electronics

MDPI

Article

A Novel Hybrid Model Using Demand Concentration Curves, Chaotic AFDB-SFS Algorithm and Bi-LSTM Networks for Heating Oil Price Prediction

Seçkin Karasu

Department of Electrical Electronics Engineering, Zonguldak Bülent Ecevit University, Zonguldak 67100, Türkiye; seckin.karasu@beun.edu.tr

Abstract

Nowadays, renewable energy sources are gaining importance, yet global energy demand is primarily met by burning fossil fuels. Fluctuations in fossil fuel availability, driven by geopolitical tensions, supply–demand changes, and natural disasters, can lead to sudden energy price spikes or supply shortages, adversely affecting the global economy. Despite its negative impact on carbon emissions and climate change, Heating Oil (HO) offers advantages over other fossil fuels in efficiency, reliability, and availability. Accurate time series prediction models for HO are crucial for stakeholders. This study proposes a novel hybrid model, integrating the Chaotic Adaptive Fitness-Distance Balance-based Stochastic Fractal Search (AFDB-SFS) algorithm with a Bidirectional Long-Short Term Memory (Bi-LSTM) network, for HO close price prediction. The dataset comprises daily observations of five financial time series (close, open, high, low, and volume) over 4260 trading days, yielding a total of 21,300 data points (4260 days $\times$ 5 variables). During the feature extraction stage, financial signal processing methods such as Demand Concentration Curve (DCC) and traditional technical indicators are utilized. A total of 189 features are extracted at appropriate intervals for each indicator. Due to the large number of features, the AFDB-SFS algorithm then efficiently identifies the most compatible feature subsets, optimizing the Bi-LSTM model based on three criteria: maximizing R^2, minimizing RMSE, and minimizing feature count. Experimental results demonstrate the proposed hybrid model's superior performance, achieving high accuracy (R^2 of 0.9959 and RMSE of 0.0364), outperforming contemporary models in the literature. Furthermore, the model is successfully implemented on the Jetson Orin Nano Developer Platform, enabling real-time, high-frequency HO price predictions with ultra-low latency (1.01 ms for Bi-LSTM), showcasing its practical utility for edge computing applications in commodity markets.

Keywords: heating oil prediction; financial signal processing; demand concentration curve; AFDB-SFS; Bi-LSTM; embedded systems

Academic Editor: Simeone Marino

Received: 30 October 2025
Revised: 1 December 2025
Accepted: 5 December 2025
Published: 7 December 2025

Citation: Karasu, S. A Novel Hybrid Model Using Demand Concentration Curves, Chaotic AFDB-SFS Algorithm and Bi-LSTM Networks for Heating Oil Price Prediction. *Electronics* **2025**, *14*, 4814. https://doi.org/10.3390/electronics14244814

1. Introduction

Energy resources are the backbone of modern society, essential for driving daily activities and industrial processes globally [1–4]. Without them, crucial systems like homes, transportation, and manufacturing would collapse. Energy sources are broadly classified into fossil fuels [5], renewable energy sources [6], nuclear energy [7], and alternative energy sources [8]. Despite environmental concerns, fossil fuels (Coal, Natural Gas, Crude Oil (CO), and derivatives like Heating Oil) remain the dominant global energy source [9]. Energy

policies must navigate these factors to build a balanced energy mix. Projections indicate that fossil fuels, including Heating Oil (HO), will supply roughly 80% of global energy consumption by 2035 without major policy shifts, ensuring their continued relevance in the near term [10,11]. Although global energy needs are rapidly shifting towards clean energy to address carbon emissions, the transition will take decades, meaning that managing the stability and predictability of the existing dominant fossil fuel market is essential. Uncontrolled price volatility in critical fuels like HO can destabilize overall energy security, hinder policy execution, and undermine confidence in the transition process itself. Among these, HO, a distillate fuel refined from CO, plays a pivotal role in energy markets, extensively used for residential and commercial heating, particularly in colder regions. HO's strategic importance stems from its widespread use and price dynamics, which make it a key indicator for energy market analysis [12]. HO prices exhibit a stronger correlation between futures and spot markets than crude oil or gasoline—a relationship that becomes even more pronounced during major market disruptions (e.g., the 2008 financial crisis or the 2020 pandemic) [12].

HO price volatility underscores the market's sensitivity to demand spikes, especially during winter months. This sensitivity highlights the critical need for time series prediction powered by machine learning and statistical models. Accurate HO price predictions are vital for strategic planning, enabling energy companies to optimize supply chains, manage risks, and stabilize costs. These tools are essential for maintaining market resilience against sudden price surges, as demonstrated during past crises [12]. While energy policies necessitate a gradual shift toward cleaner alternatives, HO's high energy density and compatibility with existing infrastructure ensure its reliability. Continued advancements in time series prediction are crucial for helping stakeholders navigate volatility, ensuring market stability and cost efficiency in a balanced energy ecosystem.

1.1. Current Approaches in the Literature

Current HO price prediction studies leverage a wide array of Artificial Intelligence (AI) and optimization techniques, frequently assessing model performance using statistical error criteria such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). Early work using Artificial Neural Networks (ANN), incorporating standard deviation and historical prices from 1997 to 2002, achieves an MSE of 0.002 and an R^2 of 0.90 [13]. Time series analysis sees the Autoregressive Fractionally Integrated Moving Average (ARFIMA) model combined with the Fractionally Integrated Generalized Autoregressive Conditionally Heteroskedastic (FIGARCH) model demonstrate superior performance over other ARFIMA-based configurations (including ARFIMA-IGARCH—Integrated Generalized Autoregressive Conditionally Heteroskedastic, and ARFIMA—GARCH—Generalized Autoregressive Conditionally Heteroskedastic) on data from 1995 to 2012, recording the lowest errors with MAE (1.156) and MSE (1.337) [14]. Hybrid approaches are common, such as the Least Squares Support Vector Machines (LSSVM) optimized with an improved Artificial Bee Colony (eABC) algorithm, which is noted for its high prediction accuracy by mitigating common training issues with Levy mutation [15]. In comparisons of various Neural Networks (NN), the Levenberg–Marquardt (LM) method yields the best HO prediction performance across several fuel datasets in terms of lowest MSE and highest R^2 [16]. Another hybrid approach integrates a complex network model with the Fourier model (2001–2016 data) to predict price fluctuations, showing superior results compared to Non-Linear Autoregressive (NAR) NN [17]. For advanced hybrid methods, the Wavelet Packet Decomposition (WPD) combined with Stochastic Time Effective Weight (SW) and Long Short-Term Memory (LSTM) (WPD-SW-LSTM) achieves the overall lowest errors for HO price prediction (MAE: 0.0212,

MSE: 0.0642, and MAPE—Mean Absolute Percentage Error: 0.4775), outperforming both single models (like Support Vector Machine (SVM) and LSTM) and other hybrid decomposition models (WPD—BPNN—Backpropagation Neural Network, CEEMDAN-LSTM—Complete Ensemble Empirical Mode Decomposition with Adaptive Noise, VMD-LSTM—Variational Mode Decomposition, and ST-GRU—Spatiotemporal Gated Recurrent Unit) [18]. Furthermore, Nonlinear Autoregressive Neural Network (NARX) models are assessed for monthly HO prices (1986–2021) and found to have an average success rate of 20%, suitable for technical analysis [19]. Optimization efforts using Bayesian Optimization (BO) identify the Support Vector Regression (SVR)-BO model as the most accurate, with the lowest Root Mean Squared Error (RMSE) of 0.0274, followed closely by GRP-BO (Gaussian Regression Process) and RT-BO (Regression Tree); in contrast, kNN-BO (k-Nearest Neighbors) and DFNN-BO (Deep Feedforward Neural Networks) record the highest RMSE [20]. Finally, a comparative study (2004–2023) shows XGBoost with lower MAE and MSE (0.792 and 1.207), yet Random Forest (RF) exhibits lower variability with a much smaller RMSE (0.007) [21].

There are several technical indicators derived from financial signal processing techniques such as Simple Moving Average (SMA) [22–25], Exponential Moving Average (EMA) [22–24], Weighted Moving Average (WMA) [25], Kaufman's Adaptive Moving Average (KAMA) [23], Commodity Channel Index (CCI) [23,24,26], Rate of Change (ROC) [23,24], Relative Strength Index (RSI), Average True Range (ATR) and Volatility Ratio (VR) [23], Average Directional Index (ADX), Stop and Reverse (SAR), Moving Average Converge Divergence (MACD) [24], and A/D Oscillator [25]. In this study, it is focused on the signal processing methods used to obtain Demand Concentration Curve (DCC) with the fusion approach since there is a wide variety of indicators in the literature.

Metaheuristic Search (MHS) approaches are important as a feature selection approach in the literature. Especially, there are many different Meta-Heuristic (MH) algorithms in the literature used to find more relevant features or improve model parameters in the prediction of different financial instruments [27–31]. Some of them can be summarized as follows: Harmony Search (HS) [27], Genetic Algorithm (GA) [27,32], Social Spider Optimization (SSO) [28], Bat Algorithm (BA) [28], Particle Swarm Optimization (PSO) [22,29], Flower Pollination Algorithm (FPA) [29], Henry Gas Solubility Optimization (HGSO) [23], Ant Colony Optimization (ACO) [24], Cuckoo Search Algorithm [30], Sliding-Window Optimization (SWO) [31] and Grey Wolf Optimization (GWO) [32–34]. As a result of the literature review, among the MH algorithms, although the Stochastic Fractal Search (SFS) method is used in the feature selection stage [34], it is seen that the use of SFS derivatives in the new application areas is a very hot topic. In addition, the fact that the performances of these methods have not yet been fully investigated at the feature selection stage can be said as one of the sources of motivation.

The development of metaheuristic search strategies has followed a progressive trajectory beginning with the SFS algorithm as the foundational exploitation–exploration framework [35]. This is enhanced by the introduction of NSM-SFS (Natural Survival Method integrated with SFS), which incorporates ecosystem-based survival scoring to improve population adaptation [36]. Subsequently, the FDB-SOS (Fitness–Distance Balance for SOS) mechanism demonstrates strong performance across benchmark functions [37], leading to the creation of FDB-SFS, which improves candidate selection and global search stability [38]. Further enhancements include the dFDB-SFS (dynamic FDB-SFS) variant that adaptively tunes the exploration–exploitation balance [38], followed by the hybrid NSM + FDB + OBL-SFS model integrating opposite-based learning for stronger global convergence [39]. Later, the theoretical refinement of dynamic adaptation strengthened the structure of dFDB-SFS [40], culminating in the advanced AFDB-SFS algorithm, which employed chaotic maps and adaptive switching strategies for superior performance [41,42].

1.2. Motivation and Contributions

In this study, the main motivation and contributions of the proposed model to the literature are summarized as follows:

A hybrid approach is introduced that employs various feature extraction techniques based on financial signal processing to predict the close price of HO. These techniques not only demonstrate the effectiveness of technical analysis indicators but also propose a fusion approach that combines traditional indicators with the DCC. This fusion of methods can potentially lead to more accurate and reliable predictions, providing valuable insights into financial markets.

The study delves into the nature of financial time series data and reveals that it exhibits a fractal structure. Understanding this fractal nature is crucial as it implies that price movements are not entirely random but possess complex patterns that, if deciphered, can offer a competitive edge in financial analysis and prediction. The recognition of this fractal structure opens up opportunities for further research in the field of fractal analysis and its application in financial prediction.

Thanks to the AFDB-SFS algorithm, fractal and stochastic dimensions are managed through indicators, and it effectively addresses the complex task of identifying useful features within a large feature set without getting trapped in local minima, supported by a chaotic scoring function.

Bi-LSTM inherently allows it to access both backward and forward context at every point in the input time series in the sequence, thus significantly improving its prediction performance.

This study not only presents a novel technique for enhancing financial prediction but also contributes to a more profound comprehension of the intricate nature of financial time series. It inspires future research, pushing the boundaries of what can be achieved in the realm of financial data analysis and prediction and potentially leading to more accurate and informed decision-making in financial markets.

1.3. Organization

The remaining part of the paper is organized as follows: Section 2 details the methodology and implementation of the proposed hybrid prediction model. Within this scope, the HO time series dataset used in the study, financial signal processing techniques, including the DCC (such as Stochastic Oscillator (SO), Williams %R Oscillator (WRO), Ultimate Oscillator (UO), RSI indicator, Momentum Oscillator (MO), CCI indicator, and Money Flow Index (MFI) Oscillator), the AFDB-SFS optimization algorithm, and the mathematical backgrounds of learning models like Bi-LSTM are presented. Furthermore, the performance metrics utilized for evaluating the models are also described in this section. Section 3 focuses on the empirical results and discussion. This section presents the time period analysis of HO Close prices, the model parameters used, the obtained performance results, and the embedded system test results. Finally, Section 4 concludes the study with an evaluation and highlights potential areas for future research.

2. Materials and Methods

2.1. Proposed Hybrid Prediction Model

The block diagram of the proposed hybrid HO time series prediction model framework is given as in Figure 1. The steps applied in the study consist of four main stages: preprocessing stage, feature extraction stage/financial signal processing stage, feature selection and performance evaluation stages.

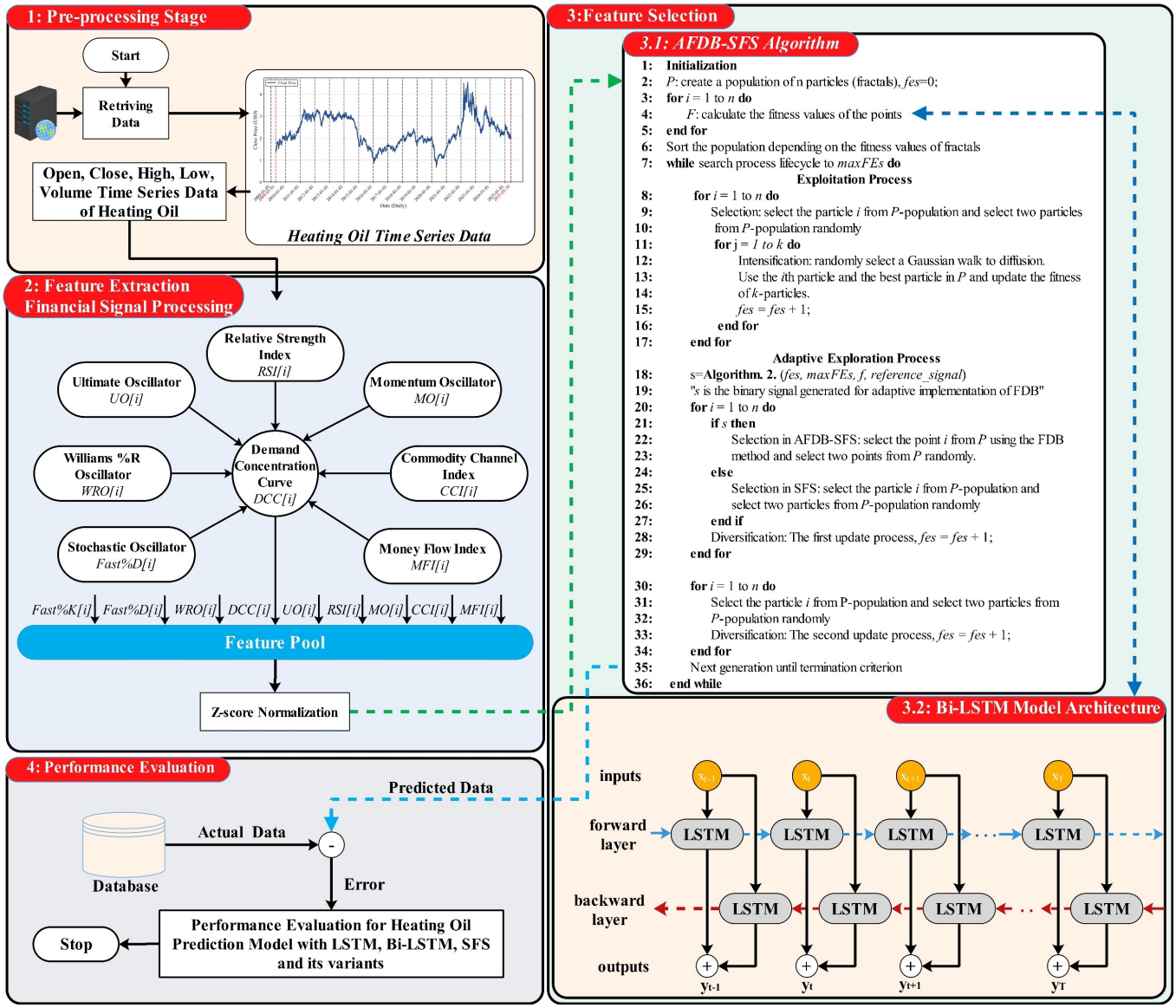


Figure 1. Block diagram of proposed framework for HO time series price prediction.

Step 1: Pre-processing stage. The first stage is the pre-processing stage where the data is obtained, statistical analysis is performed.

Step 2: Feature Extraction/Financial signal processing stage. The second stage is the feature extraction stage where a feature pool is created using features extracted from signal processing methods and features such as the DCC indicator extracted with the fusion approach. In the feature extraction step, 189 features are extracted using features based on technical indicators such as CCI, RSI, SO, WRO, MO, MFI, UO and DCC. After this stage, normalization steps are applied.

Step 3: Feature selection stage. After these stages, the feature selection stage, which is an important stage is applied. Hybrid models are developed by combining LSTM and Bi-LSTM models with optimization methods such as SFS, FDB-SFS, dFDB-SFS, and AFDB-SFS. HO is predicted with the AFDB-SFS-Bi-LSTM approach, which imitates the formation of fractals in nature based on the Bi-LSTM approach, by creating different sub-feature groups. GA- and PSO-based optimization approaches are employed for benchmarking in the study.

Step 4: Performance evaluation stage. In the final stage, the obtained model is compared with other models based on statistical error criteria. It is benchmarked against SFS and its variants. Testing of LSTM and Bi-LSTM models is also conducted.

2.2. HO Time Series Dataset

The HO time series dataset used in the study consists of a total of 4260 daily data from 1 May 2009 to 30 May 2025 [43]. It consists of Open, High, Low, Close and Volume time series. There are 21,300 data in total from 5 × 4260. According to the data in Table 1, there are 4260 observations for each time series. The average opening price for the Open time series is 2.29 (std = 0.68), while it shows a distribution slightly skewed to the right (skewness = 0.24), it exhibits a low kurtosis compared to the normal distribution, with a kurtosis value of −0.58. The High time series shows an average highest price of 2.32 (std = 0.69), a distribution skewed to the right (skewness = 0.29) and low kurtosis with a kurtosis value of −0.51. In the Low column, the average low price is calculated as 2.26 (std = 0.67), a distribution skewed to the right (skewness = 0.19) and low kurtosis with a kurtosis value of −0.67 are observed. Finally, the average close price for the Close column is determined as 2.29 (std = 0.68), right skewed distribution (skewness = 0.24) and kurtosis value is determined as −0.59. The average transaction volume for the Volume column is 39,915.19 (std = 33.678.13). This column exhibits a highly skewed distribution (skewness = 4.36) and a high kurtosis value (kurtosis = 49.87).

Table 1. Statistical values calculated for the HO time series.

Time Series	Count	Mean	Std	Min	Max	Skewness	Kurtosis
Open	4260	2.29	0.68	0.67	4.59	0.24	−0.58
High	4260	2.32	0.69	0.73	4.67	0.29	−0.51
Low	4260	2.26	0.67	0.63	4.41	0.19	−0.67
Close	4260	2.29	0.68	0.67	4.56	0.24	−0.59
Volume	4260	39,915.19	33,678.13	1.00	658,816.00	4.36	49.87

The distribution of HO Open, High, Low, and Close price profiles from 2009 to 2025 using violin-boxen plots is given as in Figure 2.

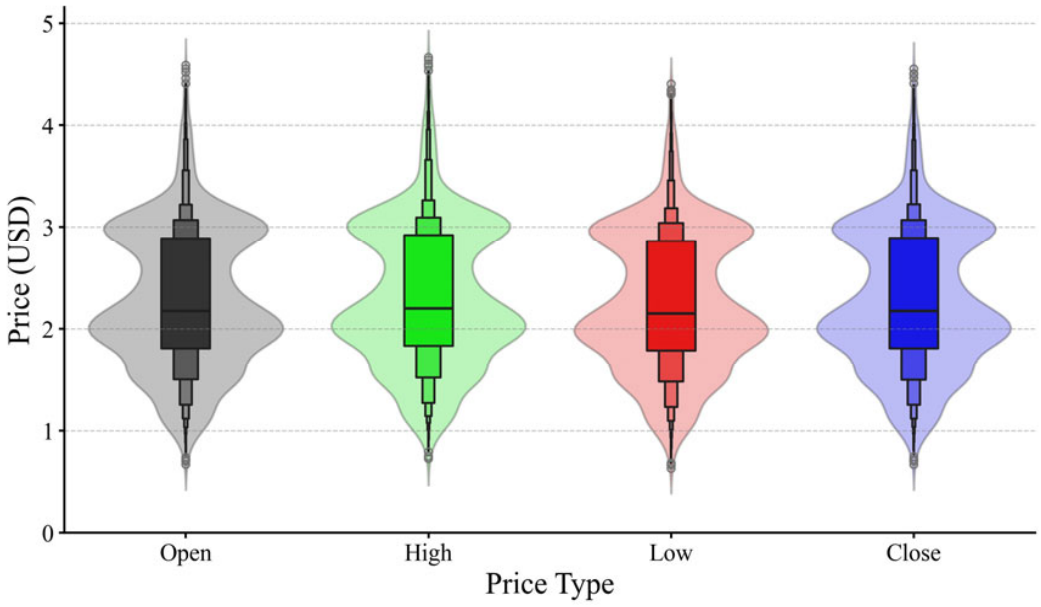


Figure 2. Violin-boxen distribution of HO prices.

