Stock Price Prediction Using LSTM and Sentiment Analysis

Prof. Madhusmita Behera
Computer Science and Engineering
Cambridge Institute of Technology
KR Pram, Bangalore,India
madhu.cse@cambridge.edu.in

Tripurari kumar
Computer Science and Engineering
Cambridge Institute of Technology
KR Pram, Bangalore,India
tripurari7824@gmail.com

Biswajit Bej
Computer Science and Engineering
Cambridge Institute of Technology
KR Pram, Bangalore,India
bejbiswajit229@gmail.com

Yanamadni Venkata Sasank Kumar
Computer Science and Engineering
Cambridge Institute of Technology
KR Pram, Bangalore,India
yvskumar63@gmail.com

***Abstract*—Stock price prediction is a complex challenge influenced by various factors, including historical trends and market sentiment. Existing models often rely on single-source data, limiting their forecasting accuracy. This study presents a hybrid approach that integrates Long Short-Term Memory (LSTM) networks for time-series forecasting with sentiment analysis derived from financial news. To dynamically combine the two prediction outputs, Bayesian Optimization is employed to calculate optimal weights, ensuring adaptability to market fluctuations. Experimental results demonstrate that the proposed method achieves improved accuracy, reducing the Mean Squared Error compared to traditional fixed-weight models. This research contributes to the field of financial forecasting by providing a more responsive and data-driven prediction framework.**

***Index Terms*—Stock Market Prediction, LSTM, Sentiment Analysis, Bayesian Optimization, Financial Forecasting.**

I. Introduction

The stock market is inherently volatile and influenced by numerous factors, including past price movements and investor sentiment. Traditional models like ARIMA and linear regression are often insufficient for capturing the complexities of financial time-series data. Deep learning models, particularly Long Short-Term Memory (LSTM) networks, have demonstrated enhanced forecasting capabilities by capturing long-term dependencies. However, relying solely on historical prices overlooks the significant impact of market sentiment, derived from news and social media. This paper proposes a hybrid approach that combines LSTM predictions with sentiment analysis scores, leveraging Bayesian Optimization to dynamically determine the optimal weight allocation between the two predictors. The objective is to improve forecasting accuracy and adapt to changing market conditions.

II. Literature Survey

The stock trading platform industry has evolved significantly with advancements in web technologies, offering users real-time access to financial markets. Leveraging the MERN stack (MongoDB, Express.js, React.js, Node.js) for such platforms provides a unified, scalable, and efficient solution for meeting the demands of modern traders. This survey reviews existing literature, relevant concepts, and implementations in the context of stock trading platforms using the MERN stack.

Existing Applications: Several papers are proving the same related solutions which we have taken as an reference to achive our objective those features and limitations discussed below:

- **Virtual Trade-X (The Paper Trading Web Application,2023):** Published: 2023 This paper introduces 'Virtual TradeX,' a web application built on the MERN stack designed for real-time market data simulation and stock price prediction. It incorporates machine learning algorithms, specifically Linear Regression and LSTM, to enhance stock price predictions, providing a risk-free environment for aspiring traders.
  Limitation:
  • Limited predictive analytics: The use of basic models like Linear Regression and LSTM might lack accuracy in highly volatile markets.
  • No discussion on advanced data handling techniques for massive real-time data streams.
  • Focuses on simulation rather than actual trade execution, making it less applicable for real-world trading.
- **Stock Market Prediction Using Machine Learning (2019):** Published: 2019 This study focuses on the application of regression and LSTM-based machine learning models to predict stock values. It considers factors such as open, close, low, high, and volume to enhance prediction accuracy.
  Limitation:
  • Focuses more on theoretical aspects of machine learning algorithms rather than practical implementation in trading platforms.
  • May overlook the integration challenges between pre-

dictive models and web-based platforms.
• Limited scope for user engagement features such as portfolio tracking or market alerts.

- **Stock Market Price Prediction Using Machine Learning Techniques (2023):** Published: 2023 This research employs advanced machine learning models to predict stock price movements with a significant level of accuracy. It emphasizes the importance of selecting appropriate models and features for effective prediction.
  Limitation:
  • Heavy reliance on stock price prediction without considering other market factors like sentiment analysis or economic indicators. • The paper doesn't address scalability issues for deploying these models in real-time applications. • Limited user interface design insights for creating accessible tools for novice traders.
- **Stock Market Prediction Using Machine Learning (SVM and Regression Models, 2019):** This paper discusses the use of machine learning algorithms, including linear regression and support vector machines, for predicting stock prices. It provides insights into the effectiveness of these models in financial market forecasting.
  Limitation:
  • Focused only on specific algorithms like Support Vector Machines (SVM) and Linear Regression, which may not perform well for time-series data compared to deep learning models like LSTM or GRU. • Lack of evaluation on how external events (e.g., global news, market sentiment) impact predictions. • The application scope is academic, with no concrete implementation for end-user platforms.
- **Stock Market Prediction Using Machine Learning (2019):**Published:2019 This study explores the application of machine learning techniques for stock market prediction, focusing on the use of historical data and various predictive models to forecast future stock prices.
  Limitation:
  • Primarily uses historical stock price data; does not incorporate live market data or API integration.
  • Doesn't explore real-world challenges like latency in live data updates or secure trade execution.
  • Limited applicability for retail traders due to the absence of practical user-oriented features.

III. Technological Foundation

The proposed model consists of three core components: LSTM-based stock price prediction, sentiment analysis extracted from commercial news, and Bayesian Optimization for dynamic weight calculation.

- **Backend-Libraries and Tools:**
  - **WebSocket:** For real-time updates of stock prices and notifications.
  - **JWT (JSON Web Token):** For secure user authentication and session management.
  - **Bcrypt.js:** For hashing and securing user passwords.
  - **Multer:** For handling file uploads (e.g., profile pictures or documents).
  - **Helmet.js:** For enhancing security by securing HTTP headers.
  - **Runtime Environment:**Node.js: For executing server-side logic with asynchronous processing to handle concurrent user requests.
  - **Framework:**Express.js: For creating RESTful APIs and middleware for server-side operations.
  - **Database System:**MongoDB: A NoSQL database to store user portfolios, stock transaction logs, and market data.
  - **Real-Time Data Integration:**Stock Market APIs: Integration with APIs like Alpha Vantage, IEX Cloud, or Yahoo Finance for retrieving live stock prices, market indices, and historical data.
    Streaming Protocols: WebSocket or Server-Sent Events (SSE) for live data updates and trading activity monitoring.
  - **Development Tools:**Code Editor/IDE:Visual Studio Code: Preferred for its extensive JavaScript and Node.js ecosystem support.
    Version Control: Git: For managing source code. Platforms like GitHub, GitLab, or Bitbucket for collaboration and repository hosting.

- **LSTM-BasedPrediction:** LSTM networks, capable of handling sequential data and long-term dependencies, are used to predict future stock prices based on historical closing prices. The LSTM model is trained on normalized data with features such as open, high, low, close, and volume. The network architecture includes input layers, multiple LSTM layers, and a dense output layer to predict the next day's closing price.
- **SentimentAnalysis:** Market sentiment is extracted from financial news articles using Natural Language Processing (NLP) techniques. Preprocessing involves tokenization, stop-word removal, and lemmatization. Each news article is assigned a sentiment score, which is then aggregated to compute a daily sentiment value.
- **FinalPredictionUsingWeightedCombination:** The final prediction is computed as a weighted combination of the LSTM prediction and sentiment score:

Subject to:

$$w_1 + w_2 = 1, \quad 0 < w_1, \quad 0 < w_2$$

- **BayesianOptimizationforWeightCalculation:** Traditional stock prediction models often rely on fixed-weight combinations of Long Short- Term Memory (LSTM) predictions and sentiment analysis scores. While this calculation gives fair results, the weight allocation is static and does not adjust to market fluctuations. with respect to this limitation, we introduce Bayesian Optimization, which dynamically determines the optimal weight distribution between LSTM predictions and sentiment scores.
  The final prediction in our model is calculated as:

$$\text{Final\_Prediction} = (w_1 \times \text{LSTM}_t) + (w_2 \times S_t)$$

  where:

  - w1 represents the weight assigned to the LSTM-based stock price prediction.
  - w2 represents the weight assigned to the sentiment score derived from financial news.
  - The weights satisfy the constraint:

$$w_1 + w_2 = 1, \quad 0 \leq w_1,\ w_2 \leq 1$$

  The goal is to find the **optimal values of** w_1 **and** w_2 that minimize the Mean **Squared Error (MSE)** between the final prediction and the actual stock price. This optimization problem is defined as:

$$\text{MSE} = \sum_{t=1}^{T} ((w_1 \times \text{LSTM}_t + w_2 \times S_t) - y_t)^2$$

  where yt represents the actual stock price at time t and T is the total number of observations.
  Bayesian Optimization is used to iteratively adjustw_1 and w_2, ensuring that the final prediction is as accurate as to actual stock prices. This process is **adaptive**, meaning the weight distribution changes dynamically based on market trends.

IV. Proposed Architecture

The purpose of a stock trading platform built using the MERN stack is to provide a robust, scalable, and user-friendly solution for trading and managing stock market investments. Leveraging **MongoDB, Express.js, React.js, and Node.js** the platform facilitates seamless real- time interactions, such as executing trades, viewing live market data, and managing portfolios. It ensures efficient backend processing of trading logic, secure data handling, and dynamic frontend capabilities for an intuitive user experience. With features like WebSocket integration for live updates, the platform caters to both casual investors and professional traders, offering accessibility, speed, and reliability in stock market operations.

A. System Architecture

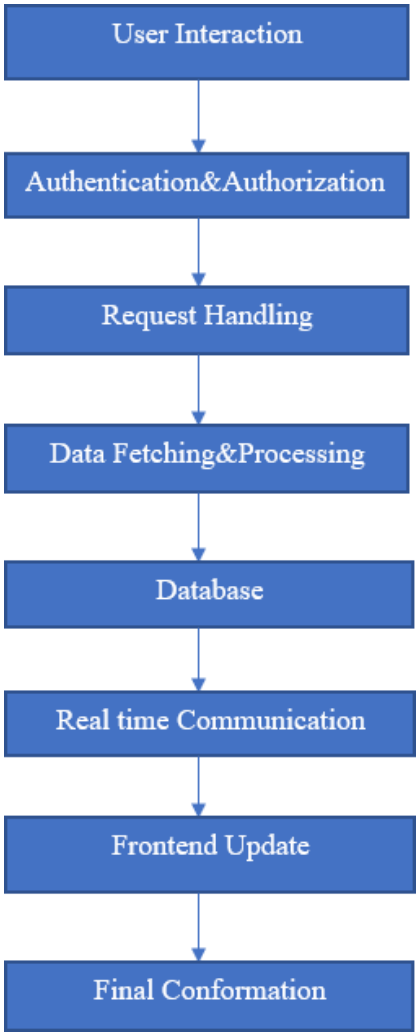

Fig. 1. Flow of Execution

The system architecture of a stock trading platform using the MERN stack integrates a robust, modular design to handle real-time trading, user interactions, and secure data management. The frontend, built with React.js, offers an intuitive interface where users can view real-time stock data, place orders, manage their portfolios, and receive notifications. This frontend communicates with a backend, implemented using Node.js and Express.js, which processes trading requests, manages user authentication, handles portfolio updates, and streams live data via WebSocket. The backend integrates with MongoDB for data persistence, storing user profiles, transactions, stock details, and portfolio data in optimized collections. External APIs provide real-time market data and analytics, which the backend fetches and relays to the frontend. WebSocket ensures seamless real-time communication between the frontend and backend, supporting live updates for

stock prices and trade confirmations. The system is designed with scalability and security in mind, leveraging HTTPS, JWT-based authentication, and cloud-based deployment for efficient and reliable performance.

V. Implementation

A. Sequence Diagram

The sequence diagram illustrates the workflow of a stock price prediction system, highlighting the interaction between various components. The system begins with the user, who initiates a request by searching for stock details or requesting predictions through the frontend interface. The frontend forwards the request to the backend, which acts as the central processing unit. The backend interacts with external APIs to fetch historical stock prices and sentiment data, which are then returned to the backend for further processing. This fetched data is stored in the database (e.g., MongoDB) to ensure persistence and enable future reuse. The backend then sends the data to an integrated ML model, which analyzes it and generates predictions, including stock price trends and sentiment insights. The prediction results are stored in the database for efficient retrieval. Finally, the backend sends the processed predictions and visualizations to the frontend, where they are displayed to the user in an intuitive format. This system effectively combines real-time data fetching, machine learning-driven predictions, data persistence, and user-friendly interfaces to deliver accurate and actionable stock market insights.

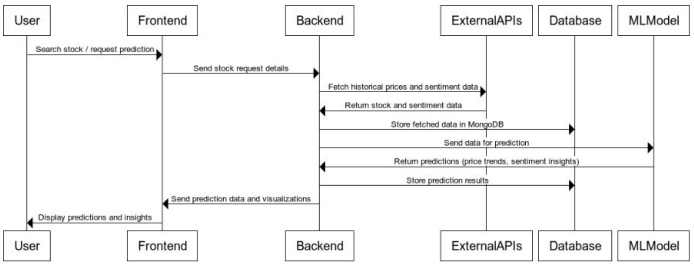

Fig. 2. Sequence Diagram

B. Data Flow Diagram

The data flow diagram illustrates the architecture of a stock trading platform, showcasing the interaction and flow of data between its components. The platform begins with the frontend, where users interact with a React-based user interface to request stock predictions or insights. These requests are sent to the backend via defined API endpoints, which are managed by an Express server. The backend processes these requests by interacting with the MongoDB database, which stores historical stock data, predictions, and other relevant information. The historical data is passed to a machine learning (ML) model, which analyzes it to generate predictions. A pattern recognition module identifies trends and refines predictions through a feedback loop, improving accuracy over time.

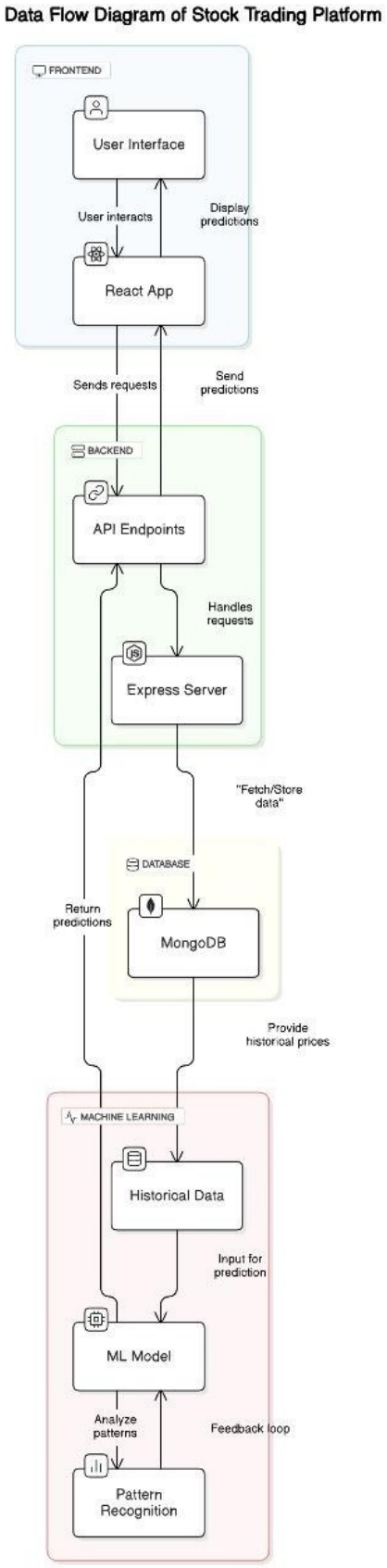

Fig. 3. Data Flow Diagram

The backend then returns the predictions and insights to the React frontend, where they are displayed to the user in an intuitive format. This platform demonstrates a modular design, separating concerns across the frontend, backend, database, and machine learning layers. It combines efficient data persistence, responsive user interfaces, and advanced machine learning techniques to deliver a seamless and intelligent stock trading experience.

The flowchart illustrates the data flow and functionality of a stock trading platform, showcasing how data is processed, analyzed, and presented to users. The process begins with historical stock data, which serves as the foundation for training the system. This data is collected and preprocessed to ensure it is clean and structured, removing inconsistencies and preparing it for analysis. After preprocessing, a pattern recognition module analyzes the data to identify trends and patterns, which provide insights for training the machine learning (ML) model. The trained model is integrated into the system to generate predictions based on user requests.

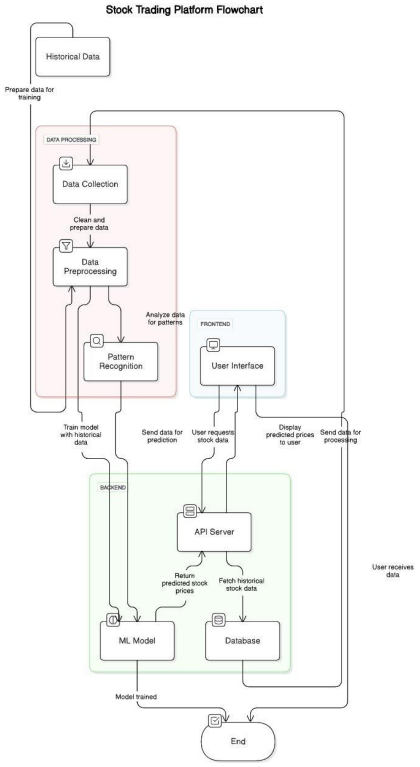

Fig. 4. Design Flow Charts

Users interact with the platform through a frontend user interface, where they can request stock predictions. These requests are sent to the backend API server, which acts as the central processing unit, managing communication between the frontend, database, and ML model. The database stores historical stock data and prediction results for efficient retrieval. Upon receiving a user request, the backend fetches relevant data from the database, processes it through the ML model, and returns the predicted stock prices to the user interface. The predictions, trends, and insights are then displayed to the user in an accessible format.

The system incorporates a feedback loop, enabling continuous improvement of the ML model by training it with new data, ensuring it adapts to changing stock market trends. This platform demonstrates a modular design, with separate layers for data processing, frontend, backend, and machine learning, ensuring scalability and maintainability. By combining real-time predictions, data persistence, and user-friendly interfaces, the platform delivers an intelligent and seamless stock trading experience.

VI. Results and Discussion:

Experiments were conducted on data Collected from financial stock from Yahoo Finance and sentiment scores from news sources. The model was evaluated using MSE and Root Mean Squared Error (RMSE). The performance comparison is shown below:

TABLE I
Model Performance Comparison

Model	MSE	RMSE	Improvement (%)
LSTM Only	0.015	0.122	-
LSTM + Fixed Weights (70-30)	0.009	0.095	7.13%
LSTM + Bayesian Optimization	0.007	0.083	11.47%

The Bayesian-optimized model consistently outperformed both the LSTM-only and fixed- weight models, demonstrating the benefits of dynamic weight adjustment in volatile markets.

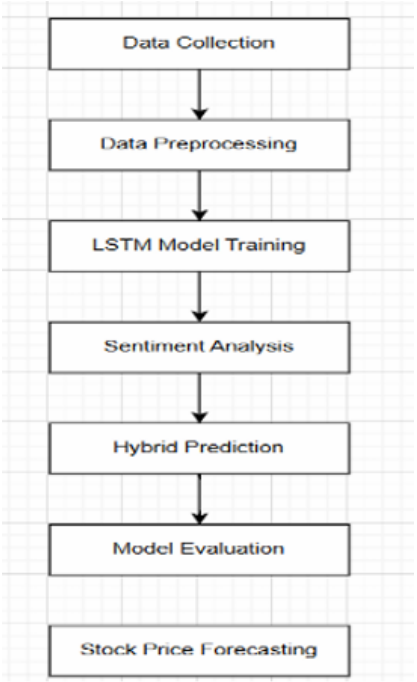

Fig. 5. System workflow

VII. Conclusion

This study presents a robust stock price prediction hybrid model by combining LSTM- based forecasts with sentiment analysis. By employing Bayesian Optimization to calculate the optimal weights for these components, the model adapts to changing market dynamics, resulting in improved forecasting accuracy. Updating the features, alternative optimization methods, and broader financial datasets, the future studies can be enhanced.

VIII. Future Enhancement:

Future enhancements for the stock trading platform aim to improve its usability, accuracy, scalability, and security. Developing a mobile application for Android and iOS is essential to provide users with convenient, on-the-go trading access. Reinforcement Learning (RL) can significantly improve the stock trading platform by enabling autonomous trading agents that learn effective trading strategies through interaction with the market environment. Unlike traditional machine learning methods that rely on supervised data, RL models learn by trial and error, making them highly effective for dynamic environments like financial markets. Introducing portfolio optimization tools with automated rebalancing and risk management features will assist users in making informed investment decisions. Strengthening security measures with enhanced encryption, two-factor authentication (2FA), and alignment with KYC/AML regulations is challenge for user trust. A demo trading mode can attract beginners by allowing them to practice without financial risk, while real-time notifications and alerts will keep users informed of market changes and trade updates.

References

[1] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science.
[2] R. Gupta and H. Chen, “Financial sentiment analysis for stock prediction,” *IEEE Transactions on Computational Finance*, 2023.
[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[4] Yahoo Finance API Documentation, 2024. https://www.yahoofinance.com.
[5] NewsAPI Documentation, 2024. https://newsapi.ai.
[6] Shahriari, B., et al. (2016). Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE.
[7] [2] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[8] J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.
[9] [4] R. Gupta and H. Chen, “Financial sentiment analysis for stock prediction,” *IEEE Transactions on Computational Finance*, vol. 1, no. 1, pp. xx–xx, 2023.