A Project Report On

STOKIS – Stock Trends Outlook using Key Indicators and Sentiment

Submitted By

Ankit Paul (16900121034)
Arpan De (16900121058)
Hindol Banerjee (16900121051)
Ritankar Jana (16900121054)
Mayukh Chakraborty (16900121033)

Department of Computer Science & Engineering
Semester- 8th
Subject: Project-III (PROJ-CS881)

Under the guidance of
Prof. Prasenjit Das

A Project Report
To be submitted in the partial fulfilment of the requirements
For the degree of
Bachelor of Technology in Computer Science and Engineering

Department of Computer Science and Engineering,
Academy of Technology

Affiliated to

Maulana Abul Kalam Azad University of Technology,
West Bengal
2024-25
Academy of Technology

CERTIFICATE

This is to certify that the project entitled **STOKIS – Stock Trends Outlook using Key Indicators and Sentiment** submitted to MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY in the partial fulfilment of the requirement for the award of the B. TECH degree in COMPUTER SCIENCE AND ENGINEERING of **Project-III(PROJ-CS881)** is carried out by

Ankit Paul (16900121034)
Arpan De (16900121058)
Hindol Banerjee (16900121051)
Ritankar Jana (16900121054)
Mayukh Chakraborty (16900121033)

under my guidance. The matter embodied in this project is genuine work done by the students and has not been submitted, whether to this university or to any other university or institute, for the fulfilment of the requirement of any course of study.

Prof. Prasenjit Das (Guide)
Department of Computer Science and Engg.
Academy of Technology**,** Aedconagar,
Hooghly-712121, West Bengal, India

Dated:

Countersigned By

Prof. Prasenjit Das
Department of Computer Science and Engg.
Academy of Technology**,** Aedconagar,
Hooghly-712121, West Bengal, India

Declaration by the student

Ankit Paul (16900121034)
Arpan De (16900121058)
Hindol Banerjee (16900121051)
Ritankar Jana (16900121054)
Mayukh Chakraborty (16900121033)

B. Tech 8th Semester
Dept. of Computer Science & Engineering
Academy of Technology

We hereby state that the project report entitled **STOKIS – Stock Trends Outlook using Key Indicators and Sentiment** has been prepared by us to fulfil the requirements of **Project-III (PROJ-CS881)** during the period January 2025 to June 2025.

Ankit Paul **Arpan De**

Ritankar Jana **Hindol Banerjee**

Mayukh Chakraborty

LIST OF FIGURES

LIST OF TABLES

Abstract

Stock price prediction is a complex task influenced by multiple factors such as historical data, market volatility, and real-time news sentiment. This project introduces STOKIS, a comprehensive stock prediction platform that leverages advanced machine learning and natural language processing (NLP) techniques for improved forecasting accuracy and user engagement.

The platform uses a hybrid CNN–Stacked BiLSTM model trained on historical stock prices enriched with technical indicators like Moving Average, MACD, ADX, RSI, etc. This architecture captures both local temporal features and long-term dependencies, achieving an average prediction accuracy of 93.54% across 60+ companies. High performance was recorded for Indian stocks like Sun Pharma (97.87%), NTPC (97.45%), and TVS Motor (96.89%), and international stocks like Goldman Sachs (95.32%), NVIDIA (92.32%) and Meta (91.61%), especially in trend-following sectors.

To incorporate sentiment-driven market movements, a FinBERT model—a BERT variant fine-tuned on financial texts—was integrated for real-time news sentiment analysis. The model classifies articles as positive, negative, or neutral with a classification accuracy of 97.28%, enabling sentiment-informed predictions.

The platform is built on a microservices architecture using FastAPI, Next.js, Tailwind CSS, and TypeScript, with PostgreSQL and MongoDB for structured and unstructured data storage. It supports automated data collection via cron jobs and features a GPT-powered finance chatbot for natural language interaction.

By combining deep learning, sentiment analysis, and a scalable software architecture, STOKIS delivers a powerful decision-support tool for investors. It outperforms traditional models and offers a modern, interactive experience, laying the foundation for future enhancements like portfolio optimization and real-time sentiment streaming.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

Stock markets are pivotal to economic health, and predicting stock prices is a challenging task due to the myriad factors that influence them. These factors include historical price data, economic indicators, and news events. Traditional prediction methods have been limited in their ability to incorporate and analyse these diverse data types effectively. With the rise of machine learning, particularly models like LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) for time series data and NLP models like FinBERT for sentiment analysis, there is a growing potential to create more comprehensive prediction models.

The motivation behind this project is to develop a stock price prediction platform that leverages modern technologies to provide accurate and actionable insights. By integrating historical stock data with news sentiment analysis, the platform aims to offer a more holistic view of the market. This integration is expected to enhance the predictive power of the models, thereby assisting investors in making informed decisions.

1.2 PURPOSE OF STUDY

The purpose of this study is to develop a comprehensive, scalable, and user-friendly platform that leverages advanced machine learning and natural language processing (NLP) techniques to predict stock prices accurately by integrating historical data and news sentiment analysis. This platform aims to provide valuable insights to investors and traders, ultimately aiding in better decision-making processes in the dynamic and complex stock market environment.

1.3 BRIEF OVERVIEW OF THE PROJECT REPORT

This project report discusses the development of a sophisticated stock price prediction platform designed to improve accuracy and user engagement by integrating historical stock data with real-time news sentiment analysis. The report begins with a comprehensive background that highlights the limitations of traditional stock prediction methods and the advantages of employing advanced machine learning techniques, such as BiLSTM and CNN for time series data and FinBERT for sentiment analysis.

The core objectives include creating a user-friendly and scalable platform using a microservices architecture. Key features include advanced stock prediction models, real-time news analysis, a responsive frontend developed with Next.js, and a finance-focused chatbot powered by GPT. The system leverages Docker for deployment, PostgreSQL for structured data, and MongoDB for unstructured data. The feasibility study underscores the economic viability and operational feasibility of the platform, emphasizing cost-effectiveness and development expertise.

The design incorporates modern engineering paradigms, including an MVC pattern and microservices architecture, to ensure modularity and scalability. Comprehensive diagrams detail the data flow, use cases, and system architecture, illustrating the efficient integration of predictive analytics and sentiment analysis for stock forecasting.

CHAPTER 2

LITERATURE REVIEW

2.1 PREVIOUS RESEARCH IN STOCK MARKET PREDICTION

The early stages (Pre 2000s) of stock price prediction research predominantly relied on statistical models designed to identify patterns and dependencies within historical price data. These models included ARIMA (Box and Jenkins, 1976[1]) and GARCH (Engle in 1982[2] & Bollerslev in 1986[3]) which were grounded in time series analysis and focused on linear relationships between data points. However, they were limited in their ability to capture complex, non-linear market dynamics and neglected external variables like news sentiment, economic reports, and global events.

The 2000s marked a paradigm shift with the adoption of machine learning techniques, which provided more robust tools for handling complex and high-dimensional data. Vapnik (1995)[4] introduced SVM to identify non-linear patterns. McCulloch and Pitts (1943)[5] laid foundations of Neural Networks and Haykin, (1999)[6] showed its versatility to capture complex interactions financial forecasting outperforming linear models.

The advent of deep learning in the 2010s introduced new possibilities for financial forecasting, driven by advancements in computational power, data availability, and algorithmic innovations. Graves (2012)[7] demonstrated LSTM's effectiveness in handwriting recognition, paving the way for its application in stock price prediction, where temporal dependencies are critical.

In recent years, Shen and Shafiq (2020)[8] developed a comprehensive deep learning system for short-term stock market trend prediction, achieving R^2 values of 0.89. Additionally, Shankarlingam and Reddy (2023)[9] demonstrated high R^2

value of 0.92 in predicting small-cap company stock prices using Python, highlighting the growing effectiveness of data science in financial forecasting.

2.2 USE OF NLP IN STOCK MARKET ANALYSIS

Early efforts in NLP focused on extracting market sentiment from textual data. Pang and Lee (2008)[10] provided a comprehensive review of sentiment analysis techniques, emphasizing methods for identifying opinions and emotions in text but the accuracy was low.

Modern research integrates NLP techniques with traditional and machine learning-based time series models to enhance predictive accuracy. Ding et al. (2014)[11] proposed a hybrid model combining sentiment analysis with LSTM networks, effectively leveraging textual information (e.g., news sentiment) alongside historical price data to improve stock price forecasts. Bollen et al. (2011)[12] demonstrated a strong correlation between Twitter sentiment and stock market volatility, highlighting the predictive potential of social media data.

2.3 GAPS IN EXISTING RESEARCH

- **Limited Real-World Testing:** Many models are based on simulated or small datasets, limiting their practical applicability.
- **Data Quality & Bias:** Issues like survivorship and look-ahead bias distort results, while unstructured data (e.g., news, social media) is often noisy and requires heavy preprocessing.
- **Model Interpretability:** Deep learning models lack transparency, making their predictions hard to explain—an issue in regulated industries.
- **Poor Multimodal Integration:** Most studies focus on single data types, missing the potential of combining numerical, textual, and economic data for better accuracy.

CHAPTER 3

PROBLEM DEFINITION & OBJECTIVES

3.1 PROBLEM STATEMENT

Stock market prediction is a complex task influenced by a wide range of factors, including historical prices, economic indicators, and news sentiment. Current prediction tools often rely on limited data sources and traditional methods, resulting in suboptimal accuracy and user engagement. There is a need for a platform that integrates historical data analysis with real-time news sentiment, utilizing advanced machine learning models, to provide more accurate predictions and an interactive experience for investors. Accurate stock price predictions can significantly enhance investment strategies, leading to better financial outcomes.

3.2 OBJECTIVES OF THE PROJECT

Key objectives of the project are as follows:

1. **Accurate Prediction:** To create a system that combines historical stock data with real-time news sentiment analysis, utilizing machine learning models such as stacked BiLSTM and CNN for time series prediction and FinBERT for sentiment analysis, to enhance prediction accuracy.
2. **Scalability and Maintainability:** To employ a microservices architecture with Docker containerization, ensuring scalability, portability, and ease of maintenance for handling large volumes of data and user interactions.
3. **Diverse Data Handling:** To use PostgreSQL for structured data and MongoDB for unstructured data, catering to the diverse data storage needs of the platform.
4. **User-Friendly Interface:** To develop a frontend with Next.js, Tailwind CSS, and TypeScript, ensuring a responsive and intuitive user experience.

5. **Interactive User Experience:** To integrate a GPT-based finance chatbot, providing real-time advice and enhancing user engagement.
6. **Automation and Updates:** To implement cron jobs for automated data collection, ensuring the platform remains current with the latest information for accurate predictions.

3.3 SCOPE OF WORK

In-Scope:

- **Backend Development:** Microservices with FastAPI, APIs for data, prediction, sentiment analysis, and chatbot. Docker-based containerization and cron jobs for automated data fetching.
- **Database Setup:** PostgreSQL for stock data, MongoDB for news sentiment analysis. Database schema design and implementation.
- **Prediction Model:** Stacked BiLSTM and CNN models using TensorFlow, PyTorch and Scikit-Learn integration.
- **News Analysis:** News crawler bot and sentiment analysis using FinBERT.
- **Frontend Development:** User interface with Next.js, Tailwind CSS, and TypeScript. Displays stock predictions and sentiment results.
- **GPT-Based Chatbot:** Finance-based chatbot integration for user queries.
- **Testing & Deployment:** Unit, integration, and performance tests. Deployment with Docker on a cloud platform like AWS, GCP or Azure.

Out-of-Scope:

- Custom machine learning models beyond specified.
- New database systems or mobile app development.
- Extensive user training.

Assumptions:

- Access to historical stock data and news.
- Computational resources for training and operations.
- Reliable GPT-based chatbot API integration.

Deliverables: Fully functional platform with documentation, test reports, and deployment scripts.

3.4 GENERAL STOCK PARAMETERS AND THEIR SIGNIFICANCE

1-Year Target Estimate:

The average stock price forecasted by analysts for the next year. Therefore,

$$1\ year\ target\ est = Forecasted\ Price$$

It helps investors predict the potential growth or decline in stock value.

52-Week High:

The highest price at which a stock has traded in the past 52 weeks. It is given by

$$52\ weeks\ high = \max\ (Closing\ price\ over\ last\ 52\ weeks)$$

It indicates the peak performance of a stock over a year.

52-Week Low:

The lowest price at which a stock has traded in the past 52 weeks and is given by

$$52\ weeks\ low = \min\ (Closing\ price\ over\ last\ 52\ weeks)$$

It reflects the worst performance of a stock over a year.

Ask Price:

The lowest price at which a seller is willing to sell a stock.

$$ask_{price} = Minimum\ Selling\ Price$$

It provides insight into the supply side of stock trading.