While other time series are also utilized in this study, the primary focus is on predicting the close price of HO. In addition to statistical values, the temporal variation in HO close price is presented in Figure 3. Black dashed lines indicate the beginning of each year, while red dashed lines mark the start and end timestamps of the HO close price time series used for the prediction study.

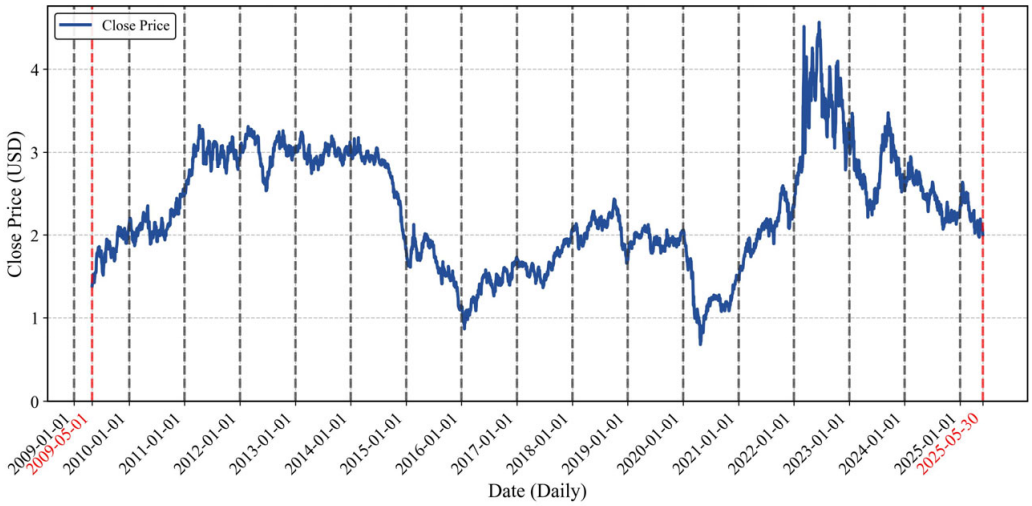


Figure 3. Daily change in the HO close price.

In Figure 4, the violin plot and strip plot are drawn together. The HO close price data distribution has been plotted over the years using a violin plot in Figure 4. The distribution in the data set is shown in more detail by visualizing the data points for each year with a strip plot on the same axis.

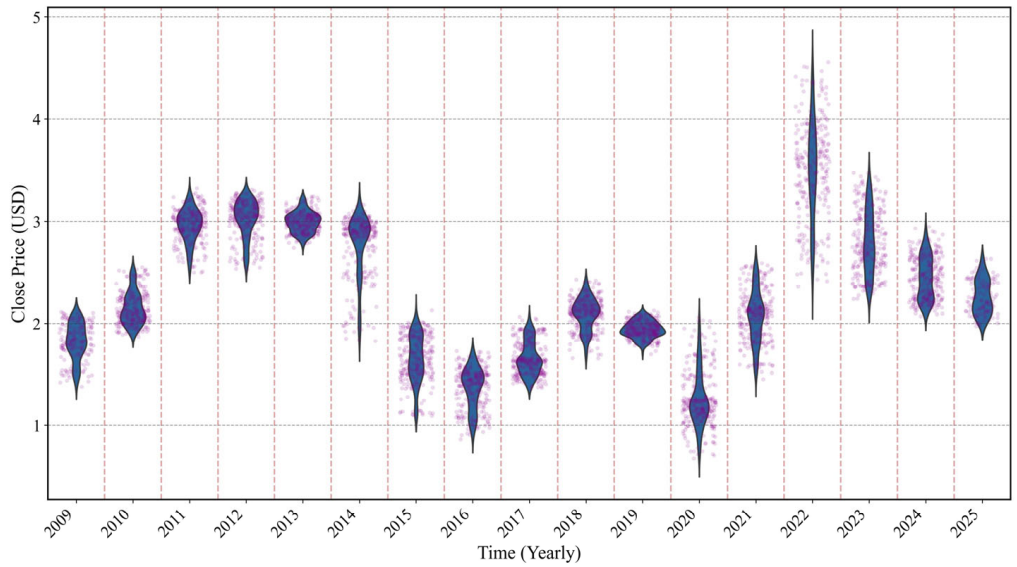


Figure 4. Violin plot and strip plot representation of the change in HO close price data over the years.

2.3. Financial Signal Processing

Financial investment strategies are primarily divided into fundamental analysis [44] and technical analysis [45]. Fundamental analysis evaluates an asset's intrinsic value by examining factors such as financial performance, management, and industry conditions, incorporating both quantitative metrics and subjective judgment. In contrast, technical analysis predicts future trends by analyzing historical market data, such as prices and trading volumes. Signal processing techniques are particularly well-suited for technical

analysis, enabling the extraction of actionable insights from time series data. Financial signal processing applies these techniques to stock prices, trading volumes, and other time series data to support investment decisions, fraud detection, and market efficiency analysis [46]. Methods include time series analysis, pattern recognition, and machine learning [47,48]. Various indicators, such as trend, momentum, volatility, and volume-based indicators, are employed depending on the market, asset, and analytical objectives.

Table 2 lists the main momentum-based indicators, their developers, development years, and standard abbreviations. The remaining sections explain indicator values derived from price time series, elaborating their mathematical formulations and signal processing methodologies across diverse financial assets.

Table 2. Timeline of key financial signal processing indicators and their developers.

Studies	Year	Developer	Indicator Name	Indicator Abbreviation
[49]	1950–1960	George C. Lane	Stochastic Oscillator	SO
[50]	1973	Larry Williams	Williams %R Oscillator	WRO
[51]	1976	Larry Williams	Ultimate Oscillator	UO
[52,53]	1978	J. Welles Wilder	Relative Strength Index	RSI
[53]	1978	J. Welles Wilder	Momentum Oscillator	MO
[54]	1980	Donald R. Lambert	Commodity Chanel Index	CCI
[55]	1989	Avrum Soudack & Gene Quong	Money Flow Index	MFI
[56]	2004	Yaşar Erdinç	Demand Concentration Curves	DCC

2.3.1. Demand Concentration Curve (DCC)

The *DCC* indicator was developed by Yaşar Erdinç by blending 7 different financial technical analysis indicators [56], and it can be depicted as shown in Figure 5. As a formulation, it can be given as in Equation (1).

$$DCC_{N_{1,\ldots,7}}[i] = \frac{Fast\%D_{N_1}[i] + WRO_{N2}[i] + UO_{N_3}[i] + RSI_{N_4}[i] + MO_{N_5}[i] + CCI_{N_6}[i] + MFI_{N_7}[i]}{7} \quad (1)$$

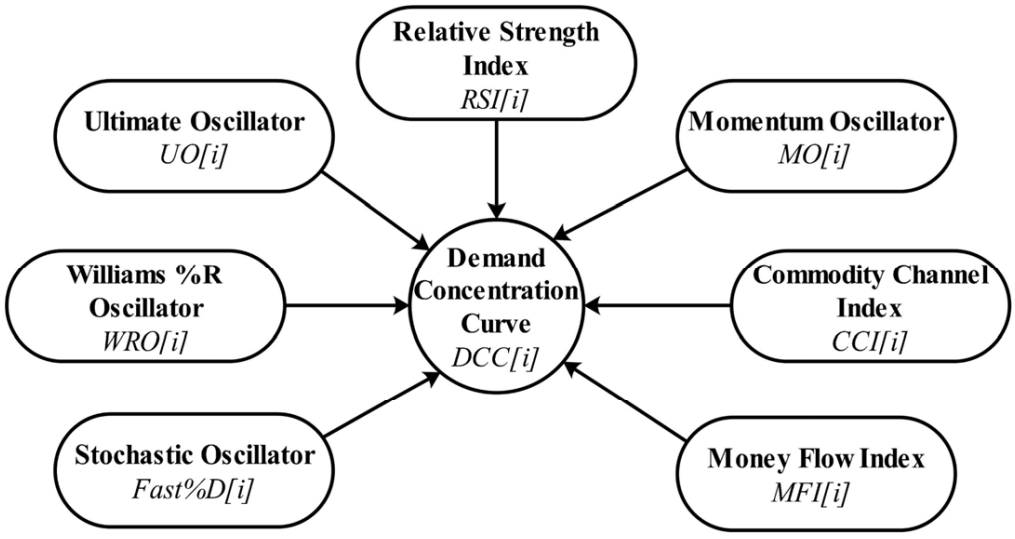


Figure 5. Representation of DCC's ingredients.

After years of accumulated experience following extensive research, this indicator has been created by averaging different metrics [56]. The purpose of this indicator is to enable the selection of assets with the least buying risk and the potential to generate profits in the medium and long term once purchased. Since all indicators fall within the range of 0–100 and −100 to 100, it is relatively straightforward to harmonize them. It has been tested on the Turkish stock market over an 8-year period, allowing for the use of different time frames for various periods [56]. For instance, when considering long-term investments, the periods of indicators can be chosen as 21 or 45 instead of 14. In this case, since daily predictions will be made, the period values have been set at 14 [56].

2.3.2. Stochastic Oscillator (SO)

The Stochastic oscillator, developed by George C. Lane in the 1950s, identifies overbought and oversold conditions by analyzing price momentum over a lookback period N, using high prices $H[i]$, low prices $L[i]$ and close price $C[i]$ [49]. The concept relies on the tendency of prices to close near the upper or lower bounds of the trading range during pronounced trends, enabling the detection of possible trend reversals. The stochastic formula has two values to calculate, $Fast\%K_N[i]$ and $Fast\%D_N[i]$. The calculation is started as shown in the following steps:

Step 1: The lowest value $L_N[i]$ in the low time series for the last N period is calculated as given in Equation (2).

$$L_N[i] = \min(L[i-1], L[i-2], \ldots, L[i-N]), i > N \tag{2}$$

Step 2: The highest value $H_N[i]$ in the high time series for the last N period is calculated as given in Equation (3).

$$H_N[i] = \max(H[i-1], H[i-2], \ldots, H[i-N]), i > N \tag{3}$$

Step 3: Using the time series $L_N[i]$, and $H_N[i]$, stochastic value $\%K_N[i]$ is calculated as given in Equation (4).

$$\%K_N[i] = 100 \times \frac{C[i] - L_N[i]}{H_N[i] - L_N[i]} \tag{4}$$

Step 4: Using $\%K_N[i]$, fast stochastic value of it is obtained through exponential smoothing as in Equation (5).

$$Fast\%K_N[i] = \frac{2 \times Fast\%K_N[i-1] + \%K_N[i]}{3}, i > N \tag{5}$$

Step 5: $Fast\%D_N[i]$, the smoothed stochastic value is calculated as the 3-period average of $Fast\%K_N[i]$, as given in Equation (6).

$$Fast\%D_N[i] = \frac{1}{3}\sum_{k=0}^{2} Fast\%K_N[i-k], i > N+2 \tag{6}$$

The Stochastic Oscillator ($\%K$ line) is graphically represented on a chart with a standardized scale ranging from 0 to 100, reflecting its normalized momentum metric. The $\%D$ line, a smoothed derivative of the $\%K$ line, is similarly plotted on the same 0-to-100 scale, enhancing trend signal stability. Overbought conditions are conventionally identified when the $\%K$ line exceeds 80, whereas oversold conditions are indicated when it falls below 20, providing traders with critical thresholds for discerning potential market entry and exit points. This indicator is applicable across a diverse range of financial instruments, including equities, commodities, currencies, and other asset classes.

2.3.3. Williams %R Oscillator (WRO)

The Williams %R oscillator, introduced by Larry Williams in 1973, is a momentum-based technical indicator used to identify overbought and oversold conditions in financial markets, including stocks, commodities, and currencies [50]. It assesses the current closing price relative to the price range over a user-specified period, denoted as N. The calculation is performed by subtracting the current close from the highest price over N, dividing this by the difference between the highest and lowest prices over the same period, and multiplying by -100 to express the result as a percentage, as shown in Equation (7).

$$WRO_N[i] = \frac{H_N[i] - C[i]}{H_N[i] - L_N[i]} \times (-100) \tag{7}$$

The WRO yields values between 0 and -100. Readings above -20 indicate an overbought condition, while readings below -80 suggest an oversold condition. Traders use these levels to identify potential market entry or exit points. To improve accuracy, the WRO_N is often combined with other technical indicators or chart patterns to confirm signals and enhance trading strategies [50].

2.3.4. Ultimate Oscillator (UO)

The UO indicator is a technical analysis indicator developed by Larry Williams in 1976 [51]. UO indicator is used to capture momentum across three different timeframes. Because of that, it reduces the generation of false trading signals that can be a handicap in other momentum indicators. The calculation of this oscillator is done with the steps as follows:

Step 1: Buying Pressure (BP) is calculated as given in Equation (8).

$$BP[i] = C[i] - \min(L[i], C[i-1]) \tag{8}$$

Step 2: True Range (TR) is calculated as given in Equation (9).

$$TR[i] = \max(H[i], C[i-1]) - \min(L[i], C[i-1]) \tag{9}$$

Step 3: The ratio of True Range average values of Buying Pressure for N_1 period is as given in Equation (10).

$$A_{N_1}[i] = \frac{\frac{1}{N_1}\sum_{n=1}^{N_1} BP[i-n]}{\frac{1}{N_1}\sum_{n=1}^{N_1} TR[i-n]}, i > N_1 \tag{10}$$

Step 4: Two more such average calculations are made for UO. The mean values for the N_1, N_2, N_3 periods are calculated as given in Equation (10). Calculation of UO is done as given in Equation (11).

$$UO_{N_1,N_2,N_3}[i] = 100 \times \frac{4A_{N_1} + 2A_{N_2} + A_{N_3}}{(4+2+1)}, i > N_1, N_2, N_3 \tag{11}$$

Here, period values are selected as $N_1 < N_2 < N_3$. In general, the original values for these period values are used as $N_1 = 7$, $N_2 = 14$, $N_3 = 28$. In the study, different values such as $N_1 < N_2 < N_3$ are produced and applied for the UO calculation.

2.3.5. Relative Strength Index (RSI)

The Relative Strength Index (RSI) is a momentum oscillator ranging from 0 to 100 that measures the speed and magnitude of recent price changes. The concept of relative strength was first employed by Robert A. Levy in 1967 as a criterion for stock selection [52],

but the modern RSI, including its specific formula and the classic overbought/oversold interpretation, was introduced by J. Welles Wilder in 1978 [53]. Readings above 70 typically signal overbought conditions and a possible pullback, while readings below 30 indicate oversold conditions and a potential rebound; some traders prefer stricter levels such as 80/20 or 90/10. The standard calculation uses a 14-period lookback and can be applied to any timeframe. Thanks to its simplicity and effectiveness, RSI remains one of the most popular technical indicators across stocks, forex, commodities, and cryptocurrencies. The following calculation steps are applied to calculate the RSI, which can be considered as one of the overbought/sold indicators:

Step 1: Firstly, it is needed to calculate the gain for each period. Gain ($up[i]$): The difference if the current closing price $C[i]$ is higher than previous day's $C[i-1]$.It is '0' otherwise. Loss ($down[i]$): The positive difference if the previous closing price $C[i-1]$ is higher than the current one $C[i]$. It is '0' otherwise.

Step 2: The simple average of the gains and losses over the specified N periods is calculated as in Equations (12) and (13), respectively.

$$AG_N[i] = \frac{1}{N_p}\sum_{n=1}^{N} up[i-n], i > N \tag{12}$$

$$AL_N[i] = \frac{1}{N_n}\sum_{n=1}^{N} down[i-n], i > N \tag{13}$$

The number of elements with a nonzero value of $up[i]$ is indicated by N_p, and the number of elements with a nonzero value of $down[i]$ is indicated by N_n.

Step 3: Relative Strength ($RS_N[i]$) ratio is calculated as given in Equation (14).

$$RS_N[i] = \frac{AG_N[i]}{AL_N[i]} \tag{14}$$

Step 4: $RSI_N[i]$ is then calculated as given in Equation (15) using the $RS_N[i]$ expression.

$$RSI_N[i] = 100 - \frac{100}{1 + RS_N[i]} \tag{15}$$

2.3.6. Momentum Oscillator (MO)

It is defined as a simple and useful indicator whose use dates back to ancient times, although the concept of momentum indicator is first mentioned by Welles Wilder [53]. The most important feature of this indicator is that it can be used in both sideways and especially trending markets. It is calculated as given in Equation (16) as the current price divided by the price of a previous period, and the quotient is multiplied by 100 with a very simple formula.

$$MO_N[i] = \frac{C[i]}{C[i-N]} \times 100 \tag{16}$$

The result is an indicator that oscillates around 100. Values below 100 indicate negative momentum, suggesting a decreasing price trend, while values above 100 indicate positive momentum, signifying an increasing price trend. With this indicator, a line that fluctuates around 100. Thus, the tempo of the market is measured by the momentum line. Traders often use the momentum indicator to identify potential trend reversals or confirm an existing trend. If the momentum indicator is bullish when the price is also in an uptrend, it can be an indication of a strong uptrend. Conversely, if the momentum indicator is bearish when the price is in a downtrend, it can be an indicator of a strong downtrend.

2.3.7. Commodity Channel Index (CCI)

This indicator is introduced by Donald Lambert in 1980 [54]. CCI is a technical analysis indicator that is utilized to assess overbought and oversold situations in a market. It is originally developed for use in the commodity markets, but it is applicable to any time series in financial markets such as stocks, currencies, crypto currencies, etc. The mainstay of the CCI indicator is the idea that prices tend to fluctuate around a moving average, and that deviations from this average can be used to identify potential trend changes. The CCI is calculated using the average deviation of the price from the moving average, and it is drawn as a line on a chart with a scale from such as -200 to $+200$. Values above $+100$ are generally considered to indicate overbought conditions, while values below -100 are considered to indicate oversold conditions. These levels can be used by traders to identify potential points of open and close positions in the market. The calculation of CCI value is done as follows:

Step 1: First of all, average daily price, it is also called Typical Price ($TP[i]$), is calculated as given in Equation (17).

$$TP[i] = \frac{H[i] + L[i] + C[i]}{3} \tag{17}$$

High (H), low (L) and close (C) time series data is used in this equation.

Step 2: Simple Moving Average (SMA) of the $TP[i-n]$ for N period is calculated as given in Equation (18).

$$SMA_N[i] = \frac{1}{N}\sum_{n=1}^{N} TP[i-n], i > N \tag{18}$$

Step 3: Mean Deviation ($MD_N[i]$) value is the average absolute deviation of $TP[i-n]$ from $SMA_N[i-n]$ over a certain number of periods N (usually 20). It is calculated as given in Equation (19).

$$MD_N[i] = \frac{1}{N}\sum_{n=1}^{N} |TP[i-n] - SMA_N[i-n]|, i > N \tag{19}$$

Step 4: The CCI_N value is calculated as the ratio of the difference between $TP[i]$ value and $SMA_N[i]$ value of divide to $MD_N[i]$ value as given in Equation (20).

$$CCI_N[i] = \frac{1}{0.015}\frac{TP[i] - SMA_N[i]}{MD_N[i]} \tag{20}$$

Since it is suggested that 70% to 80% of the $CCI_N[i]$ value lies in a range between 100 and -100, the constant value is initially used as 0.015 [54].

2.3.8. Money Flow Index (MFI)

The MFI is developed by two traders, Avrum Soudack and Gene Quong [55]. MFI is a technical analysis indicator that measures the amount of money flowing in and out of a security over a specific period of time. MFI is similar to RSI in that it measures overbought and oversold conditions in a security. MFI is calculated using both price and volume data. It combines price and volume information to determine the buying and selling pressure behind a security. The calculation of this indicator is started by performing the following steps:

Step 1: First of all, TP value calculation given in Equation (17) is made using the average of H, L and C time series.

Step 2: The $MF[i]$ value is calculated as given in Equation (21) by multiplying the $TP[i]$ value and the transaction volume ($Vol[i]$) value.

$$MF[i] = TP[i] \times Vol[i] \tag{21}$$

Step 3: The $M_{in}[i]$ and $M_{out}[i]$ values are created by comparing the previous $TP[i]$ with the $TP[i-1]$ values. If the $TP[i]$ value is higher than $TP[i-1]$, $M_{in}[i]$ value is equal to $MF[i]$ as given in Equation (22). If the $TP[i]$ value is lower than $TP[i-1]$, $M_{out}[i]$ value is equal to $MF[i]$ as given in Equation (23).

$$M_{in}[i] = \begin{cases} MF[i] & if\ TP[i] > TP[i-1] \\ 0 & if\ otherwise \end{cases} \tag{22}$$

$$M_{out}[i] = \begin{cases} MF[i] & if\ TP[i] < TP[i-1] \\ 0 & if\ otherwise \end{cases} \tag{23}$$

Step 4: The $PMF_N[i]$ value, which is the sum of $M_{in}[i-n]$ is calculated as given in Equation (24). In a same way, the $NMF_N[i]$ value, which is the sum of $M_{out}[i-n]$ is calculated as given in Equation (25).

$$PMF_N[i] = \frac{1}{N}\sum_{n=1}^{N} M_{in}[i-n], i > N \tag{24}$$

$$NMF_N[i] = \frac{1}{N}\sum_{n=1}^{N} M_{out}[i-n], i > N \tag{25}$$

Step 5: Money Ratio ($MR_N[i]$) is calculated as given in Equation (26).

$$MR_N[i] = \frac{PMF_N[i]}{NMF_N[i]} \tag{26}$$

Step 6: $MFI_N[i]$ is then calculated as given in Equation (27) using $MR_N[i]$ expression.

$$MFI_N[i] = 100 - \frac{100}{1 + MR_N[i]} \tag{27}$$

MFI is usually calculated over a 14-day period, although it can be adjusted depending on the trader's preferences. MFI is displayed as a single line that oscillates between 0 and 100. A reading above 80 indicates that a security is overbought, while a reading below 20 indicates that a security is oversold. Traders use MFI to identify potential trend reversals and to confirm trends in conjunction with other technical indicators.

