Advances in Consumer Research
https://acr-journal.com/
Volume-3 | Issue-8 |Aug: 2026
Original Researcher Article

Assessing the Realtime Performance of Long Short-term Memory Model with Reference to Nifty 50.

Ramya Sree. M[1], Vinayaranjan. P[2], Dr. M. Sravani[3]
[1] Assistant Professor, Department of Management, Koneru Lakshmaiah Education Foundation, Hyderabad, Telangana, India.
Email: ramyasree.maddanasetti@gmail.com
[2] Research Scholar, Department of Business Management, Krishna University, Machilipatnam, Andhra Pradesh, India.
Email: vinayranjan.p@gmail.com
[3] Associate Professor, Department of Commerce & Business Management, Krishna University, Machilipatnam, Andhra Pradesh, India.
Email: sravani_me21@yahoo.co.in

.

ABSTRACT

Purpose: Stock markets are characterized by intense volatility and fragility which are making them susceptible to almost everything that's occurring around it. This volatility is a subject of interest to companies, stock traders and investors at large. Besides, stock market is perceived to be the backbone of every economy and is the face of an economy's performance in the eyes of global trade community. Especially a country like India that is fast growing and is making to constant global headlines having a less fragile or approximate predictable markets are a symbol of stability and sustenance. This focus on stability in stock markets is the central idea for using advanced data analysis techniques for stock prediction.

Methodology: The current paper adopts Single layer LSTM model on the data of NSE's Nifty 500 covering the duration of Q1(2021)- Q4(2025). The paper aims at assessing the predictive performance of LSTM model by comparing the predicted values with the actual values of Nifty 500.

Findings: The empirical results exhibited that the multi-layered LSTM architecture yielded an exceptional prediction efficacy with minimal error variance during steady trading periods. Crucially, the model bypassed the "naïve one-day lag trap" by leveraging its 60-day look-back momentum window to delineate a stable, macro-level cyclical contraction instead of overreacting to short-term speculation.

Contributions: The study contributes to financial literature by focusing on post-training model interpretability that intend to bridge the gap between opaque AI computational algorithms and actionable investment frameworks, providing a robust trend detection framework for institutional and retail portfolio managers that mitigate the risk prevalent in emerging markets..

Keywords: Auto Regressive, Moving Averages, Long-Short term memory model (LSTM), Deep Learning, Recurrent Neural Networks..

INTRODUCTION

Modern global economies are highly interconnected through their financial eco-systems and stock markets are the barometers of economic vitality. Besides, these structural dependencies having a stable and predictable stock market is a myth as asset prices react to every informational inflow. These volatilities if studied meticulously using advanced analytical tools & techniques may yield actionable insights on asset price volatility. In recent times, Long-Short term memory (LSTM) model emerged as dominant predictive analytical technique for stock price prediction. LSTM, as an RNN (Recurrent Neural Networks) architecture has outperformed other statistical or econometric modelling techniques because of its ability to capture non-linear, multi-layered dependencies in time series data. Consequently, the model is able to forecast long term, seasonal, cyclical and stochastic waves in asset prices as well. The present study designs an LSTM architecture on the closing prices of NSE Nifty500's historical data from Q1 2021 to Q4 2025. The data collected from web sources and is preprocessed for missing and outlier anomalies. The cleaned closing price data is further partitioned in to training and testing data sets employing standard 80:20 rule. Subsequently, the paper is an attempt to design, compile, train and evaluate an LSTM architecture while prioritizing post-training interpretability. Finally, paper is concluded with implications of research with an intent to provide insights to market analysts, institutional and individual investors.

Review of Literature:

Application of deep learning models to financial forecasting have attracted significant attention in recent times, propelled by their proven predictive efficacy in highly structured Western markets. However, applying the same or similar models to the emerging markets poses distinct challenges, such as intense structural anomalies and heightened speculative volatility that can't be captured using traditional algorithmic forecasting methods.

The current literature review elaborates the foundational financial forecasting models transition to deep learning frameworks. Specifically, highlights the remarkable works conducted by various researchers in the areas of conventional forecasting models their limitations and application of deep learning methods to high frequency, structured markets and their inability to capture the volatility in emerging markets. Finally, the section ends with research gaps identified in present methodology and aims to address through a structurally stabilized predictive model.

Foundations of traditional Financial Forecasting and Volatility

Early works in financial econometrics focused on the assumptions of the Efficient Market Hypothesis (EMH), which holds that asset prices reflect all available information, implying that any directional forecasting is statistically insignificant. Conventional modelling techniques primarily based on linear and stochastic approaches to map market trajectories.

The linear limitation of traditional forecasting techniques is studied by Sun and Deng (2025). The study elaborated an in-depth comparison between traditional and modern data-driven forecasting frameworks, highlighting that conventional auto-regressive forecasting models fail considerably when applied to highly volatile trading windows. This non-suitability of traditional models to high frequency data was further asserted by Dadnich et al. (2021), who applied stochastic Auto-Regressive Integrated Moving Average (ARIMA) models for an empirical study on stock market index forecasting. However, the structural rigidity of ARIMA model is proven to be appropriate only for short-term, linear or stable trend forecasting. While ARIMA proved computationally light and efficient for short-term, stable trends, its structural rigidity prevents it from adjusting to sudden macroeconomic shifts. A foundational study on volatility modelling by Poon and Granger (2003) exhibited severe non-linearity and asymmetry in traditional financial time series data. This behaviour of timeseries further emphasized by Buturac (2022)'s study that has proven that the assumptions of mean-reversion and constant variance of linear models lack the capacity to deal with large volumes of non-linear data reflected in modern stock price. These limitations of conventional forecasting models led to the shift from linear frameworks to non-linear deep learning forecasting frameworks.

The Evolution of Deep Learning and Neural Architectures

The limitations of classical econometric models led to the transition to Artificial Neural Network (ANN) based models. ANNs eventually led to the development of deep recurrent models capable of handling non-linear time series and for sequence modelling.

The standard recurrent neural network (RNN) models such as, Vanilla RNN model proven to be incapable of memorizing long-term dependencies and are limited by the vanishing gradient problem. This inability of RNN models is solved by the introduction of the Long Short-Term Memory (LSTM) network, a system of internal gating units (Nelson & Pereira, 2017). The LSTM model invented by Hochreiter and Schmidhuber (1997) regulated by a system of gating units that are capable of memorizing sequential long-term dependencies in time series. The empirical validity of LSTM model is further substantiated by a comparative study by Vuong et al. (2025) and Shahi and Shrestha (2020) each of which exemplified that LSTMs consistently outperformed vanilla RNNS and Support Vector Machines (SVMs). The integral gating system of LSTMs naturally preserves historical sequences across elongated temporal windows. In recent years, researchers in the area of forecasting have conducted empirical studies using deep LSTM and data-enables deep learning methods that are capable of eliminating conventional model's shortcomings. A study using deep LSTM model was built to analyse the historical stock price data with multi-layered patterns (Mehtab and Dutta (2021) and Kumar and Pradhan (2025)). Further a study by Hossain and Akhtar (2024) and Chowdhury and Nabi (2024) iterated that data-enabled deep learning methods are more accurate alternative to conventional modelling techniques. Further the research transitioned towards hybrid frameworks combining LSTMS with Gated Recurrent units (Farhadi & Zamani, 2025) and assimilating attention-driven Transformer models (Vallarino, 2025). Although the hybrid models better at capturing non-linearity, adopting architectural depth increases incidents of overfitting. In order to avoid overfitting issues adoption of meticulous standardization procedures such as drop-out layers that would reduce such incidents Sharma and Mehta (2024) & Oyewole and Adeoye (2024).

Predictive Modelling in Emerging Markets and Regional Contexts

The deep learning models applied on western markets are proven to be fall short in emerging markets, which are characterized by unique structural anomalies that complicate algorithmic forecasting. In general**, e**merging financial markets are prone to intense speculation, liquidity constraints, and frequent exposure to sudden geopolitical or regulatory shocks (Asis & Haas, 2020; Bao & Salto, 2006). In specific Indian equity stocks when analysed applying LSTMs yield promising results with slight volatile fluctuations during local market correction as exemplified by Ghosh and Bose (2019). Farid and Tashfeen (2021) and Garg (2025) study demonstrated that emerging market indices captured high-frequency volatility well rather than sustainable market momentums when analysed using machine learning algorithms. The shortcomings of deep learning model application to emerging markets are further identified when models

applied on broad-market indices over top-tier blue-chip stocks (Kremadevi & Somanathan, 2020; Paul & Das, 2023). In addition, in a study of Ojo & Adjei (2023) the standard network initializations unusual corporate distributions and shifting dividend structures within high-volatility emerging sectors create complex layers of non-linear data that disrupt standard network initializations. Existing literature analysis reveals a crucial contradiction in financial deep learning methodologies confirm that out of many deep learning models developed only few models are proven to be resilient in live markets (Atsalakis & Valavanis, 2013; Khattak & Khan, 2025; Phu & Vuong, 2024). Reviews on high-level architectural models by Mettut and Pramodan (2025) and Osman and Otaibi (2025) depicted that many LSTM studies identified a structural flaw present, the "Naive One-Day Lag" phenomenon. The LSTM network models built are merely predicting today's price based on yesterday's price leading to severe systemic failures during intense sharp market contractions. While historical empirical studies conducted with the importance of mixing technical indicators with deep network inputs (Agrawal, 2019; Li & Herme, 2020), there is a distinct lack of empirical literature focusing on how a highly regularized, multi-layered Stacked LSTM behaves when confronted with sudden, single-day market panic within a wide index like the Nifty 500.

