Stock Price Forecasting Using N-BEATS Deep Learning Architecture

Abstract

Stock prices present unique forecasting challenges due to factors such as market volatility, investor sentiment, and economic indicators, which contribute to significant fluctuations in time series data. This paper addresses these complexities by applying Deep Learning (DL) models to predict stock prices, with a particular focus on the S&P 500 index. Although DL models have shown remarkable success in fields like image processing and natural language processing, they require specialized architectures to effectively handle time series forecasting. This study examines the Neural Basis Expansion Analysis for Interpretable Time Series Forecasting (N-BEATS) model, a novel DL architecture specifically tailored for time series data, using S&P 500 stock price data. The performance of N-BEATS is benchmarked against three baseline models: Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU). The evaluation metrics include Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). Results indicate that the N-BEATS model consistently surpasses the other models in all metrics. Additionally, the Diebold-Mariano (DM) test further validates the superior predictive accuracy of the N-BEATS model compared to the alternatives. This research underscores the potential of the N-BEATS model to significantly improve stock price forecasting, offering valuable insights for investors, financial analysts, and other market participants.

Keyword:*Stock price,Basis expansion, Convolutional Neural Network (CNN), Deep learning, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), N-BEATS*

1.Introduction

Time series analysis is a crucial tool used across various fields to understand and predict future values based on historical data. Its applications span finance, economics, environmental studies, and more, with one prominent use being the analysis of stock prices(Sezer et al. 2020). In the financial sector, accurate time series forecasting is essential for making informed investment decisions, managing risks, and developing effective trading strategies(Shahvaroughi Farahani and Razavi Hajiagha 2021). For instance, the S&P 500 index, a benchmark for the U.S. stock market, is frequently analyzed through historical price data provided by platforms such as

Google Finance. Forecasting these prices is vital for traders, investors, and policymakers to navigate market volatility and optimize financial outcomes.

A range of time series techniques exists for forecasting stock prices, each with its strengths and limitations. Traditional models like AutoRegressive Integrated Moving Average (ARIMA) are well-regarded for their simplicity and effectiveness in capturing linear relationships within data(LIU 2024). However, ARIMA models struggle with nonlinear patterns and require assumptions about data stationarity, limiting their applicability in more complex scenarios(Racocha 2020). As a result, Machine Learning (ML) models emerged as a more sophisticated alternative, offering enhanced capabilities for handling nonlinearity and larger datasets. Yet, ML models often demand extensive manual feature extraction, which can be challenging and labor-intensive(Azevedo et al. 2024).

The emergence of Deep Learning (DL) models marks a significant leap forward in time series forecasting, overcoming many of the limitations associated with traditional and ML approaches(Li and Bastos 2020). Early DL models, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, were pioneering in their ability to capture sequential dependencies in time series data(Hewamalage et al. 2021). Despite their advancements, these models often faced challenges such as vanishing gradients and high computational demands. Building on these initial successes, the N-BEATS (Neural Basis Expansion Analysis for Time Series) model introduces a novel architecture that features stacked blocks of fully connected layers for both forecasting and backcasting. Unlike traditional models, N-BEATS does not rely on time-series-specific components like trend or seasonality, allowing it to dynamically adapt to complex patterns and enhance predictive accuracy(Wang et al. 2022).

Various studies have explored a range of statistical, Machine Learning (ML), and Deep Learning (DL) algorithms for stock price prediction. Conejo et al. (2005) investigated day-ahead electricity price forecasting using ARIMA models. For stock price prediction, Mehtab and Sen (2020)combined statistical, ML, and DL approaches, and Patarwal et al. (2018) introduced the ELM-AE method, which surpasses existing techniques in terms of Mean Squared Error (MSE). Tripathi and Sharma (2023) found that Deep Neural Networks (DNNs) outperformed LSTM and CNN-LSTM models in Bitcoin price predictionSinghal V et al. (2022) improved stock market

forecasting by combining wavelet decomposition with N-BEATS. Aslam et al. (2023)achieved promising results in predicting wind power using the N-BEATS model and Sbrana and Lima de Castro (2023)investigated its performance in forecasting cryptocurrency.Nayak et al. (2024a) utilized Deep learning techniques including NBEATS for improved forecasting of price of TOP crops in India. These studies underscore the efficacy of ML and DL algorithms for price forecasting and the importance of selecting models suited to the specific characteristics of the data.

In this study, we employed the novel deep learning approach N-BEATS alongside baseline models such as CNN, LSTM, and GRU to forecast S&P 500 stock prices sourced from Google Finance. By utilizing these advanced models, the research aims to improve the accuracy of stock price predictions and provide valuable insights into the stock market. This comprehensive analysis highlights not only the effectiveness of N-BEATS compared to traditional models but also its potential for enhancing stock price forecasting.

2. Materials and Methods

2.1Data description

This study utilizes the Closing price of S&P 500 stock, which contains 1340 observations from January 1, 2018, to April 30, 2023. The data is obtained from 'Google Finance'.

2.2 Deep learning techniques used for forecasting price series

2.2.1 Convolutional Neural Network (CNN)

A convolutional neural network (CNN) excels in identifying visual patterns and is distinguished by its specialized architecture. Central to a CNN are three key types of layers: convolutional layers, sub-sampling layers, and fully connected layers. The architecture typically consists of multiple convolutional and sub-sampling layers stacked together, followed by a series of fully connected layers (see Fig. 1).

