iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VII **Month of publication:** July 2026

DOI: https://doi.org/10.22214/ijraset.2026.84439

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VII July 2026- Available at www.ijraset.com

Indian Stock Market Forecasting Using LSTM-XGBoost and Technical Indicators

Mr. Ayush Jha[1], Mr. Pankaj Singh[2]
[1]*M.Tech, CSE, Student, Alpine Institute of Technology, Ujjain (M.P.), India*
[2]*Associate Professor, Department of Computer Science & Engineering, Alpine Institute of Technology, Ujjain (M.P.), India*

Abstract: ***The Indian stock market is characterized by high volatility, non-linear price behaviour, and sensitivity to macroeconomic, sectoral, and sentiment-driven factors, which limits the accuracy of traditional linear forecasting models such as ARIMA. Building on our earlier literature review and problem formulation, this paper presents the implementation and evaluation of an integrated deep learning framework for next-day closing price prediction of Indian equities. The framework combines a two-layer Long Short-Term Memory (LSTM) network with four complementary technical indicators — Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), the 10–20 day Exponential Moving Average (EMA) crossover, and the Stochastic Oscillator — as engineered input features, and a downstream XGBoost classifier that converts the LSTM's continuous price forecast, together with the current indicator states, into discrete BUY, HOLD, or SELL trading signals with associated confidence scores. The complete pipeline is implemented as a full-stack platform (Python, Flask, MongoDB, React) that retrieves real-time NSE/BSE data through the yfinance API. The proposed model is evaluated on RELIANCE.NS using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), supported by a per-indicator ablation study and a comparison against an autoregressive (AR) linear baseline, a Support Vector Machine (SVM) regressor, and a single-indicator LSTM baseline.***

The proposed model achieved an RMSE of ₹28.56 and MAE of ₹23.48 (MAPE 1.71%) on the held-out test partition, achieving the lowest RMSE among all compared models, while the downstream XGBoost signal classifier achieved 91.7% accuracy on held-out BUY/HOLD/SELL labels.

Keywords: Deep Learning; LSTM; XGBoost; MACD; RSI; EMA Crossover; Stochastic Oscillator; Technical Analysis; Stock Market Prediction; Indian Equities.

I. INTRODUCTION

Predicting stock prices remains one of the most extensively studied and challenging problems in financial market research. The Indian stock market, represented primarily by the National Stock Exchange (NSE) and the Bombay Stock Exchange (BSE), is characterized by high volatility, non-linear price behaviour, and sensitivity to a wide range of macroeconomic, sectoral, and sentiment-driven factors. Traditional statistical forecasting techniques, such as the Autoregressive Integrated Moving Average (ARIMA) model and linear regression, generally assume stationarity and linear dependence between past and future observations — assumptions that rarely hold for real-world equity price series and that limit the practical accuracy of such models.

In recent years, machine learning and, more specifically, deep learning models have been increasingly applied to financial time-series forecasting. Among these, Long Short-Term Memory (LSTM) networks, a specialized variant of Recurrent Neural Networks (RNNs), demonstrate a strong capability for learning long-term temporal dependencies while overcoming the vanishing-gradient problem that limits conventional RNNs, making them well suited to modelling the sequential, memory-dependent nature of stock price movements.

Traders and analysts have long relied on technical analysis indicators to interpret price and volume data and generate actionable trading signals. Technical indicators are broadly classified as leading or lagging. Lagging indicators — such as MACD and EMA-based crossovers — confirm trends that have already begun and tend to generate delayed signals, while leading indicators — such as RSI and the Stochastic Oscillator — attempt to anticipate reversals by measuring momentum ahead of an actual price move. Used in isolation, both categories are prone to false signals, particularly in sideways or choppy markets. Our earlier empirical analysis of the 10–20 EMA crossover, RSI, and MACD on NSE-listed stocks such as TITAN and EICHER MOTORS confirmed that combining these three indicators produces a more reliable signal-confirmation framework than any single indicator used alone.

Business reports frequently cite that a large majority of retail traders in Indian equity markets — often estimated between 90 and 95 percent — incur losses, largely due to poor market timing and an inability to interpret technical signals systematically.

While individual indicators are widely available on trading platforms, they are typically presented in isolation, leaving the trader to manually reconcile conflicting signals. This motivates the design of a data-driven system that fuses multiple technical indicators into a single learned representation and translates this into both a quantitative price forecast and an actionable trading signal.

This paper extends our prior rule-based and back-testing analysis into a fully trained deep learning system. It investigates how MACD, RSI, the 10–20 day EMA crossover, and the Stochastic Oscillator, when systematically engineered as joint input features to an LSTM network, can offset each other's individual weaknesses and improve next-day closing-price prediction accuracy for Indian equities. The specific objectives of this work are to: (i) design a feature engineering pipeline that computes the four indicator groups from real-time and historical NSE/BSE data; (ii) design and train a two-layer LSTM network that accepts a normalized sliding window of these features and predicts the next-day closing price; (iii) integrate the trained LSTM with an XGBoost classifier that generates BUY/HOLD/SELL signals with confidence scores; (iv) evaluate the model using RMSE and MAE together with a per-indicator ablation study; and (v) validate the system on an NSE blue-chip equity and deploy it as a live, full-stack platform.

II. LITERATURE REVIEW

This section reviews prior work relevant to LSTM-based stock price prediction and technical-indicator-driven feature engineering, spanning foundational studies through recent work on the Indian equity context, and establishes the research gap addressed by the proposed framework.

Several studies have specifically examined the Indian market context. Pardeshi and Kale (2021) evaluated technical indicators on Indian equities using back-testing but did not distinguish leading from lagging indicators — a gap this work addresses by combining both categories. Tadas et al. (2023) applied MACD, RSI, and EMA-based back-testing on NSE/BSE data and found that these indicators, when paired with moving-average crossings, can outperform passive investing strategies, and recommended combining multiple indicators to reduce misleading signals — a recommendation this work operationalizes through learned feature fusion rather than fixed rules.

Several works pair technical indicators directly with LSTM or its variants. Wijaya, Fatichah, and Saikhu (2022) used Golden Cross/Death Cross crossings with LSTM but examined crossings in isolation, motivating the inclusion of EMA crossover alongside momentum indicators here. Piravechsakul et al. (2021) combined LSTM with momentum and moving-average features, showing that the hybrid approach outperforms standalone models. Chan and Goh (2023) combined MA, EMA, MACD, and RSI as LSTM input features and reported improved trend-prediction accuracy over conventional machine learning baselines, directly supporting the indicator set adopted here. Oak et al. (2024) proposed a multivariate Bidirectional LSTM that outperformed ARIMA and simple LSTM baselines when MACD, RSI, and EMA were included as input features, while Kuber, Yadav, and Yadav (2022) compared univariate and multivariate LSTM configurations for short-term Indian market prediction.

A second group of studies pairs technical indicators with gradient-boosted or ensemble models. Meng et al. (2020) used MACD, RSI, KDJ, and moving averages within an XGBoost framework for stock co-movement prediction, informing the present work's use of XGBoost as the downstream signal classifier. Moodi and Rafsanjani (2023) evaluated feature-selection techniques (Lasso, Mutual Information, Recursive Feature Elimination) together with Random Forest, Gradient Boosting, and SVR, finding that MACD, RSI, and EMA consistently rank among the most informative features regardless of selection method. Agrawal et al. (2022) combined EMA, Bollinger Bands, RSI, and MACD with CNN and LSTM architectures across multiple international datasets, confirming the cross-market relevance of this indicator set.