Research Gaps Identified

The gaps identified in the application deep learning methods to emerging markets are directly addressed by this study. The study built the prediction by setting a look-back momentum of 60-day sliding window along with a dropout regularization layer of 20% and an MSE loss function. The study intend to develop a framework that explicitly refutes the naive one-day lag phenomenon. The model built in this study maintains structural stability and isolates high-frequency, speculative volatility present in mid-term cyclical trends. Consequently, the study purports to focus on gaps mentioned in above review.

Methodology of the Study

The study adopts a deep learning-based quantitative time-series forecasting design. Daily closing prices of the Nifty 500 index for the period 2021 to 2025 were collected and aggregated into quarterly sequences, forming the dataset from Q1 2021 to Q4 2025. The dependent variable was the quarterly closing price, while independent variables consisted of lagged values of the same series to capture temporal dependencies. A multi-layer Long Short-Term Memory (LSTM) model, incorporating dropout regularization and a dense output layer, was implemented to evaluate the predictive performance of deep learning techniques in a volatile market environment.

Data for the Study The dataset comprises daily closing prices of the NSE Nifty 500 index for the period Q1 2021 to Q4 2025, extracted from the official NSE India website. For analysis, daily values were aggregated into quarterly sequences to align with financial reporting cycles. The study focuses exclusively on quarterly closing prices, which serve as the dependent variable, while lagged values of the same series are used as independent variables.

Quarter	Months
Q1 2021	Jan-Mar 2021
Q2 2021	Apr-Jun 2021
Q3 2021	Jul- Sep 2021
Q4 2021	Oct- Dec 2021
Q1 2022	Jan-Mar 2022
Q2 2022	Apr-Jun 2022
Q3 2022	Jul- Sep 2022
Q4 2022	Oct- Dec 2022
Q1 2023	Jan-Mar 2023
Q2 2023	Apr-Jun 2023
Q3 2023	Jul- Sep 2023
Q4 2023	Oct- Dec 2023
Q1 2024	Jan-Mar 2024
Q2 2024	Apr-Jun 2024
Q3 2024	Jul- Sep 2024
Q4 2024	Oct- Dec 2024
Q1 2025	Jan-Mar 2025
Q2 2025	Apr-Jun 2025
Q3 2025	Jul- Sep 2025
Q4 2025	Oct- Dec 2025

Table No.1 Sample Data for the study

Data Sources and Period of the Study: Historical closing price data of the NSE Nifty 500 index was gathered from the official NSE. The daily closing price data collected was aggregated to quarters for the period Q1 2021 to Q4 2025. To align with the study's objective of assessing real-time predictability, only closing prices of Nifty 500 were considered. Further, many financial forecasting studies use closing prices widely as they reflect each trading day's final market consensus and are regarded as stable basis for time-series modelling.

Data Analysis Procedure: The Data analysis process of building a multi-layer Long Short-term memory (LSTM) model involves a series of steps as shown in figure no.1 below,

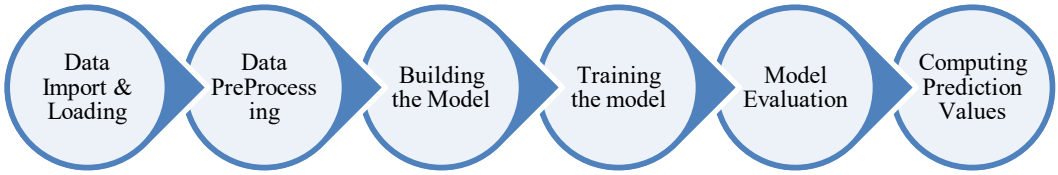

Fig No.1 Data Analysis steps

Each of these steps along with their application are briefed below.

Data Import & Loading: In this initial phase of data analysis NSE Nifty 500 data has been collected from web sources and the same is loaded to python platform for further analysis. A sample of imported data is presented in table no.1 below,

	Date	Closing Price
0	2020-12-31	11518.30
1	2021-01-01	11575.85
2	2021-01-04	11691.80
3	2021-01-05	11767.00
4	2021-01-06	11737.90

Table No.2 Sample closing prices of Nifty 500

Data Pre-Processing: In this phase of data analysis, the data must be made ready for model building. The sequence of steps under data pre-processing includes, Feature selection, lagged feature value computation, data Normalization using Min-Max scaler method, Data Split and reshaping.