Convolutional layers are designed to process inputs from neighboring nodes, similar to how cells in the visual cortex work. These layers use shared local weights, which not only help conserve memory but also improve classification performance. Following the convolutional layers, sub-sampling layers perform non-linear down-sampling, reducing the dimensionality of the data. This

reduction lowers local sensitivity and computational complexity, enhancing the network's ability to learn features and patterns more effectively(Sánchez-Reolid et al. 2022).

The final stage involves fully connected layers, which are akin to those in standard neural networks. These layers conduct extensive matrix computations with all activations and nodes. After the convolutional and sub-sampling layers have extracted features, the fully connected layers are responsible for reasoning and generating the model's output. CNNs are trained through backpropagation, which optimizes the model to minimize the difference between actual and target output values. This layered structure, combining convolutional, sub-sampling, and fully connected layers, enables CNNs to adeptly recognize patterns and features in visual data, making them highly effective for a range of computer vision applications.

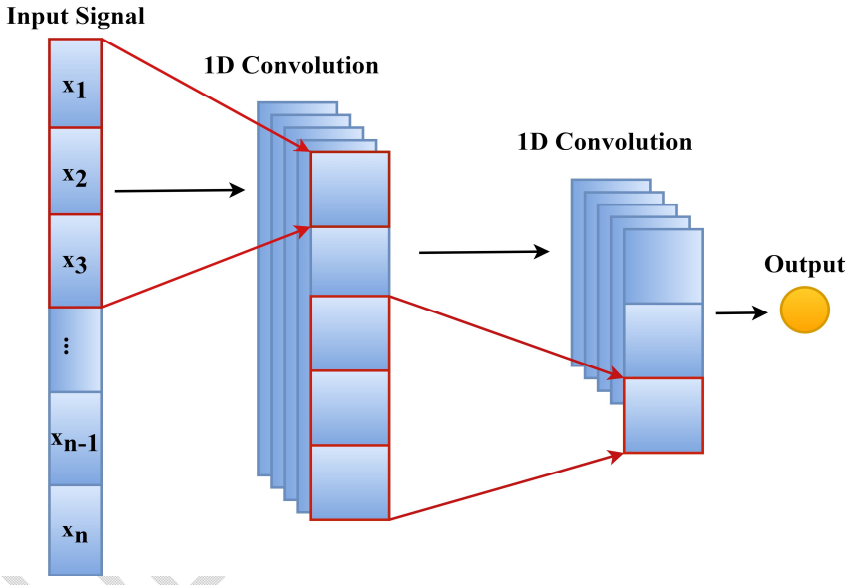

Fig. 1: One-dimension convolutional neural network (1D-CNN) architecture.

2.2.2 Long Short-Term Memory (LSTM)

LSTM was introduced by Hochreiter and Schmidhuber in 1997 to address a critical limitation of traditional Recurrent Neural Networks (RNNs)—their inability to effectively manage long-term dependencies in sequences. The LSTM model integrates specialized gating mechanisms into the RNN architecture to overcome this issue. These gates include the forget gate, the input gate, and the output gate, all implemented as sigmoid layers as shown in Fig. 2(Marino et al. 2016)

The forget gate determines the relevance of previous information by deciding whether to retain or discard it. It outputs f_t, which ranges from 0 to 1, where 0 means complete discarding and 1 indicates full retention. The calculation for the forget gate is:

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right)$$

Similarly, the input gate determines which new values should be updated by processing the previous output (h_{t-1}) and the current input (x_t). It uses a weight matrix W_i, a sigmoid function σ, and a bias term b_i, generating a candidate value for the current cell state $\widehat{C}_t$:

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$$

The candidate cell state value $\widehat{C}_t$ is calculated using the hyperbolic tangent function:

$$\widehat{C}_t = tanh(W_c . [h_{t-1}, x_t] + b_c)$$

The updated cell state C_t combines the forget gate output f_t, the previous cell state C_{t-1}, the input gate output i_t, and the new candidate state $\widehat{C}_t$:

$$C_t = f_t * C_{t-1} + i_t * \widehat{C}_t$$

The output gate then determines how much of the cell state should affect the current output. The output o_t is calculated using a sigmoid function, and the final output h_t is derived by applying the hyperbolic tangent function to the current cell state and scaling it by the output gate value:

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

In the LSTM model's fully connected layer, the Rectified Linear Unit (ReLU) activation function is utilized, while the mean square error (MSE) serves as the loss function for performance optimization (Bakir et al. 2018).