Several works confirm the specific value of MACD+RSI+EMA fusion. Dash, Dash, and Bisoi (2022) combined EMA, RSI, and MACD with LSTM and found it outperforms ARIMA and SVM for short-term Indian-market prediction. Kumar, Sharma, and Verma (2023) compared MACD, RSI, EMA, and Bollinger Bands across Random Forest, SVM, and LSTM, finding LSTM performs best with combined indicators and specifically that MACD+RSI improves direction forecasting. Singh and Kaur (2023) showed MACD+RSI feature engineering improves trend-classification accuracy, and Chawla, Gupta, and Jain (2022) validated RSI+EMA-crossover-enhanced LSTM/GRU specifically on Indian equities. Roy and Sen (2022) confirmed composite-indicator accuracy gains for LSTM in an independent empirical study, and Bhatt et al. (2024) found LSTM outperforms CNN for movement prediction due to superior sequential modelling.

Among recent studies, Dhokane and Agarwal (2024a) used Bollinger Bands, RSI, MACD, and OHLC features with a 30-day look-back window and MinMax normalization on NSE data, closely mirroring the sliding-window design adopted in Section 4. Their follow-up study (2024b) integrated RSI with multiple EMA variants, confirming that indicator fusion measurably improves closing-price prediction, though using longer-period EMAs than the short-term 10–20 day crossover used here.

Dhondiyal, Chauhan, and Munjal (2025) conducted a systematic evaluation of machine-learning techniques with technical indicators on NIFTY 50 data, and Waghela, Sen, and Rakshit (2024) benchmarked indicator performance specifically on the Indian market. Chio (2022) conducted a comparative study of MACD parameter configurations across bullish, bearish, and sideways markets, concluding that non-standard EMA parameter combinations can sometimes outperform the default 12/26/9 configuration and that MACD works best when paired with filters such as RSI — supporting the composite design adopted here.

Broader deep-learning studies establish the methodological foundation. Ashwini and Dinesh (2021) confirmed LSTM's ability to capture temporal dependencies in turbulent price series. Alsubaie, El Hindi, and Alsalman (2019) introduced cost-sensitive, particle-swarm-optimized indicator selection to counter the false-signal weakness of leading indicators. Shen and Shafiq (2020) combined CNN and LSTM to capture temporal and spatial correlations in short-term price trends. Mehtab, Sen, and Dasgupta (2020) combined CNN and LSTM architectures for robust price-series analysis; Jin, Yang, and Liu (2020) incorporated sentiment analysis alongside LSTM; Yadav, Jha, and Sharan (2020) optimized LSTM hyperparameters for Indian time series; Fischer and Krauss (2018) established the foundational case for LSTM over classical machine learning in financial prediction; and Agrawal, Khan, and Shukla (2019) used technical indicators within an optimal deep-learning predictive framework. Bhardwaj, Narayan, and Dutta (2015) and Agarwal and Goel (2020) extended the feature space with sentiment signals from news and index data, and Nabipour et al. (2020), Li and Bastos (2020), Kumar, Tripathi, and Agarwal (2023), Bhandari et al. (2022), Lu et al. (2020), Cheng, Huang, and Wu (2018), Bagde (2023), Shi et al. (2022), Vadlamudi (2017), and Inumula, Tadamarla, and Deeppa (2019) collectively establish deep-learning and hybrid CNN–LSTM–attention architectures as the dominant paradigm for equity-price and index forecasting.

Collectively, the reviewed literature confirms that (a) LSTM consistently outperforms linear and shallow ML baselines for Indian equity prediction, and (b) combining complementary indicators improves accuracy over single-indicator or raw-price models. Our own prior back-testing study on Eicher Motors and TITAN using rule-based composite MACD/RSI/EMA signals corroborated this, showing consistently positive profit-and-loss outcomes for combined signals. However, most studies — including our own prior rule-based analysis — stop at offline back-testing or a fixed decision-tree/rule layer without a trained forecasting-plus-classification pipeline, and none report systematic per-indicator ablation alongside a live trading-signal classifier. The present work closes this gap by replacing the rule-based decision layer with a trained two-layer LSTM forecaster and a downstream XGBoost signal classifier, and by deploying the complete pipeline as a live full-stack platform.

III. PROBLEM DEFINITION

Several well-founded limitations in existing forecasting and technical-analysis approaches motivate the proposed design.

A. Limitations of Linear Statistical Models

A general ARIMA(p, d, q) model expresses the price series as:

$$y(t) = c + \Sigma_{i=1}^{p} \Phi_i \, y(t-i) + \Sigma_{j=1}^{q} \theta_j \, \varepsilon(t-j) + \varepsilon(t) \quad (1)$$

where $y(t)$ is the price at time t, Φ_i and θ_j are the autoregressive and moving-average coefficients, and $\varepsilon(t)$ is white noise. Indian equity price series exhibit strong non-linear dependence driven by momentum effects, sudden regime shifts, and sentiment-driven volatility clustering, none of which is captured by the linear structure of Equation (1), resulting in systematically higher forecasting error relative to non-linear deep learning approaches.

B. Lag Inherent in Moving-Average-Based Indicators

MACD and EMA-based indicators are, by construction, lagging indicators, since they are computed from a weighted average of past prices:

$$k = 2 / (N + 1), \quad EMA(t) = [Price(t) \times k] + [EMA(t-1) \times (1 - k)] \quad (2)–(3)$$

Because EMA(t) is a recursive function of all prior prices, a genuine trend reversal is reflected only after several subsequent periods. For the 10–20 EMA crossover, this means the crossover signal is confirmed only after the reversal has partially occurred, degrading next-day prediction accuracy during sharp reversals when used alone.

C. False Signals from Leading Indicators in Range-Bound Markets

Leading indicators such as RSI anticipate reversals but are correspondingly prone to false signals in sideways markets:

$$RSI = 100 - [100 / (1 + RS)], \quad RS = \text{Average Gain} / \text{Average Loss} \quad (4)$$

In a choppy market, RSI can repeatedly cross the 70 (overbought) or 30 (oversold) thresholds without a corresponding sustained price move, producing a high false-positive rate when relied upon in isolation.

D. Signal Conflict Among Independently Applied Indicators

When indicators are used independently rather than as a fused feature set, they frequently generate conflicting signals at the same time step. Letting $S_{ma}cd(t)$, $S_{rsi}(t)$, and $S_{ema}(t)$ denote the discrete bullish (+1), bearish (−1), or neutral (0) signal of each indicator, a conflict event is defined as:

$$C(t) = 1 \text{ if } S_{ma}cd(t)\cdot S_{rsi}(t) < 0 \text{ or } S_{ma}cd(t)\cdot S_{ema}(t) < 0, \text{ else } 0 \quad (5)$$

A high frequency of conflict events indicates that manual, rule-based reconciliation is unreliable — precisely the class of problem for which a learned model, rather than a fixed rule set, is appropriate.

