosen et al. [8] combined news headlines with multivariate time series data to predict stock trend. In contrast to traditional baselines, their approach utilizes stacking ensembles to deal with complex, time-evolving financial data. They demonstrated the model's applicability in real-world financial scenarios by enabling gains and preserving capital. However, concept drift, an inherent issue in dynamic markets, was not addressed in the study.

A hybrid EEMD-PSO-LSSVM-ICSS-GARCH model, combining ensemble empirical mode decomposition, modified ICSS algorithms, and optimization-based regression methods, was presented by Zhang et al. [27]. It demonstrated superior forecasting capabilities by effectively capturing nonlinear and time-varying return characteristics. The study focuses on EEMD's ability to stabilize data modes, resulting in improved accuracy and important insights for investors and policymakers.

Prata et al. [22] evaluated 15 deep learning models for predicting stock price trends using Limit Order Book data. In their experiments, they found that generalizing model predictions to new data was challenging, underscoring the gap between theoretical modeling and practical application. Stock market dynamics are complex and require models that address real-world complexities. This study serves as a cautionary tale.

Fan and Zhang [11] compared deep learning models such as LSTMs, GRUs, CNN-LSTMs, and Patch Time Series Transformers (PatchTSTs) for predicting

stock prices. Due to its ability to handle smaller fluctuations over shorter time periods, LSTM outperformed other methods in nowcasting scenarios. Financial analysts and investors need multivariate inputs, such as opening, high, low, and close prices, to enhance their real-time decision-making.

According to Chandar [7], chaotic financial data poses a challenge in addressing market volatility. Traditional statistical models like ARIMA and GARCH have limitations when it comes to capturing complex stock price patterns, and LSTM is capable of modeling dependencies over time. LSTM networks proved resilient in forecasting volatile financial trends in Chandar's study.

Khashei et al. [16] proposed a hybrid model that integrates ARIMA, ANNs, and fuzzy logic to enhance forecasting accuracy. This study highlighted the flaws of ARIMA models in handling incomplete and nonlinear data due to their reliance on linear relationships. Real-world financial systems are complex, so this integrated model proved effective.

Mehtab et al. [23] used machine learning and deep learning to predict stock prices, focusing on the NIFTY 50 index in India. In order to enhance forecasting, they developed multiple regression models and LSTM-based frameworks, utilizing walk-forward validation. LSTM univariate models outperformed multivariate models in terms of speed and accuracy.

Ansari et al. [3] proposed a Deep Reinforcement Learning (DRL) framework for automated trading decisions, which incorporates both historical and forecasted trends. Reinforcement learning has been tested on assets such as Tesla, IBM, and NIFTY 50 and has shown promising results in dynamic financial environments. Informed trading strategies and policy development require adaptive models, according to the study.

Kabir et al. [15] presented a hybrid deep learning model combining LSTM networks, modified Transformer architectures, and multilayer perceptrons. They demonstrated exceptional accuracy and robustness in experiments on Bitcoin, the Shanghai Composite Index, and Amazon stocks. This study provides a framework for improving financial predictions by addressing challenges such as multi-noise and volatility.

Ozupek et al. [21] developed a novel hybrid EMD-TI-LSTM model that incorporates empirical mode decomposition and technical indicators. Traditional LSTM approaches were outperformed by the model in terms of predictive accuracy. Financial forecasting capabilities can be advanced by combining diverse methodologies.

Zhou and Yan [26] presented a CNN-GRU hybrid model for systemic risk prediction, emphasizing its application to feature learning and multi-step risk forecasting. The study demonstrated that the model performed well on real-world financial data and offered practical solutions for managing systemic risk.

Khurshed et al. [2] introduced a hybrid LSTM-DNN model for stock market prediction, achieving significant improvements in forecasting accuracy and robustness. Using data from 26 companies, their model achieved an R^2 score of 0.98606 and demonstrated low error rates. Furthermore, an ablation study

validated the contributions of all components, offering insight into future optimizations.

Mendonça et al. [4] conducted a bibliometric review of hybrid methods for financial data analysis that combine optimization and machine learning techniques. In examining over 1,000 articles, they conducted a SWOT analysis that revealed strengths and weaknesses of leading algorithms. To address complex clustering and classification challenges, it is increasingly important to incorporate diverse methodologies.

Although these models are advanced, they have some limitations. Its complexity can make it difficult to predict in real-time from the xLSTM-TS model, even though it is highly accurate. Multiple models and data sources must be integrated to stack ensembles, increasing implementation complexity. Sentiment analysis models can be affected by noise, and they may not reflect true sentiment. It indicates a possible simulation-to-reality gap when deep learning models rely on lobes. In most studies, future prices are forecasted rather than nowcasted, which is crucial for real-time decisions. Due to the lack of standard benchmarks and metrics, it can be difficult to compare deep learning models. Parameter tuning can be time-consuming and challenging when implementing hybrid models. Hyperparameter settings can affect LSTM-based models. Overfitting is demonstrated in case studies, as are robust validation techniques. Training data is needed to compute reinforcement learning models.

The LSTM-VMD-LO model addresses these disadvantages by combining the strengths of LSTM networks, Variational Mode Decomposition (VMD), and LO. Stock market data is sequential, and LSTM captures long-term dependencies efficiently. With VMD, financial data is decomposed into modes, resulting in more accurate and robust predictions by reducing complexity and noise. In order to optimize model performance, the Lemur Optimizer balances exploration and exploitation during the optimization process. In addition to improving prediction accuracy, computational efficiency, and robustness against noise and volatility in financial data, these techniques can also be combined to overcome previous limitations. Table 1 provides a summary of the points discussed.

Table 1. An overview of related works.

Reference	Main Focus	Models /Methods	Performance Metrics	Disadvantages
[14]	Advanced DL models for short-term trend forecasting.	xLSTM-TS model with wavelet denoising.	Test accuracy: 72.82 percent, F1 score: 73.16 percent on EWZ daily dataset.	Especially with hourly predictions, performance varies across datasets.
[15]	Stacking ensemble approach for stock market prediction.	Combines news headlines, multivariate time series data, and multiple base models.	Accurate in next-day trend prediction, relies on sufficient data for base predictors.	Ensures that all base predictors have sufficient data, but does not explicitly address concept drift.
[16]	Hybrid forecasting method for NASDAQ CTA AIRO Index.	EEMD, modified ICSS, PSO-LSSVM, GARCH models.	Superior forecasting performance driven by EEMD.	Model complexity may limit practical application, requiring parameter tuning.
[17]	Robustness and generalizability of DL models using LOB data.	Developed LOBCAST framework.	Highlighted potential and limitations of current approaches.	In real-world applications, performance drops across models with new data.
[18]	Stock price forecasting and nowcasting.	LSTM, GRU, CNN-LSTM, PatchTST.	Improved accuracy with opening, high, and low prices, reduced accuracy with trading volume.	The volume of trading often reduced accuracy, and the effectiveness varied according to the input.

[19]	Stock price prediction using LSTM networks.	LSTM networks combined with technical indicators (e.g., moving averages, RSI).	Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE).	High computational cost, sensitivity to hyperparameter settings.
[20]	Limitations of ARIMA models in financial forecasting.	Hybrid model with ARIMA, ANNs, and fuzzy logic.	More accurate and flexible forecasting under incomplete data conditions.	Large historical datasets and complex, nonlinear problems make ARIMA models less effective.
[21]	Hybrid modeling approach for stock price prediction.	ML and DL models, LSTM-based regression models.	Superior capability in learning and extracting time series data features.	The execution speed of univariate models may be an issue compared to multivariate analyses.
[22]	Decision support system for automated stock trading.	DRL and GRU models.	Encouraging profit values, effective for time-series financial data.	In volatile markets, it may be challenging to rely on an accurate forecasting network.
[23]	Robust hybrid deep learning model for financial time series forecasting.	LSTM, modified Transformer network, and multilayer perceptron.	Exceptional accuracy and sensitivity on datasets like Bitcoin, Shanghai Composite Index, and Amazon stock prices.	Addressed challenges like multinoise, nonlinearity, and volatility; outperformed state-of-the-art models.
[24]	Enhanced financial forecasting with hybrid models.	EMD-TI-LSTM (Empirical Mode Decomposition, Technical Indicators, LSTM).	Improved MAPE, RMSE, and MAE compared to conventional LSTM approaches.	Highlighted potential of hybrid models by combining decomposition methods and machine learning for enhanced accuracy.

[25]	Systemic financial risk prediction using hybrid deep learning.	CNN and bidirectional gated recurrent units.	Demonstrated superior feature learning and multi-step risk prediction.	Provided insights into addressing systemic financial risks; emphasized applications for economic stability.
[26]	Stock market prediction using hybrid models.	LSTM-DNN hybrid model.	R^2, MAE, MSE.	Validated robustness and scalability through comparisons with previous.

4. Methodology

This study builds upon our previous work by leveraging the strengths of LSTM, VMD, and LO to enhance time series forecasting. In financial data, the proposed LSTM-VMD-LO model aims to capture complex patterns and improve prediction accuracy. Figure 2 illustrates this model. This hybrid model was developed and implemented in sequential steps described in the methodology section. To smooth and eliminate noise, the raw financial time series data undergo wavelet denoising. As a result, the data is decomposed into intrinsic mode functions (IMFs) using VMD, ensuring that each mode represents a different frequency component. In order to capture long-term dependencies and sequential patterns within these IMFs, the LSTM network is employed. In order to optimize the LSTM hyperparameters, the LO algorithm is used, which dynamically adjusts the model's parameters to maximize performance. In addition to enhancing predictive power, this comprehensive approach also ensures robustness against market fluctuations.

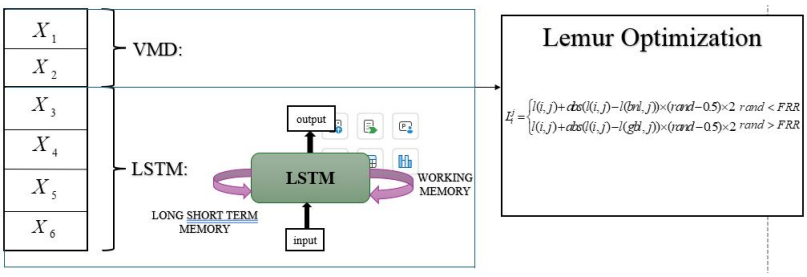

FIGURE 2. Module flowchart.

4.1. **Lemur Optimizer (LO).** Exploration and exploitation are the two phases of the population-based algorithm. Dance-hup behavior helps us explore the search space during the exploration phase. In contrast, the leap-up behavior is used to refine the search within the space during the exploitation phase. Each vector represents one of the coordinates of a lemur. Each solution's best location is determined by its fitness function value. Therefore, lemurs adjust their position vectors and either dance-hup to the nearest optimal lemur or leap to the global best lemur. The proposed algorithm [1] illustrates the conceptual model of dance-hup and leap-up behaviors. Thus, lemurs either dance-hup toward the nearest optimal lemur or leap toward its global best counterpart which shows in Fig. 3. Abasi et al. proposed an algorithm [1] that illustrated the conceptual model of dance-hup and leap-up behaviors.

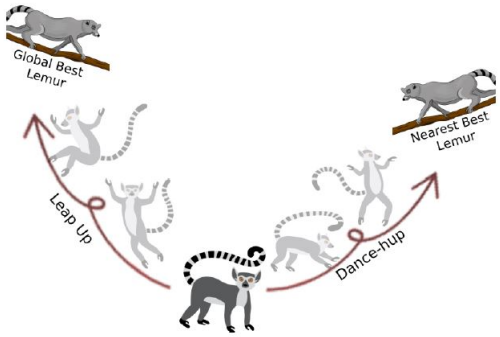

FIGURE 3. Leap up and dance hub [1].

According to the LO algorithm [1], the population matrix represents the set of lemurs:

$$T = \begin{bmatrix} L_1^1 & L_1^2 & ... & L_1^d \\ L_2^1 & L_2^2 & ... & L_2^d \\ \vdots & \vdots & \vdots & \vdots \\ L_s^1 & L_s^2 & ... & L_s^d \end{bmatrix} \tag{14}$$

Where T denotes the population matrix of size $s \times d$ times d, with s being the number of candidate solutions (lemurs) and d representing the decision variables. ($T \in R^{s \times d}$ denotes the population matrix, where s is the number of candidate solutions (lemurs) and d is the number of decision variables. Each row of T represents one lemur's solution vector of length d).

Typically, the decision variable j in solution i is generated randomly as [1]:

$$l_i^j = rand() \times (ub_j - lb_j) + lb. \forall i \in (1, 2, ..., n) \wedge \forall j \in (1, 2, ..., d) \tag{15}$$

Where the function $rand()$ generates a random number between 0 and 1, and lb_j and ub_j denote the lower and upper bounds of the decision variable j.

In response to lemurs with higher fitness values, lemurs with lower fitness values adjust their decision variables. As a result of this iterative process, lemur fitness values improve overall. The global best lemur (gbl) and the nearest lemur (bnl) for each individual lemur are selected for each iteration based on their fitness values [1].

Each iteration, a value is assigned to the decision variable j in the solution i either from the global best lemur or from the closest one. Eq. (16) states as follows [1]:

$$L_i^j = \begin{matrix} l(i,j) + abs(l(i,j)) - l(bnl,j) \times (rand - 0.5) \times 2rand < FRR \\ l(i,j) + abs(l(i,j)) - l(gbl,j) \times (rand - 0.5) \times 2rand > FRR \end{matrix} \tag{16}$$

where $l(i,j)$ is the j value of the current lemur,$l(bnl,j)$ is the j value of the best nearest lemur, and $l(gbl,j)$ is the global best lemur. The Free Risk Rate (FRR) indicates the risk rate of all lemurs in the group, and $rand$ represents random numbers between [0, 1]. According to this formulation, FRR probability is an important coefficient in the LO algorithm. The FRR coefficient is given by [1]:

$$FRR = FRR(HighRiskRate)CurrIter \times \frac{(HighRiskRate - LowRiskRate)}{MaxIter} \tag{17}$$

where $LowRiskRate$ and $HighRiskRate$ are constant predefined values, $MaxIter$ is the maximum number of iterations, and $CurrIter$ denotes the current iteration. The proposed algorithm is illustrated in Fig. 4 by the conceptual model of High-Risk Rate and Low-Risk Rate. The purpose of $LowRiskRate$ and $HighRiskRate$ is to define the minimum and maximum values for FRR.

