SUMMER INTERNSHIP [2025]

Project Report on

"AI- Driven Stock Prediction Tool"

By -

ADITYA S HEGDE

Under the supervision of

Dr.Shriram Hegde, IIT Delhi

Prof. C. S. Rao, IIT Delhi

Duration

1st July - 1st August 2025

CERTIFICATE

DECLARATION

I, **ADITYA S HEGDE,** am a student of the fifth semester B.Tech in Computer Science and Engineering (AI&ML), at School of Engineering, Dayananda Sagar University, hereby declare that the Internship project titled "**AI-Driven Stock Prediction Tool**" has been carried out by me and submitted in complete fulfillment of the Internship.This project was undertaken during the period from 1st July to 1st August 2025, under the joint mentorship of **IIT Delhi** and **Aspire InfoLabs**. The work submitted here is a **result of my efforts, research, development, and learnings.**.

ACKNOWLEDGEMENT

It is with immense gratitude that I take this opportunity to acknowledge and appreciate the support, guidance, and mentorship I received during the successful completion of my internship project titled **"AI-Driven Stock Price Prediction with Sentiment Analysis."**

I would first like to extend my heartfelt thanks to **DR.SHRIRAM HEGDE Sir** , whose continuous guidance, encouragement, and belief in my capabilities played a pivotal role in helping me secure this internship opportunity. His mentorship, alongside that of **PROD. C. S. RAO Sir**, both esteemed faculty members at **IIT Delhi**, has been invaluable throughout the project. Their deep insights, constructive feedback, and technical expertise shaped this project from its foundational concepts to its final implementation.

I am also sincerely thankful to **Aspire InfoLabs** for collaborating on this internship and providing practical exposure to real-world challenges in financial data analysis and AI deployment. Their professional environment and industry-aligned expectations pushed me to apply theoretical knowledge into meaningful solutions.

Finally, I would like to acknowledge the constant support of my peers throughout the journey to deliver the best possible version of this project.

This experience has been transformative, and I am deeply grateful to all those who made it possible.

-ADITYA S HEGDE

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This internship focused on creating an AI-driven stock price prediction tool combining LSTM-based time series analysis with sentiment extracted from news headlines. The pipeline handles financial data acquisition, exploratory analysis, sophisticated merging of asynchronous news and price data, model training, and deployment in a Streamlit interface. Practical challenges such as API limits, data quality, and portability across US and Indian stock markets were addressed end-to-end. The experience significantly advanced my understanding of deploying deep learning and NLP solutions for real-world financial forecasting.

CHAPTER 1 – INTRODUCTION

The project titled **"AI-Driven Stock Price Prediction Tool"** is developed in this context, aiming to combine both numerical and textual data to build a more informative prediction pipeline. In the rapidly evolving landscape of financial technology, forecasting stock prices with precision and reliability remains one of the most challenging tasks for researchers, investors, and analysts. While traditional models often rely solely on numerical data such as historical prices, moving averages, and statistical indicators modern financial systems are also heavily influenced by qualitative factors like news events, market sentiment, and socio-political developments therefore this project is focused on giving results based on such data.

Specifically, the system leverages **Long Short-Term Memory (LSTM)** networks a type of deep learning architecture suitable for time-series forecasting to predict next day stock closing prices. This model is further enhanced by incorporating sentiment analysis of financial news headlines, adding a layer of contextual understanding to the purely quantitative approach.

The internship focused on building a platform that includes:

Data acquisition (from stock APIs and web scraping),

Exploratory Data Analysis (EDA),

Data preprocessing and engineering,

Sentiment scoring using NLP techniques,

Model training and evaluation

Deployment using an interactive dashboard built with **Streamlit**.

While the initial implementation targeted U.S tickers (such as AAPL), the system was later extended to handle Indian tickers that are listed on NSE/BSE, along with localized financial news obtained via web scraping, highlighting its adaptability and practical relevance.

Key Highlights of the Project:

Multi-Modal Data Fusion: Integration of stock price history with external sentiment signals for enriched forecasting.

User-Centric Interface: A responsive and interactive Streamlit dashboard for real-time exploration and prediction.

Cross-Market Compatibility: Designed to accommodate both U.S. and Indian stock tickers and their associated news ecosystems.

Engineering Resilience: The pipeline includes solutions for handling data mismatches, missing values, timezones, and scraping inconsistencies.

CHAPTER 2 – PROBLEM DEFINITION

Investing in the stock market is inherently uncertain, as price movements are influenced by a wide range of factors including historical performance, market trends, economic indicators, global events, and public sentiment. While traditional financial models often rely heavily on numerical and statistical data (such as moving averages, price momentum, or volatility), they frequently fall short when it comes to accounting for non-quantitative factors like breaking news, unexpected announcements, or shifts in investor psychology.

This project stems from the recognition that real-world financial decision-making is increasingly influenced by both structured price data and unstructured information such as financial news. However, integrating these two types of data presents several challenges:

Key Challenges:

Asynchronized Data:

Financial news articles and headlines are published at irregular, unpredictable times, whereas stock prices follow strict daily trading calendars. This temporal mismatch makes it difficult to align sentiment data with corresponding stock price movements during model training.

Feature Relevance and Selection:

Determining which features meaningfully contribute to price forecasting is not straightforward. Overloading the model with irrelevant or redundant features (e.g., too many sentiment categories or noisy indicators) can degrade performance, while omitting useful ones may weaken predictive power.

Market Adaptability:

Tools or data pipelines developed for the U.S. stock market (e.g., AAPL, TSLA) often require significant adjustments to support the Indian market (e.g., RELIANCE.NS). Differences in ticker formats, news source accessibility, and market holidays pose practical difficulties in making such systems universally applicable.

Data Quality Issues:

Inconsistent data formats, missing values, timezone mismatches, duplicate headers, and incomplete scraped content are common when working with publicly available data. These issues, if not handled carefully, can break models or introduce inaccuracies.

Accessibility Issue:

There's an accessibility gap**,** many retail investors, especially beginners, lack the tools or expertise to interpret raw stock data and financial news efficiently. This can lead to "analysis paralysis" which is a state of overthinking that hinders decision-making, decisions based purely on guesswork or hype.

CHAPTER-3 OBJECTIVES

Primary Objective:

To build a deployable AI-based stock price prediction system that utilizes Long Short-Term Memory (LSTM) neural networks for modeling historical stock price sequences, augmented with sentiment insights extracted from real-world financial news headlines. The system is designed to support both U.S. and Indian stock markets, adapting appropriately to the data formats and sources of each.

Specific Objectives:

1. Comprehensive Data Engineering:

Create automated scripts for fetching and cleaning structured stock price data (using yfinance) and unstructured news headlines (via APIs and web scraping).

Handle ticker format differences and data access challenges between U.S. stocks (e.g., AAPL) and Indian stocks (e.g., RELIANCE.NS).

Implement CSV generation and organization workflows for storing cleaned and merged datasets.

2. Modeling and Feature Extraction:

Train a baseline LSTM model using only historical price data to establish a foundational prediction benchmark.

Extend the model to accept additional features, such as sentiment scores, and compare performance between baseline and enhanced versions.