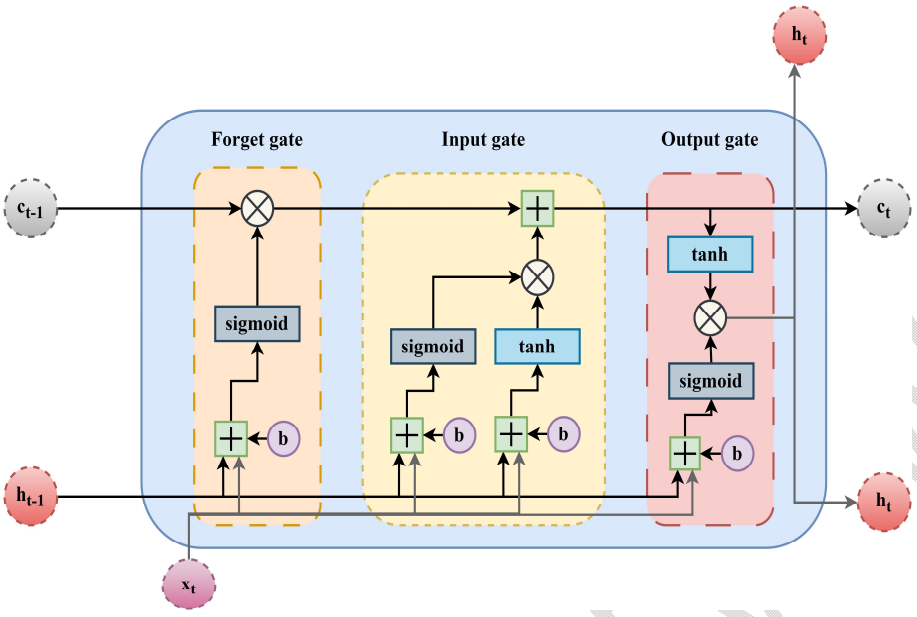

Fig. 2: LSTM architecture

2.2.3 Gated Recurrent Unit (GRU)

Introduced in 2014, Gated Recurrent Units (GRUs) offer a streamlined and more efficient alternative to Long Short-Term Memory (LSTM) layers (Chung et al. 2014). Unlike LSTMs, which use separate input and forget gates, GRUs combine these into a single update gate and unify the hidden and cell states as shown in Fig. 2, resulting in fewer parameters and reduced computational complexity. This simplification enhances the efficiency and cost-effectiveness of GRUs.

GRUs are designed to prioritize recent information, which is often more relevant for predicting future outcomes compared to older data. This focus on recent events helps GRUs perform effectively by retaining pertinent information while discarding less relevant past data. The GRU's reset gate controls how much of the previous information is discarded, while the update gate manages the integration of new information into the current state(Nayak et al. 2024b).

The update gate is computed as:

$$Z_t(Update\ Gate) = \sigma(W_z.[h_{t-1}, x_t] + b_z)$$

The reset gate is calculated using:

$$R_t(Reset\ Gate) = \sigma(W_R.[h_{t-1}, x_t] + b_R)$$

The new candidate state, $\widehat{H}_t$, is obtained by applying the hyperbolic tangent $(tanh)$function to the product of the reset gate and the previous hidden state:

$$\widehat{H}_t = tanh(W.[R_{t*}(h_{t-1}, x_t)] + b_z)$$

This calculation allows the GRU to manage the flow of relevant information effectively. The final hidden state, h_t, is computed by combining the previous hidden state and the new candidate state, moderated by the update gate:

$$h_t = (1 - Z_t) * h_{t-1} + Z_t * h_t)$$

In GRUs, the update and reset gates, controlled by sigmoid activation functions, regulate the recurrent connections and input values. The weight matrices W_Z, W_Z and W, along with the bias terms b_Z and b_R, determine how input values are processed within these gates. The final hidden state represents a blend of the previous state and the new candidate, adjusted according to the update gate's output.

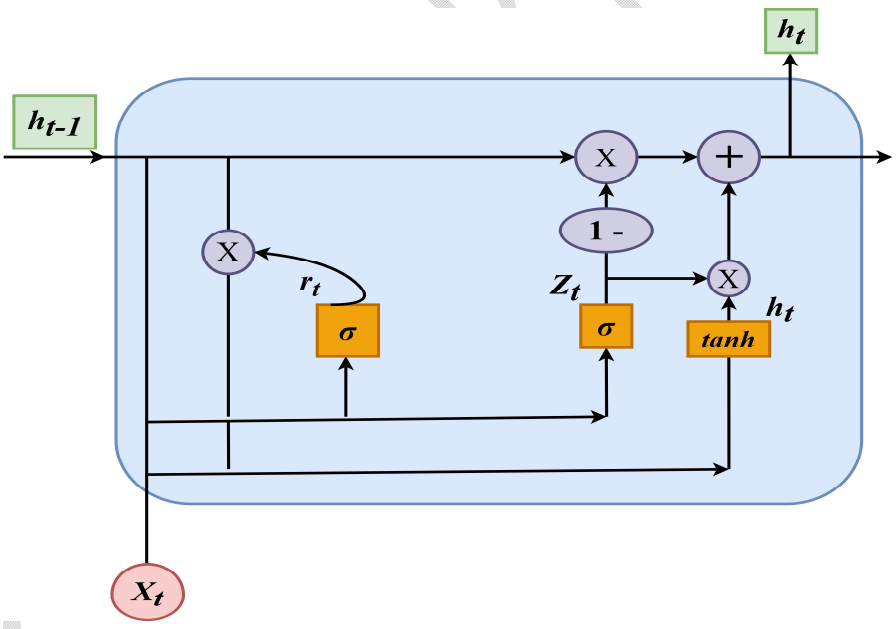

Fig. 3: GRU architecture

2.2.4 Neural Basis Expansion Analysis for Interpretable Time Series Forecasting (N-BEATS)

N-BEATS enhances data interpretation and prediction accuracy through basis expansion, a technique that transforms the original data into a higher-dimensional space to capture non-linear relationships (Oreshkin et al. 2021). Unlike traditional methods, which manually select basis expansion techniques, N-BEATS employs a neural network to automatically determine and optimize the most effective data augmentation strategy during training. This adaptive approach allows the model to tailor the basis expansion to the specific characteristics of the dataset, resulting in improved interpretability and predictive performance.