Table 3 illustrates the time series utilized in the calculation of various technical indicators. The SO, WRO, UO, CCI, MFI, and DCC utilize High, Low, and Close prices, with Volume being used only for MFI and DCC. In contrast, the RSI and MO rely solely on the Close price, excluding High, Low, and Volume. The Open price is not used by any of the indicators. Most indicators, including SO, WRO, UO, CCI, MFI, and DCC, incorporate High, Low, and Close for a comprehensive analysis of price movements, while RSI and MO focus exclusively on Close. Volume is relevant only for momentum-based indicators like MFI and DCC, highlighting the diverse approaches of these indicators in analyzing momentum, trend, or volume.

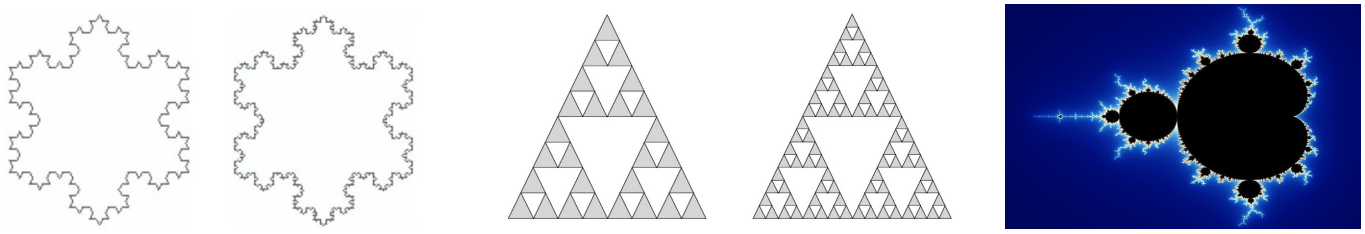
Table 3. HO time series types used in indicator calculations.

Indicators	Open	High	Low	Close	Volume
SO	−	+	+	+	−
WRO	−	+	+	+	−
UO	−	+	+	+	−
RSI	−	−	−	+	−
MO	−	−	−	+	−
CCI	−	+	+	+	−
MFI	−	+	+	+	+
DCC	−	+	+	+	+

2.4. Foundation of Adaptive FDB-SFS Optimization Technique

2.4.1. Fractal Concept

The concept of fractals is first introduced by Mandelbrot and refers to the self-similarity of an object or quantity across all scales [57]. Mandelbrot discovered virtual stock prices using simple rules while he was working at IBM [58]. This also contributed to the development of fractal mathematics and served as an example of computer-generated graphics [58]. Fractals can repeat themselves infinitely by iteration [57]. They are geometric shapes or patterns that exhibit self-similarity, meaning that they have the same or similar patterns at different scales or levels of magnification. In other words, if you zoom in on a fractal, you will see smaller versions of the same shape repeating over and over again. Fractals can be found in nature, such as in snowflakes, branching patterns of trees, the contours of coastlines, the shapes of clouds, mountains, the shape of galaxies and many biological systems. Apart from that, it can also be generated mathematically using iterative processes or recursive formulas such as many man-made objects such as computer-generated graphics and digital images. Generally, fractals have applications in various fields, including computer graphics, art, biology, physics, and economics. They have also become popular in popular culture, appearing in movies, music videos, and video games. Some of the most famous fractals include the Koch snowflake in Figure 6a, the Sierpinski triangle in Figure 6b, and the Mandelbrot set in Figure 6c.



(**a**) (**b**) (**c**)

Figure 6. Example of (**a**) the Koch snowflake [59], (**b**) the Sierpinski triangle [59], (**c**) the Mandelbrot set [60].

2.4.2. Fractal Search Algorithm

The Fractal Search (FS) algorithm integrates fractal growth, specifically through the Diffusion-Limited Aggregation (DLA) approach, with potential theory [35]. This combination effectively balances exploration and exploitation within the search space. The method relies on three straightforward rules to achieve an optimal solution [35]:

Rule 1: Electrical Potential Energy: Each particle in the search space is assigned an electrical potential energy value, which reflects its fitness or state relative to the optimization problem.

Rule 2: Diffusion and Particle Generation: Particles undergo diffusion, leading to the creation of new, randomly distributed particles. The energy of the original particle is redistributed among these newly formed particles, fostering exploration through the introduction of randomness.

Rule 3: Retention of Top Particles: Following diffusion and particle creation, only a small fraction of the highest-performing particles from each generation is retained, while others are discarded. This selective process focuses the search on the most promising regions of the solution space.

Suppose P particles are considered to find the solution, where $1 \leq P \leq 20$. Initially, each particle P_i is randomly placed in the search space, and each particle is assigned equal energy E_i obtained from a specific equation as given in Equation (28).

$$E_i = \frac{E}{P} \tag{28}$$

In this equation, E represents the maximum electric potential energy considered for problem resolution. During the execution of the fractal optimizer, particles undergo diffusion in each generation, generating additional particles through Levy flight. Levy flight, a form of Brownian motion with random steps, occasionally results in a significant leap to another region of space. This flight pattern is analogous to modeling the propagation of an epidemic. In this specific model, it is utilized to simulate Diffusion-Limited Aggregation (DLA) growth. The Levy distribution is defined as given in Equation (29).

$$L(x) = \frac{1}{\pi}\int_0^{\infty} \exp\left(-\alpha q^{\beta}\right)\cos(qx)dx \tag{29}$$

Here, β represents the distribution index, constrained within $0 < \beta \leq 2$, and α denotes the distribution scale factor. During the diffusion process, particles generate new points at random locations. The generation of these points is governed by both Lévy flight and Gaussian distributions, as outlined in Equations (30) and (31), respectively.

$$x_i^q = x_i + \alpha_i^q \otimes Levy(\eta) \tag{30}$$

Here, q indicates the number of particles produced from the diffusion of the primary particle, and the symbol $\otimes$ signifies element-wise multiplication.

$$x_i^q = x_i + \beta \times Gaussian(P_i, |BP|) - \left(\gamma \times BP - \gamma' \times P_i\right) \tag{31}$$

In this equation, β is defined as $\frac{\log(g)}{g}$, where g represents the number of iterations. The term $Gaussian(P_i, |BP|)$ denotes a Gaussian distribution with mean P_i standard deviation equal to the position of the best point as BP, γ and γ' are employed as random parameters sampled within the range [0, 1]. The Fractal Search algorithm alternates randomly between Lévy and Gaussian walks [35]. Lévy distributions typically facilitate rapid convergence, whereas Gaussian walks enhance the precision of the final solution [35].

The search mechanism relies on random walks, which can make rapid convergence uncertain. The parameter α plays a critical role in ensuring swift convergence. Two distinct equations, presented in Equations (32) and (33), define α: one promotes exploration across a wider search space, while the other focuses on achieving a more precise solution.

$$\alpha_i = \frac{\log\left(\min\left(\hat{E}\right)\right) \times (UB - LB)}{g \times \log(E_i)} \tag{32}$$

$$\alpha_i = \frac{(UB - LB)}{(g \times \log(E_i))^{\varepsilon}} \tag{33}$$

In this framework, $min(\hat{E})$ represents the minimum energy of any particle within the system, while UB and LB define the upper and lower boundaries of the search space, respectively. The energy of particle P_i, denoted as E_i, is associated with a fixed power ε, typically set to $\frac{3}{2}$.

Following particle diffusion, energy allocation to the newly created particles is streamlined: particles with higher fitness values are assigned greater energy. Let q represent the number of particles generated from the diffusion of particle P_i with energy E_i. Each new particle, indexed by j (1 to q), is assigned fitness value f_j. The energy distribution is described in Equation (34).

$$E_i^j = \left\{ \left(\frac{f_j}{f_i + \sum_{k=1}^{q} f_k} \right) \right\} \times E_i \tag{34}$$

In (34), f_i denotes the fitness value of the primary particle prior to diffusion. Although this model proved effective for both local and global search tasks, the iterative process increased computational complexity due to the generation of new particles via diffusion. To mitigate this, only a small subset of top-performing particles—less than 10% of the total in each generation—is retained for the subsequent iteration. The energy saved by discarding the remaining particles is then allocated to both the retained particles and the creation of new ones. Let ϕ denote the total energy obtained from discarded particles, and μ represent the rate at which this energy is allocated between retained and newly created particles [35]. The energy distribution for retained particles is outlined in Equation (35).

$$E_{new}^t = E_{old}^t + \left\{ \left(\frac{f_t}{\sum_{k=1}^{\theta} f_k} \right) \times \phi \right\} \times \mu \tag{35}$$

Here, E_{new}^t and E_{old}^t represent the energy of particle t after and before the energy distribution process, respectively, while θ denotes the total number of particles in the iteration. To determine the number of new particles created and randomly placed in the search space for each diffusing particle, Equation (36) is applied.

$$v = \frac{\log(number\ of\ discarded\ particles)}{\log(maximum\ diffusion)} \tag{36}$$

The energy allocated for creating each new particle is uniform across all new particles and is described in Equation (37).

$$E_c' = \frac{\phi \times (1 - \mu)}{v},\ c = 1, 2, \ldots, v \tag{37}$$

The pseudocode for the original Fractal Search (FS) algorithm is detailed in reference [35].

2.4.3. Stochastic Fractal Search (SFS) Algorithm and Its Variants

Stochastic Fractal Search (SFS) algorithm is primarily designed to tackle the crucial challenge of balancing exploitation and exploration, which enhances search diversity in Metaheuristic Search (MHS) methods [35]. However, a significant drawback, common

to many MHS approaches, is its reliance on fitness value information for updates; this mechanism often fails to adequately capture essential concepts like environmental adaptation and can lead to suboptimal survivor selection and premature convergence [36]. To overcome this, the Natural Survival Method (NSM) was proposed as an innovative population management strategy that uses a unique scoring system inspired by natural ecosystems to evaluate an individual's environmental adaptation ability. NSM has been successfully integrated into physics-based SFS, biology-based, and evolution-based methods, and Wilcoxon pairwise comparison (WPC) tests show that these NSM-enhanced variants consistently outperform their counterparts in optimization tasks, although this comes with the drawback of increased computational complexity [36]. Concurrently, the Fitness-Distance Balance (FDB) mechanism, initially shown to deliver superior results when applied to the Symbiotic Organism Search (SOS) algorithm (FDB-SOS) across 90 benchmark functions and tested using the Wilcoxon Rank Sum (WRS) test [37], was adapted to create FDB-SFS. This variant improves candidate selection and has been proven competitive among 39 MHS algorithms, effectively mitigating local optima entrapment and maintaining a balanced exploration-exploitation dynamic as evaluated by WRS and Friedman tests on 94 benchmark functions [38]. Building on these advances, a robust hybrid SFS approach was developed, integrating NSM, FDB guiding mechanisms, and Opposite Based Learning (OBL) techniques, which excels in finding global solution points [39]. To address the issue of premature convergence early in the search, the dynamic dFDB-SFS was introduced; this method adaptively adjusts the coefficient (ω) that governs the exploitation-exploration balance, prioritizing diversity initially and shifting to exploitation later [38,40]. Finally, the AFDB-SFS algorithm further refined this adaptive strategy by incorporating an innovative adaptive switching mechanism that uses chaotic maps and various switching functions to dynamically implement FDB, thus significantly enhancing the search performance and establishing AFDB-SFS as a powerful tool for complex engineering problems [41].

2.4.4. SFS Algorithm

The Stochastic Fractal Search (SFS) algorithm, introduced by Hamid Salimi in 2015 [35], draws inspiration from the natural proliferation of fractals. While the original Fractal Search algorithm excels in problem-solving, it faces challenges, including the need for precise parameter tuning and limited information sharing among particles. To overcome these limitations, the SFS algorithm was developed, incorporating two primary phases: the diffusion process and the update process [35]. During the diffusion process, particles disperse around their current positions, enhancing the likelihood of identifying global minima while avoiding entrapment in local minima. In contrast to the original Fractal Search, SFS employs a static diffusion approach, where the best-performing particle is retained, and others are eliminated. In the update process, particles adjust their positions by leveraging the locations of other group members, thereby improving search efficiency through enhanced information exchange. Two statistical techniques, namely Lévy flight and Gaussian distribution, were explored for the diffusion process [35].

Preliminary studies indicate that Lévy flight offers rapid convergence, whereas Gaussian walks excel at identifying global minima. Consequently, the SFS algorithm adopts Gaussian distribution for its diffusion process [35]. The Gaussian walks involved in diffusion are defined by Equations (38) and (39), with diversification achieved through the application of random methods.

$$GW1 = Gaussian(\mu_{BP}, \sigma) - \left(\varepsilon \times OP - \varepsilon' \times P_i\right) \tag{38}$$

$$GW2 = Gaussian(\mu_P, \sigma) \tag{39}$$

In this framework, ε and ε' are uniformly distributed random numbers within the range [0, 1]. The terms OP and P_i denote the optimal position and the position of the i-th particle in the group, respectively. The Gaussian distribution employs four parameters: the first two are μ_{BP}, set equal to OP, and σ; the latter two are μ_P, equal to P_i, and σ. The standard deviation for these parameters is computed as specified in Equation (40).

$$\sigma = \left| \frac{\log(g)}{g} \times (P_i - OP) \right| \tag{40}$$

The algorithm addresses a D-dimensional optimization problem, with individuals initialized randomly within predefined bounds. The initialization of each individual is determined using the equation provided in Equation (41).

$$P_j = LB + \varepsilon \times (UB - LB) \tag{41}$$

Here, ε is a uniformly distributed random number in the range [0, 1]. After initialization, the fitness function of each point is evaluated to identify OP. The algorithm operates through two core processes: diffusion and exploration. Diffusion facilitates exploitation by allowing particles to move around their current positions, while exploration enhances the search across the solution space. The exploration phase incorporates two statistical methods. In the first method, all points are ranked according to their fitness values, and each point is assigned a probability as described in Equation (42).

$$P_{ai} = \frac{rank(P_i)}{N_M} \tag{42}$$

Here, $rank(P_i)$ represents the rank of point P_i, and N_M is the total number of points. This probability ensures that higher-quality solutions are more likely to be selected.

In the second statistical method, points meeting a specific probability condition are updated using Equation (43).

$$P_i'(j) = P_r(j) - \varepsilon \times (P_t(j) - P_i(j)) \tag{43}$$

Here, P_i' denotes the new position of point P_i, P_r and P_t are randomly selected points from the group. These processes aim to improve the likelihood of discovering superior solutions and enhance the optimization process. The first statistical operation modifies a point's position by considering the positions of other points in the group, promoting diversification. Before proceeding to the second operation, the scores from the first statistical method are ranked again according to Equation (42). If a new point, P_i', satisfies the condition $P_{ai} < \varepsilon$, its position is updated using Equations (44) and (45); otherwise, no update is performed [35].

$$P_i'' = P_i' - \hat{\varepsilon} \times (P_t' - BP) \quad if\ \hat{\varepsilon} \leq 0.5 \tag{44}$$

$$P_i'' = P_i' + \hat{\varepsilon} \times (P_t' - P_r') \quad if\ \hat{\varepsilon} > 0.5 \tag{45}$$

The parameter $\hat{\varepsilon}$ consists of random numbers generated by a Gaussian normal distribution [35]. The pseudocode for the original Stochastic Fractal Search (SFS) algorithm is provided in reference [35].

2.4.5. Fitness-Distance Balance (FDB) SFS Algorithm

The Fitness-Distance Balance (FDB) approach, introduced by Kahraman et al. in 2020, was effectively integrated into the Symbiotic Organisms Search (SOS) algorithm [37]. The primary goal of the FDB method is to enhance the performance of MHS algorithms by guiding the search process efficiently. Unlike other selection techniques, the FDB method

calculates score values for solution candidates, using these scores as the basis for selection. The scoring process takes into account both the fitness values of the candidates and their distance from the population's best solution [37,38].

This approach prioritizes candidates with high fitness while avoiding the selection of those too similar to the best solution, thereby mitigating the risk of premature convergence commonly faced in MHS algorithms. The implementation of the FDB method involves the following steps:

Step 1: Determine the score values for the solution candidates within the population P. Let X_{best} represent the optimal solution in the population. The scores of these candidates are represented by the S_p vector, as shown in Equation (46).

$$S_p = \begin{bmatrix} s_1 \\ \vdots \\ s_n \end{bmatrix}_{n \times 1} \tag{46}$$

Step 2: When determining the score of each solution candidate, both their fitness and distance values are considered. The distance between the i-th solution candidate in the P-population and X_{best}, the optimal solution in P, is computed as outlined in Equation (47).

$$D_{P_i} = \sqrt{\left(x_1^{P_i} - x_1^{P_{best}}\right)^2 + \left(x_2^{P_i} - x_2^{P_{best}}\right)^2 + \cdots + \left(x_m^{P_i} - x_m^{P_{best}}\right)^2} \tag{47}$$

The Euclidean metric is employed to compute the distance. Accordingly, the distance vector D_p for the solution candidates in the P-population is expressed as shown in Equation (48).

$$D_p = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}_{n \times 1} \tag{48}$$

Step 3: The fitness value is another key parameter in the score calculation. The fitness values of the solution candidates are captured in the F vector. To avoid dominance during score computation, both the F and D_p vectors are normalized. The score for the i-th solution candidate is calculated as shown in Equation (49).

$$S_{P_i} = \omega \times normF_{P_i} + (1 - \omega) \times normD_{P_i} \tag{49}$$

Step 4: The ω parameter regulates the influence of fitness and distance values, taking any value within the range [0, 1]. When ω = 1, the score depends entirely on fitness, favoring exploitation. When ω = 0, it relies solely on distance, promoting exploration. This adjustable parameter enables the FDB method to effectively balance exploration and exploitation in MHS algorithms [37].

Similarly to other MHS approaches, achieving a balance between exploitation and exploration in the SFS algorithm can be challenging [38]. To address this, the FDB method introduces a diversity operator in the SFS algorithm, offering an innovative approach inspired by natural fractal patterns [37,38]. The pseudocode for the FDB-SFS algorithm is provided in reference [38].

2.4.6. Dynamic FDB-SFS Algorithm

The dynamic Fitness-Distance Balance (*dFDB*) method, proposed by Kahraman et al. (2022) [40], enhances population-based MHS algorithms by dynamically balancing exploitation and exploration to efficiently converge to the global optimum. Unlike the static

Fitness-Distance Balance (FDB) method, *dFDB* employs a dynamic weight coefficient, ω_{dFDB}, which varies within [0, 1] throughout the search process. This variation is governed by three parameters: the fitness evaluation number (*fes*) counter, the f-frequency, and the switching reference signal. These parameters adjust ω_{dFDB} based on the search lifecycle, typically following a negative sawtooth function [40]. The dFDB score for solution candidates in the population P is calculated as in Equation (50).

$$S_{dFDB\_P_i} = \omega_{dFDB} \times normF_{P_i} + (1 - \omega_{dFDB}) \times normD_{P_i} \quad (50)$$

A large ω_{dFDB} (close to 1) emphasizes the fitness term, promoting exploitation by prioritizing solutions with superior fitness values for intensive local searches. Conversely, a smaller ω_{dFDB} increases the influence of the distance term, improving exploration by favoring various solutions different from Pbest and thus reducing the risk of getting stuck in local optima. This dynamic balance guarantees robust and efficient optimization.

The dFDB-SFS algorithm, an enhancement of the Stochastic Fractal Search (SFS), modifies the guide selection in its update mechanisms to improve optimization performance. In the first update mechanism, SFS uses three guides, with Guide-1 selected randomly, leading to inefficient random searches in complex problem spaces. In contrast, dFDB-SFS selects Guide-1 using the dynamic Fitness-Distance Balance (dFDB) method, which leverages both fitness and distance, enhancing search efficiency. In the second update mechanism, SFS employs two convergence equations with 50% probability, using a sequential guide selection method. dFDB-SFS replaces this with dFDB-based selection for the guide P_i, combining fitness and distance scores to improve adaptability and diversity. The 50% sequential and 50% dFDB-based guidance in dFDB-SFS mitigates local optima traps and premature convergence. By dynamically balancing exploitation (via fitness) and exploration (via distance), dFDB-SFS outperforms SFS, as evidenced in experimental comparisons [40].

2.4.7. Adaptive FDB (AFDB)-SFS Algorithm

The Adaptive FDB-SFS method, proposed by Duman et al. (2023) [41]. The AFDB-SFS method is given in Algorithm 1. According to Algorithm 1, the search process consists of two main stages: exploitation and exploration. These stages are critical to improve the performance of the model. The AFDB-SFS algorithm differs from the SFS and FDB-SFS algorithms. In particular, innovations in the design of the exploration phase of these algorithms increase the effectiveness of AFDB-SFS. The differences in the exploration phases of these three algorithms are explained in detail in the method section. In the AFDB-SFS algorithm, two different selection methods are applied to determine the P_i vector: the rank-based method and the FDB-based selection method. The adaptive switching mechanism comes into play to decide which of these methods will be used in the search process. The adaptive switching mechanism enables the algorithm to dynamically adapt to the environmental conditions and the structure of the data. Therefore, this process is defined as "adaptive exploration" in AFDB-SFS. The adaptive switching mechanism and guide selection process are implemented in lines 18 and 21 of Algorithm 1. These lines include information about how the algorithm is optimized and how the selected methods come into play. Signal generation for adaptive switching in AFDB-SFS algorithm is given also in Algorithm 2. Due to its strong performance in the AFDB-SFS algorithm [41], in this study, the Singer map is utilized in Algorithm 2. In this study, 30 iterations are used, and the variations of each chaotic map function listed in Table 4 are illustrated in Figure 7. In Algorithm 1, the adaptive exploration process, the population selection mechanism dynamically balances exploration and exploitation as iterations progress. This balance is achieved by employing a binary signal generated using a threshold value derived from the chaotic Singer map, which determines whether the AFDB-SFS or SFS algorithm is selected at each step. The

deterministic nature of the Singer map, with fixed parameters, ensures reproducibility, as identical initial conditions yield the same sequence over 30 iterations. This sequence, illustrated in Figure 7, generates a threshold value that is compared against a predefined criterion to produce a binary signal (0 or 1), where 0 triggers SFS and 1 activates AFDB-SFS in Algorithm 1. This adaptive switching mechanism allows the algorithm to emphasize exploration in early iterations by favoring AFDB-SFS's chaotic optimization, which explores diverse feature combinations, and shift toward exploitation in later iterations by leveraging SFS's simpler selection strategy.

Algorithm 1. AFDB-SFS pseudo code [41]

Initialization
P: create a population of n particles (fractals), *fes* = 0;
for *i* = 1 to *n* **do**
    F: calculate the fitness values of the points
end for
Sort the population depending on the fitness values of fractals
while search process lifecycle to *maxFEs* **do**
Exploitation Process
    for *i* = 1 to *n* **do**
        Selection: select the particle *i* from *P*-population and select two particles
        from *P*-population randomly
        for *j* = 1 to *k* **do**
            Intensification: randomly select a Gaussian walk to diffusion.
            Use the *i*th particle and the best particle in *P* and update the fitness
            of *k*-particles.
            fes = *fes* + 1;
        end for
    end for
Adaptive Exploration Process
    s = **Algorithm. 2.** (*fes*, *maxFEs*, *f*, *reference_signal*)
    "*s* is the binary signal generated for adaptive implementation of FDB"
    for *i* = 1 to *n* **do**
        if *s* **then**
            Selection in AFDB-SFS: select the point *i* from *P* using the FDB
            method and select two points from *P* randomly.
        else
            Selection in SFS: select the particle *i* from *P*-population and
            select two particles from *P*-population randomly
        end if
        Diversification: The first update process, *fes* = *fes* + 1;
    end for
    for *i* = 1 to *n* **do**
        Select the particle *i* from P-population and select two particles from
        P-population randomly
        Diversification: The second update process, *fes* = *fes* + 1;
    end for
    Next generation until termination criterion
end while

The main purpose of designing it in the AFDB-SFS algorithm is to finetune the exploitation–exploration balance and ensure the continuity of this balance. This balance directly affects the overall performance of the algorithm, increasing the learning capacity of the model. Other operations in the pseudocode given in Algorithm 1 are similar to the SFS and FDB-SFS algorithms. This similarity allows AFDB-SFS to achieve better performance by building on existing methods. Table 4 lists ten prominent chaotic map functions, their mathematical equations, and parameter ranges. It also provides the mathematical formulation and parameter settings of the Singer map, while its uniform chaotic behavior supports a robust adjustment of the exploration-exploitation trade-off in [41].

When the literature is examined, it is observed that SFS and its variants are frequently encountered [42]. However, the examination of significant variants, namely the SFS, FDB-SFS, dFDB-SFS, and AFDB-SFS algorithms, will be within the scope of this study. The performance and characteristics of SFS and its variants, including FDB-SFS, dFDB-SFS, and AFDB-SFS, are summarized in Table 5, which highlights their key features, evaluation methods, strengths, and limitations.

Algorithm 2. Signal generation for adaptive switching in AFDB-SFS algorithm [41]

Inputs
fes, *maxFEs*, *f*, *reference_signal*
Output
true/false
1: n = 1:*maxFEs*
2: z←round(*maxFEs/f*)
3: y←mod(*fes*,*z*)
Use a reference signal to generate a *thld* (switching threshold value)
4: *thld* ←*freference_signal*(*y*)
5: **if** (rand(0,1) > *thld*) then **return** true
6: **else return** false

Table 4. Chaotic map functions [61].

No	Function Name	Equation	Range
1	Chebyshev map	$x_{i+1} = \cos(\cos^{-1}(x_i))$	$(-1, 1)$
2	Circle map	$x_{i+1} = mod\left(x_i + v - \left(\frac{u}{2\pi}\right)\sin(2\pi x_i), 1\right), u = 0.5, v = 0.2$	(0, 1)
3	Gauss/mouse map	$x_{i+1} = \begin{cases} 1 & if\ x_i = 0 \\ \frac{1}{mod(x_i,1)} & if\ otherwise \end{cases}$	(0, 1)
4	Iterative map	$x_{i+1} = \sin\left(\frac{u\times\pi}{x_i}\right)$	$(-1, 1)$
5	Logistic map	$x_{i+1} = u \times x_i(1 - x_i), u = 4$	(0, 1)
6	Piecewise map	$x_{i+1} = \begin{cases} \frac{x_i}{T} & if\ 0 \le x_i < T \\ \frac{x_i - T}{0.5 - T} & if\ T \le x_i < 0.5 \\ \frac{1 - T - x_i}{0.5 - T} & if\ 0.5 \le x_i < 1 - T \\ \frac{1 - x_i}{T} & if\ 1 - T \le x_i < 1 \end{cases}$	(0, 1)
7	Sine map	$x_{i+1} = \frac{u}{4}\sin(\pi \times x_i)$, u = 4	(0, 1)
8	Singer map	$x_{i+1} = u\left(7.86x_i - 23.31x_i^2 + 28.75x_i^3 - 13.302875x_i^4\right),\ u = 1.07$	(0, 1)
9	Sinusoidal map	$x_{i+1} = u \times x_i^2\sin(\pi \times x_i)$, u = 2.3	(0, 1)
10	Tent map	$x_{i+1} = \begin{cases} \frac{x_i}{0.7} & if\ x_i < 0.7 \\ \frac{10}{3} \times (1 - x_i) & if\ x_i \ge 0.7 \end{cases}$	(0, 1)

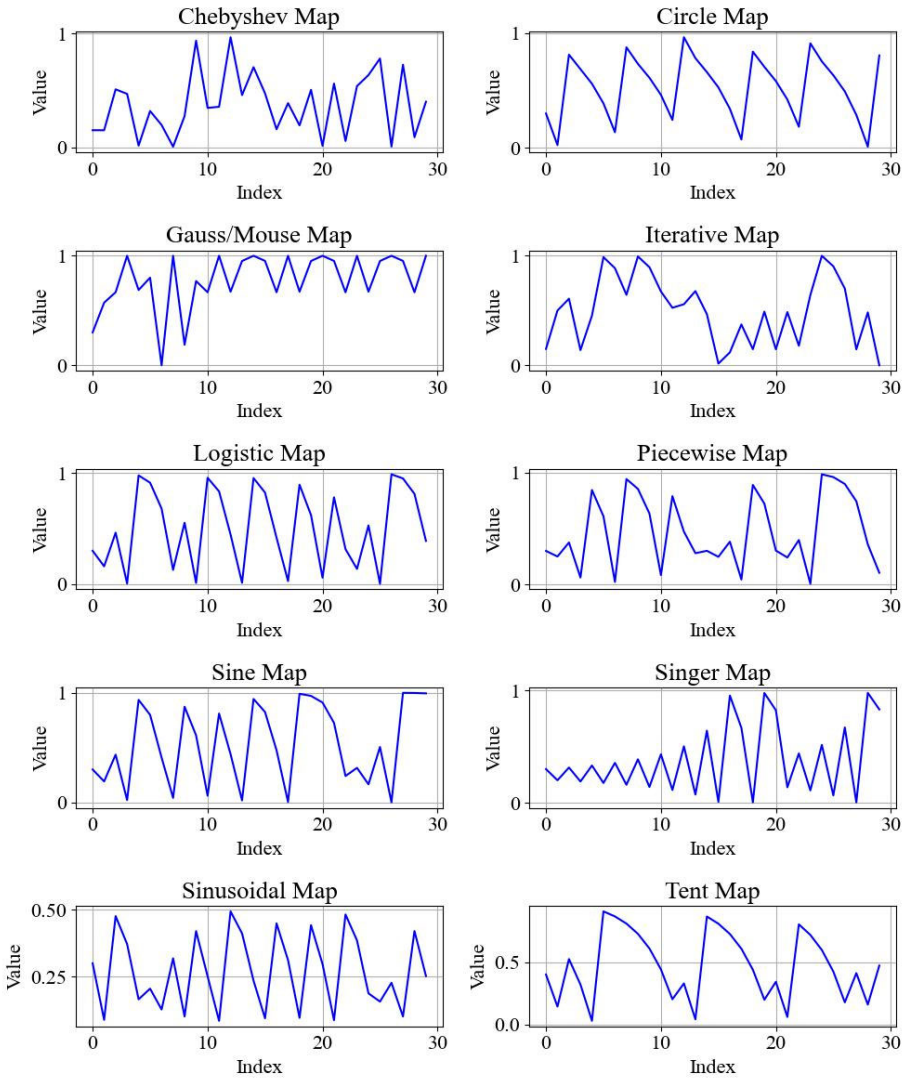


Figure 7. Reference signal derived from chaotic map functions for maxFEs = 30.

Table 5. Evaluation and comparison of SFS and its variants.

Studies	Algorithm	Key Features	Evaluation Methods	Strengths	Limitations
[35]	SFS	Balances exploitation-exploration, enhances search diversity	ANOVA test	Improves diversity in search	Poor environmental adaptation, prone to premature convergence
[38]	FDB-SFS	Integrates Fitness Distance Balance for guided search (used in SOS and SFS)	Wilcoxon and Friedman	Outperforms other MHS methods, avoids local optima	May struggle with scalability in high-dimensional problems due to fitness-distance balance calculations
[40]	dFDB-SFS	Dynamically adjusts exploitation-exploration balance via coefficient (w)	Wilcoxon and Friedman	Addresses premature convergence with adaptive phases	Increased computational complexity due to dynamic weighting and switching signal
[41]	AFDB-SFS	Adaptive switching with chaotic maps and functions	Wilcoxon and Friedman	Fine-tunes balance, enhances search with adaptive exploration, converges closest to optimal point	Higher computational overhead due to adaptive switching and chaotic map integration

2.5. LSTM Neural Networks

2.5.1. LSTM Model

Long Short-Term Memory (LSTM) is a specialized type of Recurrent Neural Network (RNN) designed to handle time series and sequential data, effectively overcoming the vanishing gradient issue common in standard RNNs [62]. This makes LSTM well-suited for learning long-term dependencies, enabling its success in applications such as language modeling, speech recognition, machine translation, and time series prediction. A defining

feature of LSTM is its cell state, which serves as an internal memory to retain information over extended periods, updated based on input and output signals. The LSTM architecture incorporates three primary gates—forget, input, and output—that regulate information flow. The forget gate, as defined in Equation (51), determines which information from the cell state should be discarded.

$$f_t = \sigma\Big(\omega_f[h_{t-1}, x_t] + b_f\Big) \tag{51}$$

The input gate, described in Equation (52), controls the extent to which new information is incorporated into the cell state.

$$i_t = \sigma(\omega_i[h_{t-1}, x_t] + b_i) \tag{52}$$

The output gate, specified in Equation (53), governs the information to be output from the model.

$$Q_t = \sigma(\omega_o[h_{t-1}, x_t] + b_o) \tag{53}$$

It decides what information will be output from the model. Additionally, the candidate value for updating the cell state is calculated as shown in Equation (54).

$$\widetilde{C}_t = tanh(\omega_c[h_{t-1}, x_t] + b_c) \tag{54}$$

It helps control the flow of information. The current cell state C_t is calculated using the previous cell state C_{t-1} along with the input and forget gates, as shown in Equation (55).

$$C_t = C_{t-1} \times f_t + i_t \times \widetilde{C}_t \tag{55}$$

The current hidden state $h_{t_{LSTM}}$, computed with the output gate and current cell state, is represented as in Equation (56).

$$h_{t_{LSTM}} = Q_t \times \tanh(C_t) \tag{56}$$

Additionally, the sigmoid activation function is defined as in Equation (57).

$$f(x) = \frac{1}{1 + exp^{-x}} \tag{57}$$

It is transforming an input value into an output between 0 and 1, while the tanh activation function is given as in Equation (58).

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{58}$$

It transforms an input value into an output between -1 and 1. Overall, LSTM's architecture effectively addresses the vanishing gradient problem, excelling in processing sequential data and capturing long-term dependencies.

2.5.2. Bi-LSTM Model

Introduced by Schuster et al. in 1997, the Bidirectional Recurrent Neural Network (BRNN) addresses the limitation of unidirectional RNNs, which struggle to incorporate historical context [63]. BRNNs employ two recurrent neural networks that process data in both forward and backward directions. The Bidirectional Long Short-Term Memory (Bi-LSTM) model enhances this framework by integrating LSTM units into the BRNN structure, combining the strengths of both architectures. In the Bi-LSTM, the forward

recurrent layer is defined in Equation (59), while the backward recurrent layer is specified in Equation (60).

$$h_t^f = \Omega\left(W_{xh}^f \times x_t + W_{hh}^f \times h_{t-1} + b^f\right) \tag{59}$$

$$h_t^b = \Omega\left(W_{xh}^b \times x_t + W_{hh}^b \times h_{t-1} + b^b\right) \tag{60}$$

The weight matrices W_{xh}^f and W_{xh}^b define the relationships between the input x_t and the hidden states in the forward and backward layers, respectively. Similarly, the weight matrices W_{hh}^f and W_{hh}^b connect the previous hidden state h_{t-1} to the current hidden state. Bias terms b^f and b^b are applied in the forward and backward layers, respectively, while Ω, a non-linear activation function, is used to compute the hidden states. The bidirectional hidden state h_t is obtained by combining the hidden states from both directions, as expressed in Equation (61).

$$h_t = h_t^b + h_t^f \tag{61}$$

In this study, the Bi-LSTM model utilizes the following parameters: h_t^b and h_t^f represent the hidden states at time step t for the forward and backward recurrent layers, respectively.

2.6. Performance Measurement of Models

The effectiveness of candidate solutions is assessed using several metrics, including the Pearson Correlation Coefficient, Root Mean Square Error (RMSE), and the Number of Selected Features (NSF). The Pearson Correlation Coefficient measures the strength of the linear relationship between two variables, with values spanning from -1 to 1. A value of 1 represents a perfect positive linear correlation, -1 indicates a perfect negative linear correlation, and 0 suggests no linear relationship. The formula for the Pearson Correlation Coefficient is provided in Equation (62).

$$R = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 \times \sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{62}$$

where x_i and y_i are the i-th observations of the variables x and y, $\overline{x}$ and $\overline{y}$ are their respective means, and n is the number of observations. RMSE quantifies the difference between predicted and actual values, with lower RMSE values indicating superior model performance. The RMSE is calculated as shown in Equation (63).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{63}$$

In this equation, y_i denotes the actual values, $\hat{y}_i$ represents the predicted values. During optimization, the objective is to maximize performance while minimizing the number of features by eliminating redundant or less impactful ones. A lower NSF ratio reflects reduced model complexity. The Number of Selected Features Ratio (NSFR) is defined as the proportion of features selected from the feature pool for model training, as described in Equation (64).

$$NSFR = \frac{Number\ of\ selected\ features}{Number\ of\ extracted\ features} \tag{64}$$

The fitness function, utilized in methods such as SFS, FDB-SFS, dFDB-SFS, and AFDB-SFS, is designed to balance three objectives: maximizing the Pearson Correlation Coefficient

(R), minimizing RMSE, and reducing the NSFR to lower model complexity. This fitness function is expressed in Equation (65).

$$Fitness\ Function = \alpha_f \times (1 - R) + \beta_f \times RMSE + \gamma_f \times NSFR \quad (65)$$

Here, α_f, β_f, and γ_f are each set to $0.\overline{3}$, reflecting the equal weighting of the three components. The term R refers to the Pearson Correlation Coefficient, RMSE represents the Root Mean Square Error, and $NSFR$ indicates the ratio of selected features for the solution being evaluated.

In the simulation study, the most suitable features for the GA, PSO, SFS, FDB-SFS, dFDB-SFS, and AFDB-SFS methods are objectively selected from an initial set of 189 features based on the explicit selection criterion defined in Equation (65). This criterion aims to maximize correlation with the fitness function, minimize the error rate, and reduce model complexity. There is no minimum selection requirement for a specific set of financial indicators, such as Fast%K, Fast%D, WRO, UO, RSI, MO, CCI, MFI, or DCC. Accordingly, the algorithms are free to ignore any irrelevant set of indicators, allowing them to select features from groups that contribute more significantly to model performance.

3. Empirical Results and Discussion

In this study, the simulations were conducted on a high-performance system equipped with an NVIDIA Tesla V100 GPU and 16 virtual CPUs, using MATLAB (2024b) for signal processing and simulation, and Python programming language (version 3.10.19) within the Spyder Integrated Development Environment (IDE) and R Programming (Rstudio 2022 12.0+353) for data visualization. The embedded system application was successfully implemented and tested on the NVIDIA Jetson Orin Nano Developer Platform, ensuring efficient real-time performance for time series forecasting. Full hardware and software specifications are provided in Tables 6 and 7, respectively. The hardware setup is ideal for tasks requiring high-performance computing and deep learning processes, offering sufficient computational power and memory to handle large-scale data and complex models efficiently.

Table 6. Hardware specifications for computations.

	Component	Specification
Simulation	GPU	NVIDIA Tesla V100 (16 GB HBM2)
	CPU	16 Virtual CPUs (Intel Xeon Broadwell)
	RAM	32 GB
	Storage	SSD-based high-speed storage
Embedded System	GPU	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
	CPU	6-core Arm Cortex-A78AE
	RAM	8 GB 128-bit LPDDR5
	Storage	SSD-based high-speed storage

Table 7. Software specifications for computations.

Component	Specification
Operating System	Windows 10
Signal Processing and Simulation	MATLAB
Data Visualization	Python, R Programming
Embedded System Application	Python

3.1. Time Period Analysis of HO Close Price and Features

In Figure 8, it is a multi-panel time series plot showing the close price of HO and various technical indicators from 2009 to 2025 years. It includes nine subplots, each with time on the x-axis and indicator values on the y-axis. The top panel plots the close price (0.5–4 units, likely dollars) in blue, with fluctuations marked by red and gray dashed lines for start/end dates and yearly boundaries. The second panel shows the SO with Fast%K (purple) and Fast%D (orange) ranging 0–100, exhibiting high-frequency oscillations. The third panel displays WRO in green (−100 to 0), the fourth UO in red (−50 to 50), the fifth RSI in brown (0–100), the sixth MO in teal (−1 to 1), the seventh CCI in gray (−250 to 250), the eighth MFI in olive (0–100), and the ninth DCC in cyan (0–100), all indicating market trends and momentum. The x-axis spans 1 May 2009 to 30 May 2025 with rotated ticks, using dashed vertical markers. Their parameters are presented in Table 8. For each indicator type, 21 features are created. Since both Fast%K and Fast%D are created for SO, a total of 42 features is created from SO.

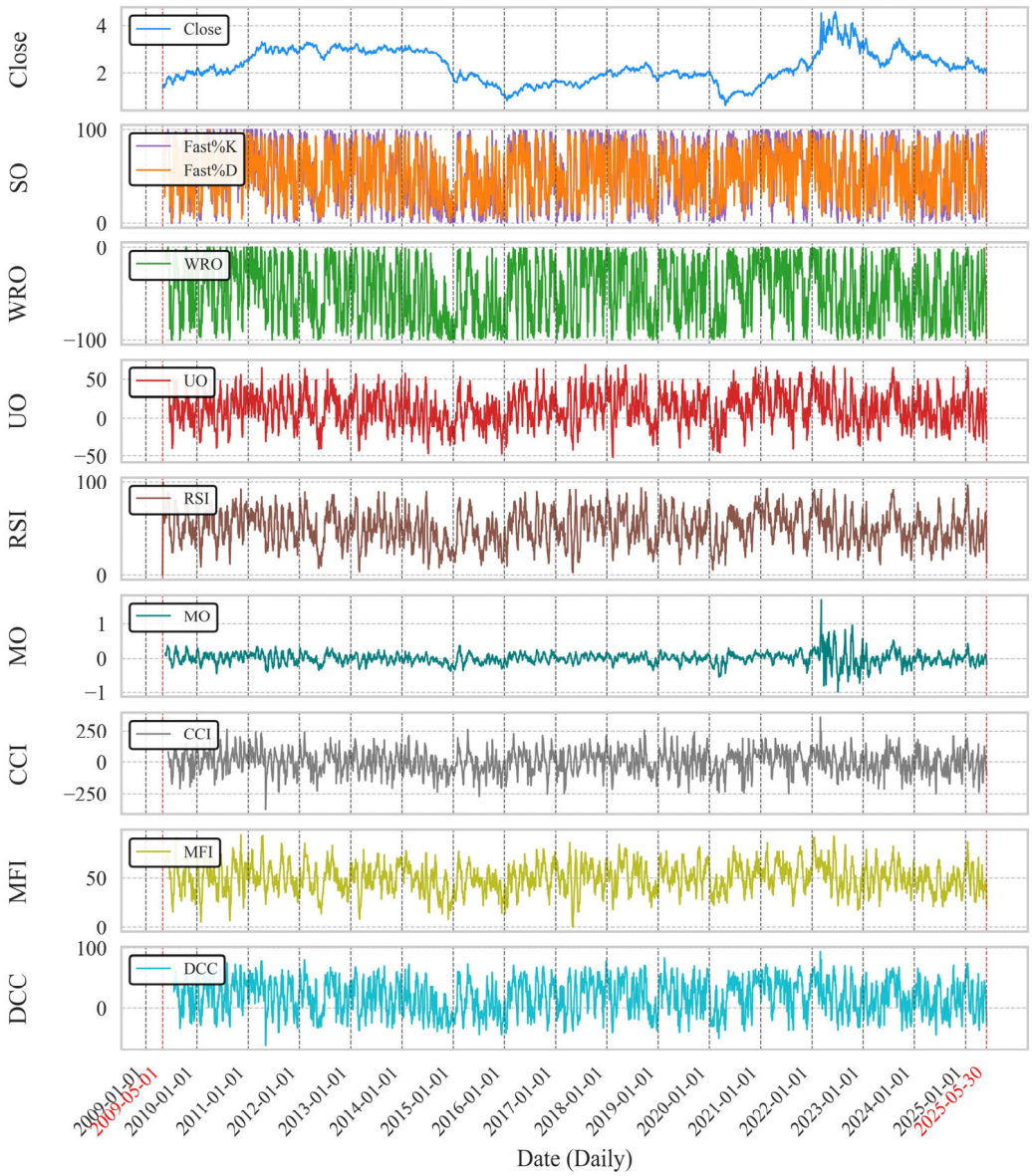


Figure 8. Close price of HO and time series features.

Table 8. Ranges of indicator parameters used for the created feature pool.

Indicator	Parameters
Fast%K	$N_{\%K}$ = 5:1:25
Fast%D	$N_{\%D}$ = 5:1:25
WRO	N_{WRO} = 5:1:25
UO	N_{UO} = 5:1:25
RSI	N_{RSI} = 5:1:25
MO	N_{MO} = 5:1:25
CCI	N_{CCI} = 5:1:25
MFI	N_{MFI} = 5:1:25
DCC	N_{DCC} = 5:1:25

In Figure 9, a chord diagram is used to visualize the relationships between features. In the diagram, each feature is represented by a segment and the lines between the segments show the degree of similarity or interaction of the features. The thickness and color intensity of the lines indicate the strength of the relationships. This diagram is designed to map the correlation between the features and the relational structure in the data set. It has been evaluated as an effective visualization tool for the analysis of interactions between financial indicators.

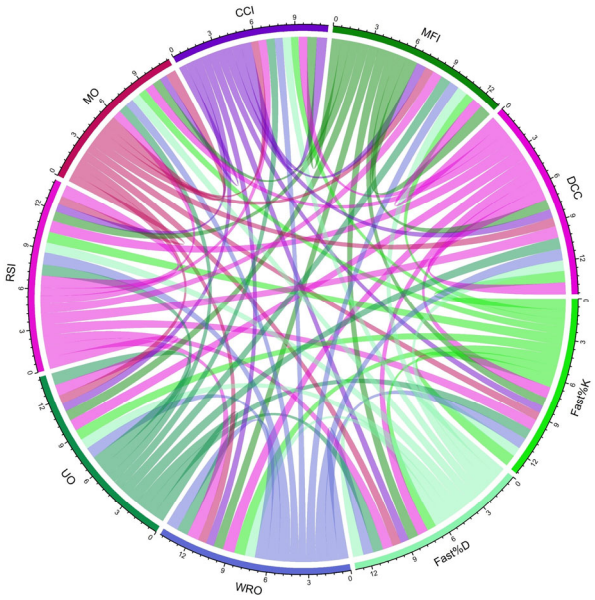


Figure 9. Chord diagram of features.

3.2. Models and Methods: Parameters and Values

In the simulation study, the parameters of the methods and models used are detailed in Table 9. GA and PSO, which are widely accepted and frequently used meta-heuristic optimization methods, are used to solve feature selection problems in this study. In addition, SFS and its variants FDB-SFS, dFDB-SFS, and AFDB-SFS, which forms the backbone of this study are optimization algorithms designed to solve feature selection problem. In GA, it is employed with a population size of 20, a maximum of 30 generations, a crossover rate of 0.80, a mutation rate of 0.10, the Roulette Wheel selection method, and Single-Point crossover type, while the Particle Swarm Optimization (PSO) is utilized with 20 particles,

a maximum of 30 generations, cognitive and social factors set to 2, and an inertia weight of 1, respectively.

Table 9. Parameters and values of models and methods used in the study.

Models/Methods	Parameters	Value
GA	Population Size	20
	Maximum Generation	30
	Crossover Rate	0, 80
	Mutation Rate	0, 10
	Selection Method	Roulette Wheel
	Crossover Type	Single-Point crossover
PSO	Number of Particles	20
	Maximum Generation	30
	Cognitive Factor	2
	Social Factor	2
	Inertia Weight	1
SFS, FDB-SFS, dFDB-SFS, AFDB-SFS	Start Point	20
	Maximum Generation	30
	Maximum Diffusion	1
SFS	Walk Function First Update Mechanism Second Update Mechanism	$GW1$ in Equation (38) Equation (43) Equations (44) and (45)
FDB-SFS	Walk Function First Update Mechanism Second Update Mechanism ω	$GW1$ in Equation (38) Equation (43) Equations (46)–(49) 0.5
dFDB-SFS	Walk Function	$GW1$ in Equation (38)
	First Update Mechanism	Equation (43)
	Second Update Mechanism	Equations (46)–(49)
	ω_{dFDB}	0.1:0.1:0.9
AFDB-SFS	First Update Mechanism	Like SFS Method
	Second Update Mechanism	Equations (46)–(49)
	Reference Signal Generator for Adaptive Switching	Chaotic Singer map in Table 4
LSTM, Bi-LSTM	Optimizer	Adam
	Activation Function	Tanh
	Batch Size	32
	Number of Neurons	250
	Epoch	30
	Dropout Rate	0.10
	Learning rate	0.005
	Kernel Initializer	glorot

SFS uses 20 starting points, 30 maximum generations, and 1 maximum diffusion using the *GW*1 walk function (Equation (38)). The first update mechanism follows Equation (43), while the second uses Equations (44) and (45). FDB-SFS shares the same walk function and first update mechanism, but applies Equations (46)–(49) for the second update and sets the weight parameter ω to 0.5. dFDB-SFS mirrors FDB-SFS in structure, but changes ω from 0.1 to 0.9 in 0.1 increments. AFDB-SFS adopts the same first update mechanism as SFS, uses Equations (46)–(49) for the second update, and includes a chaotic Singer map in Table 4 as the reference signal generator. The LSTM and the Bi-LSTM model architecture, employed for predicting the target variable based on the features selected by AFDB-SFS and other optimization techniques, are detailed to ensure replicability by other researchers. The architecture consists of a single LSTM or Bi-LSTM layer followed by a dense output layer. Table 9 summarizes the key hyperparameters and configurations, optimized through grid search on the training set. The model uses the Adam optimizer with a learning rate of 0.005, a Tanh activation function, and a Glorot kernel initializer to stabilize weight initialization. The recurrent layer contains 250 neurons, processing input sequences with a dimensionality of up to 189, depending on the number of selected features (which may be fewer than 189). A dropout rate of 0.10 is applied to prevent overfitting, and the batch size is set to 32. The model is trained for 30 epochs, with early stopping implemented to halt training if the validation loss does not improve for 5 consecutive epochs.

3.3. Model Training, Validation and Test Stages for Deep Learning Models

In simulation stage, information is provided about the characteristics of the data used for training and testing the LSTM and Bi-LSTM models, as well as the metrics used to evaluate model performance. To ensure that meaningful values are generated under different parameters, excluding "NaN" values, feature extraction approaches from 4260 data points are evaluated by performing predictions starting from the 100th value during the training phase. 15% of the dataset is reserved for the testing phase. The data from 1 May 2009, to 24 February 2022, is designated as the training dataset, while the data from 27 February 2023, to 30 May 2025, is designated as the test dataset. The dataset (4260 samples) is split into 85% (3611 samples) for model development and 15% (649 samples) for testing, with the test set reserved exclusively for final evaluation to prevent data leakage. The 3611-sample development set is divided using the hold-out validation method into 90% training (3250 samples) and 10% validation (361 samples) sets, with 100 samples excluded from training and validation. These 100 samples are reserved to support the calculation of financial signal processing methods, ensuring that the model development process remains independent of the data used for these computations, thus preventing potential bias or leakage. Early stopping (5 epochs tolerance) and z-score normalization mitigate over-fitting, while the temporal split ensures generalizability.

3.4. Performance of GA and PSO Based Models

The GA and PSO algorithms are integrated with Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) models, resulting in the GA-LSTM, GA-Bi-LSTM, PSO-LSTM, and PSO-Bi-LSTM methods, to provide a more comprehensive benchmarking widely used feature selection and prediction techniques, thereby enabling a robust evaluation of the relative merit of the proposed model. Figures 10 and 11 present the performance comparisons of the following models, respectively: GA-LSTM vs. GA-Bi-LSTM (Figure 10) and PSO-LSTM vs. PSO-Bi-LSTM (Figure 11). These box-whisker plots compare the fitness values of two different models over a certain number of generations. The X-axis represents the generation number, and the Y-axis represents the fitness value. The orange boxes in the graph represent the LSTM model architecture, and the blue boxes represent the Bi-

LSTM model architecture. The average fitness values of both models are also shown with dashed lines.

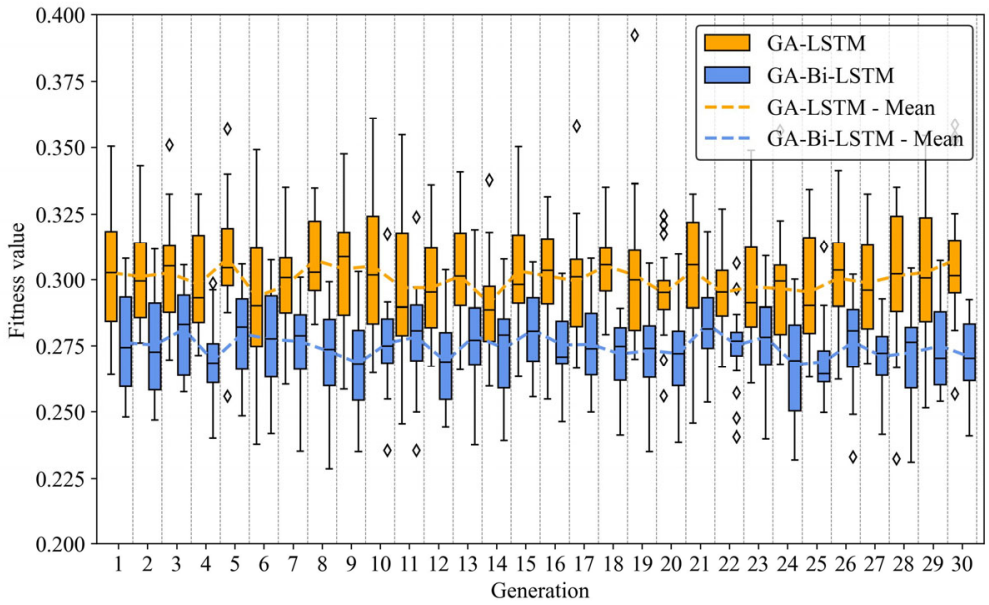


Figure 10. Performance comparison of GA-LSTM and GA-Bi-LSTM models.

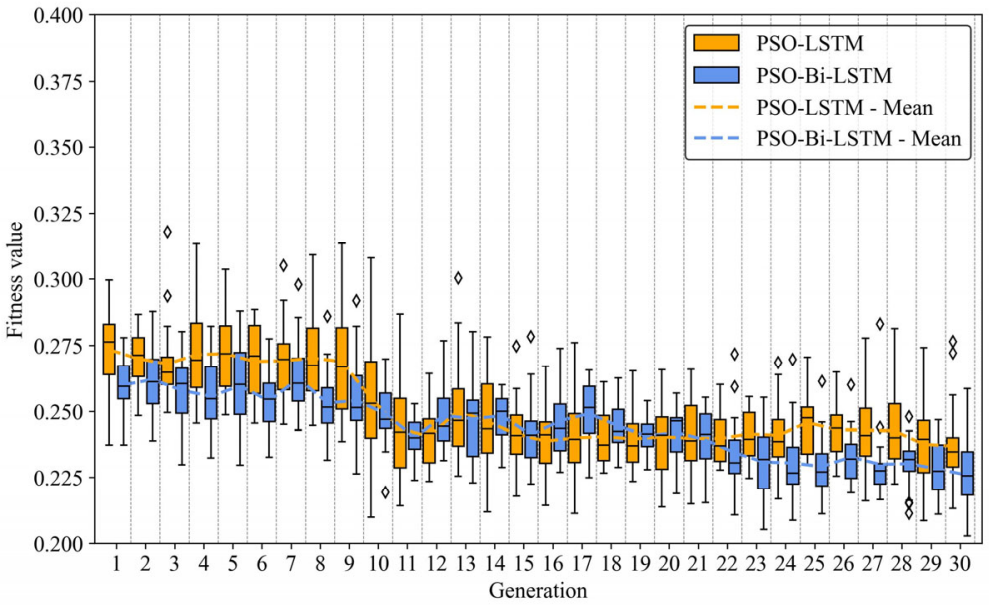


Figure 11. Performance comparison of PSO-LSTM and PSO-Bi-LSTM models.

When Figures 10 and 11 are evaluated together, it can be observed that models based on the GA approach exhibit minimal variation in fitness values across generations and tend to get trapped in local minima. In contrast, methods based on PSO show a decrease in fitness values over generations, leading to models with high predictive performance; however, the convergence speed toward the global optimum remains notably low. The fitness values for these methods, as shown in Table 10, are as follows: GA-LSTM (0.2324), GA-Bi-LSTM (0.2286), PSO-LSTM (0.2085), and PSO-Bi-LSTM (0.2028). For example, while the R^2 value increases, the RMSE value decreases. These results indicate that PSO-based models, particularly PSO-Bi-LSTM, achieve lower fitness values, suggesting better optimization performance compared to GA-based models.

Table 10. Comparison of fitness value for traditional benchmark models.

Models	Fitness Value	R^2	MAE	MSE	RMSE	MAPE
GA-LSTM	0.2324	0.8840	0.1509	0.0452	0.2127	4.7799
GA-Bi-LSTM	0.2286	0.9180	0.1353	0.0416	0.2038	4.1424
PSO-LSTM	0.2085	0.9248	0.1183	0.0312	0.1766	3.6826
PSO-Bi-LSTM	0.2028	0.9288	0.1191	0.0303	0.1741	3.7350

3.5. Performance of SFS and Its Variants Based Models

Figures 12–15 present the performance comparisons of the following models, respectively: SFS-LSTM vs. SFS-Bi-LSTM (Figure 12), FDB-SFS-LSTM vs. FDB-SFS-Bi-LSTM (Figure 13), dFDB-SFS-LSTM vs. dFDB-SFS-Bi-LSTM (Figure 14), and AFDB-SFS-LSTM vs. AFDB-SFS-Bi-LSTM (Figure 15). When Figure 12 is examined, the fitness value is high at the beginning, but this value tends to decrease as the generations progress. The average fitness value also decreases as the generations progress, but, in general, it can be said that the SFS-Bi-LSTM model performs better than SFS-LSTM model. When Figure 13 is examined, it is seen that the fitness values obtained according to the results decrease as the generation increases. In particular, the lowest fitness value is obtained with FDB-SFS-Bi-LSTM model. The results obtained with FDB-SFS-Bi-LSTM model are better than SFS-LSTM model. However, it is seen that it lags behind FDB-SFS-Bi-LSTM model. When the performances of the dFDB-SFS-LSTM and dFDB-SFS-Bi-LSTM models are examined throughout the generation in Figure 14, it is seen that the variance of the solutions obtained by the dFDB-SFS-LSTM model is higher than FDB-SFS-LSTM and it lags behind in terms of performance. However, the variance of the dFDB-SFS-Bi-LSTM model is lower than the FDB-SFS-LSTM model and it can be said that high performance is obtained. When Figure 15 is examined, the performance of the AFDB-SFS-LSTM and AFDBSFS-Bi-LSTM models is plotted across generations. Although the AFDB approach does not yield very strong results among the LSTM models, the AFDB-SFS-Bi-LSTM approach achieves the best solution set among the other models, with the most optimal convergence occurring in the early generations.

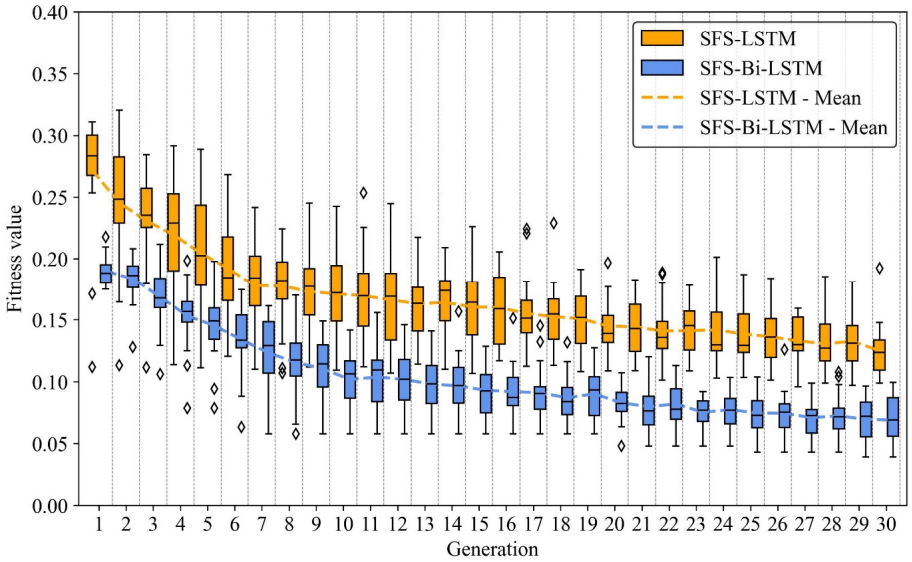


Figure 12. Performance comparison of SFS-LSTM and SFS-Bi-LSTM models.

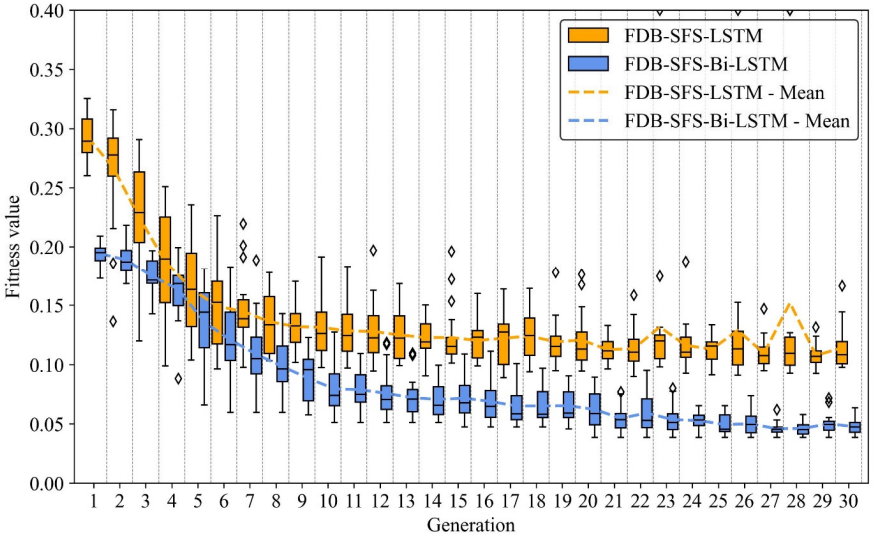


Figure 13. Performance comparison of FDB-SFS-LSTM and FDB-SFS-Bi-LSTM models.

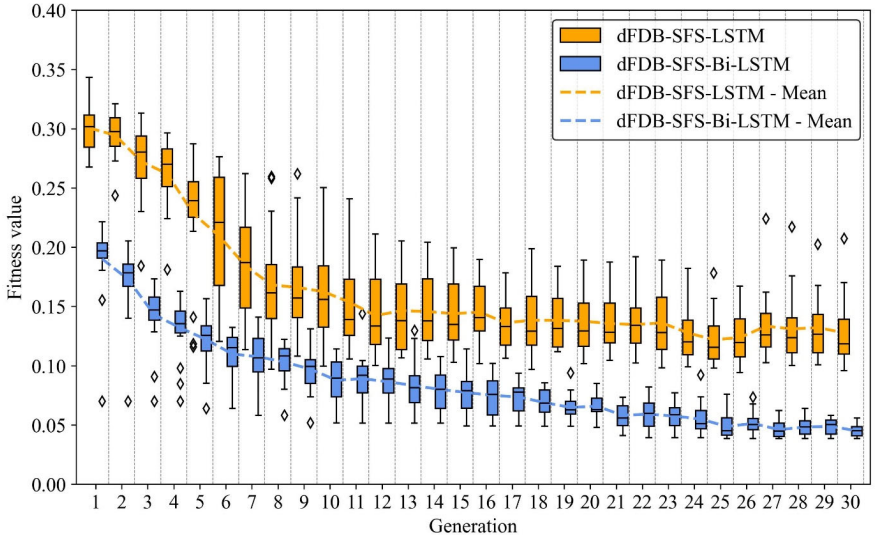


Figure 14. Performance comparison of dFDB-SFS-LSTM and dFDB-SFS-Bi-LSTM models.

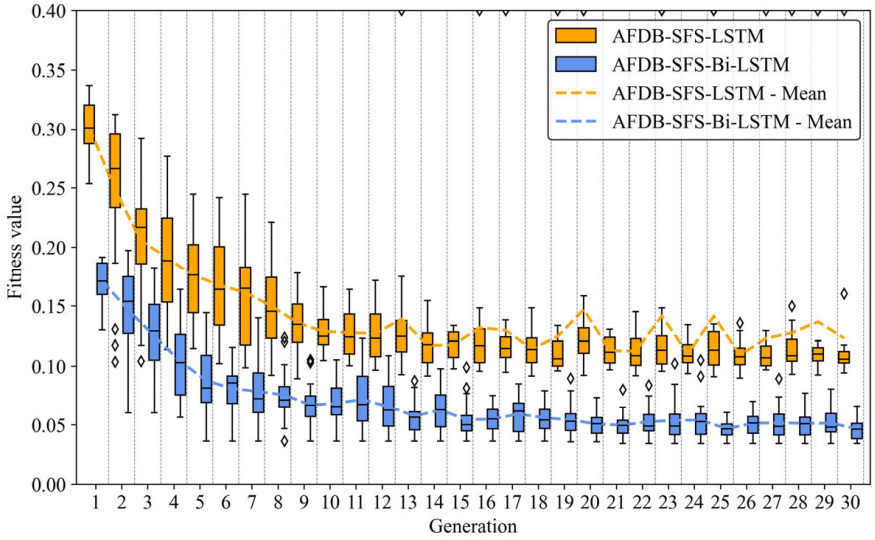


Figure 15. Performance comparison of ADFB-SFS-LSTM and ADFB-SFS-Bi-LSTM models.

The best results obtained for model performances throughout the generation are shown in Figure 16. When all the results are evaluated together, it is seen that there is a clear performance difference between the LSTM and Bi-LSTM models. The Bi-LSTM model is more successful than the LSTM model in finding the most suitable model for HO time series prediction. Comparison of the fitness value and member ranking in the population for different models is given in Table 11.

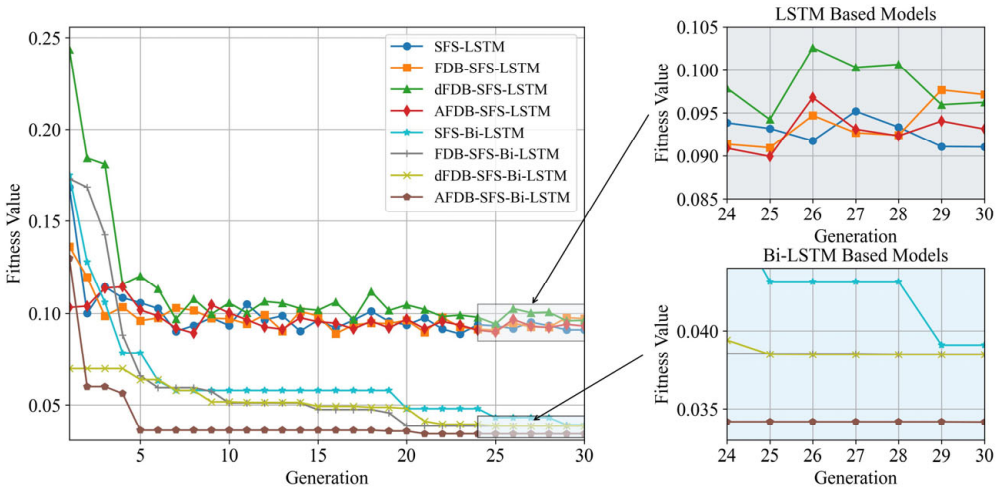


Figure 16. Performance comparison of all SFS models and its variants with LSTM and Bi-LSTM.

Table 11. Comparison of fitness value and member rank in population for all models.

Models	Fitness Value	Member Rank in Population
AFDB-SFS-Bi-LSTM	0.03416	418th
dFDB-SFS-Bi-LSTM	0.03848	505th
FDB-SFS-Bi-LSTM	0.03850	552nd
SFS-Bi-LSTM	0.03910	572nd
SFS-LSTM	0.08885	450th
FDB-SFS-LSTM	0.08898	307th
AFDB-SFS-LSTM	0.08922	142nd
dFDB-SFS-LSTM	0.09424	482nd

Considering the values shown in Figure 16 and presented in Table 11, in the comparison between LSTM models based on fitness error values, it should be noted that the optimum results found in the population member in the early generations are not a random factor when looking at the performances of the prominent models. It can be said that the strategies used (such as SFS, FDB-SFS, dFDB-SFS, AFDB-SFS) have the ability to find faster solutions for optimal solutions since they are found in low population member numbers in early generations. These strategies act more efficiently in the search and optimization processes and provide access to appropriate results in a shorter time. The fitness error value of the SFS-LSTM model is 0.08885, and the optimal population member number is 450 out of a total of 600 solutions, consisting of 30 generations with 20 populations in each generation. This model quickly reached the optimal solution using the basic SFS approach. However, the fitness error value of the FDB-SFS-LSTM model, which exhibits a very close performance to this model, is 0.08898 and the optimal population member number is 307. Although FDB-SFSLSTM showed a lower performance than SFS-LSTM with a very small difference in fitness error value, it achieved similar results with fewer population mem-

bers thanks to the FDB strategy. This shows the efficiency of the FDB-SFS-LSTM model. Although the AFDB-SFS-LSTM model fell slightly behind in terms of performance with a fitness error value of 0.08922, it draws attention by reaching the optimal solution with only the 142nd member number. Thanks to the AFDB strategy, the model is able to reach a high performance with a much lower number of population members. This strategy shows that it accelerates the solution process by narrowing the search area more quickly and effectively thanks to its adaptive features. Finally, the dFDB-SFS-LSTM model is the model with the highest error rate with a fitness error value of 0.09424 and an optimal member number of 482. Although it worked with the dFDB strategy, it required a higher number of population members compared to the other models. Although this model accelerates the solution search process thanks to the effective use of dynamic features, it remains lower than other LSTM models in terms of performance.

When Bi-LSTM models are examined, the effects of the advantages provided by the strategies become clearer. The AFDB-SFS-Bi-LSTM model provided the best performance with a fitness error value of 0.03416 with the optimal solution found at individual 418th. The success of AFDB-SFS-Bi-LSTM is related to the fact that the AFDB strategy accelerates the search process thanks to its adaptive and dynamic structure and reaches the optimum solution quickly. In other words, the reason why the optimal solution is found in earlier generations is that the strategy is effective and fast. The dFDB-SFS-Bi-LSTM model performs strongly with a fitness error value of 0.03848, and the best population member number is 505. The dFDB strategy accelerated the search process, reaching the optimum solution relatively quickly. However, it appears to have lagged behind the AFDB-SFS-Bi-LSTM in terms of both fitness error and the member order where the best population member is located. The FDB-SFS-Bi-LSTM model, which ranks third, offers similar performance with a fitness error value of 0.03850 and a population member number of 552. Although the FDB strategy of this model tries to optimize the solutions with fewer population members, it reaches the optimal result with slightly more population members than the other Bi-LSTM models. Finally, the SFS-Bi-LSTM model is the model with the highest error rate with a fitness error value of 0.03910 and a population member number of 572. The reason this model requires a higher population member number is that the SFS strategy is simpler and less effective than the other strategies. Therefore, SFS-Bi-LSTM requires more population members compared to other advanced strategies (FDB, dFDB, AFDB). When comparing the performances of these models, the low population member number is actually related to the efficiency of the strategies. Strategies such as AFDB, dFDB, and FDB narrow down the search space faster and reach optimum solutions in a shorter time. Especially AFDB-SFS-Bi-LSTM models achieved high performance even in very early generation thanks to their adaptive strategies included chaotic map scoring function. This shows that these models use advanced strategies that accelerate the solution process. As a result, strategy selection is a critical factor in maintaining or increasing performance by reducing the population size. While simpler strategies such as SFS require more population members, strategies such as FDB, dFDB, and especially AFDB increase the performance of the models by providing faster and more efficient solutions.

3.5.1. Selected Features Importance Analysis

In Figure 17, the total selection numbers of CCI, RSI, Fast %K, Fast %D, WRO, MO, MFI, UO, DCC indicators throughout the generation are given for the models created. Three radar charts are given for each indicator; the chart in Figure 17a illustrates the number of features selected across all generations, the chart in Figure 17b shows the number of features selected in the final generation, and the chart in Figure 17c depicts the number of features selected for the best solution.

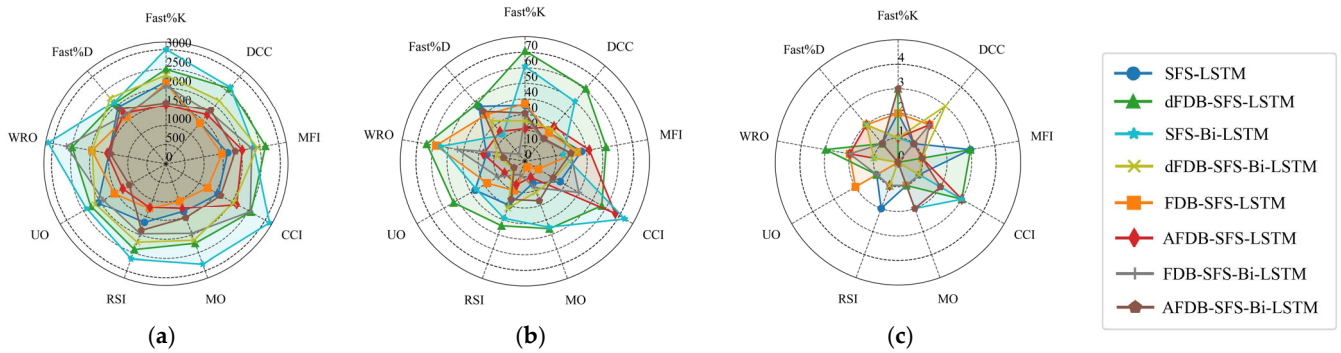


Figure 17. Radar chart displaying the number of selected features (**a**) in all generations, (**b**) in last generation, (**c**) in best generation for HO prediction models.

In Figure 18, the evolution of selected features across generations in the proposed hybrid HO time series prediction model is depicted, with feature indices on the x-axis (grouped into ranges such as 1–21, 22–42,. . ., 169–189) and generation numbers (1 to 600) on the y-axis. Feature types are color-coded (e.g., Fast%K, Fast%D, WRO, UO, RSI, MO, CCI, MFI, DCC) solely to distinguish between them, while white spaces within each range represent the selected features at each generation. It is observed that a wide variety of features are selected at the initial stages, while the number of selected features decreases as the generations progress. This reduction suggests an optimization process where the models converge toward a more focused set of relevant features over time.

In Figure 19, it is presented a graphical representation of the relationships between different models and feature types using a split chord diagram. This type of diagram is structured with two semi-circles: the left semi-circle represents the models (e.g., SFS-LSTM, AFDB-SFS-LSTM, etc.), and the right semi-circle represents the feature types (e.g., CCI, RSI, Fast%K, etc.). The chords (lines) connecting the two semi-circles illustrate the correlations or dependencies between each model and the features it utilizes, with the thickness or presence of a chord typically indicating the strength of the relationship.

In this specific diagram, the models include variations such as SFS-LSTM, FDB-SFS-LSTM, dFDB-SFS-LSTM, AFDB-SFS-LSTM, and their Bi-LSTM counterparts, while the features encompass technical indicators like CCI, RSI, Fast%K, Fast%D, WRO, MO, MFI, UO, and DCC. The diagram visually captures how each model relies on specific features for its predictive tasks, likely in a financial time-series context, such as stock price prediction.

In Figure 20, the number of features selected for the best solution is shown with a bar graph for all models. It is also given mean number of selected feature values for each indicator. SFS-LSTM excels with MFI (3 features) for volume-price analysis and uses RSI and Fast%K (both 2) for momentum, showing a strong focus on market strength. It also leverages Fast%D, MO, UO, and DCC (all 1) for balanced dynamics. However, it lacks WRO (0 feature), missing overbought/oversold insights. SFS-Bi-LSTM shines with CCI and Fast%K (both 3) for trend and rapid price detection, and uses Fast%D and MO (both 2) for momentum. However, it skips RSI, WRO, MFI, and UO (all 0), limiting its scope on relative strength and multi-timeframe analysis.

FDB-SFS-LSTM leverages Fast%K and Fast%D (both 2), and WRO and UO (both 2) for overbought/oversold and multi-timeframe insights. It also uses MO, MFI, and DCC (all 1) moderately. However, it skips CCI and RSI (both 0), missing trend and relative strength data. FDB-SFS-Bi-LSTM uses WRO (2) for overbought/oversold analysis and balances CCI, RSI, Fast%D, MO, MFI, and DCC (all 1) for trend and momentum. Weaknesses include no Fast%K or UO (both 0), limiting rapid price and multi-timeframe focus.

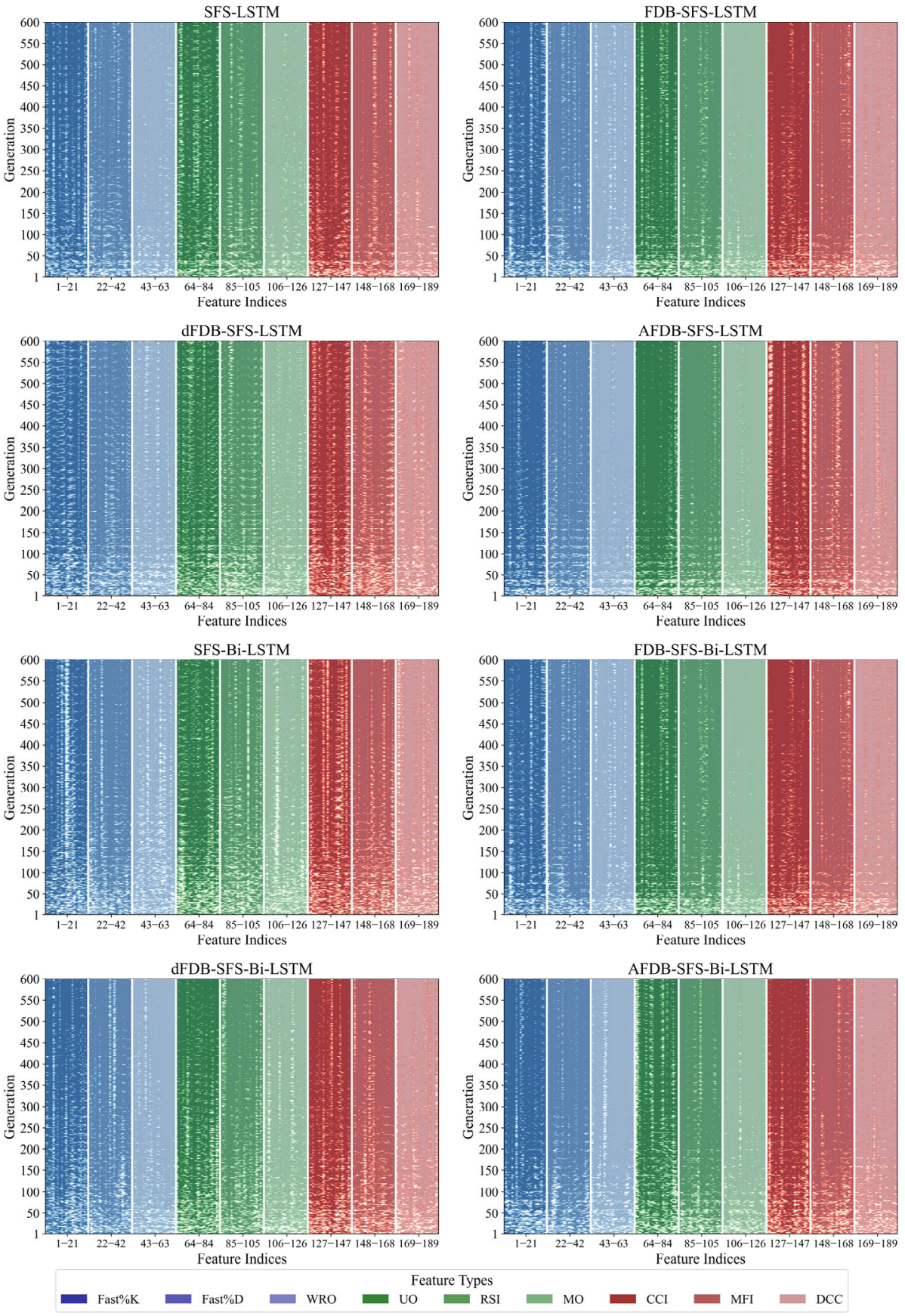


Figure 18. Changing of features over generations as heatmap in all models.

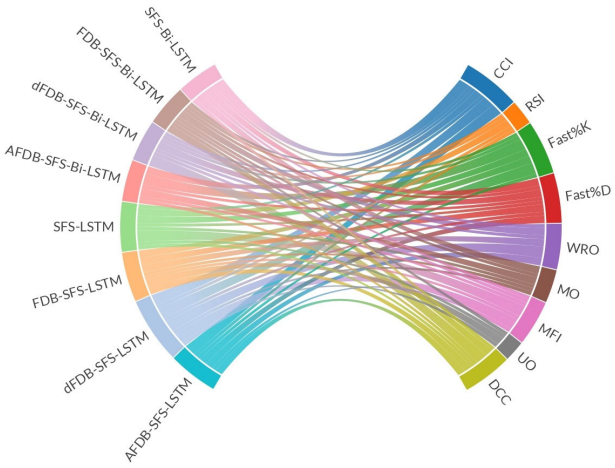


Figure 19. Visualization of model-feature correlations via split chord diagram.

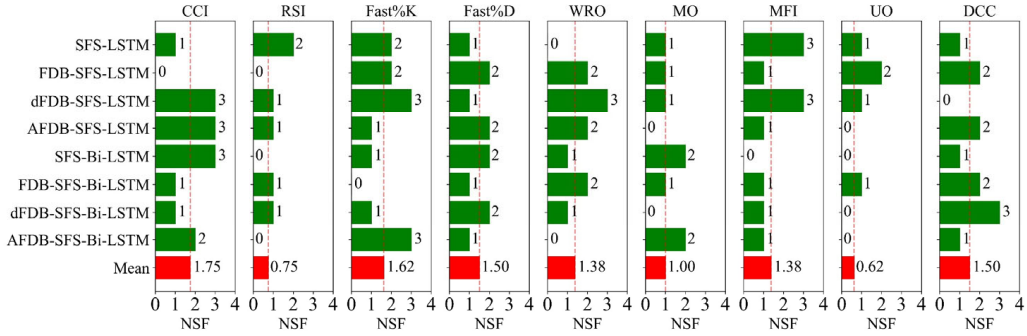


Figure 20. Distribution of selected number of features for all models.

dFDB-SFS-LSTM shines with CCI, Fast%K, and MFI (all 3) for trend, price speed, and volume-price insights, and uses WRO (3) for overbought/oversold conditions. It also employs Fast%D and RSI (both 1) moderately. Weaknesses include no MO, UO, or DCC (all 0), missing momentum and cyclic data. dFDB-SFS-Bi-LSTM uses DCC (3) heavily for cyclic patterns, with CCI and MFI (both 1) for trend and volume. It also leverages Fast%K, RSI, Fast%D, and WRO (all 1) for balance. However, it skips MO and UO (both 0), limiting momentum depth.

AFDB-SFS-LSTM demonstrates significant strengths, particularly with its highest connection to CCI (3 features), indicating a strong reliance on this indicator for trend analysis and identifying overbought/oversold conditions. Additionally, the model effectively utilizes Fast%D, WRO, and DCC (all with 2 features), enabling a balanced approach to analyzing momentum and price movements. However, weaknesses are evident as MO and UO (both with 0 feature) are not utilized, suggesting that the model does not focus on certain momentum indicators (e.g., MO) or multi-timeframe analysis (e.g., UO). AFDB-SFS-Bi-LSTM excels with Fast%K (3 features), emphasizing rapid price movement detection, and balances this with CCI (2 features) for trend analysis and MO (2 features) for momentum. It incorporates DCC (1 feature), showing a slight focus on cyclic patterns that may help identify periodic market trends in financial data. However, it overlooks RSI, WRO, and UO (all scoring 0), limiting its ability to capture relative strength, overbought/oversold conditions, or multi-timeframe momentum.

3.5.2. Comparison of Model Performances

The plot of the actual and predicted HO time series is given as in Figure 21 for all models.

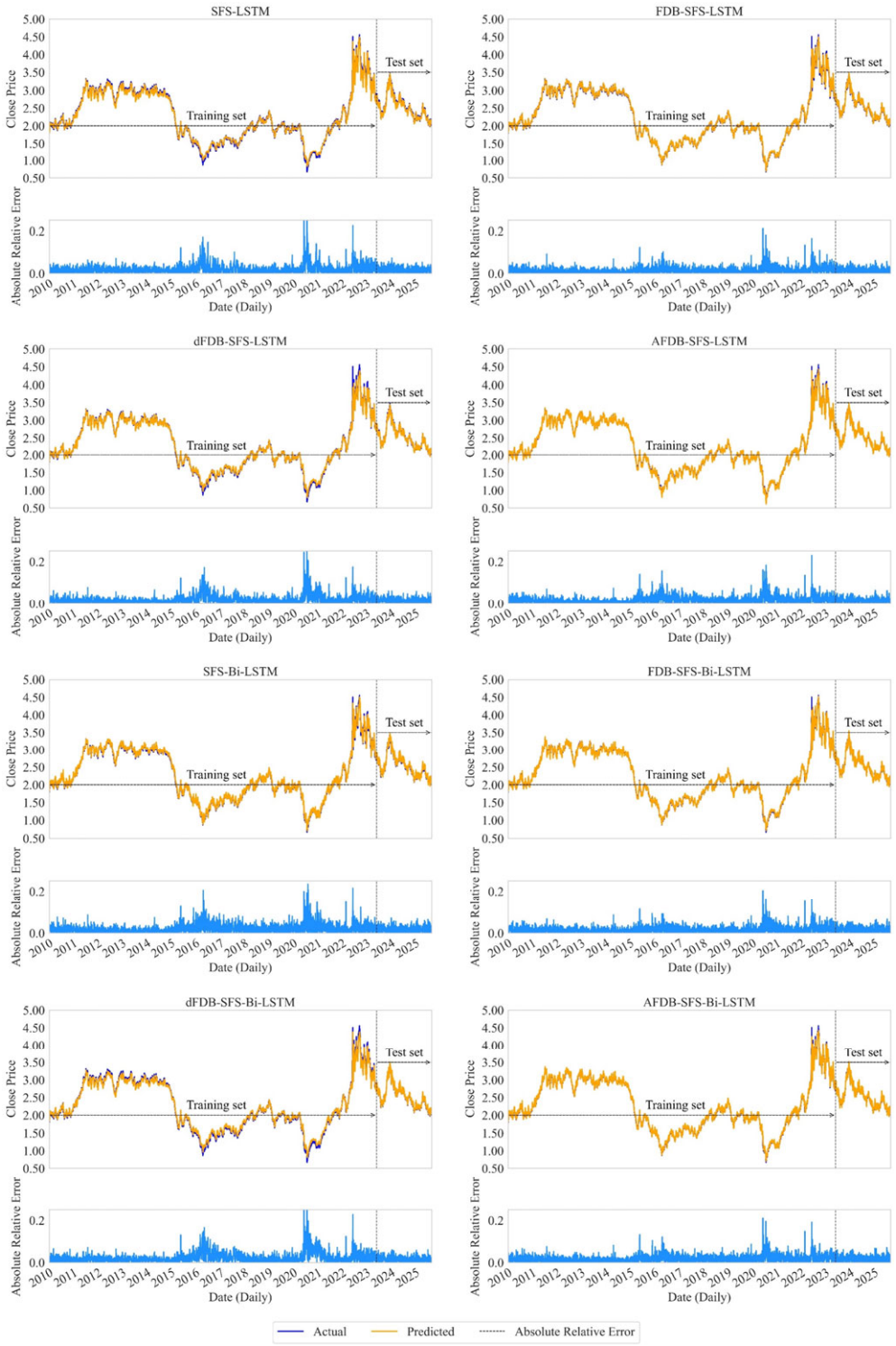


Figure 21. The predicted results for all models in the training and test stages for HO time series prediction.

It is seen that the prediction performance is high for the test data between 27 February 2023 and 30 May 2025. The plots of the actual and predicted results for the LSTM and Bi-LSTM based models are given in Figures 22 and 23, respectively. In these graphs, the last two months are also enlarged. Here, a visual examination is provided and the prediction results produced by the models in response to the test data are given.

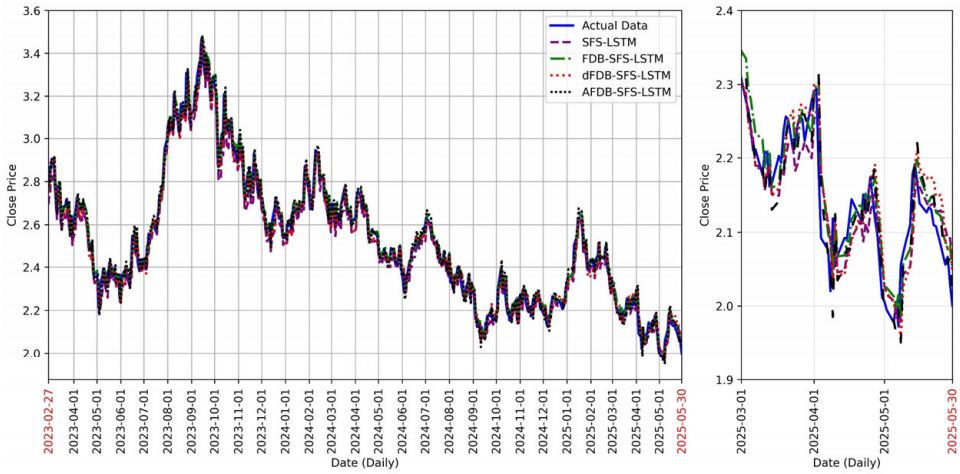


Figure 22. Real vs. prediction results for HO time series prediction model with LSTM based models.

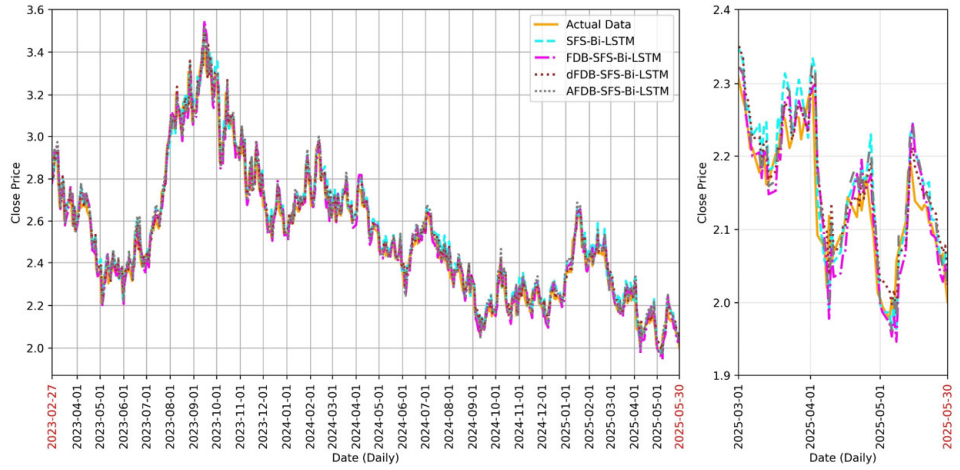


Figure 23. Real vs. prediction results for HO time series prediction model with Bi-LSTM based models.

Linear regression analysis evaluates the relationship between actual and predicted values in prediction models. The resulting coefficients, which indicate the strength and direction of this relationship, along with the R^2 value, which measures the model's explanatory power, are summarized in Table 12. The R^2 value reflects how well the model captures data variability, with higher values indicating a better fit. Additionally, the regression plot, illustrating the alignment of predicted and actual values, is provided in Figure 24. Together, Table 12 and Figure 24 offer a clear overview of the models' prediction performance and reliability.

Table 12. Linear regression coefficients of all models.

Methods	a	b	R^2
SFS-LSTM	0.9549	0.0895	0.9691
FDB-SFS-LSTM	0.9912	0.0332	0.9679
dFDB-SFS-LSTM	0.9616	0.0895	0.9699
AFDB-SFS-LSTM	0.9980	0.0102	0.9691
SFS-Bi-LSTM	0.9943	0.0384	0.9933
FDB-SFS-Bi-LSTM	0.9977	0.0109	0.9936
dFDB-SFS-Bi-LSTM	0.9883	0.0442	0.9936
AFDB-SFS-Bi-LSTM	0.9922	0.0370	0.9959

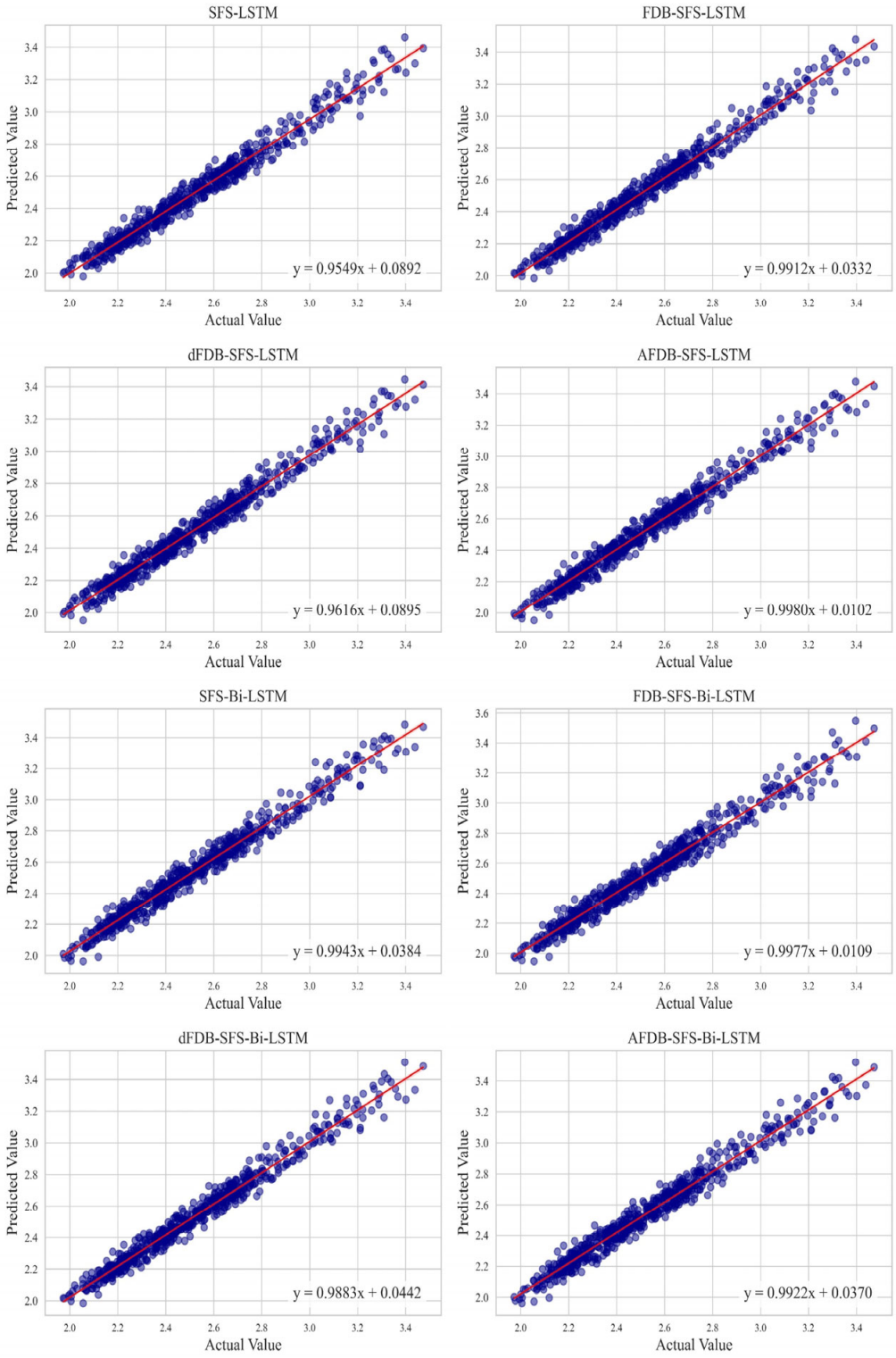


Figure 24. The regression plot of error for all models.

Statistical error values of all models are given in Table 13. This comprehensive table allows the evaluation of the strengths and weaknesses of the models together. The combination of these metrics provides a holistic view of the model performance. Since there is no general consensus in the literature, different statistical metrics can be used. By calculating values for all models with five different statistical metrics, it is possible to compare the created models with the models in the literature more easily.

Table 13. Statistical error values of all models.

Methods	Fitness Value	R^2	MAE	MSE	RMSE	MAPE
SFS-LSTM	0.0910	0.9691	0.1017	0.0223	0.1492	3.1970
FDB-SFS-LSTM	0.0971	0.9679	0.1044	0.0239	0.1545	3.2660
dFDB-SFS-LSTM	0.0962	0.9699	0.0986	0.0211	0.1453	3.1050
AFDB-SFS-LSTM	0.0931	0.9691	0.1076	0.0258	0.1608	3.3307
SFS-Bi-LSTM	0.0391	0.9933	0.0278	0.0021	0.0461	0.9304
FDB-SFS-Bi-LSTM	0.0385	0.9936	0.0271	0.0020	0.0449	0.9047
dFDB-SFS-Bi-LSTM	0.0385	0.9936	0.0267	0.0020	0.0449	0.8920
AFDB-SFS-Bi-LSTM	0.0342	0.9959	0.0173	0.0013	0.0364	0.5769

Upon examining Table 13, it is observed that LSTM-based models exhibit higher MAE, MSE, RMSE, and MAPE values compared to Bi-LSTM-based models, while their R^2 values remain lower. The fitness value obtained from the optimization process is naturally higher in LSTM-based models compared to Bi-LSTM-based models due to the structure of the fitness function. Notably, the AFDB-SFS-Bi-LSTM approach demonstrates superior performance compared to all other models. The strong performance of the AFDB-SFS-Bi-LSTM method can be attributed to its chaotic fractal structure, which enhances its ability to capture complex patterns and dynamics in the data.

3.5.3. Results of the Statistical Significance Test

The Wilcoxon rank sum test results, comparing the models based solely on their obtained values relative to the actual values, are presented in Table 14, including the z-value, *p*-value, and H-value for each pairwise comparison. The z-value indicates the standardized difference between the rank sums of the two groups, the *p*-value determines the statistical significance of this difference, and the H-value, which equals 1, confirms the rejection of the null hypothesis, indicating that the hypothesis of differing medians has been validated. Specifically, a low *p*-value (typically < 0.05) and an H-value of 1 suggest that the median prediction errors of one model differ significantly from another. The superior performance of the AFDB-SFS-Bi-LSTM model, attributed to its chaotic fractal structure, is statistically validated by these test results, reinforcing its ability to capture complex data patterns effectively.

Table 14. Wilcoxon rank sum test results of all models.

Methods	H	z-Value	*p*
SFS-LSTM	1	−10.9153	9.74×10^{-28}
FDB-SFS-LSTM	1	−5.4605	4.75×10^{-8}
dFDB-SFS-LSTM	1	−3.3860	7.09×10^{-4}
AFDB-SFS-LSTM	1	−2.5007	1.24×10^{-2}
SFS-Bi-LSTM	1	−10.9800	4.77×10^{-28}
FDB-SFS-Bi-LSTM	1	−2.1209	3.39×10^{-2}
dFDB-SFS-Bi-LSTM	1	−7.4063	1.82×10^{-13}
AFDB-SFS-Bi-LSTM	1	−7.5904	3.19×10^{-14}

3.5.4. Ablation Study

In the AFDB-SFS-Bi-LSTM model, an ablation study is conducted to verify the importance of the 10 features selected by AFDB-SFS by retraining the model by removing each feature individually. Table 15 shows the percentage changes in performance metrics (ΔR^2%, ΔMAE%, ΔMSE%, ΔRMSE%, ΔMAPE%) due to feature removal. Negative ΔR^2% values indicate a decrease in explanatory power, while positive ΔMAE%, ΔMSE%, ΔRMSE%, and ΔMAPE% values reflect increased prediction errors, emphasizing the importance of the feature. For example, removing DCC (6) decreased R^2 by 0.23% and increased RMSE by 1.39%, highlighting its critical role. In contrast, removing relatively less influential features, such as MFI (9), had a smaller effect (ΔR^2% = −0.10%, ΔRMSE% = +2.52%). Overall, the chaotic AFDB-SFS method was found to identify performance-enhancing features and yield a less complex model. This analysis supports the robustness and validity of the selected feature subset. To enhance interpretability, it is investigated the financial meaning and functional role of the key features selected by the proposed AFDB-SFS method. Notably, in addition to individual technical indicators, AFDB-SFS also selects composite indicators such as DCC (6), which is derived from the dynamic combination of multiple financial signals. In contrast to single indicators (e.g., CCI, MO, MFI), DCC (6) captures the time-varying dependence structure and co-movements among these signals simultaneously. This enables a richer and more holistic representation of market conditions by modeling the joint evolution of indicators rather than treating them independently. Sensitivity analysis reinforces its critical role: excluding DCC (6) leads to a substantial increase in prediction error, confirming that inter-indicator dynamics significantly contribute to the model's predictive power. Thus, incorporating such composite features not only boosts prediction accuracy but also improves interpretability by revealing that both individual signals and their dynamic interrelationships are essential for robust financial prediction.

Table 15. Ablation study results.

Selected Features	ΔR^2%	ΔMAE%	ΔMSE%	ΔRMSE%	ΔMAPE%
CCI (5)	−0.24%	3.03%	5.26%	2.86%	2.72%
CCI (15)	−0.35%	2.56%	7.52%	3.64%	2.55%
Fast%K (6)	−0.28%	2.10%	4.51%	2.34%	2.17%
Fast%K (10)	−0.18%	1.16%	1.50%	0.78%	1.01%
Fast%K (25)	−0.32%	2.33%	3.76%	2.08%	2.42%
Fast%D (5)	−0.50%	1.51%	6.77%	3.21%	1.45%
MO (16)	−0.44%	0.93%	2.26%	1.30%	1.08%
MO (23)	−0.35%	5.36%	9.77%	4.77%	5.35%
MFI (9)	−0.10%	2.33%	5.26%	2.52%	2.08%
DCC (6)	−0.23%	1.51%	3.01%	1.39%	1.64%

3.6. Comparative Performance Results with Literature

In the study, the proposed hybrid model is comprehensively compared with existing HO prediction models reported in the literature. The results in Table 16 provide a detailed comparison, showcasing the proposed model's performance in terms of accuracy, predictive capability, and robustness against the baseline models. Specifically, the R^2 values indicate the model's explanatory power, while MAE, MSE, RMSE, and MAPE metrics highlight its error characteristics and predictive precision.

Table 16. Comparison of the proposed hybrid model with existing HO prediction models in the literature.

Studies	Year	Methods	Time Period	R^2	MAE	MSE	RMSE	MAPE
[13]	2005	NN	From December 1997 to November 2002	0.8600	0.1000	-	-	-
[14]	2013	ARFIMA-FIGARCH	From 3 January 1995 to 31 July 2012	-	1.156	1.337	-	-
		ARFIMA-IGARCH		-	1.158	1.342	-	-
		ARFIMA-GARCH		-	1.157	1.339	-	-
		ARFIMA-GARCH		-	1.197	1.433	-	-
[15]	2014	BPNN	From December 1997 to November 2002	0.8433	-	-	-	0.156724
		GA-LSSVM		0.9236	-	-	-	0.076421
		ABC-LSSVM		0.9282	-	-	-	0.071769
		LSSVM with eABC		0.9367	-	-	-	0.063310
[16]	2014	LMNN	From 9 July 1992 to 16 October 2012	0.9938	-	0.000115	-	-
		SCGNN		0.9808	-	0.005040	-	-
		GDANN		0.7940	-	0.793000	-	-
		BNN		−0.6456	-	5.570000	-	-
		BRNN		0.9963	-	0.005730	-	-
[17]	2018	CNM	From 1 June 2001 to 18 July 2017	-	-	-	0.3560	0.2141
[18]	2021	SVM	From 1 January 2009 to 23 October 2019	-	0.1056	-	0.2156	7.2594
		BPNN		-	0.0844	-	0.1980	5.2090
		LSTM		-	0.0714	-	0.1529	3.1326
		WPD-BPNN		-	0.0505	-	0.1271	2.0593
		WPD-LSTM		-	0.0463	-	0.0945	1.8461
		CEEMDAN-LSTM		-	0.0376	-	0.0805	1.2376
		VMD-LSTM		-	0.0497	-	0.1014	1.8509
		ST-GRU		-	0.0408	-	0.0732	1.0338
		WPD-SW-LSTM		-	0.0212	-	0.0642	0.4775
[19]	2024	NN-SCG	From June 1986 to September 2021	0.9906	-	-	-	-
[20]	2024	SVR-BO	From 1993 to 2023	-	-	-	0.0274	-
		kNN-BO		-	-	-	0.0576	-
		RT-BO		-	-	-	0.0290	-
		GRP-BO		-	-	-	0.0283	-
		DFNN-BO		-	-	-	0.0576	-
[21]	2024	XGBoost	From 5 January 2004 to 29 December 2023	-	0.792	-	1.207	1.321
		RF		-	1.206	-	1.127	0.007
Proposed Hybrid Model		AFDB-SFS-Bi-LSTM	From 1 May 2009 to 30 May 2025	0.9959	0.0173	0.0013	0.0364	0.5769

The AFDB-SFS-Bi-LSTM model is among the best performing models in the provided comparison, showing excellence in R^2, MAE, and MSE, and a competitive performance in RMSE. The AFDB-SFS-Bi-LSTM model demonstrates exceptional performance across multiple metrics with 0.9959 R^2, 0.0173 MAE, 0.0013 MSE, 0.0364 RMSE, and 0.5769 MAPE evaluated from 1 May 2009 to 30 May 2025. Its R^2 is one of the highest and competes closely with the BRNN model, which achieves a slightly better R^2 of 0.9963, showing excellent explanatory power for both models. However, the MSE of BRNN of 0.005730 is significantly higher than the MSE of AFDB-SFS-Bi-LSTM of 0.0013, indicating that although BRNN explains a marginally higher proportion of variance, it produces larger squared prediction errors compared to the proposed model. The MAE of AFDB-SFS-Bi-LSTM is the lowest among the models reporting this metric, and its MSE is only surpassed by LMNN (0.000115), highlighting its superior precision. Its RMSE is competitive, but slightly higher than SVR-BO (0.0274), RT-BO (0.0290), and GRP-BO (0.0283), but it outperforms most of the others. Although its MAPE is not the lowest, it is still better than many models, especially those in [18,21]. Statistical validation from the Wilcoxon rank sum test (z-value, *p*-value, H = 1) confirms that its superior performance is not due to chance, especially against LSTM-based models. The chaotic fractal structure enhances the processing capability, making it a solid choice for applications that require high prediction accuracy, such as time series prediction or complex pattern recognition. For scenarios that prioritize MAPE, models such as eABC or LSSVM with RF are considered to perform well, but they lack the comprehensive metric performance of AFDB-SFS-Bi-LSTM. Overall, the AFDB-SFS-Bi-LSTM model's exceptional

balance of high explanatory power and low error metrics establishes it as a highly effective solution for complex prediction tasks.

3.7. Real-Time Embedded System Test Results

The model obtained upon completion of the simulation has been deployed on the Jetson Orin Nano, as illustrated in Figure 25. In Figure 26, it is illustrated the average computation times (in microseconds) of various technical indicators implemented on the NVIDIA Jetson Nano Orin platform. The indicators included are DCC, MFI, CCI, MO, RSI, UO, WRO, Fast%D, and Fast%K, which are commonly used in this study. The chart illustrates the computation differences among the indicators, with DCC having the longest average computation time due to its combination of multiple signal processing techniques, and MO having the shortest. In Figure 27, the box and whisker plot provides a visual summary of the computation times (in microseconds) for the AFDB-SFS-LSTM and AFDB-SFS-Bi-LSTM models on the NVIDIA Jetson Orin Nano. AFDB-SFS-LSTM is more computationally efficient, while AFDB-SFS-Bi-LSTM, despite its higher average computation time and variability, provides better prediction performance. On the Jetson Orin Nano (15 W mode), AFDB-SFS-LSTM achieves 470 us average latency and AFDB-SFS-Bi-LSTM 1010 µs. The lighter model is faster, while the more accurate bidirectional variant is preferred when prediction performance takes priority.

Figure 25. Jetson Orin Nano developer platform [64].

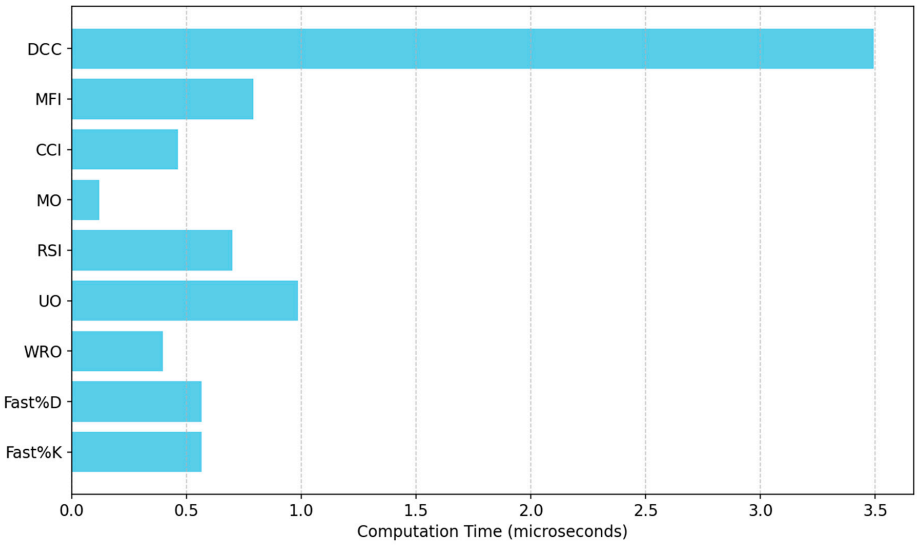


Figure 26. The average computation times of the signal processing techniques on the NVIDIA Jetson Orin.

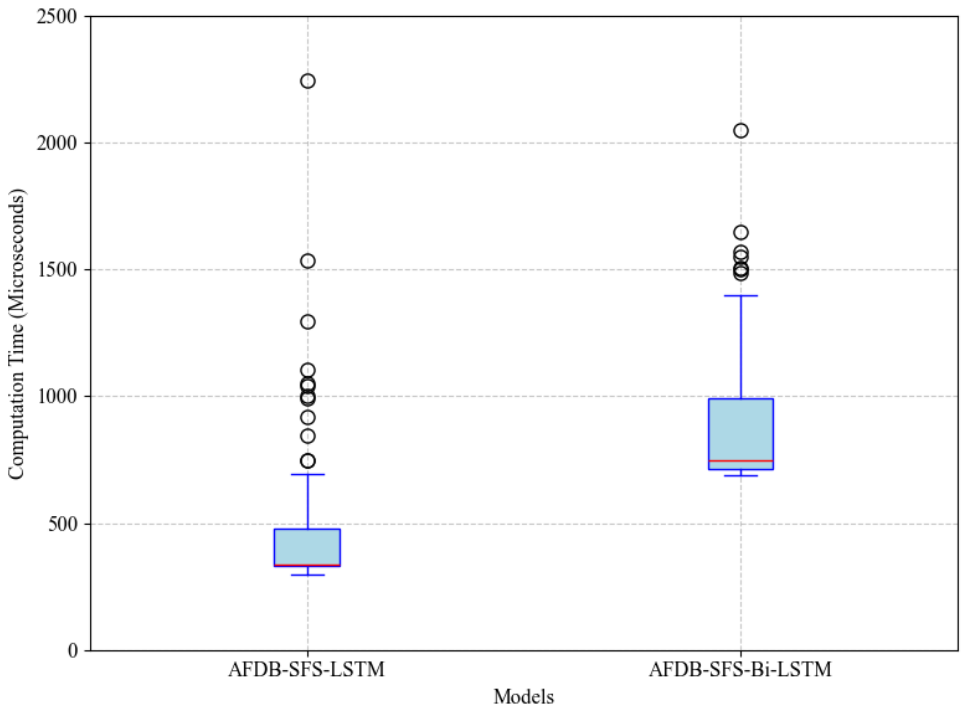


Figure 27. Box and whisker plot showing the average computation times for AFDB-SFS-LSTM and AFDB-SFS-Bi-LSTM based models on the NVIDIA Jetson Orin.

Table 17 presents a comparison of the AFDB-SFS-Bi-LSTM and AFDB-SFS-LSTM models with the PatchTST, Informer, and Autoformer models, all trained and tested using the features selected by the AFDB-SFS-Bi-LSTM method. The comparison is performed in terms of error metrics (R^2, RMSE, MAPE), temperature, power consumption, model size, and latency (inference time). The comparative results reveal a striking dominance of the proposed AFDB-SFS-based architectures over well-known state-of-the-art time-series prediction models on the evaluated dataset. The AFDB-SFS-Bi-LSTM model achieves outstanding predictive performance with an R^2 of 0.9959 (extremely close to perfect correlation), an RMSE of only 0.0364, and a MAPE of just 0.5769%—metrics that are 3–6 times better than the strongest baseline (Autoformer: R^2 = 0.9436, RMSE = 0.1341, MAPE = 3.7535%) and dramatically superior to Informer (R^2 = 0.9309) and especially PatchTST (R^2 = 0.8362, RMSE = 0.2285, MAPE = 4.8305%). This indicates that the bidirectional processing and the AFDB-SFS mechanism capture temporal dependencies with exceptional precision. Meanwhile, the unidirectional AFDB-SFS-LSTM, while slightly less accurate (R^2 = 0.9691, RMSE = 0.1608, MAPE = 3.3307%), still comfortably outperforms all Transformer-based baselines and offers the fastest inference time by a large margin (0.47 ms vs. the next fastest Informer at 2.60 ms), making it ideal for real-time and latency-critical applications. In terms of resource efficiency, classic models like Informer (56 KB), PatchTST (89 KB), and Autoformer (110 KB) are remarkably lightweight and consume the least power (6099–7500 mW), but their significantly lower forecasting accuracy limits their practical usefulness on this task. The AFDB-SFS models, despite being larger (1–2 MB) and consuming 7800–9100 mW with moderate temperature increases (59.6–61.5 °C), provide a far superior accuracy–efficiency trade-off. Consequently, AFDB-SFS-Bi-LSTM is the clear choice when maximum prediction accuracy is the primary objective, making both proposed models highly competitive for practical deployment compared to existing state-of-the-art architectures.

Table 17. Edge AI Benchmarking: Performance and Hardware Efficiency Comparison of Time Series Prediction Models.

Methods	R^2	RMSE	MAPE	Latency (ms)	Size (MB)	Power (mW)	Temperature (°C)
PatchTST	0.8362	0.2285	4.8305	4.90	0.089	6099	59.20
Informer	0.9309	0.1484	4.2926	2.60	0.056	7500	60.10
Autoformer	0.9436	0.1341	3.7535	5.90	0.110	7450	61.00
AFDB-SFS-LSTM	0.9691	0.1608	3.3307	0.47	1.006	7800	59.60
AFDB-SFS-Bi-LSTM	0.9959	0.0364	0.5769	1.01	2.011	9100	61.50

4. Conclusions and Future Research

The proposed hybrid approach, which concurrently combines financial signal processing, the chaotic fractal structure of the AFDB-SFS algorithm, and the predictive power of Bi-LSTM networks, successfully delivers exceptional accuracy in HO price prediction. The implementation of a multi-objective fitness function—maximizing R^2, minimizing RMSE, and reducing feature count—resulted in a highly robust and efficient model, achieving an R^2 of 0.9959 and the lowest MAE (0.0173). The model maintains low error rates and complexity, which is formally validated by the Wilcoxon rank sum test (H = 1). Consequently, this framework presents a robust solution for informed decision-making in the energy sector, promising transformative applications across financial and commodity markets. The integration of the DCC indicator proved particularly valuable. Crucially, the model's highly optimized structure allows for its effective application in high-frequency trading operations, as demonstrated by its ultra-low response time when deployed on the Jetson Orin Nano Developer Platform. Despite these significant strengths, a key limitation is the increased computational overhead during the offline AFDB-SFS feature selection phase, which may restrict applicability in extremely resource-constrained environments. Future research will focus on extending this framework by incorporating larger and more diverse datasets, including macroeconomic and geopolitical variables, exploring alternative deep learning algorithms, and implementing a real-time data integration pipeline for continuous market application.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Stern, D.I. The role of energy in economic growth. *Ann. N. Y. Acad. Sci.* **2011**, *1219*, 26–51. [CrossRef] [PubMed]
2. Woodcock, J.; Banister, D.; Edwards, P.; Prentice, A.M.; Roberts, I. Energy and transport. *Lancet* **2007**, *370*, 1078–1088. [CrossRef]
3. Yang, Z.; Shao, S.; Yang, L.; Liu, J. Differentiated effects of diversified technological sources on energy-saving technological progress: Empirical evidence from china's industrial sectors. *Renew. Sustain. Energy Rev.* **2017**, *72*, 1379–1388. [CrossRef]
4. Huebner, G.M.; Hamilton, I.; Chalabi, Z.; Shipworth, D.; Oreszczyn, T. Explaining domestic energy consumption–the comparative contribution of building factors, socio-demographics, behaviours and attitudes. *Appl. Energy* **2015**, *159*, 589–600. [CrossRef]
5. Helm, D. The future of fossil fuels—Is it the end? *Oxf. Rev. Econ. Policy* **2016**, *32*, 191–205. [CrossRef]
6. De Vries, B.J.; Van Vuuren, D.P.; Hoogwijk, M.M. Renewable energy sources: Their global potential for the first-half of the 21st century at a global level: An integrated approach. *Energy Policy* **2007**, *35*, 2590–2610. [CrossRef]
7. Davis, L.W. Prospects for nuclear power. *J. Econ. Perspect.* **2012**, *26*, 49–66. [CrossRef]
8. O'Rourke, F.; Boyle, F.; Reynolds, A. Tidal current energy resource assessment in ireland: Current status and future update. *Renew. Sustain. Energy Rev.* **2010**, *14*, 3206–3212. [CrossRef]
9. Tugcu, C.T.; Ozturk, I.; Aslan, A. Renewable and non-renewable energy consumption and economic growth relationship revisited: Evidence from g7 countries. *Energy Econ.* **2012**, *34*, 1942–1950. [CrossRef]
10. Gruenspecht, H. *International Energy Outlook 2011*; U.S. Energy Information Administration: Washington, DC, USA, 2010.

11. Atabani, A.E.; Silitonga, A.S.; Badruddin, I.A.; Mahlia, T.; Masjuki, H.; Mekhilef, S. A comprehensive review on biodiesel as an alternative energy resource and its characteristics. *Renew. Sustain. Energy Rev.* **2012**, *16*, 2070–2093. [CrossRef]
12. Mensi, W.; Brahim, M.; Hammoudeh, S.; Tiwari, A.K.; Kang, S.H. Time-varying causality and correlations between spot and futures prices of natural gas, crude oil, heating oil, and gasoline. *Resour. Policy* **2024**, *93*, 105077. [CrossRef]
13. Malliaris, M.E.; Malliaris, S.G. Forecasting energy product prices. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005.
14. Kang, S.H.; Yoon, S.M. Modeling and forecasting the volatility of petroleum futures prices. *Energy Econ.* **2013**, *36*, 354–362. [CrossRef]
15. Mustaffa, Z.; Yusof, Y.; Kamaruddin, S.S. An enhanced artificial bee colony optimizer for predictive analysis of heating oil prices using least squares support vector machines. In *Biologically-Inspired Techniques for Knowledge Discovery and Data Mining*; Information Science Reference: Hershey, PA, USA, 2014; pp. 149–173.
16. Chiroma, H.; Abdul-Kareem, S.; Abdullahi Muaz, S.; Khan, A.; Sari, E.N.; Herawan, T. Neural network intelligent learning algorithm for inter-related energy products applications. In Proceedings of the Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, 17–20 October 2014; Proceedings, Part I. pp. 284–293.
17. Tian, L.; Chen, H.; Zhen, Z. Research on the forward-looking behavior judgment of heating oil price evolution based on complex networks. *PLoS ONE* **2018**, *13*, e0202209. [CrossRef]
18. Wang, J.; Wang, J. A New Hybrid Forecasting Model Based on SW-LSTM and Wavelet Packet Decomposition: A Case Study of Oil Futures Prices. *Comput. Intell. Neurosci.* **2021**, *2021*, 7653091. [CrossRef] [PubMed]
19. Jin, B.; Xu, X. Price forecasting through neural networks for crude oil, heating oil, and natural gas. *Meas. Energy* **2024**, *1*, 100001. [CrossRef]
20. Lahmiri, S. Fossil energy market price prediction by using machine learning with optimal hyper-parameters: A comparative study. *Resour. Policy* **2024**, *92*, 105008. [CrossRef]
21. Kim, A.; Ryu, D.; Webb, A. Forecasting oil futures markets using machine learning and seasonal trend decomposition. *Investig. Anal. J.* **2025**, *54*, 205–218. [CrossRef]
22. Karasu, S.; Altan, A.; Bekiros, S.; Ahmad, W. A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. *Energy* **2020**, *212*, 118750. [CrossRef]
23. Karasu, S.; Altan, A. Crude oil time series prediction model based on LSTM network with chaotic Henry gas solubility optimization. *Energy* **2022**, *242*, 122964. [CrossRef]
24. Karasu, S.; Yalçın, N.A. Digital currency time series prediction based on financial signal processing, ant colony optimization and machine learning techniques. In Proceedings of the 2022 International Conference on Engineering Technologies (ICENTE), Konya, Turkey, 17–19 November 2022.
25. Karasu, S.; Altan, A.; Saraç, Z.; Hacioğlu, R. Prediction of Bitcoin prices with machine learning methods using time series data. In Proceedings of the 2018 26th signal processing and communications applications conference (SIU), Izmir, Turkey, 2–5 May 2018.
26. Altan, A.; Karasu, S. The effect of kernel values in support vector machine to forecasting performance of financial time series. *J. Cogn. Syst.* **2019**, *4*, 17–21.
27. Göçken, M.; Özçalıcı, M.; Boru, A.; Dosdoğru, A.T. Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Syst. Appl.* **2016**, *44*, 320–331. [CrossRef]
28. Shahvaroughi Farahani, M.; Razavi Hajiagha, S.H. Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models. *Soft Comput.* **2021**, *25*, 8483–8513. [CrossRef]
29. Emirmahmutoğlu, E.; Atay, Y. A feature selection-driven machine learning framework for anomaly-based intrusion detection systems. *Peer-to-Peer Netw. Appl.* **2025**, *18*, 161. [CrossRef]
30. Altan, A.; Karasu, S.; Bekiros, S. Digital currency forecasting with chaotic meta-heuristic bio-inspired signal processing techniques. *Chaos Solitons Fractals* **2019**, *126*, 325–336. [CrossRef]
31. Chou, J.S.; Nguyen, T.K. Forward forecast of stock price using sliding-window metaheuristic-optimized machine-learning regression. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3132–3142. [CrossRef]
32. Shang, Y.; Zheng, M.; Li, J.; Zheng, X. An effective feature selection approach based on hybrid Grey Wolf Optimizer and Genetic Algorithm for hyperspectral image. *Sci. Rep.* **2025**, *15*, 1968. [CrossRef] [PubMed]
33. Altan, A.; Karasu, S.; Zio, E. A new hybrid model for wind speed forecasting combining long short-term memory neural network, decomposition methods and grey wolf optimizer. *Appl. Soft Comput.* **2021**, *100*, 106996. [CrossRef]
34. Nemati, Z.; Mohammadi, A.; Bayat, A.; Mirzaei, A. Metaheuristic and data mining algorithms-based feature selection approach for anomaly detection. *IETE J. Res.* **2024**, *70*, 6040–6054. [CrossRef]
35. Salimi, H. Stochastic fractal search: A powerful metaheuristic algorithm. *Knowl.-Based Syst.* **2015**, *75*, 1–18. [CrossRef]
36. Kahraman, H.T.; Katı, M.; Aras, S.; Taşci, D.A. Development of the Natural Survivor Method (NSM) for designing an updating mechanism in metaheuristic search algorithms. *Eng. Appl. Artif. Intell.* **2023**, *122*, 106121. [CrossRef]

37. Kahraman, H.T.; Aras, S.; Gedikli, E. Fitness-distance balance (FDB): A new selection method for meta-heuristic search algorithms. *Knowl.-Based Syst.* **2020**, *190*, 105169. [CrossRef]
38. Aras, S.; Gedikli, E.; Kahraman, H.T. A novel stochastic fractal search algorithm with fitness-distance balance for global numerical optimization. *Swarm Evol. Comput.* **2021**, *61*, 100821. [CrossRef]
39. Isen, E.; Duman, S. Improved stochastic fractal search algorithm involving design operators for solving parameter extraction problems in real-world engineering optimization problems. *Appl. Energy* **2024**, *365*, 123297. [CrossRef]
40. Kahraman, H.T.; Bakir, H.; Duman, S.; Katı, M.; Aras, S.; Guvenc, U. Dynamic FDB selection method and its application: Modeling and optimizing of directional overcurrent relays coordination. *Appl. Intell.* **2022**, *52*, 4873–4908. [CrossRef]
41. Duman, S.; Kahraman, H.T.; Kati, M. Economical operation of modern power grids incorporating uncertainties of renewable energy sources and load demand using the adaptive fitness-distance balance-based stochastic fractal search algorithm. *Eng. Appl. Artif. Intell.* **2023**, *117*, 105501. [CrossRef]
42. El-Shorbagy, M.A.; Bouaouda, A.; Abualigah, L.; Hashim, F.A. Stochastic Fractal Search: A Decade Comprehensive Review on Its Theory, Variants, and Applications. *Comput. Model. Eng. Sci. (CMES)* **2025**, *142*, 2339–2404. [CrossRef]
43. Heating Oil. Available online: https://www.investing.com/commodities/heating-oil (accessed on 25 October 2025).
44. Abarbanell, J.S.; Bushee, B.J. Fundamental analysis, future earnings, and stock prices. *J. Account. Res.* **1997**, *35*, 1–24. [CrossRef]
45. Edwards, R.D.; Magee, J.; Bassetti, W.C. *Technical Analysis of Stock Trends*; CRC Press: Boca Raton, FL, USA, 2018.
46. Feng, Y.; Palomar, D.P. A signal processing perspective on financial engineering. *Found. Trends Signal Process.* **2016**, *9*, 1–231. [CrossRef]
47. Yang, R.; Yu, L.; Zhao, Y.; Yu, H.; Xu, G.; Wu, Y.; Liu, Z. Big data analytics for financial Market volatility forecast based on support vector machine. *Int. J. Inf. Manag.* **2020**, *50*, 452–462. [CrossRef]
48. Akansu, A.N.; Malioutov, D.; Palomar, D.P.; Jay, E.; Mandic, D.P. Introduction to the issue on financial signal processing and machine learning for electronic trading. *IEEE J. Sel. Top. Signal Process.* **2016**, *10*, 979–981. [CrossRef]
49. Lane, G.C. Lane's stochastics. *Tech. Anal. Stock. Commod.* **1984**, *2*, 87–90.
50. Williams, L.R. *How I Made One Million Dollars Last Year: Trading Commodities*; Windsor Books: Brightwaters, NY, USA, 1979.
51. Williams, L. The ultimate oscillator. *Tech. Anal. Stock. Commod.* **1985**, *3*, 140–141.
52. Levy, R.A. Relative strength as a criterion for investment selection. *J. Financ.* **1967**, *22*, 595–610. [CrossRef]
53. Wilder, J.W. *New Concepts in Technical Trading Systems*; Trend Research: Greensboro, NC, USA, 1978.
54. Lambert, D.R. Commodity channel index: Tool for trading cyclic trends. *Tech. Anal. Stock. Commod.* **1983**, *1*, 120–122.
55. Quong, G.S.; Soudack, A. Volume-weighted RSI: Money flow. *Tech. Anal. Stock. Commod.* **1989**, *7*, 76–77.
56. Erdinç, Y. *Yatırımcı ve Teknik Analiz Sorgulanıyor*; Siyasal Kitabevi: Ankara, Turkey, 2004; s.48.
57. Mandelbrot, B.B.; Aizenman, M. *Fractals: Form, Chance, and Dimension*; Echo Point Books & Media: Brattleboro, VT, USA, 1979.
58. Mandelbrot, B.; Hudson, R.L. *The Misbehavior of Markets: A Fractal View of Financial Turbulence*; Basic Books: New York, NY, USA, 2007.
59. Chen, Y.; Chen, L. *Fractal Geometry*; Earthquke Press: Beijing, China, 1998; pp. 5–7.
60. Beyer, W. Wikipedia: The Mandelbrot Set Within a Continuously Colored Environment. Available online: https://en.wikipedia.org/wiki/Mandelbrot_set (accessed on 25 October 2025).
61. Goel, V.; Kaur, A. A Novel Technique for Optimization of Artificial Neural Network Using Ensemble of Chimp, Harris Hawks and Manta Ray Foraging Optimization Algorithms for Enhancing Software Maintainability Prediction. *Arab. J. Sci. Eng.* **2025**, *50*, 16053–16087. [CrossRef]
62. Zeng, C.; Ma, C.; Wang, K.; Cui, Z. Predicting vacant parking space availability: A DWT-Bi-LSTM model. *Phys. A Stat. Mech. Its Appl.* **2022**, *599*, 127498. [CrossRef]
63. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
64. Jetson Orin Nano Developer Kit. Available online: https://www.nvidia.com/tr-tr/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit (accessed on 25 October 2025).