Compute sentiment scores using a pre-trained transformer-based NLP model (cardiffnlp/twitter-roberta-base-sentiment via HuggingFace).

Apply normalization techniques like MinMaxScaler to scale input features appropriately.

3. Data Alignment and Merging:

Implement date-aware merging using pandas.merge_asof with backward tolerance to associate each trading day with the nearest available sentiment score from prior headlines.

Address the asynchronous nature of news and trading data through logic that ensures alignment across time zones and non-trading days.

---

4. Deployment and Usability:

Build a lightweight, interactive Streamlit dashboard that allows users to:

Select a stock ticker

Visualize actual vs. predicted closing prices

View predictions generated from either the price-only or price+sentiment model

Ensure that the app allows for smooth addition of new tickers, markets, or models in the future.

5. Error Handling and Robustness:

Identify and address practical challenges such as:

Missing data rows

Timezone misalignments

News scraping errors

Inconsistent CSV headers

Integrate fallback mechanisms (e.g., using price-only model if sentiment-enhanced model is unavailable).

6. Evaluation and Analysis:

Evaluate model performance using loss metrics such as:

Root Mean Squared Error (RMSE)

Mean Squared Error (MSE)

Visualize prediction results through trend plots to assess how closely the model tracks real stock movements.

Analyze how the inclusion of sentiment features impacts prediction accuracy particularly during volatile news events.

Document technical obstacles encountered, how they were resolved, and recommendations for future work, such as model scaling or inclusion of additional sentiment sources.

CHAPTER-4 LITERATURE REVIEW

Stock price forecasting lies at the intersection of econometrics, machine learning, and natural language processing (NLP). This review synthesizes the foundational methods and recent advancements that guided the design and implementation of this project, which focuses on LSTM-based price prediction enhanced with sentiment analysis.

Traditional Time-Series Models:

Earlier approaches to financial forecasting employed statistical techniques such as:

ARIMA (AutoRegressive Integrated Moving Average)

GARCH (Generalized Autoregressive Conditional Heteroskedasticity)

Linear Regression

These models work well with stationary, low-noise time series, but they often fail in financial contexts due to the following reasons:

Non-linearity

Sudden market shifts

External sentiment drivers

While these were reviewed for background understanding, they were not implemented in the current project.

Deep Learning and LSTM Networks:

The LSTM (Long Short-Term Memory) model, a type of Recurrent Neural Network (RNN), is designed to capture long-term dependencies in sequential data. This is particularly important in stock price forecasting, where patterns span across days or weeks.

LSTMs solve the **vanishing gradient problem** that affects standard RNNs (Hochreiter & Schmidhuber, 1997).

Studies like Fischer & Krauss (2018) demonstrate their **superiority over ARIMA and shallow models** in volatile markets.

In this project, an LSTM model forms the core of the predictive engine, trained on **sliding 60-day windows** of normalized stock price data.

Sentiment Analysis in Financial Forecasting:

Research shows that market sentiment, especially from financial news headlines, can significantly impact short-term price movements.

Bollene al. (2011) demonstrated that Twitter mood could predict Dow Jones Industrial Average trends.

Sentiment becomes particularly impactful during news-heavy periods like earnings announcements or economic policy shifts.

This project extracted sentiment from headlines using transformer-based models (RoBERTa), scoring polarity to incorporate as a numeric feature in LSTM predictions.

Transformers and RoBERTa for NLP:

The Transformer architecture (Vaswani et al., 2017) revolutionized NLP by enabling models to understand contextual relationships in text.

RoBERTa, a robust variant of BERT, has shown state-of-the-art performance in sentiment classification tasks.

HuggingFace's cardiffnlp/twitter-roberta-base-sentiment model was used in this project to quantify sentiment in scraped headlines.

Data Fusion and Alignment Techniques:

One of the major technical challenges is merging asynchronously arriving news sentiment with synchronous price data.

The literature recommends using pandas.merge_asof() with a specified time tolerance to assign the latest relevant sentiment to a stock's trading day.

This was successfully implemented in the project with a 7-day backward window, allowing each trading day to inherit the most recent sentiment score available.

Such alignment is essential for training multi-feature LSTM models where both price and sentiment influence the output.

Dashboards and User Interpretability

Recent tools like Streamlit and Dash have lowered the barrier to interactive data visualization, enabling real-time model deployment without heavy frontend development.

This project used Streamlit to build a user-facing dashboard where users can do the following:

Choose a stock

View Plots

Explore the impact of sentiment

CHAPTER-5 PROJECT DESCRIPTION

Tech Stack

The project leveraged a wide array of technologies, libraries, and frameworks for data collection, modeling, sentiment analysis, and deployment:

Data Collection & Preprocessing:

yfinance : For historical stock price data (U.S. and Indian equities).

Pandas, Numpy : For data manipulation and preprocessing.

Scrapy : For custom web scraping from financial news portals (Indian news sources).

Machine Learning & Deep Learning:

TensorFlow / Keras : For building and training LSTM models.

scikit-learn : For preprocessing (e.g., MinMaxScaler) and evaluation metrics (e.g., RMSE).

Natural Language Processing (NLP):

HuggingFace Transformers: Used the cardiffnlp/twitter-roberta-base-sentiment model for news sentiment scoring.

Visual Representation:

matplotlib, seaborn : For EDA plots, trend lines, and heatmaps.

Fig1 : EDA O/P US Stocks closing prices

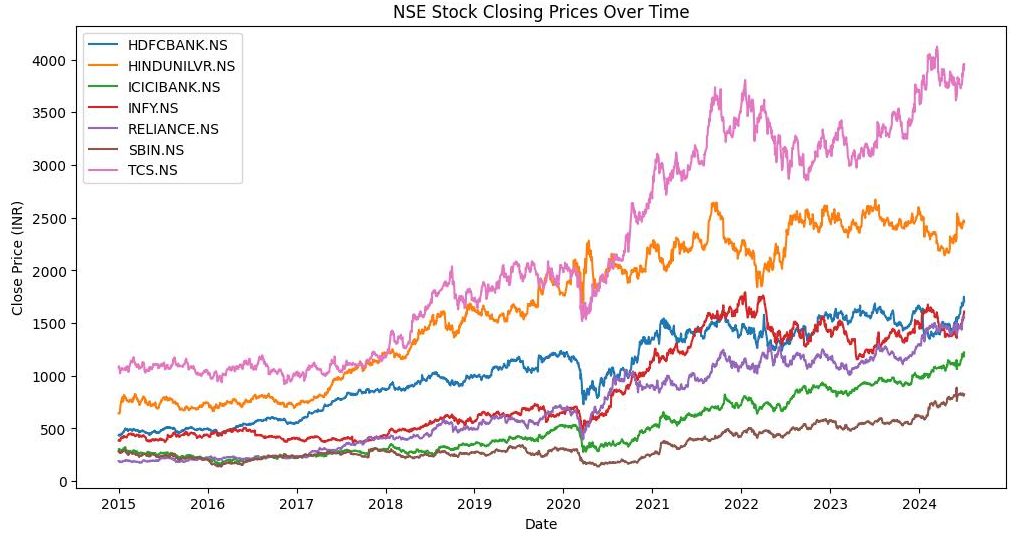

Fig2 : EDA O/P Indian Stocks closing prices

Dashboard & Deployment:

Streamlit: To create a lightweight, interactive web-based interface for real-time predictions and visualizations

Development & Workflow:

Jupyter Notebook : For experimentation and prototyping.