Beta (5- Monthly) ($\boldsymbol{\beta}$):

A measure of a stock's volatility compared to the market over 5 months.

$$\beta = \frac{Cov(R_s, R_m)}{Var(R_m)}$$

It helps investors understand the risk level of a stock.

Bid Price:

The highest price a buyer is willing to pay for a stock.

$$bid = Maximum\ Buy\ Price$$

It helps investors understand the risk level of a stock.

Earnings Per Share (EPS):

A company's profit allocated to each outstanding share.

$$EPS = \frac{Net\ Income - Dividends}{Outstanding\ share}$$

It helps to Assesses a company's profitability.

Earnings Date:

The scheduled date for a company to report its earnings. It indicates key financial updates for investors.

Ex-Dividend Date:

The date on which a stock begins trading without the value of its next dividend payment. It determines dividend eligibility for investors.

Dividend Yield:

A stock's annual dividend payments as a percentage of its price.

$$yield = \frac{Annual\ Dividends\ Per\ Share}{Price\ Per\ Share} \times 100$$

It measures the income generation potential of a stock.

Market Capitalization:

The total value of a company's outstanding shares.

$$market_{capitilization} = Share\ Price \times Outstanding\ Shares$$

It reflects a company's size and market value.

Price-to-Earnings Ratio (PE_{ratio}):

The ratio of a company's share price to its earnings per share.

$$PE_{ratio} = \frac{Share\ Price}{Earning\ Per\ Share}$$

It assesses whether a stock is overvalued or undervalued.

Quote Price:

The current price of a stock in the market.

$$Price_{quote} = Current\ Marked\ Price$$

It reflects real-time value for buying or selling decisions.

3.5 TECHNICAL INDICATORS OF STOCK AND ITS SIGNIFICANCE

Volume Weighted Average Price (VWAP)

VWAP is the average price of a security weighted by its trading volume over a specific time. Institutional traders use VWAP to assess execution quality which helps them to identify fair value and potential support/resistance levels. It acts as a common benchmark for algorithmic trading.

It is calculated as follows:

$$VWAP = \frac{\sum_{i=1}^{n} P_i V_i}{\sum_{i=1}^{n} V_i}$$

where,

P_i = Price of trade *i*

V_i = Volume of trade *i*

n =Number of trades

Moving Average (MA)

A calculation used to analyze data points by creating a series of averages of different subsets of the full data set. It smooths price data to identify trends. Common periods used are 13, 30, 50, 100, and 200 days It is used to generate trading signals and support/resistance levels

Simple Moving Average (SMA) is given by

$$SMA = \frac{\sum_{i=1}^{n} P_i}{n}$$

where,

n = Period Duration

P_i = Price of trade i

Exponential Moving Average (EMA) is given by

$$EMA_t = \alpha \times P_t + (1 - \alpha) \times EMA_{t-1}$$

where,

$\alpha = \frac{2}{n+1}$

P_t = Price of trade t

n = Period Duration

Average Directional Index (ADX)

ADX measures the strength of a trend, regardless of its direction. Values above 25 indicate a strong trend whereas values below 20 indicate a weak trend. It is used to determine if a market is trending or ranging.

It is calculated as follows:

$$ADX = \frac{\sum_{k=1}^{n} \frac{|+DI_k \; - \; -DI_k|}{|+DI_k \; + \; -DI_k|} \times 100}{n}$$

where,

$+DI$ is Positive Directional Indicator

$-DI$ is Negative Directional Indicator

Bollinger Bands (BBands)

A volatility indicator consisting of three lines: a middle band (SMA) and upper/lower bands at standard deviation levels. It measures volatility and relative price levels and helps to identify potential overbought/oversold conditions. It is used for trend identification and breakout detection.

It is calculated as follows:

$$Middle\ Band = \frac{1}{n} \sum_{i=t-n+1}^{t} P_i$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=t-n+1}^{t}(P_i - Middle\ Band)^2}$$

$$Upper\ Band = Middle\ Band + k \cdot \sigma$$

$$Lower\ Band = Middle\ Band - k \cdot \sigma$$

where,

n is Period (typically 20)

k is Number of Standard deviations (typically 2)

P_i is Price at time i

σ Is Standard Deviation of the price over period n

Moving Average Convergence Divergence (MACD)

A trend-following momentum indicator showing the relationship between two moving averages. It identifies trend direction and momentum and signals potential entry/exit points. It also shows bullish/bearish divergences.

It is calculated as follows:

$$MACD\ Line = EMA(12) - EMA(26)$$

$$Signal\ Line = EMA(9)\ of\ MACD\ Line$$

$$MACD\ Histogram = MACD\ Line - Signal\ Line$$

Relative Strength Index (RSI)

A momentum oscillator measuring the speed and magnitude of price changes. It indicates overbought/oversold conditions, shows potential price reversals and identifies bullish/bearish divergences.

It is calculated as follows:

$$RSI = 100 - \frac{100}{1 + RS}$$

where,

$$RS = \frac{Average\ Gain}{Average\ Loss}$$

3.5.7 Stochastic Oscillator (Stoch)

It is also a momentum indicator comparing a closing price to its price range over a specific period. It identifies overbought/oversold conditions, shows potential trend reversals and is used for momentum confirmation.

It is calculated as follows:

$$Lowest\ Low = \min(Low_i, Low_{i-1}, \dots\dots\dots\dots\dots\dots, Low_{i-k+1})$$

$$Highest\ High = \max(High_i, High_{i-1}, \dots\dots\dots\dots\dots\dots, High_{i-k+1})$$

$$\%K = \frac{Close - Lowest\ Low}{Highest\ High - Lowest\ Low} \times 100$$

$$\%D = SMA_d(\%K)$$

where,

Lowest Low is the minimum price over the last k periods

Highest High is the maximum price over the last k periods

Low_i is the Lowest Price at time i

$High_i$ is the Highest Price at time i

Close is the current closing price

k is the number of periods for calculating the Lowest Low (typically 14)

$\%K$ is Stochastic value

d is Number of periods for the moving average (typically 3)

The $\%K$ Line represents the raw stochastic value, showing the current closing price's position relative to the high-low range over the k-period. The $\%D$ Line is a smoothed version of the $\%K$ Line and is often used as a signal line in trading.

CHAPTER 4

FEASIBILITY STUDY

4.1 TECHNICAL FEASIBILITY

The platform's technical feasibility revolves around the implementation of innovative technologies to support its architecture and operational requirements. It uses well-established technologies such as:

- **Backend:** FastAPI (Python) for building microservices.
- **Frontend:** Next.js with Tailwind CSS and TypeScript for a responsive and modern UI.
- **Databases:** PostgreSQL for structured stock data and MongoDB for unstructured news data.
- **Machine Learning:** TensorFlow, PyTorch, and Scikit-learn for building the CNN – Stacked BiLSTM model.
- **NLP:** FinBERT (Hugging Face and Transformers) for sentiment analysis.
- **Containerization:** Docker for deploying microservices in a scalable manner.
- **Chatbot:** GPT-based AI bot for finance-related queries.

All the tools and libraries used in the project are open-source and widely supported by the developer community. Docker containers can be deployed on cloud platforms like AWS, Azure, or Google Cloud, ensuring scalability.

4.2 ECONOMIC FEASIBILITY

The project mostly uses open-source tools and libraries which reduces software licensing costs. It only requires AlphaVantage API to calculate the values of technical indicators which cost $99/month (Rs. 8500/month) when daily limit exceeds. Cloud hosting (e.g., AWS, Azure) will incur costs ($1-2/hour, i.e., Rs. 87-

175/hour) but these can be optimized using free-tier services during development. Training machine learning models (e.g., BiLSTM and CNN) will require GPUs, which will also be expensive ($1-2/hour, i.e., Rs. 87-175/hour). Data storage costs for PostgreSQL and MongoDB Atlas are manageable, especially with cloud providers which offer scalable storage solutions.

To get the return on investment, the platform can be monetized by offering premium features (e.g., advanced predictions, personalized insights) to users. It can also attract partnerships with financial institutions or data providers. The benefits of providing accurate stock price predictions and sentiment analysis will outweigh the development and operational costs.

4.3 OPERATIONAL FEASIBILITY

The team has good expertise and experience in Python, FastAPI, machine learning, NLP, front-end development, Docker, and cloud platforms to ensure smooth development, deployment, and maintenance. The microservices architecture will allow for easy updates and maintenance of individual components. Automated CI/CD pipelines (e.g., GitHub Actions, Jenkins) will ensure efficient deployment and testing. The platform ensures data security by encrypting sensitive data and using user authentication and authorization (like OAuth2) to protect user data. The project targets investors, traders, and financial analysts who need stock predictions and sentiment analysis. The user-friendly front end (Next.js, Tailwind CSS) ensures a positive user experience. The GPT-based chatbot provides additional value by answering finance-related queries. This ensures that the project is operationally feasible.

CHAPTER 5

SYSTEM ANALYSIS

5.1 EXISTING SYSTEM ANALYSIS

The current systems available in the market for stock prediction and analysis generally include standalone tools or platforms with limited integration capabilities. These systems typically provide:

- Basic stock price trend visualization based on historical data.
- Predictive models focused on singular indicators such as moving averages or RSI (Relative Strength Index).
- General-purpose AI chatbots that do not cater specifically to stock market-related queries.

While these platforms provide some level of assistance, they often lack seamless integration between predictive analytics, real-time news insights, and interactive guidance.

5.2 LIMITATIONS OF EXISTING SYSTEM

- **Predictive Accuracy:** Many existing solutions rely on traditional methods or singular indicators, which may not capture the complexities of modern financial markets.
- **Real-Time Integration:** Limited ability to integrate real-time news sentiment analysis with stock prediction models.
- **User Experience:** Lack of an intuitive AI-driven chatbot specifically designed to assist users with stock-related queries and guide them through platform features.
- **Data Depth:** Existing systems do not utilize advanced data pipelines to provide a comprehensive analysis, often leading to shallow predictions.

5.3 PROPOSED SYSTEM FEATURES

The proposed system addresses these limitations by featuring following:

- **Advanced Stock Prediction:** It utilizes multiple indicators such as moving averages, Bollinger bands, RSI, MACD, and machine learning models to predict buy and sell signals. It also incorporates deep learning techniques for trend prediction and anomaly detection.
- **News Analysis with Sentiment Insights:** The platform aims to use news crawler to fetch real time financial news and sentiment analysis powered by natural language processing (NLP) to gauge market sentiment and predict stock movements based on news trends.
- **AI Chatbot Assistance:** It will provide interactive chatbot to guide users through the application, contextual answers to stock-related questions and application navigation. It will be integrated seamlessly with prediction and news analysis features.
- **Data Visualization and Reports:** Intuitive dashboards will be designed to visualize prediction results and sentiment analysis.

5.4 IMPROVEMENT OVER THE EXISTING SYSTEM

- **Enhanced Predictive Accuracy:** Our system leverages ensemble learning models that combine multiple predictors like sentiment analysis with prediction pipelines to improve relevance and accuracy.
- **Real-Time Insights:** It combines live stock market data with real-time news sentiment analysis for timely recommendations.
- **User-Centric AI Guidance:** Unlike generic AI chatbots, our solution focuses on user-specific financial queries and application guidance.
- **Comprehensive Analysis:** It offers a unified platform that combines stock predictions, sentiment analysis, and interactive assistance in one place.
- **Scalability and Flexibility:** It is designed using a modular architecture using Next.js for the frontend, FastAPI and Docker for backend processing, and scalable ML models for prediction.
- **Improved User Engagement:** Intuitive UI/UX with personalized recommendations. AI-driven chat interface for continuous user engagement.