For univariate point forecasting, the objective is to predict a future value vector $y \in R^H = [y_{T+1}, y_{T+2}, \dots, y_{T+H}]$over a forecast horizon H which is depicted in Figure 4. This prediction is based on a historical time series $[y_1, \dots, y_T] \in R^T$ and uses a lookback window of length $\leq T$, represented by $x \in R^t = [y_{T-t+1}, \dots, y_T]$, ending with the last observed value y_T. The predicted values are denoted as $\hat{y}$.

N-BEATS is built on three core principles: creating a simple yet expressive deep learning architecture, avoiding dependence on time-series-specific components like trend or seasonality, and ensuring model extendibility for better interpretability. The model inputs the lookback period, while the forecast period contains actual values used to evaluate predictions. The input sequence length is usually a multiple of the forecast horizon *H*, ranging from 2*H* to 7*H*. The architecture consists of stacked layers, with each stack composed of multiple blocks. Each block has four fully connected layers and generates two outputs: a forecast and a backcast. The forecast predicts future values, while the backcast allows for immediate comparison with the input sequence, helping assess the model's fit. Each block calculates expansion coefficients θ and performs basis expansion g, enhancing the model's adaptability and accuracy(Nayak et al. 2024a).

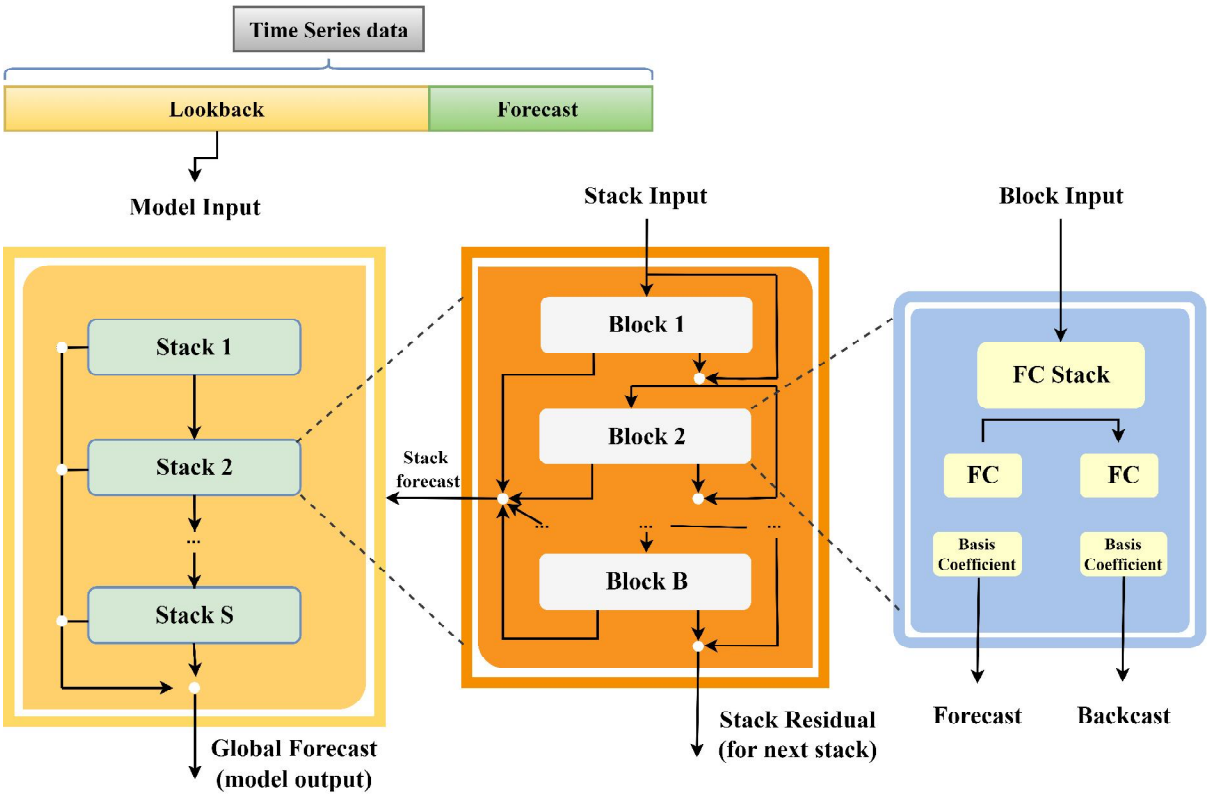

Fig. 4: The architecture of N-BEATS

In this architecture, only the first block receives the actual input sequence. Subsequent blocks are given the residuals from the previous block, ensuring that each block captures information missed by the previous one. The basic block, depicted in Fig. 4, has a fork architecture. The l^{th}block processes its input x_l and produces two vectors, $\hat{x}_l$ and $\hat{y}_l$. For the first block, x_l represents the input history lookback window ending with the last observed value. For later blocks, $x_l$comprises residual outputs from earlier blocks. Each block generates a forward forecast $\hat{y}_l$ of length H and a backcast $\hat{x}_l$, an estimate of x_l within the block's functional space.

Internally, each basic block includes two components. The first part is a fully connected network that produces forward θ_l^f and the backward θ_l^b expansion coefficients. The second part involves backward $g_l^b$and the forward g_l^f basis layers, which project the expansion coefficients θ_l^f and θ_l^b onto basis functions to generate the backcast$\hat{x}_l$ and the forecast outputs $\hat{y}_l$.

The operations of the first part of the l^{th}block is defined by:

$$h_{l,1} = FC_{l,1}(x_l), \qquad h_{l,2} = FC_{l,2}(h_{l,1}), h_{l,3} = FC_{l,3}(h_{l,2}), \qquad h_{l,4} = FC_{l,4}(h_{l,3})$$

$$\theta_l^b = Linear_l^b(h_{l,4}), \qquad \theta_l^f = Linear_l^b(h_{l,4})$$

where the Linear layer represents a simple linear projection, $\theta_l^f = W_l^f h_{l,4}$. The FC layer refers to a fully connected layer with ReLU non-linearity. This part aims to predict the forward expansion coefficients θ_l^f to optimize the accuracy of $\hat{y}_l$ and the backward expansion coefficients θ_l^b to produce an estimate of x_l, facilitating the removal of irrelevant components for downstream blocks.

The second part of the block maps the expansion coefficients θ_l^f and θ_l^b to outputs using basis layers:

$$\hat{y}_l = g_l^f(\theta_l^f), \hat{x}_l = g_l^b(\theta_l^b)$$

which is further defined as:

$$\hat{y}_l = \sum_{i=1}^{dim(\theta_l^f)} \theta_{l,i}^f v_i^f, \qquad \hat{x}_l = \sum_{i=1}^{dim(\theta_l^b)} \theta_{l,i}^b v_i^b$$

where v_i^f and v_i^b are the forecast and backcast basis vectors, respectively, and $\theta_{l,i}^f$ is the i^{th} element of θ_l^f.

The N-BEATS architecture diverges from classical residual networks by introducing a hierarchical doubly residual topology with two residual branches: one for the backcast prediction and another for the forecast branch. This topology is described by:

$$x_l = x_{l-1} - \hat{x}_{l-1}, \qquad \hat{y} = \sum_l \hat{y}_l$$

For the first block, the input is the model-level input x, so $x_1 \equiv x$. Subsequent blocks analyze residuals from previous blocks, removing well-approximated signal portions to simplify the forecasting task for later blocks. This structure aids in smoother gradient backpropagation. Each block produces a partial forecast $\hat{y}$, aggregated first at the stack level and then at the overall network level. The final forecast $\hat{y}$is the sum of these partial forecasts. The model's design allows for arbitrary g_l^b and g_l^f functions, enhancing transparency to gradient flows. When g_l^b and g_l^f are structured and shared across layers, the model gains interpretability through the

aggregation of meaningful partial forecasts. The residual connections capture missed information from previous blocks, with the final forecast being a composite of all partial predictions.

2.3Data Pre-processing and Normalization

To achieve effective fitting of deep learning models and ensure unbiased extrapolation, it is crucial to preprocess and normalize the data series. Normalization transforms the values of both series to a uniform range between 0 and 1 while maintaining their original shape. This process enhances the robustness of model training and improves the model's ability to generalize patterns from the data. The normalization is carried out using the following formula:

$$X'_t = \frac{X_t - X_{min}}{X_{max} - X_{min}}$$

where X_{min} andX_{max} represent the minimum and maximum values of the series, respectively, and X_t is the value at time *t*. The result, X'_t, is the normalized value

2.4Hyperparameter tuning:

Here, we provide a comprehensive overview of the hyperparameters used in developing the various forecasting models. The fine-tuning process utilized the random search method for hyperparameter optimization. Specifically, random search was employed to optimize hyperparameters for the DL models. Training accuracy was evaluated across a range of randomly selected hyperparameter combinations, with the final configuration selected based on the highest achieved accuracy, as detailed in Table 1. In our study, we implemented four different algorithms: CNN, LSTM, GRU, and N-BEATS to forecast the S&P 500 stock price.

Table 1: The hyperparameters and their values of different models used for comparison

Models	**Hyperparameters**	**Values**
N-BEATS	Fully connected layers	4
	Lookback	7
	Horizon	1
	Stacks	30
	Neurons per layer	512
	Epochs	500

	Loss function	MAE
	Optimizer	Adam
CNN	Filters	128
	Kernel size	5
	Batch size	128
	Epochs	100
	Loss function	MAE
	Optimizer	Adam
LSTM	Inputs	128
	Activation function	ReLU
	Batch size	128
	Epochs	100
	Loss function	MAE
	Optimizer	Adam
GRU	Inputs	128
	Activation function	ReLU
	Batch size	128
	Epochs	100
	Loss function	MAE
	Optimizer	Adam

2.5Performance measure:

a) Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}}{y_i}\right|$$

b) Mean Absolute Error (MAE)

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

c) Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

where,N is the number of observations in the dataset, $y_i$is the true values of the variable being predicted and $\hat{y}$ is the predicted values of the variable.

3. Results and Discussion

3.1 Descriptive statistics

Table 2 presents the descriptive statistics of the experimental datasets utilized in this study, while Fig. 5 depicts the actual time series plot of the S&P 500 stock price. The graphs clearly indicate that the data exhibit non-stationarity, a finding further confirmed by statistical tests.

Table 2: Descriptive statistics of S&P 500 stock price.

Descriptive Statistics	**Price (U.S. dollars ($) per share)**
Minimum	2237.40
Mean	3484.71
Maximum	4796.56
Standard Deviation	663.27
Coefficient of Variation (%)	19.03
Skewness	0.23
Kurtosis	-1.35
Shapiro-Wilk's test	0.92 (<0.0001)

The value in the parentheses indicates *p*-value

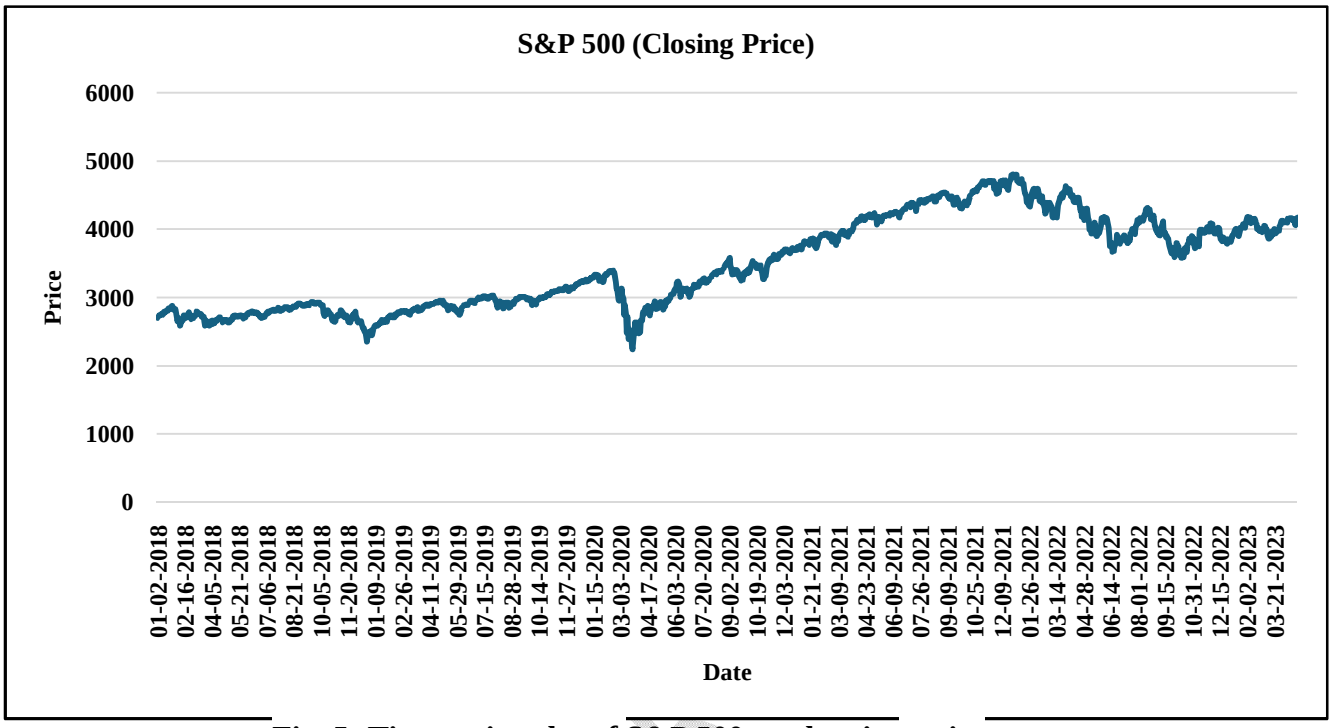

Fig. 5: Time series plot of S&P 500 stock price series

The S&P 500 stock price, ranging from $ 2237.4 to $ 4796.56, demonstrates significant volatility, as reflected in the standard deviation detailed in Table 2. The data is positively skewed and exhibits platykurtic behavior, indicating a non-normal distribution, which is further validated by the Shapiro-Wilk test. The dataset comprises 1340 observations, divided into a training set (80%) and a testing set (20%). The training set, containing 1072 observations, is utilized for model development, while the testing set, with 268 observations, is employed for model validation and post-sample prediction. This partitioning strategy enables a comprehensive analysis of S&P 500 stock pricedynamics, ensuring robust model performance evaluation across both subsets.

3.2 Test for Stationarity

Stationarity is a crucial factor in forecasting models and is assessed in this study using the Augmented Dickey-Fuller (ADF) test (Avinash et al. 2024). The ADF test's null hypothesis suggests that the series is non-stationary or possesses a unit root. However, the results, as shown in Table 3, confirm that the series is stationary.

Table 3: ADF test result of S&P 500 stock price.

Data	**Augmented Dickey-Fuller**		**Remarks**
	Statistic	***p* - value**	
S&P 500 stock price	-1.13	0.69	Stationary

3.3 Performance Evaluation

The performance evaluation of each model on the S&P 500 stock price dataset was conducted using metrics such as MAPE, MAE, and RMSE, as detailed in Table 4. The N-BEATS model consistently outperformed the others, achieving the lowest MAPE of 1.15, MAE of 45.85, and RMSE of 59.42. These results underscore the N-BEATS model's exceptional capability in capturing the underlying patterns and dynamics of the S&P 500 stock price series, resulting in more accurate forecasts.

Table 4: Results obtained by different models on the testing dataset for S&P 500 stock price.

Models	**MAPE (%)**	**MAE**	**RMSE**
CNN	1.61	63.80	80.95
LSTM	1.70	67.69	85.19
GRU	1.66	66.12	83.53
N-BEATS	**1.15**	**45.85**	**59.42**

Table 5: Diebold–Mariano test results of S&P 500 stock price.