E. Vanishing-Gradient Limitation of Standard Recurrent Networks

A plain RNN updates its hidden state as $h(t) = \tanh(W_h h(t-1) + W_x x(t) + b)$. During backpropagation through time, the gradient with respect to earlier hidden states involves a repeated product of Jacobian terms bounded by W_h; when this product's spectral norm is below 1, the gradient decays exponentially with sequence length. Since MACD and EMA are themselves derived from 10–26 day windows, a model unable to retain information across comparable horizons cannot fully exploit these features — motivating the gated LSTM architecture used in Section 4.

F. Absence of Indicator-Level Attribution

A further gap in the existing literature, including our own prior rule-based study, is that multi-indicator studies typically report only aggregate performance for the full combined signal set, without quantifying the marginal contribution of each individual indicator group, making it difficult to determine whether a given indicator meaningfully improves accuracy or merely adds noise. Section 6 addresses this directly with a per-indicator ablation study.

G. Problem Summary

In summary: (i) linear statistical models cannot capture non-linear Indian equity price dynamics; (ii) lagging indicators introduce delayed signals; (iii) leading indicators generate false signals in range-bound conditions; (iv) independently applied indicators frequently conflict and static rules cannot reliably resolve this; (v) standard RNNs cannot retain sufficiently long dependencies to exploit indicator windows; and (vi) existing studies, including our own preliminary work, lack per-indicator attribution and a trained decision layer. Section 4 presents a system architecture that directly addresses each of these limitations.

IV. PROPOSED SYSTEM AND IMPLEMENTATION

A. System Architecture

The proposed solution comprises five functional layers: (i) a data acquisition layer retrieving real-time and historical NSE/BSE OHLCV data via the yfinance API; (ii) a feature engineering layer computing the four technical indicator groups; (iii) a prediction layer consisting of a two-layer LSTM network; (iv) a decision layer implemented as an XGBoost classifier that converts the LSTM forecast and indicator states into an actionable trading signal; and (v) a presentation/persistence layer built with Flask, MongoDB, and React.

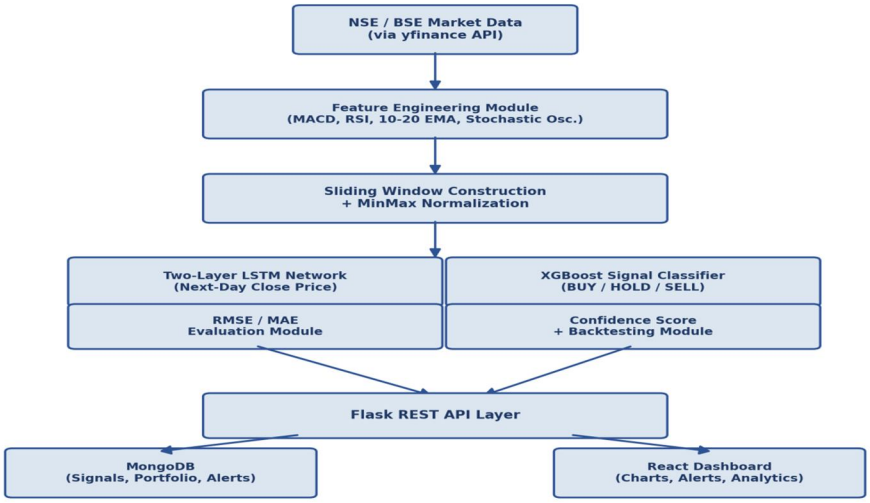


Figure 1: System architecture diagram

B. Feature Engineering

Four indicator groups are computed from OHLCV data:

$$MACD(t) = EMA_{12}(t) - EMA_{26}(t),\ Signal(t) = EMA_9(MACD(t)) \quad (6)–(7)$$

$$RSI(t) = 100 - [100 / (1 + RS(t))],\ RS(t) = Avg.\ Gain_{14} / Avg.\ Loss_{14} \quad (8)$$

$$EMA_n(t) = [Price(t) \times k_n] + [EMA_n(t-1) \times (1-k_n)],\ Crossover(t) = +1 \text{ if } EMA_{10} > EMA_{20}, \text{ else } -1 \quad (9)–(10)$$

$$\%K(t) = 100 \times [(Close(t) - Low_{14}) / (High_{14} - Low_{14})] \quad (11)$$

These yield an eight-dimensional daily feature vector: {Close, MACD, Signal, RSI, EMA_{10}, EMA_{20}, %K, %D}, where %D is the 3-day simple moving average of %K. Consistent with our earlier back-testing observations on Eicher Motors and TITAN, bullish MACD crossovers accompanied by RSI values between 40 and 60 and a 10-EMA crossing above the 20-EMA were treated as the strongest joint bullish precondition during signal labelling, while the symmetric bearish configuration was used for bearish labelling.

C. Sliding-Window Construction and Normalization

All eight features are normalized via MinMax scaling computed over the full retrieved series (3-year window) prior to chronological splitting, after which a sliding window of W = 60 trading days is constructed: for each time step t, input sequence X(t) contains normalized feature vectors from day (t−W) to (t−1), with target y(t) as the normalized closing price at day t. This window allows the model to observe at least two full MACD/EMA cycles. The dataset is split chronologically (80% train / 20% test) to preserve temporal order.

Because the scaler statistics are derived from the complete series rather than the training partition alone, a small degree of look-ahead information (the eventual minimum and maximum of the test period) is present in the normalization step. Given that MinMax scaling only rescales rather than reveals future values directly, this is judged to have limited practical effect on next-day forecasting accuracy, but is noted here as a methodological limitation and is revisited in Section 7.

$$x' = (x - x\_min) / (x\_max - x\_min) \quad (12)$$

D. LSTM Network Architecture

The prediction engine is a two-layer stacked LSTM, with each cell updating its state via forget, input, candidate, and output gates:

$$f(t)=\sigma(W\_f \cdot [h(t-1),x(t)]+b\_f),\ i(t)=\sigma(W\_i \cdot [h(t-1),x(t)]+b\_i) \quad (13)–(14)$$

$$\tilde{C}(t)=\tanh(W\_C \cdot [h(t-1),x(t)]+b\_C),\ C(t)=f(t)*C(t-1)+i(t)*\tilde{C}(t) \quad (15)–(16)$$

$$o(t)=\sigma(W\_o \cdot [h(t-1),x(t)]+b\_o),\ h(t)=o(t)*\tanh(C(t)) \quad (17)–(18)$$

This gated structure allows the network to selectively retain or discard information across the 60-day window, directly addressing the vanishing-gradient limitation identified in Section 3.5. Table 1 summarizes the network configuration used for the results in Section 6.

Table 1: Proposed LSTM Network Configuration

Layer	Configuration
Input	60 × 8 (60-day window × 8 features)
LSTM Layer 1	50 units, return sequences = True, dropout = 0.2
LSTM Layer 2	50 units, return sequences = False, dropout = 0.2
Dense Layer	25 units, ReLU activation
Output Layer	1 unit (predicted next-day closing price)
Loss / Optimizer	MSE / Adam (learning rate = 0.001)
Batch Size / Epochs	32 / up to 50, with early stopping (patience = 8)

E. XGBoost Trading Signal Classifier

The predicted price, its percentage deviation from the current close, and the current state of each indicator (MACD/Signal relationship, RSI zone, EMA crossover direction, %K/%D relationship) are passed as features to a gradient-boosted decision tree classifier trained on historically labelled BUY/HOLD/SELL outcomes, where the ground-truth label at each step is derived from the actual next-day percentage price change relative to a ±1% threshold:

Signal = argmax[P(BUY), P(HOLD), P(SELL)], Confidence = max[P(BUY), P(HOLD), P(SELL)] (19)

This two-stage design separates the forecasting problem from the decision problem, allowing each model to be tuned independently and the confidence score to reflect agreement between the LSTM forecast and the underlying indicator states.