CHAPTER 6

SOFTWARE & HARDWARE REQUIREMENT SPECIFICATIONS

6.1 SOFTWARE REQUIREMENT SPECIFICATION

6.1.1 System Software

To ensure smooth development and deployment, we used the following system software:

Component	Specification
Operating System	Ubuntu 20.04 LTS / Windows 10/ Windows 11
Web Server	Uvicorn (ASGI server for FastAPI)
Containerization	Docker and Docker Compose
CI/CD Integration	GitHub Actions
Version Control	Git (hosted on GitHub)

Table 6.1: System Software Specification

6.1.2 Application Software

This includes the frameworks, databases, APIs, and machine learning libraries used in the project:

Component	Specification
Backend Framework	FastAPI (Python-based for building microservices)
Frontend Framework	Next.js with TypeScript and Tailwind CSS for responsive UI
Databases	PostgreSQL (stock price historical data), MongoDB Atlas (news data)
ML Libraries	TensorFlow, PyTorch, Scikit-learn
NLP and Sentiment Model	FinBERT (from Hugging Face Transformers)
Authentication System	OAuth2 with JWT

APIs Used	Yahoo Finance (for stock data)
Chatbot Integration	GPT-3.5 based chatbot (via OpenAI API)

Table 6.2: Application Software Specifications

6.1.3 Development Tools

Tool	**Usage**
VS Code / PyCharm / WebStorm	Code development
Postman	API testing
Swagger UI	Auto-generated API docs
GitHub	Source code hosting and version control
pgAdmin / MongoDB Compass	Database visualization and management

Table 6.3: Development Tools with their Usage

6.2 HARDWARE REQUIREMENT SPECIFICATION

6.2.1 Minimum Development Environment

This is the hardware setup we used during the development phase of our project:

Component	**Specification**
Processor	Intel i5 8th Gen or Ryzen5 4000 Series or equivalent
RAM	8 GB
Storage	512 GB SSD
Graphics Card	Integrated (sufficient for development)
Internet	Stable broadband (10 Mbps minimum)

Table 6.4: Development Environment Specification

6.2.2 Recommended for Model Training (Locally or on Cloud)

For training the CNN – Stacked BiLSTM and sentiment analysis models, higher compute power was required:

Component	**Specification**
Processor	Intel i7 / AMD Ryzen 7
RAM	16 GB or more
GPU	NVIDIA GTX 1650 or higher / Cloud GPU (AWS T4)
Storage	512 GB SSD
Cloud Provider	AWS EC2 / Google Cloud / Azure (for training)

Table 6.5: Machine Learning Model Training Environment Specification

CHAPTER 7

SYSTEM DESIGN

7.1 SYSTEM ARCHITECTURE

The platform is designed as a microservice-based architecture, leveraging various technologies to ensure scalability, reliability, and maintainability. The system is divided into multiple layers, each handling specific functionalities. Its architecture is shown in *Fig 7.1*.

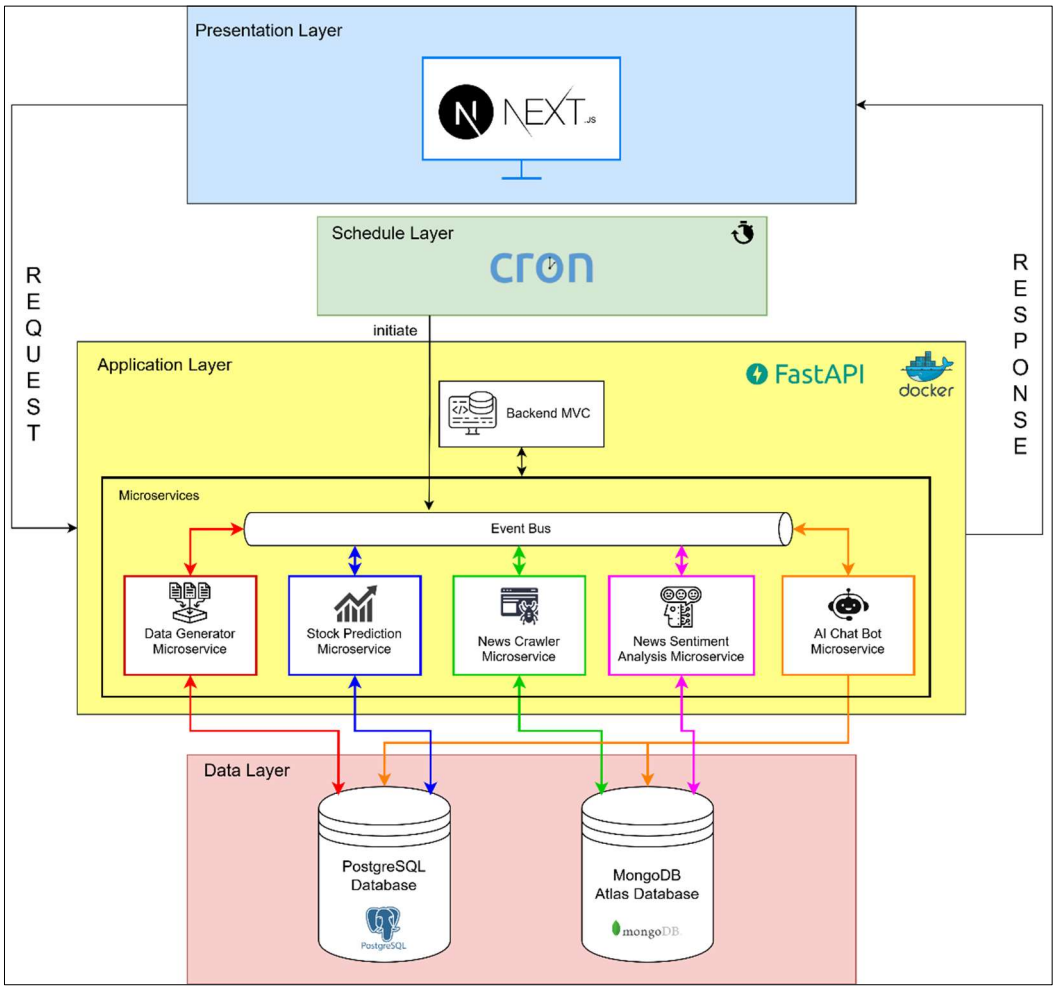

Fig 7.1: System Architecture of the platform

7.2 SOFTWARE ENGINEERING PARADIGMS APPLIED

7.2.1 MVC (Model-View-Controller) Pattern

The Model-View-Controller (MVC) is a software architectural pattern which divides the application into three interconnected components: Model, View, and Controller. This separation of concerns makes the application easier to manage, scale, and maintain.

- **Model:** The Model represents the application's data and business logic. It includes the CNN – Stacked BiLSTM model, FinBERT model and database interactions. It interacts with the database to fetch, store, and update data, contains the core functionality of the application (e.g., algorithms, calculations), and notifies the View when data changes
- **View:** The View is responsible for displaying the data to the user. It represents the user interface (built with Next.js, Tailwind CSS, and TypeScript) of the application. It receives data from the Model and displays stock price predictions, news sentiment analysis, and chatbot interactions to the user. It does not contain any business logic.
- **Controller:** The Controller acts as an intermediary between the Model and the View. It handles user input, processes requests, and updates the Model or View accordingly. It includes FastAPI backend that handles API requests and response.

7.2.2 Microservices Architecture

The microservice architecture for the stock price prediction platform is designed to be modular, scalable, and resilient. Each microservice is responsible for a specific task, allowing for independent development, deployment, and scaling. The architecture leverages FastAPI for building RESTful APIs and Docker for containerization. *Fig 7.2* shows the microservice architecture in detail.

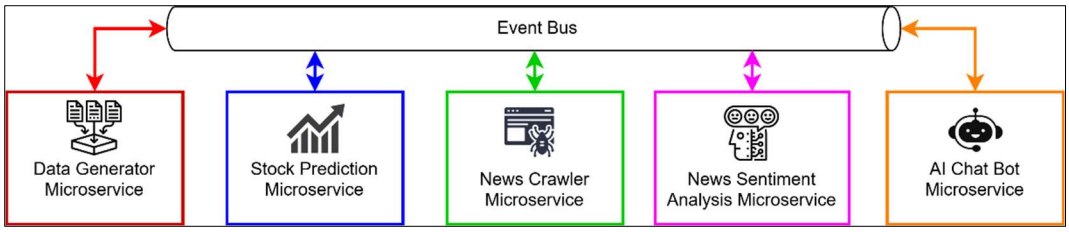

Fig 7.2 Microservice Architecture with Event Bus

The roles of the microservices are explained below:

- **Data Collection Microservice:** Fetches stock data from Yahoo Finance and stores it in PostgreSQL. It uses cron job to trigger data fetching.
- **Stock Prediction Microservice:** Uses CNN – Stacked BiLSTM model to predict future stock prices based on historical data from PostgreSQL.
- **News Crawler Microservice:** Fetches news articles from different news sources and stores in MongoDB Atlas and uses cron job to trigger news fetching.
- **News Sentiment Analysis Microservice:** Analyses news sentiment using FinBERT by retrieving news data from MongoDB Atlas.
- **AI Chat Bot Microservice:** Provides GPT-based real-time finance chat interactions and user support.

7.3 DATA FLOW DIAGRAM (DFD)

Fig 7.3 shows the DFD of the entire platform.

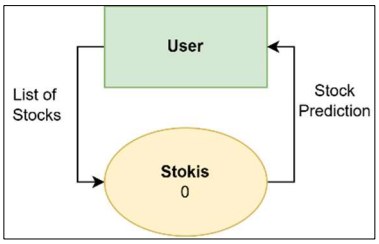

Fig 7.3(i): Level 0 DFD

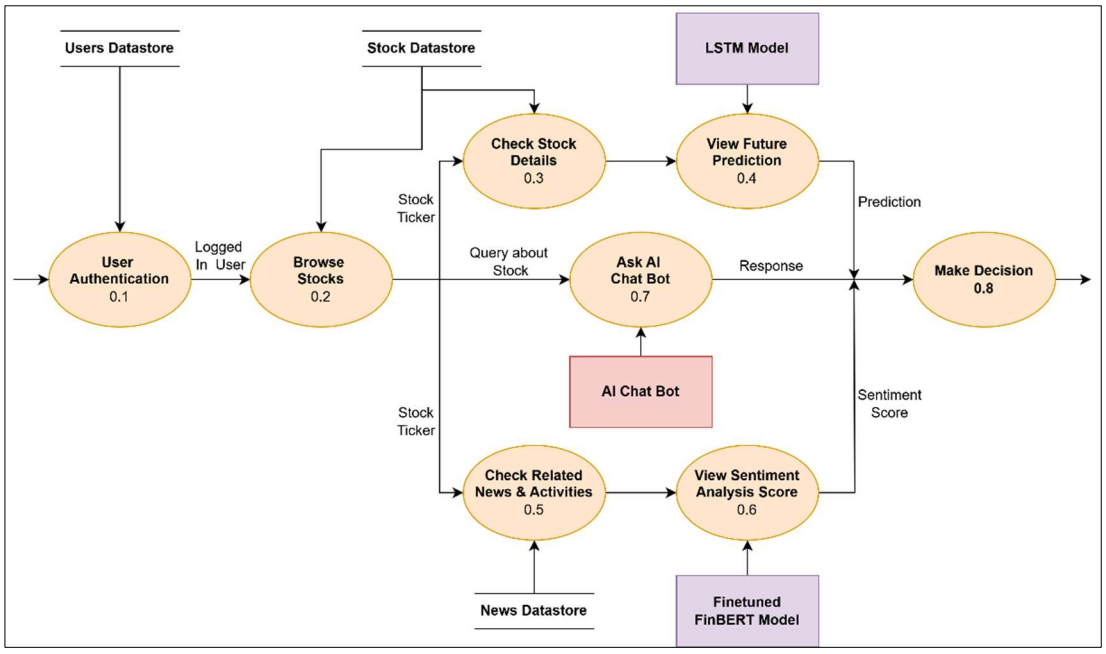

Fig 7.3(ii) Level 1 DFD

It outlines the flow of data through the stock price prediction platform, highlighting the interactions between external entities, microservices, and data stores. The system is designed to efficiently collect, process, and present data to users, integrating predictive analytics and sentiment analysis for comprehensive forecasting.

7.4 USE-CASE DIAGRAM

The use-case diagram of the project is as follows. It provides a visual representation of the interactions between users (actors) and the system and helps to identify the functional requirements of the system and the roles of different users. *Fig 7.4* shows the use-case diagram of the platform.

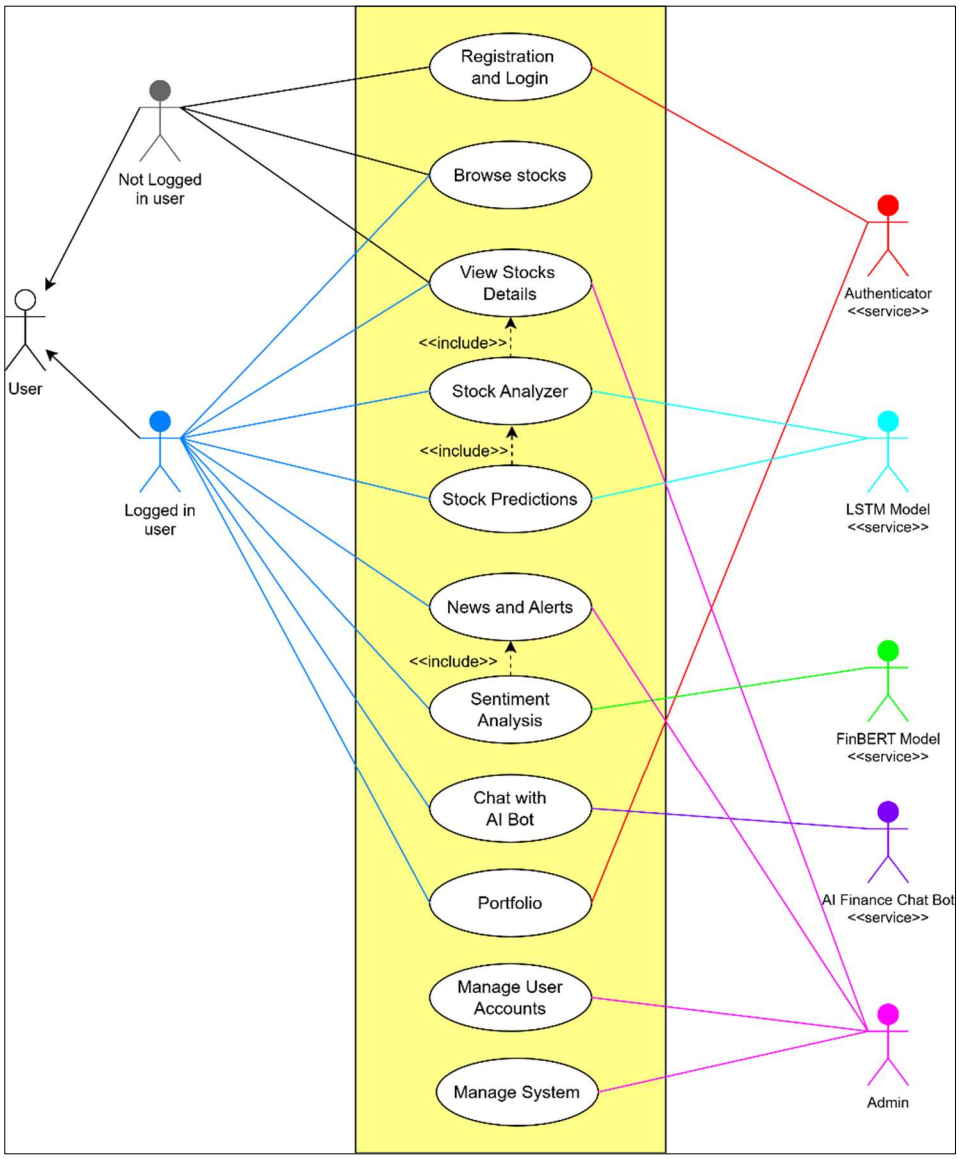

Fig. 7.4: Use-case Diagram

CHAPTER 8

DATABASE DESIGN

8.1 POSTGRESQL DATABASE DESIGN

Users' data and stock data will be stored in PostgreSQL. It is shown in *Fig 8.1*.

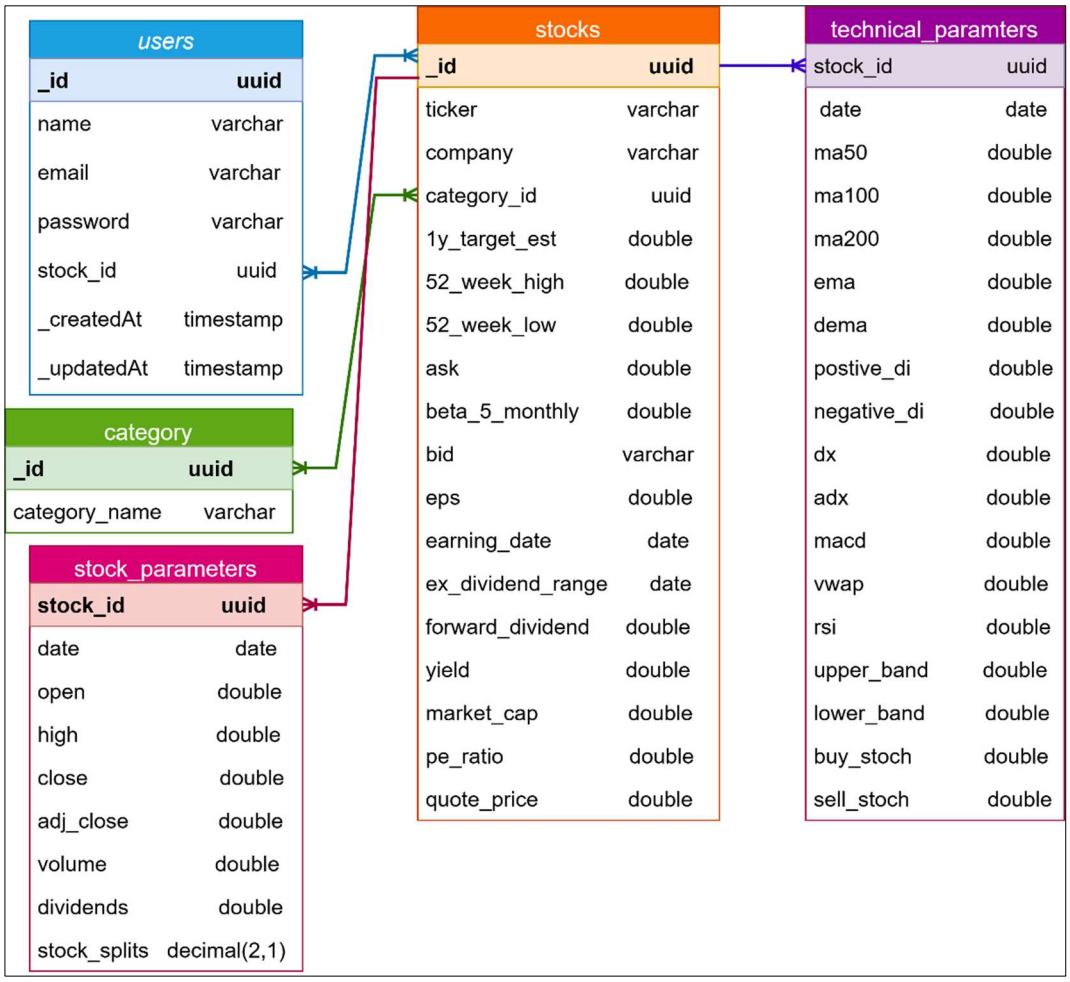

Fig 8.1 PostgreSQL Database Design

8.2 MONGODB DATABASE DESIGN

The news data will be stored in MongoDB Atlas, a NoSQL based database. Its design is shown in *Fig 8.2.*

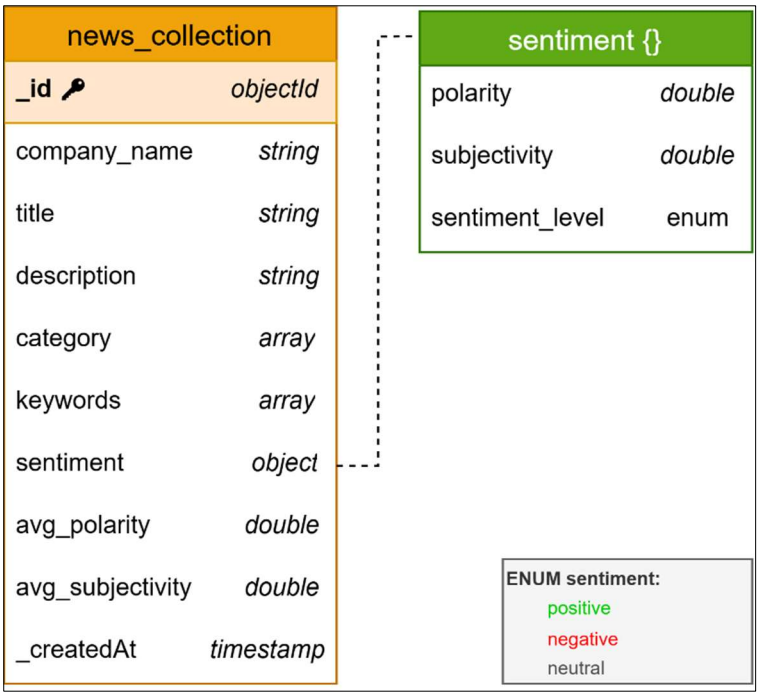

Fig. 8.2: MongoDB Database Design

CHAPTER 9

IMPLEMENTATION

9.1 DATA COLLECTION

9.1.1 Historical Stock Data Fetching Microservice

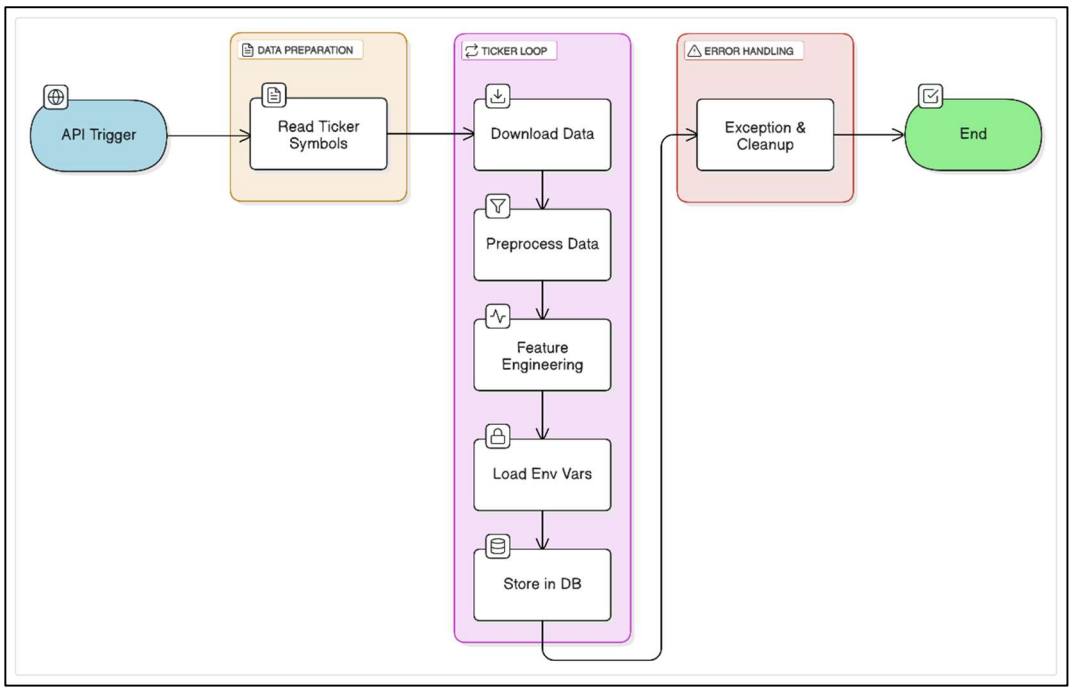

Fig. 9.1: Architecture of Historical Stock Data Collection Microservice

The data collection process begins with a GET request to the */generate-data* API endpoint, which triggers a pipeline that reads stock ticker symbols from a tickers.txt file. Historical daily stock data since 2000 is then downloaded using the *yfinance* library, with automatic adjustments for corporate actions like stock splits and dividends. The data is pre-processed by resetting the index and standardizing column formats for consistency. Technical indicators—such as RSI, MACD, ADX and Bollinger Bands—are generated using the *ta* library for feature engineering. Environment variables, including the database URL, are securely loaded with python-dotenv. Processed data is stored in a PostgreSQL database using

SQLAlchemy, with each ticker's data saved in a corresponding table that is replaced on each run to maintain freshness. Robust error handling ensures meaningful responses and safe resource cleanup.

9.1.2 Algorithm for Stock News Scrapper Microservice

Given a company *C*, the service processes news articles in several distinct stages, each designed to refine and structure the data for efficient retrieval and summarization. *Fig. 9.2* outlines the entire process.

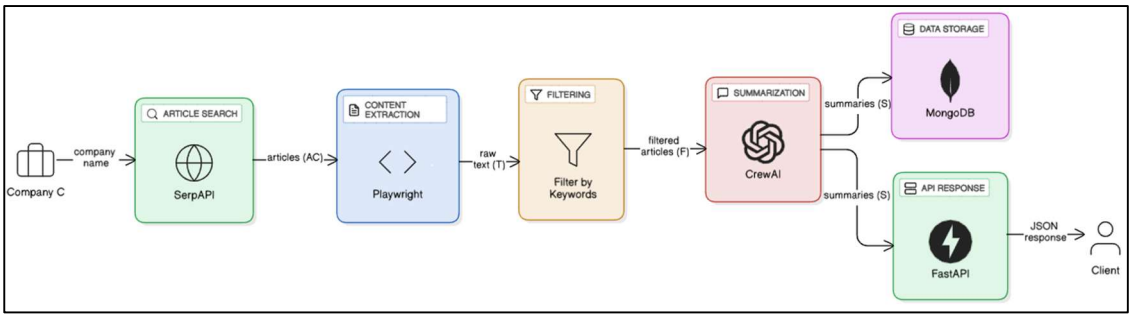

Fig. 9.2: Stock News Scrapper and Summarization Pipeline

Step 1: Article Search: Find all recent news articles related to the given company *C* using SerpAPI:

$$A_C = SerpAPI(C)$$

where:

- A_C is the set of all articles retrieved for the company *C*.
- The search returns a collection of metadata, including titles, URLs, and publication dates.

Step 2: Content Extraction: Extract the main content of each article found in the previous step using Playwright:

$$T = \{t_i \mid t_i = Playwright(a_i)\ for\ each\ a_i \in A_C\}$$

where:

- T is the set of raw text extracted from the URLs in A_C.
- Each t_i represents the main article body, stripped of ads, navigation, and other noise.

Step 3: Filtering: Filter the extracted content to keep only articles related to the stock market:

$$F = \{t_i \mid t_i \in T\ and\ re.search(K, t_i) \neq \emptyset\}$$

where:

- F is the set of filtered articles.
- K is a predefined set of stock-related keywords (e.g., “stock”, “market”, “shares”).
- The function research(K, t_i) checks if the content t_i contains any of the relevant keywords.

Step 4: Summarization: Generate concise summaries for each filtered article using CrewAI:

$$S = \{s_i \mid s_i = CrewAI(f_i)\ for\ each\ f_i \in F\}$$

where:

- S is the set of summaries, each typically 150-200 words.
- CrewAI uses the raw text to produce human-like summaries, capturing key points and insights.

Step 5: Data Storage: Store the summarized articles in the MongoDB database:

$$DB.insert(S)$$

where:

- The function DB.insert(S) saves each summary with its associated metadata.

Step 6: API Response

Return the structured response to the client as a JSON object:

$$Response = \{"company" : C, "articles" : S\}$$

where:

- The response includes the company name C and the list of summarized articles S.
- This response is served through FastAPI, making the data accessible to external clients.

9.2 STOCK PRICE PREDICTION – CNN STACKED BILSTM MODEL

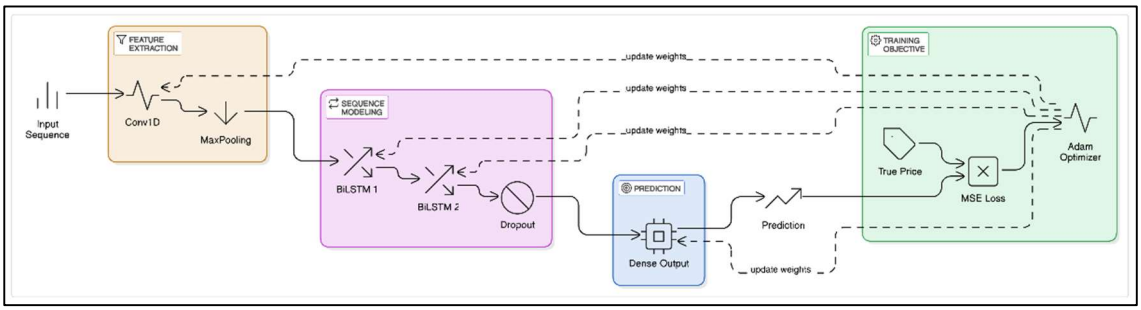

Fig. 9.3: Architecture of CNN – Stacked BiLSTM Model

Predict future stock prices using a hybrid deep learning model combining 1D Convolutional Neural Networks (CNN) and Stacked Bidirectional Long Short-Term Memory (BiLSTM) networks, based on historical price data and technical indicators.

Input and Data Preparation

Let:

- $P = \{p_1, p_2, \dots \dots, p_T\}$: Sequence of daily closing prices over T days.
- w: Length of the time window used to predict the next price.
- $X_t = \{p_{t+w-1}, \dots \dots, p_t\}$: Input sequence window.
- $y_t = p_{t+1}$: Target price to be predicted.

Technical Indicators

Moving Average (MA):

$$SMA = \frac{\sum_{i=1}^{n} P_i}{n}$$

Exponential Moving Average (EMA):

$$EMA_t = \alpha \times P_t + (1 - \alpha) \times EMA_{t-1}$$

MACD:

$$MACD\ Line = EMA(12) - EMA(26)$$

Bollinger Bands:

$$MACD\ Line = EMA(12) - EMA(26)$$

Momentum:

$$Momentum_t = p_t - p_{t-1}$$
$$log - Momentum_t = log(p_t - p_{t-1})$$

Feature Matrix and Target Vector Construct:

- Feature matrix $X_i \in R^{w \times f}$, where f is the number of features ($Features + Indicators$).
- Target output $y_i = p_{i+1} \in R$

CNN Feature Extraction

The first layer is a 1D convolution layer applied over the time axis to extract local patterns:

$$Z_{t,k}^{2} = ReLU(\sum_{i=0}^{K-1}\sum_{j=1}^{F} W_{i,j,k}^{(1)} . X_{t+i,j} + b_{k}^{(1)})$$

where

- K, kernel size (e.g., 3)
- $W^{(1)}$: convolution filter weights
- $b^{(1)}$: bias term
- $Z^{(1)}$: output feature map after convolution
- $ReLU(x) = max(0, x)$: activation function

MaxPooling Layer

After convolution, temporal down sampling is performed:

$$Z_{t,k}^{(2)} = max(Z_{t:tt+P,k}^{(1)})$$

Where:

- P: pooling window size (e.g., 2)
- This reduces temporal resolution but keeps dominant features.

Stacked BiLSTM Layers Pass the extracted features through stacked BiLSTM layers.

First Bidirectional LSTM Layer

LSTM processes sequence forward and backward:

$$\overrightarrow{h_t} = LSTM_{forward}(Z_t^{(2)})$$

$$\overleftarrow{h_t} = LSTM_{backward}\left(Z_t^{(2)}\right)$$

$$h_t^{(1)} = [\overrightarrow{h_t}; \overleftarrow{h_t}]$$

Where

- $h_t^{(1)} \in R^{2d}$; concatenation of forward and backward LSTM hidden states
- d: number of units in one LSTM direction

Second Bidirectional LSTM Layer

Same as above, applied on top of previous output:

$$\overrightarrow{h_t^{(2)}} = LSTM_{forward}(h_t^{(1)})$$

$$\overleftarrow{h_t^{(2)}} = LSTM_{backward}\left(h_t^{(1)}\right)$$

$$h_t^{(2)} = [\overrightarrow{h_t^{(2)}}; \overleftarrow{h_t^{(2)}}]$$

Dropout Layer

$$h_{dropped}^{(2)} = h^{(2)} \cdot m$$

Where $m \sim Bernoulli(1-p)$ is the dropout mask and p is the dropout probability

Output Layer

The last hidden state from BiLSTM is used for prediction:

$$\hat{y} = W_0 \cdot h_T^{(2)} + b_0$$

Where:

- $\hat{y}$: predicted price
- $h_T^{(2)}$: final hidden state (last time step)
- W_0, b_0: weights and bias for output layer

Loss Function (Mean Squared Error)

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

Where:

- N: Number of training examples.
- y_i: Actual price.
- $\hat{y}_i$: Predicted price.

Optimization Parameters are updated using an optimizer like Adam:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta L$$

Where η is the learning rate.

Inference For a new input sequence X_{new}:

$$\hat{y}_{future} = Model(X_{new})$$

9.3 NEWS SENTIMENT ANALYSIS – FINBERT MODEL

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model developed to capture deep contextual understanding of language by processing text bidirectionally. It employs multiple transformer encoder layers to learn rich and nuanced language representations, enabling superior performance on a wide range of natural language processing tasks.

Key Components:

- **Input Representation:** Combines token, segment, and positional embeddings.
- **Transformer Layers**: Stacks of multi-head self-attention and feed-forward layers.
- **Pre-training Objectives**: Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) for context learning**.**

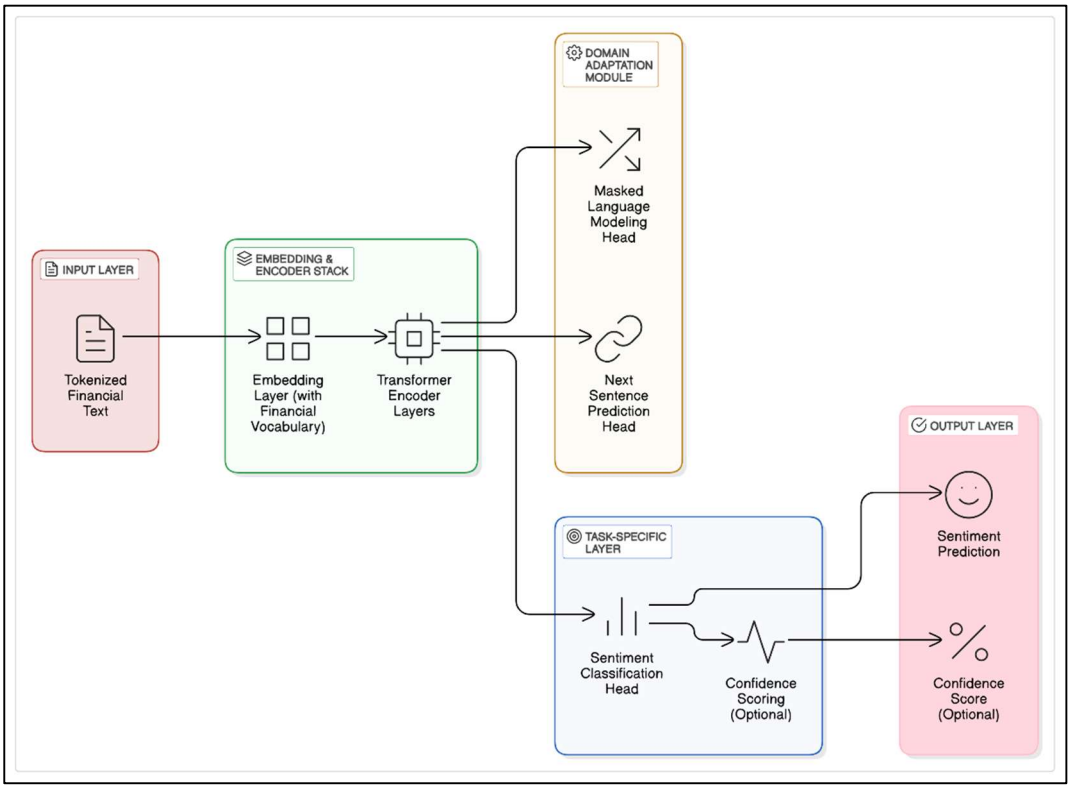

Fig. 9.4: Architecture of FinBERT Model

Importance of Fine-Tuning:

Fine-tuning is critical for adapting BERT to domain-specific tasks like financial sentiment analysis. It enables the model to:

- Capture domain-specific linguistic patterns (e.g., financial jargon).

- Achieve higher accuracy on downstream tasks like sentiment classification.
- Reduce the need for large-scale pre-training from scratch.

FinBERT Fine-Tuning Algorithm

Step 1: Domain-Specific Vocabulary Extension

Extend the BERT vocabulary with financial terms to handle domain-specific language effectively.

$$V_{fin} = V_{BERT} \cup V_{financial}$$

Step 2: Masked Language Modelling (MLM)

$$L_{fin}^{MLM} = -E_{x \sim X_{fin}} log\ P(x_m | x_{/m})$$

x: A tokenized input sequence sampled from the financial domain corpus V_{fin}.

x_m: The token (or tokens) that have been **masked** in the sequence x.

$x_{/m}$: The remaining tokens in the sequence **excluding the masked token(s)**.

Increase the masking probability for financial terms:

$$P(mask \mid x) = \begin{cases} 0.15\ if\ x \notin V_{financial} \\ 0.20\ \ if\ x \in V_{financial} \end{cases}$$

Step 3: Finetuning Stages

- **Stage 1: Domain Adaptation**

  $$L_{adapt} = \alpha L_{fin}^{MLM} + \beta L_{NSP}$$

  where,

  $\alpha = 0.7, \beta = 0.3$

  L_{NSP} NSP loss is the binary cross-entropy for sentence pair classification:

  $$L_{NSP} = -[y \cdot log(\hat{y}) + (1-y) \cdot log(1-\hat{y})]$$

  $$NSP_{label} = \begin{cases} 1, & if\ B\ is\ the\ next\ sentence\ after\ A \\ 0, & if\ B\ is\ a\ random\ sentence \end{cases}$$

  $y \in \{0,1\}$*:* true label for next sentence relationship

  $\hat{y}$: predicted probability from the model

- **Stage 2: Task-Specific Training**

  Optimize for the task-specific loss with regularization to prevent overfitting

  $$L_{sent} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\ log(\hat{y}_{ic}) + \lambda || \theta - \theta_{BERT} ||^2$$

  where,

  $y_{ic} \in \{0,1\}$ a binary indicator of whether example *i* belongs to class *c*

$\hat{y}_{ic} \in \{0,1\}$ **predicted probability** (from the softmax layer) that the i^{th} input belongs to class c.

Step 3: Optimization (AdamW)

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t} + \in} + \lambda(\theta_t - \theta_{BERT})$$

where,

Learning Rate (η) = 2e-5

Weight decay **(**λ)=0.01

Batch size = 32

Training Epochs = 3

Step 5: Linear Rate Schedule

Linear warmup for the first w steps, then decay

$$\eta_t = \begin{cases} \eta \cdot \frac{t}{w}, & ,if\ t < w \\ \eta \cdot max(0, \frac{T-t}{T-w}) & ,otherwise \end{cases}$$

Step 6: Sentiment Classification

Predicts the sentiment (positive, negative, neutral) for each input

$$P(y|x) = softmax(W_c . FinBERT(x) + b_c)$$

Step 7: Confidence Scoring

Measures the confidence of the sentiment prediction.

$$conf(x) = \max_y P(y|x) - \frac{1}{|Y|-1} \sum_{x \neq y} P(y'|x)$$

9.4 FINANCIAL CHATBOT IMPLEMENTATION

Our chatbot combines AI language capabilities with stock market analysis to make informed investment decisions.

Key Features:

- **Stock Performance Analysis:** Provides detailed historical price data, trend analysis, and key performance metrics for individual stocks.
- **Customizable Timeframes:** Enables users to analyze stock performance over various time periods such as days, weeks, months, or years.

- **Technical Indicators:** Supports a range of technical indicators including Relative Strength Index (RSI), moving averages, and other commonly used metrics in financial analysis.
- **Investment Signals:** Offers buy/sell recommendations based on technical analysis, helping users identify potential investment opportunities.
- **Stock Comparison Tool:** Allows for comparative analysis between multiple stocks to evaluate relative performance.
- **Natural Language Query Support:** Users can interact with the platform using plain English queries, such as "How has TCS performed in the last 6 months?" The system interprets the question and returns relevant analysis.
- **Visual Reporting:** Generates dynamic charts and graphs to visually represent stock trends, comparisons, and analytical insights.

Architecture

The architecture of the chatbot LLM is as follows:

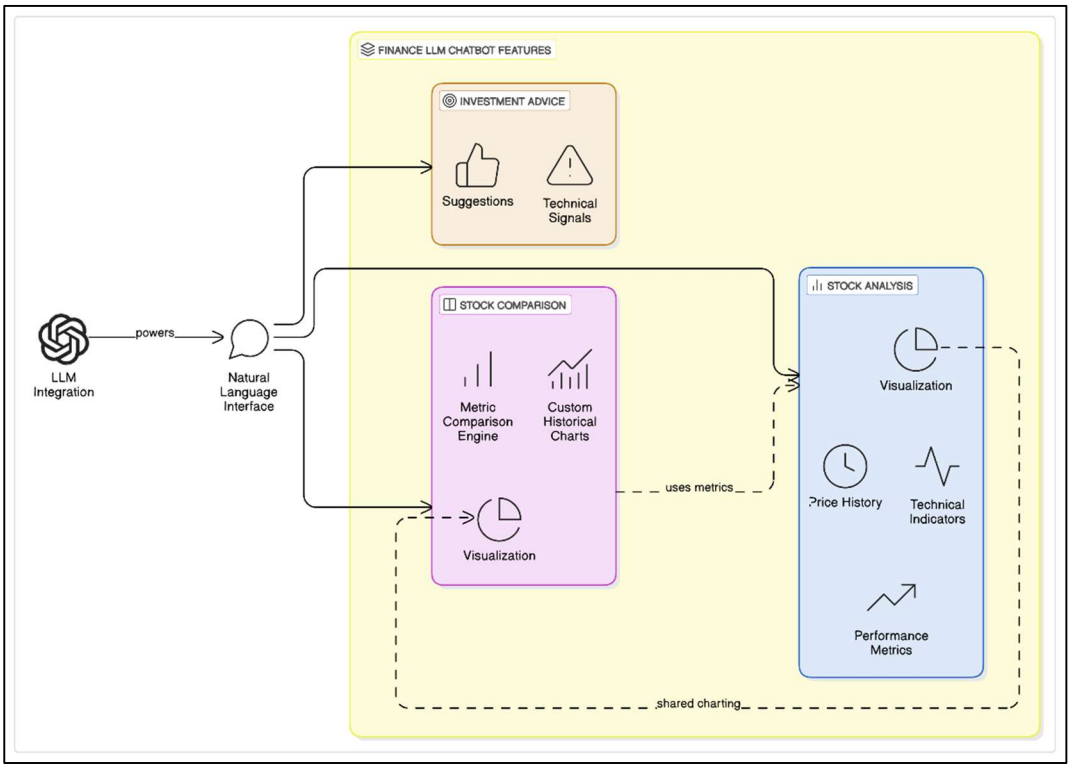

Fig. 9.5 Architecture of Financial LLM chatbot

The Finance LLM Chatbot is designed using a modular architecture that combines financial data analysis, natural language processing, and visualization to

provide insightful responses to user queries. At its core, the system is initiated through a command-line interface in main.py, which sets up the FinanceChatbot class along with its dependencies. These include services for stock data retrieval (StockDataService), entity mapping (CompanyMapper), technical analysis (StockAnalyzer), large language model interaction (LLMClient), and chart generation (Visualizer). Upon receiving a user query, the chatbot first determines the intent using keyword matching and regular expressions, then extracts relevant entities such as company names or stock tickers. If the query involves stock-related data, it fetches historical price data via *yfinance*.

Once data is obtained, the system computes technical indicators like moving averages, RSI, and price trends using StockAnalyzer, and creates visual charts using Visualizer. For broader financial or non-stock-specific questions, the chatbot leverages Hugging Face models. It selects an appropriate model (like FLAN-T5, LLaMA, or GPT-style models), formats the prompt accordingly, and communicates with the Hugging Face API while handling authentication, retries, and errors. The architecture is robust against failures, implementing fallback mechanisms for both data retrieval and LLM interaction, ensuring the chatbot remains functional even under adverse conditions. The final output combines structured data insights and LLM-generated explanations, offering users clear and accurate financial guidance.

9.5 FRONTEND DEVELOPMENT USING NEXT.JS

The frontend of Stokis is built using Next.js, a powerful React-based framework that enables both server-side rendering (SSR) and static site generation (SSG).

Features Leveraged

1. **API Integration**:
    - Next.js's API routes facilitate seamless communication between the frontend and the backend ML models. These routes enable efficient fetching of prediction results and historical data from the backend.
2. **Routing and Navigation**:
    - The built-in routing system of Next.js was utilized to create an intuitive and structured navigation experience. Users can easily

switch between features like stock analysis, prediction results, and user dashboards without delays.

Styling with Tailwind CSS

To ensure a modern and visually appealing design, Tailwind CSS was integrated into the Next.js project. Tailwind's utility-first approach allowed for rapid development of responsive and consistent UI components. Custom themes and components were designed to align with STOKIS's branding, creating a cohesive and professional appearance.

Responsiveness and Accessibility

The frontend was developed with a mobile-first design philosophy, ensuring compatibility across a variety of devices and screen sizes. Additionally, accessibility best practices were followed to make the application inclusive for users with diverse needs.

Advantages of Using Next.js

- **Performance**: With SSR and SSG, Next.js delivers optimal performance, ensuring users have a smooth experience.
- **Developer Experience**: Features like hot module replacement and built-in CSS support accelerated development.
- **Scalability**: The modular architecture of Next.js supports future expansion of Stokis with minimal technical debt.

By leveraging the capabilities of Next.js and Tailwind CSS, the frontend of Stokis achieves a balance between functionality, aesthetics, and user experience, laying the groundwork for an effective and reliable stock prediction application.

CHAPTER 10

RESULTS & DISCUSSION

10.1 PERFORMANCE OF CNN – STACKED BiLSTM

In this project, a hybrid deep learning architecture combining Convolutional Neural Networks (CNN) with stacked Bidirectional Long Short-Term Memory (BiLSTM) layers was implemented to predict stock prices, specifically for the Bank Nifty index. This hybrid approach was chosen to leverage the strengths of both architectures: CNN for local feature extraction and stacked BiLSTM for capturing long-term temporal dependencies in sequential financial data.

10.1.1 Model Training

The model was trained using historical stock data obtained from Yahoo Finance, enriched with a variety of technical indicators, including moving averages, MACD, RSI, ADX, etc. The dataset was split into training and testing sets, with a look-back window of 60 time steps to capture sufficient historical context.

The model was compiled using the Adam optimizer and mean squared error as the loss function. It was trained over 75 epochs with early stopping to prevent overfitting, using a batch size of 128 and a validation split of 20%.

10.1.2 Comparative Study: Transitioning from Stacked LSTM to CNN-Stacked BiLSTM for Stock Price Prediction

The comparative study of transitioning from Stacked LSTM model to CNN-Stacked BiLSTM model highlights the significant improvement in performance achieved by integrating convolutional and bidirectional recurrent layers. The Stacked LSTM architecture, comprising a sequential stack of LSTM layers with dropout for regularization, achieves an average test accuracy of approximately 70%. While LSTMs are effective in capturing temporal dependencies, they often struggle with hierarchical or local features that can be critical in certain tasks.

On the other hand, the CNN-Stacked BiLSTM model combines the strengths of Convolutional Neural Networks (CNNs) for local feature extraction and Bidirectional LSTMs (BiLSTMs) for contextual sequence learning. The convolutional layers effectively detect salient patterns in subsequences, which are then passed to BiLSTMs that process the data in both forward and backward directions. This hybrid approach significantly boosts the model's learning capability, achieving an average test accuracy of 93.54%, far surpassing the LSTM-only model. This clearly demonstrates the superiority of the CNN-BiLSTM architecture in capturing both spatial and sequential dependencies for complex pattern recognition tasks.

Layer (type)	**Output Shape**	**Param #**
lstm_8 (LSTM)	(None, 60, 50)	12,600
dropout_8	(None, 60, 50)	0
lstm_9 (LSTM)	(None, 60, 60)	26,640
dropout_9	(None, 60, 60)	0
lstm_10 (LSTM)	(None, 60, 80)	45,120
dropout_10	(None, 60, 80)	0
lstm_11 (LSTM)	(None, 120)	96,480
dropout_11	(None, 120)	0
dense_2 (Dense)	(None, 1)	121

Table 10.1 Stacked LSTM Model Architecture

Layer (type)	**Output Shape**	**Param #**
conv1d (Conv1D)	(None, 58, 64)	2,368
max_pooling1d	(None, 29, 64)	0
conv1d_1 (Conv1D)	(None, 27, 32)	6,176
max_pooling1d_1	(None, 13, 32)	0
bidirectional (BiLSTM)	(None, 13, 100)	33,200
bidirectional_1 (BiLSTM)	(None, 50)	25,200
dense_1 (Dense)	(None, 16)	816
dropout_4	(None, 16)	0
dense_2 (Dense)	(None, 1)	17

Table 10.2 CNN-Stacked BiLSTM Model Architecture

10.1.3 CNN – Stacked BiLSTM Model Evaluation

We evaluated our stock prediction model across 60+ publicly listed companies from diverse sectors including Automotive, Banking, FMCG, Pharmaceuticals, Real Estate, Power, IT, and Metals. The model was trained on historical data obtained from Yahoo Finance and tested for its generalization ability using a hold-out test set. The model achieved high prediction accuracy in stable, trend-following stocks, particularly in sectors such as Automotive, Public-Sector Banking, and Energy. *Table 10.3* shown below is a summary of the accuracy across key sectors:

Sector	**Representative Stocks**	**Accuracy Range**
Energy	ONGC, NTPC, POWERGRID	94–97%
Automotive	TVSMOTOR, BAJAJ-AUTO, EICHERMOT	93–97%
Banking	SBIN, BANKBARODA, UNIONBANK	90–96%
Pharmaceuticals	SUNPHARMA, CIPLA, DRREDDY	87–97%
FMCG	ITC, COLPAL, NESTLEIND	85–95%
Real Estate	DLF, GODREJPROP, OBEROIRLTY	82–96%
Metals & Mining	JSWSTEEL, TATASTEEL, VEDL	75–92%
IT Services	TCS, INFY, WIPRO	70–80%

Table 10.3 Sector-wise predictive performance

The model achieved the highest accuracy for Sun Pharma (97.87%), NTPC (97.45%), TVS Motor (96.89%), and Bank of Baroda (96.32%), indicating strong predictive alignment with price trends in these stocks. It has also performed great for international stocks like Goldman Sachs (95.32%), NVIDIA (92.32%), Meta (91.61%), etc. On the contrary, few stocks like Britannia (73.04%), TCS (70.79%), Wipro (70.69%) exhibited poor performance, due to high volatility, external event sensitivity, or model feature inadequacy. The average test accuracy across all valid stocks was approximately **93.54%,** indicating the model's robust predictive capability across diverse sectors and market behaviours

10.1.4 Visualization

The prediction plot *Fig. 10.1* clearly demonstrates the model's ability to track the general trend and fluctuations of the Sun Pharma stock. Although some

deviations exist, the forecasted prices align well with the actual movements, indicating the effectiveness of combining CNN with BiLSTM for stock price prediction.

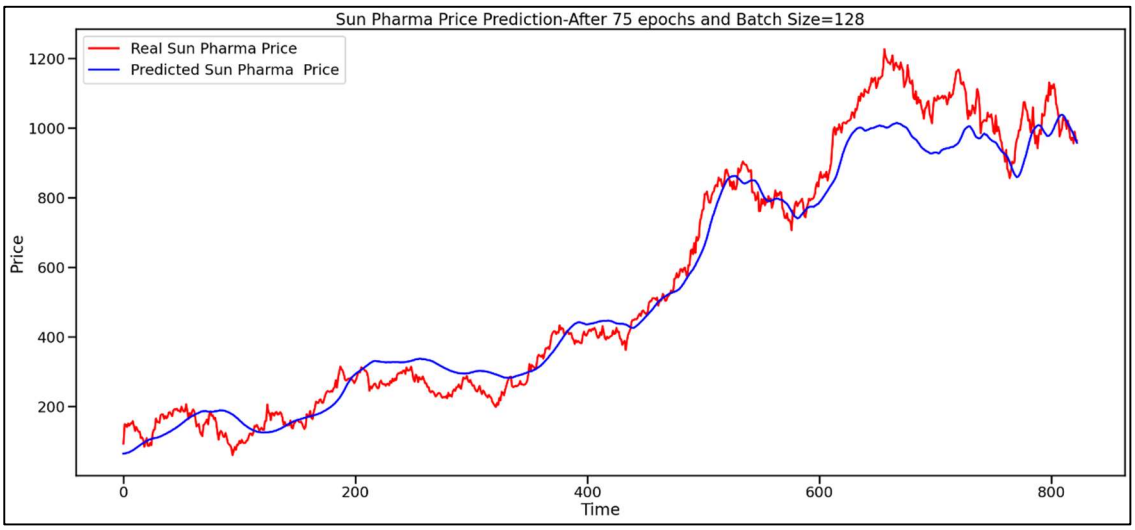

Fig. 10.1 Sun Pharma Price Prediction Visualization

10.1.5 Stochastic and Modelling Insights

The prediction accuracy varied slightly across stocks, suggesting the presence of heteroscedastic behaviour and sectoral volatility. The following insights emerged:

- Stocks with strong mean-reversion or trending patterns yielded higher accuracy, implying that the model learned consistent patterns effectively.
- Volatile stocks or those affected by exogenous events (e.g., IT sector, financial sector stocks sensitive to regulatory decisions) exhibited poor predictability.

10.1.6 Discussion

The results suggest that the CNN-Stacked BiLSTM model is capable of learning complex patterns in financial time series data. The CNN layers efficiently extracted local features, while the BiLSTM layers modelled the temporal dependencies in both forward and backward directions.

The results also affirm that our model performs reliably on structured, trend-based stocks and can serve as a valuable forecasting tool in stable market conditions. However, for volatile or highly reactive stocks, the model requires further optimization and feature enhancement. These findings offer a foundation for future improvements using hybrid deep learning architectures and attention-based temporal modelling.

10.2 PERFORMANCE OF FINBERT MODEL

This section presents the outcomes of applying the fine-tuned FinBERT model for financial news sentiment analysis. FinBERT, a pre-trained BERT model specifically adapted for financial sentiment classification, was fine-tuned on a labelled dataset consisting of financial news articles. The objective was to classify each article into one of three sentiment categories: positive, neutral, or negative.

10.2.1 Model Training and Evaluation

The financial news dataset obtained from Kaggle (Sentiment Analysis for Financial News by Ankur Sinha) was split into training (81%), validation (9%), and testing (10%) subsets, stratified by the sentiment label to maintain class balance. The model was fine-tuned using the Hugging Face Trainer API for 5 epochs, with early stopping and evaluation on the validation set after each epoch. Tokenization was performed using the ProsusAI/finbert tokenizer, and label encoding was aligned with the model's expectations (positive: 0, negative: 1, neutral: 2).

The fine-tuned model achieved the following performance:

Accuracy: 97.28%

Table 10.4 represents the class-wise value of Precision and Recall.

Class	Precision	Recall
Positive	0.960	0.982
Negative	0.982	0.973
Neutral	0.958	0.968

Table 10.4 Precision and Recall per Class

This high accuracy with good precision and recall scores indicates that the model is highly effective in classifying the sentiment of financial news articles, with minimal misclassification.

10.2.2 Visualization

To further analyze the model's performance, a confusion matrix was plotted to visualize the classification outcomes across the three sentiment classes. The *Fig 10.2* below shows a clear diagonal dominance, indicating that most predictions were correct and closely aligned with the true labels.

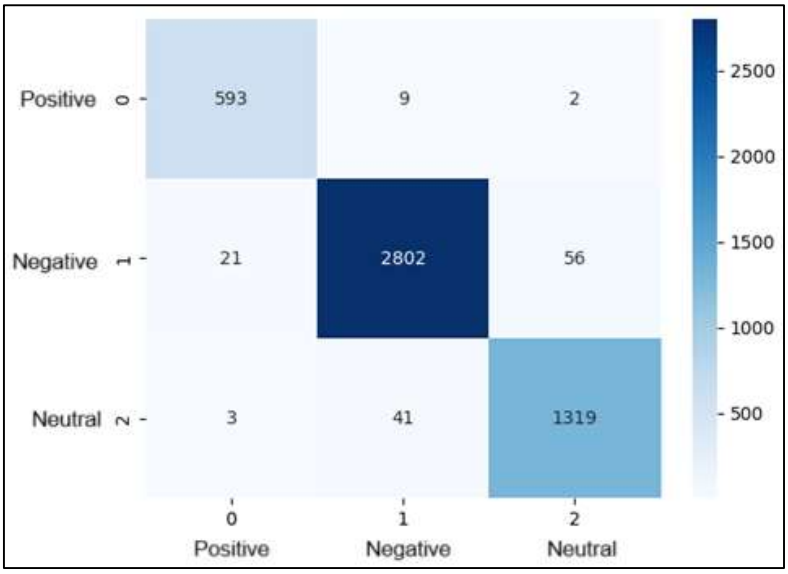

Fig. 10.2 Confusion Matrix for FinBERT Model

This heatmap offers an intuitive understanding of the model's precision across categories:

- Strong accuracy in distinguishing positive and negative news.
- Slight overlap in misclassifying neutral articles, which is often a challenge due to their subtle sentiment cues.

10.2.3 Model Deployment and Usability

After training, the FinBERT model and tokenizer were saved locally and then loaded into a Hugging Face pipeline for sentiment inference. This facilitates direct sentiment extraction from raw financial text inputs, making the model highly suitable for integration into real-time financial analysis platforms.

10.2.4 Discussion

The fine-tuned FinBERT model demonstrates excellent generalization on unseen financial news data. This performance is crucial for use cases like:

- Enhancing stock price prediction by incorporating sentiment as a feature.
- Supporting traders in understanding market sentiment trends.
- Automating news filtering and classification in financial platforms.

The key strengths of this model include:

- Context-aware sentiment understanding using BERT’s bidirectional encoding.
- Domain-specific accuracy due to FinBERT's financial pretraining.

However, the model may still face challenges with ambiguous or sarcastic language, and performance may vary with different news sources or regions.

CHAPTER 11

CONCLUSION

This project successfully developed and evaluated an intelligent, AI-driven platform for stock market prediction by integrating historical stock price analysis with real-time news sentiment using advanced machine learning and natural language processing techniques. The primary objective was to enhance prediction accuracy and provide investors with a reliable tool for decision-making.

To achieve this, a hybrid CNN-Stacked BiLSTM model was implemented, which demonstrated significant improvements in prediction accuracy compared to conventional LSTM-only models. Specifically, the CNN-BiLSTM model achieved a test accuracy of 93.54%, while the LSTM-only model reached approximately 70%. This confirmed the effectiveness of combining convolutional layers (for capturing local temporal features) with bidirectional recurrent layers (for understanding sequence dynamics from both past and future contexts).

In parallel, FinBERT, a domain-specific transformer model, was fine-tuned on a labelled dataset of financial news articles to classify sentiment into positive, neutral, or negative. The FinBERT model achieved a high test accuracy of 97.28%, highlighting its ability to capture the nuanced and context-sensitive language used in financial news. This sentiment data was incorporated into the platform to complement technical analysis and further enhance stock price prediction accuracy.

Additionally, the project incorporated a GPT-based financial chatbot, providing users with an interactive interface to ask natural language queries about stock performance, trends, and news sentiment. The chatbot was able to leverage both structured data and model outputs to generate informative and real-time responses.

Overall, the platform addressed several limitations of existing systems, such as lack of real-time sentiment integration, absence of interactive guidance, and limited predictive capability. The use of a microservices architecture, containerized deployment using Docker, and cloud scalability ensures that the system is robust, maintainable, and ready for real-world applications. The user-friendly frontend built with Next.js and Tailwind CSS offers a modern, responsive interface for seamless user interaction.

CHAPTER 12

FUTURE WORKS

While this project lays a strong foundation for AI-powered stock market prediction, there remain several avenues for future enhancement and expansion:

1. Integration of Macroeconomic and Global Indicators

Currently, the model relies primarily on technical indicators and news sentiment. Future versions could incorporate macroeconomic variables (e.g., inflation rate, interest rate, unemployment) and global events (e.g., geopolitical risks, commodity prices) to offer broader market context and improve prediction robustness.

2. Real-Time Sentiment Streaming

Instead of analyzing static news data, future iterations can include real-time streaming sentiment analysis using live RSS feeds, social media APIs (like Twitter or Reddit), and financial news wires. This would provide more immediate and reactive market insights.

3. Multi-Asset Support

The platform currently supports stock-level predictions. Future enhancements can expand its scope to include commodities, cryptocurrencies, mutual funds, and ETFs, offering a more comprehensive financial advisory system.

4. Enhanced Explainability with SHAP or LIME

To improve model transparency and interpretability—particularly important for regulated financial environments—techniques such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) could be integrated. These tools would help explain model predictions to users in a human-understandable manner.

5. Adaptive Learning Models

To adapt to changing market dynamics, reinforcement learning or online learning models can be integrated to continuously retrain and update the system with the latest data. This will allow the platform to evolve with the market rather than rely on static historical data alone.

6. Mobile Application Development

A mobile version of the platform can significantly enhance user accessibility and engagement. Push notifications for buy/sell signals, real-time alerts on sentiment shifts, and voice assistant integration could be explored in future development.

7. Portfolio Optimization Module

In addition to stock-level prediction, a future enhancement could include a portfolio management system that provides asset allocation suggestions based on risk tolerance, historical returns, and market sentiment. Techniques such as Markowitz Portfolio Theory or Monte Carlo Simulation could be applied.

8. Multilingual Support

Expanding the platform to support sentiment analysis in regional languages (e.g., Hindi, Bengali, Tamil) will increase inclusivity and applicability across broader user demographics in multilingual countries like India.

9. Blockchain Integration

To ensure data integrity and transparent audit trails for predictions and model decisions, future versions of the platform may explore storing prediction logs and transaction histories on a blockchain ledger.

10. Regulatory Compliance Features

With increasing scrutiny in the FinTech space, future versions can integrate modules for automated compliance checks, GDPR-ready user data handling, and AI ethics auditing tools.