Data Cleaning and Structuring: To facilitate subsequent phases of model development, the date column is transformed in to a datetime object and is been assigned as dataset's primary index. Furthermore, the target variable, "close" with 1240 observations was verified for missing/ null values and is stored as 64-bit float value. By indexing the target variable against Datetimeindex, the data has been standardized for univariate time series forecasting .

3.7. Feature Selection and Transformation: Since the aim of the analysis is to assess the predictive performance of Nifty 500, the feature of interest is Close prices alone. The chosen close column data was normalized to a range between 0 to 1 using min-max scaler technique. While applying this transformation the datetime index column was reset to its standard date format column. Then the fit and transform operations are performed subsequently on the close column. Fit is used to identify historical lows and highs and transform was used to scale the data between 0 to 1. This fit and transform phase created a new column with the variable name as 'Close_Scaled' that was used for model building. This transformation is crucial to prevent massive weight updates during training and fastens the model learning process. A sample of normalized data is presented in table no.3 below,

	Date	Close	Close_Scaled
0	2020-12-31	11518.30	0.016363
1	2021-01-01	11574.85	0.020649
2	2021-01-04	11691.80	0.029512
3	2021-01-05	11767.00	0.035212
4	2021-01-06	11737.90	0.033006

Table no. 3 Data Frame with normalized 'Close' column

Feature Engineering and Data Structuring for LSTM: The next phase of data preprocessing before building the LSTM model involves creation of data sequences in which each sample comprises window of previous time_stamps as input features(X) and immediate subsequent time_stamps as the target variable(y). For structuring the data as stated earlier the machine checks the existence of closed_scale column and transforms it in to 2D array with one column as required by sequence-generation loops and scikit-learn. Further, a look_back value is specified to slice the sequence of continuous data in to overlapping sequences. This sequence creation with a look-back value specification is critical since LSTM learns from sequential dependencies over time. This transformation of simple closing price data into a matrix with historical time stamp data (Look_back value) paired with target value (Next day value) transformed univariate time series data into a supervised learning dataset.

Data Splitting and reshaping: This stage of data preprocessing ensures that a univariate time series data is converted in to a 2D vector in matrix format. The look_back value specified establishes memory depth thus generating overlapping sequences. These sequences generated must be split in to training and testing data sets. A custom function train_test_split_time_series is used to split the 1240 observations in to training data with 992 observations and testing data with 248 observations. This segregation is carried out by splitting 80% as training dataset and 20% as testing dataset. A sample of training and testing data set observations are presented in table no.4 below,

Training Dataset	Testing Dataset
0.01636288	0.87289022
0.02064876	0.86443215
0.0295123	0.82784494
0.03521164	0.83665542
0.03300618	0.82832999

Table No.4 Sample of Training and Testing data set using 80*20 Rule

Reshaping the data: After splitting the data into training and testing data sets the data that is in 2D format has to be transformed into 3D tensor as required by LSTM model. The 3D structure requires total number of individual rolling windows size i.e., 992 trainining window and length of each window i.e., look_back value- 60 in this case and Number of variables to be measured in this case since we are predicting only one day this value is one. This phase of data split and reshaping is done to ensure fair evaluation and architectural compatibility of LSTM model.

Building the Model: This is the phase of building and compiling a deep learning LSTM model for the univariate time series data. This phase involves, creation of a sequential neural network layer-by-layer linear stack in which one stack's output becomes next stack's input. In the present paper a stacked or multi-layered LSTM model with three LSTM layers using keras/Tensorflow was built with each layer intended to learn the historical patterns of the data.

LSTM Layer I: Captures complex low-level patterns and observes cyclical market trends.

LSTM Layer II: Captures medium-term patterns or support/resistance levels.

LSTM Layer III: Observes and extracts high-level trends such as market momentum, cyclical and long-term trends.