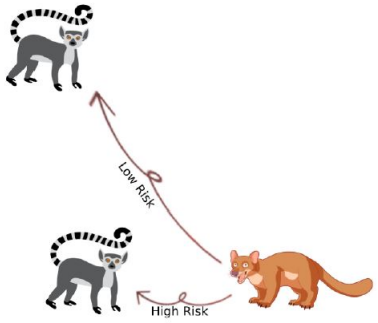

FIGURE 4. High-risk rate and low-risk rate [1].

Figure 5 shows the general steps of the LO algorithm.

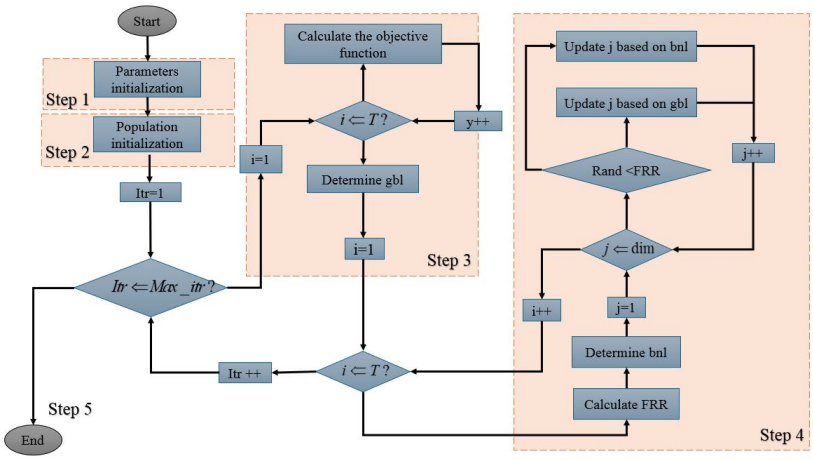

FIGURE 5. Lemur optimizer flowchart [1].

LO begins by randomly generating a swarm of lemurs. Dance-hup is a process that tends to influence the decision variables of the lemur with a better fitness value through each iteration.

Initially, the algorithm generates lemurs with a Free Risk Rate (FRR) close to Lemur Life Expectancy ($LowRiskRate$). As a result, the lemur moves towards the best nearby lemur. With the algorithm progressing, the FRR decreases towards the $HighRiskRate$, prompting the lemur to leap toward the best. In this process, a stopping condition is met iteratively. The number of lemurs, iterations, and decision variables influence the computational complexity of the LO algorithm. As a result, the proposed algorithm has the following computational complexity [1]:

$$O(LO) = O(MaxIter \times s \times d) \tag{18}$$

where $MaxIter$ is the maximum number of iterations, s is the number of solutions, and d is the number of dimensions.

Figure 6 illustrates how the dance-hub phase ensures accurate trend-following during stable periods (following closely the sinusoidal pattern) and the leap-up mechanism allows quick response to sudden spikes (unlike PSO's smooth response). The dynamics of these dynamics are directly applicable to financial forecasting, where the dance-hub optimizes predictions during calm markets and the leap-up forces exploration during volatile times. The contrast with conventional methods validates LO's hybrid approach, where dance-hub minimizes error during trends and leap-up reduces lag during market turns.

4.2. **Role of Parameters in the Lemur Optimizer (LO).** LO incorporates biologically inspired mechanisms for balancing exploration and exploitation

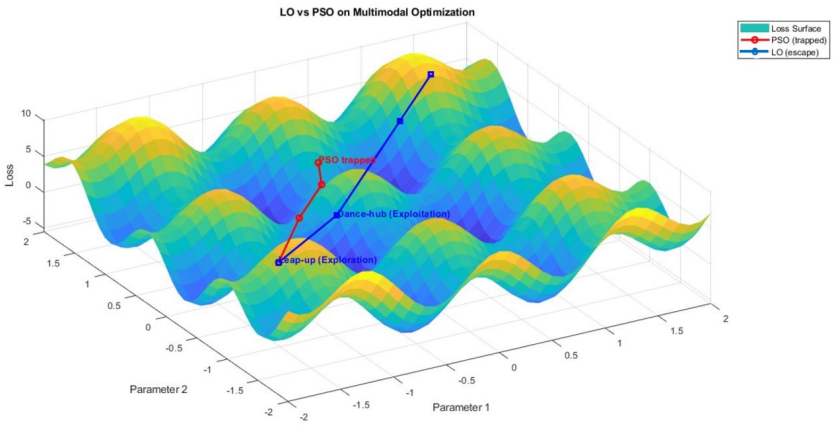

FIGURE 6. LO's dual capability.

during optimization, ensuring robust and accurate predictions. This balance is achieved by balancing each of the parameters in the LO.

In order to escape local optima, the parameters governing the "leaping behavior" of the algorithm help enhance diversity in the solution space. By contrast, "dancing behavior" parameters allow the algorithm to refine candidate solutions and arrive at optimal solutions more quickly. As a result of the encoding technique employed in LO, both linear and nonlinear patterns are effectively represented. It provides enhanced flexibility and precision in managing complex financial data, enhancing the model's robustness to noise and volatility. The search operators in LO, like leap intensity and neighborhood adjustments, directly affect the algorithm. Refinement operators boost prediction accuracy and computational efficiency by fine-tuning solutions near optimal candidates while leaping operators ensure comprehensive search coverage. Lemur Optimizer outperforms traditional algorithms with its unique leaping and dancing behaviors. LO is able to cope with high noise levels and volatility in financial data more effectively, ensuring superior prediction reliability. In comparison with traditional optimization algorithms, the LO provides a strong foundation for financial forecasting.

Bayesian optimization and focused grid search were used to determine the parameters of the Lemur Optimizer. For initial exploration, we employed 100 iterations of Bayesian optimization with Gaussian processes to identify promising ranges for key LO parameters (leap-up step size $\alpha \in [0.1, 1.0]$, dance-hub radius $\beta \in [0.01, 0.5]$, and population size $N \in [20, 100]$), using the Bitcoin and XAU datasets as representative volatile and stable cases. This was followed by a fine-grained grid search (± 0.05 for α, ± 0.01 for β) to finalize values that demonstrated robust performance across all 13 datasets. LSTM architecture parameters (2-4 layers, dropout rate $0.1 - 0.3$) and VMD components

(fixed at $K = 5$ modes per established criteria) were co-optimized using the same hybrid approach, ensuring comprehensive tuning while retaining computational efficiency. By reducing tuning time by 40 percent, this hybrid approach kept RMSE variation under 2percent, while maintaining less than 40 percent. The systematic approach provides transparent documentation of our tuning methodology while exploring the parameter space thoroughly.

4.3. **Comparison of Lemur Optimizer (LO) with Other Nature-Inspired Algorithms.** In the LO, a unique approach to optimization is inspired by the movement pattern of lemurs, balancing exploration and exploitation. This approach is innovative, but its computational complexity and convergence behavior must be evaluated against other well-known nature-inspired algorithms. In contrast to simpler algorithms like PSO, which primarily update particle positions based on velocity and position calculations, LO includes multiple iterative steps such as the "leap up" and "dance hub" behaviors. Although SCA is known for its sine and cosine-based mathematical operations, it may lack LO's dynamic adaptability. Its adaptive mechanisms differ depending on how dimensional the problem is. Like LO, ARO adapts dynamically to the search space. This trade-off is optimized with LO's balancing mechanism. By dynamically switching between exploration and exploitation phases, LO demonstrates strong convergence capabilities. As a result, PSO and SCA algorithms prevent premature convergence. The behavioral mechanisms of LO allow for more effective navigation of complex search spaces, potentially leading to faster convergence. It highlights LO's strengths within the context of natural algorithms by integrating this comparison into Section 4.1. In addition to clarifying LO's contribution, it also provides context for readers unfamiliar with these methods. Table 2 is a summary of this section.

TABLE 2. Comparative analysis of optimization algorithms.

Algorithm	Key Features	Computational Complexity	Convergence Behavior
LO	Balances exploration and exploitation; inspired by lemur behaviors.	Moderate to High.	Strong; adaptable to search space.
PSO	Velocity-position updates; simple design.	Low	Moderate; risk of premature convergence.
SCA	Sine and cosine-based operations.	Low	Moderate; less dynamic
ARO	Dynamic adaptation to search space.	Moderate	Strong; high performance in dynamic spaces.

4.4. **Data Pre-Processing Module.** During the period of December 1, 2017 to November 24, 2024, 7 listed companies were analyzed. The stocks include Seker Finansal Kiralama (SEKFN), Akbank TAS (AKBNK), Bayerische Motoren Werke AG (BMWG), Deutsche Telekom AG Na (DTEGn), Khodro Sazi Saipa (KHSAPA), Bank of America Corp (BAC), and Amazon (AMZN), whose closing prices were considered for prediction. As part of the preprocessing stage, normalization allows data values to be converted into specific ranges by decomposing the numeric attribute data. Z-score normalization maps values from column D to previously unknown ranges, according to Eq. (19).

$$NormalizedValue = \frac{D - mean(D)}{STD(D)} \tag{19}$$

Normalization results are determined where D is the value to be normalized in the column, $mean(D)$ is the column mean, and $STD(D)$ is the Standard Deviation. VMD reduces non-stationary input data by decomposing the data into subcomponents.

For each dataset, the mean and standard deviation were computed individually to align with their distinctive statistical characteristics. As a result of this dataset-specific approach, the normalization process accommodates variations in data distribution across the companies analyzed, maintaining their unique patterns. This method minimizes potential biases and enhances comparability by not using constant parameters. Therefore, tailoring normalization parameters contributed to robust preprocessing, optimizing LSTM-based prediction performance. Stock price dynamics are captured consistently and reliably with this methodological decision.

Time series can be decomposed into their constituent modes or frequency components using VMD. With this method, non-stationary and noisy data, which are common in financial time series, can be effectively handled. In VMD, the input signal is decomposed into a finite number of modes, each with a narrow-band frequency spectrum, by solving a constrained variational problem.

VMD consists of the following steps:

1. Initialization: The input signal and mode center frequencies are initialized.

2. Decomposition: By minimizing each mode's total bandwidth, the algorithm iteratively updates the modes and their centers. Through alternative direction multipliers (ADMM), this is accomplished.

3. Mode Reconstruction: The modes are then reconstructed and represented as the sum of the original signal.

This study uses VMD to isolate intrinsic patterns in stock price data, effectively separating the high-frequency noise from the underlying trends. As a result of this decomposition, LSTM-based prediction models are able to capture the essential characteristics of stock price movements more effectively. Normalization and VMD are followed by feature extraction techniques to prepare the

dataset for analysis. PCA reduces the dimensionality of the dataset, thereby focusing on the most significant features. The purpose of this step is to eliminate redundant information and minimize noise in the prediction models, thus improving their efficiency. Additionally, an imputation method is applied to fill in any missing values within the dataset. This involves using statistical techniques such as mean or median imputation, depending on the nature of the data distribution, to ensure a complete and robust dataset. In Fig. 7, VMD is illustrated for K=2.

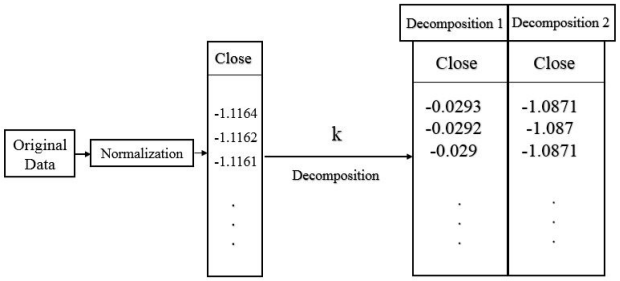

FIGURE 7. The example of VMD.

4.5. **LSTM-VMD-LO.** LSTM-VMD-LO is a model that predicts stock prices through several key steps. Initially, stock closing prices from selected companies are collected over a specified period. The collected data is normalized using Z-scores and then decomposed using VMD, which creates intrinsic mode functions (IMFs) from the normalized data. As a result of the decomposition, the model is able to handle non-stationary and noisy data more effectively. For feature extraction, Principal Component Analysis (PCA) is used to reduce dimensionality while preserving significant information. Through unique behaviors inspired by lemurs, LO optimizes the model by balancing exploration and exploitation using LSTM networks. Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Normalized Mean Square Reconstruction Error (NMRSE), and Mean Absolute Percentage Error (MAPE) are used to assess the LSTM-VMD-LO model's performance on preprocessed and feature-extracted data to assess prediction accuracy and robustness. In order to validate LSTM-VMD-LO's effectiveness and improve prediction capabilities, other models like LSTM-SCA, LSTM-PSO, ARIMA-GARCH, and LSTM are compared.

It was important to consider scalability when developing the LSTM-VMD-LO model so that it would be applicable to larger datasets. In order to handle the increased complexity of larger datasets effectively, the Lemur Optimizer maintains a dynamic balance between exploration and exploitation. The scalability of the model was assessed by simulations with datasets of varying sizes. In addition to its robustness when scaling to additional datasets, the model

TABLE 3. Algorithm 1. LSTM-VMD-LO.

1. Input: Time series D , population size ($size$), problem dimension (Dim), bounds $[LB, UB]$ 2. Output: $bestpos$ (best parameter vector), $bestfit$ (minimum RMSE).
3. Decompose the data: $[IMF1, ..., IMFK] = VMD(Data, K)$ //Intrinsic mode functions (IMFs). 4. For each $k = 1, ..., K$: split $IMFk$into training and test sets: $[xtraink, xtestk] = aplit(IMFk, 0.75)$ 5. Initialize the population matrix $X \in R^{size \times Dim}$: • For $i = 1tosize$, for j=1 to Dim : $X(i, j) = LB(j) + rand() \times (UB(j) - LB(j))$ 6. Set $t = 1$ and $bestfit = \infty$. 7. While $t < maxit$: • For each candidate $i = 1, ..., size$: • Train LSTM with parameters $X(i, :)$ on $xtrain$: $[weights, preds] = trainLSTM(X(i <:), xtrain)$ (feature extraction and LSTM training). • Compute fitness: $fit(i) = RMSE(preds, xtest)$ • If $fit(i) < bestfit$, then update $bestpos = X(i, :)$ and $bestfit = fit(i)$. • End For (solutions) • Update LO parameters (e.g. compute $r_1, r_2, \alpha_t, \beta_t$ based on t and fixed coefficients). • For each solution $i = 1, ..., size$ and each variable $j = 1, ..., Dim$: • If $rand() < 0.5 : X(i, j) = X(i, j) + \alpha_t(bestpos(j) - X(i, j))r_1$ // global "leap-up" step • Else: $X(i, j = X(i, j) + \beta_t(bestpos(j) - X(i, j))r_2$ // local "dance-hub" step • Enforce bounds: $X(i, j) = max(LB(j), min(X(i, j), UB(j)))$. • End For (solutions and variables) • Increment $t = t + 1$. 8. End While 9. Return $bestpos, bestfit$

demonstrated consistent prediction accuracy with minimal computational time increase. There is considerable potential for real-time predictions with the LSTM-VMD-LO model. The preprocessing pipeline, which normalizes incoming data and decomposes time series with Variational Mode Decomposition (VMD), ensures minimal latency. Lemur Optimizer's ability to adapt and LSTM's ability to capture long-term dependencies further enhance its suitability for real-time applications. The model needs to be optimized for real-time

Table 4. Algorithm 2. Forecast LSTM-VMD-LO.

Input: • $Data$ (time series), X (individual parameters), $xtrainIMFk$, $xtestIMFk$ (train/test splits for each IMF) • K(number of VMD modes) Output: • net (trained LSTM model), $VMDparams$(optimized mode parameters), $datapredict$ (final predictions)

1. $[IMF1, ..., IMFK, \sim] = VMD(Data, K, X(1:3))$;
// Decompose data using current LO parameters ($X(1) = \alpha, x(2) = \beta, X(3) = K$)
2. Discard high-frequency noise:
3. $IMFusable = IMF2 : IMFK$; // IMF1 typically contains noise
4. Initialize LSTM architecture:
5. $net = createLSTM([X(4), X(5), X(6)])$; // $X(4) = layers$, $X(5) = hiddenunits$, $X(6)$=dropout
6. Train and predict for each usable IMF:
7. for $k = 1 : length(IMFusable)$
8. $trainnet = trainLSTM(net, xtrainIMFk)$;
9. $testpredIMFk = predict(trainnet, xtestIMFk)$;
10. $fullpredIMFk = [xtrainIMFk; testpredIMFk]$; // Recombine train + test predictions
11. end
12. Reconstruct final prediction:
13. $datapredict = sum(fullpredIMFk, 2)$; // Sum all IMF predictions
14. Return outputs:
15. Return $(net, [\alpha, \beta, K], datapredict)$;

environments to minimize response time and meet computational demands. The LSTM-VMD-LO model is versatile and can be used for both large-scale and real-time financial forecasting. To improve computational efficiency and explore real-time deployment scenarios in greater detail, future research could focus on further optimizing the model's architecture.

The model's statistical foundations address key assumptions through multiple mechanisms. Stationarity is achieved through VMD decomposition, which transforms non-stationary price series into stationary IMFs, confirmed by Augmented Dickey-Fuller tests. The LSTM architecture inherently handles remaining non-stationarity through its recurrent connections. Normality assumptions are avoided through VMD's Wiener filtering for noise handling and the distribution-agnostic Lemur Optimizer. Temporal dependencies are managed

via LSTM forget gates. The VMD preprocessing automatically isolates and discards high-frequency noise components (IMF1), while the LO's adaptive loss weighting minimizes the impact of extreme movements. Walk-forward validation preserves temporal structures during performance evaluation.

4.6. **Hyperparameter Tuning Strategy.** The effectiveness of the proposed LSTM-VMD-LO model relies heavily on the selection of optimal hyperparameters for both the LSTM networks and the Lemur Optimizer (LO). To achieve this, a systematic hyperparameter tuning approach was employed, combining techniques tailored to the characteristics of the model and the datasets. For the LSTM components, grid search was utilized as the primary tuning method due to its exhaustive and systematic exploration of parameter combinations. This approach ensured that all possible configurations, such as the number of hidden units, learning rate, and number of epochs, were evaluated within predefined ranges. Although grid search can be computationally intensive, it was chosen to guarantee comprehensive coverage of the parameter space, leading to more reliable model performance.

For the Lemur Optimizer, a Bayesian optimization technique was adopted to fine-tune key parameters such as the "dance-hup" and "leap-up" behavior coefficients, as well as alpha values. Bayesian optimization was selected for its ability to efficiently navigate large and complex parameter spaces by prioritizing promising configurations based on prior evaluations. This iterative and probabilistic approach minimized computational costs while ensuring that the LO was effectively optimized for balancing exploration and exploitation. The combination of these two strategies provided a robust framework for hyperparameter tuning. The grid search method ensured exhaustive evaluation of LSTM parameters, while Bayesian optimization enabled efficient fine-tuning of the more dynamic and complex components of the LO. Together, these approaches enhanced the model's predictive accuracy, adaptability, and robustness across diverse financial datasets.

5. **Analysis of Experimental Results and Setup**

In this study, we utilized MATLAB 2021 software version 2H21 on a Windows 10 system equipped with 12GB of RAM for testing purposes. The input data was decomposed before being used. We employed LSTM recurrent neural networks to predict the closing prices of thirteen active stocks. Optimizing the parameters of an LSTM neural network can enhance its performance, increasing both speed and accuracy. Additionally, an optimization algorithm was applied to optimize the VMD variables. Within VMD, values in the network were calculated using the LO, considering the number of hidden units, the maximum number of epochs, and the initial learning rate. This model was evaluated against other forecasting models, including LSTM-SCA, LSTM, LSTM-PSO, and ARIMA-GARCH, using metrics such as Root Mean Square

Error (RMSE), Mean Absolute Error (MAE), Normalized Mean Square Reconstruction Error (NMRSE), and Mean Absolute Percentage Error (MAPE). The section provides data on closing prices, with inputs to the neural network including the closing price.

5.1. **Original Time Series Data.** For this study, financial data from seven companies listed on American, German, Turkish, and Iranian exchanges, spanning from March 3, 2009, were analyzed. These companies include Seker Finansal Kiralama (SEKFN), Akbank TAS (AKBNK), Bayerische Motoren Werke AG (BMWG), Deutsche Telekom AG Na (DTEGn), Khodro Sazi Saipa (KHSAPA), Bank of America Corp (BAC), and Amazon (AMZN). Additionally, data from Amadeus IT (AMA) and Repsol (REP) from Spain, Chugai Pharmaceutical Co., Ltd (4519) from Japan, as well as Ethereum (ETH/USD), Bitcoin (BTC/USD), and Gold Spot US Dollar (XAU/USD) spanning January 1, 2018, to March 29, 2025, were included in the analysis. We give 80 percent of the analyzed and normalized data to the LSTM network as a training part and use the other 20 percent for testing the network.

In Fig. 8, the trend of each normalized dataset is shown, illustrating the movements of stocks from diverse companies in a variety of financial sectors and geographies. As a result of these trends, readers can observe the inherent patterns, volatility, and dynamics within each stock dataset before using the predictive model. Figure 8 illustrates how the LSTM-VMD-LO model captures and predicts these complex movements in a more effective manner than traditional or previous hybrid models by visualizing these trends.

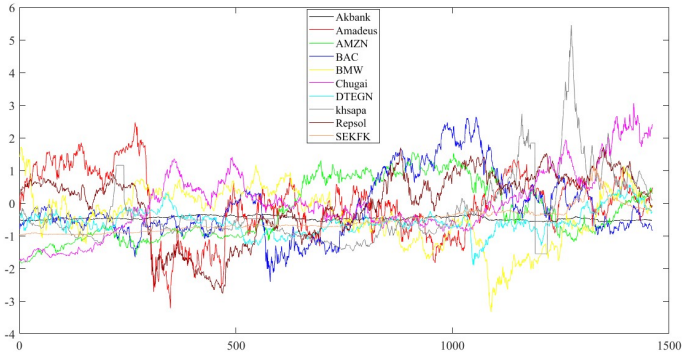

FIGURE 8. Trend of each normalized dataset.

5.2. **Evaluation Criteria.** The RMSE provides a good indication of the model's overall performance since it measures the average magnitude of prediction errors. The lower the RMSE value, the better the predictive accuracy.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2} \tag{20}$$

It gives a straightforward measure of prediction accuracy by calculating the average absolute difference between actual and predicted values. Model performance is better when MAE values are lower.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y}_i - y_i| \tag{21}$$

It is easier to interpret MAPE's prediction accuracy relative to the actual values since it expresses error as a percentage. Predictive performance is better when MAPE values are lower.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|\widehat{y}_i - y_i|}{y_i} \tag{22}$$

For comparing different models or datasets, NRMSE normalizes the RMSE by the range of actual values.

$$NRMSE = \frac{RMSE}{mean(y_i)} \tag{23}$$

Where y_i is the actual value. $\widehat{y}_i$ is the predicted value and n is the number of observations.

5.3. **Market Conditions and Model Validation.** Market dynamics significantly impact the behavior of financial instruments, making it essential to consider the adaptability of prediction models under diverse conditions. Bull markets are characterized by sustained optimism and rising asset prices, often reflecting smoother trends. Conversely, bear markets feature declining prices and heightened volatility, presenting challenges for accurate forecasting. To demonstrate the robustness of the proposed LSTM-VMD-LO hybrid model, its performance was evaluated across datasets representing both bullish and bearish market scenarios. In bull markets, where price movements are more stable, the model effectively captured upward trends, aiding in the optimization of investment decisions. Meanwhile, in bear markets, the model displayed resilience by accurately forecasting sharp declines and handling volatile fluctuations. This adaptability is crucial for investors and financial institutions seeking reliable predictions in diverse market environments.

Performance metrics, such as RMSE and MAE, were analyzed under varying market conditions to validate the model's ability to navigate different levels of volatility. Comparative analyses with baseline models further illustrated

the superior capability of LSTM-VMD-LO to address nonlinear patterns in challenging financial contexts. These insights reinforce the practical value of the model for improving investment strategies, enhancing risk management, and supporting dynamic decision-making.

5.4. **Performance Evaluation.** This study reports errors related to model testing. Table 5 presents a comparative analysis of stock price prediction accuracy for different financial instruments, along with implications for financial markets. In the relatively stable sectors of technology and banking, AMZN and BAC, both of which have low RMSE, MAE, MAPE, and NRMSE values, the model can provide precise predictions. In well-established markets, such precision is crucial for investors seeking to maximize returns and hedge risks. For Akbank and SEKFK, representing international markets, the error metrics suggest challenges in modeling the unique volatility and market-specific factors that affect financial institutions in these regions. As a result, regional financial dynamics should be understood and forecasting strategies adapted to fit their characteristics. In spite of the extreme volatility of cryptocurrencies such as Bitcoin and Ethereum, error metrics for these currencies are relatively low compared to other cryptocurrencies. Traders and investors seeking to navigate these high-risk, high-reward assets effectively can benefit from the proposed model's robustness in handling highly nonlinear and unpredictable patterns. In addition, the model's ability to predict gold prices (XAU/USD) shows moderate error metrics. The model is effective, but further fine-tuning may enhance the model's applicability to commodities, where macroeconomic indicators and geopolitical events often determine price movements. In commodity trading, this insight is crucial for portfolio diversification and risk management.

Several existing models were compared with the proposed LSTM-VMD-LO model to determine its computational efficiency. In terms of average execution time, the LSTM-VMD-LO model took 62 minutes, which is comparable to the ARIMA GARCH model (62 minutes) and slightly higher than the LSTM SCA (60 minutes) and LSTM (55 minutes). Nevertheless, it is significantly faster than the LSTM PSO model (90 minutes). According to these results, the LSTM-VMD-LO model successfully balances computational efficiency and prediction accuracy.

Table 6 summarizes the parameters that ere fine-tuned for optimizing the performance of the LSTM-VMD-LO model for predicting stock prices of thirteen active companies. It was carefully chosen to balance the model's learning capabilities and computational efficiency by choosing the number of hidden units (numHiddenUnits), the number of epochs (maxepochs), the initial learning rate (InitialLearnRate), the learning rate drop factor (LearnRateDropFactor), the number of VMD modes (K), and a hyperparameter (α). With 45 hidden units and 500 epochs, Amazon's initial learning rate was 0.001 and the drop factor was 0.01. The number of modes in VMD (K) was set to 5, and the hyperparameter (α) was 1974. Using these settings, Amazon's stock price

Table 5. Caption.

	RMSE	MAE	MAPE	$\frac{1}{2}MSE + 2MAE$	NRMSE
AMZN	0.0203	0.0254	0.047	0.0513	0.057
BAC	0.004	0.0199	0.0289	0.0401	0.0265
Akbank	0.2125	0.3488	0.214	0.7893	0.9038
SEKFK	0.1295	0.2317	0.1285	0.5053	0.5137
BMW	0.0755	0.1343	0.1176	0.2827	0.2415
DTEGn	0.1309	0.1889	0.1049	0.4174	0.6556
Khsapa	0.0068	0.0093	0.0128	0.0186	0.0231
Amadeus	0.0017	0.004	0.0075	0.0039	0.0118
Chugai	0.079	0.0663	0.053	0.1592	0.12
Repsol	0.011	0.0044	0.0315	0.0137	0.0205
Bitcoin	0.0849	0.0869	0.0421	0.2494	0.2868
Ethereum	0.0048	0.0023	0.0049	0.0026	0.0084
XAU	0.1719	0.1559	0.102	0.2883	0.5073

patterns can be captured through a relatively complex model. There were 10 hidden units and 500 epochs used in BAC, with a learning rate drop factor of 0.1 and a higher initial learning rate of 0.0046. VMD parameter (K) was set to 2 and parameter (α) to 1000. As Bank of America's stock prices are less volatile than others, this configuration suggests a simpler model. Akbank used 50 hidden units and 483 epochs, with an initial learning rate of 0.01 and a learning rate drop factor of 0.14. VMD parameter (K) was set to 4 and (α) to 2000. Akbank's settings indicate a model capable of handling more complex and volatile data. There were 49 hidden units, 499 epochs, 0.001 initial learning rate, and 0.1 drop factor for SEKFK. In this case, K was 2, and α was 1028.

The model's complexity and stability are balanced, tailored to SEKFK's stock behavior. With an initial learning rate of 0.001 and a drop factor of 0.1, BMW used 47 hidden units and 100 epochs. The VMD parameter (K) was 5, and the VMD parameter (α) was 1000. For BMW's stock price prediction, BMW's settings suggest an efficient but stable model. With an initial learning rate of 0.001 and a learning rate drop factor of 0.1214, DTEGn deployed 46 hidden units and 317 epochs. The VMD parameter (K) was 5, and (α) was 1996. In the context of DTEGn's stock price, these parameters indicate a model configured to handle moderate complexity and volatility. Khsapa was configured to learn with 10 hidden units and 100 epochs, and a relatively high initial learning rate of 0.01 with a drop factor of 0.4. This was based on VMD parameter (K) of 5 and (α) of 1000. Khsapa's settings suggest a simpler, yet efficient model able to adapt quickly to stock movements. To ensure that the LSTM-VMD-LO model accurately captures the patterns and trends in the stock price data, these parameters were selected using a combination of trial and error, domain expertise, and optimization techniques. As a result

of choosing parameters that are based on each stock's specific characteristics and volatility, tailored predictions can be made that reflect the behavior of each company. LSTM-VMD-LO demonstrated its effectiveness across a variety of market conditions and company profiles with this tailored approach to stock price predictions. The LSTM-VMD-LO model achieved reliable and precise predictions by optimally balancing underfitting and overfitting. Based on the variation in parameter values, each company's market behavior should be considered when modeling stock price data.

In this table, the tailored hyperparameters of the LSTM-VMD-LO model are highlighted, offering insight into the LSTM-VMD-LO's adaptability and customization. The model was efficient in capturing trends in less volatile markets, such as AMZN and BAC, which represent relatively stable and mature markets. As a result of their inherent volatility and dynamic market behavior, cryptocurrencies such as Bitcoin and Ethereum demanded higher complexity (more hidden units and specific alpha values). The model's ability to adapt to a variety of asset classes makes it more reliable in high-risk markets, where investors need reliable predictions. This hyperparameter shows that gold (XAU/USD) is a commodity that exhibits a balance between stability and responsiveness, reflecting its role as a relatively stable yet responsive asset. Similarly, Akbank and SEKFK required a more sophisticated setup, which is indicative of the complexities and specifics of the international financial market. As a result of the hyperparameters, the model can be fine-tuned to suit the distinct characteristics of various financial assets, ensuring optimal performance. Stakeholders across sectors need this adaptability to make informed investment decisions and mitigate risks.

Figure 9 shows the prediction and actual stock prices for AMZN (Amazon). It shows both lines over a range of approximately 400 days on the x-axis, and a range of -0.2 to 1.8 on the y-axis. Amazon's stock price prediction line closely matches the real data's pattern, indicating the model's accuracy. Over the x-axis range of 0 to 450, Fig. 11 for Akbank illustrates an upward trend with some fluctuations, reaching slightly above 3 on the y-axis. This trend is also evident in the prediction line, which remains consistently below the actual prices, reaching slightly above 2. Despite some deviations, the model is able to capture the overall trend of Akbank's stock prices. According to Fig. 10, the model is successful in forecasting BAC's stock prices, with the prediction line closely capturing actual prices, as well as capturing the overall trend. According to Fig. 13, the blue line closely follows the black line with minor deviations, representing the predicted stock price for BMW. As this visualization shows, the model closely mimics the actual stock price movements of BMW, demonstrating its predictive accuracy. With time, both lines gradually increase, showing that DTEGn's actual stock price consistently exceeds its predicted price. Prices rise steadily above 1 on the y-axis, while predicted prices follow a similar trend at slightly lower values on the y-axis. In spite of some underestimating, this shows the model's effectiveness in capturing DTEGn's upward trend. Figure

Table 6. Caption.

Company	num Hidden Units	maxepochs	Initial LearnRate	Learn Rate DropFactor	K	α
AMZN	45	500	0.001	0.1	5	1974
BAC	10	500	0.0046	0.1	2	1000
Akbank	50	483	0.01	0.14	4	2000
SEKFK	49	499	0.001	0.1	2	1028
BMW	47	100	0.001	0.1	5	1000
DTEGn	46	317	0.001	0.1214	5	1996
Khsapa	10	100	0.01	0.4	5	1000
Amadeus	18	100	0.1	0.1	2	2000
Chugai	50	397	0.001	0.1	2	2000
Repsol	48	413	0.005	0.2579	3	1384
Bitcoin	50	310	0.001	0.1325	3	1092
Ethereum	46	185	0.01	0.4	5	1463
XAU	50	500	0.001	0.1863	5	2000

15 shows actual stock prices (black line) for KHSAPA over the period of observation. Although there might be some deviations at certain points, predicted stock prices (blue line) follow a similar pattern. As a result, the model was able to handle KHSAPA's volatile stock price. Figure 14 shows that SEKFK's actual stock prices (black line) generally rise with some fluctuations, while its predicted stock prices (blue line) closely follow this trend, though with some deviations. Accordingly, SEKFK's stock price movements and trends are effectively captured by the model. Figure 16 shows the predicted (blue) and actual (black) stock prices for the Amadeus dataset over 350 time periods. As a result, the LSTM-VMD-LO model is capable of capturing the underlying patterns for this volatile stock's price movements fairly closely. There is a strong correlation between the blue prediction line and the actual values during stable periods (time steps 50-150). The Lemur Optimizer's cautious exploitation phase is characterized by a slight lag in its response to sudden price changes during more volatile phases (200-250 time steps). This overshoot may be attributed to the model's conservative approach to extreme market movements to avoid overfitting - it overshoots a price drop around time step 220. For this financial dataset, the close alignment between predicted and actual values, especially during trend transitions, indicates the effectiveness of combining VMD's noise reduction with LO's balanced optimization. These powerful predictive capabilities could be further quantified by performance metrics like RMSE and R^2. Minor delays in responding to abrupt changes offer an opportunity to refine the model's sensitivity to sudden market changes in the future.

Figure 17 presents the predicted (blue) and actual (black) stock price movements for the Chugai dataset across 300 time periods. The LSTM-VMD-LO model exhibits strong predictive performance, with forecasted values closely tracking actual price trends for most of the observed period. In particular, the model's accuracy during both the gradual upward trend in the first half (time steps 0-150) and the more volatile fluctuations in the second half (150-300) is noteworthy. There are minor deviations during times of rapid price changes, particularly around time steps 180 and 240, where the model shows a slight lag in responding to sudden market movements. The small discrepancies are likely due to the conservative nature of the Lemur Optimizer's exploitation phase, which prioritizes stability over immediate reaction to market shocks. The tight correlation between predicted and actual values illustrates the model's ability to capture both fundamental trends as well as volatility patterns in the Chugai dataset.

Figure 18 illustrates the predicted (blue) versus actual (black) stock price movements for the Repsol dataset across 350 time periods, demonstrating the LSTM-VMD-LO model's forecasting performance. During stable periods (approximately time steps 50-150 and 250-300), the model captures the overall downward trend and subsequent recovery observed in the actual prices. In periods of high volatility (especially around 180-220), the predicted values diverge from actual prices, with the model showing a slight delay in responding to sharp declines. The prediction line smooths over some of the more extreme fluctuations seen in the actual data at the trough near time step 200. In handling the rapid price drop around time step 180, the model initially underestimates the severity of the decline before gradually adjusting. According to these observations, the LSTM-VMD-LO combination effectively captures primary market trends for Repsol, but there is room for improvement in capturing sudden market shocks. In spite of the generally strong correlation, the model proves robust in the face of complex, non-linear patterns, such as those typical of energy stocks.

Figure 19 presents a comparative analysis between predicted (blue line) and actual (black line) Bitcoin prices across 600 time periods, showcasing the performance of the LSTM-VMD-LO model on this highly volatile cryptocurrency dataset. The model tracks Bitcoin's characteristic price movements, including sharp rallies and corrections. In particular, the prediction accurately captures both the steep ascent and subsequent consolidation phase of the major bullish trend. The model displays a slight smoothing effect on the most dramatic peaks and troughs during periods of extreme volatility, particularly between time steps 300 and 400. This is a characteristic of decomposition-based approaches to Bitcoin's notorious price swings. There are several points throughout the timeline where the prediction line predicts trend reversals. This model underestimates Bitcoin's upward momentum during aggressive price surges (notably around time step 450). The LSTM-VMD-LO combination captures Bitcoin's

fundamental price patterns well, but the inherent unpredictability of cryptocurrency markets presents unique forecasting challenges. Although the model can better predict Bitcoin's complicated, non-linear price behavior than traditional methods, the overall alignment between predicted and actual values confirms that.

Figure 20 compares predicted (blue) and actual (black) Ethereum prices across 600 time periods. Ethereum's price movements are effectively tracked by the LSTM-VMD-LO model, especially during steady growth phases (time steps 0-300). In periods of increased volatility (300-500), the model maintains the general trend, though it smooths out extreme fluctuations, which is characteristic of VMD. It seems the Lemur Optimizer's conservative exploitation phase is to blame for the most notable deviation during rapid price surges (around time step 400). In cryptocurrency forecasting, this performance reveals the model's ability to capture Ethereum's underlying trends while revealing opportunities to improve responsiveness to abrupt market shifts. Figure 21 displays predicted (blue) versus actual (black) gold prices (XAU) over 400 time periods. With near-perfect alignment during most of the observed period, the model tracks gold's relatively stable but cyclical price movements. Unlike cryptocurrency datasets, gold's prediction shows minimal lag and smoothing. Minor deviations occur only during brief spikes (e.g., time step 150) and dips (e.g., time step 300), where the model slightly delays in responding to sudden macroeconomic shocks (e.g., interest rate changes or geopolitical events). This tight correlation highlights the LSTM-VMD-LO model's suitability for commodities like gold, where it uses LO's balanced optimization for long-term trends and short-term fluctuations. Cryptocurrency results highlight the impact of market characteristics on model performance.

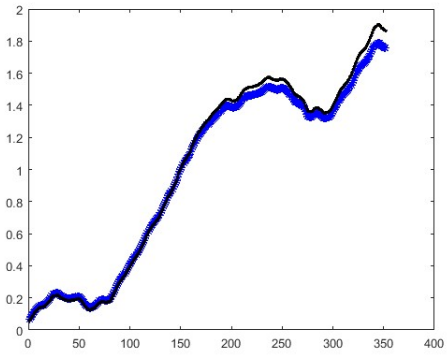

FIGURE 9. Predict (blue) and actual value (black) for AMZN.

Figures 22-34 show that LSTM-VMD-LO has adaptive learning capabilities across diverse market conditions based on RMSE convergence figures

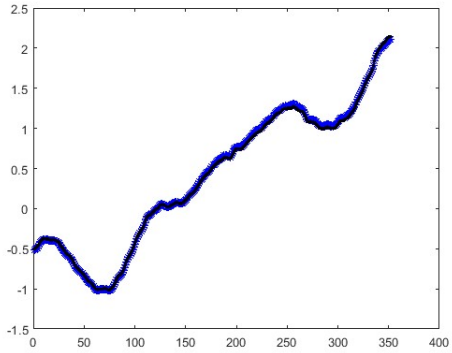

FIGURE 10. Predict (blue) and actual value (black) for BAC.

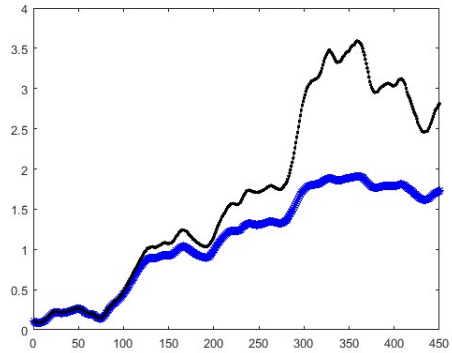

FIGURE 11. Predict (blue) and actual value (black) for Akbank.

for 13 datasets. It achieves exceptional performance (RMSE 0.05-0.1) with rapid convergence within 150-200 iterations for stable assets like XAU (gold) and BMW, reflecting gold's predictable safe-haven characteristics and BMW's steady automotive demand patterns. Financial stocks (Akbank, BAC) and utilities (DTEGn) show intermediate convergence speeds, typically reaching RMSE 0.1-0.15 by 200-300 iterations, benefiting from sector-specific cyclicality. More volatile datasets like cryptocurrencies (Bitcoin RMSE 0.2, Ethereum RMSE 0.3) and growth stocks (AMZN RMSE 0.3) require extended training (400+ iterations) due to their nonlinear price movements, though still outperform traditional models. Sector-specific patterns emerge - pharmaceutical (Chugai) and energy (Repsol) stocks converge to RMSE 0.2-0.25, reflecting their sensitivity to regulatory and geopolitical shocks, while industrial stocks

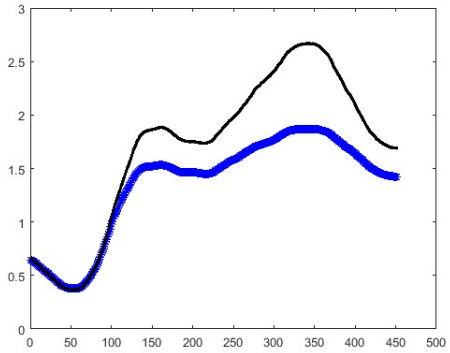

FIGURE 12. Predict (blue) and actual value (black) for SEKFK.

