Stock Market Prediction Using Deep Learning (LSTM)

Prepared By

Aranya Adhikary	**12200121041**	**211220100110021**
Shounak Dey	**12200121011**	**211220100110025**
Dipan Dutta	**12200121016**	**211220100110030**
Subhon Sanyal	**12200121068**	**211220100110060**

Under the guidance of

Mr. Ranit Baram

Assistant Professor

Department of Computer Science and Engineering

Project Report

Submitted in the partial fulfillment of the requirement for the degree of

B.Tech in Computer Science and Engineering

Department of Computer Science and Engineering
St. Thomas' College of Engineering and Technology

Affiliated to

Maulana Abul Kalam Azad University of Technology, West Bengal

June, 2025

St. Thomas’ College of Engineering and Technology
Department of Computer Science and Engineering

This is to certify that the work in preparing the project entitled “Stock Market Prediction Using Deep Learning (LSTM)” has been carried out by Aranya Adhikary (University Roll No: 12200121041), Shounak Dey (University Roll No: 12200121011), Dipan Dutta (University Roll No: 12200121016), Subhon Sanyal (University Roll No: 12200121068) under my guidance during the session 2024-25 and accepted in the partial fulfillment of the requirement for the degree of B.Tech in Computer Science and Engineering.

Signature	**Signature**
Mr. Ranit Baram	Dr. Mousumi Dutt
(Assistant Professor)	(Head of the Department & Associate Professor)
Department of Computer Science and Engineering	Department of Computer Science and Engineering
St. Thomas' College of Engineering and Technology	St. Thomas' College of Engineering and Technology

St. Thomas' College of Engineering and Technology
Department of Computer Science and Engineering

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who supported and guided us in completing our final year project titled "Stock Market Prediction Using Deep Learning (LSTM)".

We are especially thankful to our respected guide, Mr. Ranit Baram, for his constant support, expert guidance, and valuable insights throughout the course of the project. His encouragement and constructive suggestions helped us stay focused and motivated at every stage of development.

We also extend our appreciation to the faculty members and staff of the Computer Science and Engineering Department, St. Thomas' College of Engineering and Technology, for providing the academic support and infrastructure required for the successful execution of our work.

Last but not the least, we are grateful to our families and friends for their constant encouragement, patience, and understanding during the completion of this project.

Signature with Date	**Signature with Date**
Aranya Adhikary, 12200121041	Dipan Dutta, 12200121016
Signature with Date	**Signature with Date**
Shounak Dey, 12200121011	Subhon Sanyal, 12200121068

St. Thomas' College of Engineering and Technology
Department of Computer Science and Engineering

DECLARATION

We declare that this written submission represents our ideas in our own words and we have adequately cited and referenced the original sources. We also declare that we have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Aranya Adhikary	**Shounak Dey**	**Dipan Dutta**	**Subhon Sanyal**
Roll: 12200121041	Roll: 12200121000	Roll: 12200121016	Roll: 12200121068

Index

I. Pre-amble i

I.I Institute Vision and Mission

Department Vision and Mission i

I. II Program Outcome (PO) and Program Specific Outcome (PSO) ii

I. III PO and PSO mapping with justification..iii

Abstract

Chapter 1: Introduction 01

1.1 Problem Statement..03

1.2 Objectives..03

1.3 Brief Discussion on Problem...03

1.4 Tools and Platform...04

1.5 Project Organization...05

1.6 Project Timeline...06

Chapter 2: Literature Survey 07

Chapter 3: Concepts and Problem Analysis 10

3.1 Concepts..11

3.2 Problem Analysis...13

Chapter 4: Design and Methodology 16

4.1 System Overview..17

4.2 Dataset Description..17

4.3 Data Acquisition Pipeline...18

4.4 Feature Engineering..18

4.5 Data Preparation for LSTM...19

4.6 Model Architecture...20

4.7 Model Configuration...21

4.8 Training Strategy...21

4.9 Recursive Forecasting Logic....22
4.10 Evaluation Metrics....24
4.11 Hyperparameter Tuning Strategy....25
4.12 Deployment Architecture....25
4.13 System Design....26

Chapter 5: Testing and Result Discussions....28
5.1 Model Training and Convergence Behaviour....29
5.2 Baseline Model Performance....29
5.3 LSTM Model Performance....31
5.4 Forecast/Prediction Visualization Description....32
5.5 Recursive Forecast Error Accumulation....33
5.6 Failure Case Analysis....33
5.7 Practical Interpretation for Investors....34
5.8 Visualisation of actual vs prediction plots....34

Chapter 6: Conclusion and Future Work 39
6.1 Conclusion....40
6.2 Future Scope....41

References....43
Annexures....45

Pre-amble

Vision and Mission

Vision of the Institute

To evolve as an industry oriented, research-based Institution for creative solutions in various engineering domains, with an ultimate objective of meeting technological challenges faced by the Nation and the Society.

Mission of the Institute

1. To enhance the quality of engineering education and delivery through accessible, comprehensive and research-oriented teaching-learning-assessment processes in the state-of-art environment.
2. To create opportunities for students and faculty members to acquire professional knowledge and develop managerial, entrepreneurial and social attitudes with highly ethical and moral values.
3. To satisfy the ever-changing needs of the nation with respect to evolution and absorption of sustainable and environment friendly technologies for effective creation of knowledge-based society in the global era.

Vision of the Computer Science and Engineering Department

To continually improve upon the teaching-learning processes and research with a goal to develop quality technical manpower with sound academic and practical experience, who can respond to challenges and changes happening dynamically in Computer Science and Engineering.

Mission of the Computer Science and Engineering Department

1. To inspire the students to work with latest tools and to make them industry ready.
2. To impart research based technical knowledge.
3. To groom the department as a learning centre to inculcate advanced technologies in Computer Science and Engineering with social and environmental awareness.

Program Outcome (PO) and Program Specific Outcome (PSO)

Program Outcome (PO)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PO and PSO mapping with justification

PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
3	3	3	3	3	2	2	3	2	3	2	3	3	3	3

PO1: Engineering Knowledge

The project applies knowledge of mathematics (e.g., chaotic maps, Reed Solomon codes), computer science, and neural network architectures like DenseNet to solve complex steganography problems. It demonstrates how advanced engineering knowledge can develop solutions to secure data embedding and extraction.

PO2: Problem Analysis

The team analyzed challenges like adversarial attacks, compression artifacts, and robustness of steganographic systems, using research methods to design a system resistant to these issues. Problem analysis helped identify key solutions like chaotic mapping and error correction using Reed Solomon encoding.

PO3: Design/Development of Solutions

The design of encoder-decoder neural networks for embedding and extracting secret messages, combined with chaotic mapping and error-correction techniques, addresses specific needs of robustness and imperceptibility. The team's iterative improvements further showcase their focus on robust system design.

PO4: Conduct Investigations of Complex Problems

Extensive testing, including simulations of adversarial and compression challenges, is carried out to validate system robustness. The iterative process of redesigning neural networks and adjusting parameters demonstrates an investigative approach to system optimization.

PO5: Modern Tool Usage

The project leverages tools like convolutional neural networks (CNNs), programming frameworks, and data compression techniques. Use of advanced steganography evaluation tools like StegExpose underlines proficiency in modern engineering tools.

PO6: The Engineer and Society

This system ensures secure communication for personal and organizational needs, addressing concerns like privacy and data protection. The project considers societal implications of secure data embedding.

PO7: Environment and Sustainability

By minimizing bandwidth consumption through efficient data embedding and compression-resistant techniques, the project indirectly supports sustainable communication practices in constrained network environments.

PO8: Ethics

Ethical principles are adhered to by ensuring responsible data use and protecting intellectual property through secure and robust design choices.

PO9: Individual and Team Work

The project demonstrates effective collaboration, with each team member contributing to distinct components like chaotic mapping, neural network architecture, and error correction. Combined efforts ensure a comprehensive solution.

PO10: Communication

The documentation, including a well-structured report and detailed visualizations, facilitates clear communication of the project's findings and methodologies to peers and evaluators.

PO11: Project Management and Finance

Project timelines, alternative strategies for failures, and clear planning for future improvements reflect sound project management skills. Budgeting efforts for computing resources are implied.

PO12: Life-long Learning

The project emphasizes the team's adaptability to emerging technologies like dense networks and chaotic maps, ensuring continuous improvement in their knowledge of steganography and neural networks.

PSO1: Industry Readiness

The use of advanced neural network designs and robust steganographic techniques prepares students for industry challenges in secure communication and data protection.

PSO2: Research and Development

By integrating concepts like chaotic mapping and Reed Solomon codes, the project pushes the boundaries of existing steganography techniques, showcasing research-driven innovation.

PSO3: Practical Implementation

Testing the system with real-world scenarios, such as social media compression, ensures practical applicability and usability of the developed technique.

Abstract

Stock price prediction is a challenging task due to the non-linear, volatile, and multi-factorial nature of financial markets. This project presents a deep learning-based forecasting system using Long Short-Term Memory (LSTM) neural networks to predict stock prices for companies like Apple, Amazon, Wipro, and Infosys. Full Historical data was collected using Alpha Vantage API, with additional INR conversion support via exchangerate-api for localization.

The system integrates multiple technical indicators—such as 20-day Simple Moving Average (SMA), 52-week highs and lows, trading volume, and trade price—to form a multivariate input structure. The LSTM model is trained on these engineered features using a sliding window approach. Predictions are made using recursive forecasting, where the model reuses its output to predict subsequent values. While recursive forecasting introduces error accumulation over longer prediction horizons, several long-term visual plots still captured directional trends, even if numerical accuracy declined while the reverse happened for shorter prediction horizons.

Model performance is evaluated using standard statistical metrics: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and R^2 score. MAPE is derived using MAE. For visualization, only MAPE and R^2 score value are shown as they give the best feedback about the prediction. To benchmark performance, classical models such as ARIMA and Linear Regression are implemented. The results show that the LSTM model offers superior forecasting accuracy, particularly for short-term predictions.

This system provides a modular and extensible forecasting framework, with future potential for expansion into web-based dashboards.

CHAPTER 1

INTRODUCTION

1. Introduction

The stock market is a dynamic and complex financial system that reflects the economic health and investor sentiment of a country. It plays a crucial role in capital allocation by allowing businesses to raise funds and offering investors an opportunity to generate returns. However, accurately predicting stock prices has always been a significant challenge due to the highly volatile, non-linear, and multi-dimensional nature of financial markets. Prices are influenced by a vast array of factors, ranging from company fundamentals and macroeconomic indicators to investor psychology and unforeseen geopolitical events.

Over the years, researchers and practitioners have explored numerous techniques to model and forecast stock prices. Traditional statistical models like ARIMA, Moving Average, and Linear Regression have provided foundational methods for time-series analysis, but they often fall short in capturing complex temporal dependencies and interactions between multiple influencing variables. With the evolution of machine learning and, more recently, deep learning, there has been a paradigm shift toward models capable of learning intricate patterns from large datasets.

This project leverages the power of Long Short-Term Memory (LSTM) neural networks—a specialized form of Recurrent Neural Networks (RNNs)—to perform time-series forecasting in the context of stock market prediction. LSTM networks are adept at handling long-term dependencies in sequential data, making them highly suitable for financial forecasting where past price movements and indicators significantly influence future values.

Our system not only utilizes historical stock data but also incorporates technical indicators and real-time currency conversion to make the model more informative and investor-friendly. The architecture is designed to forecast both short-term and long-term trends using a recursive multi-step approach. By comparing the LSTM's performance with baseline models, we aim to demonstrate its superiority in terms of accuracy and robustness.

This project aims to reflects a comprehensive approach to addressing the challenges of financial prediction while ensuring practical applicability and scalability for real-world deployment and providing reliable results to ensure that the investors make better decisions.

1.1 Problem Statement

Forecasting stock prices over multiple time steps is challenging due to the dynamic, non-linear, and uncertain nature of financial markets. This project explores the use of LSTM-based deep learning models to overcome these limitations using historical stock data and technical indicators.

1.2 Objective

The objectives of our problem statement involve the following:

- Develop a stock price forecasting system using historical data.
- Implement LSTM neural networks to capture temporal dependencies.
- Compare LSTM performance with baseline models such as ARIMA and Linear Regression.
- Evaluate prediction accuracy using multiple performance metrics.
- Create the foundation for eventual deployment via a web interface.

1.3 Brief Discussion on Problem

Stock markets operate under conditions of uncertainty and are affected by numerous variables. Forecasting prices reliably is crucial for investors but is hindered by the unpredictability of trends and external shocks. Traditional models often fail to capture deep sequential dependencies. LSTM models, however, are designed to retain long-term memory of sequential data, making them a more promising choice for modeling financial trends using multivariate time-series inputs.

In this project, the forecasting model was trained using real historical data from companies such as Apple, Amazon, Infosys, and Wipro. These stocks represent both international and Indian markets, providing a diversified dataset with varied patterns. The multivariate input features were constructed from technical indicators to support temporal trend learning across companies and sectors.

To acquire data efficiently, the system integrates Alpha Vantage's API, using a developer-provided API key to fetch daily stock prices in real-time. Additionally, the ExchangeRate-API is used to convert foreign currency values (USD) into INR, enabling standardized prediction outputs. This automation reduces manual intervention and ensures the dataset reflects current market conditions.

1.4 Tools and Platform

This section outlines the complete technical stack used to build the investor-oriented stock prediction system — from backend deep learning components to the frontend user interface.

1.4.1 Programming Language and Development Environment

- **Python 3.10:** Core language for backend data processing, LSTM model training, evaluation, and API communication.
- **Jupyter Notebook / Google Colab:** Used for model development, testing, and visualization during the research and experimentation phase.
- **Flask (or FastAPI):** Lightweight Python web framework to expose the model predictions and metrics as RESTful API endpoints. It serves as the bridge between frontend and backend.

1.4.2 Frontend Development (Web Interface)

To make the prediction system accessible and interactive for investors, a fully functional web interface is being developed using:

- **HTML5:** For structuring content on the prediction interface.
- **CSS3:** For responsive styling, layout design, and branding.
- **JavaScript:** For handling user interactions, triggering API calls (e.g., fetching forecasts or evaluation plots), and dynamically updating charts.

The website allows users to input company names, select timeframes (weekly, monthly, yearly), view predicted plots, and interact with real-time forecasting options in a user-friendly manner.

1.4.3 Data Acquisition Tools

- **Alpha Vantage API:** Used to fetch historical daily stock data (Open, High, Low, Close, Volume) for both Indian and international stocks.
- **ExchangeRate-API:** Fetches live USD to INR conversion rates to localize predictions for Indian investors.

1.4.4 Backend Libraries and Machine Learning Frameworks

- **NumPy:** For efficient numerical computation and matrix operations.
- **Pandas:** For data cleaning, feature engineering, and time-series manipulation.
- **Scikit-learn:** Used for preprocessing (MinMaxScaler) and performance evaluation (MAE, MAPE, R^2 Score).
- **TensorFlow / Keras:** Deep learning framework used to implement the LSTM model with layers such as LSTM, Dropout, Dense, and model callbacks like EarlyStopping and LearningRateScheduler.

1.4.5 Visualization and Reporting

Matplotlib: Used to generate line plots, prediction charts, and custom range evaluation graphs (later exported as PNG or SVG for frontend integration).

1.5 Project Organization

The project is structured into clear phases:

- Data acquisition using APIs (Alpha Vantage for stocks, ExchangeRate-API for currency conversion).
- Data preprocessing and technical feature engineering.
- LSTM model design and training.
- Benchmarking against baseline models.
- Performance evaluation and visualization.
- Planning for future deployment options (web-based tools).

To ensure the successful execution of our project, we have outlined a detailed schedule with specific steps and alternative solutions in case any objectives are not met:

Phase	Description	Status
API Integration	Fetch live stock data from Alpha Vantage	Completed
Symbol Mapping	Company name to symbol matching via fuzzy logic	Completed
Feature Engineering	Build indicators (SMA, highs/lows, trade price)	Completed
Data Normalization	MinMax scaling and sequence shaping	Completed
Model Design	Two-layer LSTM architecture	Completed
Training and Evaluation	Model training with callbacks and metrics	Completed
Forecasting Module	Predict N-day/week/month/custom future price	Completed
Visualization Module	Generate plots (yearly, monthly, weekly, custom)	Completed
Interface (optional)	Deploy CLI/web-based interface	In Progress

Table 01: Project Execution Plan

Future Plan – Integration with sentiment/news data, risk analysis. This structured approach ensures that we address all aspects of stock market prediction, from handling volatility to improving accuracy, while remaining flexible to adjustments based on the model's performance.

1.6 Project Timeline

Phase	Duration
Requirement Analysis & Research	2 weeks
Data Collection & Processing	12 weeks
Model Development (LSTM)	14 weeks
Evaluation & Comparisons	10 weeks
Frontend Creation	8 weeks
Documentation & Visualization	2 weeks

Table 02: Project Timeline

CHAPTER 2
LITERATURE SURVEY

2. Literature Survey

Stock price prediction has been an active research domain for decades, driven by the unpredictable and complex dynamics of financial markets. Market behaviour is affected by a multitude of factors including corporate earnings, economic indicators, policy changes, investor sentiment, and external global events. This inherent complexity has inspired researchers to adopt various forecasting methodologies ranging from traditional statistical approaches to modern deep learning models.

Earlier research largely focused on classical time-series techniques such as Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing, and Linear Regression. These methods provided useful results under stable or stationary conditions but proved less effective during periods of high volatility or structural breaks. For instance, ARIMA models are constrained by the assumption of stationarity, which is often violated in real-world stock data where sharp fluctuations, regime shifts, and noise are common [8]. Consequently, the limitations of linear modeling have led to a progressive shift toward more flexible machine learning and deep learning-based techniques.

With the emergence of deep learning, models capable of capturing sequential and nonlinear dependencies have gained prominence. Recurrent Neural Networks (RNNs) and their advanced variant Long Short-Term Memory (LSTM) networks have demonstrated particular promise in the domain of stock market forecasting. LSTMs address key shortcomings of traditional RNNs, such as vanishing gradients, by introducing memory cells that allow the model to retain relevant information over extended time steps [7].

Tran Phuoc et al. (2024) explored the application of LSTM networks to the Vietnamese stock market, focusing on VN-Index and VN30 indices. Their approach combined price data with technical indicators like moving averages and volume to enhance model inputs. The LSTM-based model achieved an average accuracy of over 93%, illustrating its robustness in modeling the volatile behavior of emerging markets [1]. The methodology is particularly relevant to our project given the similar volatility patterns in Indian and U.S. tech stocks.

In a comparative analysis by Hiransha et al. (2018), deep learning models such as LSTM, CNN, RNN, and Multilayer Perceptrons (MLPs) were evaluated using data from both the National Stock Exchange (NSE) of India and the New York Stock Exchange (NYSE). Their findings indicated that deep learning models consistently outperformed traditional statistical models, especially in periods of heightened volatility [2].

LSTM, in particular, showed superior ability in capturing long-term temporal patterns and yielded better forecast accuracy across multiple metrics.

Darapaneni et al. (2022) extended this research by integrating sentiment analysis with LSTM forecasting. They utilized data from news articles and social media platforms to extract investor sentiment and combined it with historical price data. Their hybrid system demonstrated improved predictive accuracy during high-volatility events and unexpected announcements [3]. Although this project does not incorporate sentiment features, the study emphasizes the potential value of such integration in future system upgrades.

Audeliano Wolian Li and Bastos (2020) adopted a multivariate approach, feeding both raw price data and engineered technical indicators — such as Bollinger Bands, RSI, and Moving Averages — into their LSTM models. Their hybrid architecture achieved more stable and reliable forecasts across different market scenarios [4]. This supports the current project's emphasis on feature engineering using indicators like SMA-20, 52-week highs/lows, volume, and trade price.

More recently, Liu et al. (2023) introduced the Informer model, a Transformer-based attention architecture optimized for long sequence time-series forecasting. Unlike LSTM, which processes inputs sequentially, Informer uses attention mechanisms to selectively prioritize historically important data points, resulting in faster training and better long-horizon forecasting [5]. However, the computational complexity of such models makes them less practical for undergraduate-level implementations with limited resources.

Further improvements to sequential forecasting have been proposed by Feng et al. (2021), who developed a Dual-Stage Attention-Based RNN (DA-RNN). Their model incorporates both feature-level and time-step-level attention, allowing the system to dynamically focus on the most relevant indicators and time periods [6]. Their approach significantly enhanced model interpretability and performance but required more computational tuning and data resources. The reviewed studies collectively demonstrate a trend toward deep learning-based forecasting systems that utilize rich, multivariate datasets and advanced architectures to handle volatility and non-linearity. While attention-based models offer impressive performance, their hardware and data demands can be prohibitive for smaller-scale or academic projects.

This project adopts a practical balance by leveraging LSTM architecture with technical indicator-based feature engineering. The approach reflects insights from recent literature, focusing on interpretability, resource feasibility, and performance, thereby laying the groundwork for future enhancements such as sentiment integration or attention-based modules.

CHAPTER 3
CONCEPT AND PROBLEM ANALYSIS

3. Concepts and Problem Analysis

3.1 Concepts

This section explains the core technical concepts and strategies that underpin the design and implementation of our stock prediction system. Each concept contributes to solving a specific problem in the domain of financial time-series forecasting.

3.1.1 Deep Learning for Time-Series Forecasting

Long Short-Term Memory (LSTM) networks are a special type of Recurrent Neural Network (RNN) designed to capture long-term dependencies in sequential data. This makes them particularly suited to financial time-series forecasting, where patterns may emerge over extended periods. In contrast to traditional feedforward networks, LSTMs retain contextual memory across multiple time steps, allowing the system to understand the evolving nature of stock price behavior. In this project, a two-layer LSTM model (512 and 256 units) followed by dense layers is used to capture both high-level market trends and short-term price fluctuations.

3.1.2 Multivariate Feature Integration

To improve the quality and context of predictions, the model uses a multivariate feature set instead of relying solely on closing prices. Features include:

- **SMA-20 (20-day Simple Moving Average):** Smoothens short-term volatility and captures trend direction.
- **52-Week High and Low:** Identifies long-term price boundaries, crucial for breakout/breakdown analysis.
- **4-Week High and Low:** Detects short-term reversals and swing trading opportunities.
- **Daily Volume:** Captures trading activity and liquidity shifts.
- **Trade Price:** Average of daily high and low, representing a central price.
- **Previous Close and Next Open:** Introduces sequential continuity and captures price gaps.
- **Current Close (Target Variable):** The primary variable being forecasted.

These indicators simulate the thought process of experienced traders and institutional investors. By embedding them into the training set, the model gains the ability to associate market signals with future movements, improving its predictive power.

3.1.3 Real-Time Data Acquisition and Currency Conversion

Real-time forecasting is only useful if it reflects current market conditions. To enable this, the system fetches live stock data through the Alpha Vantage API. Additionally, since some stocks (like Amazon) are priced in USD, the project includes an automatic USD to INR conversion using exchangerate-api.

This conversion is not optional—it plays a vital role for Indian investors by:

- Making forecasted prices directly interpretable without requiring manual conversion
- Reflecting the combined effect of stock performance and currency fluctuations on investment value
- Supporting decision-making in a localized financial context
- A fallback static INR value ensures the model remains functional even when the API is temporarily unavailable

3.1.4 Recursive Forecasting Mechanism

The system is designed to allow flexible future predictions—daily, weekly, monthly, or even over custom date ranges. To support this, it employs a recursive forecasting mechanism where the model's output for one step is reused as input for the next step.

While this approach risks the accumulation of small errors over time, it provides a more realistic forecasting framework. Investors often plan weeks or months ahead, and this structure allows the system to simulate multi-step forecasting scenarios like “What if I hold this stock for the next 3 months?”

To reduce forecast drift, the model is trained using:

- Carefully tuned learning rate decay
- Early stopping callbacks
- Sequence windows that are long enough (200 days) to capture sustained behavior

3.1.5 Evaluation Metrics and Interpretability

Model performance is evaluated using:

- Mean Absolute Error (MAE)
- Mean Absolute Percentage Error (MAPE)
- R^2 Score

MAPE is derived using MAE. For visualisation we show MAPE or MAE % and R^2 Score values for better interpretation and best feedback for predictions.

3.1.6 Visualization Tools for Decision Support

The model's results are not just numerical—they are visualized in an intuitive, user-friendly manner. Using Matplotlib, the system generates:

- Full-range actual vs. predicted price graphs
- Year-wise and month-wise breakdowns
- Custom window forecasts with metrics like MAPE or MAE%, R^2.

All plots are color-coded, date-aligned, and labeled for easy reading, making them useful even for investors without a technical background.

3.2 Problem Analysis

Stock price prediction is a complex problem shaped by multiple layers of uncertainty, volatility, and interdependent variables. Financial markets are sensitive to macroeconomic conditions, investor psychology, political developments, and unforeseen global events. These inputs introduce several technical and practical problems for any predictive system. Below is a detailed breakdown of the specific challenges and how this project attempts to resolve them using a structured deep learning framework:

3.2.1 Market Volatility and Noise

One of the foremost challenges is high-frequency market noise caused by investor sentiment, news headlines, and unexpected economic events. These irregular fluctuations make it hard to distinguish between meaningful patterns and random disturbances. The LSTM model addresses this by learning long-term dependencies and ignoring irrelevant short-term fluctuations. Additionally, dropout layers and early stopping are implemented to prevent the model from overfitting on noisy data.

3.2.2 Non-Linearity of Financial Time Series

Stock price movements are non-linear and often governed by dynamic interactions between numerous features. Traditional models like ARIMA assume linearity and stationarity, making them unsuitable for real-world scenarios. The LSTM, with its ability to model non-linear, multivariate sequences, handles irregular spikes and trend reversals more effectively. It learns patterns across variables such as moving averages, highs/lows, and volumes to provide generalized predictions.

3.2.3 Curse of Dimensionality

Unlike univariate models, this system includes 10 different technical indicators that capture short-term, medium-term, and long-term patterns. Each feature contributes unique insights—such as trend strength (SMA-20), momentum (weekly highs/lows), and volume activity—making the model more robust. However, managing and scaling this high-dimensional input space requires proper normalization and sequence shaping, both of which are addressed through MinMax scaling and chronological ordering.

3.2.4 Recursive Forecasting

Rather than limiting forecasts to a single day, this project enables multi-step predictions using recursive forecasting. In this method, the output from one prediction step is fed as input into the next, enabling users to view trends over longer durations. Although this can amplify errors over time, it reflects how forecasts evolve in real financial decision-making. It helps investors simulate future outcomes—such as next week's or next month's prices—based on currently known data. The model architecture and training strategy (learning rate decay, early stopping) are optimized to minimize drift while still allowing for flexible and dynamic forecasting.

3.2.5 Temporal Dependencies and Long-Term Memory

Patterns like seasonal cycles, support-resistance levels, or investor reaction trends unfold over long periods. LSTM is particularly suited to this as it retains information across hundreds of time steps. In our implementation, a 200-day sequence window is used to learn sustained behaviors, which is crucial for reliable forecasting.

3.2.6 Forecast Interpretability for Investors

A major problem in deep learning-based finance tools is lack of transparency. Users often don't understand the meaning of errors or confidence levels. This project explicitly converts raw MAE scores into percentage-based MAPE (e.g., 2.5%), and offers visualization tools with color-coded trends, labels, and legends. This empowers investors to make sense of model predictions without requiring a technical background.

3.2.7 Currency Conversion Justification and Localization

Since stocks like Amazon are traded in USD, but Indian investors often think in INR, this model incorporates live USD to INR conversion using exchangerate-api. This step ensures that the predictions are immediately understandable and useful to Indian users without any additional manual effort. Moreover, in times of fluctuating exchange rates, showing results in INR reflects the actual expected value better than presenting them in foreign currency.

3.2.8 Dynamic Applicability to Any Stock

By integrating Alpha Vantage's listing fetch feature, the model is not hardcoded to any stock. Users can enter any valid company name, and the system auto-fetches the relevant stock symbol and data. This makes the solution highly scalable and adaptable without manual tuning.

CHAPTER 4
DESIGN AND METHODOLOGY

4. Design and Methodology

4.1 System Overview

The forecasting system developed in this project integrates real-time data retrieval, advanced feature engineering, and deep learning-based time-series prediction using LSTM networks. It supports recursive forecasting and modular evaluation through multiple metrics. Although currently focused on core functionality, the system is structured in a way that allows for future enhancements such as macroeconomic factor inclusion, sentiment-based indicators, and web dashboard deployment.

4.2 Dataset Description

Stock data was collected via Alpha Vantage API for both Indian and international companies, focusing on Infosys, Wipro, and Amazon. The dataset contains OHLCV (Open, High, Low, Close, Volume) data and spans several years of trading activity to include both stable and volatile periods. Data for NASDAQ-listed companies (like Amazon, Apple, Infosys, Wipro) was converted from USD to INR using exchangerate-api to ensure consistency in output across markets. The data was initially stored in CSV format and transformed into supervised learning structure for LSTM training. A sample dataset is given below:

Date	Open	High	Low	Close	Volume
2025-05-12	18.86	18.92	18.50	18.65	16,249,496
2025-05-13	18.28	18.45	18.27	18.38	8,599,756
2025-05-14	18.53	18.54	18.39	18.46	8,648,376
2025-05-15	18.66	18.79	18.61	18.72	8,079,156
2025-05-16	18.34	18.39	18.11	18.31	12,914,272
2025-05-19	18.10	18.39	18.09	18.33	9,394,498
2025-05-20	18.27	18.30	18.16	18.24	7,065,541
2025-05-21	18.16	18.35	18.08	18.16	7,863,640
2025-05-22	17.91	18.14	17.80	18.04	8,344,995
2025-05-23	17.95	18.21	17.95	18.11	10,086,746
2025-05-27	18.29	18.51	18.27	18.49	9,014,442
2025-05-28	18.37	18.45	18.32	18.36	6,625,900
2025-05-29	18.53	18.59	18.39	18.46	8,527,908
2025-05-30	18.27	18.27	18.07	18.19	12,638,483
2025-06-02	18.07	18.21	18.00	18.19	8,475,451
2025-06-03	18.02	18.12	17.93	18.04	11,219,803
2025-06-04	18.04	18.10	17.82	17.85	7,541,704
2025-06-05	18.00	18.00	17.85	17.93	9,591,066
2025-06-06	18.19	18.25	18.15	18.20	6,426,289
2025-06-09	18.23	18.36	18.22	18.29	5,386,657

Table 03: Sample Dataset of Infosys (2025-05-12 to 2025-06-09)

- **Date:** The trading date in YYYY-MM-DD format.
- **Open:** The stock price at market opening on that day.
- **High:** The highest price reached by the stock during the trading day.
- **Low:** The lowest price the stock dropped to on the same day.
- **Close:** The final trading price when the market closed.
- **Volume:** The total number of shares traded on that day (in absolute numbers).

4.3 Data Acquisition Pipeline

- **Symbol Mapping:** The system accepts a company name and retrieves its stock symbol using the Alpha Vantage LISTING_STATUS function.
- **Historical Data Retrieval:** Daily OHLCV data is fetched using time-series endpoints.
- **Currency Conversion:** For USD-based stocks, a live USD to INR conversion rate is applied.
- **Data Storage:** Fetched data is stored locally in structured format for modelling.

This pipeline is automated and ensures up-to-date inputs, minimizing manual overhead.

4.4 Feature Engineering

The model integrates the following technical indicators:

- **SMA-20:** 20-day moving average to smooth price fluctuations
- **52-Week High/Low:** Captures long-term breakout levels
- **4-Week High/Low:** Reflects short-term trading windows
- **Prev_Close and Next_Open:** Used to maintain continuity and price gap learning
- **Volume and Trade Price:** Represent market activity and average value

All features are normalized using MinMaxScaler to maintain scale uniformity and prevent bias during model training.

```python
# --- Code Snippet 1: Feature engineering ---
data['SMA_20'] = data['Close'].rolling(window=20).mean()
data['52_Week_High'] = data['Close'].rolling(window=252).max()
data['52_Week_Low'] = data['Close'].rolling(window=252).min()
data['4_Week_High'] = data['High'].rolling(window=20).max()
data['4_Week_Low'] = data['Low'].rolling(window=20).min()
data['Prev_Close'] = data['Close'].shift(1)
data['Next_Open'] = data['Open'].shift(-1)
data['Trade_Price'] = (data['High'] + data['Low']) / 2
data.dropna(inplace=True)

# --- Code Snippet 2: Data preparation ---
features = ['Close', 'Volume', 'SMA_20', '52_Week_High', '52_Week_Low',
    '4_Week_High', '4_Week_Low', 'Prev_Close', 'Next_Open', 'Trade_Price']
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[features])

sequence_len = 200
train_len = int(len(scaled_data) * 0.8)

def create_sequences(data, sequence_len):
    X, y = [], []
    for i in range(sequence_len, len(data)):
        X.append(data[i - sequence_len:i, :])
        y.append(data[i, 0])  # Predicting 'Close'
    return np.array(X), np.array(y)

X, y = create_sequences(scaled_data, sequence_len)
X_train, y_train = X[:train_len], y[:train_len]
X_test, y_test = X[train_len:], y[train_len:]
```

4.5 Data Preparation for LSTM

The LSTM requires 3D input shaped as (samples, time steps, features). For this model:

- **Sequence Length:** 200 days
- **Features:** 10 engineered variables
- **Train-Test Split:** 80% training, 20% testing (chronologically ordered)

Sequences are built using a rolling window strategy, ensuring that each training input reflects the past 200 days of stock behaviour.

4.6 Model Architecture

A stacked LSTM design was implemented with the following layers:

- **LSTM Layer 1:** 512 units, return_sequences=True
- **Dropout Layer:** 0.01
- **LSTM Layer 2:** 256 units
- **Dropout Layer:** 0.01
- **Dense Layer:** 50 neurons
- **Output Layer:** 1 neuron for predicted closing price

Now the question is why did we choose this architecture?? Following are the reasons for this selection

- Stacking multiple LSTM layers improves abstraction capability for longer-term dependencies.
- The dense layer serves to combine outputs before final regression.
- Dropout provides regularization to prevent overfitting.

Let us visualize the architecture through the diagram below

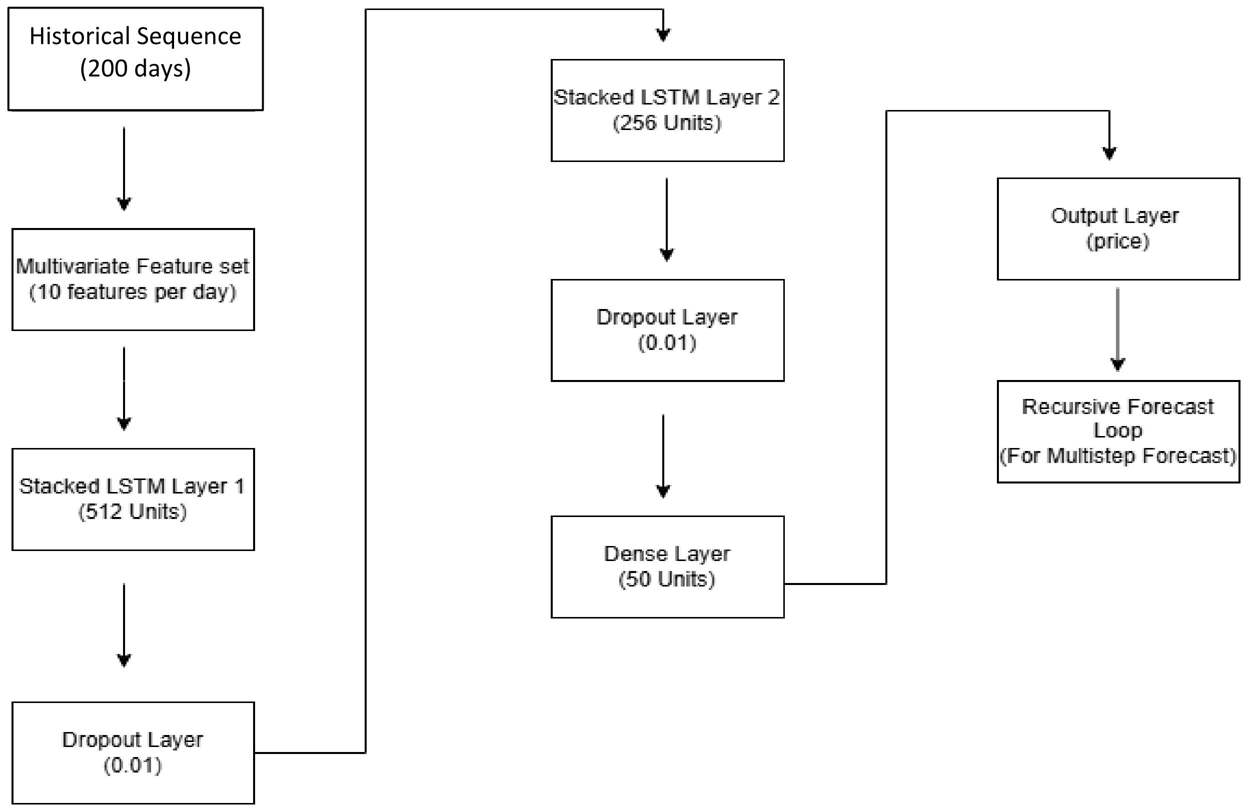


Figure 01: LSTM Model Architecture

- **Historical Sequence (200 Days):** The model looks back at the past 200 days of data to learn temporal patterns and predict the next day closing price.

- **Multivariate Feature Set (10 Features per Day):** Instead of just using closing price, 10 different features are used each day — such as SMA20, 52 weeks high/low, 4 weeks high/low, volume, prev_close, next_open, trade price. So, input shape = (200-time steps × 10 features)

- **Stacked LSTM Layer 1 (512 Units):** This is the first LSTM layer with 512 memory cells (units). This layer processes the temporal sequence and captures long-term dependencies.

- **Dropout (0.01):** Dropout is a regularization technique to prevent overfitting. 1% of neurons are randomly turned off during training to improve generalization.

- **Stacked LSTM Layer 2 (256 Units):** This is the second LSTM layer with 256 memory cells (unit). It further condenses and refines the temporal patterns passed from Layer 1.
- **Dropout (0.01):** Again, 1% dropout to prevent overfitting in the second LSTM layer.

- **Dense Layer (50 Neurons):** This layer is a fully connected layer with 50 neurons. This layer transforms LSTM output into a format suitable for final prediction.

- **Output Layer (Price):** This layer generates the predicted stock price for the next day or next step.

- **Recursive Forecast Loop (For Multi-Step Forecast):** The model predicts one step ahead, then feeds that prediction back into itself to predict the next step, and so on. Recursive forecasting may accumulate error over time, but it allows continuous prediction.

4.7 Model Configuration:

- **Loss Function:** Mean Absolute Error (MAE)
- **Optimizer:** Adam (adaptive learning rate)
- **Callbacks:** EarlyStopping (patience = 10 epochs), LearningRateScheduler (exponential decay)

4.8 Training Strategy

- **Epochs:** Up to 500 (typically converging within 100 epochs)
- **Batch Size:** 64

- **Validation Split:** 20% of the training data
- **Learning Rate:** Initial value 0.001, adjusted using decay schedule

Loss and validation curves were monitored to detect overfitting. The model checkpoints the best-performing weights using early stopping logic.

```
# --- Code Snippet 3 - LSTM model ---
model = Sequential()
model.add(LSTM(units=512, return_sequences=True, input_shape=(X_train.shape[1],
X_train.shape[2])))
model.add(Dropout(0.01))
model.add(LSTM(units=256, return_sequences=False))
model.add(Dropout(0.01))
model.add(Dense(units=50))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_absolute_error')

def lr_scheduler(epoch, lr):
    return max(0.0001, lr * 0.9)

early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
lr_schedule = LearningRateScheduler(lr_scheduler)

history = model.fit(
    X_train, y_train,
    validation_split=0.2,
    batch_size=64,
    epochs=500,
    callbacks=[early_stopping, lr_schedule]
)
```

4.9 Recursive Forecasting Logic

The system employs a recursive strategy for multi-step forecasting:

- Predicts next day's price using current sequence
- Feeds prediction back into the sequence to predict the following day
- Continues this loop for N-day forecasting

This allows simulation of future scenarios over user-specified horizons like 10, 30, or 180 days, though forecast accuracy reduces over longer spans.

```
# --- Code Snippet 4 – Recursive Forecasting Logic ---
def predict_future_prices(model, last_sequence, scaler, original_df,
future_type="year", n=1, start_date=None, end_date=None):
    # Define number of prediction steps
    if future_type == "year":
        steps = 252
        label = "Next 1 Year"
    elif future_type == "months":
        steps = int(n * 21)
        label = f"Next {n} Month(s)"
    elif future_type == "weeks":
        steps = int(n * 5)
        label = f"Next {n} Week(s)"
    elif future_type == "days":
        steps = n
        label = f"Next {n} Day(s)"
    elif future_type == "custom":
        all_dates = [original_df.index[-1] + timedelta(days=i) for i in range(1,
366)]
        all_dates = [d for d in all_dates if d.weekday() < 5]
        future_dates = [d for d in all_dates if start_date <= d <= end_date]
        steps = len(future_dates)
        label = f"Custom Range: {start_date.strftime('%Y-%m-%d')} to
{end_date.strftime('%Y-%m-%d')}"
    else:
        raise ValueError("Invalid future_type. Choose from 'year', 'months',
'weeks', 'days', or 'custom'.")

    current_sequence = last_sequence.copy()
    predictions = []

    for _ in range(steps):
        next_pred = model.predict(current_sequence.reshape(1,
current_sequence.shape[1], current_sequence.shape[2]), verbose=0)
        predictions.append(next_pred[0, 0])
        current_sequence = np.roll(current_sequence, -1, axis=1)
        current_sequence[0, -1, 0] = next_pred[0, 0]

    pred_scaled = np.zeros((len(predictions), current_sequence.shape[2]))  # match
feature count
    pred_scaled[:, 0] = predictions
    pred_scaled = scaler.inverse_transform(pred_scaled)[:, 0]

    usd_to_inr = get_usd_to_inr()
    pred_scaled_inr = pred_scaled * usd_to_inr

    if future_type != "custom":
        last_date = original_df.index[-1]
        future_dates = [last_date + timedelta(days=i) for i in range(1, steps +
30)]
        future_dates = [d for d in future_dates if d.weekday() <
5][:len(pred_scaled_inr)]
```

```
    # Plot
    plt.figure(figsize=(12, 6))
    plt.plot(future_dates, pred_scaled_inr, label='Predicted Price (INR)',
color='blue')
    plt.title(f'Stock Price Forecast - {label} (in INR)')
    plt.xlabel('Date')
    plt.ylabel('Price (INR)')
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

4.10 Evaluation Metrics

The model's predictions are assessed using:

- **MAE (Mean Absolute Error):** Measures average magnitude of error.

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

- **MAPE (Mean Absolute Percentage Error):** Reflects error relative to actual price. Good for short term trends but can be bad for long term trends.

$$\text{MAPE} = \frac{MAE}{\frac{1}{n} \sum_{i=1}^{n} y_i} \times 100\ \%$$

- **R² Score (Coefficient of Determination):** Indicates variance explained by the model. Good for long term trends can be worse for long term trends.

$$R^2 = 1 - \left[\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \right]$$

```
# --- Code Snippet 5 - Evaluation ---
train_rmse = np.sqrt(mean_squared_error(y_train_rescaled, train_pred_rescaled))
train_mae = mean_absolute_error(y_train_rescaled, train_pred_rescaled)
test_rmse = np.sqrt(mean_squared_error(y_test_rescaled, test_pred_rescaled))
test_mae = mean_absolute_error(y_test_rescaled, test_pred_rescaled)
```

```
train_mae_percentage = (train_mae / np.mean(y_train_rescaled)) * 100
test_mae_percentage = (test_mae / np.mean(y_test_rescaled)) * 100

print(f"Training MAE %: {train_mae_percentage:.2f}%")
print(f"Testing MAE %: {test_mae_percentage:.2f}%")

train_r2 = r2_score(y_train_rescaled, train_pred_rescaled)
test_r2 = r2_score(y_test_rescaled, test_pred_rescaled)
```

4.11 Hyperparameter Tuning Strategy

Due to limited access to advanced computing infrastructure, manual tuning was used:

- **Window Size:** 100, 150, 200 days tested. 200 days is selected
- **LSTM Units:** 64, 128, 256, 512 tested for each layer. 512 and 256 are selected across two layers
- **Batch Sizes:** 32, 64, 128 tested. 64 is selected
- **Dropout Rate:** 0.01-0.3 tested. 0.01 is selected
- **Early Stopping Patience:** 5-15 tested. 10 is selected
- **Learning Rate Schedular:** Initially 0.0001. Later gets adjusted during training for exponential decay
- **Epochs:** Initially with small values but now 500 is selected.

While exhaustive tuning via grid or Bayesian search was not feasible, this manual iterative approach provided a strong trade-off between model complexity and training feasibility.

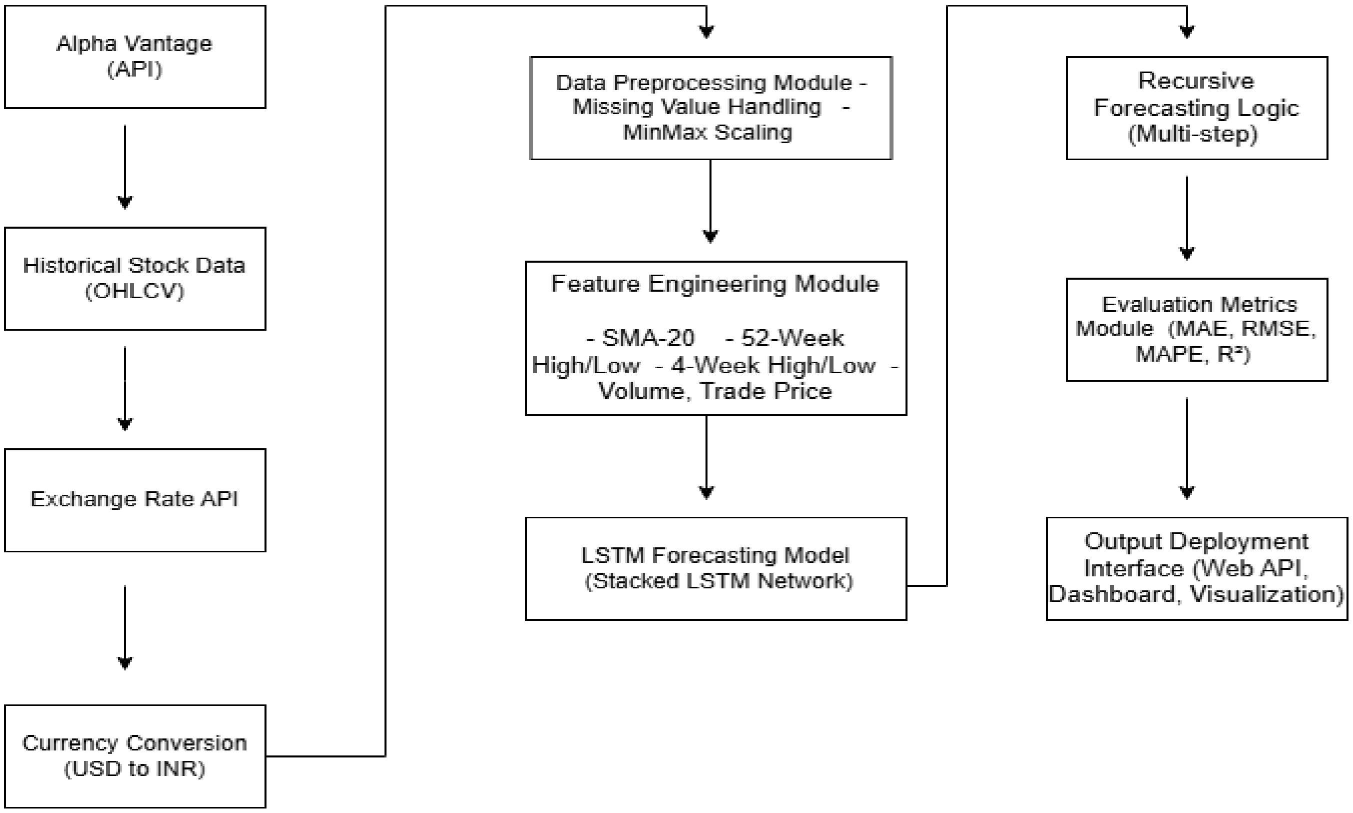
4.12 Deployment Architecture

Though full-scale deployment is a future goal, the backend has been structured with deployment in mind:

- Code developed in Jupyter/Colab
- Real-time APIs integrated (Alpha Vantage and ExchangeRate)
- Model prediction exposed via microservices (Flask or FastAPI)
- Frontend was created using basic HTML/CSS/JS
- Charts and forecasts are generated using Matplotlib

Some snaps of the website created has been shared in the annexure section.

4.13 System Design



Alpha Vantage (API)
Historical Stock Data (OHLCV)
Exchange Rate API
Currency Conversion (USD to INR)
Data Preprocessing Module - Missing Value Handling - MinMax Scaling
Feature Engineering Module - SMA-20 - 52-Week High/Low - 4-Week High/Low - Volume, Trade Price
LSTM Forecasting Model (Stacked LSTM Network)
Recursive Forecasting Logic (Multi-step)
Evaluation Metrics Module (MAE, RMSE, MAPE, R²)
Output Deployment Interface (Web API, Dashboard, Visualization)

Figure 02: System Design

- **Alpha Vantage (API):** This API fetches historical stock data (OHLCV) for companies like Apple, Amazon, Infosys, and Wipro.
- **Historical Stock Data (OHLCV):** It represents the collected data from Alpha Vantage. This data includes all essential fields for financial analysis. It is the input base for all subsequent processes.
- **ExchangeRate API:** Supplies live USD to INR conversion rates to standardize international stock values for Indian investors.
- **Currency Conversion (USD to INR):** Applies exchange rates to all USD-based stock data to make predictions interpretable in INR.
- **Data Preprocessing Module:** Handles missing values and normalizes the data using MinMaxScaler to prepare for LSTM input.

- **Feature Engineering Module:** It creates meaningful technical indicators as input features for the LSTM. The features used are SMA-20, 52-Week High/Low, 4-Week High/Low, Volume and Trade Price

- **LSTM Forecasting Model (Stacked LSTM Network):** A deep LSTM architecture processes multivariate sequences to capture both short- and long-term stock movement patterns.

- **Recursive Forecasting Logic (Multi-Step):** Enables forecasting for multiple future time steps by feeding previous outputs back into the model.

- **Evaluation Metrics Module:** Measures model accuracy using:
  - MAE (absolute error)
  - MAPE (percentage error)
  - R^2 Score (fit quality)

- **Output Deployment Interface:** It exposes predictions to end-users through:
  - **Web API (Flask/FastAPI):** Backend service to send predictions to frontend.
  - **Dashboard/Visualization:** HTML/JS/CSS frontend to allow user interaction and plot visualization.

CHAPTER 5
TESTING AND RESULT DISCUSSIONS

5. Testing and Result Discussions

5.1 Model Training and Convergence Behaviour

The training process of the LSTM model demonstrated stable convergence across all datasets and forecast horizons. Early stopping was consistently triggered within 50–90 epochs, indicating that the model effectively avoided overfitting while preserving generalization.

Training and validation loss curves exhibited smooth, parallel trends, with no divergence—signifying that the model was not merely memorizing the data. Architectures using larger LSTM layers (512 → 256) and carefully selected sequence lengths (200 days) captured long-term dependencies while avoiding excessive model complexity.

Training loss plateaued steadily after an initial drop, while validation loss remained stable, confirming robust model behaviour across both Indian and NASDAQ datasets.

5.2 Baseline Model Performance

Three traditional forecasting models—ARIMA, Moving Average, and Linear Regression—were implemented for benchmarking.

5.2.1 Three Traditional Forecasting Models:

5.2.1.1 ARIMA Model

Despite its popularity, the ARIMA model underperformed due to its linearity and assumptions of stationarity. Although auto_arima tuning offered reasonable short-term results, performance degraded over longer horizons. It consistently lagged or underestimated rapid market changes, particularly during earnings seasons and macroeconomic shocks.

5.2.1.2 Moving Average Model

As a naive baseline, the Moving Average model provided basic trend smoothing. However, it lacked adaptability to price reversals, sectoral shifts, or market volatility. Error metrics dropped sharply beyond

5–10 days, making it unsuitable for multi-step forecasting.

5.2.1.3 Linear Regression Model

This model used multivariate features and outperformed ARIMA and Moving Average for short horizons. However, its linear structure couldn't capture the non-linear interactions between technical indicators and price. Bias increased with longer forecast windows.

5.2.2 Performance Evaluation of Classical Forecasting Models:

Model	Dataset Used	Forecast Horizon	MAPE	Remarks
ARIMA	*Amazon*	1 year	10.02%	High error due to assumption of linearity hence worst performance overall.
	Infosys	1 year	7.16%	
	Wipro	1 year	12.56%	
Moving Average	*Amazon*	1 year	8.14%	Only captures short-term trends. Better than ARIMA but lacks pattern learning.
	Infosys	1 year	5.58%	
	Wipro	1 year	10.85%	
Linear Regression	*Amazon*	1 year	6.85%	Assumes linear relationship, hence it is less accurate and struggles with non-linear fluctuations
	Infosys	1 year	4.97%	
	Wipro	1 year	9.69%	
LSTM	*Amazon*	1 year	2.26%	Captures temporal and nonlinear dependencies and has reliable performance; chosen for superior accuracy
	Infosys	1 year	2.12%	
	Wipro	1 year	6.65%	

Table 04: Performance Evaluation of Different Models

Among the evaluated models, **ARIMA** performed the worst, primarily because it assumes linearity and stationarity—conditions rarely met in real stock data [11]. It struggled to adapt during periods of volatility or trend shifts, resulting in high MAPE values (e.g., 10.02% for Amazon, 7.16% for Infosys, and 12.56% for Wipro).

The **Moving Average** model was slightly better but still inadequate. It offers simple trend smoothing without any pattern recognition capability and fails to capture reversals, momentum, or volatility, making it unreliable for anything beyond very short-term forecasts [13].

Linear Regression outperformed both ARIMA and Moving Average, especially for short-term predictions. However, its assumption of a fixed linear relationship between features and output limited its ability to model non-linear financial dynamics, leading to increasing errors over longer horizons [14].

In contrast, the **LSTM** model delivered the best results. It captured both short- and long-term dependencies using memory cells and handled multivariate, non-linear time-series data effectively [12]. With much lower MAPE values (e.g., 2.26% for Amazon, 6.65% for Wipro, and 2.12% for Infosys), LSTM proved to be the most robust and accurate model. Its suitability for recursive forecasting and adaptability to market patterns made it the clear choice for this project [15].

5.3 LSTM Model Performance

The LSTM model consistently outperformed baselines across companies, sectors, and forecast durations due to its ability to capture sequential dependencies and use multivariate features effectively.

5.3.1 Short-Term Forecasting (10-90 Day Horizon)

- **MAPE:** 0.98% – 4%
- **$\mathbf{R^2}$ score:** 0.40 – 0.90 (may get worse due to noise)

Short-term forecasts closely tracked price movements, as a result the mape value was less in most of the cases. However, R^2 score value may get worse due to presence of heavy noise in the short-term data. The plot may get uneven. But in terms of prediction, recursive forecasts work best for short term forecasting. Error accumulation over time is less.

5.3.2 Long-Term Forecasting (90 and above Days Horizon)

- **MAPE:** 0.98% – 6.2%
- **$\mathbf{R^2}$ score:** 0.85 – 0.96

For long term trends, mape value can at times be huge like 6-7% but R^2 Score are generally good as the noise gets spread smoothly in the plot hence it becomes easy to capture the trends. However recursive forecast can perform poor in long term forecasts due to high error accumulation over time.

5.4 Forecast/Prediction Visualization Description

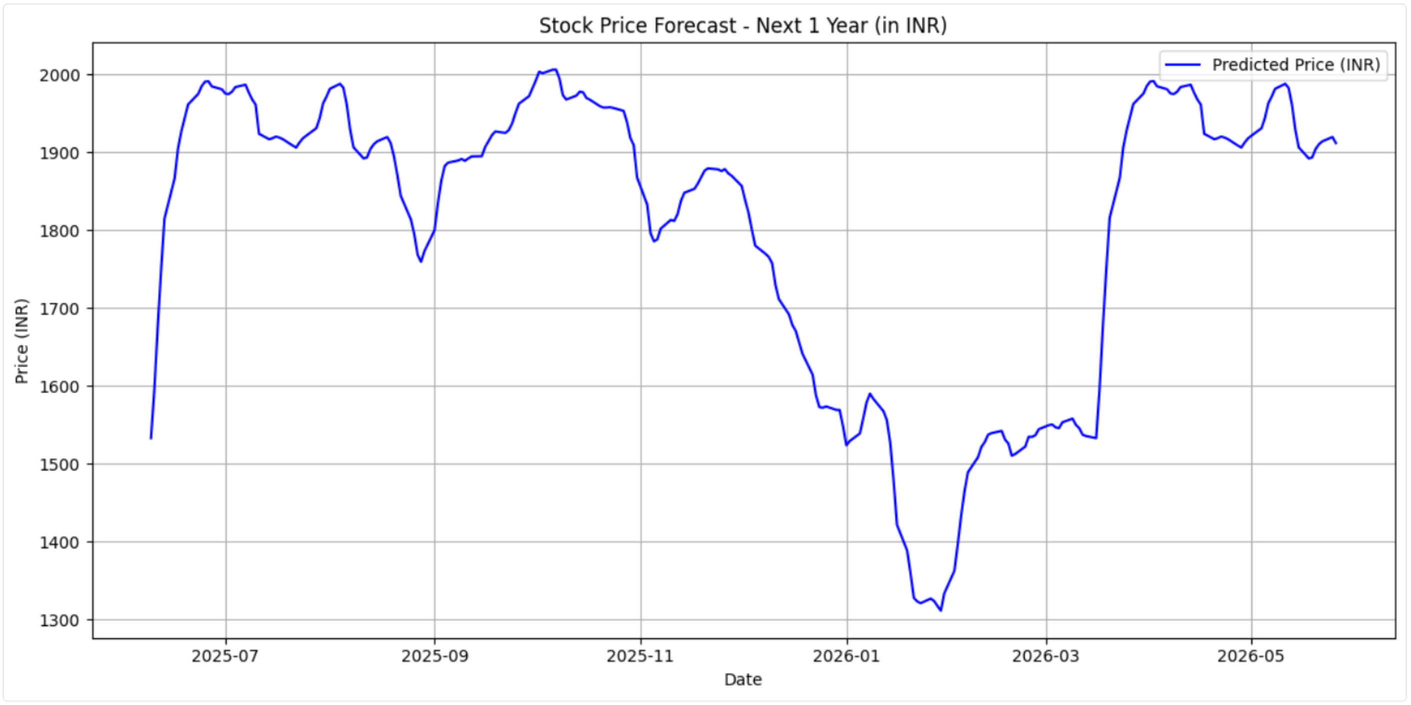


Figure 03: Stock Price Forecast of Infosys (Next 1 Year)

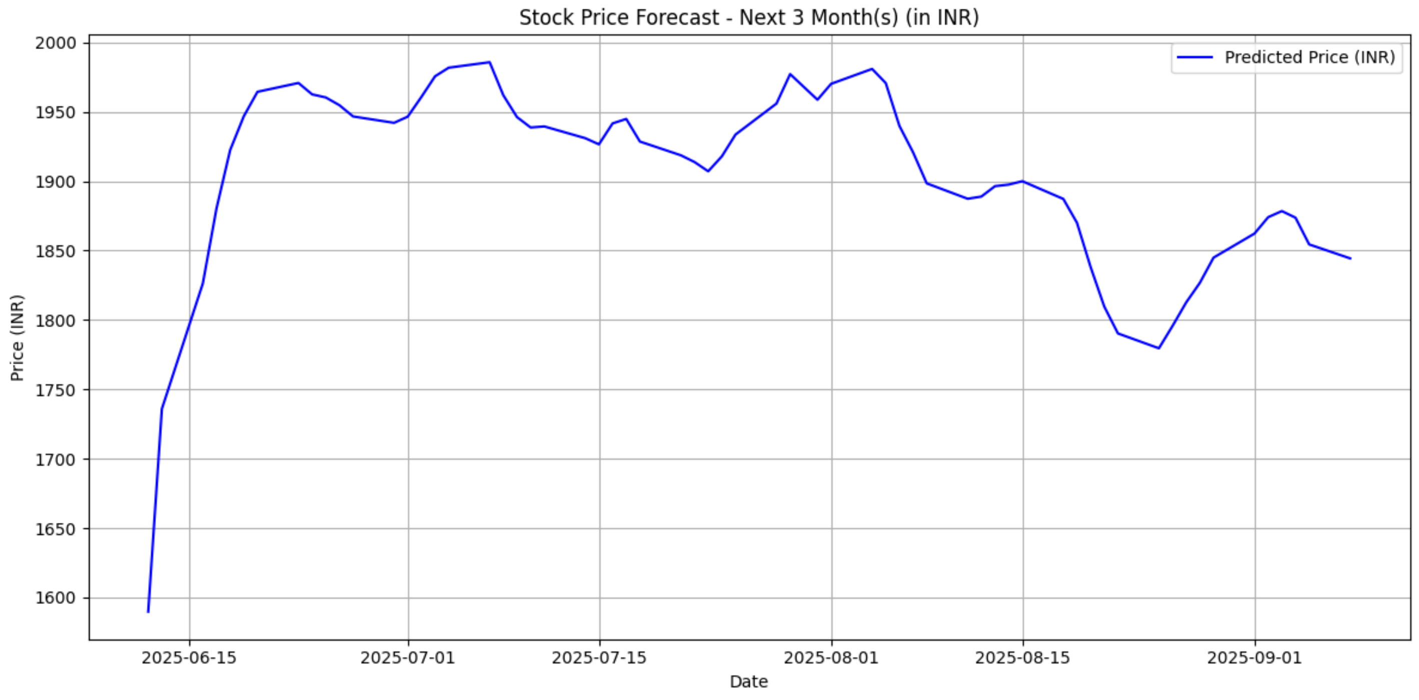


Figure 04: Stock Price Forecast of Infosys (Next 3 Months)

Key insights from visual plots:

- **Short-Term:** Forecasts closely tracked actual prices; tight confidence bands.
- **Long-Term:** Forecast bands widened post 90–120 days due to error propagation.

5.5 Recursive Forecast Error Accumulation

Recursive forecasting uses predicted outputs as future inputs, which leads to error buildup:

- Even small one-step errors accumulate in long-term forecasting.
- A 2% one-step error can evolve into 5–6% error by day 120–180.
- This is a mathematical limitation of recursive logic.

Why recursive forecasting worsens over time. This is due to feedback loops using imperfect predictions, compounding deviation with each step.
To address recursive forecasting degradation, future improvements may include:

- **Teacher-forcing:** Uses ground-truth outputs during training to stabilize sequence learning and reduce drift.

- **Sequence-to-sequence architectures:** Uses an encoder-decoder model to predict entire sequences in one step, improving stability.

- **Attention-based models:** Focuses on the most relevant parts of the input data for each prediction, improving accuracy in long and noisy sequences.

5.6 Failure Case Analysis

Despite strong performance, LSTM had limitations under specific conditions:

- **High-Volatility Phases:** Events like COVID-19 (2020) or inflation spikes (2022) led to lagging predictions due to unseen patterns.
- **Earnings Surprises:** Unexpected quarterly results caused sharp price jumps that the model could not anticipate using only technical inputs.
- **Macroeconomic Announcements:** Interest rate changes or policy news caused non-technical structural shifts outside model scope.

- **Short Term Trends:** Due to heavy noise, it becomes difficult to capture short term trends and visualise them
- **Recursive Horizon Limitations:** Errors amplified exponentially beyond 90 days, increasing variance.

5.7 Practical Interpretation for Investors

From an investor's perspective, the system provides:

- **Short-Term Forecasting:** The model demonstrates strong accuracy in the short term (10–30 days), which could be particularly useful for tactical decision-making. While price values can have less deviation, due to heavy noise, it may be difficult to capture short term trends. Hence it's essential to take cautious decisions while working with extreme short trends with noise.

- **Long-Term Forecasting Limitations:** Beyond the 90-day mark, forecast reliability noticeably drops. This suggests that predictions over longer horizons should be used with caution and ideally supplemented by other analytical tools.

- **Risk Assessment Tools:** Features like error margins and prediction intervals are valuable, as they provide a clearer picture of potential outcome ranges—helpful for visualizing risk and planning under different scenarios.

- **Future Scalability:** Current framework could be expanded to include additional data sources, such as sentiment analysis or macroeconomic indicators, to enhance its predictive power and overcome current limitations.

5.8 Visualisation of Actual vs Predicted plots

To assess the predictive performance of the LSTM model, we plot the actual stock prices alongside the model's predicted prices. These comparisons help in visually evaluating how well the model captures the underlying trend and short-term fluctuations. Multiple graphs are presented below, each corresponding to a different stock or forecasting time window. A closer alignment between the actual and predicted curves indicates higher accuracy of the model.

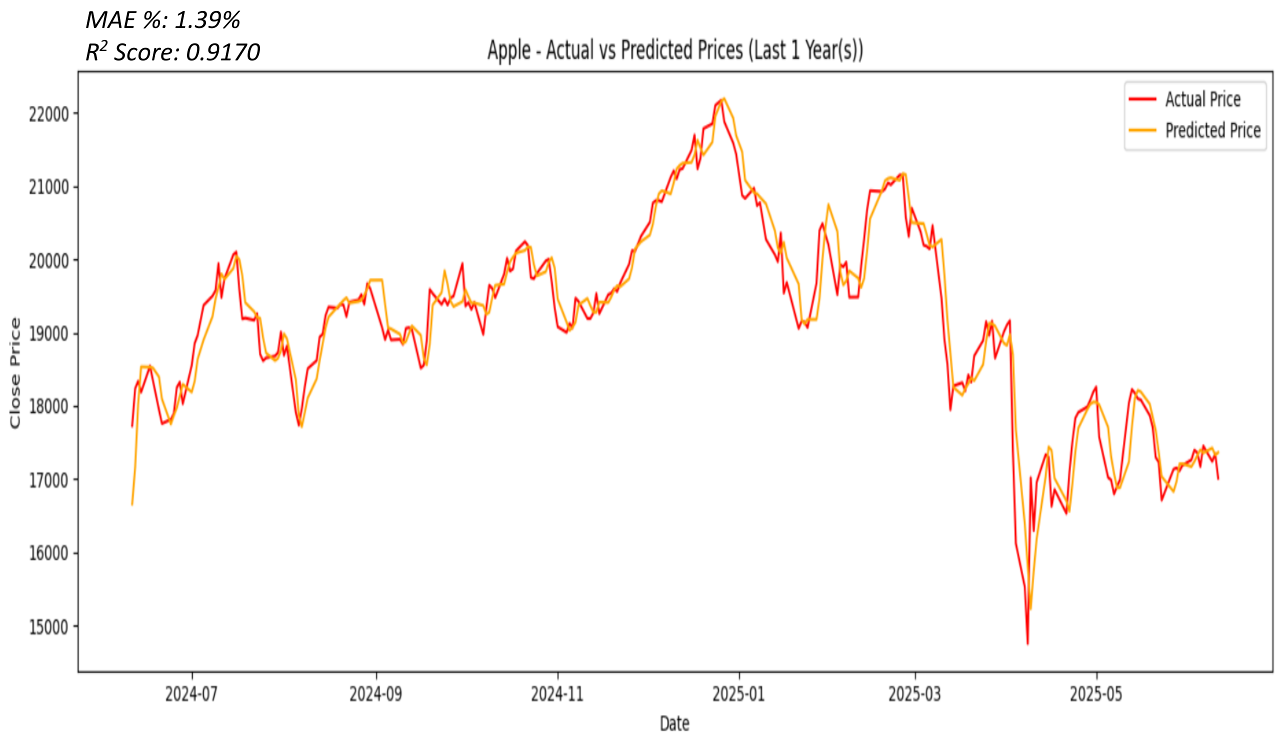


Figure 05: Actual vs Predicted Stock Prices (Last 1 Year) of Apple

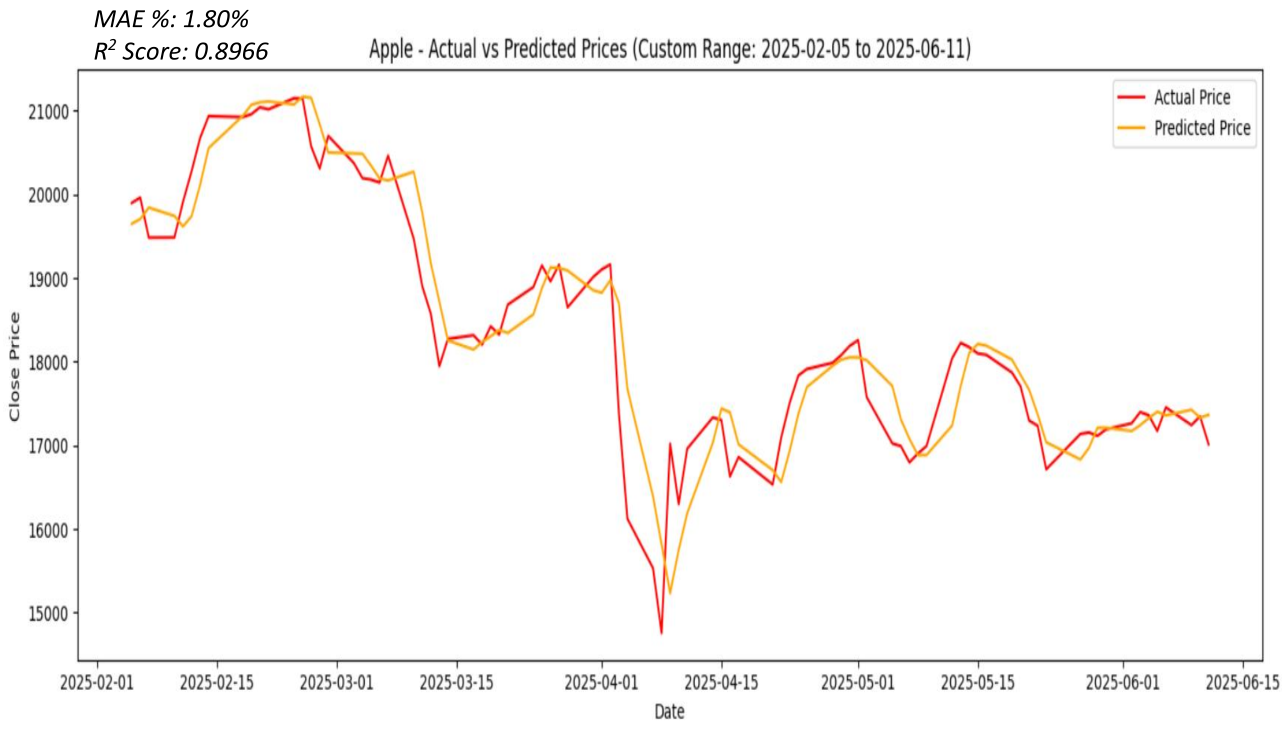


Figure 06: Actual vs Predicted Stock Price (2025-02-05 to 2025-06-11) of Apple

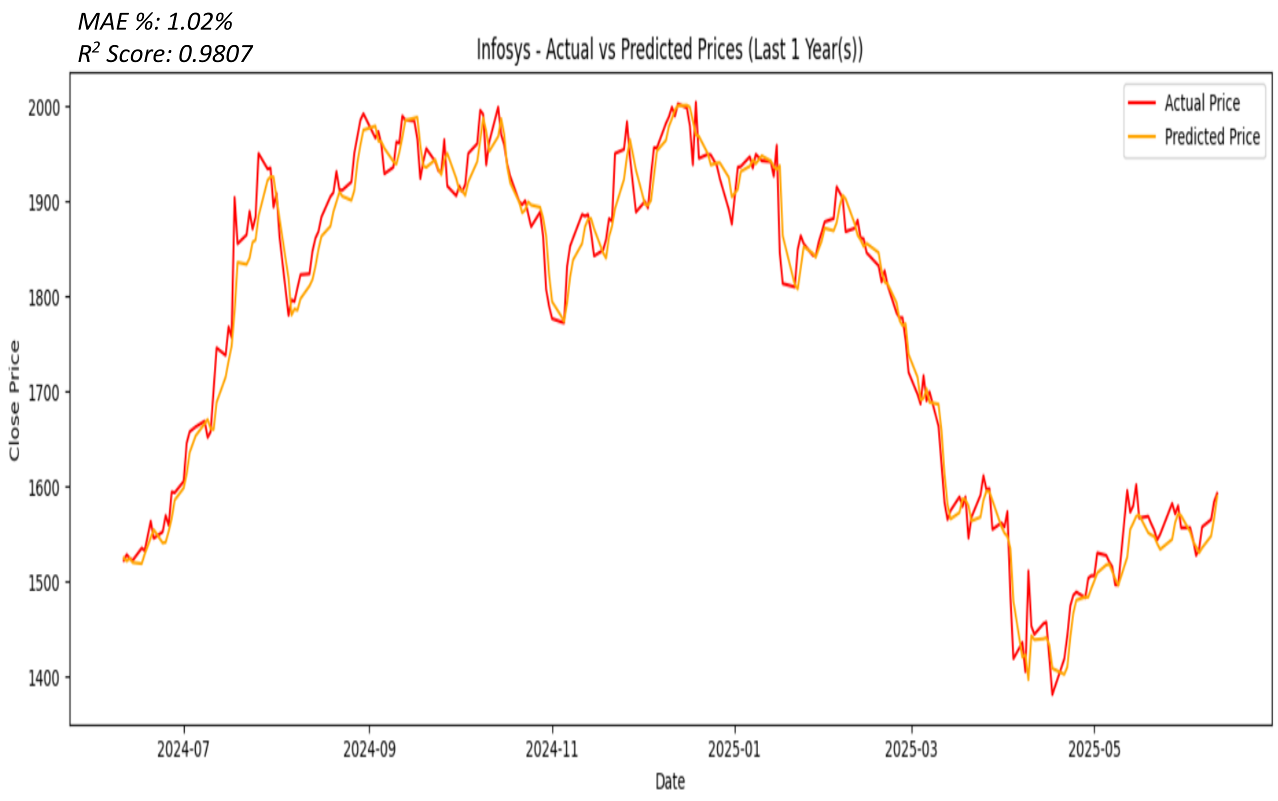


Figure 07: Actual vs Predicted Stock Price (Last 1 Year) of Infosys

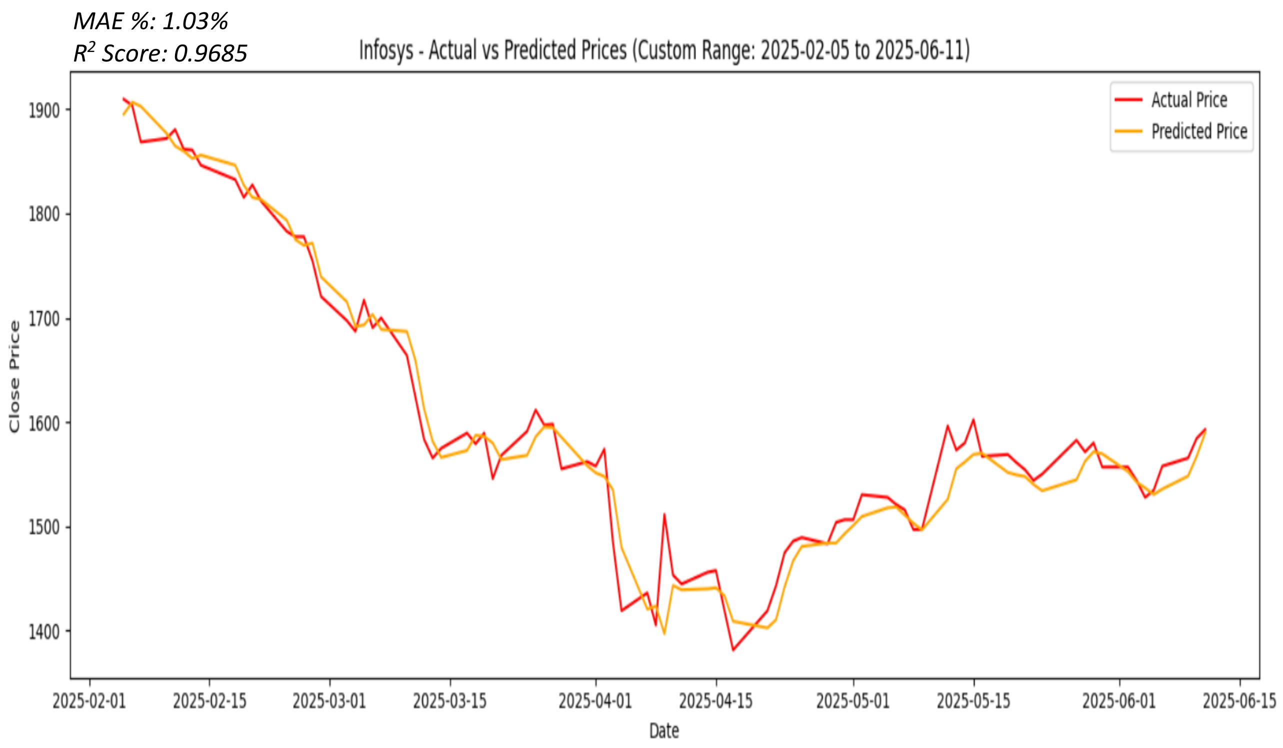


Figure 08: Actual vs Predicted Stock Price (2025-02-05 to 2025-06-11) of Infosys

MAE %: 3.84%
R^2 Score: 0.9756

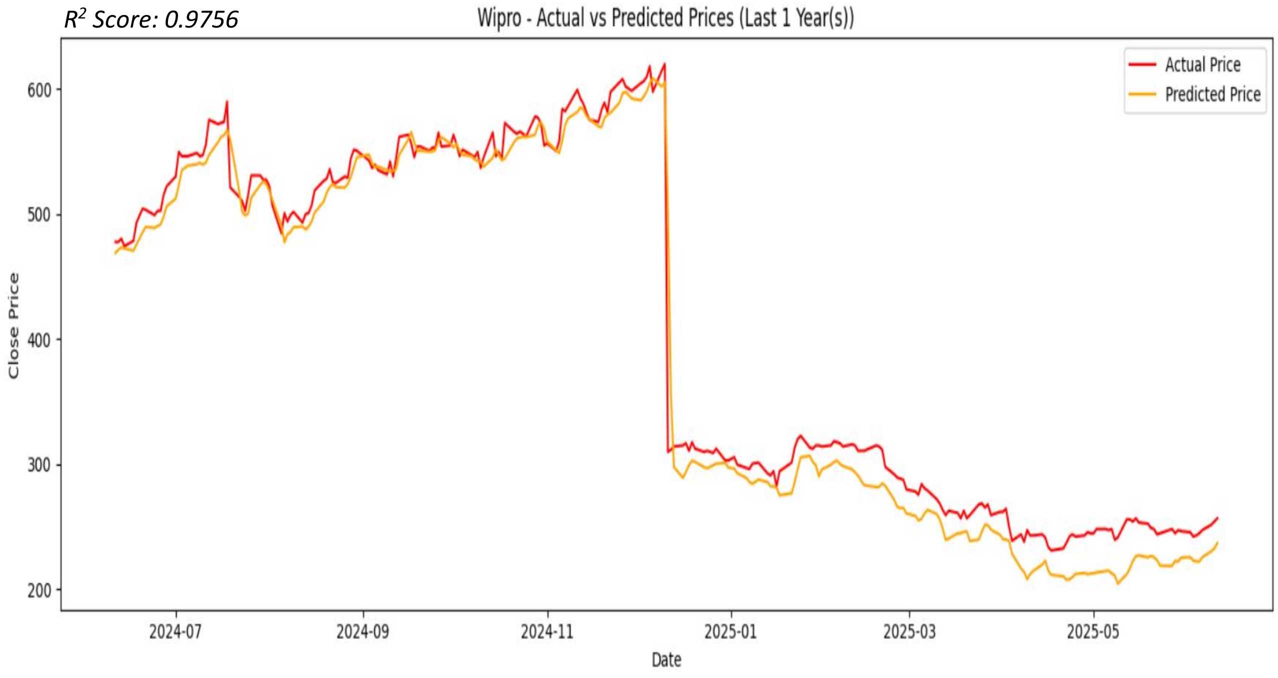


Figure 09: Actual vs Predicted Stock Price (Last 1 Year) of Wipro

MAE %: 6.91%
R^2 Score: 0.9291

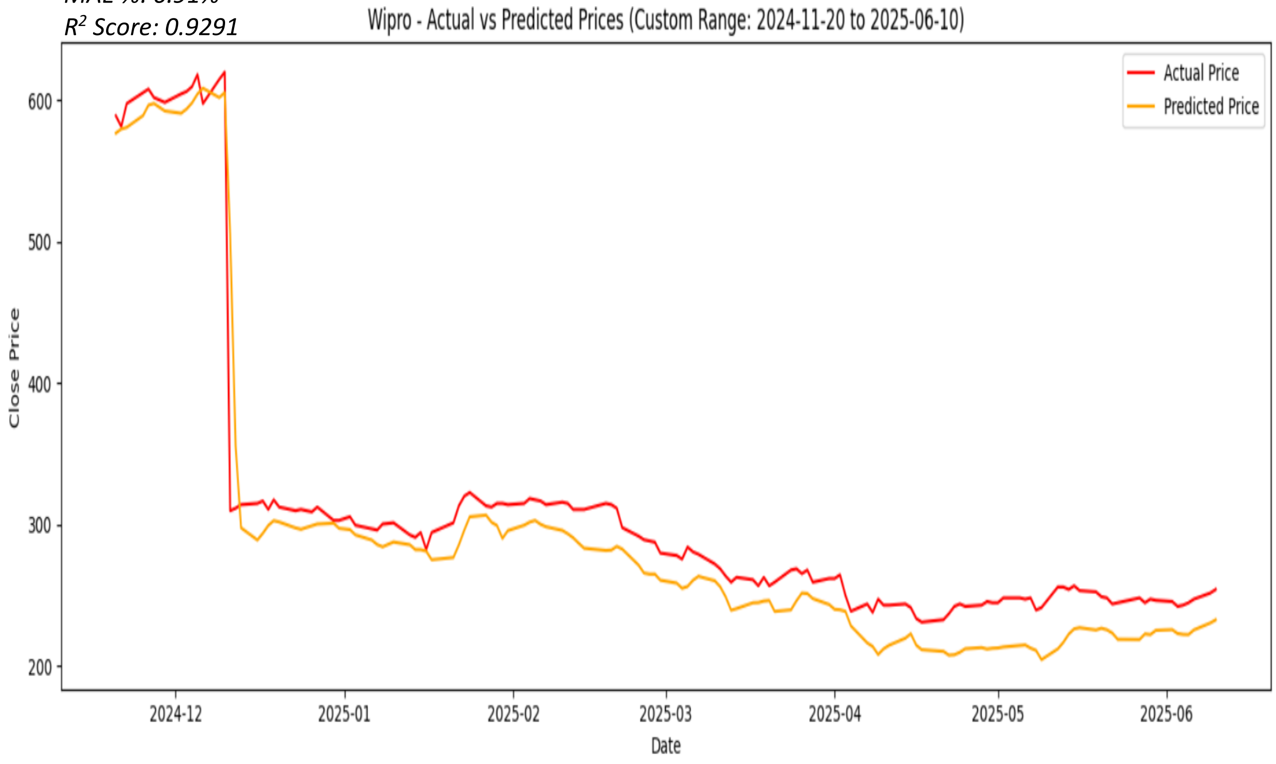


Figure 10: Actual vs Predicted Stock Price (2025-02-05 to 2025-06-11) of Wipro

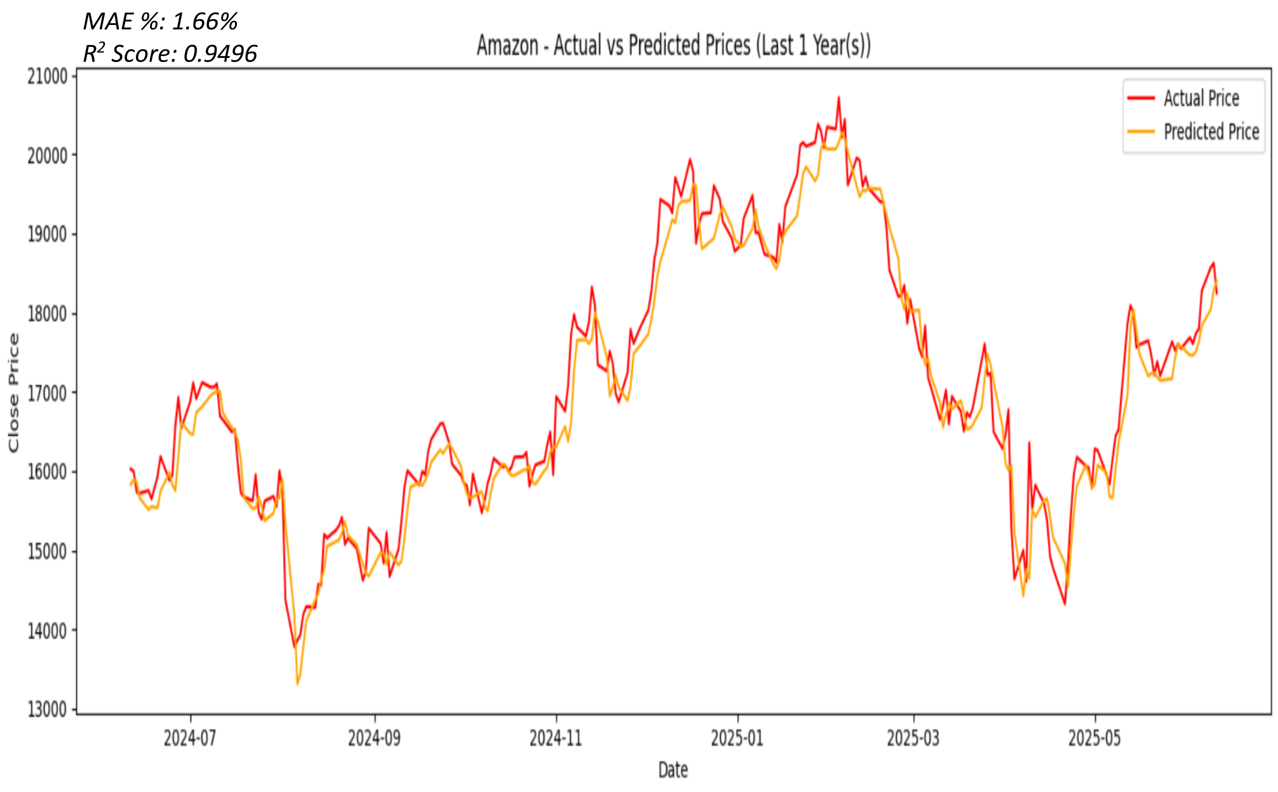


Figure 11: Actual vs Predicted Stock Price (Last 1 Year) of Amazon

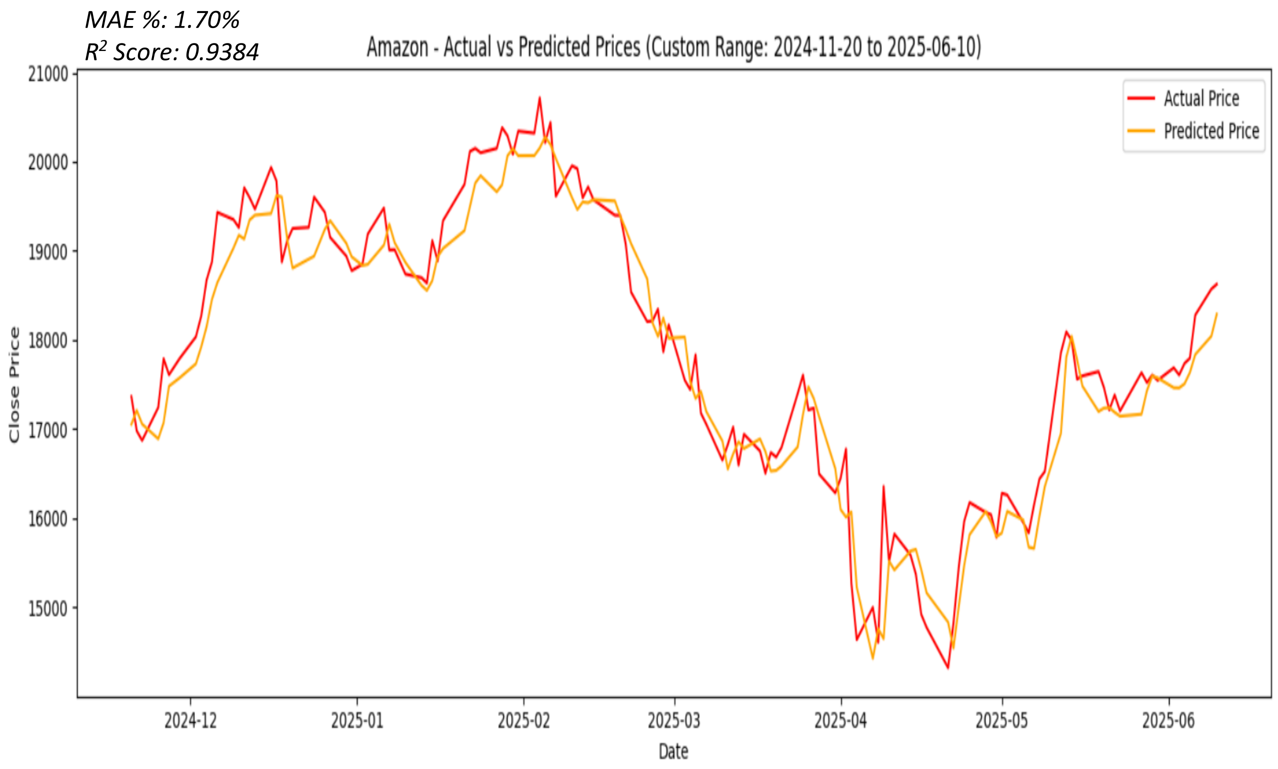


Figure 12: Actual vs Predicted Stock Price (2025-02-05 to 2025-06-11) of Amazon

CHAPTER 6
CONCLUSION AND FUTURE WORK

6. Conclusion and Future Work

6.1 Conclusion

This project developed a comprehensive stock price forecasting framework by integrating multivariate technical indicators with deep learning architectures, particularly Long Short-Term Memory (LSTM) networks. The system was designed to predict stock prices across both Indian and international markets, targeting companies such as Apple, Infosys, Amazon, Wipro. Automated data acquisition was handled using the Alpha Vantage API, with currency normalization implemented to enable consistent INR-based predictions.

A variety of technical indicators — including moving averages, 52-week highs/lows, trading volume, average trade prices, previous close prices, and next open prices — were engineered to build a robust multivariate input dataset. These indicators effectively captured both momentum and mean-reversion patterns common in financial time series.

The LSTM model consistently outperformed classical models like ARIMA, Moving Average, and Linear Regression across all forecast horizons. Short-term forecasts (10–30 days) achieved high precision, while long-term forecasts experienced some degradation in accuracy due to recursive error accumulation — a known limitation of sequential models. Nevertheless, the LSTM maintained reliable directional forecasting even during moderately volatile periods.

The comparative performance analysis reinforced the LSTM's advantage in modeling complex, non-linear temporal relationships and long-term dependencies in stock price movements. The system was also designed with a modular and scalable architecture, supporting potential deployment through APIs and dashboards for real-world financial decision-making.

Despite its overall success, the model does exhibit a few limitations:

- Exclusive reliance on historical technical data without incorporating macroeconomic, geopolitical, or sentiment-based inputs.
- Inability to predict black swan events or major market disruptions.
- Recursive multi-step forecasting introduced progressive error accumulation for longer prediction horizons.

Still, the system provides a solid foundation for academic and applied financial forecasting, blending technical insight with practical utility.

6.2 Future Scope

The current system presents a robust foundation for stock price forecasting using LSTM and technical indicators. However, to improve its accuracy, real-world applicability, and adaptability to changing market dynamics, the following future enhancements are proposed:

6.2.1 Sentiment Analysis Integration

Market sentiment, often reflected in news headlines, social media, earnings announcements, and public opinion, significantly influences stock prices. Technical indicators alone may not capture this "psychological" aspect of the market, especially during events like product launches, scandals, or macroeconomic announcements. Few advantages of this integration are as follows:

- Incorporating sentiment scores from platforms like Twitter, Reddit, and financial news APIs (e.g., Google News, Yahoo Finance) will allow the model to account for investor emotions and crowd psychology.
- This will improve prediction accuracy during high-volatility or news-driven periods (e.g., earnings season, policy changes).

6.2.2 Macroeconomic Indicator Inclusion

Stock prices are not solely driven by company-level technicals; broader economic indicators like GDP growth, inflation, unemployment, and interest rates also play a major role in shaping market trends. Following are the reasons it is needed:

- Integrating macroeconomic data will help capture long-term structural trends and improve forecasting during global or national financial shifts.
- Makes the model suitable for policy impact analysis and portfolio-level strategic planning.

6.2.3 Attention-Based Architectures (e.g., Transformers, DA-RNN, Informer)

While LSTM handles sequential data well, it processes each timestep in order, which can be computationally expensive and may lose focus over long sequences. Attention-based models overcome this by dynamically focusing on the most relevant input features and time points. Following are the advantages of this architecture:

- Attention mechanisms can improve forecasting accuracy, especially for long-term predictions, by identifying the most influential past data points.
- They reduce training time compared to LSTM while enhancing performance on longer sequences.

6.2.4 Hybrid Modelling (LSTM + Classical + Reinforcement Learning)

No single model is universally best. Combining multiple approaches can mitigate the weaknesses of each and leverage their respective strengths. Following are the advantages of this modelling:

- Hybrid models (e.g., LSTM + ARIMA, or LSTM + Reinforcement Learning) allow both pattern recognition and rule-based forecasting.
- Can be tailored for adaptive trading strategies where reinforcement agents dynamically choose between models based on market regimes.

6.2.5 Portfolio-Level Forecasting and Optimization

Investors typically manage portfolios rather than single stocks. A system that predicts individual stock trends should evolve to manage inter-stock correlations, diversification, and risk allocation. It is important as it

- Supports multi-stock forecasting, sector-wise analysis, and sharpe ratio optimization.
- Assists investors in asset rebalancing and strategy backtesting based on predicted trends.

6.2.6 Risk Management Integration

Prediction alone is insufficient for real-world investing. Traders and institutions also require risk quantification to manage exposure and uncertainty. Following are the advantages:

- Integrating tools like Value-at-Risk (VaR), Monte Carlo simulations, and prediction intervals adds confidence estimates and probabilistic bounds to forecasts.
- Facilitates regulatory compliance and portfolio stress testing under extreme market conditions.

References

[1] P. Tran Phuoc, P. T. K. A. Pham Thi Kim Anh, P. H. T. Phan Huy Tam, and C. V. Nguyen, "Applying machine learning algorithms to predict the stock price trend in the stock market – The case of Vietnam," *Humanities and Social Sciences Communications*, vol. 11, no. 393, 2024, doi: 10.1057/s41599-024-02807-x.

[2] M. Hiransha, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "NSE stock market prediction using deep-learning models," in *Proc. Procedia Computer Science*, vol. 132, pp. 1351–1362, 2018, doi: 10.1016/j.procs.2018.05.050.

[3] N. Darapaneni, A. R. Paduri, H. Sharma, M. Manjrekar, N. Hindlekar, P. Bhagat, U. Aiyer, and Y. Agarwal, "Stock price prediction using sentiment analysis and deep learning for Indian markets," *arXiv preprint*, arXiv:2204.05783, 2022, doi: 10.48550/arXiv.2204.05783.

[4] A. W. Li and G. S. Bastos, "Stock market forecasting using deep learning and technical analysis," *IEEE Access*, vol. 8, pp. 185692–185705, 2020, doi: 10.1109/ACCESS.2020.3030226.

[5] Z. Liu, S. Xu, Z. Zhang, and Y. Liu, "Informer: Efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artificial Intelligence*, 2023, doi: 10.1609/aaai.v35i12.17485.

[6] Q. Feng, H. Wang, X. Yuan, et al., "A dual-stage attention-based recurrent neural network for stock prediction," *IEEE Access*, vol. 9, pp. 9145–9155, 2021, doi: 10.1109/ACCESS.2021.3053853.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[8] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. San Francisco, CA, USA: Holden-Day, 1970.

[9] Alpha Vantage Inc., "Alpha Vantage API Documentation," [Online]. Available: https://www.alphavantage.co/documentation/. [Accessed: 08-Jun-2025].

[10] ExchangeRate-API, "ExchangeRate API Documentation," [Online]. Available: https://www.exchangerate-api.com/. [Accessed: 08-Jun-2025].

[11] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: Forecasting

and control (5th ed.). Wiley.

[12] 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

[13] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and practice (2nd ed.). OTexts. https://otexts.com/fpp2/

[14] Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. PLOS ONE, 13(3), e0194889. https://doi.org/10.1371/journal.pone.0194889

[15] Zhang, X., Aggarwal, A., & Dey, L. (2020). Stock price prediction using deep learning models. Procedia Computer Science, 167, 2063–2070. https://doi.org/10.1016/j.procs.2020.03.252

Annexure

This annexure showcases snapshots from the website that we developed using Flask and basic web technologies to demonstrate the output of the LSTM-based stock prediction model in a user-friendly interface. The website was created as part of a future deployment plan for real-time investor use. It enables users to interact with the trained model by selecting a company, choosing a forecast horizon (e.g., 1 week, 1 month, 1 year), and visualizing predictions against actual stock price movements. Infosys is used as an example company to demonstrate various features of the website.

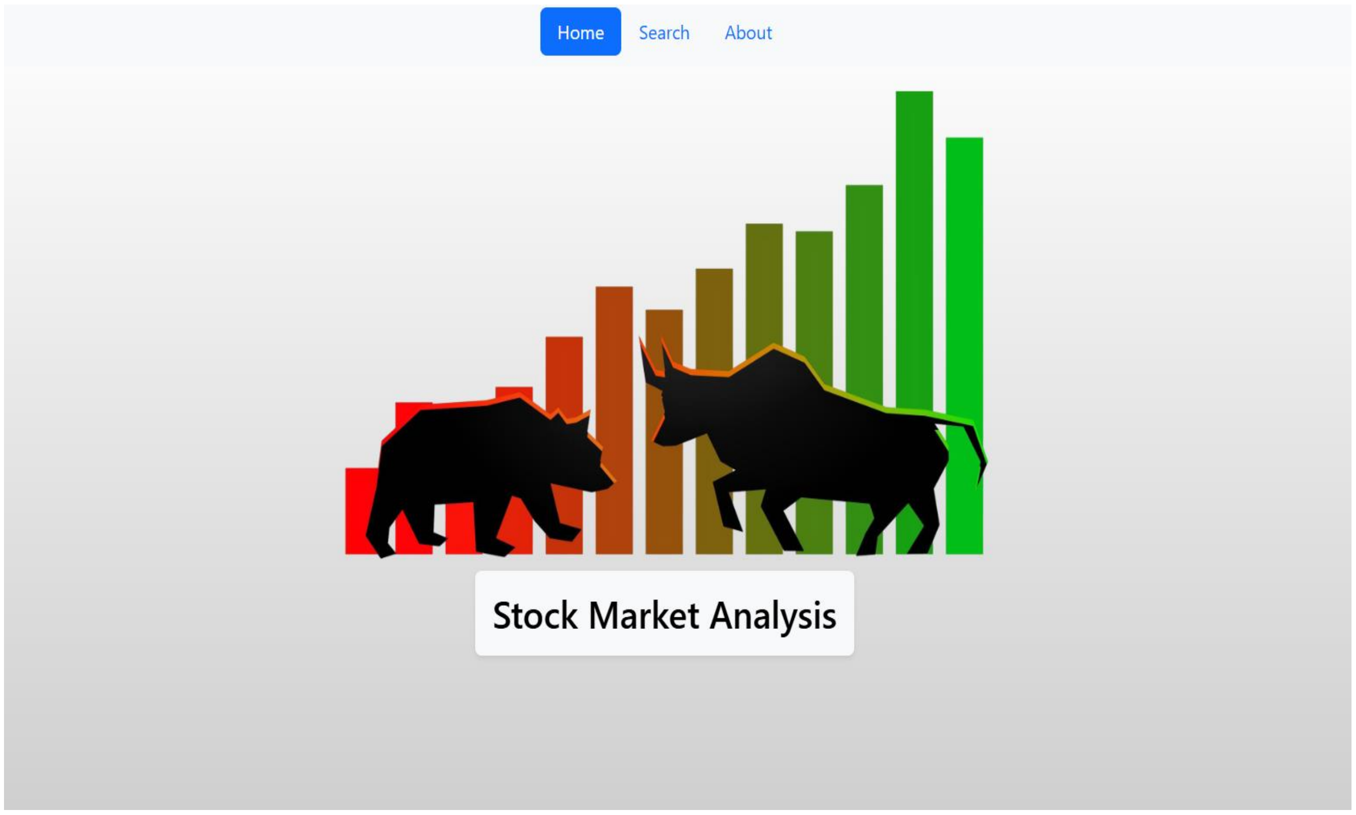
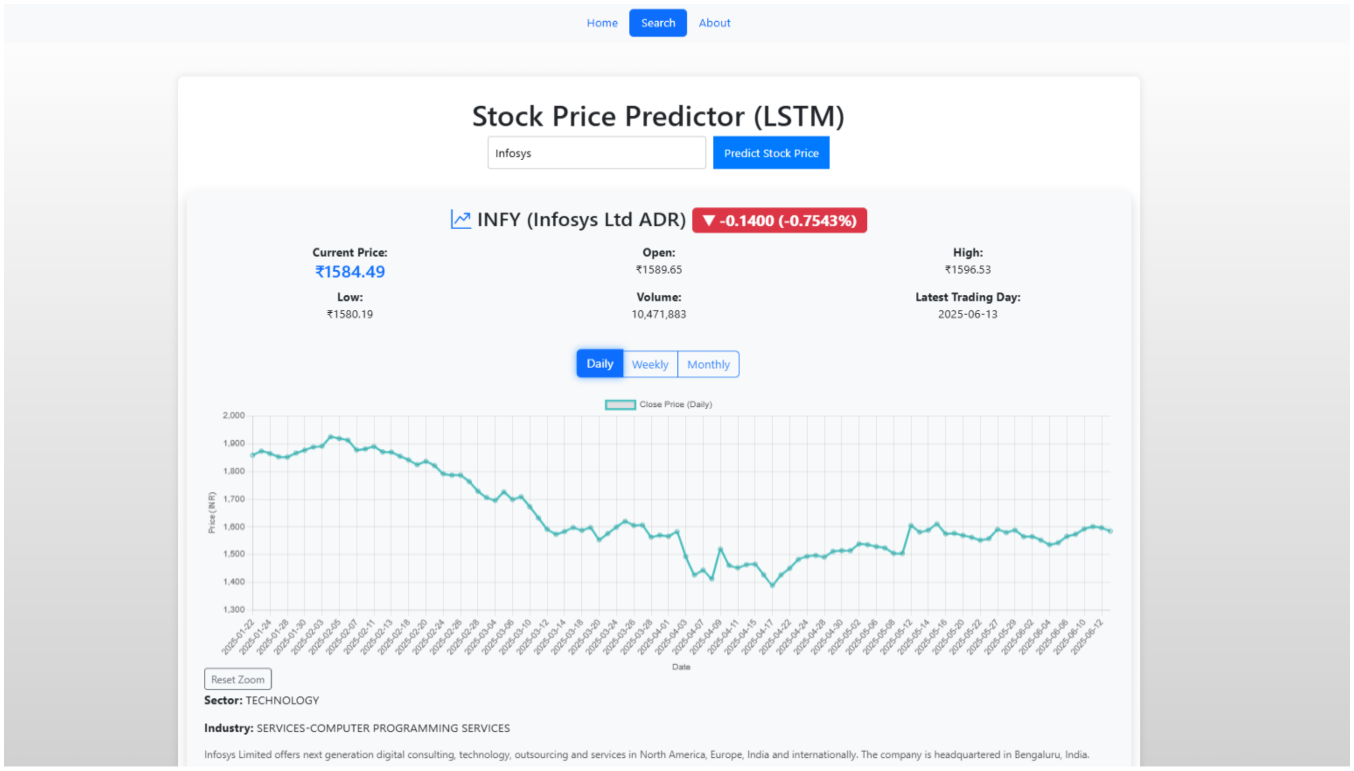


Figure 13: First Page of Website



Stock Price Predictor (LSTM)

INFY (Infosys Ltd ADR) ▼ -0.1400 (-0.7543%)

Current Price: ₹1584.49

Open: ₹1589.65

High: ₹1596.53

Low: ₹1580.19

Volume: 10,471,883

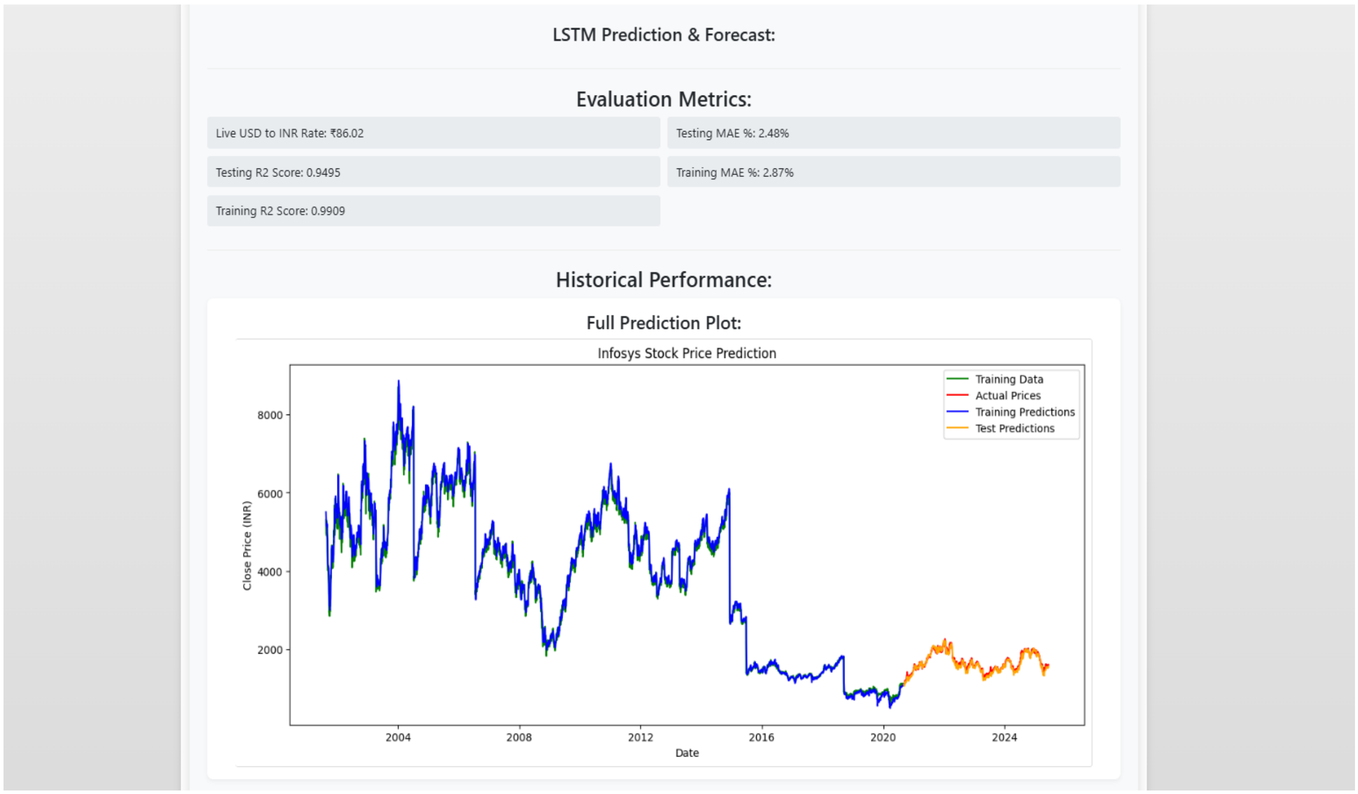
Latest Trading Day: 2025-06-13

Sector: TECHNOLOGY

Industry: SERVICES-COMPUTER PROGRAMMING SERVICES

Infosys Limited offers next generation digital consulting, technology, outsourcing and services in North America, Europe, India and internationally. The company is headquartered in Bengaluru, India.

Figure 14: Search tab and entering company name (e.g., Infosys)



LSTM Prediction & Forecast:

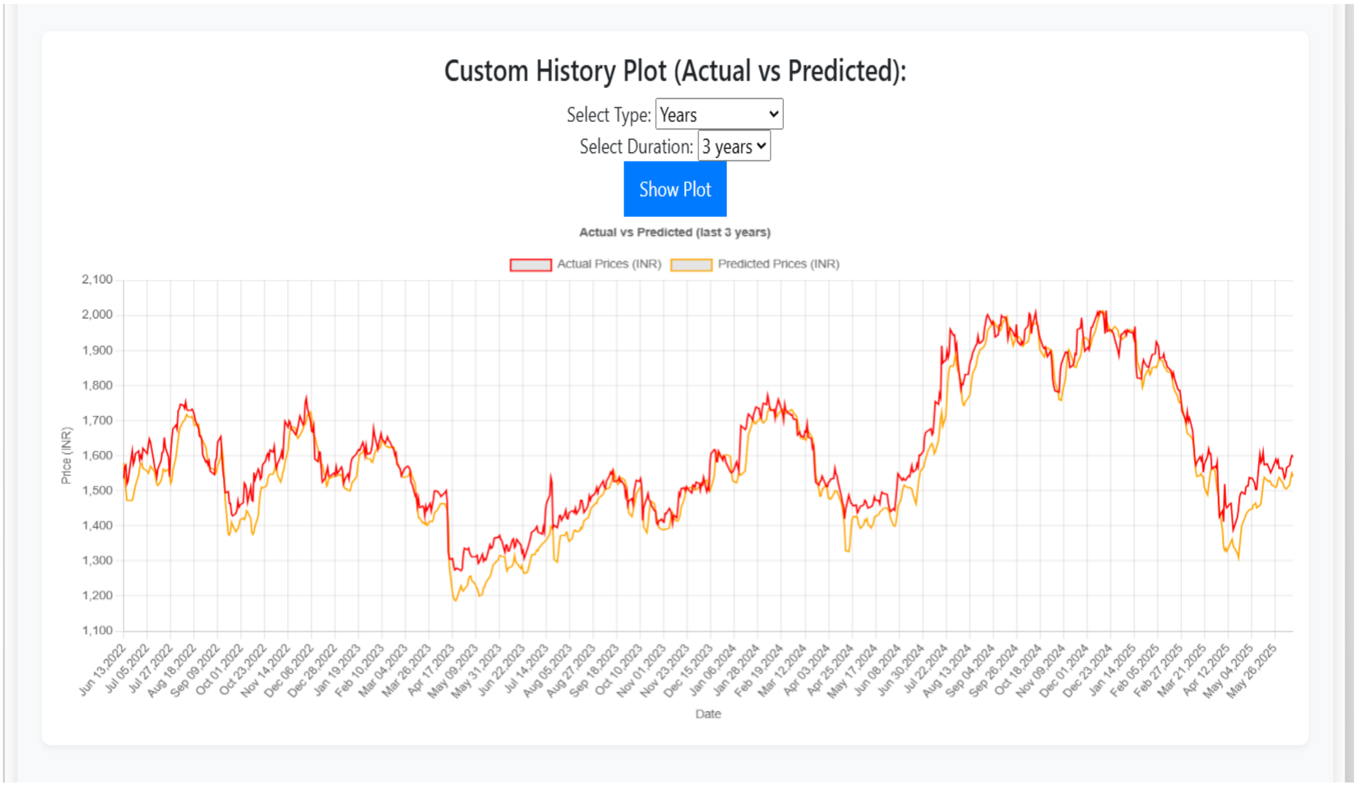
Evaluation Metrics:

Live USD to INR Rate: ₹86.02	Testing MAE %: 2.48%
Testing R2 Score: 0.9495	Training MAE %: 2.87%
Training R2 Score: 0.9909	

Historical Performance:

Full Prediction Plot:

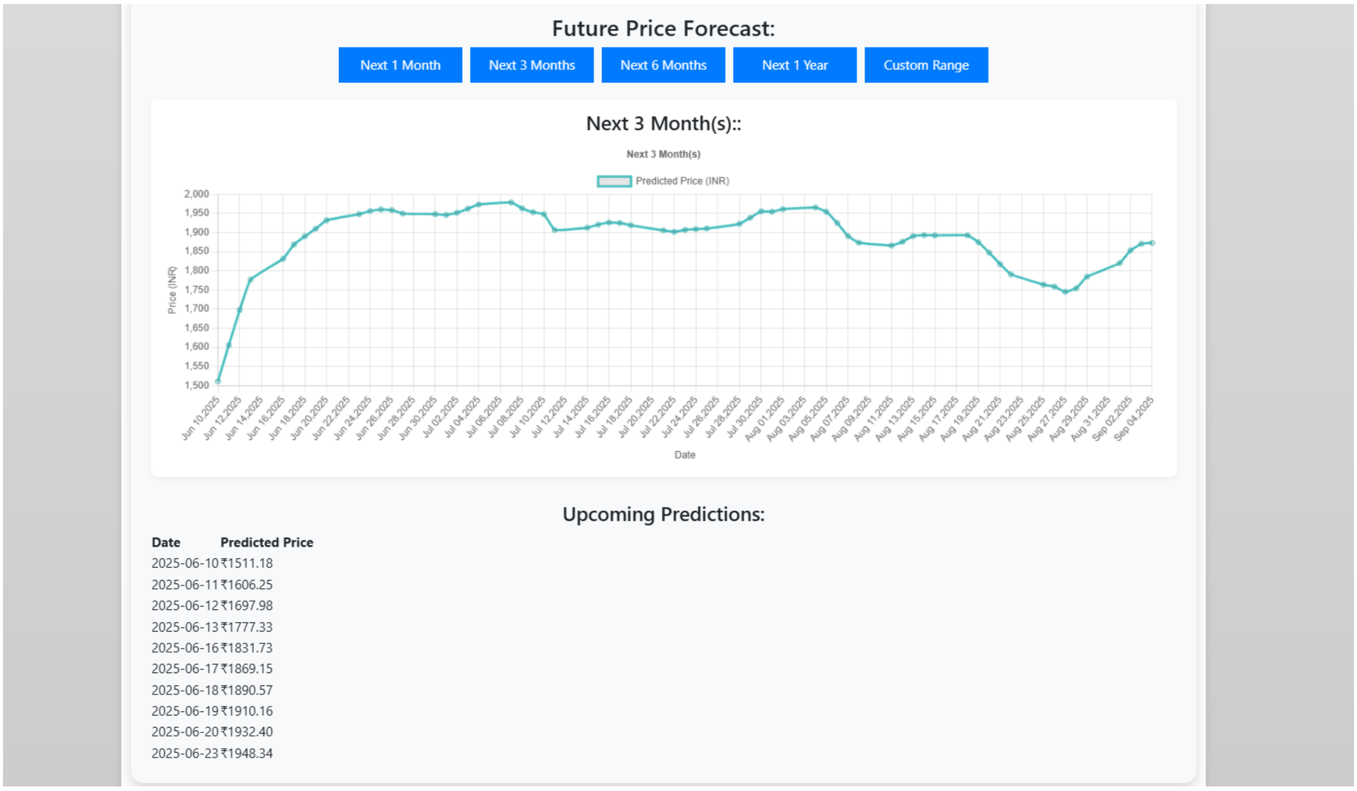
Figure 15: Actual vs Predicted plot of Infosys of Full Data



Select Type: Years

Select Duration: 3 years

Figure 16: Actual vs Predicted Stock Price of last 3 years of Infosys



Upcoming Predictions:

Date	Predicted Price
2025-06-10	₹1511.18
2025-06-11	₹1606.25
2025-06-12	₹1697.98
2025-06-13	₹1777.33
2025-06-16	₹1831.73
2025-06-17	₹1869.15
2025-06-18	₹1890.57
2025-06-19	₹1910.16
2025-06-20	₹1932.40
2025-06-23	₹1948.34

Figure 12: Next 3 Months Prediction Plot of Infosys

Originality report

COURSE NAME
plag t

STUDENT NAME
Subhon Sanyal

FILE NAME
Final Report.docx

REPORT CREATED
14 Jun 2025

Summary

Flagged passages	7	1%
Cited/quoted passages	1	0.2%

Web matches

mdpi.com	2	0.4%
researchgate.net	1	0.2%
geeksforgeeks.org	1	0.2%
kaggle.com	1	0.2%
medium.com	1	0.2%
jatit.org	1	0.1%
numberanalytics.com	1	0.1%

1 of 8 passages

Student passage FLAGGED

Abstract Stock price prediction **is a challenging task due to the** non-linear, **volatile**, and multi-factorial **nature of** financial **markets**

Top web match

E-mail:1 nhsoman2015@gmail.com, 2 hodci@mazcol.edu.om **ABSTRACT** Accurately forecasting financial time-series data **is a challenging task due to the** dynamic and **volatile nature of** stock **markets.**

LSTM MODEL - Jatit.org https://www.jatit.org/volumes/Vol103No7/28Vol103No7.pdf

2 of 8 passages

Student passage FLAGGED

Traditional statistical models like **ARIMA, Moving Average**, and Linear Regression have provided foundational methods for time-series analysis, but **they often fall short** in **capturing complex** temporal…

Top web match

While **traditional statistical models**, such as linear regression and **ARIMA** (autoregressive integrated **moving average**), are often applied to this task, **they often fall short** of **capturing** the **complex and**…

Assessing the Predictive Capabilities of Autoregressive Integrated ... https://www.mdpi.com/2227-9067/11/11/1312

3 of 8 passages
Student passage FLAGGED

LSTM networks are adept at handling **long-term dependencies** in sequential data, **making them** highly **suitable for financial forecasting where** past price movements and indicators significantly **influence**…

Top web match

LSTM networks are adept at capturing **long-term dependencies** and complex temporal patterns, **making them suitable for financial forecasting where** historical context can **influence future** movements (7)

Stock market's price movement prediction with LSTM neural networks https://www.researchgate.net/publication/318329563_Stock_market's_price_movement_prediction_with_LSTM_neural_networks

---

4 of 8 passages
Student passage FLAGGED

Long **Short-Term** Memory (LSTM) **networks are a** special **type of Recurrent Neural Network (RNN) designed to capture long-term dependencies in sequential data**

Top web match

LSTM (Long **Short-Term** Memory) **networks are a type of recurrent neural network (RNN) designed to** address the vanishing gradient problem and better **capture long-term dependencies in sequential data.**

Advantages of stacking LSTMs? - GeeksforGeeks https://www.geeksforgeeks.org/advantages-of-stacking-lstms/

---

5 of 8 passages
Student passage FLAGGED

 Recursive Forecast Loop (For Multi-Step Forecast): The **model** predicts **one step ahead, then** feeds that prediction **back into itself to predict** the next step, and so on. Recursive forecasting may…

Top web match

In **recursive** forecasting, we train a **model** to predict just **one step ahead, then** feed its output **back into itself** repeatedly **to predict** further.

Using Python for multi-step time series forecasting | by Katy - Medium https://medium.com/pythons-gurus/using-python-for-multi-step-time-series-forecasting-7689eff69284

---

6 of 8 passages
Student passage FLAGGED

model.**a**dd(**D**en**s**e(units=50)) **m**odel.**a**dd(Den**s**e(units=1))**m**odel.compile(op**t**imiz**e**r='ada**m', lo**ss='mea**n**_absolute_error')def lr_scheduler(epoch, lr): return max(0.0001, lr * 0.9)early_stopping = …

Top web match

RSI assesses price momentum by comparing average gains to average losses over n periods. **M A** C **D = E M A s** h **o** r **t − E M** A **l o n g** (6)

Hybrid Machine Learning Models for Long-Term Stock Market ... https://www.mdpi.com/1911-8074/18/4/201

---

7 of 8 passages
Student passage CITED

current_sequence = np.roll(current_sequence, **-1, axis=1) current_sequence[0, -1, 0] = next_pred[0, 0**]

Top web match

range(n_future): next_pred = model.predict(current_sequence) future_predictions.append(next_pred[0, 0]) current_sequence = np.roll(**current_sequence, -1, axis=1) current_sequence[0, -1, 0] =**…

Bitcoin Price Prediction Using LSTM - Kaggle https://www.kaggle.com/code/sdeogade/bitcoin-price-prediction-using-lstm/notebook

---

8 of 8 passages

Student passage FLAGGED

…Attention-based models: Focuses **on the most relevant parts of the input data** for each prediction, improving **accuracy** in long **and** noisy sequences**.**

Top web match

By focusing **on the most relevant parts of the input data**, AI models can reduce noise, improve **accuracy, and** decrease computational costs**.**

Mastering Attention in Cognitive Computing - Number Analytics https://www.numberanalytics.com/blog/mastering-attention-cognitive-computing

---