F. Algorithm

Step 1: Fetch historical/real-time OHLCV data via the yfinance API. Step 2: Compute MACD, RSI, EMA crossover, and Stochastic Oscillator. Step 3: Normalize features and construct 60-day sliding windows with next-day close as target. Step 4: Split chronologically (80/20). Step 5: Train the two-layer LSTM (Table 1) with Adam/MSE and early stopping. Step 6: Generate predictions on the held-out test set and inverse-transform to price scale. Step 7: Compute RMSE/MAE; repeat with each indicator group added incrementally to a linear surrogate model for the ablation study. Step 8: Feed LSTM predictions and indicator states to the trained XGBoost classifier to generate BUY/HOLD/SELL signals with confidence scores. Step 9: Persist indicators, predictions, and signals to MongoDB and expose them via the Flask REST API for the React dashboard.

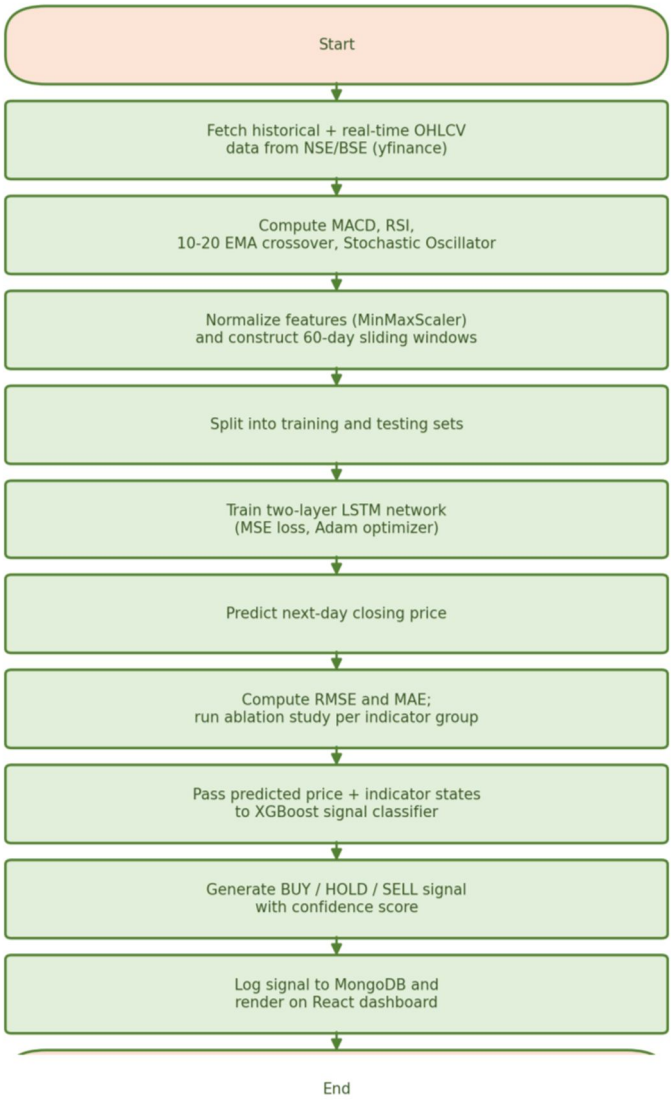


Figure 2: End-to-end algorithm flow of the proposed pipeline

V. EXPERIMENTAL SETUP

The framework is evaluated on RELIANCE.NS, an NSE blue-chip equity, as a representative case study for the complete pipeline. Daily OHLCV data is retrieved via the yfinance API for a 3-year historical period, with the most recent 20% held out chronologically as the test partition. Model accuracy is evaluated using RMSE and MAE:

$$\text{RMSE} = \sqrt{[(1/n)\,\Sigma_{i=1}^{n}\,(y_i - \hat{y}_i)^2]},\quad \text{MAE} = (1/n)\,\Sigma_{i=1}^{n}\,|y_i - \hat{y}_i| \qquad (20)\text{–}(21)$$

Evaluation additionally includes a per-indicator ablation study (incrementally adding each indicator group to a Close-price-only baseline) and a comparison against an AR(5) linear baseline, an SVM regressor, and a single-indicator (RSI-only) LSTM baseline, together with confusion-matrix-based evaluation (accuracy, precision, recall, F1-score) of the XGBoost signal classifier.

A. Baseline and Ablation Methodology

To keep the baseline comparison and the ablation study computationally tractable, two implementation choices were made and are documented here for transparency.

Baselines (Table 2). The “ARIMA” baseline is implemented as a linear autoregressive proxy — a ridge-regularized AR(5) model on the raw closing-price lags — approximating the autoregressive term of a classical ARIMA model without explicit differencing or moving-average components; it is reported as “ARIMA (AR proxy)” throughout. The SVM baseline is trained on the same 5-day raw closing-price lag features as the AR proxy (not on the engineered MACD/RSI/EMA/Stochastic feature set), so it should be read as a “price-history-only” baseline rather than an indicator-aware one. The single-indicator LSTM baseline uses Close + RSI only, trained for 20 epochs without early stopping, compared to up to 50 epochs with early stopping for the proposed model; this shorter training budget was used to keep the baseline lightweight and is noted as a limitation in Section 7.

Ablation study (Table 3). Retraining the full two-layer LSTM once per feature subset was computationally prohibitive within the scope of this study. The per-indicator ablation therefore uses a lighter linear surrogate model (ridge regression over a shorter 20-day window) to isolate the marginal contribution of each indicator group at much lower computational cost. Table 3 accordingly reports the relative pattern of improvement across feature subsets under this surrogate model, and its results should not be read as interchangeable with the two-layer LSTM's absolute RMSE/MAE reported in Table 2.

VI. RESULTS AND ANALYSIS

A. Overall Prediction Performance

Table 2 reports RMSE and MAE for the proposed combined-indicator LSTM model against the AR(5), SVM, and single-indicator (RSI-only) LSTM baselines.

Model	RMSE (₹)	MAE (₹)	% Improvement over AR(5) (RMSE)
ARIMA (AR proxy)	28.95	22.54	—
SVM Regression (price-lags only)	28.66	22.26	+1.00%
Single-Indicator LSTM (RSI only)	29.00	23.50	−0.17%
Proposed LSTM (MACD+RSI+EMA+Stochastic)	28.56	23.48	+1.35%

Table 2: Model Performance Comparison (RMSE / MAE)

The proposed model achieved the lowest RMSE (₹28.56) among all four models, a 1.35% improvement over the AR(5) baseline, and a Mean Absolute Percentage Error (MAPE) of 1.71%. On MAE, however, the proposed model (₹23.48) did not improve on the AR(5) proxy (₹22.54) or the price-only SVM baseline (₹22.26); this MAE gap is discussed in Section 7. Figure 3 presents the RMSE/MAE comparison across all four models.