Models	**CNN**	**LSTM**	**GRU**	**N-BEATS**
CNN	-	-8.01 (0.82)	-3.38 (0.95)	**5.92** **(0.004)**
LSTM	-8.01 (0.82)	-	9.50 (0.86)	**3.64** **(0.008)**
GRU	-3.38 (0.95)	9.50 (0.86)	-	**5.08** **(0.005)**

N-BEATS	**5.92 (0.004)**	**3.64 (0.008)**	**5.08 (0.005)**	-

Values in the parentheses indicates *p*-value

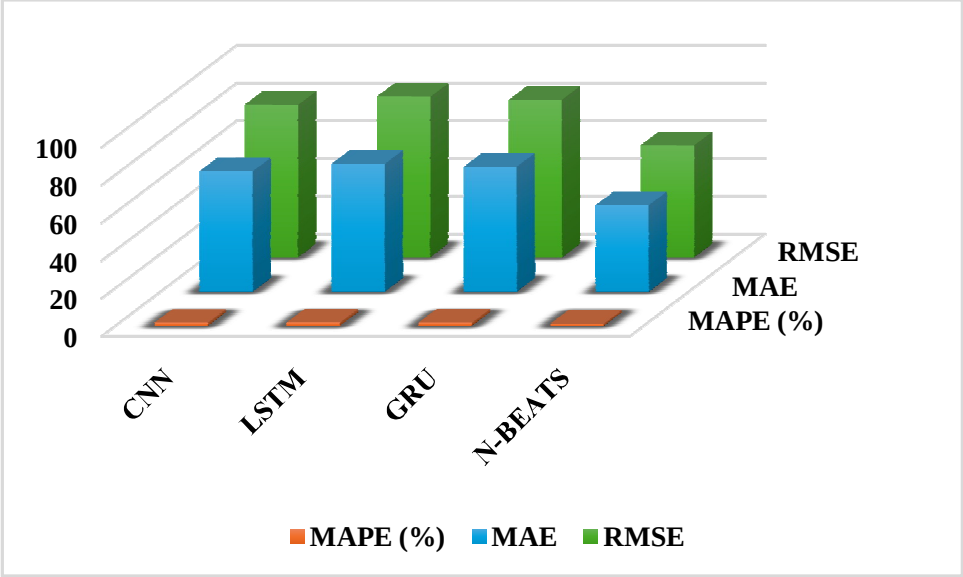

Fig. 6:Bar diagram of evaluation criteria of S&P 500 series on the testing dataset

Traditional models such as ARIMA often fall short in capturing nonlinear patterns, while parametric nonlinear models like GARCH are limited by their restrictive assumptions. To overcome these limitations, researchers have increasingly turned to machine learning (ML) methods. However, ML's predictive accuracy can falter with large datasets due to the necessity for manual feature extraction. As a result, deep learning (DL) architectures, including CNN, LSTM, and GRU, have gained popularity for modeling stock price data. This study introduces the N-BEATS algorithm for forecasting the S&P 500 price series from Google Finance. Comparative analysis reveals that N-BEATS surpasses CNN, LSTM, and GRU in predictive accuracy, achieving the lowest values for key performance metrics like MAPE, MAE, and RMSE, as depicted in Fig. 6. These results highlight N-BEATS' superior effectiveness in capturing trends and patterns within time series data.

N-BEATS' impressive performance is largely due to its innovative architecture, which utilizes stacked blocks of fully connected layers for both backcasting and forecasting. This architecture is further enhanced by the incorporation of residual links and double residual stacking, which significantly improves the model's learning capacity and prediction accuracy. The model's

adaptability and flexibility allow it to effectively manage diverse time series patterns and various data types. Validation through the Diebold-Mariano (DM) test underscores N-BEATS' superior forecasting accuracy when compared to other benchmark models, as detailed in Table 5. This confirms N-BEATS capability to capture complex patterns and its efficacy across different forecasting tasks. While the current N-BEATS model does not include spatiotemporal modeling, future adaptations aimed at integrating such data could enhance its utility, particularly in scenarios involving spatial-temporal dynamics. Overall, this study demonstrates that N-BEATS is a highly effective and advanced tool for time series forecasting, offering significant benefits for both practitioners and researchers in the field.

4. Conclusion

Accurate stock price forecasting is essential for making well-informed investment decisions and maintaining market stability. Traditional approaches like ARIMA often struggle with the intricacies of nonlinear patterns, leading to the adoption of machine learning (ML) techniques. Despite their advantages, ML models encounter difficulties with extensive manual feature extraction, particularly when dealing with large datasets. The development of deep learning (DL) architectures, including CNNs, LSTMs, GRUs, and notably N-BEATS, has marked a significant advancement in stock price prediction. N-BEATS stands out due to its unique architecture, which incorporates stacked blocks, residual connections, and a double residual stacking approach, enabling it to effectively capture complex time series patterns. This innovative framework not only enhances prediction accuracy but also performs exceptionally well in comparative analyses and validation tests, such as the Diebold-Mariano test. While the N-BEATS model is powerful for time series forecasting, it has limitations such as reduced interpretability, high computational cost, and sensitivity to hyperparameters, which may lead to overfitting, especially in smaller datasets. It is primarily designed for univariate forecasting and lacks support for spatiotemporal modeling, limiting its application in more complex scenarios. Future research could focus on integrating spatiotemporal data, developing hybrid models, and expanding the model to multi-variate forecasting. Applying N-BEATS across various financial markets could also test its generalizability and robustness.. Overall, N-BEATS is a robust and advanced tool for stock price forecasting, providing valuable insights for both practitioners and researchers.