Modular .py scripts : For the finalized training pipeline.

Git : For version control and project tracking.

Architecture & Components:

Data Acquisition & Exploratory Data Analysis (EDA):

Employed the yfinance Python package to collect historical stock data for both U.S. equities and Indian tickers, adapting symbol formats (.NS for NSE and .BO for BSE).

Created visualizations to identify price trends and inter-stock correlations using line plots, moving averages, and heatmaps.

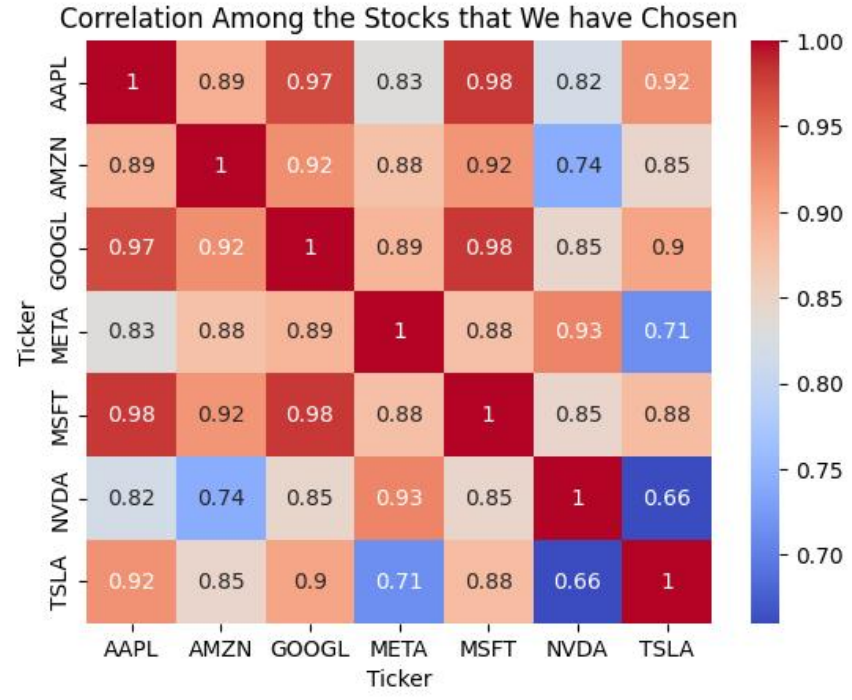

Fig3: EDA O/P Stock Correlation of US based Tickers

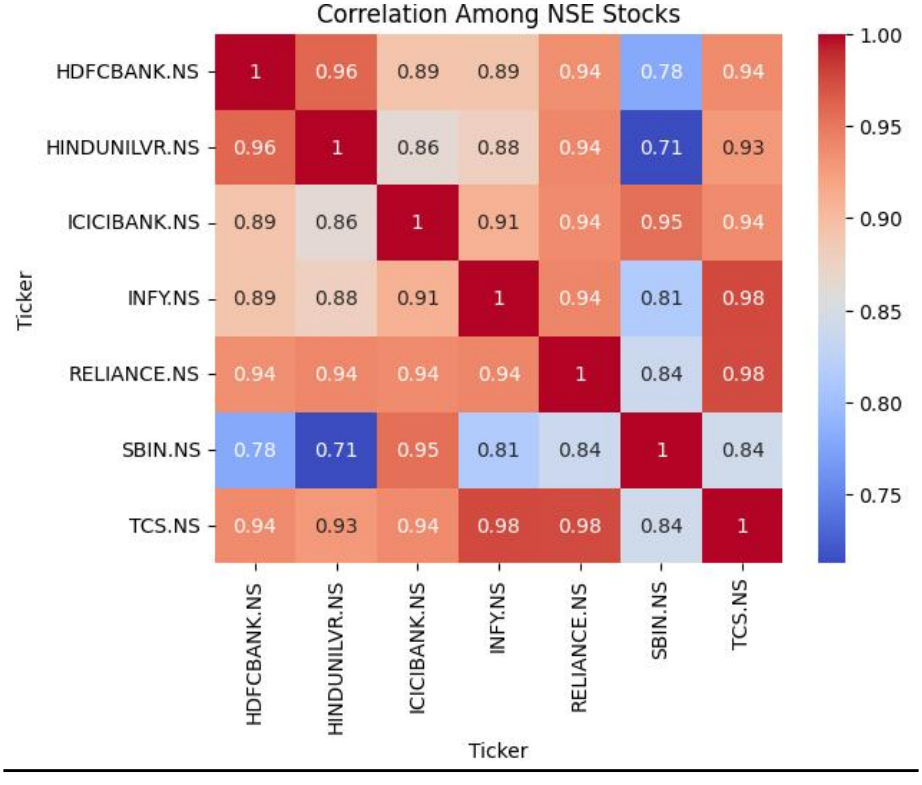

Fig4: EDA O/P Correlation of Indian Tickers

Established a structured file management system, organizing data outputs (CSVs), trained models, and merged datasets for different tickers.

LSTM-Based Modeling:

Constructed univariate LSTM models using Keras, tuning hyperparameters such as:

Number of LSTM units

Dropout rate (to prevent overfitting)

Batch size

Sequence length (60-day rolling window as input, 61st day as output)

Applied MinMaxScaler to normalize the input features before training.

Only the target values were inverse-transformed after prediction to maintain accuracy in the final output visualizations.

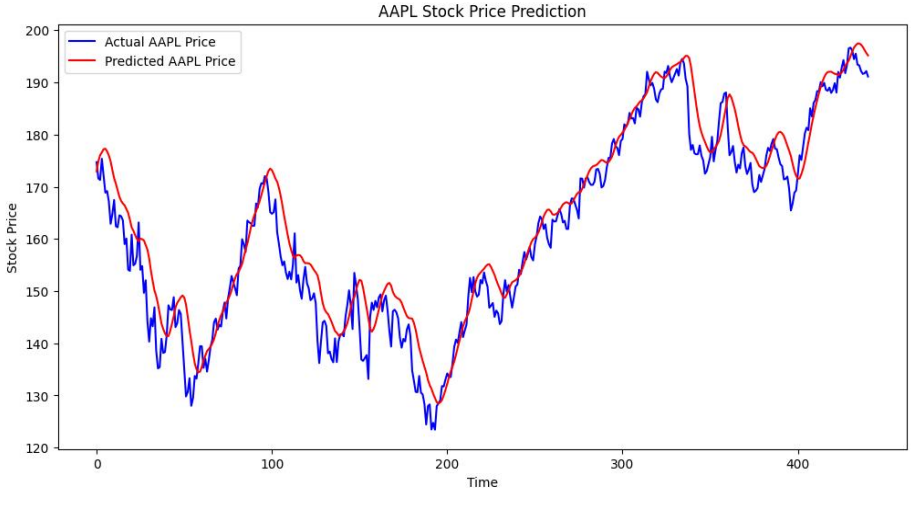

Fig5: LSTM Prediction of US based Ticker

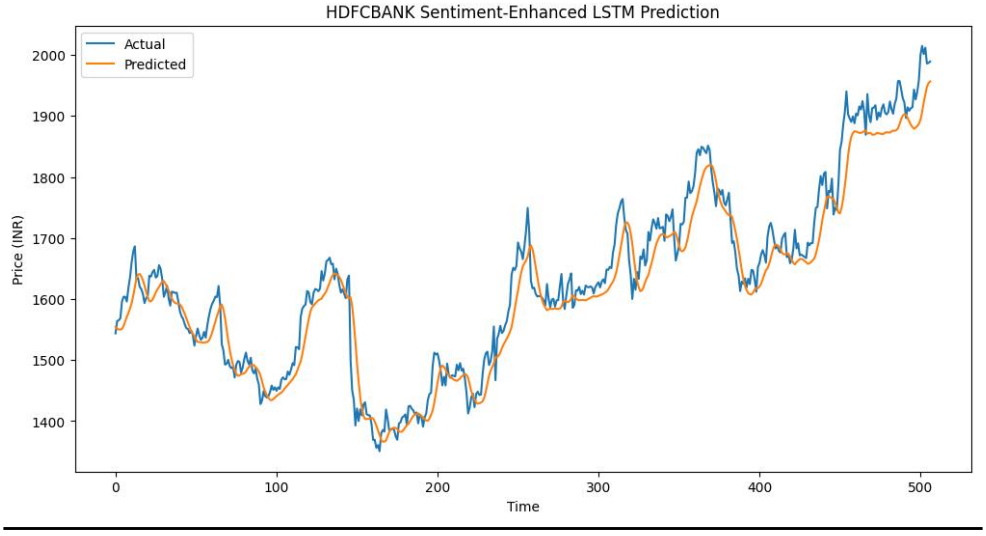

Fig6 : LSTM Prediction of Indian based Ticker

Sentiment Analysis (NLP):

Initially integrated NewsAPI for fetching financial headlines for U.S. stocks.

Due to limitations in coverage and rate-limiting, switched to a Scrapy-based custom crawler to extract news headlines from Indian financial portals, notably The Economic Times.

Utilized the HuggingFace cardiffnlp/twitter-roberta-base-sentiment model to score each headline, converting unstructured text into numeric sentiment scores.

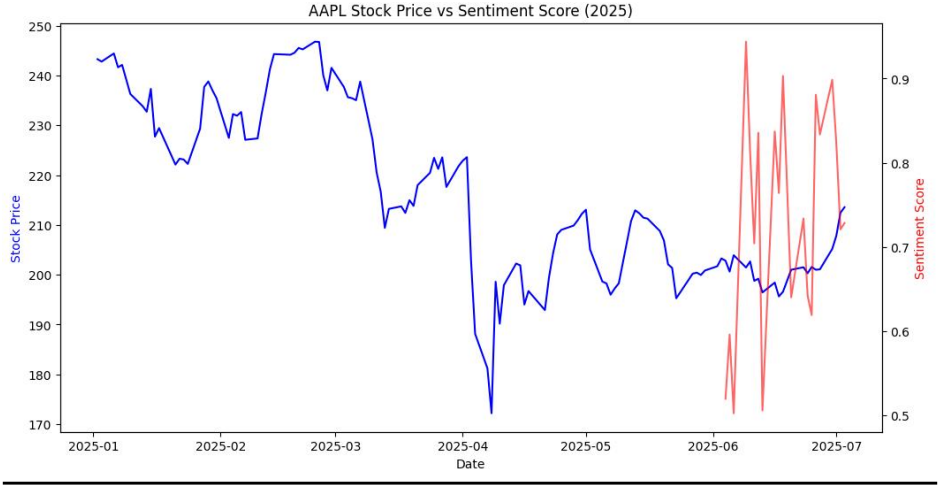

Fig7: Stock vs Sentiment Score

Data Alignment & Merging:

Implemented pandas.merge_asof() to align daily stock prices with the nearest past sentiment score, using a 7-day tolerance window.

This addressed the challenge of asynchronous data streams, such as:

Market holidays

Irregular news publishing

Non-overlapping time zones

Modeling Pipeline Automation:

Prototyped training and evaluation pipelines in Jupyter notebooks, then refactored into modular Python scripts.

Designed the codebase to separately handle baseline (price-only) and sentiment-enhanced models.

Added logic in the backend to:

Automatically detect if a sentiment-enhanced model is available for the selected stock

Fallback to the baseline model when sentiment data or models are unavailable

Deployment and Dashboard:

Built a Streamlit-based interactive dashboard to:

Allow users to choose stocks and timeframes

View predicted vs actual price trends

Compare predictions with or without sentiment enhancement

Ensured a clean and modular UI, suitable for demonstration and basic end-user interaction.