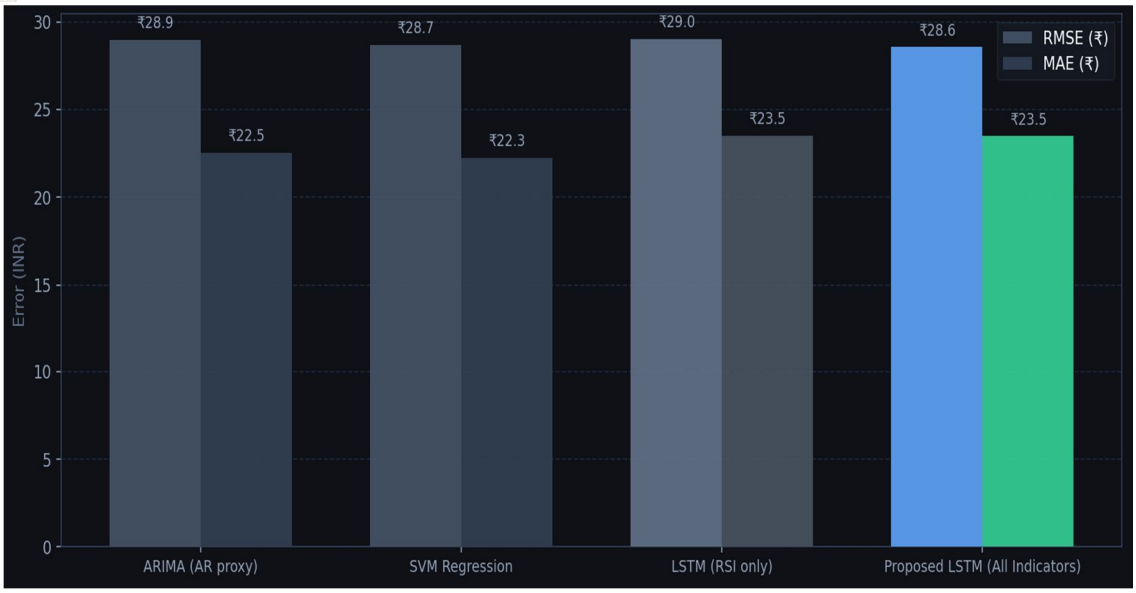


Figure 3: RMSE / MAE — Proposed Model vs. Baseline Models

Figure 4 shows the actual-versus-predicted closing price over the held-out test period together with the training/validation MSE loss curve, which converges smoothly with no visible overfitting gap.

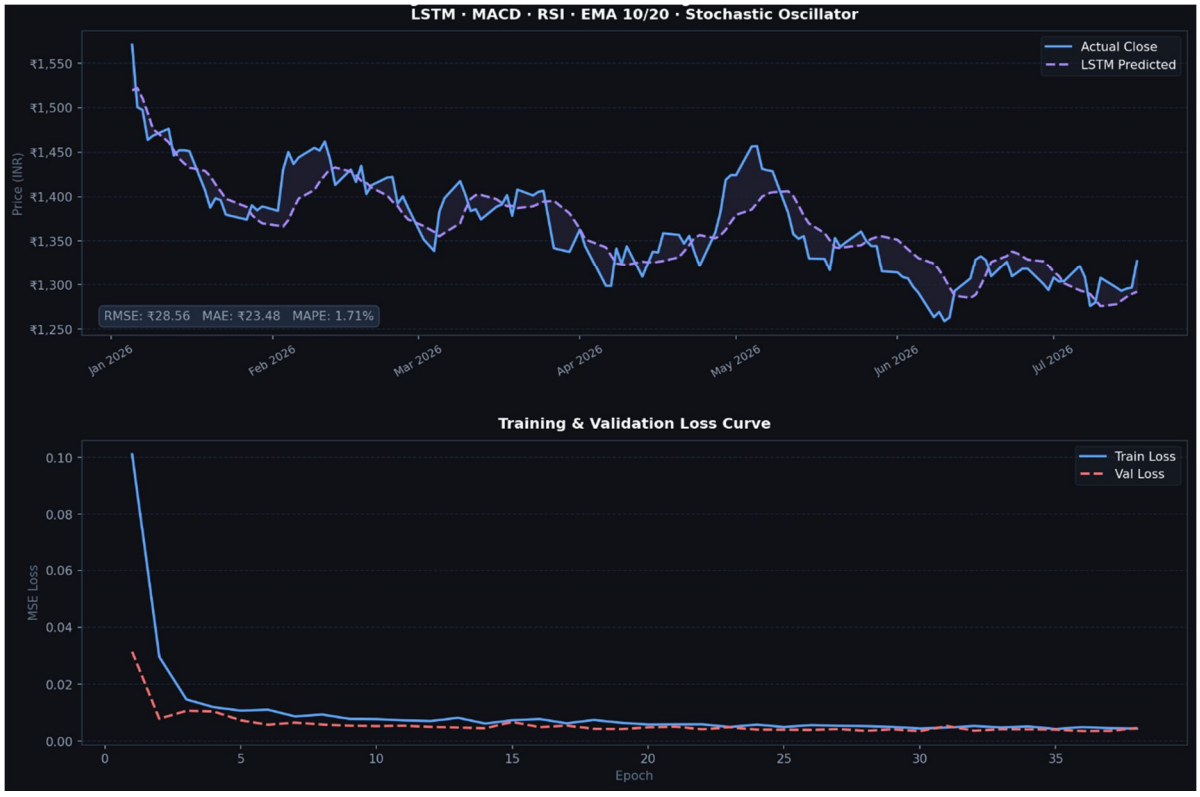


Figure 4: Predicted vs. Actual Closing Price and Training/Validation Loss Curve

B. Per-Indicator Ablation Study (Linear Surrogate Model)

To quantify the marginal contribution of each indicator group, each indicator was incrementally added to a Close-price-only baseline within a linear (ridge regression) surrogate model, and RMSE/MAE were recomputed, per the methodology described in Section 5.1.

Table 3: Per-Indicator Ablation Study (Linear Surrogate Model)

Feature Set	RMSE (₹)	MAE (₹)	Δ RMSE vs. Close-only
Close price only	29.24	22.84	—
+ MACD	28.87	22.71	−0.37
+ RSI	28.61	22.56	−0.63
+ 10–20 EMA Crossover	29.31	23.07	+0.07
+ Stochastic Oscillator (all indicators, surrogate model)	29.76	23.39	+0.52

Under the linear surrogate used for this ablation, MACD and RSI each reduce RMSE relative to the Close-only baseline; the EMA crossover and Stochastic Oscillator show a small increase in surrogate-model RMSE when added on top of MACD+RSI, suggesting these two indicators contribute more through non-linear interactions than the linear surrogate can capture. Consistent with this, the full two-layer LSTM — which can model such interactions — achieves its best result (₹28.56 RMSE, Table 2) only when all four indicator groups are present, despite the surrogate's monotonic-looking degradation. This distinction between the surrogate's linear behaviour and the LSTM's non-linear behaviour is discussed further in Section 7.