Appendices

Appendix A: List of Technical Terms

Technical Term	Description
1-Year Target Estimate	The average stock price forecasted by analysts for the next year, used to predict potential growth or decline in stock value.
52-Week High	The highest price at which a stock has traded in the past 52 weeks, indicating peak performance.
52-Week Low	The lowest price at which a stock has traded in the past 52 weeks, reflecting worst performance.
Adam Optimizer	An optimization algorithm used for updating model parameters during training, based on adaptive moment estimation.
API Routes	Next.js feature for creating server-side API endpoints to facilitate communication between frontend and backend.
Ask Price	The lowest price a seller is willing to accept for a stock, indicating supply-side dynamics.
Authentication System	A system using OAuth2 with JWT for securing user access and protecting data.
AutoRegressive Integrated Moving Average (ARIMA)	A statistical model for time series forecasting, focusing on linear relationships.
Average Directional Index (ADX)	A technical indicator measuring trend strength, with values above 25 indicating a strong trend and below 20 indicating a weak trend.
Backend Framework	Software framework (e.g., FastAPI) used to build server-side logic and APIs.
Batch Size	The number of training examples processed in one iteration during model training (e.g., 32 or 128).
Beta (5-Monthly)	A measure of a stock's volatility compared to the market over five months, indicating risk level.
Bid Price	The highest price a buyer is willing to pay for a stock, reflecting demand-side dynamics.
Bidirectional Long Short-Term Memory (BiLSTM)	A recurrent neural network architecture processing sequences in both forward and backward directions to capture temporal dependencies.

Binary Cross-Entropy	A loss function used for binary classification tasks, such as next sentence prediction in NLP models.
Blockchain Ledger	A decentralized digital ledger for ensuring data integrity and transparent audit trails.
Bollinger Bands (BBands)	A volatility indicator consisting of a simple moving average and upper/lower bands at standard deviation levels, used for identifying overbought/oversold conditions.
Chart Generation	The process of creating visual representations (e.g., graphs, plots) of stock trends and technical indicators.
Cloud Hosting	Deployment of applications on cloud platforms like AWS, Azure, or GCP for scalability and accessibility.
Company Mapper	A service in the chatbot architecture for mapping company names to stock tickers or other identifiers.
Confusion Matrix	A visualization tool showing the performance of a classification model by comparing predicted and actual labels.
Containerization	The use of Docker to package applications and their dependencies into portable containers.
Continuous Integration/Continuous Deployment (CI/CD)	Automated pipelines (e.g., GitHub Actions, Jenkins) for testing and deploying code changes.
Convolutional Neural Network (CNN)	A deep learning model using convolutional layers to extract local patterns from data, applied to time series in this project.
Cron Jobs	Scheduled tasks for automating data collection, such as fetching stock data or news articles.
Data Collection Microservice	A microservice responsible for fetching stock data from sources like Yahoo Finance and storing it in a database.
Data Flow Diagram (DFD)	A diagram illustrating the flow of data through a system, showing interactions between entities, processes, and data stores.
Data Storage	The process of storing structured (e.g., PostgreSQL) and unstructured (e.g., MongoDB) data for analysis and retrieval.
Database Schema Design	The process of defining the structure and organization of database tables or collections.
Deep Learning	A subset of machine learning using neural networks with multiple layers to model complex patterns.
Dividend Yield	A stock's annual dividend payments as a percentage of its price, measuring income generation potential.

Docker Compose	A tool for defining and running multi-container Docker applications.
Dropout Layer	A regularization technique in neural networks where random units are dropped during training to prevent overfitting.
Early Stopping	A technique to halt model training when validation performance stops improving, preventing overfitting.
Earnings Date	The scheduled date for a company to report its earnings, providing key financial updates.
Earnings Per Share (EPS)	A company's profit allocated to each outstanding share, assessing profitability.
Entity Mapping	The process of identifying and mapping entities (e.g., company names, stock tickers) in user queries.
Event Bus	A messaging system for facilitating communication between microservices in a microservices architecture.
Ex-Dividend Date	The date a stock trades without the value of its next dividend payment, determining dividend eligibility.
Exponential Moving Average (EMA)	A moving average giving more weight to recent prices, used for trend identification.
FastAPI	A Python-based web framework for building high-performance RESTful APIs.
Feature Engineering	The process of creating and selecting features (e.g., technical indicators) to improve model performance.
Feature Matrix	A matrix of input features (e.g., technical indicators, historical prices) used for training machine learning models.
FinBERT	A BERT model fine-tuned for financial sentiment analysis, classifying text as positive, negative, or neutral.
Financial News Articles	Textual data from news sources used for sentiment analysis in stock prediction.
Frontend Framework	Software framework (e.g., Next.js) used to build user interfaces.
Generalized AutoRegressive Conditional Heteroskedasticity (GARCH)	A statistical model for modelling time series volatility.
Git	A version control system for tracking code changes, hosted on platforms like GitHub.
Hugging Face Transformers	A library providing pre-trained NLP models like FinBERT for sentiment analysis.

Hybrid Model	A model combining multiple architectures (e.g., CNN and BiLSTM) to leverage their strengths.
Input Representation	The combination of token, segment, and positional embeddings in transformer models like BERT.
Keyword Matching	A technique for identifying user intent in queries by matching specific keywords.
Learning Rate	A hyperparameter controlling the step size during model optimization (e.g., 2e-5 for FinBERT).
Linear Rate Schedule	A learning rate adjustment strategy with linear warmup and decay during training.
Look-Back Window	The number of historical time steps (e.g., 60) used as input for predicting future values.
Loss Function	A function measuring the difference between predicted and actual values (e.g., Mean Squared Error).
Market Capitalization	The total value of a company's outstanding shares, reflecting its size and market value.
Masked Language Modelling (MLM)	A pre-training objective in BERT where tokens are masked, and the model predicts them.
MaxPooling Layer	A layer in CNNs that down samples feature maps by selecting maximum values, reducing temporal resolution.
Mean Absolute Error (MAE)	A metric measuring the average absolute difference between predicted and actual values.
Mean Squared Error (MSE)	A metric measuring the average squared difference between predicted and actual values.
Microservices Architecture	A design pattern where the application is split into small, independent services.
Mobile-First Design	A design approach prioritizing compatibility with mobile devices for responsive UI.
Model-View-Controller (MVC)	A software architectural pattern dividing application logic into model, view, and controller components.
Momentum	A technical indicator measuring the rate of price change over a period.
MongoDB Atlas	A cloud-based NoSQL database for storing unstructured data like news articles.
Moving Average (MA)	A calculation averaging data points over a period to identify trends (e.g., Simple Moving Average, Exponential Moving Average).
Moving Average Convergence Divergence (MACD)	A trend-following momentum indicator based on the difference between two moving averages.

Multi-Head Self-Attention	A mechanism in transformer models allowing the model to focus on different parts of the input simultaneously.
Natural Language Processing (NLP)	A field of AI focused on processing and understanding human language.
Neural Networks	Computational models inspired by the human brain, used for tasks like financial forecasting.
Next Sentence Prediction (NSP)	A pre-training objective in BERT for predicting whether two sentences are consecutive.
Next.js	A React-based framework for building server-side rendered and static web applications.
OAuth2	An authentication protocol using JSON Web Tokens (JWT) for secure user access.
Overbought/Oversold Conditions	Market states indicated by technical indicators like RSI or Stochastic Oscillator, suggesting potential price reversals.
Portfolio Optimization	Techniques like Markowitz Portfolio Theory for optimizing asset allocation based on risk and return.
PostgreSQL	A relational database management system for storing structured stock data.
Price-to-Earnings Ratio (P/E)	The ratio of a company's share price to its earnings per share, assessing valuation.
Quote Price	The current market price of a stock, used for real-time trading decisions.
R-squared (R^2)	A metric indicating the proportion of variance in the dependent variable explained by the model (e.g., 0.946 for CNN-BiLSTM).
Regular Expressions	Patterns used for extracting entities or matching keywords in text processing.
Reinforcement Learning	A machine learning approach where models learn by interacting with an environment to maximize rewards.
Relative Strength Index (RSI)	A momentum oscillator measuring price change speed and magnitude, indicating overbought/oversold conditions.
Representational State Transfer (REST)	An architectural style for designing networked applications, used in API design.
Responsive UI	A user interface designed to adapt seamlessly to different devices and screen sizes.
Sentiment Analysis	The process of determining the emotional tone (positive, negative, neutral) of text, such as financial news.

Sentiment Classification	Assigning a sentiment label (positive, negative, neutral) to text inputs using models like FinBERT.
SerpAPI	An API for searching and retrieving news articles from the web.
SHAP (SHapley Additive exPlanations)	A method for explaining model predictions by assigning feature importance.
Simple Moving Average (SMA)	A moving average calculated by averaging prices over a fixed period.
Static Site Generation (SSG)	A Next.js feature for pre-rendering web pages at build time for performance.
Stock Analyzer	A service in the chatbot architecture for computing technical indicators and analyzing stock trends.
Stock Data Service	A service for retrieving historical stock data, typically via APIs like yfinance.
Stochastic Oscillator	A momentum indicator comparing a stock's closing price to its price range over a period, used for identifying overbought/oversold conditions.
Support Vector Machine (SVM)	A machine learning model for classification and regression, used for identifying non-linear patterns.
Tailwind CSS	A utility-first CSS framework for rapid and responsive UI development.
Technical Indicators	Metrics like RSI, MACD, and Bollinger Bands used to analyze stock price trends and signals.
Temporal Dependencies	Relationships in time series data where past and future values influence predictions.
Time Series Analysis	The study of ordered data points to identify patterns, trends, and dependencies.
Tokenization	The process of converting text into tokens for input to NLP models like FinBERT.
Transformer Layers	Layers in transformer models (e.g., BERT) using multi-head self-attention and feed-forward networks.
Use-Case Diagram	A diagram showing interactions between users and system functionalities.
Version Control	The management of code changes using tools like Git.
Visualizer	A service in the chatbot architecture for generating charts and graphs of stock data.
Volume Weighted Average Price (VWAP)	An average price weighted by trading volume, used to assess execution quality and support/resistance levels.
yfinance	A Python library for downloading historical stock data from Yahoo Finance.

Appendix B: Abbreviations Used

Abbreviation	Full Form / Contextual Meaning
AC	Set of all articles retrieved for the company
ADX	Average Directional Index
API	Application Programming Interface
ARIMA	AutoRegressive Integrated Moving Average
ASGI	Asynchronous Server Gateway Interface
AWS	Amazon Web Services
BBands	Bollinger Bands
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short-Term Memory
CI/CD	Continuous Integration/Continuous Deployment
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
DFD	Data Flow Diagram
EMA	Exponential Moving Average
EPS	Earnings Per Share
ETFs	Exchange-Traded Funds
FinBERT	Financial BERT (BERT model fine-tuned on financial data)
FLAN-T5	Fine-tuned LAnguage Net Text-To-Text Transfer Transformer
GARCH	Generalized AutoRegressive Conditional Heteroskedasticity
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
JSON	JavaScript Object Notation
JWT	JSON Web Token
LIME	Local Interpretable Model-Agnostic Explanations

LLaMA	Large Language Model Meta AI
LLM	Large Language Model
LSTM	Long Short-Term Memory
MA	Moving Average
MACD	Moving Average Convergence Divergence
MAE	Mean Absolute Error
ML	Machine Learning
MLM	Masked Language Modelling
MSE	Mean Squared Error
MVC	Model-View-Controller
NLP	Natural Language Processing
NSP	Next Sentence Prediction
P/E	Price-to-Earnings Ratio
R^2	R-squared
REST	Representational State Transfer
RSI	Relative Strength Index
SHAP	SHapley Additive exPlanations
SMA	Simple Moving Average
SSG	Static Site Generation
SSR	Server-Side Rendering
SVM	Support Vector Machine
UI/UX	User Interface/User Experience
VWAP	Volume Weighted Average Price

Bibliography

1. Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.
2. Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econometrical: Journal of the econometric society, 987-1007.
3. Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. Journal of econometrics, 31(3), 307-327.
4. Vapnik, V. (2013). The nature of statistical learning theory. Springer science & business media.
5. McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5, 115-133.
6. Haykin, S. (1998). Neural networks: a comprehensive foundation. Prentice Hall PTR.
7. Graves, Alex. Supervised Sequence Labelling with Recurrent Neural Networks. 2012th ed. vol. 385. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. Web.
8. Shen, J., Shafiq, M.O. (2020). Short-term stock market price trend prediction using a comprehensive deep learning system. J Big Data 7, 66.
9. Shankarlingam G, Reddy KT. (2023) Predicting a Small Cap Company Stock Price using Python with Best Accuracy Rate: How the Data Science Working for Predictions and Accuracy Rate. Indian Journal of Science and Technology. 16(48): 4620-4623.
10. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in information retrieval, 2(1–2), 1-135.
11. Ding, X., Zhang, Y., Liu, T., & Duan, J. (2014, October). Using structured events to predict stock price movement: An empirical investigation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1415-1425).
12. Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of computational science, 2(1), 1-8.