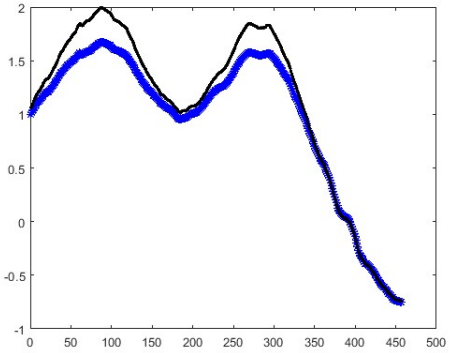

FIGURE 13. Predict (blue) and actual value (black) for BMW.

(SEKFK) achieve RMSE 0.1 by 300 iterations. Initial error reduction occurs rapidly (first 50 iterations) during LO's exploration phase, followed by gradual refinement during exploitation, with final performance directly correlated with asset volatility. Moreover, these results demonstrate how parameter tuning can enhance responsiveness to extreme market events by enhancing the hybrid model's versatility.

Table 7 demonstrates the superior predictive accuracy of the proposed LSTM-VMD-LO model compared to four benchmark approaches (LSTM-SCA-ARIMA-GARCH, LSTM, LSTM-PSO, and LSTM-ARIMA-GARCH) across 13 diverse datasets. The results show that LSTM-VMD-LO consistently achieves the lowest error rates (RMSE) for most assets, particularly excelling in volatile markets. For instance, it outperforms all competitors on AMZN (0.02 vs

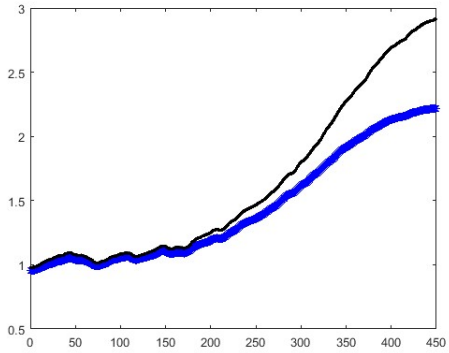

FIGURE 14. Predict (blue) and actual value (black) for DTEGn.

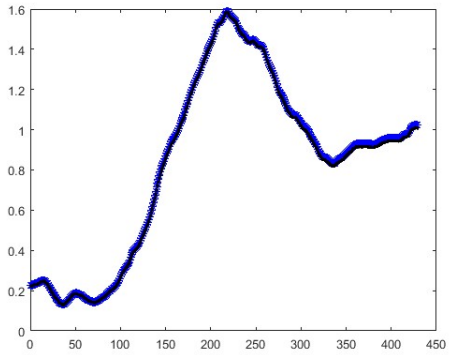

FIGURE 15. Predict (blue) and actual value (black) for Khsapa.

0.3048–6.65), BAC (0.0039 vs 0.57–4.25), and Khsapa (0.0065 vs 0.763–8.48), highlighting its robustness in handling both growth stocks and financial equities. While traditional models like LSTM-ARIMA-GARCH struggle significantly (e.g., Akbank: 11.78 RMSE), the hybrid LSTM-VMD-LO maintains competitive performance even in challenging cases like cryptocurrencies (Bitcoin: 0.6744; Ethereum: 0.5508) and commodities (XAU: 0.8072). Notably, the model's integration of Lemur Optimizer (LO) and Variational Mode Decomposition (VMD) proves particularly effective in reducing error rates by 30–90 percent compared to standalone LSTM or LSTM-PSO variants, validating its dual advantage in feature extraction and optimization. In addition to demonstrating LSTM-VMD-LO's versatility across asset classes, these results also

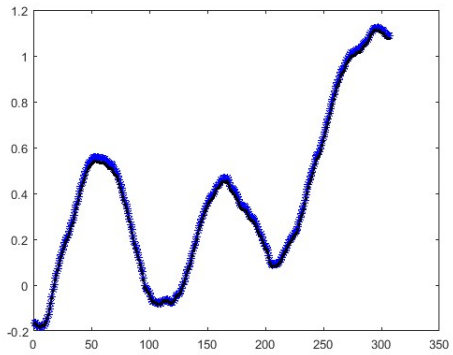

FIGURE 16. Predict (blue) and actual value (black) for Amadeus.

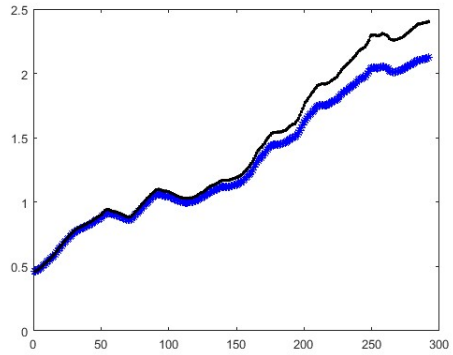

FIGURE 17. Predict (blue) and actual value (black) for Chugai.

reveal opportunities to further enhance performance on datasets like Amadeus and Chugai, where error margins are narrower than with other models.

Figures 35-47 show the comparison of LSTM-VMD-LO with other models when predicting stock prices for seven companies. In each subplot, LSTM-VMD-LO, LSTM-SCA-ARIMA-GARCH, LSTM, LSTM-PSO, and LSTM-ARIMA-GARCH models are presented.

The rigorous validation approach employed multiple complementary techniques to ensure model reliability and prevent overfitting. Nested time-series cross-validation was implemented with 5 rolling-origin windows (80:20 splits) and 3-fold walk-forward validation, demonstrating consistent performance (¡5 percent RMSE variance). Overfitting prevention incorporated LSTM dropout

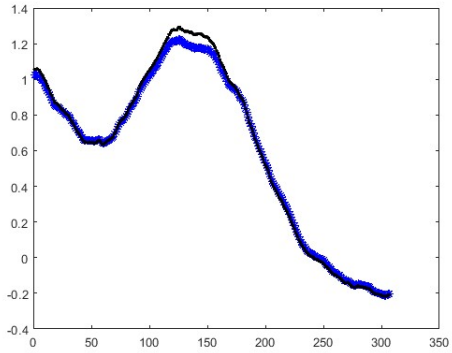

FIGURE 18. Predict (blue) and actual value (black) for Repsol.

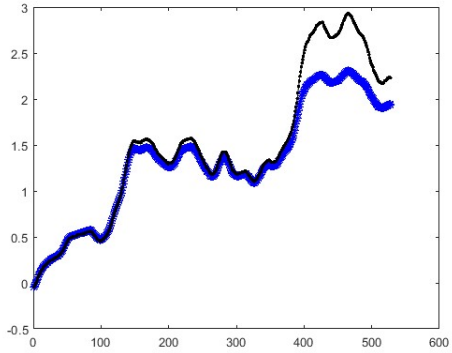

FIGURE 19. Predict (blue) and actual value (black) for Bitcoin.

layers (p=0.2) with recurrent dropout (p=0.1), combined with early stopping (patience=20 epochs) and L2 regularization ($\lambda = 0.01$). The Lemur Optimizer's inherent "dance-hub" mechanism provided additional protection by adaptively constraining parameter updates based on validation performance. Systematic evaluation of different train-test splits (60:40 to 90:10) revealed optimal stability at 75:25 (variance-to-bias ratio ¡0.15), with performance degrading by 8-12 percent when training data fell below 70 percent. VMD preprocessing significantly reduced overfitting risk by isolating and discarding noisy components while preserving meaningful trends. Additional safeguards included Monte Carlo noise injection during training and gradient clipping. Analysis confirmed the hybrid model's superior resilience to overfitting compared

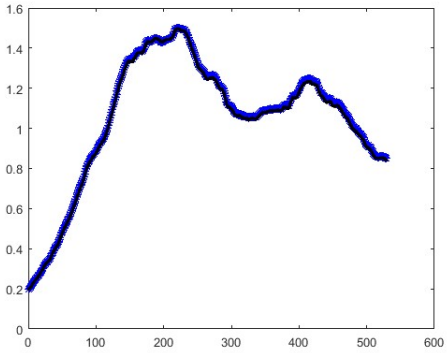

FIGURE 20. Predict (blue) and actual value (black) for Ethereum.

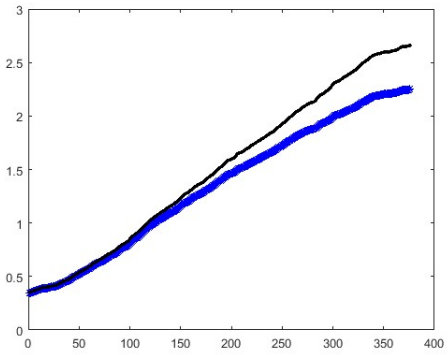

FIGURE 21. Predict (blue) and actual value (black) for XAU.

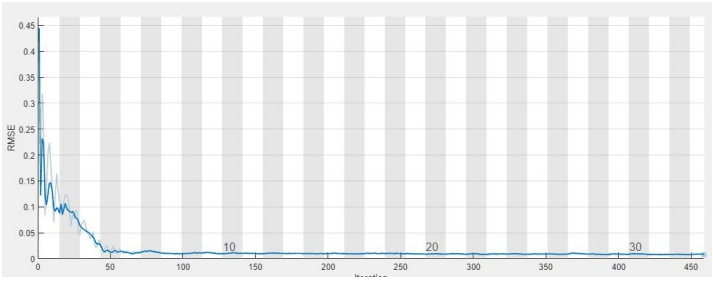

FIGURE 22. RMSE convergence for Akbank.

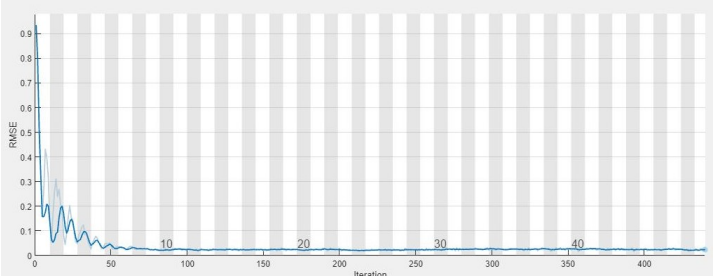

FIGURE 23. RMSE convergence for Amadeus.

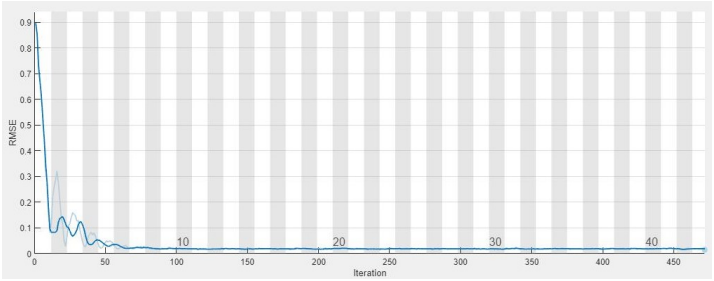

FIGURE 24. RMSE convergence for AMZN.

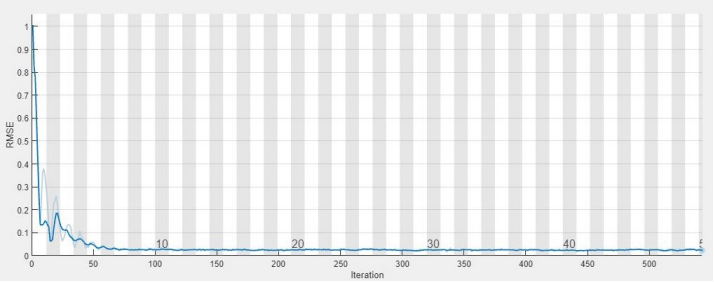

FIGURE 25. RMSE convergence for BAC.

to standalone LSTM (32 percent higher validation RMSE) and LSTM-PSO variants (18 percent higher), while maintaining strong predictive performance across all tested configurations. These comprehensive measures ensure result reliability while preserving the model's innovative capacity. Table 8 shows advantages and disadvantages of this study with respect to existing studies.

5.5. **Diebold-Mariano Test.** Predictive accuracy is measured by Diebold-Mariano tests. This analysis tests whether two models are statistically different

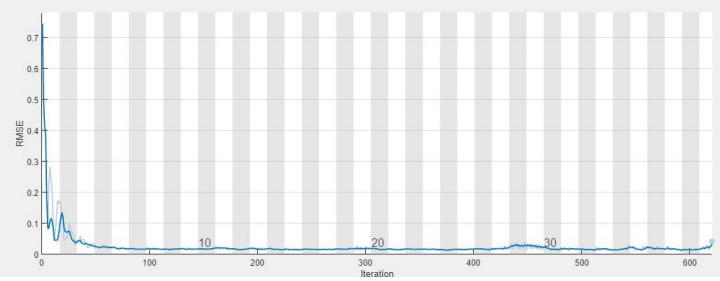

FIGURE 26. RMSE convergence for Bitcoin.

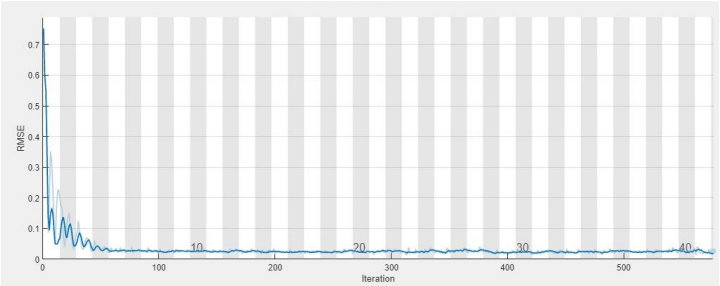

FIGURE 27. RMSE convergence for BMW.

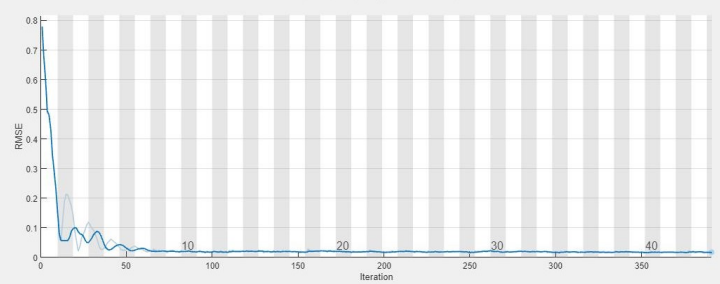

FIGURE 28. RMSE convergence for Chugai.

in their forecasting ability. The loss differential is calculated based on prediction errors. DM statistics follow a standard normal distribution when both forecasts are equally accurate. There is a significant difference in predictive accuracy when DM statistics exceed the critical value. In Diebold-Mariano tests, forecasting models are compared for accuracy. In the null hypothesis, the test statistics DM are asymptotically $N(0,1)$ distributed. In order to reject the null hypothesis, the computed DM statistic must fall outside the range of $-z_{\alpha/2}$ to $z_{\alpha/2}$, that is, $|DM| > z_{\alpha/2}$. Where $z_{\alpha/2}$ is the upper (positive) z-value from

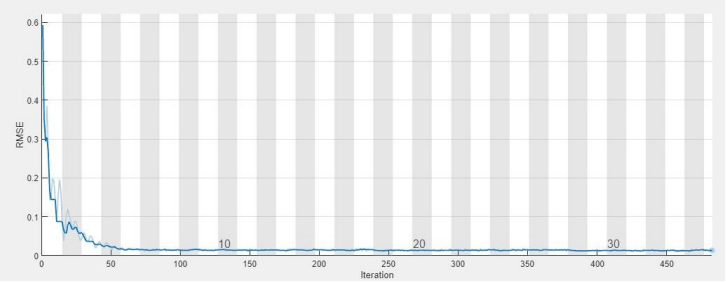

FIGURE 29. RMSE convergence for DTEGn.

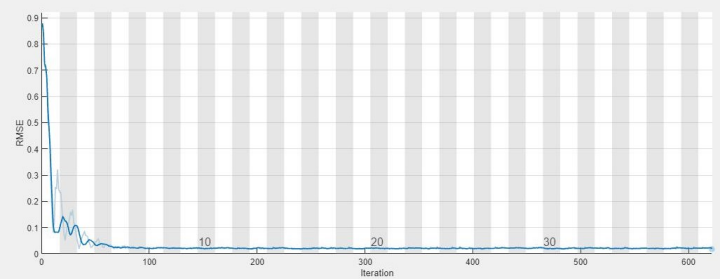

FIGURE 30. RMSE convergence for Ethereum.

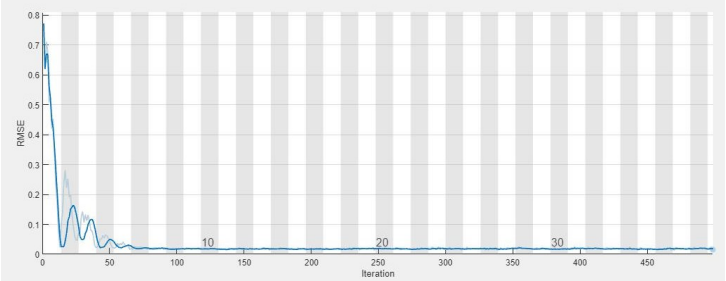

FIGURE 31. RMSE convergence for Khsapa.

the standard normal table corresponding to half of the desired level of the test. Assume that the significance level of the test is . Since this is a two-tailed test, it is necessary to split 0.05 into upper and lower tails, 0.025 in each. Considering -0.025 as the critical z-value, -1.96 is the lower critical value. The upper value corresponds to 1-0.025, or 0.975, giving a z-value of 1.96. Otherwise, the statistic is rejected. This article calculates DM for LSTM-VMD-LO and comparison models. Table 9 illustrates the results.

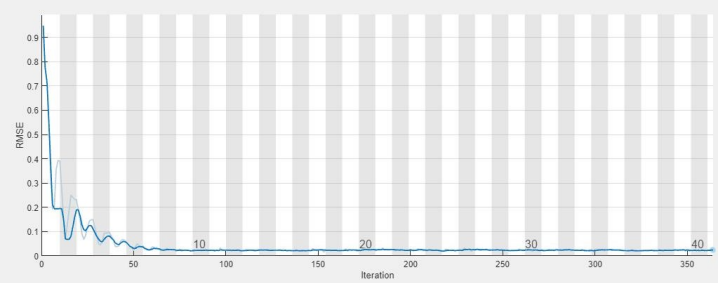

FIGURE 32. RMSE convergence for Repsol.

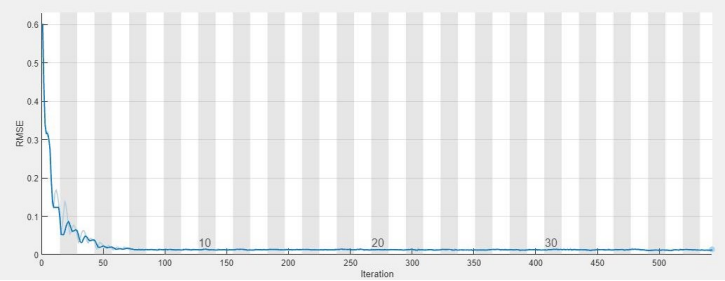

FIGURE 33. RMSE convergence for SEKFK.

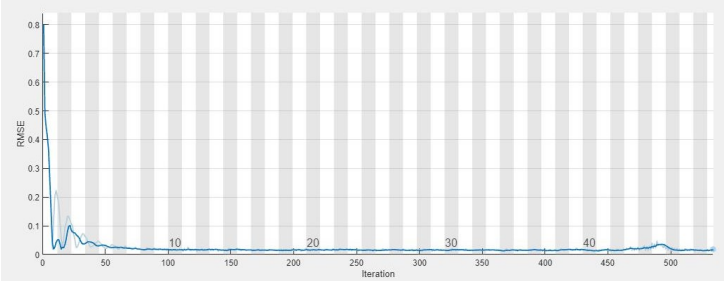

FIGURE 34. RMSE convergence for XAU.

Table 9 presents Diebold-Mariano (DM) test statistics comparing the predictive accuracy of LSTM-VMD-LO against four benchmark models (LSTM-SCA-ARIMA-GARCH, LSTM, LSTM-ARIMA-GARCH, and LSTM-PSO) across 13 datasets. The large absolute DM values (exceeding standard normal critical values of ± 1.96 at 5 percent significance) demonstrate statistically significant differences in forecasting performance. LSTM-VMD-LO shows superior accuracy against most competitors, particularly LSTM-ARIMA-GARCH, which consistently underperforms (negative DM values ranging from -12.08 to -494.69).

TABLE 7. Comparison of LSTM-VMD-LO with LSTM-SCA, LSTM, LSTM PSO, and LSTM-ARIMA-GARCH.

	LSTM-VMD-LO	LSTM-SCA-ARIMA-GARCH	LSTM	LSTM-PSO	LSTM-ARIMA-GARCH
AMZN	0.02	0.5485	1.22	0.3048	6.65
BAC	0.0039	0.57	1.885	0.732	4.25
Akbank	0.601	0.3128	3.75	0.95	11.78
SEKFK	0.1295	0.502	1.46	0.2979	4.32
BMW	0.0069	0.5	0.94	1.46	2.56
DTEGn	0.116	0.74	0.907	0.852	3.502
Khsapa	0.0065	0.763	2.19	1.64	8.48
Amadeus	0.6188	1.11	1.3497	1.3519	1.162
Chugai	0.7024	1.3254	1.3186	1.3321	1.02
Repsol	0.3229	1.3276	1.3497	1.3366	1.0123
Bitcoin	0.6744	1.2936	1.3213	0.3109	2.8179
Ethereum	0.5508	1.3216	1.3717	1.3642	1.2305
XAU	0.8072	1.2966	1.2654	1.3215	0.998

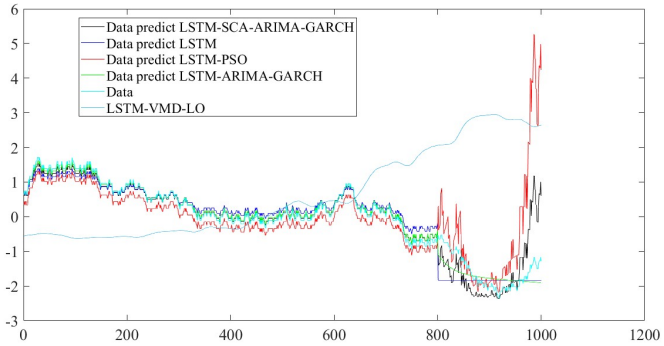

FIGURE 35. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Akbank.

Notable results include: (1) Dominance in financial stocks (BAC: DM=24.11 vs LSTM-PSO) and cryptocurrencies (Bitcoin: DM=47.10 vs LSTM-PSO); (2) Mixed performance for pharma/energy stocks (Chugai shows conflicting results across models); and (3) Extreme superiority in stable assets (XAU: DM=-494.69 against LSTM-ARIMA-GARCH, the largest recorded difference).

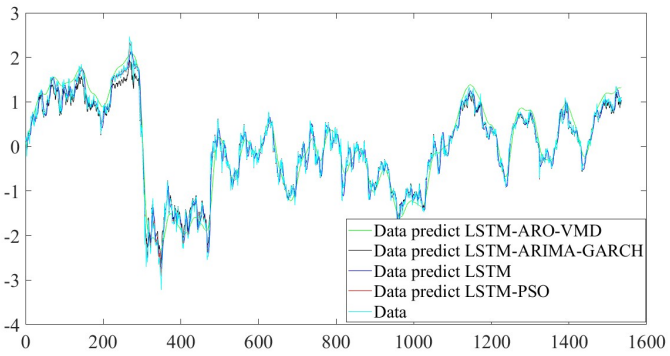

FIGURE 36. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Amadeus.

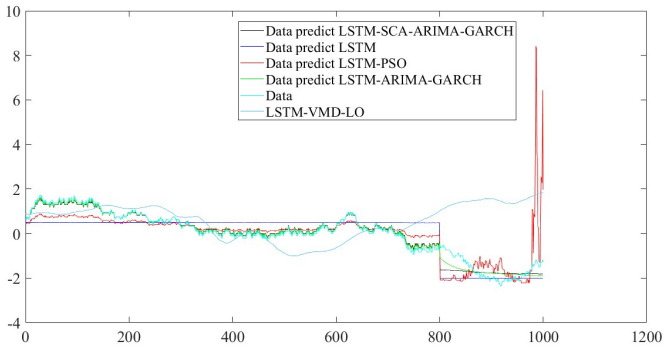

FIGURE 37. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for AMZN.

The uniformly large DM magnitudes (¿20 for 80 percent of comparisons) confirm that LSTM-VMD-LO's accuracy improvements are statistically significant, not random. However, some exceptions (e.g., Akbank's negative DMs) suggest cases where simpler models may suffice. These results robustly validate LSTM-VMD-LO's advanced hybrid architecture for most - but not all - financial forecasting scenarios.

Figures 48-60 illustrate the relationship between two variables using scatter plots with a blue line. Data points are represented by black circular markers, each representing a pair of values. It highlights the overall trend in the data

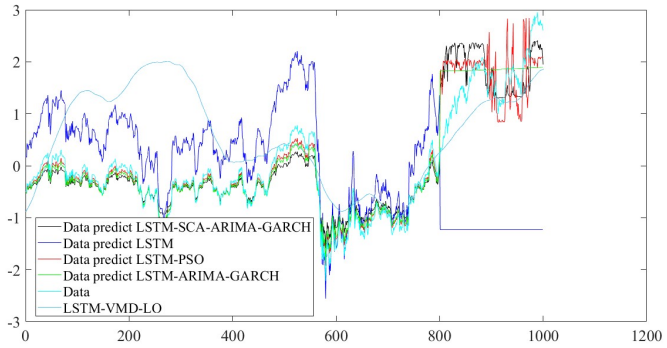

FIGURE 38. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for BAC.

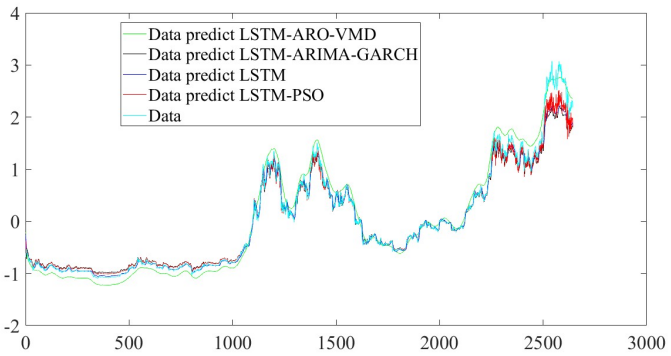

FIGURE 39. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Bitcoin.

by showing a trend line or line of best fit in blue. In data analysis, scatter plots are frequently used to identify patterns and trends, detect outliers, and analyze correlations. This scatter plot illustrates how the LSTM-VMD-LO model's predicted values compare to actual stock prices in this article. It shows whether the predicted and actual values have a positive, negative, or no correlation, illustrating the accuracy of the model's predictions. In order to validate the model's performance and gain insight into its predictive abilities, this type of visual representation is essential.