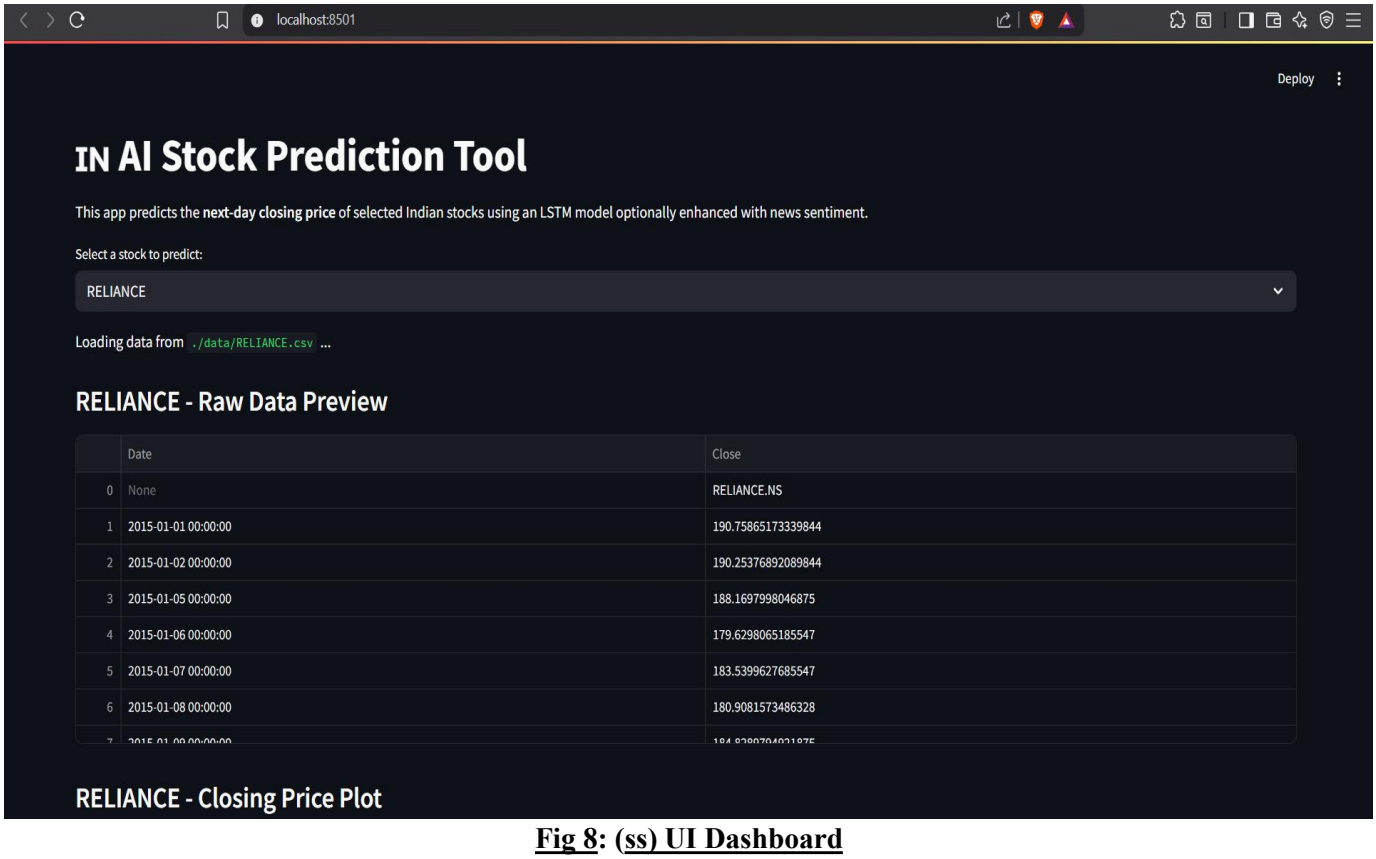

Fig 8: (ss) UI Dashboard

Fig 9: (ss)UI Dashboard

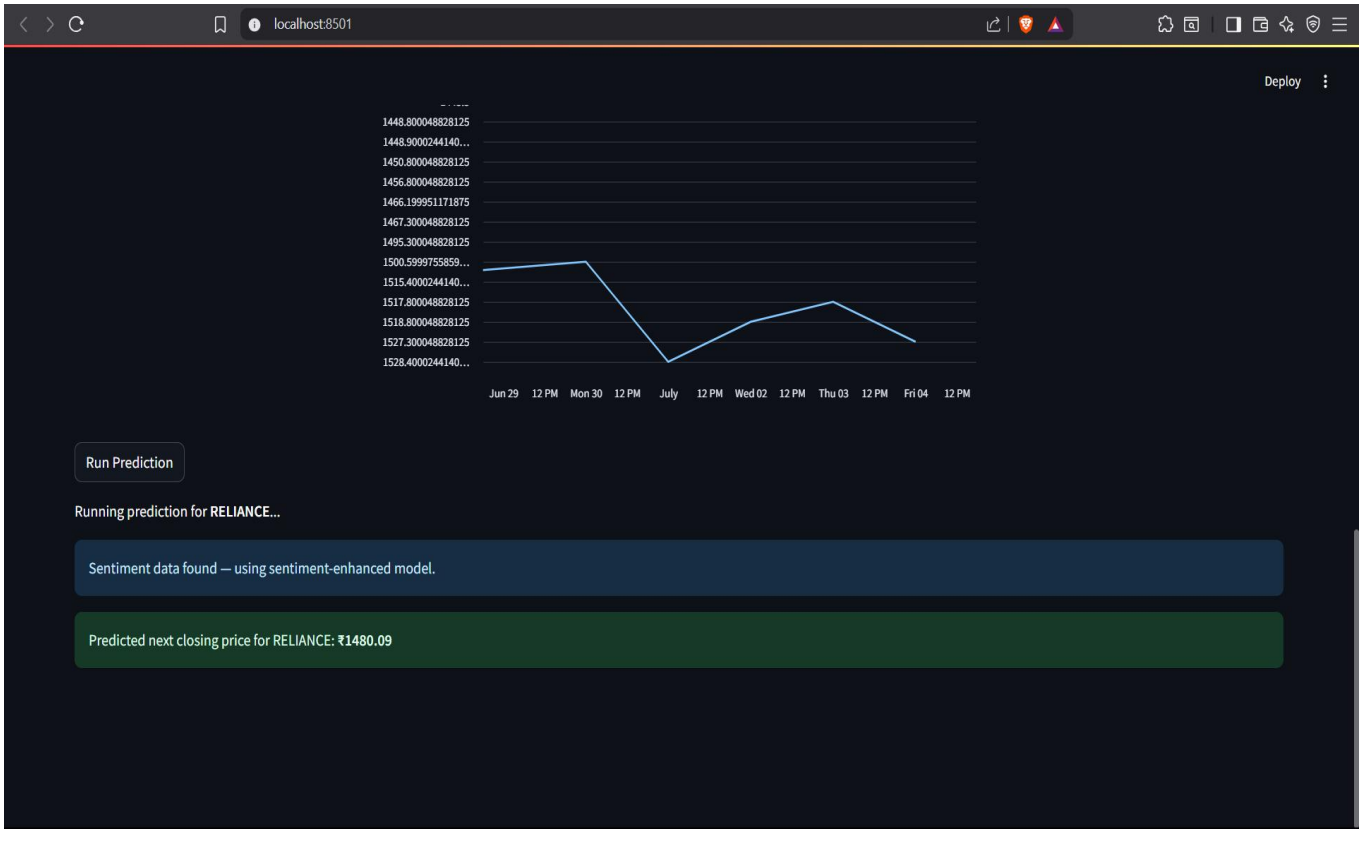

Fig 10: (ss) UI Dashboard

CHAPTER-6. METHODOLOGY

1. Project Initiation

Defined a clear project scope, success criteria, and milestones in coordination with academic mentors.

Organized the repository using a modular directory structure (e.g., data/, models/, scripts/, dashboard/).

Used Git for version control and collaborated through frequent commits.

Maintained a .gitignore to exclude large datasets, intermediate model files, and sensitive credentials.

2. Data Engineering & EDA

Acquired historical price data using yfinance for both U.S. and Indian equities (e.g., AAPL, INFY.NS).

Scraped Indian news headlines using Scrapy spiders targeting The Economic Times and similar sources.

Performed exploratory data analysis (EDA) using pandas, seaborn, and matplotlib to Identify trends, Spot missing data or anomalies, Understand inter-stock correlations via heatmaps and time-series plots.

3. Modeling & Sentiment Integration

Built and trained LSTM models using TensorFlow and Keras for univariate forecasting.

Engineered rolling window features (for eg: 60-day sequences to predict the 61st day).

Scored news sentiment using HuggingFace's RoBERTa model and converted raw sentiment into numerical features.

Integrated sentiment scores with price features to train enhanced LSTM models.

Evaluated whether sentiment data improved performance, especially during volatile or news-heavy market periods.

4. Pipeline Robustness & Data Merging

Wrote reusable Python functions to fix malformed headers, drop missing rows, normalize features using MinMaxScaler .

Merged asynchronous news and stock data using pandas.merge_asof() with a rolling tolerance window (7 days), allowing alignment between sparse news and daily price data.

5. Multi-market Adaptation

Adapted the system to support Indian stocks by adjusting ticker formats (e.g., .NS for NSE), customizing scraping logic for Indian news portals, accounting for regional holidays and market closures

Ensured that the same LSTM modeling and sentiment integration logic could be reused across markets.

6. UI Design & Deployment

Built an interactive dashboard using Streamlit, enabling dropdown selection of stocks, visualization of predicted vs. actual prices, toggle between baseline (price-only) and sentiment-augmented models

Incorporated conditional logic to fall back to baseline models when sentiment data is unavailable, handle cases where scraped headlines or model files were missing, maintained lightweight performance for deployment.

7. Documentation & Reporting

Maintained day-wise progress logs to track key learnings, bugs, and breakthroughs.

Created inline code comments and modular script documentation for easy future maintenance.

Compiled a comprehensive project report with clear descriptions of data sources, architecture, models, and challenges faced.

CHAPTER-7 RESULT AND ANALYSIS

1. Prediction Results:

Baseline LSTM models (trained solely on historical price data) demonstrated solid predictive performance, effectively capturing short-term market trends and inflection points.

The hybrid LSTM models which incorporated sentiment scores derived from financial news headlines outperformed their baseline counterparts during:

High-volatility periods

Earnings seasons

Headline-sensitive events (for eg: stock downgrades or political announcements)

For eg: In the case of AAPL (Apple Inc.), the hybrid model more accurately anticipated short-term spikes or dips that aligned with major news.

Visualization: A plotted comparison of actual vs. predicted closing prices showed strong alignment, especially when sentiment was included as a feature.

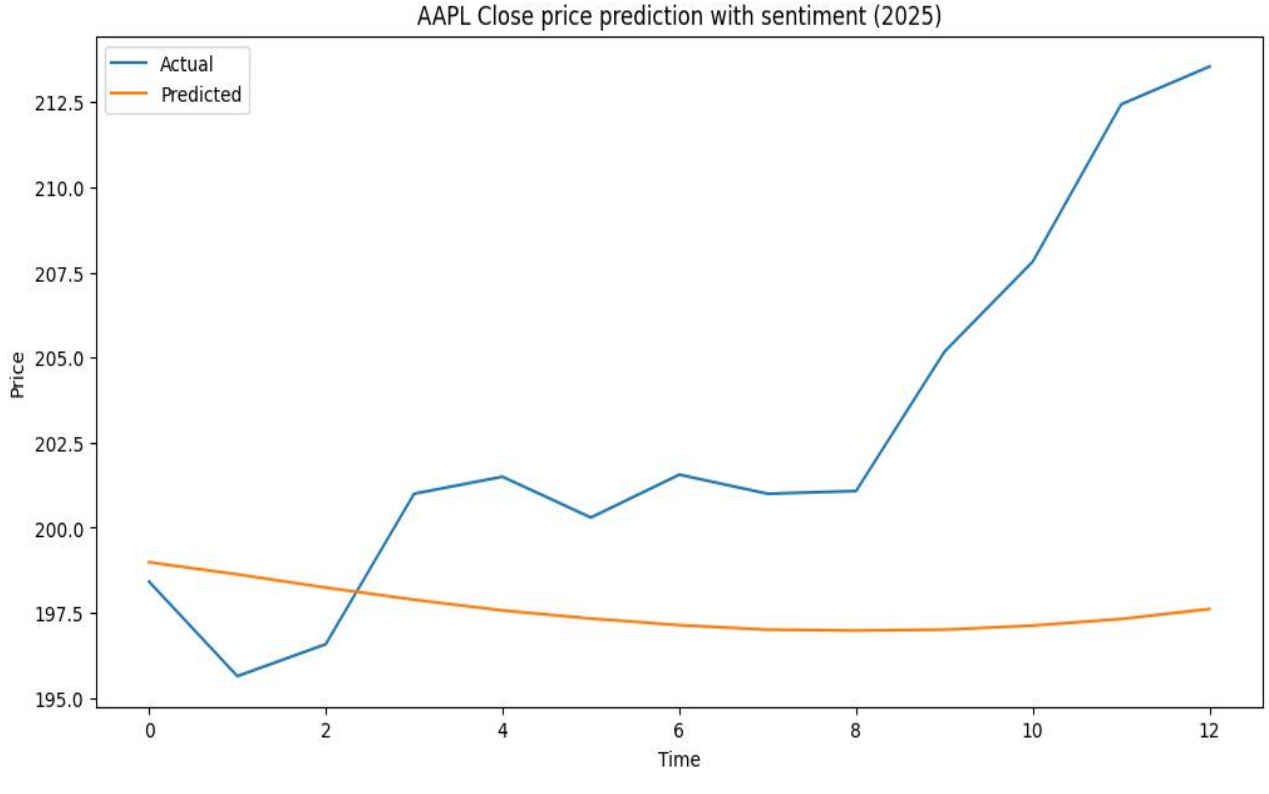

Fig 11: Actual vs Predicted with sentiment analysis

2. EDA & Visual Insights:

Correlation heatmaps between technology sector stocks (for eg: AAPL, MSFT, GOOGL) revealed strong interdependence, validating the rationale for sector-wise modeling. Likewise for NSE-BSE based stocks (for eg: RELIANCE, TCS, HDFCBANK, INFY, ICICIBANK).

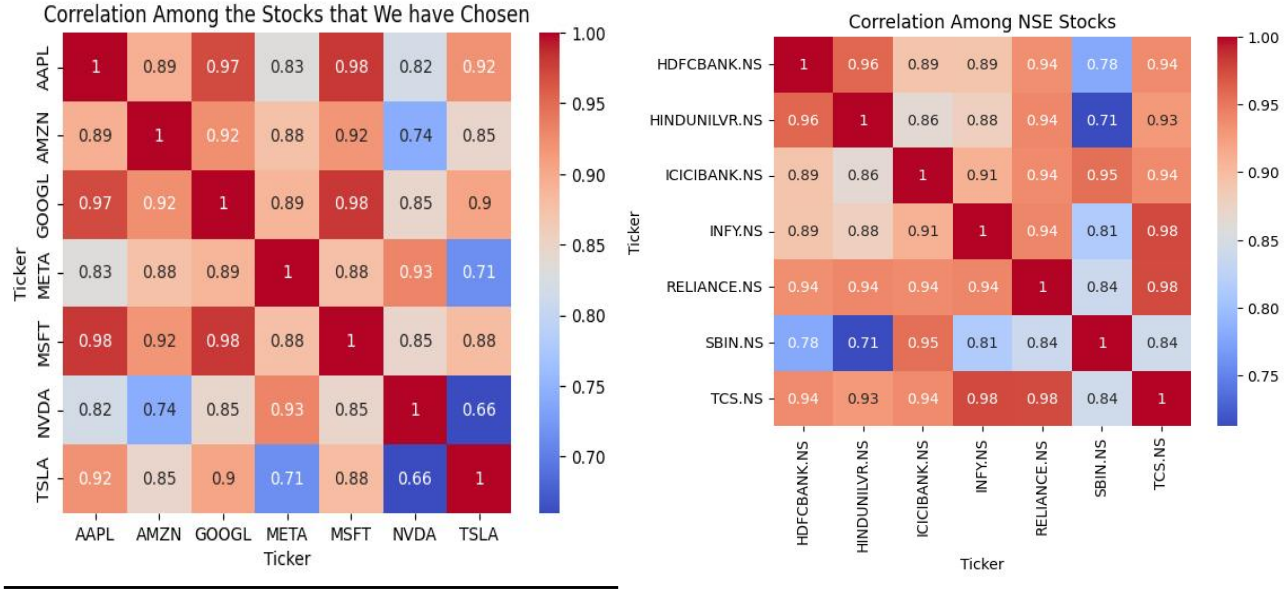

Fig 12: EDA Correlations

Time-series visualizations helped identify trends, outliers, and momentum changes.

The sentiment-overlaid plots illustrated how external news sentiment added value especially in explaining sudden price moves that were otherwise invisible to pure technical indicators.

3. Pipeline Robustness & Extensibility:

The architecture transitioned smoothly from U.S. tickers to Indian tickers (e.g., RELIANCE.NS, INFY.NS), demonstrating true cross-market compatibility.

Scrapy-based spiders for Indian news scraping operated reliably across days and sources (e.g., Economic Times, MoneyControl).

Merging stock prices with news sentiment using pandas.merge_asof() (7 day window) proved essential to bridge the non-aligned nature of news publication and trading sessions.

Frequent anomalies like missing headlines, yfinance connection drops, malformed CSV headers were handled through custom pre-processing scripts and exception logic.

4. Performance Metrics:

Dashboard performance:

Maintained sub-second response times for inference and visualization.

Model evaluation:

RMSE (Root Mean Square Error) metrics were used to benchmark accuracy.

Sentiment-augmented models generally had lower RMSE, particularly when significant news coverage was present.

5. Key Challenges & Solutions:

Challenge	Solution
Limited NewsAPI history	Shifted to Scrapy spiders for broader news coverage
Misaligned date and time formats	Applied time-zone normalization and merge_asof() logic
Header issues in scraped/CSV files	Added checks to automatically detect and correct malformed headers
Missing data on weekends/holidays	Tolerance-based backward merging handled such cases gracefully
Sentiment pipeline integration (RoBERTa)	Wrapped into reusable scoring functions with error catching
yfinance API drops	Added retry logic and saved local backups of critical stock data

6. User Experience & Dashboard Usability:

Built using Streamlit, the dashboard offered dropdown menus to select tickers, Time window selection for predictions, Toggle between price-only and hybrid LSTM models, Visualizations of input data and prediction outputs.

Users (even non-technical ones) could understand model performance visually, Explore the influence of sentiment in real-time, Use the tool with minimal computational resources.

Below are some screenshots of the UI:

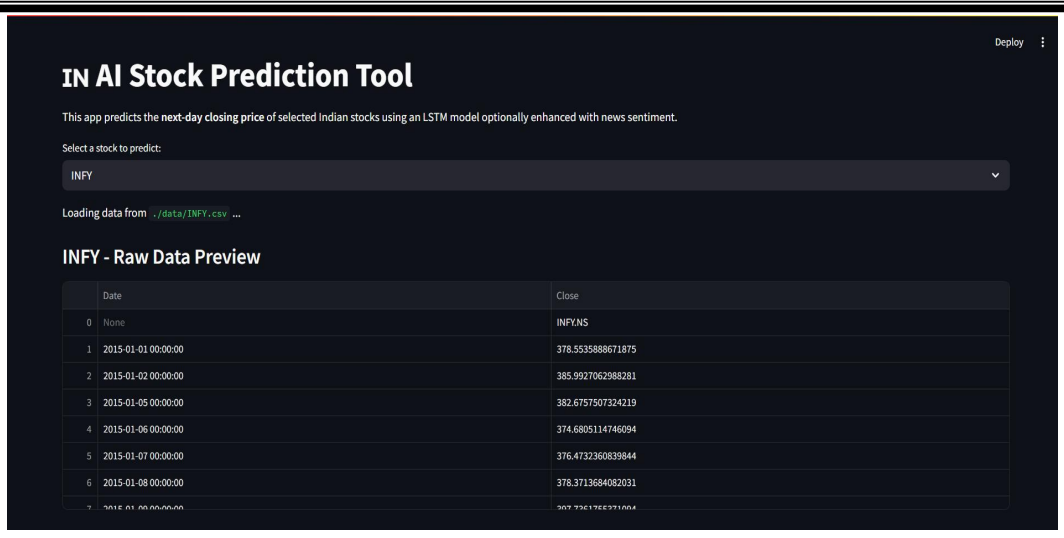

Fig 13: Screenshot

Fig 14: Screenshot

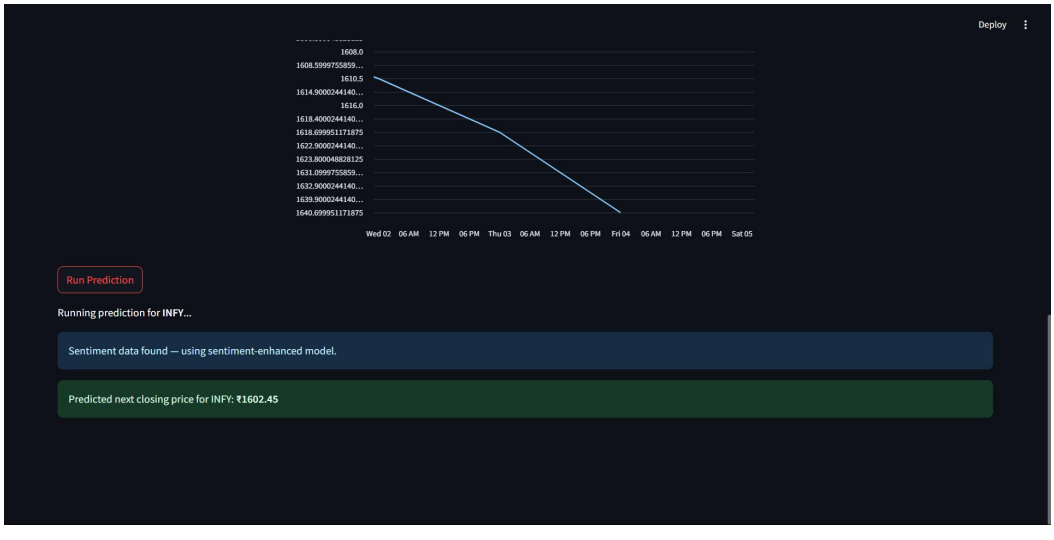

Fig 15: Screenshot

CHAPTER-8 CONCLUSION:

The internship culminated in the successful development of a functional, real-world AI platform that integrates deep learning (LSTM), natural language processing (NLP), and robust data engineering to perform next-day stock price prediction enhanced by financial news sentiment. This project represented a complete, end-to-end pipeline—from raw data collection and preprocessing, to model training and real-time deployment—emphasizing practicality, scalability, and user accessibility.

Through this journey, I significantly enhanced my hands-on skills in python programming and modular scripting, Data engineering and preprocessing for time-series and textual data, Deep learning model design (LSTM) and training workflows, Transformer-based NLP for sentiment extraction (via RoBERTa), Deployment using Streamlit to build a clean, interactive interface.