Availability of data and material

Data will be available based on the request with the corresponding author.

Code availability

Code will be available on request to the corresponding author.

Declarations

Ethics approval, consent to participate, and consent for publication

The manuscript does not report on or involve the use of any animal or human data and "not applicable" in this section.

Disclaimer (Artificial intelligence)

Option 1:

Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc) and text-to-image generators have been used during writing or editing of manuscripts.

Option 2:

Author(s) hereby declare that generative AI technologies such as Large Language Models, etc have been used during writing or editing of manuscripts. This explanation will include the name, version, model, and source of the generative AI technology and as well as all input prompts provided to the generative AI technology

Details of the AI usage are given below:

1.Author(s) hereby declare that NO generative AI technologies such as Large Language Models (ChatGPT, COPILOT, etc) and text-to-image generators have been used during writing or editing of manuscripts.

2.

3.

Reference

Aslam, M., J.-S. Kim, and J. Jung. 2023. Multi-step ahead wind power forecasting based on dual-attention mechanism. Energy Reports 9 (December 2023):239–251.

Avinash, G., V. Ramasubramanian, M. Ray, R. K. Paul, S. Godara, G. H. H. Nayak, R. R. Kumar, B. Manjunatha, S. Dahiya, and M. A. Iquebal. 2024. Hidden Markov guided Deep Learning models for forecasting highly volatile agricultural commodity prices. Applied Soft Computing 158 (June 2024):111557.

Azevedo, K., L. Quaranta, F. Calefato, and M. Kalinowski. 2024. A Multivocal Literature Review on the Benefits and Limitations of Automated Machine Learning Tools. arXiv preprint arXiv:2401.11366 (2024).

Bakir, H., G. Chniti, and H. Zaher. 2018. E-Commerce Price Forecasting Using LSTM Neural Networks. International Journal of Machine Learning and Computing 8(2):169–174.

Chung Junyoung, Gulcehre Caglar, Cho KyungHyun, and Bengio Yoshua. 2014. Evaluation of datasets. In: NIPS 2014 Workshop on Deep Learning (2014).

Conejo, A. J., M. A. Plazas, R. Espinola, and A. B. Molina. 2005. Day-Ahead Electricity Price Forecasting Using the Wavelet Transform and ARIMA Models. IEEE Transactions on Power Systems 20(2):1035–1042.

Hewamalage, H., C. Bergmeir, and K. Bandara. 2021. Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting 37(1):388–427.

Li, A. W., and G. S. Bastos. 2020. Stock market forecasting using deep learning and technical analysis: a systematic review. IEEE access 8 (2020):185232–185242.

LIU, H. 2024. Time Series Predictive Control in Robotics. EDP Sciences.

Marino, D., K. Amarasinghe, and M. Manic. 2016. Building Energy Load Forecasting using Deep Neural Networks.

Mehtab, S., and J. Sen. 2020. A time series analysis-based stock price prediction using machine learning and deep learning models. International Journal of Business Forecasting and Marketing Intelligence 6(4):272.

Nayak, G. H. H., M. W. Alam, K. N. Singh, G. Avinash, R. R. Kumar, M. Ray, and C. K. Deb. 2024a. Exogenous variable driven deep learning models for improved price forecasting of TOP crops in India. Scientific Reports 14(1):17203.

Nayak, G. H. H., W. Alam, K. N. Singh, G. Avinash, M. Ray, and R. R. Kumar. 2024b. Modelling monthly rainfall of India through transformer-based deep learning architecture. Modeling Earth Systems and Environment 10(3):3119–3136.

Oreshkin, B. N., G. Dudek, P. Pełka, and E. Turkina. 2021. N-BEATS neural network for mid-term electricity load forecasting. Applied Energy 293 (July 2021):116918.

Patarwal, P., A. Dagar, R. Bala, and R. Singh. 2018. Financial Time Series Forecasting Using Deep Learning Network. In 23–33.

Racocha, K. 2020. Impact of Drought on Commodity Market Forecasting (2020).

Sánchez-Reolid, R., F. López de la Rosa, M. T. López, and A. Fernández-Caballero. 2022. One-dimensional convolutional neural networks for low/high arousal classification from electrodermal activity. Biomedical Signal Processing and Control 71 (January 2022):103203.

Sbrana, A., and P. A. Lima de Castro. 2023. N-BEATS Perceiver: A Novel Approach for Robust Cryptocurrency Portfolio Forecasting. Computational Economics (September 22, 2023).

Sezer, O. B., M. U. Gudelek, and A. M. Ozbayoglu. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. Applied soft computing 90 (2020):106181.

Shahvaroughi Farahani, M., and S. H. Razavi Hajiagha. 2021. Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models. Soft computing 25(13):8483–8513.

Singhal V, Mathew J, and Agarwal M. 2022. Fusion of Wavelet Decomposition and N-BEATS for improved Stock Market Forecasting. Research Square (2022).

Tripathi, B., and R. K. Sharma. 2023. Modeling Bitcoin Prices using Signal Processing Methods, Bayesian Optimization, and Deep Neural Networks. Computational Economics 62(4):1919–1945.

Wang, X., C. Li, C. Yi, X. Xu, J. Wang, and Y. Zhang. 2022. EcoForecast: An interpretable data-driven approach for short-term macroeconomic forecasting using N-BEATS neural network. Engineering Applications of Artificial Intelligence 114 (2022):105072.