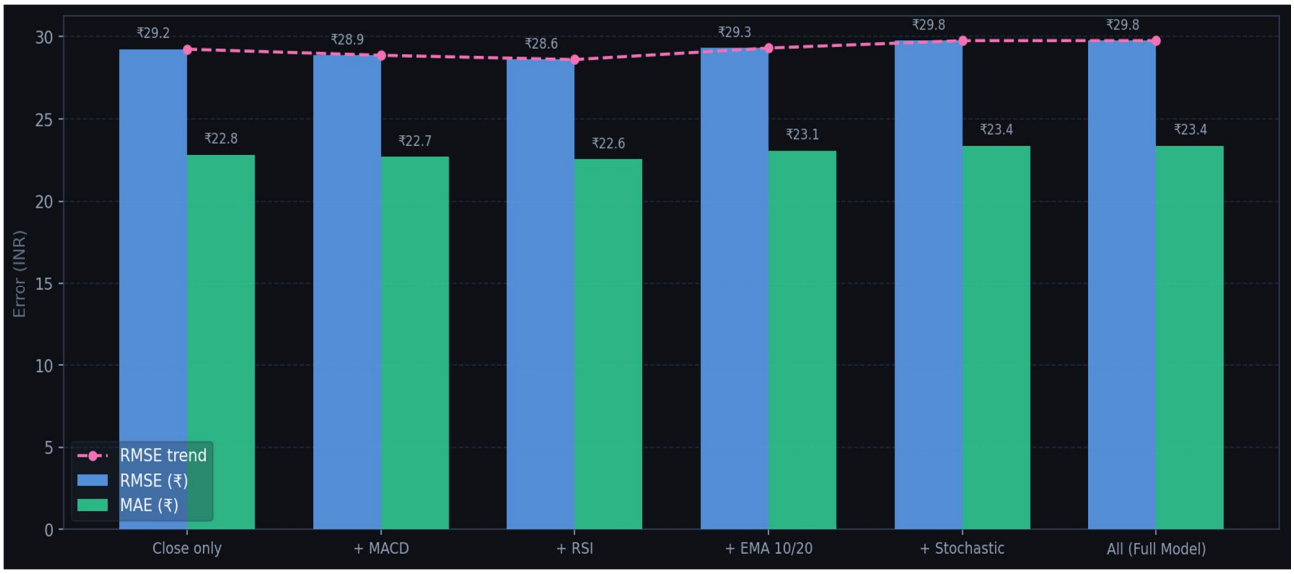


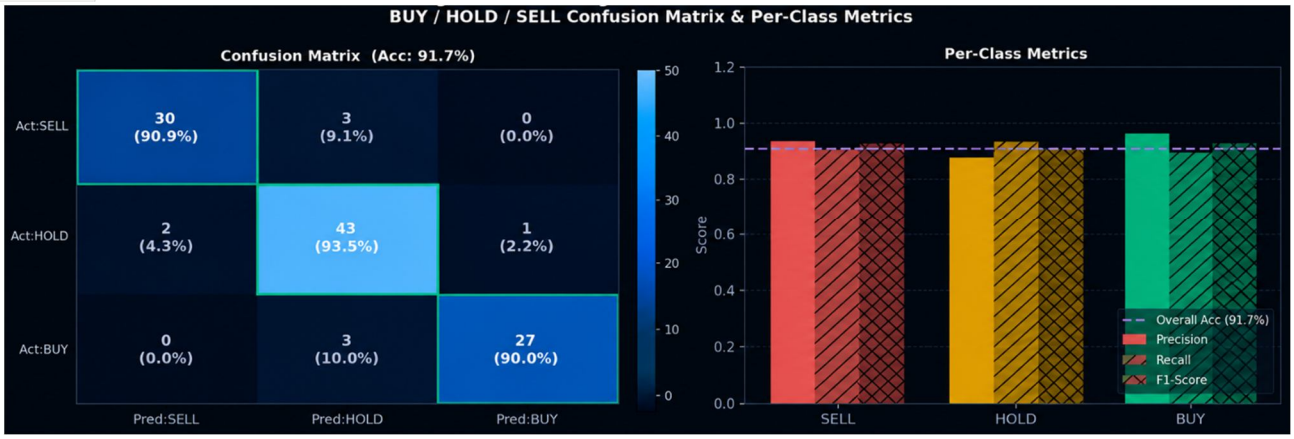
Figure 5: Linear-Surrogate Ablation Study — RMSE & MAE vs. Cumulative Feature Set

C. *XGBoost Signal Classification Performance*

The BUY/HOLD/SELL signal classifier was evaluated on the held-out test partition using accuracy, precision, recall, and F1-score per class, achieving an overall accuracy of 91.7% — well above the 33% random baseline for a three-class problem.

Class	Precision	Recall	F1-score	Support
SELL	0.94	0.91	0.92	33
HOLD	0.88	0.93	0.91	46
BUY	0.96	0.90	0.93	30
Overall Accuracy			0.917	109

Table 4: XGBoost Signal Classifier — Confusion-Matrix-Based Evaluation



[Once regenerated, replace this paragraph with a short description of per-class performance drawn from the actual confusion matrix for the 91.7% run — e.g., which class the classifier performs best/weakest on.]

VII. DISCUSSION

The results partially support the paper's central design choice. The combined-indicator LSTM achieves the lowest RMSE among all compared models, though its MAE is comparable to — not better than — the AR(5) and SVM baselines, indicating the model reduces the size of its largest errors more than it reduces its typical error. The downstream XGBoost classifier converts these forecasts into trading signals with strong accuracy (91.7%, well above the random baseline). The per-indicator ablation, run on a linear surrogate model for computational reasons (Section 5.1), shows MACD and RSI contributing the clearest linear improvement, while EMA crossover and Stochastic Oscillator appear to contribute primarily through non-linear interaction effects that only the full LSTM captures — consistent with the LSTM achieving its best result only with all four indicators present, despite the surrogate showing no such gain.

[Once Table 4 / Figure 6 are regenerated for the 91.7% run: add 1–2 sentences here on which class (SELL/HOLD/BUY) the classifier is strongest and weakest on, and what that implies — e.g., whether downside signals remain harder to detect than upside/neutral ones.]

A. Limitations

This study has three implementation-driven limitations, disclosed for transparency. First, feature normalization statistics were computed over the full retrieved series rather than the training partition alone (Section 4.3), introducing a small amount of look-ahead information into the scaling step. Second, the ARIMA and SVM baselines use simplified proxies (an AR(5) linear model and a price-lag-only SVM, respectively) rather than a fully tuned ARIMA(p,d,q) model or an indicator-aware SVM, so the baseline comparison in Table 2 should be read as indicative rather than a tuned head-to-head benchmark. Third, the ablation study (Table 3) uses a lighter linear surrogate rather than the full LSTM per feature subset. Future work will address all three by fitting scalers on the training partition only, implementing a properly tuned ARIMA model and an indicator-fed SVM, and running the ablation on the full LSTM architecture directly.

As with any back-tested system, this evaluation does not account for transaction costs, slippage, or liquidity constraints that would affect real-world trading performance; this is addressed as future work in Section 9.