One of the most insightful outcomes of this project was observing how the inclusion of sentiment data measurably improved predictive performance, especially during high-impact market periods. The transition from U.S.-based tickers to Indian equities added a layer of complexity that demanded creative problem-solving, particularly in the areas of data scraping, alignment, and market-specific adaptation.

Key Learnings:

LSTM Effectiveness: LSTM networks are highly effective at learning and predicting stock price sequences due to their ability to remember temporal dependencies.

Value of Sentiment: When properly scored and time-aligned, sentiment data provides meaningful enhancements to forecasting models, helping capture market behavior driven by news events.

Engineering Over Modeling: A large portion of project success depended not on model complexity, but on careful data engineering, error handling, and architecture design.

Modularity and Documentation: Writing reusable, clearly documented code made the system easier to scale, debug, and adapt to new stock tickers or features.

Future Directions:

While this version of the project ends with LSTM-based modeling, it lays a strong foundation for future work:

1.Multi-Timeframe Forecasting: Extend predictions beyond next-day closing prices to hourly, daily, or weekly horizons.

2.Sentiment Diversity: Integrate other sentiment sources such as analyst ratings, Twitter sentiment (using BERTweet or VADER), and Reddit forums.

3.Model Expansion:

Experiment with ensemble techniques that combine LSTM with classical models (e.g., ARIMA) or more complex models like Graph Neural Networks (GNNs).

Investigate attention-based models like Transformers for financial time-series.

4.Automation & Personalization:

Fully automate data refresh, training, and deployment.

Customize predictions based on user preferences or portfolio holdings.

REFERENCES

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. The theoretical basis for the LSTM model.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654–669. The Book demonstrates LSTM effectiveness in stock forecasting.

yfinance Developers. (2023). yfinance: Yahoo! Finance market data downloader. GitHub Repository. Retrieved from https://github.com/ranaroussi/yfinance Used for acquiring historical stock price data for both US and Indian equities.

Hugging Face. (2023). Transformers: *State-of-the-art Natural Language Processing for PyTorch and TensorFlow 2.0*. Retrieved from https://huggingface.co/ The source of the RoBERTa-based sentiment classification model.

Barbieri, F., Anke, L. E., Camacho-Collados, J., & Neves, L. (2020). TweetEval: Unified benchmark and comparative evaluation for tweet classification. Proceedings of the Findings of EMNLP 2020. Basis for using cardiffnlp/twitter-roberta-base-sentiment model for sentiment scoring of the headlines.

Pandas Development Team. (2023). pandas: Python data analysis library. Retrieved from https://pandas.pydata.org/ The library used for data preprocessing, cleaning, and merging sentiment with stock data.

Python Software Foundation. (2023). Python: The programming language. Retrieved from https://www.python.org/ Primary programming language used throughout the entire project.

TensorFlow/Keras Developers. (2023). Keras API documentation. Retrieved from https://keras.io/ Used to implement, train, and save LSTM models.

Streamlit Inc. (2023). Streamlit: The fastest way to build data apps. Retrieved from https://streamlit.io/ Framework used to build the interactive user dashboard for stock prediction.

Scrapy Developers. (2023). Scrapy: A fast high-level web crawling and web scraping framework for Python. Retrieved from https://scrapy.org/ Used for scraping financial news data from The Economic Times and other Indian news sources when API coverage was insufficient.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30, 5998–6008. Introduced the Transformer architecture, foundational to the RoBERTa sentiment model.

Deng, Y., & Bloomberg, J. (2020). Temporal alignment of market news and stock returns. Journal of Financial Data Science, 2(4), 85–99. Supportive literature for approach in aligning asynchronous news and stock prices using merge_asof.

Matplotlib Developers. (2023). Matplotlib: Visualization with Python. Retrieved from https://matplotlib.org/ Used for plotting correlation heatmaps, line graphs, and trend visualizations in EDA.

scikit-learn Developers. (2023). scikit-learn: Machine learning in Python. Retrieved from https://scikit-learn.org/ Used for metrics such as RMSE and data scaling via MinMaxScaler.

NumPy Developers. (2023). NumPy: Fundamental package for scientific computing in Python. Retrieved from https://numpy.org/ Essential for handling arrays and numerical computations used across modeling and data processing.

GitHub Repository:

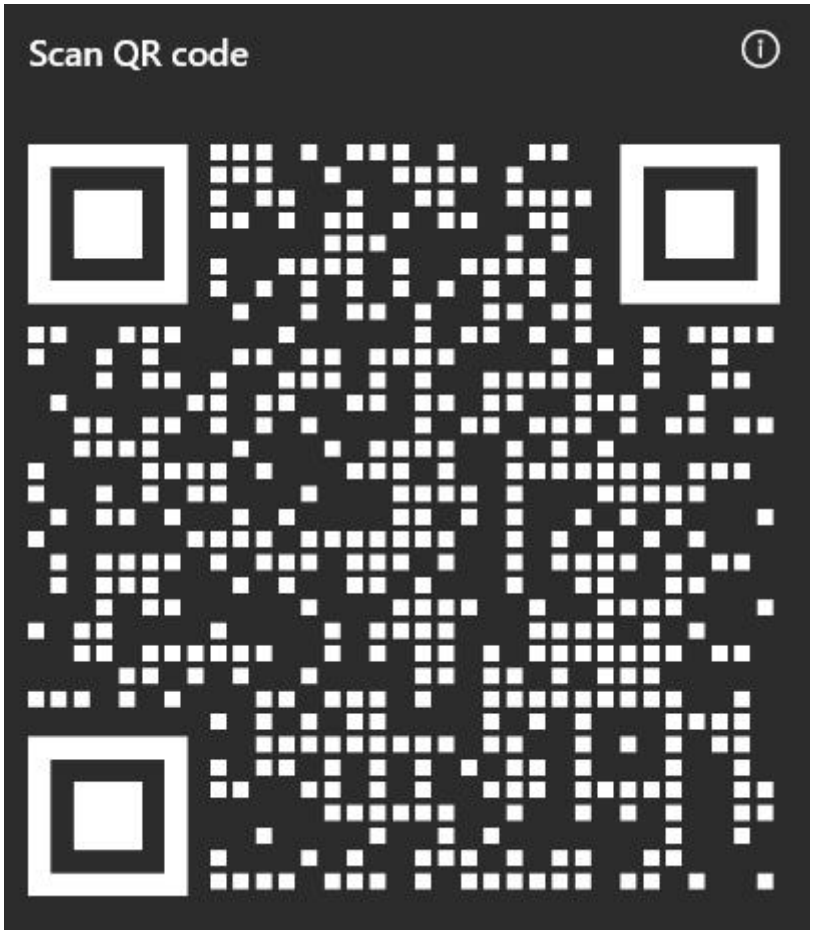

Fig 16:QR code

Link : https://github.com/Adi7coder/AI-Stock-prediction-tool/tree/master/

Instructions:

To run the app do the following in the terminal:
first run: **.\venv\Scripts\activate**
then run : **streamlit run streamlit_app/app.py**
The app will automatically open and start running in the browser.
If not so, then click manually on the urls generated something like this:
Local URL: http://localhost:8501
Network URL: http://192.168.42.238:8501