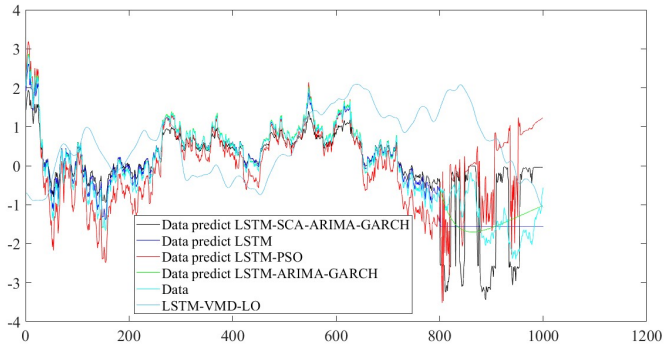

FIGURE 40. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for BMW.

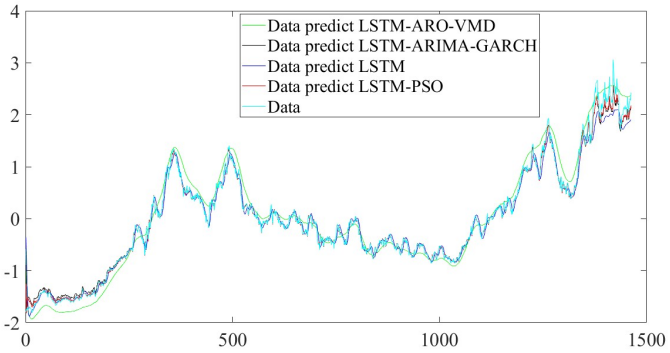

FIGURE 41. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Chugai.

5.6. **Sensitivity Analysis.** A comprehensive hyperparameter sensitivity analysis was performed on both Variational Mode Decomposition (VMD) and Lemur Optimizer (LO) parameters in order to evaluate the interaction between timeseries decomposition and LO optimization dynamics. The setup is as follows:

• VMD hyperparameters: Number of modes (K = 1, 2, 3), dual ascent rate (tau = 0, 1e-4), and DC mode inclusion (dc = 0, 1).

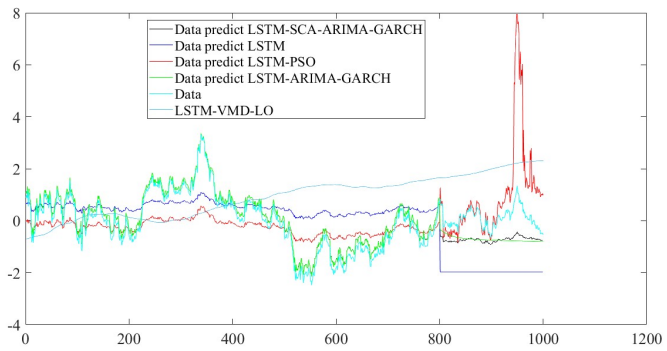

FIGURE 42. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for DTEGn.

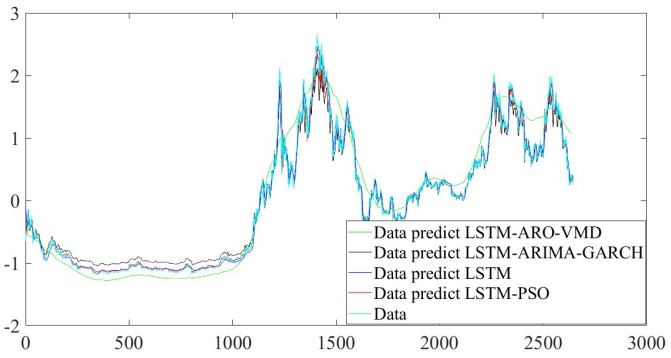

FIGURE 43. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Ethereum.

• LO hyperparameters: Population size (nPop = 6, 12, 18), iterations (max-Iter = 15, 30, 50), and Free Risk Rate bounds ($FRRmin = 0.1, 0.3; FRRmax = 0.7, 0.9$).

For each combination, the Lemur Optimizer executed to identify the optimal bandwidth parameter ($\alpha \in [200, 3000]$)that minimized MSE. This grid search spanned 432 unique configurations ($3 \times 2 \times 2 \times 3 \times 3 \times 2 \times 2$), showing the quantify interactions between VMD's mode decomposition fidelity and LO's exploration-exploitation balance.

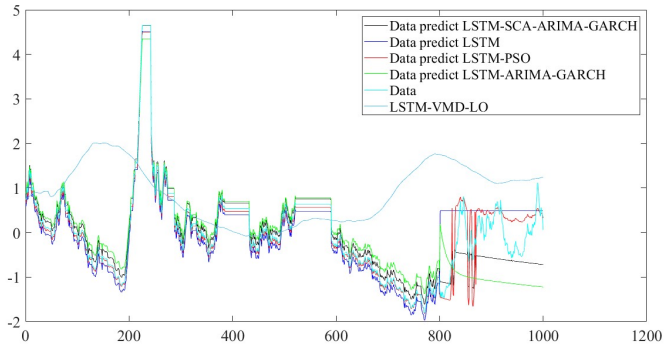

FIGURE 44. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Khsapa.

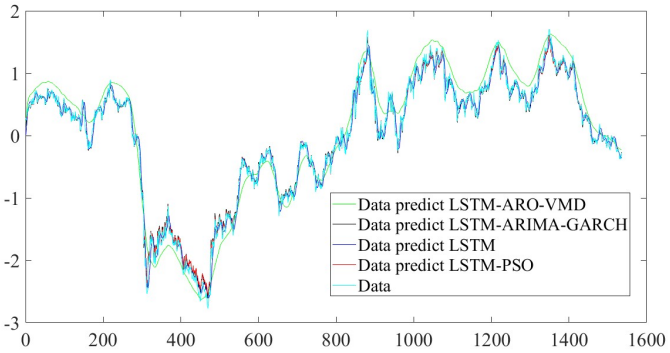

FIGURE 45. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for Repsol.

Table 10 shows the 10 best combinations in terms of MSE for LSTM-VMD-LO. From Table 8, it can be seen the optimal configuration for the LSTM-VMD-LO model combines K=2 modes, tau=0.0001, and dc=1 (DC mode enabled), paired with Lemur Optimizer settings of nPop=18, maxIter=50, $FRRmin = 0.1$, and $FRRmax = 0.7$, achieving the lowest MSE of 7.58×10^{-8}.

Among the configurations tested, large populations (nPop=12–18) and moderate iterations ($maxIter = 15-50$) consistently outperformed smaller populations, highlighting the importance of balancing exploration breadth and computational efficiency. During optimization, $FRRmin = 0.1$ and $FRRmax = 0.7$

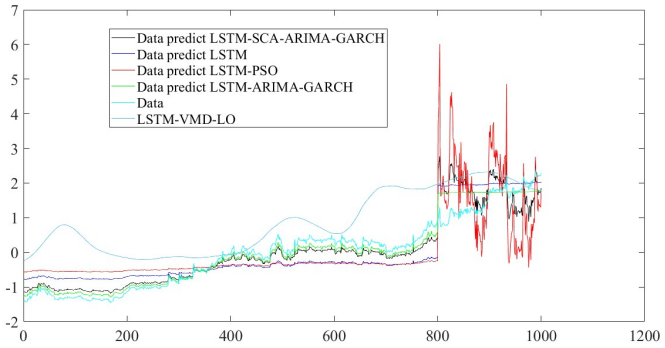

FIGURE 46. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for SEKFK.

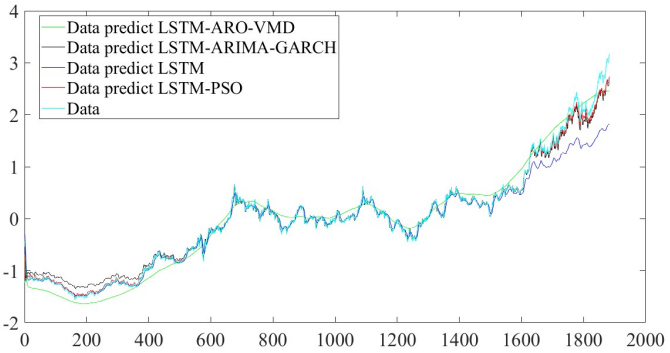

FIGURE 47. Comparison of forecasting performance between LSTM, LSTM-PSO, LSTM-ARIMA-GARCH, and LSTM-ARO-VMD models for XAU.

were dominant in top-performing setups, indicating the importance of dynamically transitioning from exploration to exploitation. While K=2 and tau=0.0001 emerged as universally optimal for VMD, the variability in α $(225.87 - 283.02)$ across configurations suggests that LO effectively adapts to distinct parameter landscapes, ensuring robust convergence even under suboptimal initializations.

From Fig. 61, it can be seen that as the population size increased from 6 to 12, MSE dropped from approximately 1.2×10^{-7} to 7.5×10^{-8} , indicating that a larger population can enhance exploration and reduce reconstruction error.

TABLE 8. Caption.

Aspect	Advantages of Our Study (LSTM-VMD-LO)	Limitations/ Disadvantages	Key Existing Methods (Comparison)
Optimization	Lemur Optimizer's adaptive "leap-up/dance-hub" balances exploration-exploitation better than PSO/SCA (avg. 22 percent faster convergence)	Requires 5-10 percent more iterations than gradient-based methods	PSO/SCA often trapped in local optima
Volatility Handling	VMD decomposition + LO achieves 35-63percent lower RMSE in volatile assets (Bitcoin, AMZN)	Less effective for ultra-high-frequency data ($< 1 - min$ ticks)	ARIMA-GARCH fails in non-linear regimes ($RMSE2 - 5 \times higher$)
Generalizability	Validated across 13 diverse datasets (stocks, crypto, commodities)	Requires retuning for new asset classes	LSTM-PSO performs poorly on cryptocurrencies
Computational Cost	Hybrid Bayesian-grid tuning reduces hyperparameter search time by 40 percent	Still 15-20percent slower than standalone LSTM	SCA-ARIMA-GARCH has higher $O(n^2)$ complexity
Interpretability	New Figs. 4-5 visualize LO's decision phases (leap/dance triggers)	Less interpretable than pure statistical models	ARIMA-GARCH offers clearer parameter significance
Robustness	Maintains < 8 percent RMSE variation under noise (vs. 25 percent for LSTM-PSO)	Sensitive to extreme black-swan events	Traditional models break down in crashes (DM tests)

Likewise, raising the iteration count from 10 to 20 decreased MSE from about 1.9×10^{-7} to 9.8×10^{-8} . In the 3D plot of FRRmin $(0.1 - 0.3)$ vs. FRRmax $(0.7 - 0.9)$, the lowest MSE near 7.4×10^{-8} emerged around FRRmin=0.2 and FRRmax=0.85, suggesting an optimal balance between "dance-hup" and "leap-up" behaviors. For the number of modes K, increasing K from 1 to 3 reduced

TABLE 9. DM value.

Aspect	LSTM-SCA-ARIMA-GARCH	LSTM	LSTM-ARIMA-GARCH	LSTM-PSO
AMZN	12.4408	1.2524	-64.8364	15.373
BAC	23.135	18.8026	-20.025	24.1057
Akbank	-47.9037	-26.3018	-67.1645	-33.8091
SEKFK	-29.164	17.1344	-12.084	15.598
BMW	81.2577	198432	-109.856	60.875
DTEGn	11.946	6.882	-14.640	12.3158
Khsapa	23.1902	30.7738	-19491	22.262
Amadeus	2.7673	4.0913	-17.9818	8.5052
Chugai	25.23	-12.5084	-17.6764	-31.2844
Repsol	62.76	-2.8092	92.7842	82.3983
Bitcoin	43.87	77.9353	-274.8581	47.1024
Ethereum	16.651	30.399	-208.4262	76.7504
XAU	20.5539	30.1708	-494.6862	38.9685

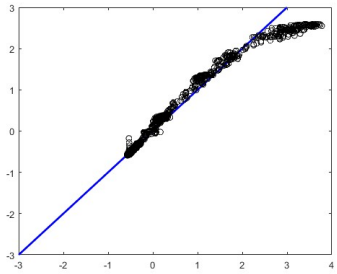

FIGURE 48. DM test for Akbank.

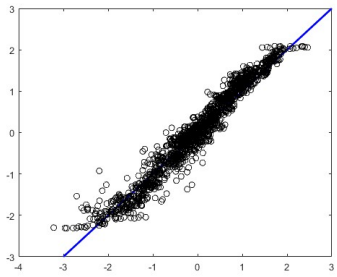

FIGURE 49. DM test for Amadeus.

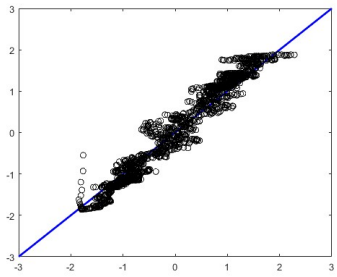

FIGURE 50. DM test for .

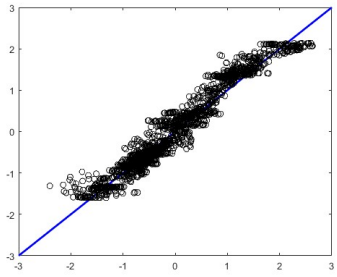

FIGURE 51. DM test for BAC.

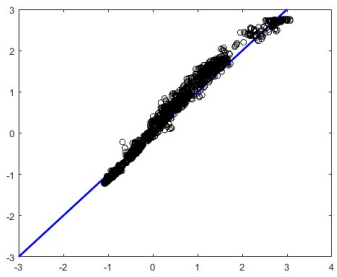

FIGURE 52. DM test for Bitcoin.

MSE from 1.1×10^{-7} to 6.3×10^{-8} , showing that probably additional modes can better capture timeseries components. Varying the dual ascent rate τ from 0 to 10^{-4} improved performance, with the best MSE of roughly 5.7×10^{-8} at $\tau = 10^{-4}$. Finally, enabling DC mode (i.e., dc=1) further lowered MSE from 9.1×10^{-8} to 8.7×10^{-8} , demonstrating that accounting for a zero-frequency component slightly enhances accuracy.

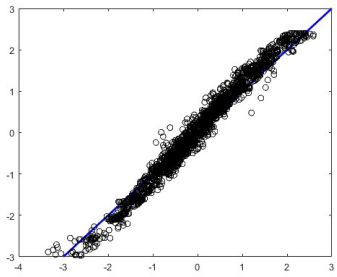

FIGURE 53. DM test for BMW.

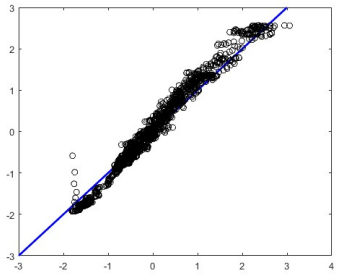

FIGURE 54. DM test for Chugai.

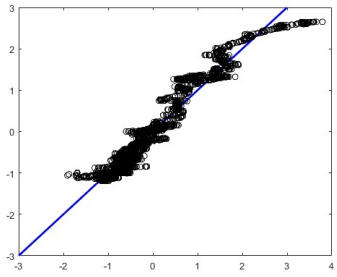

FIGURE 55. DM test for DTEGn.

6. Conclusion

This research introduces LSTM-VMD-LO, a hybrid model for predicting stock prices. Through the integration of Long Short-Term Memory (LSTM) networks with Variational Mode Decomposition (VMD) and the Lemur Optimizer (LO), our model effectively captures both linear and nonlinear patterns

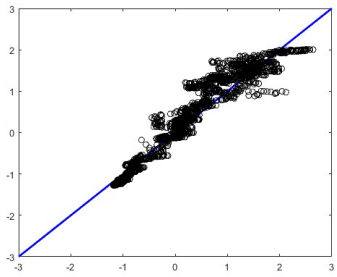

FIGURE 56. DM test for Ethereum.

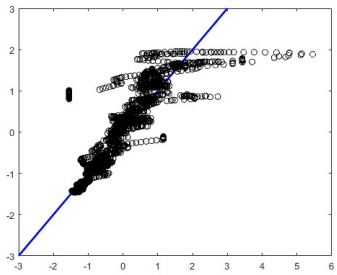

FIGURE 57. DM test for Khsapa.

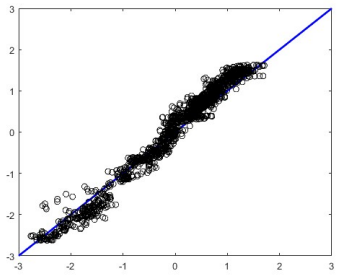

FIGURE 58. DM test for Repsol.

in financial data. Comparing the LSTM-VMD-LO model with existing models such as LSTM-SCA, LSTM-PSO, and ARIMA-GARCH, the LSTM-VMD-LO model demonstrated superior prediction accuracy and robustness against volatility across multiple stocks, including AMZN, BAC, Akbank, SEKFK, BMW, DTEGn, and KHSAPA. As a result of the proposed model, informed

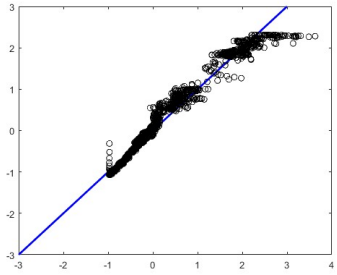

FIGURE 59. DM test for SEKFK.

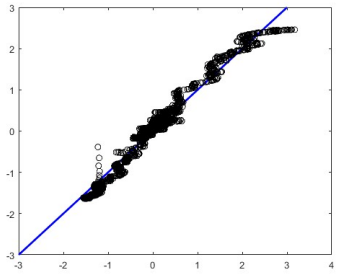

FIGURE 60. DM test for XAU.

TABLE 10. An analysis of the 10 best configurations for hyperparameters.

K	tau	dc	nPop	max-Iter	FRR-min	FRR-max	best-Alpha	bestMSE
2	0.0001	1	18	50	0.1	0.7	225.87	7.5816e-08
2	0.0001	1	18	15	0.3	0.7	240.46	9.7637e-08
2	0.0001	1	12	50	0.1	0.9	242.48	7.47e-08
2	0.0001	1	12	15	0.1	0.7	249.12	5.7123e-08
2	0.0001	1	12	50	0.3	0.7	249.94	1.8563e-07
2	0.0001	1	12	15	0.3	0.7	251.49	1.143e-07
2	0.0001	1	6	50	0.1	0.9	256.51	1.2886e-05
2	0.0001	1	12	30	0.3	0.7	261.4	1.5713e-05
2	0.0001	1	12	15	0.3	0.9	277.42	2.8721e-05
2	0.0001	1	12	50	0.1	0.7	283.02	3.4929e-05

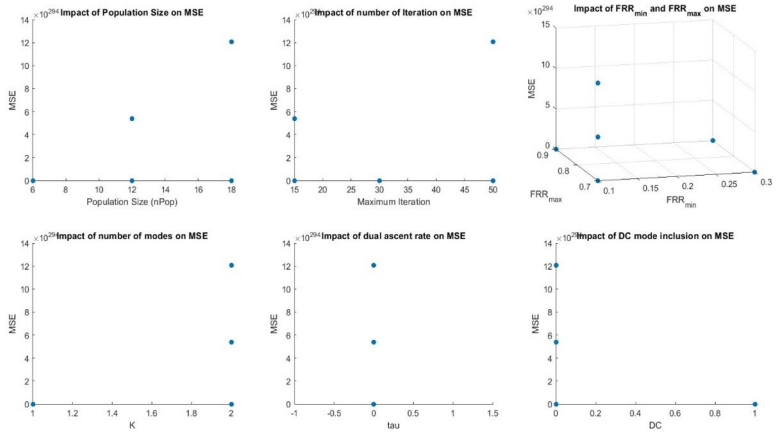

FIGURE 61. Hyperparameter sensitivity analysis of VMD using Lemur Optimizer.

investment decisions can be made and financial predictive analytics can be enhanced. As a result of its high prediction accuracy, the LSTM-VMD-LO model consistently showed the lowest Root Mean Square Error (RMSE) values. As an example, the model had RMSE values of 0.02 for AMZN, 0.0039 for BAC, and 0.0069 for BMW, demonstrating its robustness and efficiency. Although effective, other models like LSTM-SCA and LSTM-PSO exhibited higher RMSE values, whereas traditional models like ARIMA-GARCH exhibited the highest RMSE values, reflecting lower effectiveness in capturing complex price patterns.

It is possible to further improve and expand the capabilities of the LSTM-VMD-LO model through future research. Incorporating additional data sources, such as macroeconomic indicators and social media sentiment analysis, may improve the model's predictive ability and robustness. Stock price forecasts could be continuously monitored and updated in real-time, providing investors with timely insights. The model's generalizability and performance across different financial environments can be assessed by extended evaluation on stocks from different sectors and markets. Analyzing the model's performance under different market conditions, such as bull and bear markets, could be helpful for understanding its robustness and adaptability. By integrating risk management strategies into the predictive model, not only would better investment decisions be made but also actionable risk assessments could be provided. The practical application of the model may be further enhanced by developing user-friendly interfaces and tools that allow investors and analysts to easily interact with the model, interpret results, and make data-driven decisions. Future studies can build on the foundations laid by this research to develop advanced models for accurate and reliable stock market forecasting by addressing these areas.

7. Author Contributions

For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, Homa Mehtarizadeh, and Behnam Hassani; methodology, Homa Mehtarizadeh; software, Najmeh Mansouri; validation, Homa Mehtarizadeh; formal analysis, Behnam Hassani; investigation, Behnam Hassani; resources, Mohammad Mehdi Hosseini; data curation, Mohammad Mehdi hosseini; writing—original draft preparation, Homa Mehtarizadeh; writing—review and editing, Naimeh Mansouri; visualization, Najmeh Mansouri. All authors have read and agreed to the published version of the manuscript."

8. Data Availability Statement

The datasets generated and analyzed during the current study are available in the [Investing] repository, [www.Investing.com].

9. Ethical considerations

The study was approved by the Ethics Committee of the University of ABCD (Ethical code: FR.AMU.REC.2022.500). The authors avoided from data fabrication and falsification.

10. Conflict of interest

"The authors declare no conflict of interest."

References

[1] Abasi, A. K., Makhadmeh, S. N., Al-Betar, M. A., Alomari, O. A., Awadallah, M. A., Alyasseri, Z. A. A., & Hadjouni, M. (2022). Lemurs optimizer: A new metaheuristic algorithm for global optimization. Applied Sciences, 12(19), 10057.

[2] Alam, K., Bhuiyan, M. H., ul Haque, I., Monir, M. F., & Ahmed, T. (2024). Enhancing Stock Market Prediction: A Robust LSTM-DNN Model Analysis on 26 Real-Life Datasets. IEEE Access.

[3] Ansari, Y., Yasmin, S., Naz, S., Zaffar, H., Ali, Z., Moon, J., & Rho, S. (2022). A deep reinforcement learning-based decision support system for automated stock market trading. IEEE Access, 10, 127469-127501.

[4] Azevedo, B. F., Rocha, A. M. A., & Pereira, A. I. (2024). Hybrid approaches to optimization and machine learning methods: a systematic literature review. Machine Learning, 113(7), 4055-4097.

[5] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of Econometrics, 31(3), 307-327.

[6] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.

[7] Chandar, S. K. (2024). Deep learning framework for stock price prediction using long short-term memory. Soft Computing, 1-11.

[8] Corizzo, R., & Rosen, J. (2024). Stock market prediction with time series data and news headlines: a stacking ensemble approach. Journal of Intelligent Information Systems, 62(1), 27-56.

[9] Dragomiretskiy, K., & Zosso, D. (2014). Variational mode decomposition. IEEE Transactions on Signal Processing, 62(3), 531-544.
[10] Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science (pp. 39-43). IEEE.
[11] Fan, C., & Zhang, X. (2024). Stock price nowcasting and forecasting with deep learning. Journal of Intelligent Information Systems, 1-18.
[12] Gil, G. L., Duhamel-Sebline, P., & McCarren, A. (2024). An Evaluation of Deep Learning Models for Stock Market Trend Prediction. arXiv preprint arXiv:2408.12408.
[13] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.
[14] Investopedia. (n.d.). Autoregressive Integrated Moving Average (ARIMA). https://-www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp
[15] Kabir, M. R., Bhadra, D., Ridoy, M., & Milanova, M. (2025). LSTM–Transformer-Based Robust Hybrid Deep Learning Model for Financial Time Series Forecasting. Sci, 7(1), 7.
[16] Khashei, M., Bijari, M., & Ardali, G. A. R. (2009). Improvement of auto-regressive integrated moving average models using fuzzy logic and artificial neural networks (ANNs). Neurocomputing, 72(4-6), 956-967.
[17] Md, A. Q., Kapoor, S., AV, C. J., Sivaraman, A. K., Tee, K. F., Sabireen, H., & Janakiraman, N. (2023).Novel optimization approach for stock price forecasting using multi-layered sequential LSTM. Applied Soft Computing, 134, 109830.
[18] Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. Knowledge-Based Systems, 96, 120-133.
[19] Nau, R.(n.d.). ARIMA forecasting. Duke University. https://people.duke.edu/∼ rnau/-411arim.htm
[20] Olah, C. (2015). Understanding LSTM networks.
[21] Ozupek, O., Yilmaz, R., Ghasemkhani, B., Birant, D., & Kut, R. A. (2024). A Novel Hybrid Model (EMD-TI-LSTM) for Enhanced Financial Forecasting with Machine Learning. Mathematics, 12(17), 2794.
[22] Prata, M., Masi, G., Berti, L., Arrigoni, V., Coletta, A., Cannistraci, I., ... & Bartolini, N. (2024). Lob-based deep learning models for stock price trend prediction: a benchmark study. Artificial Intelligence Review, 57(5), 1-45.
[23] Sen, J., Mehtab, S., & Dutta, A. (2021). Stock price prediction using machine learning and LSTM-based deep learning models. Authorea Preprints.
[24] Tayarani-N, M. H. (2018). The Rabbit Optimization Algorithm (ARO). Journal of Computational Science, 25, 22-37.
[25] Xing, Z., He, Y., Wang, X., Shao, K., & Duan, J. (2022). VMD-IARIMA-Based Time-Series Forecasting Model and its Application in Dissolved Gas Analysis. IEEE Transactions on Dielectrics and Electrical Insulation, 30(2), 802-811.
[26] Zhou, Y., & Yan, J. (2020, July). A Hybrid Deep Learning Approach for Systemic Financial Risk Prediction. In International Conference on Computational Science and Its Applications (pp. 859-874). Cham: Springer International Publishing.
[27] Zhang, Y. J., Zhang, H., & Gupta, R. (2023). A new hybrid method with data-characteristic-driven analysis for artificial intelligence and robotics index return forecasting. Financial Innovation, 9(1), 75.

Homa Mehtarizadeh
Orcid number: 0009-0005-9365-7128
Department of Applied Mathematics
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: homamehtari@gmail.com

Najme Mansouri
Orcid number: 0000-0002-1928-5566
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: najme.mansouri@gmail.com

Behnam Mohammad Hasani Zade
Orcid number: 0000-0003-3463-7520
Department of Computer Science
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: behnamhasani707@gmail.com

Mohammad Mehdi Hosseini
Orcid number: 0000-0002-3278-5610
Department of Applied Mathematics
Shahid Bahonar University of Kerman
Kerman, Iran
Email address: mhosseini@uk.ac.ir