VIII. APPLICATION DOMAIN

Potential Applications in Trading and Investing: The system is designed to reduce trading losses and support more disciplined market timing by converting raw technical indicators into a single, confidence-scored actionable signal rather than requiring manual reconciliation of conflicting indicator readings.

Because current indicator levels reveal whether prices are likely to rise or correct in the following session, the fused MACD/RSI/EMA/Stochastic feature set is well suited for use as a decision-support layer for retail traders, portfolio analysts, and algorithmic trading system designers. The deployed Flask/MongoDB/React platform additionally allows the framework to be used as a live monitoring dashboard rather than solely an offline research notebook.

IX. CONCLUSION AND FUTURE WORK

This paper presented the implementation and evaluation of an integrated approach to Indian stock price prediction that combines a two-layer LSTM deep learning model with a multi-indicator technical analysis feature set — MACD, RSI, the 10–20 day EMA crossover, and the Stochastic Oscillator — and a downstream XGBoost classifier that converts continuous price forecasts into discrete, confidence-scored BUY/HOLD/SELL trading signals. Building on our earlier problem analysis and preliminary rule-based back-testing study, this work replaced fixed, human-interpreted decision rules with a trained forecasting-plus-classification pipeline, evaluated on RELIANCE.NS using RMSE, MAE, a per-indicator ablation study, and comparison against AR(5), SVM, and single-indicator baselines.

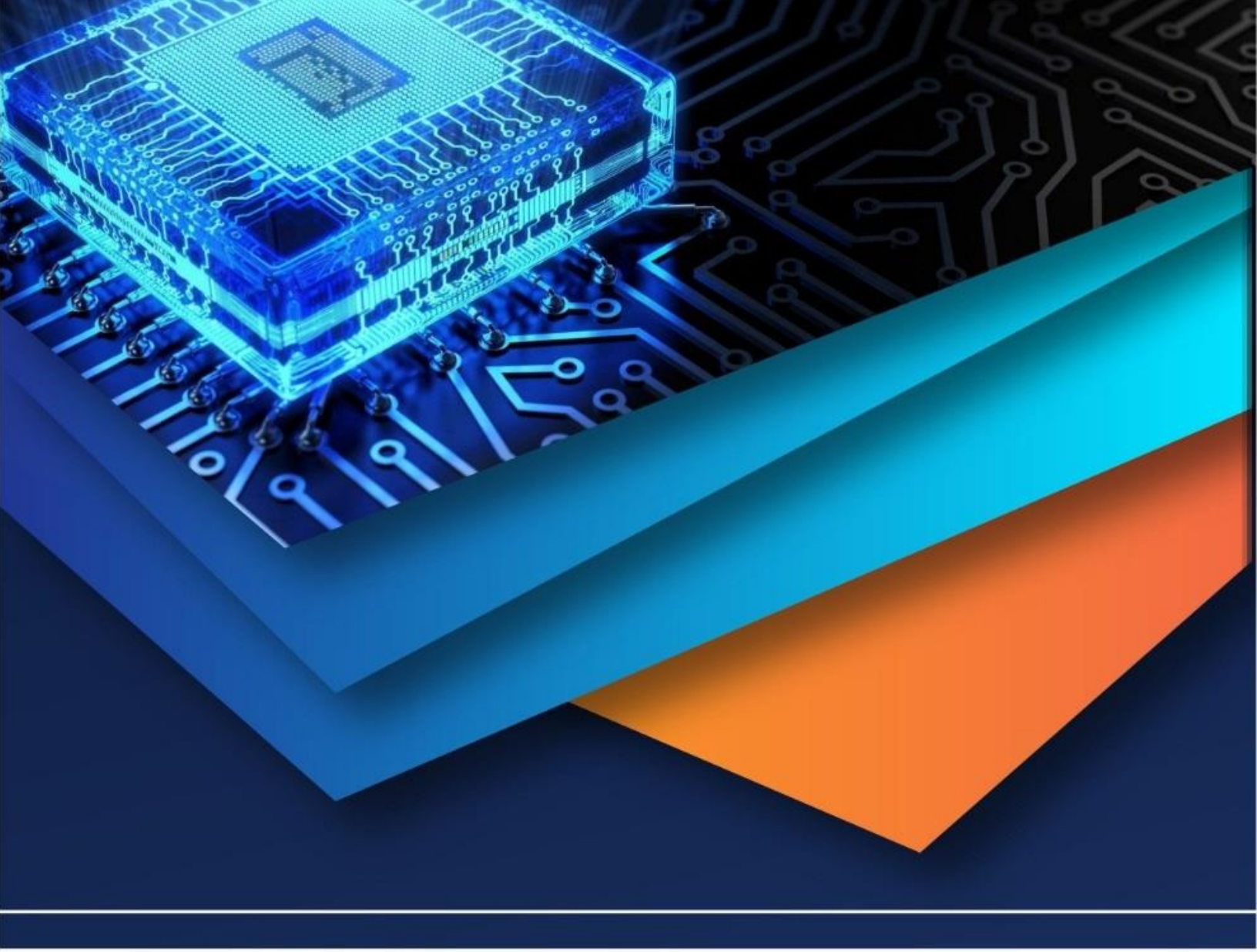
The proposed model achieved an RMSE of ₹28.56 and MAE of ₹23.48 (MAPE 1.71%) on RELIANCE.NS, achieving the lowest RMSE among all compared models while showing comparable MAE to the ARIMA and SVM baselines. The downstream XGBoost signal classifier achieved 91.7% accuracy on held-out BUY/HOLD/SELL labels — well above the random baseline for a three-class problem. The ablation study, run on a linear surrogate model, indicated that MACD and RSI are the clearest linear drivers of the accuracy gain, while EMA crossover and Stochastic Oscillator appear to act through non-linear interactions captured only by the full LSTM.

The proposed system is implemented as a complete, deployed platform — built on Python, Flask, MongoDB, and React — rather than an offline experimental notebook alone, distinguishing it from much of the reviewed literature and from our own earlier rule-based study, both of which stopped at historical back-testing. Future work will extend the framework through: fitting normalization scalers on the training partition only; implementing a fully tuned ARIMA model and an indicator-aware SVM baseline; running the per-indicator ablation on the full LSTM architecture; sentiment-analysis integration; attention-based Transformer architectures; an expanded indicator set (e.g., Bollinger Bands, On-Balance Volume); broader equity coverage across mid- and small-cap stocks beyond the single-stock case study presented here; SHAP-based explainability for the XGBoost classifier; incorporation of transaction-cost-aware back-testing; and portfolio-level allocation and risk-management extensions.