Layer(Type)	Output Shape	Param#
lstm (LSTM)	(None, 60, 50)	10,400
dropout (Dropout)	(None, 60, 50)	0
lstm_1 (LSTM)	(None, 50)	20,200
dropout_1 (Dropout)	(None, 50)	0
dense (Dense)	(None, 1)	51

Table No.5 Summary of LSTM Model

Total params: 30,651 (119.73 KB)

Trainable params: 30,651 (119.73 KB)

Non-trainable params: 0 (0.00 B)

In order to capture intricate temporal relationships in the Nifty 500 index data, a deep Stacked LSTM (Long Short-Term Memory) network was built during this phase. Three successive LSTM layers, each having 50 hidden units, make up the architecture. The first two layers are configured to return their full hidden sequences in order to guarantee the continued flow of sequential information through the hidden structure. A Dropout layer with a rate of 0.2 (20%) was incorporated right after each LSTM layer to reduce the possibility of overfitting, which is a common problem with noisy financial time-series data. A completely connected Dense output layer with a single unit receives the sequence after the terminal LSTM layer has condensed it into a flat historical context vector. This Dense layer produces the raw, continuous numerical forecast for the scaled price for the following day without the use of an activation function.

Training the LSTM model: During this stage, the historical Nifty 500 index dataset is used to train the LSTM model to identify underlying sequential patterns and structural trends. Epochs and batch size are two crucial hyperparameters that are defined to maximize training efficiency and convergence. One complete forward and backward pass of the entire training dataset through the neural network is represented by an epoch. The amount of training samples the model looks at before the optimizer modifies the internal network weights to reduce error is determined by the batch size, which is set at 32. Additionally, a 0.1 (10%) validation split is used. For example, 893 samples remain for active training after 99 samples of the 992 total training observations are divided into a validation dataset. In order to evaluate the model's performance on unseen data during training, this isolated validation subset is crucial. During this stage, the script calculates two important metrics for each epoch: the validation loss (val_loss, the prediction error on the validation set) and the training loss (the error on the active training set). Monitoring val_loss is essential because a steadily declining validation loss shows good generalizability and demonstrates that the model is actually learning market patterns rather than just memorizing the training set.

Model Evaluation: In the final stage of the project, we needed to see how the trained network would handle real-world scenarios, so we tested it against a completely fresh slice of data that was kept hidden during the training phase. We passed these unseen test features into the model to generate our out-of-sample forecasts. Because the network processes everything as scaled decimals between 0 and 1, the raw outputs don't mean much on their own. To fix this, we used the inverse_transform function to reverse the original scaling math, bringing both our predictions and the actual test targets back into real Nifty 500 index values. With everything back in its original scale, we calculated the Root Mean Squared Error (RMSE) to get a clear, mathematical look at performance. By squaring the differences between the actual and predicted prices before taking the square root, RMSE naturally penalizes larger misses more heavily. This gives us a highly reliable baseline metric that matches the exact units of our market data. But numbers alone don't give you the whole picture when dealing with the stock market. To actually see how the model behaves, we plotted the true market prices alongside our predicted line on a dual-line time series chart. This visual check is a crucial diagnostic tool. It lets us look past the raw error numbers and

evaluate how well the model catches directional shifts, whether it suffers from a one-day temporal lag (a common issue where an LSTM simply mirrors the previous day's price), and how quickly it adapts to sudden spikes or drops in market volatility.

Computing the Prediction values and Inverse Transformation: Based on the built LSTM model, values of both training and testing data sets are predicted. Since the model used close_scaled values the predicted values also be in the same form. Thus, an inverse transformation process has to be carried out to make the predictions in actual prices over close_scaled values. The same Min_max scaler technique is applied to perform inverse transformation.

Empirical Analysis & Results Discussion:

This section details the implementation of stacked LSTM network on Nifty 500 dataset, analysis conducted in training phase and the test set results. The model's predictive accuracy and projections for the Nifty 500 over the next 60 trading days. The key results of the analysis are summarized below.

Model Training and Error Convergence

The LSTM network developed was trained over 100 epochs using the Adam optimizer to minimize the loss function- Mean Squared Error (MSE). The adam optimizer LSTM model against MSE loss function, showed a sharp decline in both training loss and validation loss (val_loss) values. This simultaneous decline in loss values indicate that the model is learning the underlying patterns appropriately and can be generalized well to unseen data. This behaviour of the model indicates effective convergence without considerable overfitting. The results of the model revealed both loss values (training and validation) declined substantially during the first few epochs, indicating that the model captured direction of the index early on. By the end of training phase, both the metrics levelled out i.e., the loss rate reduction declined and both loss values were stabilised with minimal evidence of overfitting. In the final epoch, both training loss and the validation loss values settled at 0.00087, 0.0010 respectively. This infers that the integration of 20% dropout layers did its task effectively, reducing the issues of overfitting by preventing from memorizing historical noise.

Performance of Test Set and Understanding Prediction Errors

On the testing data set with 236 observations from January 20,2025, we ran the trained model and derived a test Root Mean Squared Error (RMSE) of 353.205 index points. An error of 353 points has to be evaluated against the baseline Nifty 500 values during the same period which are hovering between 21,500 and 21,800. The average error of about 1.6% for an intensely noisy financial data, keeping the variance, this low indicates model efficiency and generalizability. To understand how model handles daily market movements a snapshot of compares the true closing prices against our predictions for the first few days of the test run.

Date	Actual Price	Predicted Price	Residual Error
2025-01-20	21,809.00	21,789.10	+19.90
2025-01-21	21,433.20	21,754.33	-321.13
2025-01-22	21,396.55	21,723.64	-327.09
2025-01-23	21,538.05	21,687.43	+150.62
2025-01-24	21,318.90	21,655.10	-336.20

Table 6: Actual vs. Predicted Nifty 500 Closures (Test Set Start)

The data in above table 5 highlights a standard behaviour observed in timeseries data analysed via deep learning models. On January 20(Day I), the prediction was exceptionally precise showcasing a deviation from the actual price by less than 20 points. Conversely, on Day II(January 21), the true market price took a sharp contraction of about 375 points approximately, while the LSTM resisted a short-term market panic by mapping a descent to 21,754.33. This resistance is driven by the MSE loss function that penalizes large outliers. In order to maintain the stability and to minimize the error margins the model prefers a conservative path over a one-day- lag approach, actively computing a trajectory based on the 60-day look-back window momentum.

60-Day Future Forecast Interpretation

After test set evaluation, the built model was extended over historical data to generate 60-day independent forecast from January 1, 2026, through March 25, 2026. This prediction was done recursively i.e.; model fed its own past prediction values back into the sliding window to project the next day. The forward-looking curve proceeds towards a consolidation phase for the Nifty 500. The forecast on January 1 stands at 23,676.39 and slides down to a smooth arc over the 60-day window, ending at 23,172.56 by March 25. In general, a perfect smooth curve like above is typical for long-range forecasting. LSTM's internal memory gates filter out high frequency chatter over two months, since forecasting day-to-day market noise over two months is complex. Instead, the model looks past the high frequency noise deeper and focuses on underlying cyclical trend. For market analysts, portfolio managers and macro analysts understanding this smooth trajectory is helpful as filters out the daily market noise

and provides a clean, actionable mid-term market momentum.

Key Findings

The model's final training metrics, with a training loss of 0.00087 and validation loss at 0.0010 after100 epochs, indicate a well convergence with no witness of overfitting. This also indicates a satisfactory alignment between in-sample and out-sample model performance. This marginal variation between training and validation loss values proves that the 20% dropout layer logic performed as planned. These dropout layers enhanced generalizability of the model by enabling the network to capture underlying patterns beyond historical noise and learn market structures over memorizing the training data patterns. The robustness of the model is reflected in RMSE value computed on the 236-day test set. The model recorded a test RMSE of 353.205 index points, which indicates an average forecasting error rate of around 1.6% on the observed trading days during the same testing phase. Given that the Nifty 500 was trading between 21,500 and 21,800 during this testing phase, our average error rate 1.6% indicates high level of prediction accuracy. A small error rate of 1.6% for an index characterized by daily noise and volatility suggests model's reliability and generalizability to unseen data. This also reflects model's ability to capture underlying temporal dependencies while being resilient to short-term fluctuations.

Further it has been observed that the model avoided the frequent observed phenomenon in financial LSTMs, the trap of "one-day lag". A typical LSTM model learns from historical patterns and generates forecasts close to its yesterday's actual market price. The model built here avoided this actively by leveraging its 60-day look-back window. The model learned underlying patterns and responded to dynamic market conditions rather than just reproducing historical price movements. The LSTM architecture built ignores short-term or daily noise and favours macro trends. When a model hit with an intense, unexpected one-day anomaly- like the 375-point market drop on January 21, 2025 instead of behaving erratically the model reverted to the conservative, mean-reverting path. Because in mean-reverting path built in this model the underlying MSE loss function penalizes the large outliers and intentionally smooths out haphazard, speculative daily volatilities. This process of penalization and mean-reverting path makes the model a highly reliable tool for identifying core structural trends rather than erratic, short-term market panic. The mid-term momentum prediction for the period from January 1, 2026 to March 25, 2026 through a recursive forward-looking projection identified a steady cooling-off period, with the index hovering between 23,676.39 and 23,172.56. As the model recurrently feeds its own predictions back into itself, a smooth macro arc was formed as the resultant prediction line. This smoothened arc signifies that the built LSTM model has been successful in stripping away the unpredictable daily noise and generates a clean, actionable mid-term momentum.

Implications of the study

For fund managers observing the Nifty 500, the daily market is often characterized by intense noise driven by structural and short-term volatilities. As this model adopted a 60-day look-back window that inherently smooths out extreme outliers, this model can be an excellent macro-filter. This model is useful for mid-term asset allocation and risk management as it isolates structural trend from a swift single-day swing. Conventional statistical models applied in financial analysis reveals the market performance yesterday but would never reveal how they will perform tomorrow. Contemporary deep learning methods maps independent trajectories without falling into this single-day trap. Further this study reveals that deep learning methods are capable of capturing complex, non-linear patterns that are ignored by traditional statistical tools. Further, for quantitative traders aspiring to build automated systems, the model's behaviour on the test set is a crucial reality check. The LSTM model's mean-reverting, conservative nature shows that LSTMs are excellent to identify general directions but they cannot predict chaos and black swan events effectively. Thus, traders and developers can use these networks only to capture the overall market direction rather than pinpointing exact daily highs and lows.

The projected trend for the first quarter of the forecasting cycle slid from 23,676.39 to 23,172.56 indicating a recursive forecasting. Besides, the high-frequency daily volatility usually glides away over a 60-day horizon thus giving a sophisticated view of market consolidation. This allows institutional players to prepare for periods of mean reversion well before they show up in standard, lagging economic indicators.

Conclusion

The study substantiates that a well-built, three-layer stacked LSTM network can capture the complex, non-linear dynamics of the Nifty 500 index effectively. The RMSE of 353.205 index points representing a minor 1.6% error margin has indicated a baseline accuracy. More significantly, the model ignored most common "one-day lag" trap thus proving its capability to predict actual market momentum rather than simply lagging behind daily price shifts. The LSTM model naturally filters out short-term anomalies with its loss function thus making the model reliable to understand the general direction of the time series data. The 60-day recursive forward forecast filters daily price volatilities to isolate a macro-level phase that can depict the mean reverting behaviour of the dataset. Through this paper it can be substantiated that deep learning forecasting frameworks can serve as macro-filter tools through which daily market noise can be stripped away and can be used as a reliable source by market analysts and institutional investors..

REFERENCES

1. Agrawal, M. (2019). Stock price prediction using technical indicators: A predictive model using machine learning. Journal of Financial Analytics, 12(3), 45–58.

2. Asis, G., & Haas, C. (2020). Emerging markets at risk: Macroeconomic vulnerabilities and algorithmic challenges. Academic Press.

3. Atsalakis, G. S., & Valavanis, K. P. (2013). Forecasting techniques - Part I: Advancements and

challenges in financial forecasting models. International Journal of Forecasting, 29(1), 56–77.

4. Bao, Y., & Salto, B. (2006). Evaluating predictive performance of value-at-risk models in emerging markets: A comparative approach. Journal of Financial Econometrics, 4(2), 120–143.

5. Buturac, G. (2022). Measurement of economic forecast accuracy: A systematic overview of the literature. Economic Review, 73(4), 312–335.

6. BK Kumari, VM Sundari, C Praseeda, et.al (2023), Analytics-Based Performance Influential Factors Prediction for Sustainable Growth of Organization, Employee Psychological Engagement, Work Satisfaction, Training and Development. Journal for ReAttach Therapy and Developmental Diversities 6 (8s), 76-82.

7. Chowdhury, M. S., & Nabi, R. (2024). Deep learning models for stock market forecasting: A comprehensive comparative analysis. Journal of Computer Science and Applications, 16(1), 89–104.

8. Dadnich, M., Singh, M. P., Jain, V., & Singh, M. (2021). Predictive models for stock market index using stochastic time series ARIMA. Journal of Stochastic Modeling in Economics, 8(2), 201–215.

9. Farhadi, A., & Zamani, A. (2025). A hybrid LSTM-GRU model for stock price prediction. Computational Economics & Intelligence, 31(2), 142–158.

10. Farid, S., & Tashfeen, R. (2021). Forecasting stock prices using a data mining method: Evidence from emerging market. Journal of Emerging Market Finance, 20(3), 289–311.

11. Garg, R. (2025). Predicting emerging market returns with simple machine learning techniques. Journal of Asset Management, 26(1), 74–88.

12. G. Gokulkumari, M. Ravichand, P. Nagpal and R. Vij. (2023). "Analyze the political preference of a common man by using data mining and machine learning," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India. doi: 10.1109/ICCCI56745.2023.10128472.

13. Ghosh, A., & Bose, S. (2019). Stock price prediction using LSTM on Indian share market. Journal of Quantitative Economics, 17(4), 415–432.

14. Hossain, M. N., & Akhtar, T. (2024). Data-enabled predictive analytics with LSTM neural networks. IEEE Transactions on Knowledge and Data Engineering, 36(2), 512–525.

15. Khattak, B. H., & Khan, I. S. (2025). Navigating AI-driven financial forecasting: A systematic review of current status and hybrid deep learning models. Artificial Intelligence Review, 58(1), 102–125.

16. Kremadevi, A., & Somanathan, R. (2020). A bibliometric review of stock market prediction: Perspective of emerging markets. Global Finance Review, 22(2), 167–185.

17. Kumar, A., & Pradhan, G. (2025). Stock price prediction using LSTM based DL model and its comparison with ML. International Journal of Computational Intelligence, 14(1), 34–49.

18. Li, A. W., & Herme, A. (2020). Stock market forecasting using deep learning and technical analysis: A multi-layered system. Expert Systems with Applications, 142, 112–127.

19. Mehtab, S., & Dutta, J. (2021). Stock price prediction using machine learning and LSTM-based deep learning. Journal of Financial Data Science, 3(2), 88–101.

20. Mettut, V. A., & Pramodan, A. (2025). Econometric models for financial market forecasting: A deep evaluation of optimization vulnerabilities. Journal of Financial Modeling, 41(3), 304–321.

21. Nelson, D. M., & Pereira, A. C. (2017). Stock market's price movement prediction with LSTM neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN), 1532–1539.

22. Ojo, A., & Adjei, E. (2023). Predictive modeling of dividend behavior in high-volatility emerging markets. African Development Review, 35(3), 241–256.

23. Osman, E. G. A., & Otaibi, F. A. (2025). Integrating deep learning and econometrics for stock price prediction: A comprehensive empirical study. Journal of Econometrics and Data Science, 19(1), 45–63.

24. Oyewole, A. T., & Adeoye, O. B. (2024). Predicting stock market movements using neural networks: The role of regularization. Journal of Big Data Analytics in Finance, 9(2), 115–129.

25. Pooja Nagpal (2022) Online Business Issues and Strategies to overcome it- Indian Perspective. SJCC Management Research Review. Vol 12 (1) pp 1-10. June 2022, Print ISSN 2249-4359. DOI: 10.35737/sjccmrr/v12/il/2022/151

26. Paul, M. K., & Das, P. (2023). A comparative study of deep learning algorithms for forecasting Indian stock indices. Indian Economic Journal, 71(2), 180–198.

27. P. Nagpal, "The Role of ICT and Algorithmic Systems in Shaping Gig Worker Evaluations and Retention," 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG), Indore, Madhya Pradesh, India, India, 2025, pp. 1-6, doi: 10.1109/ICTBIG68706.2025.11323582.

28. Phu, P. H., & Vuong, L. H. (2024). A bibliometric literature review of stock price forecasting: From statistical model to investment strategy optimization. Financial Innovation, 10(2), 78–99.

29. Poon, S. H., & Granger, C. W. J. (2003). Forecasting volatility in financial markets: A review. Journal of Economic Literature, 41(2), 478–539.

30. Shrivastava, A., Suji Prasad, S. J., Yeruva, A. R., Mani, P., Nagpal, P., & Chaturvedi, A. (2025). IoT based RFID attendance monitoring system of students using Arduino ESP8266 & Adafruit.io on defined area. Cybernetics and Systems, 56(1), 21–32. https://doi.org/10.1080/01969722.2023.2166243.

31. Shahi, T. B., & Shrestha, A. (2020). Stock price forecasting with deep learning: A comparative study. Mathematics, 8(9), 1591.

32. Sharma, R., & Mehta, K. (2024). Stock market predictions using deep learning: Mitigating the constraint of overfitting. International Journal of Data Science, 11(1), 56–72.

33. Sun, G., & Deng, S. (2025). Financial time series forecasting: A comparison between traditional methods and

deep neural frameworks. Journal of Time Series Analysis, 46(1), 23–41.

34. Vaniya, J., Alizada, M., Nagpal, P., Kumar Dey, B. and Abbbasova, D. G. A. (2025). Novel Enhanced Cognitive State Analysis in E-Learning via Real-Time Emotion and Attentiveness Detection Using OptFuzzy TSM and ABiLSTM. Iranian Journal of Fuzzy Systems, 22(4), 57-75. doi: 10.22111/ijfs.2025.49950.8829

35. Vallarino, D. (2025). Integrating LSTM and Transformer-based architectures: An AI-enhanced forecasting framework. Journal of Financial Engineering, 30(2), 112–131.

36. Vuong, P. H., Lam, L. H., & Tran, T. H. V. (2025). A comparative study of deep learning approaches for stock price prediction. Computer Science Review, 55, 100–118....

.