Journal of King Saud University - Computer and Information Sciences 36 (2024) 102180

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full length article

An efficient hybrid approach for forecasting real-time stock market indices

Riya Kalra [a,1], Tinku Singh [b,1], Suryanshi Mishra [c], Satakshi [c], Naveen Kumar [d], Taehong Kim [b,*], Manish Kumar [a]

[a] Department of IT, Indian Institute of Information Technology Allahabad, Prayagraj, 211012, Uttar Pradesh, India
[b] School of Information and Communication Engineering, Chungbuk National University, Cheongju-si, 28644, Chungcheongbuk-do, South Korea
[c] Department of Maths, SHUATS, Prayagraj, 211007, Uttar Pradesh, India
[d] School of Computing, DIT University, Dehradun, 248009, Uttrakhand, India

ARTICLE INFO

Keywords:
Real-time forecasting
Stock market indices
Incremental learning
Deep learning
Technical indicators

ABSTRACT

The stock market's volatility, noise, and information overload necessitate efficient prediction methods. Forecasting index prices in this environment is complex due to the non-linear and non-stationary nature of time series data generated from the stock market. Machine learning and deep learning have emerged as powerful tools for identifying financial data patterns and generating predictions based on historical trends. However, updating these models in real-time is crucial for accurate predictions. Deep learning models require extensive computational resources and careful hyperparameter optimization, while incremental learning models struggle to balance stability and adaptability. This paper proposes a novel hybrid bidirectional-LSTM (H.BLSTM) model that combines incremental learning and deep learning techniques for real-time index price prediction, addressing these scalability and memory challenges. The method utilizes both univariate time series derived from historical index prices and multivariate time series incorporating technical indicators. Implementation within a real-time trading system demonstrates the method's effectiveness in achieving more accurate price forecasts for major stock indices globally through extensive experimentation. The proposed model achieved an average mean absolute percentage error of 0.001 across nine stock indices, significantly outperforming traditional models. It has an average forecasting delay of 2 s, making it suitable for real-time trading applications.

1. Introduction

In recent years, researchers have increasingly focused on predicting stock market indices rather than individual stock prices (Sezer et al., 2020). Stock market indices reflect diverse aspects of market momentum and global stock market trends (Jasic and Wood, 2004), making them typically more volatile than individual stocks. Consequently, predicting these indices is particularly challenging due to the non-linear and non-stationary nature of time series data in stock markets.

Stock market data is considered time-series data because it is generated at regular intervals. It can be categorized into univariate and multivariate time series. A univariate time series model relies solely on a single variable, like closing prices, while a multivariate time series includes multiple variables such as technical indicators, moving averages, and volume. This makes multivariate models more responsive to recent trends and price movements and more reliable for stock price forecasts. In high-frequency trading (HFT), these methods are especially advantageous because they enable traders to quickly respond to market changes and take advantage of small price fluctuations.

Machine learning and deep learning have emerged as powerful tools for extracting meaningful insights from large, complex stock market data, which is generated at regular intervals. Among them, Long Short-Term Memory (LSTM) networks are particularly effective for time series

* Corresponding author.
E-mail addresses: mit2021055@iiita.ac.in (R. Kalra), tinku.singh@cbnu.ac.kr (T. Singh), 19phmath105@shiats.edu.in (S. Mishra), satakshi@shiats.edu.in (Satakshi), naveen.kumar@dituniversity.edu.in (N. Kumar), taehongkim@cbnu.ac.kr (T. Kim), manish@iiita.ac.in (M. Kumar).
[1] These authors contributed equally to this work.

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2024.102180
Received 14 May 2024; Received in revised form 26 August 2024; Accepted 26 August 2024
Available online 29 August 2024

forecasting. However, traditional LSTM models operate in an offline batch mode, requiring retraining with the entire dataset whenever new data becomes available, which is both time-consuming and prone to issues such as catastrophic forgetting (Maloof and Michalski, 2004). On the other hand, incremental learning allows models to update their parameters as new data arrives. This approach is particularly advantageous in HFT scenarios, where decisions must be made rapidly based on the latest market conditions. However, incremental models can face challenges related to memory constraints and balancing the integration of new information with historical knowledge.

To address these challenges, this paper proposes a novel hybrid model H.BLSTM, which combines the strengths of both deep learning and incremental learning. The H.BLSTM model leverages bidirectional LSTM networks to capture temporal dependencies in both past and future directions, enhancing its ability to understand complex patterns in multivariate time series stock market data. Additionally, the model incorporates a hybrid learning mechanism that combines incremental updates during trading sessions with comprehensive batch retraining at the end of each session. This dual approach ensures that the model remains responsive to new data while retaining the robustness of a fully trained model. Our motivation lies in leveraging deep learning and incremental models for index forecasting to address the challenges posed by fast-paced HFT data and ensure accurate real-time predictions. This paper makes the following contributions:

- Introducing the hybrid multivariate time series forecasting method, which effectively addresses the challenges of forecasting non-linear and non-stationary financial time series by integrating deep learning with incremental learning, ensuring adaptability in real-time stock market prediction in HFT environments.
- Combining technical indicators with historical index prices to create a more informative multivariate time series, leading to more accurate financial forecasts.
- Establishing a dual learning mechanism in the H.BLSTM model that combines incremental updates during trading sessions with batch retraining post-session, striking a balance between model stability and adaptability in HFT environments.
- Demonstrating the effectiveness of the H.BLSTM model through extensive experimentation across nine major global stock indices in terms of prediction accuracy and response time, while highlighting its robustness and real-time applicability to financial trading systems, and acknowledging its limitations with suggestions for future improvements.

The remainder of this paper is organized as follows: Section 2 offers an extensive review of the existing literature, and Section 3 introduces the fundamental concepts and theories necessary for understanding the proposed methodology. Section 4 presents the detailed framework and implementation of the proposed H.BLSTM model, and Section 5 presents the evaluation metrics, the dataset employed, and a discussion of the experimental results. Finally, Section 6 summarizes the key findings and contributions of this study and discusses potential avenues for future research.

2. Literature review

The aim of finance research has shifted from macro to micro as technology has advanced, leading to the analysis of high-frequency (HF) data rather than low-frequency data. HF data are highly nonlinear, as demonstrated by Jobson and Korkie (1980), and the non-normality of HF financial data was highlighted by Jacquier et al. (2004), who proposed a stochastic volatility model. With the rapid advancements in IoT technologies, the concept of real-time forecasting in financial markets can draw parallels to real-time data processing in smart systems. Just as smart systems rely on continuous, HF data to detect anomalies and ensure safety, HFT relies on real-time market data to predict and react to market movements effectively (D'Angelo et al., 2023). According to Ap Gwilym and Sutcliffe (2012), HF data offers many opportunities to analyze market activity more thoroughly. Sun et al. (2014) found that HFT can reduce execution costs by delivering liquidity using tick-level data from 105 United States (US) stocks between January 2008 and October 2010. Guo et al. (2018) suggested an adaptive support vector regression (SVR) for stock data on various time scales including 5-min, 30-min, and daily data on the Shanghai stock market, showing that SVRs with dynamically variable learning parameters produce better results than other approaches, including SVRs and back-propagation neural networks. Rundo (2019) proposed a combined algorithm utilizing supervised deep learning and reinforcement learning to forecast the short-term movement in the Foreign Exchange (FOREX) market maximizing return on investment. Zhou et al. (2018) demonstrated a generic framework for adversarial training to predict the HF stock market utilizing LSTM along with convolutional neural networks (CNN). According to them, the proposed approach can improve forecast accuracy and reduce forecast error effectively.

Several innovative approaches have been proposed to enhance financial forecasting models in recent years. An intraday trading strategy with several objectives is suggested in research by Si et al. (2017). The main idea is to represent and trade intraday financial signals using a multi-objective deep reinforcement learning methodology. To extract market-deep features, the authors created a deep neural network. They then used a reinforcement learning framework (with ad-hoc LSTMs) to generate continuous trading decisions. Borovkova and Tsiamas (2019) introduced a machine learning framework employing an ensemble model that combines several LSTM neural networks to determine intraday directional price forecasts. The framework was applied to 22 large-capital US stocks, using around 19,000 observations made per stock at 5-min intervals. While effective, this method can be computationally expensive and may not scale well with an increasing number of stocks or higher frequency data. McGroarty et al. (2019) introduced an agent-based simulation environment to analyze algorithmic trading strategies. Their main aim was to identify emerging trends resulting from complex market interactions. The model can accurately replicate various market characteristics, such as clustering volatility, the autocorrelation of returns, concave price impact, long-term memory in order flow, and extreme price occurrences. However, agent-based models often require extensive calibration and validation against real-world data, which can be time-consuming and may not always generalize well to different market conditions.

Researchers have suggested multiple incremental learning algorithms for solving different problems (Li et al., 2019). Often, incremental learning means the growth or contraction of model structures (Qin et al., 2015; Osório and Amy, 1999; Xu and Wang, 2016). In other scenarios, researchers have proposed a few methods of controlled modification of learner weights, usually executed by retraining samples with significant forecasting errors (Xing et al., 2016; Gu et al., 2014). Qiu et al. (2017) presents a model with ensemble incremental learning that forecasts stock prices. This model consists of two learning models: random vector functional link network and SVR, as well as two decomposition techniques: discrete wavelet transform and empirical mode decomposition. Wang et al. (2021) suggested a new variant of LSTM called IncLSTM that could train the LSTM model incrementally by merging ensemble learning with transfer learning for time series data. The proposed method improved prediction accuracy by 15.6% and reduced training time by 18.8% compared to traditional methods.

The EnsPKDE and IncLKDE algorithms, a unique hybrid machine learning approach with beneficial performance for time series prediction, are given in a study by Zhu and Dai (2021). The fundamental learner is the online sequential-extreme learning machine using kernels. The authors offer a novel incremental learning framework and employ dynamic ensemble pruning with kernel density estimation as a reference, leading to a notable improvement in prediction performance. Nevertheless, the approach's reliance on kernel density estimation may limit its applicability to more complex, high-dimensional

datasets. Shahparast et al. (2023) introduced a new incremental interval type-2 fuzzy classifier for forecasting the upward and downward future trends of the stock market. It responds quickly and online, processes each piece of stock information once, and produces acceptable results in contrast to deep neural networks and batch techniques. However, the fuzzy classifier may not capture the intricate patterns in financial data as effectively as deep learning models, particularly for very HFT scenarios. CNN-LSTM Autoencoders, which combine the feature extraction capabilities of autoencoders with the predictive accuracy of deep neural networks, can also be explored for financial time series forecasting to effectively adapt to market volatility (D'Angelo et al., 2021).

The models discussed in the literature exhibit several limitations that impact their applicability to real-time financial forecasting. The limitations are significant because real-world financial data often involve multiple interacting variables that need to be analyzed together for accurate forecasting. Some models, relying on predefined time scales and static learning parameters, may not effectively capture the complexities inherent in HFT environments. Machine learning models may be better suited to analyzing univariate than multivariate time series data due to their complexity, dimensionality, and dynamic changes (Rundo, 2019; Zhou et al., 2018; Borovkova and Tsiamas, 2019; Si et al., 2017). This limitation is significant because real-world financial data often involves multiple interacting variables that need to be analyzed together for accurate forecasting. Deep learning models, which are more complex than traditional machine learning models, can effectively capture relationships and dynamics in multivariate time series data. Moreover, several researchers have used deep learning to improve forecasting, but deep learning models are trained in a single offline batch process where all the training information is entered into the model to learn at once. This static training approach poses a major drawback as it fails to adapt to new incoming data in real-time, limiting the model's effectiveness in dynamic environments such as financial markets. These methods may struggle to handle very HF data or adapt promptly to rapid changes in market conditions without frequent retraining. Addressing these limitations is crucial for developing robust models that can reliably forecast financial trends in dynamic and fast-paced markets.

A persistent and open issue in deep learning is allowing neural networks to incrementally learn via non-stationary input streams (Chen and Liu, 2018; Hadsell et al., 2020). This is crucial for financial applications where market conditions constantly evolve, and the ability to update the model with new information without retraining from scratch is highly desirable. Furthermore, model training takes significantly longer than generating the data, and the input data is constantly produced, whereas the model forecast occurs in real-time. The motivation of our research is to enhance the accuracy of real-time indices forecasting by using models incrementally and by transforming univariate indices series into multivariate indices series leveraging the benefits of technical indicators. Our research aims to address the limitations of current models by improving adaptability and real-time performance, ensuring that the forecasting model remains relevant and accurate as new data becomes available.

3. Preliminary overview

In the domain of stock market prediction, a comprehensive understanding of various analytical methods and models is essential. This section provides a foundational overview of the key components used in this study, including technical indicators, ML and incremental learning approaches. These tools and techniques form the underlying structure for enhancing real-time stock price prediction and addressing the dynamic challenges of financial markets.

3.1. Technical indicators

Technical analysis involves evaluating charts and using technical indicators like moving averages to forecast future index movements. A study by Oriani and Coelho (2016) showed that using lagging technical indicators, such as the exponential moving average (EMA) and weighted moving average (WMA), as inputs to neural networks can improve the accuracy of index forecasts compared to using only closing prices. This research uses the EMA, which gives more weight to recent data points. The EMA for a given period, P_d, is calculated as follows:

$$ema_{(\mathcal{T})} = \beta \cdot Q_{\mathcal{T}} + (1-\beta) \cdot ema_{(\mathcal{T}-1)}$$

where $ema_{(\mathcal{T})}$ is the EMA at time $\mathcal{T}$, $Q_{\mathcal{T}}$ is the closing price at time $\mathcal{T}$, and β is the smoothing coefficient, defined as:

$$\beta = \frac{2}{P_d + 1}$$

HF traders often use shorter-term moving averages, like EMA5 (Oriani and Coelho, 2016), to identify short-term trends and trading opportunities faster than longer-term averages, such as EMA20.

Correlation/Covariance Analysis: The correlation and covariance functions are essential for understanding the dependence between indices. The auto-covariance function (ACVF) measures the covariance of a time series with itself at different lags, while the auto-correlation function (ACF) normalizes this to make it dimensionless. For a stochastic process $\{C_v, C_r | \quad v, r \in T\}$, the ACVF and ACF are defined as:

$$\begin{aligned}\chi_{(v,r)} = C_{var}[C_v, C_r] &= E[(C_v - E[C_v])(C_r - E[C_r])] \\ &= E[C_v C_r] - E[C_r]E[C_r] \\ C_{rel}[C_v, C_r] &= \frac{C_{var}[C_v, C_r]}{\sqrt{Var[C_v]Var[C_r]}}\end{aligned}$$

Here, $\chi_{(v,r)}$ and $C_{rel}[C_v, C_r]$ are the auto-covariance and auto-correlation functions, respectively. For indices (ψ, ξ), these functions can be computed as:

$$\hat{\gamma}_{(\psi,\xi)} = \frac{1}{N-1}\sum_{t=1}^{N}(\psi_t - \bar{\psi})(\xi_t - \bar{\xi}) \tag{1}$$

$$\hat{\vartheta}_{(\psi,\xi)} = \frac{\sum_{t=1}^{N}(\psi_t - \bar{\psi})(\xi_t - \bar{\xi})}{\sqrt{\sum_{t=1}^{N}(\psi_t - \bar{\psi})^2 \sum_{t=1}^{N}(\xi_t - \bar{\xi})^2}} \tag{2}$$

The sample covariance and correlation for indices ψ and ξ are given by Eqs. (1) and (2) respectively. For lag 0, the ACVF and ACF simplify the variance and auto-correlation at lag 0:

$$\hat{\gamma}_0 = \frac{1}{N-1}\sum_{t=1}^{N}(\xi_t - \bar{\xi})^2$$

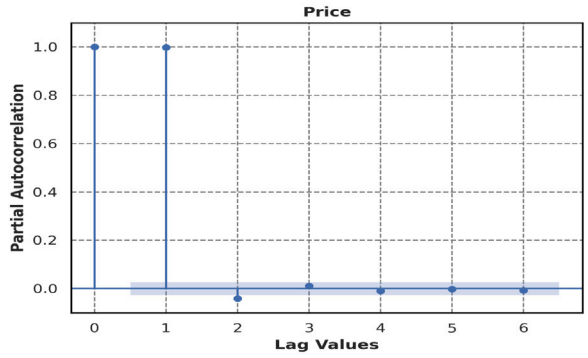
$$\hat{\vartheta}_0 = \frac{\sum_{t=1}^{N}(\xi_t - \bar{\xi})^2}{\sum_{t=1}^{N}(\xi_t - \bar{\xi})^2}$$

The partial autocorrelation function (PACF) measures the direct relationship between S_t and S_{t-m}, accounting for intervening lags. For example, with variables ζ_1 and ζ_2, the PACF can be calculated as:

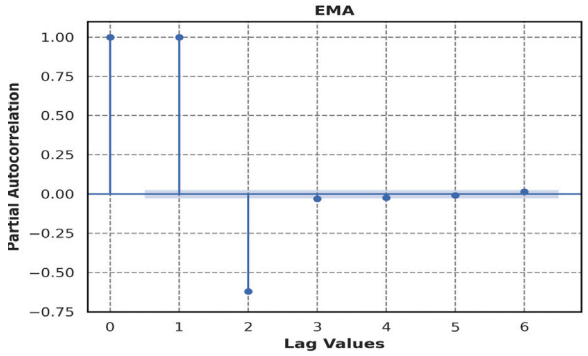
$$\frac{C_{var}(Y, \zeta_3 | \zeta_1, \zeta_2)}{\sqrt{Var(Y|\zeta_1, \zeta_2)Var(\zeta_3|\zeta_1, \zeta_2)}}$$

where Y and ζ_3 are the variables of interest, and ζ_1, ζ_2, and ζ_3 are the predictors.

According to Fig. 1, the 15-min lag values at positions 0 and 1 are positively correlated with the current observations. Price and the EMA have a strong correlation up to a lag value of 3 or 45 min, but after that point, the correlation is negligible. The findings indicate that a maximum of three lags are necessary for accurate prediction. The lag values (3, 9, 27) were also experimentally tested for reliability and consistency with the forecasting models, and it was discovered that lag value 3 was adequate in most cases.



(a)



(b)

Fig. 1. Partial auto-correlation for price and EMA.

3.2. Incremental linear regression

Incremental learning models are effective when data changes frequently, arrives in streams, or when retraining on the complete dataset is computationally expensive or infeasible. These models are crucial for online learning scenarios like the stock market, where real-time predictions are required. Key factors include model architecture, update mechanism, data characteristics, and performance on new data. Incremental linear regression is a representative method, learning new data points one at a time, making it ideal for time-series forecasting. This approach updates parameters with each new observation, using techniques like stochastic gradient descent to adapt to changing data distributions over time, thus extending the model's knowledge (Iscen et al., 2020). Estimators create mini-batches for learning, allowing training and testing on single instances, which is suitable for HFT environments.

At time step t, the model updates weights according to:

$$W_{\ell r_{(t)}} = W_{\ell r_{(t-1)}} + \sigma_t(L_t + \hat{L}_t)K_t \tag{3}$$

where $\sigma_t > 0$ is the step size. This update rule, Eq. (3) minimizes prediction error, thereby enhancing model accuracy. The metric of effectiveness is the cumulative regret after t steps, defined as:

$$R_{I\mathcal{L}} = \sum_{t=(1,\ldots,T)} (L_t - \hat{L}_t) - \sum_{t=(1,\ldots,T)} (L_t - W^T_{\ell r_t} K_t)^2 \tag{4}$$

In Eq. (4), $R_{I\mathcal{L}}$ represents the cumulative regret, indicating the total loss from suboptimal predictions. Minimizing cumulative regret ensures optimal long-term performance, which is crucial in HFT scenarios.

3.3. Decision Tree (DT)

DT learning involves using data points for regression and classification, represented mathematically as an acyclic graph with a fixed root node. Each node represents an attribute, while edges denote decisions based on that attribute. Online DTs improve as new data arrives, adapting to changes by expanding the tree and updating node statistics. The Hoeffding Adaptive Tree Regressor (HATR) is an online DT variant that handles regression. When a node exhibits drift, HATR creates an alternate tree and replaces the node once enough information is collected, reflecting the observed changes.

Let $\overrightarrow{\mathcal{U}}$ represent a stream of instances, denoted as $\overrightarrow{\mathcal{U}} = \{(x^{\mathbf{t}}, \mathfrak{Z}^{\mathbf{t}})\}_{\mathbf{t}=1}^{\infty}$, which may be unbounded. Each instance $(x^{\mathbf{t}}, \mathfrak{Z}^{\mathbf{t}})$, with timestamp $\mathbf{t}$, is drawn from an input space $\mathcal{X} \subset \mathbf{R}^{\mathbf{m}}$, $\mathbf{m} \in \mathbf{Z}^{+}$, and a target space $\mathbf{Z} \in \mathbf{R}$. The input space can also relate to the feature space of stocks in financial markets. The regression task $\mathcal{F} : \mathcal{X} \rightarrow \mathbf{Z}$ is used to learn and update the model continuously as new instances arrive over time. Therefore, $\mathcal{F}$ must be updated online.

In streaming situations, the labels may become available with a delay, but Hoeffding trees reduce this limitation by evaluating the most promising feature with high probability, as described by Domingos and Hulten (2000). The Hoeffding bound, calculated as:

$$\mathfrak{h}_b = \sqrt{\frac{\mathcal{R}^2 \ln(\frac{1}{\psi})}{2\eta}},$$

ensures that the mean of heuristic function $\mathcal{H}_f$ within range $\mathcal{R}$ is approximately $(\mathcal{J} - \mathfrak{h}_b)$ with a probability of $(1 - \psi)$. Hoeffding trees operate under the assumption that the indices' prices, projected in streaming scenarios, closely match the data distribution observed in a sample of length η.

3.4. K-Nearest Neighbors (KNN)

KNN is a simple, efficient, and reliable nonlinear regression technique that forecasts output values based on the K nearest neighbors of the input samples (Hilman et al., 2018). The smoothing parameter K determines the method's adaptability. Unlike traditional models, the KNN regressor does not require a thorough training phase; it simply relies on the primary dataset's inputs and outputs. Online KNN updates occur as new data points arrive, adjusting the weights of neighbors to reflect changing data distributions. It maintains and updates the most recent window-size samples for training, making predictions by averaging the values of the nearest neighbors' recorded samples.

In incremental KNN regression, we estimate a response variable $\theta \in \mathbf{R}$ from instances $\mathcal{S}_{IKNN} = \{s_1, s_2, s_3, \ldots\}$ of features $\phi \in \mathbf{R}^d$. This model continuously learns from arriving streaming data, operating within a network of stock nodes where each node potentially represents an infinite stream of data. For a set $\mathcal{E} = \{(\theta_1, c_1), (\theta_2, c_2), (\theta_3, c_3), \ldots\}$, with $\theta_i \in \mathbf{R}^{(d-1)}$ and $c_i \in \mathbf{R}$, representing stock values, we predict the value of the dth node using the other $(d-1)$ nodes. As new data arrives, the algorithm updates models $\mathcal{H} = \{\mathfrak{h}_1, \mathfrak{h}_2, \ldots\}$, where each model $\mathfrak{h}_{i-1}(\theta_i) = \hat{c}_i$. After each prediction, the model $\mathfrak{h}_i$ is updated, incorporating new data into short-term memory. This memory dynamically adjusts, minimizing the window size to focus on the most recent data, ensuring only the latest concepts are retained. The interleaved train-test queries $\mathcal{Q}(\mathcal{E})$ are computed as:

$$\mathcal{Q}(\mathcal{E}) = \sqrt{\frac{1}{\mathbf{t}} \sum_{i=1}^{\mathbf{t}} [\mathfrak{h}_{i-1}(\theta_i) - c_i]^2}$$

The testing windows are defined as:

$$\mathcal{M}_{\mathcal{Y}} = \{(u_{\mathbf{t}-\mathcal{Y}+1}, v_{\mathbf{t}-\mathcal{Y}+1}), \ldots, (u_{\mathbf{t}}, v_{\mathbf{t}})\}$$
$$\mathcal{Y} \in \{m, m/2, m/4, \ldots\}, \ \mathcal{Y} \geq \mathcal{M}_{min}$$
$$\mathcal{M}_{S\mathcal{Y}_{\mathbf{t}+1}} = \text{argmin}_{\mathcal{E} \in \{\mathcal{M}_m, \mathcal{M}_{m/2}, \ldots\}} \mathcal{Q}(\mathcal{E})$$

Data that falls out of the window as the short-term memory adjusts is not discarded but maintained:

$$\mathcal{R}_t = \mathcal{M}_{S\mathcal{Y}_{\mathbf{t}}} / \mathcal{M}_{S\mathcal{Y}_{(\mathbf{t}+1)}}$$

where $\mathcal{M}_{S\mathcal{Y}_t} = \{(s_i, c_i) \in \mathbf{R}^d \times \mathbf{R} \ : \ i = (\mathbf{t}-m+1, \ldots, \mathbf{t})\}$. This setup ensures the model retains only the most relevant information, filtering consistent data into long-term memory and discarding obsolete samples when memory capacity is reached.

4. Methodology

This section details the methodology for forecasting stock prices using HF data. The proposed approach is specifically designed to handle univariate and multivariate time series data. The methodology is structured into four key stages: data collection and preprocessing, calculation of technical indicators for multivariate time series, incremental updates involving the initialization and continuous learning of the H.BLSTM model, and finally, the prediction and evaluation of model performance using various metrics. Fig. 2 and Algorithm 1 offer a comprehensive guide to understanding and implementing the proposed forecasting approach. Specifically, Fig. 2 visually illustrates the flow from data collection to real-time predictions, while Algorithm 1 provides the step-by-step pseudocode for initializing the model with historical data and updating it incrementally with real-time inputs. The following subsections provide a detailed explanation of each stage of the methodology.

Algorithm 1 Pseudocode for H.BLSTM

1: Data Collection and Preprocessing:
   - I. Gather historical data via web scraping
   - II. Stream real-time stock prices as live data
   - III. Define a function to preprocess data:
     - a. Remove null values
     - b. Remove duplicate instances
     - c. Ensure the order of data
     - d. Convert string date-time to numerical timestamp
   - IV. Apply preprocessing to the data

2: Calculate Technical Indicators (for multivariate time series):
   - I. Define a function to calculate EMA and other indicators
   - II. Apply the function to the preprocessed historical and live data if multivariate

3: Incremental Updates:
   - I. Initialize and train the H.BLSTM model using the preprocessed historical data
   - II. Define a function to update the model with new data:
     - a. Preprocess new data
     - b. Calculate technical indicators if multivariate
     - c. Update the model with the new data

4: Forecast and Evaluate:
   - I. Make predictions with the updated model
   - II. Evaluate the predictions using MAE, RMSE, and MAPE

4.1. Data collection and preprocessing

This step involves collecting and preprocessing data for time-series forecasting. Historical data is gathered through web scraping, while real-time stock prices are streamed at 15-min intervals to ensure up-to-date inputs. The collected data undergoes a thorough preprocessing phase to enhance its quality and usability. This process includes removing null values and duplicate instances, ensuring the correct chronological order, and converting string date-time values into numerical timestamps.

4.2. Calculate technical indicator

Technical indicators enhance the model's performance by capturing significant relationships and dependencies between variables. In this study, EMA5, as explained in Section 3.1, is particularly well-suited for our analysis. EMA5 is computed from the preprocessed data to smooth out short-term price fluctuations, thereby enabling the model to detect and utilize significant patterns more effectively. By incorporating EMA5, the model is better equipped to manage the complexities inherent in multivariate time series data, ultimately leading to improved prediction accuracy.

4.3. Incremental updates

In real-time forecasting, the ability to adapt quickly to new data is crucial. Incremental updates allow models to refine their predictions as new data becomes available, making them highly effective for dynamic environments like the stock market.

H.BLSTM model: H.BLSTM model is designed to handle both univariate and multivariate time series and is initially trained using preprocessed historical data. As new data arrives, the model undergoes incremental updates, allowing it to adapt to changing market conditions without requiring complete retraining. This hybrid approach ensures that the model remains responsive to new data while mitigating the computational constraints associated with continuous updates.

Fig. 3 visually represents the architecture of the H.BLSTM model. To design the H.BLSTM architecture, we altered the BLSTM equations (Singh et al., 2022) to enable the model to process input information as it becomes available. The methodology follows a clear sequence of steps, ensuring that each stage of the process is executed in the correct order.

A. Input Data Processing: Initially, the model receives input data from the Online Training Window, where the data is organized into appropriate time intervals. This input data can include either univariate or multivariate time series, depending on the use case.

B. Forward and Backward Passes of the H.BLSTM Model: Once the input data is prepared, it is fed into the H.BLSTM model. The forward ($\rightarrow$) pass of the model is mathematically represented as follows:

$$\overrightarrow{\mathcal{F}_{(\mathbf{T})}} = \vec{\alpha}(\overrightarrow{\Gamma^{(\mathcal{F})}} * \overrightarrow{\Theta_{(\mathbf{T})}} + \overrightarrow{\lambda^{(\mathcal{F})}} * \overrightarrow{\delta_{(\mathbf{T-1})}} + \overrightarrow{\Omega^{(\mathcal{F})}})$$
$$\overrightarrow{\mathcal{I}_{(\mathbf{T})}} = \vec{\alpha}(\overrightarrow{\Gamma^{(\mathcal{I})}} * \overrightarrow{\Theta_{(\mathbf{T})}} + \overrightarrow{\lambda^{(\mathcal{I})}} * \overrightarrow{\delta_{(\mathbf{T-1})}} + \overrightarrow{\Omega^{(\mathcal{I})}})$$

$$\overrightarrow{\mathcal{O}_{(\mathbf{T})}} = \vec{\alpha}(\overrightarrow{\Gamma^{(\mathcal{O})}} * \overrightarrow{\Theta_{(\mathbf{T})}} + \overrightarrow{\lambda^{(\mathcal{O})}} * \overrightarrow{\delta_{(\mathbf{T-1})}} + \overrightarrow{\Omega^{(\mathcal{O})}})$$
$$\overrightarrow{\mathcal{U}_{(\mathbf{T})}} = \vec{\alpha}(\overrightarrow{\Gamma^{(\mathcal{U})}} * \overrightarrow{\Theta_{(\mathbf{T})}} + \overrightarrow{\lambda^{(\mathcal{U})}} * \overrightarrow{\delta_{(\mathbf{T-1})}} + \overrightarrow{\Omega^{(\mathcal{U})}})$$

$$\overrightarrow{\mathcal{K}'_{(\mathbf{T})}} = \overrightarrow{\mathcal{U}_{(\mathbf{T})}} \odot tanh(\overrightarrow{\Gamma^{(\mathcal{K})}} * \overrightarrow{\Theta_{(\mathbf{T})}} + \overrightarrow{\lambda^{(\mathcal{K})}} * \overrightarrow{\delta_{(\mathbf{T-1})}} + \overrightarrow{\Omega^{(\mathcal{K})}}) + (1 - \overrightarrow{\mathcal{U}_{(\mathbf{T})}}) \odot \overrightarrow{\mathcal{K}'_{((\mathbf{T-1}))}}$$

$$\overrightarrow{\mathcal{K}_{(\mathbf{T})}} = \overrightarrow{\mathcal{F}_{(\mathbf{T})}} \odot \overrightarrow{\mathcal{K}_{(\mathbf{T-1})}} + \overrightarrow{\mathcal{I}_{(\mathbf{T})}} \odot \overrightarrow{\mathcal{K}'_{(\mathbf{T})}}$$
$$\overrightarrow{\delta_{(\mathbf{T})}} = \overrightarrow{\mathcal{O}_{(\mathbf{T})}} \odot tanh(\overrightarrow{\mathcal{K}_{(\mathbf{T})}})$$

where $\overrightarrow{\mathcal{F}_{(\mathbf{T})}}, \overrightarrow{\mathcal{I}_{(\mathbf{T})}}, \overrightarrow{\mathcal{O}_{(\mathbf{T})}}$, and $\overrightarrow{\mathcal{U}_{(\mathbf{T})}}$ represent the forget, input, output, and update gates, respectively, at time T. The update gate $\overrightarrow{\mathcal{U}_{(\mathbf{T})}}$ of the model decides whether or not to employ the latest input to retain the cell state. When the update gate is near 1, a newly acquired input will be employed to update the cell state. In contrast, the prior cell state will be retained when the update gate is near 0. The functions $tanh$ and $\vec{\alpha}$ represent the hyperbolic tangent and sigmoid functions, respectively. The $\overrightarrow{\Gamma^{(\mathcal{F})}}, \overrightarrow{\Gamma^{(\mathcal{I})}}, \overrightarrow{\Gamma^{(\mathcal{O})}}, \overrightarrow{\Gamma^{(\mathcal{U})}}$ and $\overrightarrow{\Gamma^{(\mathcal{K})}}$ are the weight functions associated with the input vector $\overrightarrow{\Theta_{(\mathbf{T})}}$, while the weight matrices $\overrightarrow{\lambda^{(\mathcal{F})}}, \overrightarrow{\lambda^{(\mathcal{I})}}, \overrightarrow{\lambda^{(\mathcal{O})}}, \overrightarrow{\lambda^{(\mathcal{U})}}$ and $\overrightarrow{\lambda^{(\mathcal{K})}}$ are associated with the hidden state of the preceding layer $\overrightarrow{\delta_{(\mathbf{T-1})}}$. Here, $\overrightarrow{\Omega^{(\mathcal{F})}}, \overrightarrow{\Omega^{(\mathcal{I})}}, \overrightarrow{\Omega^{(\mathcal{O})}}$, and $\overrightarrow{\Omega^{(\mathcal{U})}}$ are the bias functions, and $\odot$ represents element-wise multiplication in this hybrid model. The

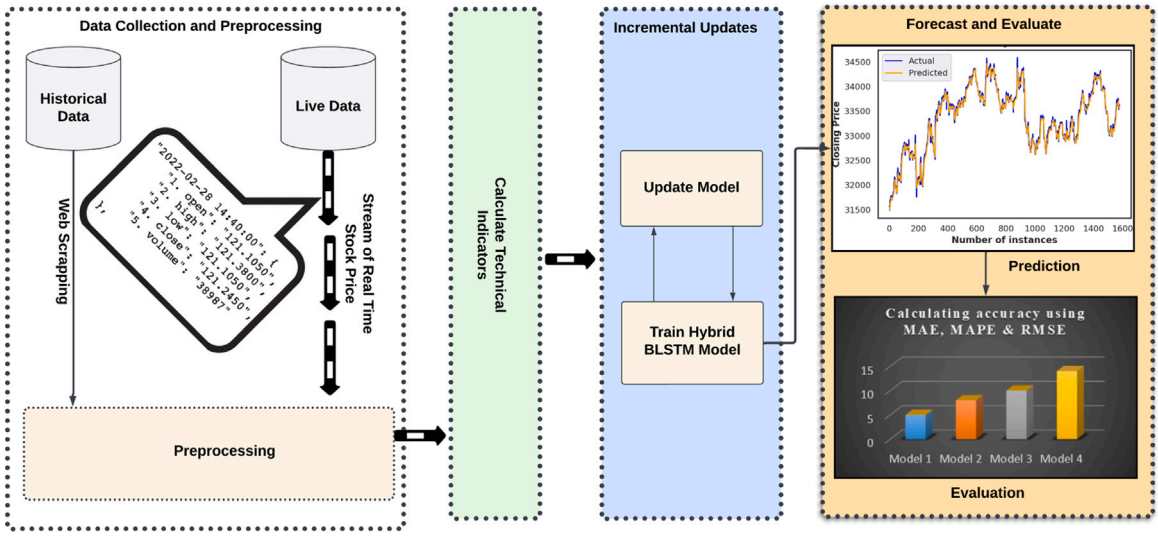


Fig. 2. General overview of the proposed approach.

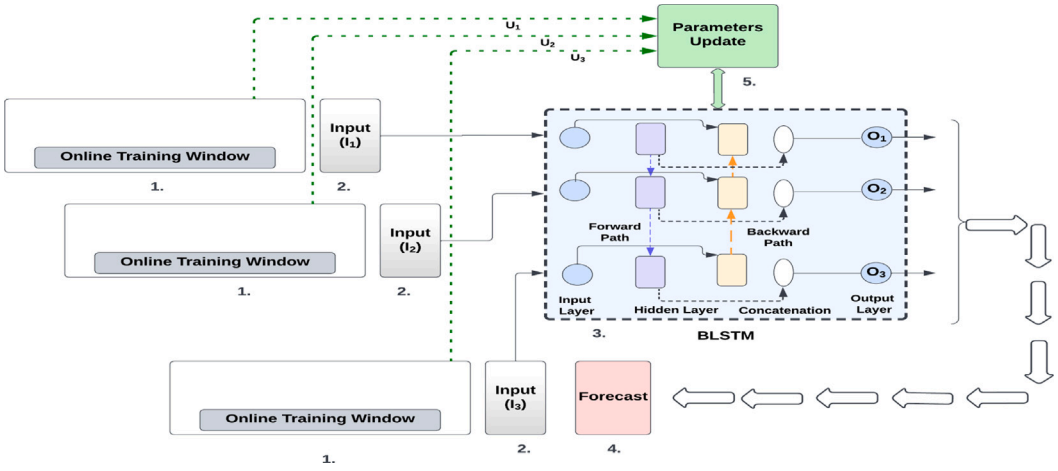


Fig. 3. Architecture of H.BLSTM model.

hidden state of the present timestamp is represented by $\overrightarrow{\delta_{(\mathbf{T})}}$, while $\overrightarrow{\delta_{(\mathbf{T}-1)}}$ represents the hidden state of the preceding timestamps of the hybrid model. The cell states are generated employing the hyperbolic tangent function, while the gates are created using the sigmoid function $\vec{\alpha}$. Every BLSTM layer employs the hidden state of the previous layer as input based on the time step.

C. *Backward Pass and Final Output Generation:* For the backward (←) process of the H.BLSTM model, the same procedure is adopted, but in reverse order. The forward and backward outputs are then concatenated to produce the final output predictions.

D. *Model Parameter Update:* At the end of each trading session, the historical data is merged with the current-day index data to form an updated dataset. This combined dataset is then used to retrain the H.BLSTM model. Retraining the model at the end of each session helps counteract any drift by re-calibrating its parameters to align with the current market conditions.

The robustness of BLSTMs, combined with the flexibility of incremental learning, makes this approach highly effective for real-time stock market prediction compared to state-of-the-art solutions. This hybrid approach enhances the model's adaptability to new data, providing a significant advantage in the dynamic environment of financial markets.

4.4. Evaluation metrics

Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) were employed to assess the forecasting models. The model evaluation metric used in conjunction with regression models is MAE. Every forecast error defines the absolute variation between the model's true and anticipated values. The mathematical formulation of the incremental model MAE (I_{MAE}) is expressed as follows:

$$I_{MAE} = \frac{1}{\mathcal{N}} \sum_{j=(1,\ldots,\mathcal{N})} (Act_j - Prec_j)$$

RMSE, which is equivalent to mean square error, however, the root of the value is considered when estimating model accuracy; as a result, it gives more weight to significant errors. Incremental model RMSE (I_{RMSE}) is computed as follows:

$$I_{RMSE} = \sqrt{\frac{1}{\mathcal{N}} \sum_{j=(1,\ldots,\mathcal{N})} (Act_j - Prec_j)^2}$$

MAPE is identical to MAE and standardized through the actual data. It uses absolute percentage errors, which solves the issue of positive and negative inaccuracies as they cancel each other out. The formulation

Table 1
The nine major stock market indices.

Code	Index	Country	Exchange	Currency
DJI	DOWJONES	US	TVC	US DOLLAR
SPX	S&P500	US	TVC	US DOLLAR
NDX	NASDAX100	US	NASDAQ	US DOLLAR
DXY	DOLLAR	US	TVC	US DOLLAR
NI225	NIKKIE 225	ASIA(JAPAN)	TVC	JAPANESE YEN
000001	SHANGHAI COMP.	ASIA(CHINA)	SSE	CHINESE YUAN
NIFTY	NIFTY50	ASIA(INDIA)	NSE	INDIAN RUPEE
FTSE	FTSE100	LONDON	ATHEX	BRITISH POUND
DAXEUR	DAX/EURO	GERMANY	EASYMARKETS	EURO

for incremental model MAPE (I_{MAPE}) is as follows:

$$I_{MAPE} = \frac{100\%}{\mathcal{N}} \sum_{j=(1,\ldots,\mathcal{N})} \frac{Act_j - Prec_j}{Act_j}$$

where Act_j, and $Prec_j$ indicate the true and anticipated values, respectively. The number of predictions is stated as $\mathcal{N}$.

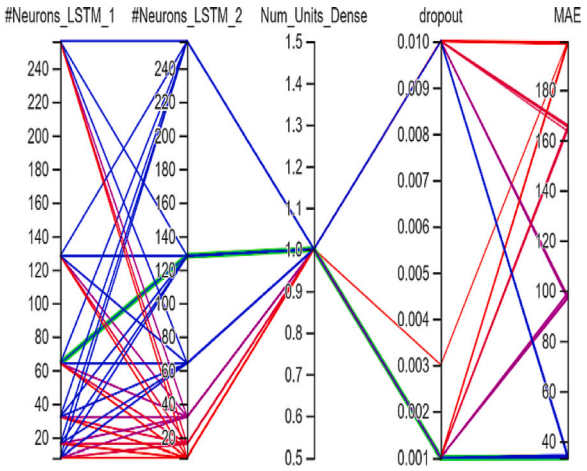


Fig. 4. The output from the tensorboard log file for hyperparameter tuning (optimization) through grid search for H.BLSTM.

5. Experimental results and discussion

Multiple forecasting models were deployed using the suggested framework on real-time data feeds from several stock exchanges. Extensive experiments were conducted using real-time stock market indices data on Google Colaboratory, which provides a single GPU cluster with an NVIDIA K80 GPU, 12 GB of RAM, and a clock speed of 0.82 GHz running Python 3.7. The outcomes for univariate and multivariate time series of selected indices were assessed using incremental learning methods. Univariate time series were derived from the indices' past prices, while multivariate time series incorporated the EMA alongside past prices.

Employed data: Investors and analysts closely monitor major stock market indices worldwide. Market indices gather a selected group of company stocks and regularly evaluate their performance to reflect the performance of the general market or a particular industry. An index helps investors understand the state of the stock market, allows them to understand the market sentiment, and makes it easy to compare the performance of an individual stock. These indices serve as barometers of overall market trends and may indicate broader economic conditions based on their performance. As stated in Table 1, one of the most popular indices has been selected for this research.

The HF historical data of 15-min time intervals as well as live indices feed have been extracted utilizing the tvDatafeed API via TradingView. It offered prices for both historical and real-time indices. There are other APIs as well that provide free HF data, such as AlphaVantage and Yahoo Finance. However, there are certain limitations, like AlphaVantage only provides stock data and not indices data, whereas Yahoo Finance, at maximum, offers 60 days of HFT data.

Splitting criteria: This paper identifies univariate models by the initials (U_) and the model name; for instance, univariate KNN has been denoted as U_KNN. The train-test split was 70%:30%, and 6000 instances were extracted for each instance until February 2023. A comparison was made between the forecasts of nine major stock market indices globally and the actual prices to verify the model's effectiveness.

Performance evaluation: Online linear regression, DTs, and KNN are all considered entirely incremental models as they can learn from new data as it comes in without having to retrain the entire dataset. While H.BLSTM can be updated with new data, it may need extra training to adjust to the most recent information, making it partially incremental. The specifications of hyperparameters shared by all the models were finalized through grid search. Fig. 4 illustrates the results of the grid search for hyperparameter optimization of the H.BLSTM model. The plot demonstrates the influence of various hyperparameter choices on the model's robustness. Different colors (blue, red, green, and purple) represent different hyperparameter combinations and their corresponding MAE values. The green line specifically highlights the grouping of parameters that yield the lowest MAE on the training dataset, indicating the optimal configuration for the model. For H.BLSTM, Adaptive Moment Estimation (Adam) optimization is used to automatically alter the network parameters and learning rate, with a maximum number of training epochs of 50. The loss function is MSE, while the activation function is RELU. MSE is a suitable loss function for the hybrid model as the goal is to minimize the average squared error between the predicted and actual output. For other incremental models, standard scalar preprocessing is utilized. Online linear regression uses the SFG optimizer to update the weights, and the optimized loss function is L2 regularization, which pushes the weights toward zero. The KNN regressor is a non-parametric regression method that keeps track of the last 50 instances of training samples. The HATR model is used for the DT, considering extra parameters (grace period = 50, model selector decay = 0.3, seed = 0).

The most commonly used metrics for assessing the correctness of time series data are MAE and RMSE. Both provide the average model prediction error expressed in units of the relevant variable. Since these are negative-oriented scores, lower values are preferable. However, MAE remains constant, and RMSE rises as the variance of the frequency distribution of error magnitudes also increases. As the test sample grows, RMSE tends to be bigger than MAE. Table 2 evaluates the predicting performance of multiple machine learning models employing RMSE and MAE for the nine major stock market indices.

In the case of univariate models, U_H.BLSTM has a lower RMSE for most of the indices except for DJI. Compared to univariate and other multivariate models, the H.BLSTM model has a lower RMSE for all indices, indicating better performance overall in terms of RMSE. When evaluating MAE for univariate models, the study found that U_KNN has the lowest MAE, whereas H.BLSTM has the least MAE overall. Deep learning models are more complex than machine learning models as they possess more parameters. A less complicated machine learning model, such as KNN, outperforms the deep learning model in terms of MAE because deep learning models are more likely to overfit simple data, like univariate data. In contrast, multivariate data is more complex; therefore, a deep learning model such as H.BLSTM performs better. The achieved results underscore the effectiveness of H.BLSTM model, particularly for complex multivariate data, where it consistently outperforms other models.

Fig. 5 presents a column graph that compares the RMSE of different models across various indices. The performance of models varies across different indices, highlighting the variability in prediction difficulty

Table 2
RMSE and MAE for the nine major stock market indices.

Indices	RMSE							
	U_LR	U_KNN	U_DT	U_H.BLSTM	LR	KNN	DT	H.BLSTM
DJI	1705.168	489.560	1121.879	561.087	1708.700	490.696	715.552	**66.282**
DXY	4.940	1.547	2.853	0.335	4.836	1.547	2.542	**0.069**
FTSE	92.770	30.405	70.055	14.910	107.820	30.362	55.782	**3.730**
NDX	731.040	208.670	424.620	81.003	672.505	210.481	375.331	**37.851**
NI225	592.403	397.073	713.319	189.020	1252.622	399.261	597.766	**78.897**
NIFTY	879.264	250.518	573.598	103.161	870.633	250.646	471.946	**21.805**
SSE	174.715	51.476	116.373	31.596	167.795	51.473	96.329	**7.206**
SPX	199.476	59.761	118.754	38.677	189.445	60.014	100.082	**8.574**
DAXEUR	614.660	188.976	419.427	100.163	598.392	188.613	359.712	**14.102**
	MAE							
DJI	376.933	80.525	265.436	425.332	381.863	82.400	172.717	**45.743**
DXY	0.835	0.096	0.416	0.254	0.804	0.097	0.328	**0.049**
FTSE	22.421	4.970	16.655	10.754	26.195	5.115	13.675	**2.584**
NDX	213.454	42.726	134.727	60.366	192.125	44.050	103.585	**26.268**
NI225	141.285	64.491	151.359	141.052	271.135	66.737	126.650	**44.604**
NIFTY	236.091	33.821	117.987	79.136	256.627	34.831	134.109	**16.300**
SSE	36.708	8.119	27.061	23.818	37.558	8.227	30.357	**5.315**
SPX	43.967	10.811	27.706	13.785	42.678	11.123	22.894	**5.617**
DAXEUR	122.706	17.830	73.573	70.487	123.321	18.094	67.851	**10.602**

Table 3
MAPE for the nine major stock market indices.

Indices	U_LR	U_KNN	U_DT	U_H.BLSTM	LR	KNN	DT	H.BLSTM
DJI	0.011	0.002	0.008	0.013	0.011	0.003	0.005	**0.001**
DXY	0.008	0.001	0.004	0.002	0.007	0.001	0.003	**0.000**
FTSE	0.011	0.002	0.008	0.005	0.012	0.002	0.006	**0.001**
NDX	0.016	0.003	0.011	0.005	0.015	0.004	0.008	**0.002**
NI225	0.005	0.002	0.006	0.005	0.010	0.002	0.005	**0.002**
NIFTY	0.014	0.002	0.007	0.005	0.015	0.002	0.008	**0.001**
SSE	0.011	0.002	0.008	0.007	0.011	0.003	0.009	**0.002**
SPX	0.011	0.003	0.007	0.004	0.011	0.003	0.006	**0.001**
DAXEUR	0.009	0.001	0.005	0.005	0.009	0.001	0.005	**0.001**
Average	0.011	0.002	0.007	0.006	0.011	0.002	0.006	**0.001**

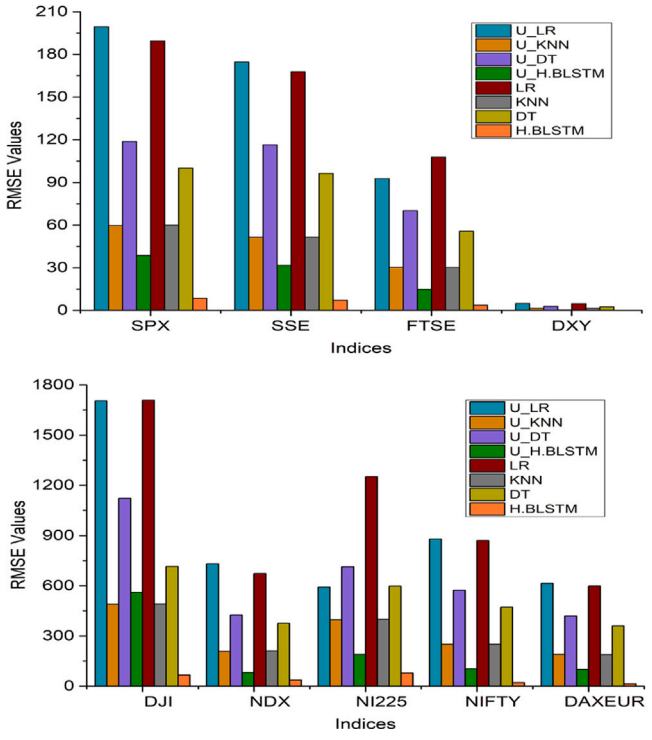


Fig. 5. RMSE of different models across nine major stock market indices.

across different financial markets. It can be noticed that indices from DJI to DAXEUR have higher RMSE values compared to indices from DXY to SPX, as the price of the DJI to DAXEUR indices is also much higher than others. Univariate linear regression is performing worst for most indices except FTSE and NI255, for which multivariate linear regression is performing worst. Additionally, the H.BLSTM model consistently outperforms others, demonstrating superior performance and robustness across various market conditions, including lower RMSE and MAE metrics for indices like NDX, SSE, and FTSE. This suggests that hybrid models, which likely incorporate both linear and non-linear components, can effectively capture the complexities of financial time series data. The consistent performance of the H.BLSTM model across diverse indices highlights its robustness and potential applicability in real-world financial forecasting scenarios.

Using a line graph, Fig. 6 compares MAE for different indices across different models. The H.BLSTM model, depicted with a brown line, consistently maintains low MAE values across all indices. This suggests a robust capability in minimizing prediction errors compared to other models. The MAE values exhibit notable variation among the indices. For instance, the MAE values for the DJI are significantly higher compared to other indices, highlighting the complexity or volatility associated with predicting this particular index. Moreover, RMSEs or MAEs cannot solely determine model accuracy because the price ranges of individual indices may vary, and even currencies used to measure them may differ. For instance, DJI's index price ranges around $33 000, while DXY has an index price range of $100. Therefore, DJI's RMSE values for all models will be higher than DXY's. It is due to even a 1% variation in DJI pricing equals $330, whereas, for DXY, it is only $1. It is required to utilize either normalized RMSE, normalized MAE, or percentage error to compare the precision of models across multiple indices. This study uses MAPE to compare the accuracy of models across

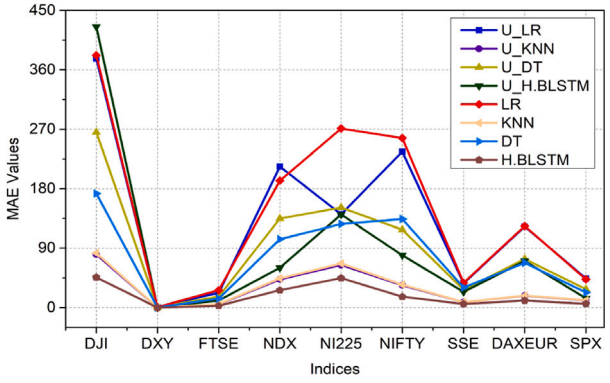


Fig. 6. MAE of different models across nine major stock market indices.

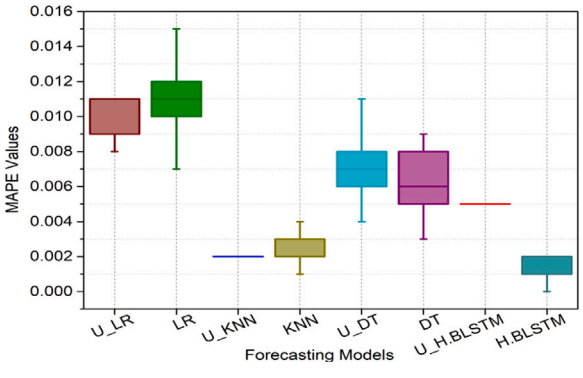


Fig. 7. MAPE of different models across nine major sock market indices.

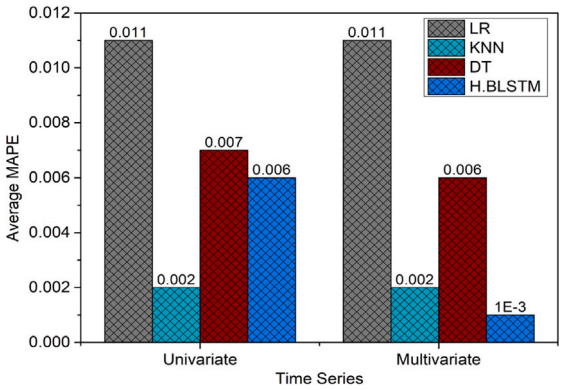


Fig. 8. Average MAPE for different models.

Table 4
Forecasting delay for different models in milliseconds (ms).

Indices extraction time	LR	KNN	DT	H.BLSTM
9:15:00	305.918	434.868	235.685	3784.949
9:30:00	149.638	192.373	207.011	1879.254
9:45:00	148.074	159.101	225.053	2226.574
10:00:00	153.087	176.467	279.092	1576.036
10:15:00	136.045	152.089	229.793	2250.285
10:30:00	127.249	175.937	266.970	1544.448
10:45:00	132.556	167.131	223.551	1801.188
11:00:00	134.307	172.926	249.565	2282.526
11:15:00	166.275	173.284	223.133	1554.219
11:30:00	135.885	192.300	207.535	2110.254
11:45:00	134.745	167.056	222.613	1623.459
12:00:00	179.266	198.117	220.946	2671.328
12:15:00	170.702	164.487	227.350	1680.428
12:30:00	154.227	206.019	236.075	2299.705
12:45:00	130.867	166.001	287.097	1582.454
13:00:00	138.888	168.160	195.831	2434.216
Average	156.108	191.645	233.581	2081.333

different indexes. MAPE is the average of the percentage errors, and Table 3 represents MAPE for the nine major stock market indices.

Using MAPE, the performances of various models can be compared on the same scale, identifying the most efficient one. The models working with the multivariate time series demonstrate better performance than their univariate counterparts. This is evident from the average MAPE values, where H.BLSTM achieves an average MAPE of 0.001, compared to the univariate models, such as U_LR with an average MAPE of 0.011. When considering univariate models, univariate KNN is better than all the other models, while H.BLSTM is the most effective among univariate and multivariate models. In Table 3, the H.BLSTM model consistently achieves the lowest MAPE across all indices, demonstrating its superior performance. Specifically, the H.BLSTM model records the lowest MAPE for indices such as DJI, DXY, FTSE, NDX, NI225, NIFTY, SSE, SPX, and DAXEUR.

This study found that using H.BLSTM models improves accuracy when analyzing multivariate time-series data compared to univariate data. However, for machine learning models, there is an insignificant change in the average MAPE between univariate and multivariate data. It could be because machine learning models usually depend on predefined features and assume that input variables are independent, which means they cannot use the additional information acquired from multivariate time-series data. On the other hand, deep learning models like H.BLSTM can automatically learn hierarchical representations of the data, capturing complex interactions between input variables. These models are more complex and can leverage additional information more effectively, improving accuracy.

Fig. 7 the box plot that visually represents the MAPE values in Table 3. The spread of the boxplots indicates variability in MAPE values. The U_LR and H.BLSTM models demonstrate low variability, with tighter interquartile ranges, indicating more consistent performance across different scenarios. Significantly, H.BLSTM exhibits the lowest variance among all models, highlighting its stability and accuracy in predictions. In contrast, the LR and DT models show greater variability, suggesting less consistent performance. The presence of outliers is minimal, which indicates that the models generally performed consistently without extreme deviations in the MAPE values. The corresponding box plot median values for the best-performing univariate and multivariate models, U_KNN and H.BLSTM, are 0.002 and 0.001, respectively. This indicates that the proposed H.BLSTM model outperforms all other models.

Fig. 8 demonstrates that the H.BLSTM model achieves the best performance, especially in multivariate time series, where it records the lowest average MAPE, signifying its superior prediction accuracy. KNN also shows strong performance in both univariate and multivariate settings. In contrast, LR consistently has the highest MAPE, reflecting poorer predictive accuracy across all scenarios. The DT model shows moderate performance but is outperformed by KNN and H.BLSTM.

The study also compared actual versus predicted index prices, Fig. 9 displays a graph comparing the current prices against the forecasted values generated by the H.BLSTM model for the selected indices. This close alignment between actual and forecasted values in the figure confirms the efficiency of the H.BLSTM model in making accurate predictions. The consistent performance across multiple indices demonstrates the model's ability to generalize effectively, learning from historical data to predict unseen data. This capability highlights the model's robustness and reliability in capturing and predicting stock market trends. The precise predictions provided by the proposed model can be highly beneficial for financial analysts and investors, offering a reliable tool for forecasting stock market movements.

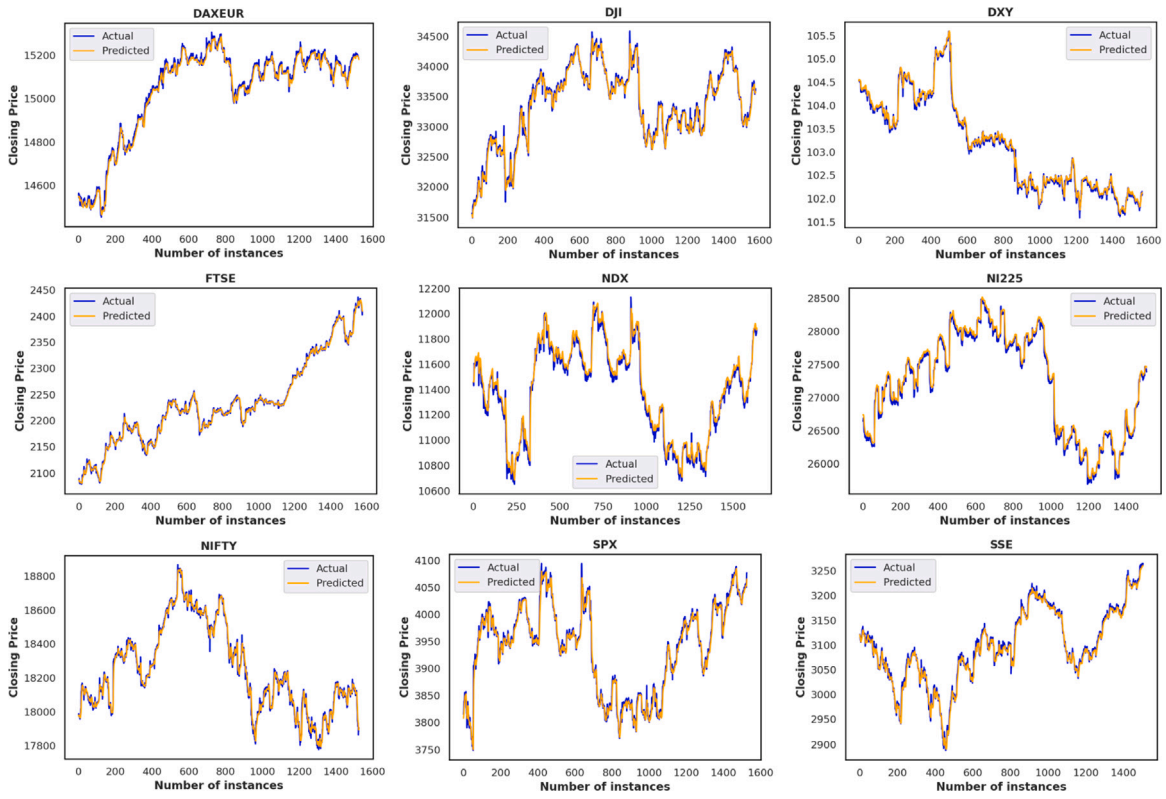


Fig. 9. Actual vs. predicted graph of the H.BLSTM model.

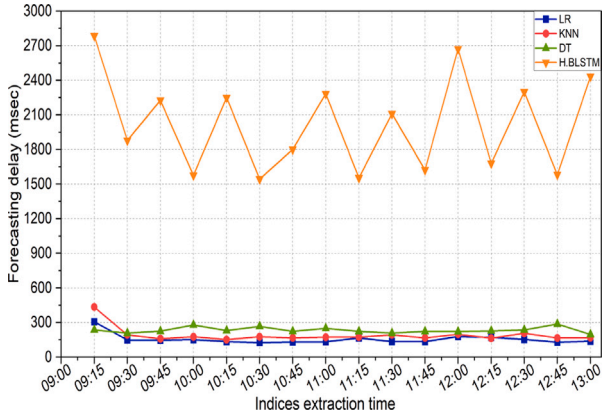


Fig. 10. Forecasting delay for different models.

Moreover, H.BLSTM is a partially incremental model that requires retraining after each trading session, but it outperforms entirely incremental machine learning models. The models were evaluated using real-time trading data collected over 15-min intervals during operational trading hours. The time difference, or latency, was calculated by comparing the actual retrieval time with the forecasting time. Table 4 displays forecasting delay for different models during the trading session. Furthermore, Fig. 10, illustrates that forecasting delays increase with the models' complexity. Still, the forecasting delay in the worst case is not more than a few seconds. A maximum forecasting delay is observed at the start of a trading session for all models. There is a general trend of increasing delay from morning to early afternoon for all models, suggesting a potential impact of increasing data processing load as the day progresses. The average forecasting delay for the H.BLSTM model is 2 s, which is relatively low and suitable for real-time trading applications. This is a significant advantage, as the H.BLSTM model can maintain high prediction accuracy without sacrificing timeliness.

These results emphasize the efficiency and effectiveness of the H.BLSTM model. Even though it requires retraining after each trading session, the model's superior accuracy and relatively low forecasting delay make it highly suitable for real-time trading applications. The balance between model complexity and performance ensures that the H.BLSTM model can provide timely and precise predictions, which is crucial for making informed trading decisions.

6. Conclusion and future work

This study introduces the H.BLSTM model, a hybrid approach that effectively addresses the unique challenges of real-time stock index forecasting in volatile markets. By combining incremental learning with deep learning, H.BLSTM achieves a balance between adaptability and stability, making it particularly well-suited for HFT environments. The integration of technical indicators with historical index data results in a more informative multivariate time series, which enhances the model's performance.

Extensive experiments on nine major stock market indices globally validated the model's performance compared to traditional methods, demonstrating its practical application in financial forecasting. Specifically, H.BLSTM consistently achieved lower MAPE values, averaging 0.001, highlighting its precision in predicting index prices. Furthermore, the model maintained an average forecasting delay of just 2 s, making it highly suitable for real-time trading applications where timely predictions are crucial. These results emphasize the model's predictive accuracy and its robustness in adapting to new data during trading sessions.

Future work could extend this approach to other financial time series, such as mutual funds, cryptocurrency, and gold. Exploring the integration of additional technical indicators and macroeconomic factors could further enhance the model's accuracy and applicability. Additionally, investigating AI-based evaluation metrics beyond RMSE,

MAE, and MAPE could provide a more comprehensive assessment of the model's performance across various forecasting scenarios.

CRediT authorship contribution statement

Riya Kalra: Writing – original draft, Software, Methodology. **Tinku Singh:** Writing – original draft, Software, Methodology. **Suryanshi Mishra:** Writing – original draft, Conceptualization. **Satakshi:** Methodology, Formal analysis. **Naveen Kumar:** Writing – review & editing, Data curation. **Taehong Kim:** Writing – review & editing, Supervision, Conceptualization. **Manish Kumar:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The datasets analyzed during this study have been web scraped through tvDatafeed API via TradingView, the complete details have been discussed in Section 5. It will also be made available on request.

Acknowledgments

This work was partly supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2022R1I1A3072355, 50%) and Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (IITP-2024-2020-0-01462, 50%).

References

Ap Gwilym, O., Sutcliffe, C., 2012. Problems encountered when using high frequency financial market data: suggested solutions. J. Financ. Manag. Anal. 25 (2).

Borovkova, S., Tsiamas, I., 2019. An ensemble of LSTM neural networks for high-frequency stock market classification. J. Forecast. 38 (6), 600–619.

Chen, Z., Liu, B., 2018. Lifelong machine learning. Synth. Lect. Artif. Intell. Mach. Learn. 12 (3), 1–207.

D'Angelo, G., Ficco, M., Robustelli, A., 2023. An association rules-based approach for anomaly detection on CAN-bus. In: International Conference on Computational Science and Its Applications. Springer, pp. 174–190.

D'Angelo, G., Palmieri, F., Robustelli, A., 2021. Effectiveness of video-classification in android malware detection through api-streams and cnn-lstm autoencoders. In: International Symposium on Mobile Internet Security. Springer, pp. 171–194.

Domingos, P., Hulten, G., 2000. Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 71–80.

Gu, B., Sheng, V.S., Tay, K.Y., Romano, W., Li, S., 2014. Incremental support vector learning for ordinal regression. IEEE Trans. Neural Netw. Learn. Syst. 26 (7), 1403–1416.

Guo, Y., Han, S., Shen, C., Li, Y., Yin, X., Bai, Y., 2018. An adaptive SVR for high-frequency stock price forecasting. IEEE Access 6, 11397–11404.

Hadsell, R., Rao, D., Rusu, A.A., Pascanu, R., 2020. Embracing change: Continual learning in deep neural networks. Trends in Cognitive Sciences 24 (12), 1028–1040. http://dx.doi.org/10.1016/j.tics.2020.09.004, URL: https://www.sciencedirect.com/science/article/pii/S1364661320302199.

Hilman, M.H., Rodriguez, M.A., Buyya, R., 2018. Task runtime prediction in scientific workflows using an online incremental learning approach. In: 2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing. UCC, IEEE, pp. 93–102.

Iscen, A., Zhang, J., Lazebnik, S., Schmid, C., 2020. Memory-efficient incremental learning through feature adaptation. In: European Conference on Computer Vision. Springer, pp. 699–715.

Jacquier, E., Polson, N.G., Rossi, P.E., 2004. Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. J. Econometrics 122 (1), 185–212.

Jasic, T., Wood, D., 2004. The profitability of daily stock market indices trades based on neural network predictions: case study for the S&P 500, the DAX, the TOPIX and the FTSE in the period 1965–1999. Appl. Financial Econ. 14 (4), 285–297. http://dx.doi.org/10.1080/0960310042000201228, arXiv:https://doi.org/10.1080/0960310042000201228.

Jobson, J.D., Korkie, B., 1980. Estimation for Markowitz efficient portfolios. J. Amer. Statist. Assoc. 75 (371), 544–554.

Li, J., Dai, Q., Ye, R., 2019. A novel double incremental learning algorithm for time series prediction. Neural Comput. Appl. 31, 6055–6077.

Maloof, M.A., Michalski, R.S., 2004. Incremental learning with partial instance memory. Artificial Intelligence 154 (1–2), 95–126.

McGroarty, F., Booth, A., Gerding, E., Chinthalapati, V.R., 2019. High frequency trading strategies, market fragility and price spikes: an agent based model perspective. Ann. Oper. Res. 282, 217–244.

Oriani, F.B., Coelho, G.P., 2016. Evaluating the impact of technical indicators on stock forecasting. In: 2016 IEEE Symposium Series on Computational Intelligence. SSCI, pp. 1–8.

Osório, F.S., Amy, B., 1999. Neurocomputing 28 (1–3), 191–205.

Qin, Y., Li, D., Zhang, A., 2015. A new SVM multiclass incremental learning algorithm. Math. Probl. Eng. 2015.

Qiu, X., Zhu, H., Suganthan, P.N., Amaratunga, G.A., 2017. Stock price forecasting with empirical mode decomposition based ensemble-support vector regression model. In: International Conference on Computational Intelligence, Communications, and Business Analytics. Springer, pp. 22–34.

Rundo, F., 2019. Deep LSTM with reinforcement learning layer for financial trend prediction in FX high frequency trading systems. Appl. Sci. 9 (20), 4460.

Sezer, O.B., Gudelek, M.U., Ozbayoglu, A.M., 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Appl. Soft Comput. 90, 106181.

Shahparast, H., Hamzeloo, S., Safari, E., 2023. An incremental type-2 fuzzy classifier for stock trend prediction. Expert Syst. Appl. 212, 118787.

Si, W., Li, J., Ding, P., Rao, R., 2017. A multi-objective deep reinforcement learning approach for stock index future's intraday trading. In: 2017 10th International Symposium on Computational Intelligence and Design. ISCID, IEEE, pp. 431–436.

Singh, T., Kalra, R., Mishra, S., Kumar, M., 2022. An efficient real-time stock prediction exploiting incremental learning and deep learning. Evol. Syst. 1–19.

Sun, E.W., Kruse, T., Yu, M.-T., 2014. High frequency trading, liquidity, and execution cost. Ann. Oper. Res. 223, 403–432.

Wang, H., Li, M., Yue, X., 2021. IncLSTM: Incremental ensemble LSTM model towards time series data. Comput. Electr. Eng. 92, 107156. http://dx.doi.org/10.1016/j.compeleceng.2021.107156.

Xing, Y., Shi, X., Shen, F., Zhou, K., Zhao, J., 2016. A self-organizing incremental neural network based on local distribution learning. Neural Netw. 84, 143–160.

Xu, S., Wang, J., 2016. A fast incremental extreme learning machine algorithm for data streams classification. Expert Syst. Appl. 65, 332–344.

Zhou, X., Pan, Z., Hu, G., Tang, S., Zhao, C., 2018. Stock market prediction on high-frequency data using generative adversarial nets. Math. Probl. Eng..

Zhu, G., Dai, Q., 2021. EnsP KDE andIncL KDE: a hybrid time series prediction algorithm integrating dynamic ensemble pruning, incremental learning, and kernel density estimation. Appl. Intell. 51, 617–645.