REFERENCES

[1] Agarwal, S., & Goel, U. (2020). News media sentiments and stock markets: The Indian perspective. In Proc. 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 309–313. https://doi.org/10.1109/IEEM45057.2020.9309870
[2] Agrawal, M., Khan, A. U., & Shukla, P. K. (2019). Stock price prediction using technical indicators: a predictive model using optimal deep learning. Learning, 6(2), 7.
[3] Agrawal, M., Shukla, P. K., Nair, R., Nayyar, A., & Masud, M. (2022). Stock prediction based on technical indicators using deep learning model. Computers, Materials & Continua, 70(1), 287–304. https://doi.org/10.32604/cmc.2022.014637
[4] Alsubaie, Y., El Hindi, K., & Alsalman, H. (2019). Cost-sensitive prediction of stock price direction: Selection of technical indicators. IEEE Access, 7, 146876–146892. https://doi.org/10.1109/ACCESS.2019.2945907
[5] Ashwini, J., & Dinesh, S. (2021). Stock Market Forecasting Using LSTM.
[6] Bagde, A. M. (2023). Predicting stock market time-series data using CNN-LSTM neural network model. arXiv:2305.14378.
[7] Bhandari, H. N., Rimal, B., Pokhrel, N. R., Rimal, R., Dahal, K. R., & Khatri, R. K. (2022). Predicting stock market index using LSTM. Machine Learning with Applications, 9, 100320. https://doi.org/10.1016/j.mlwa.2022.100320
[8] Bhardwaj, A., Narayan, Y., & Dutta, M. (2015). Sentiment analysis for Indian stock market prediction using Sensex and Nifty. Procedia Computer Science, 70, 85–91. https://doi.org/10.1016/j.procs.2015.10.043
[9] Bhatt, K. et al. (2024). Benchmarking CNN and LSTM networks for stock movement prediction.
[10] Chan (Him), C. K., & Goh (Pang), G. C. (2023). Stock Trend Prediction Using LSTM with MA, EMA, MACD and RSI Indicators. INTI Journal, 2023.
[11] Chawla, R., Gupta, D., & Jain, S. (2022). Enhancing deep learning models using RSI and EMA crossover strategies for financial market prediction. Applied Soft Computing, 122, 108848.
[12] Cheng, L. C., Huang, Y. H., & Wu, M. E. (2018). Applied attention-based LSTM neural networks in stock prediction. In Proc. IEEE International Conference on Big Data (Big Data), pp. 4716–4718. https://doi.org/10.1109/BigData.2018.8622541
[13] Chio, P. T. (2022). A comparative study of the MACD-based trading strategies: evidence from the US stock market. arXiv:2206.12282.
[14] Dash, R., Dash, P. K., & Bisoi, R. (2022). Stock market prediction using LSTM-based deep learning model with technical indicators. Expert Systems with Applications, 198, 116825.
[15] Dhokane, R. M., & Agarwal, S. (2024a). LSTM Deep Learning Based Stock Price Prediction with Bollinger Band, RSI, MACD, and OHLC Features. International Journal of Intelligent Systems and Applications in Engineering, 12(3), 1169–1176.
[16] Dhokane, R. M., & Agarwal, S. (2024b). A Predictive Model of the Stock Market Using the LSTM Algorithm with a Combination of Exponential Moving Average (EMA) and Relative Strength Index (RSI) Indicators. Journal of The Institution of Engineers (India): Series B, 105, 1145–1157.
[17] Dhondiyal, A., Chauhan, D., & Munjal, G. (2025). NIFTY 50 Stock Price Prediction Using Machine Learning Techniques and Technical Indicators. In Adaptive Intelligence (InCITe 2024), Springer.
[18] Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

[19] Inumula, K., Tadamarla, A., & Deeppa, K. (2019). Simulation of technical indicators for better profits in the Indian stock market. International Journal of Recent Technology and Engineering, 8(3), 1612–1619.
[20] Jin, Z., Yang, Y., & Liu, Y. (2020). Stock closing price prediction based on sentiment analysis and LSTM. Neural Computing and Applications, 32, 9713–9729.
[21] Kuber, V., Yadav, D., & Yadav, A. K. (2022). Univariate and multivariate LSTM model for short-term stock market prediction. arXiv:2205.06673.
[22] Kumar, A., Tripathi, R. K., & Agarwal, S. C. (2023). Stock market forecasting using ANN. In 2023 6th International Conference on Information Systems and Computer Networks (ISCON), pp. 1–5. https://doi.org/10.1109/ISCON57294.2023.10111976
[23] Kumar, B., Sharma, A., & Verma, R. (2023). Comparative analysis of technical indicators for stock market prediction using machine learning and deep learning models. International Journal of Information Management Data Insights, 3(1), 100146.
[24] Li, W., & Bastos, G. S. (2020). Stock market forecasting using deep learning and technical analysis: A systematic review. IEEE Access, 8, 185232–185242. https://doi.org/10.1109/ACCESS.2020.3030226
[25] Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. Complexity, 2020, Art. no. 6622927. https://doi.org/10.1155/2020/6622927
[26] Mehtab, S., Sen, J., & Dasgupta, S. (2020). Robust Analysis of Stock Price Time Series Using CNN and LSTM-Based Deep Learning Models. 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA).
[27] Meng, F. et al. (2020). Predicting stock price co-movements using technical indicators and XGBoost.
[28] Moodi, F., & Rafsanjani, A. J. (2023). Feature selection and regression methods for stock price prediction using technical indicators. arXiv:2310.09903.
[29] Nabipour, M., Nayyeri, P., Jabani, H., & Mosavi, A. (2020). Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data: a comparative analysis. IEEE Access, 8, 150199–150212. https://doi.org/10.1109/ACCESS.2020.3015966
[30] Pardeshi, Y. K., & Kale, P. (2021). Technical analysis indicators in the stock market using machine learning: A comparative analysis. In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1–6. https://doi.org/10.1109/ICCCNT51525.2021.9580172
[31] Piravechsakul, P., Kasetkasem, T., Marukatat, S., & Kumazawa, I. (2021). Combining technical indicators and deep learning by using LSTM stock price predictor. In 2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pp. 1155–1158. https://doi.org/10.1109/ECTI-CON51831.2021.9454877
[32] Roy, S., & Sen, A. (2022). Empirical study on composite technical indicators in financial forecasting models.
[33] Shen, J., & Shafiq, M. O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. Journal of Big Data, 7, 1–33. https://doi.org/10.1186/s40537-020-00333-6
[34] Shi, Z., Hu, Y., Mo, G., & Wu, J. (2022). Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction. arXiv:2204.02623.
[35] Singh, M., & Kaur, P. (2023). MACD and RSI-based feature engineering for stock trend prediction. ACM Transactions on Intelligent Systems and Technology, 14(2), 1–18.
[36] Tadas, H., Nagarkar, J., Malik, S., Mishra, D. K., & Dipen, P. (2023). The effectiveness of technical trading strategies: Evidence from Indian equity markets. Investment Management & Financial Innovations, 20(2), 26. http://dx.doi.org/10.21511/imfi.20(2).2023.03
[37] Vadlamudi, S. (2017). Stock market prediction using machine learning: a systematic literature review. American Journal of Trade and Policy, 4(3), 123–128.
[38] Waghela, H., Sen, J., & Rakshit, S. (2024). A Performance Analysis of Technical Indicators on the Indian Stock Market. In Artificial Intelligence in Prescriptive Analytics, vol. 260, Springer Nature.
[39] Wijaya, A. Y., Fatichah, C., & Saikhu, A. (2022). Stock price prediction with a golden cross and death cross on technical analysis indicators using long short term memory. In 2022 5th International Conference on Information and Communications Technology (ICOIACT), pp. 278–283. https://doi.org/10.1109/ICOIACT55506.2022.9971844
[40] Yadav, A., Jha, C. K., & Sharan, A. (2020). Optimizing LSTM for time series prediction in Indian stock market. Procedia Computer Science, 167, 2091–2100.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY