Stock price prediction using LSTM

A Project Report

Submitted by:

Puru Garg – 211B232

Under the guidance of:

Dr. Gaurav Saxena

MAY 2025

Submitted in partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science & Engineering

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,

AB ROAD, RAGHOGARH, DT. GUNA - 473226, M.P., INDIA

DECLARATION

We hereby declare that the work reported in the B. Tech. project entitled as **"Stock price prediction using LSTM"**, in partial fulfillment for the award of degree of, Bachelor of Technology submitted at Jaypee University of Engineering and Technology, Guna, as per best of our knowledge and belief there is no infringement of intellectual property right and copyright. In case of any violation, we will solely be responsible.

Puru Garg – 211B232

Department of Computer Science and Engineering,

Jaypee University of Engineering and Technology,

Guna, M.P., India

Date:

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY

Grade 'A+' Accredited with by NAAC & Approved U/S 2(f) of the UGC Act, 1956

A.B. Road, Raghogarh, Dist: Guna (M.P.) India, Pin-473226

Phone: 07544 267051, 267310-14, Fax: 07544 267011

Website: www.juet.ac.in

CERTIFICATE

This is to certify that the work titled **"Stock price prediction using LSTM"**, submitted by **", Puru Garg (211B232)"**in partial fulfillment for the award of degree of Bachelor of Technology (Computer science Engineering) of Jaypee University of engineering & Technology, Guna has been carried under my supervision. As per best of my knowledge and belief there is no infringement of intellectual property rights and copyright. Also, this work has not been submitted partially or whole to any other university or Institute for the award of this or any other degree or diploma.

In case of any violation concern student will solely be responsible.

(Dr. Gaurav Saxena)

Assistant Professor (SG)

Date:

ACKNOWLEDGEMENT

Any endeavor cannot lead to success unless and until a proper platform is provided for the same. This is the reason we find ourselves very fortunate to have undergone our minor project work under the supervision of **Dr. Gaurav Saxena**. Our sincere gratitude to him, for having faith in us and thus allowing us to carry out a project on a technology completely new to us, for which we had to research and learn many new things, which will help us deal with advanced work in future.

Secondly, we would like to thank the **Department of Computer Science and Engineering** that created this opportunity.

Last but not least we want to thank our friends and family who continuously encouraged and helped us in any way possible.

Puru Garg – 211B232

Date:

EXECUTIVE SUMMARY

In an era where financial markets are driven by data, predictive analytics has become a cornerstone for informed decision-making. This project focuses on building a user-friendly Stock Price Prediction platform using a modern data science and machine learning approach. The platform is designed as a Streamlit web application that empowers users to explore and forecast stock trends effortlessly. Users can input a stock ticker symbol (e.g., TSLA for Tesla), retrieve up to 10 years of historical stock data, and receive predictions for the next month's stock prices, providing valuable insights for investors and analysts.

The project integrates robust data retrieval mechanisms to fetch historical data, ensuring accuracy and comprehensiveness. Advanced machine learning models are employed to identify trends, patterns, and dependencies in stock movements. The focus is not only on prediction but also on delivering a visually engaging experience, with interactive charts and graphs that allow users to analyze historical trends and understand prediction outputs intuitively.

This project underscores the practical application of financial forecasting tools, demonstrating how historical data, machine learning, and intuitive interfaces can drive smarter investment strategies. By making predictive analytics accessible to a broader audience, it bridges the gap between technology and finance, enabling users to make data-driven investment decisions.

List Of Figures

Table of Contents

CHAPTER-1 INTRODUCTION

1.1 Aim

The aim of this project is to develop an intuitive and reliable Stock Price Prediction platform that simplifies the process of forecasting stock trends for users. By utilizing historical data and machine learning techniques, the project provides accurate predictions for the next month's stock prices, enabling users to make well-informed financial decisions. It seeks to bridge the gap between complex financial tools and everyday users, making advanced predictive capabilities more accessible.

This platform also aims to promote financial literacy by offering insights into market trends and helping users understand the role of data in forecasting. By combining user-friendliness with robust analytics, the tool ensures that both novice investors and experienced analysts can derive maximum value from its features.

In addition to its predictive capabilities, the platform incorporates visualizations such as line charts and trend curves to help users interpret complex data with ease. These graphical elements provide an at-a-glance understanding of price movements and market behavior, enhancing the overall user experience. The use of LSTM (Long Short-Term Memory) networks - a form of recurrent neural network especially suited for time-series forecasting - allows the system to capture long-term dependencies and patterns in stock market data, leading to more accurate and meaningful predictions.

To ensure the platform's adaptability, the system is designed to accommodate various stock tickers, letting users explore and compare predictions across multiple companies or sectors. The modular design allows for future integration of additional features, such as sentiment analysis, technical indicators, or real-time news feeds, further improving the decision-making process.

By merging technological sophistication with an accessible interface, this project not only empowers users to take control of their investment choices but also fosters a deeper appreciation of data-driven finance.

1.2 Key Facts

1. Historical Data Retrieval:
   - The platform retrieves up to 10 years of historical stock data from reliable financial databases, ensuring a comprehensive dataset for analysis.
   - It uses APIs or public repositories for seamless data integration and accuracy.

2. Machine Learning Models:
   - The platform employs advanced machine learning techniques to identify trends and predict stock prices.
   - Models such as LSTM (Long Short-Term Memory) or regression-based approaches are used to handle the time-series nature of stock data.

3. User-Friendly Interface:
   - Built on Streamlit, the application provides an intuitive and interactive user interface.

   1. Users can input stock tickers, view visualizations, and access prediction results without requiring technical expertise.

4. Data Visualization:
   - The platform offers dynamic charts and graphs to showcase historical trends and predicted values, making data interpretation straightforward.
   - Tools like Matplotlib or Plotly are integrated for a polished visualization experience.

5. Scalability:
   - The project is designed to be scalable, with future capabilities like real-time stock updates, external economic data integration, and portfolio management features.

1.3 Objectives Of The Project

1. Simplify Stock Price Prediction:

- Automate the analysis of stock market trends, reducing the manual effort and complexity involved in traditional forecasting methods.
- Provide users with reliable predictions for the next month's stock prices, helping them make informed decisions.

2. Promote Accessibility:

- Make predictive tools accessible to a broader audience, including individual investors, small businesses, and students.
- Design an interface that ensures even users with limited technical or financial expertise can benefit from the platform.

3. Enhance Financial Literacy:

- Help users understand how historical trends influence future stock movements through interactive visualizations and insights.
- Encourage data-driven decision-making by providing clear and actionable predictions.

4. Support Educational and Research Goals:

- Provide a practical application of machine learning in finance for students and researchers.
- Serve as a learning tool for exploring concepts like time-series analysis, predictive modelling, and data visualization.

5. Future Expansion and Adaptability:

- Lay the groundwork for integrating additional features like real-time stock updates, sentiment analysis, and economic indicator tracking.
- Ensure the platform remains adaptable to evolving user needs and technological advancements.

1.4 Scope Of The Project

The Stock Price Prediction project aims to explore, analyze, and forecast stock price movements using historical market data. The scope of this project is broad, encompassing various aspects of financial analysis, data science, and machine learning, while offering practical applications in trading, investment, and risk management. It delves into the intricacies of time-series forecasting by leveraging advanced deep learning models, particularly Long Short- Term Memory (LSTM) networks, known for their ability to capture temporal dependencies in sequential data. In addition to building predictive models, the project also focuses on data preprocessing techniques such as normalization, handling missing values, and creating appropriate input sequences for model training. It further includes the use of visualization tools to aid in understanding market trends and communicating insights effectively. The system is designed to support multiple stock tickers and can be adapted to different markets and time frames, enhancing its utility for various user needs. Moreover, the project aims to democratize access to predictive analytics by integrating the model into a user-friendly platform, allowing individuals with limited technical background to benefit from its forecasts. This dual emphasis on technical rigor and accessibility ensures that the project is not only a technological solution but also an educational and strategic tool for better financial planning and decision-making. The key areas within the project's scope include:

1. User-Friendly Interface

The project features a Streamlit-based application where users can input stock tickers (e.g., TSLA for Tesla) and fetch historical stock data. By making the app interactive and intuitive, it bridges the gap between complex financial algorithms and end-users who may not have a technical background. The interface displays visualizations such as historical price charts, moving averages, and predicted vs. actual stock prices, allowing users to interpret trends with ease.

2. Data Collection and Preprocessing

The project retrieves data for the past 10 years using APIs or financial libraries like Yahoo Finance or Alpha Vantage. The data includes critical features like opening price, closing price, volume, and adjusted closing price. Preprocessing steps such as handling missing values, smoothing noise, and feature scaling ensure data is cleaned and ready for accurate modeling.

3. Predictive Analytics and Machine Learning

The project leverages machine learning models like LSTM (Long Short-Term Memory), ARIMA (Autoregressive Integrated Moving Average), or other time-series forecasting techniques. These models are trained to recognize patterns in historical data and provide robust predictions for the stock price trend in the upcoming month. LSTM, in particular, excels at capturing long-term dependencies and sequential relationships in stock price data, making it ideal for financial time-series forecasting. The system undergoes rigorous training and validation using past stock data, ensuring the models generalize well and minimize prediction errors. Key evaluation metrics such as RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and MAPE (Mean Absolute Percentage Error) are used to assess model performance. Additionally, the model outputs are visualized to compare predicted values with actual market performance, aiding in transparency and trust in the predictions.

4. Financial Analysis and Insights

In addition to predictions, the project also offers analytical features, such as visualization of historical trends, moving averages, and volatility patterns. These insights help users understand the behavior of the stock and make informed decisions. The platform includes interactive charts that display stock price fluctuations over various time frames, enabling users to identify seasonal trends, price spikes, and dips. Key technical indicators like the Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), and Bollinger Bands are integrated into the analysis, providing users with a comprehensive understanding of market momentum, trends, and potential entry/exit points. These indicators are visualized alongside the stock's predicted price to offer a complete picture of the stock's potential movement. The app also highlights market volatility, showing periods of high and low uncertainty, which can help users assess risk and determine optimal trading strategies. By providing both predictive insights and historical analysis, the platform serves as an all-in-one tool for users to gauge stock performance and make data-driven investment decisions.

5. Practical Applications

This project has significant applications in:

- Investment Planning: Helping investors determine the best times to buy or sell stocks.
- Risk Management: Assisting portfolio managers in assessing the risk associated with specific stocks.
- Algorithmic Trading: Providing a foundation for automated trading systems to execute trades based on predictions.

6. Scalability and Extensibility

While the current focus is on a one-month prediction, the project is designed to be scalable. Future iterations could include additional features like:

- Real-time data fetching for intraday predictions.
- Sentiment analysis of news or social media to incorporate qualitative factors into predictions.
- Multi-stock analysis to study correlations and trends across different market segments.

7. Educational and Research Value

For aspiring data scientists and financial analysts, the project provides a valuable case study on the application of machine learning in finance. It offers insights into working with time-series data, feature engineering, and building prediction pipelines. By examining how LSTM networks are used for stock price forecasting, learners can deepen their understanding of deep learning techniques and their relevance in real-world financial applications.

8. Limitations and Challenges

The project acknowledges inherent limitations, such as the unpredictable nature of the stock market, external economic factors, and the risk of overfitting models to historical data. By addressing these challenges, the project opens avenues for continuous improvement and innovation.

1.5 Problem Formulation

The stock market is a highly dynamic environment, influenced by a wide range of factors such as global events, economic policies, market sentiment, and individual company performance. Predicting stock prices is a complex and challenging task due to the market's inherent volatility and unpredictability. Despite these challenges, accurate stock price predictions can significantly benefit investors and traders in making informed financial decisions. The market exhibits non-linear and stochastic behavior, which complicates traditional forecasting techniques that assume consistent trends. Additionally, external events such as political instability, natural disasters, and sudden economic shifts can drastically alter stock prices in unpredictable ways. As a result, a robust model needs to account for both historical patterns and unforeseen events, striking a balance between overfitting past data and underfitting new data. Given these challenges, the goal of this project is to develop a machine learning-based predictive model that uses historical data to forecast stock price movements.

The model aims to capture the inherent trends and cycles in stock prices, while incorporating methods such as LSTM networks to learn temporal patterns and provide reliable predictions. However, the unpredictability of the market means that any model must also include uncertainty and provide users with actionable insights based on available data, rather than definitive predictions.

This project aims to address the following problems:

1. Complexity in Analyzing Stock Data

Analyzing stock market trends requires advanced knowledge of financial concepts and technical tools. Many retail investors lack access to such expertise, which limits their ability to effectively interpret market data and identify profitable opportunities. Stock market analysis involves a variety of techniques such as technical analysis, fundamental analysis, and quantitative modeling, each of which requires a deep understanding of market behavior, indicators, and statistical methods. For instance, technical analysis relies heavily on the interpretation of charts, patterns, and indicators like moving averages, RSI, and MACD, while fundamental analysis involves examining financial statements, earnings reports, and macroeconomic factors.

2. Unpredictability of Market Movements

The stock market is driven by multiple external and internal factors, making its behavior difficult to forecast. Traditional analytical methods often fail to capture the nuanced patterns and trends present in historical data. Traditional analytical methods, such as linear regression and statistical models, often fail to capture the nuanced patterns and complex relationships present in historical data. These methods may assume linearity or fixed relationships between variables, which do not always hold true in the dynamic and non-linear nature of financial markets. As a result, these models are limited in their ability to account for sudden market shifts or emerging trends that can have a significant impact on stock prices.

3. Need for Accurate and Accessible Predictive Models

While machine learning has demonstrated its potential in stock price prediction, existing solutions are often either too technical for non-experts or lack user-friendly interfaces. There is a need for a robust and accessible tool that can leverage machine learning to provide reliable forecasts. Moreover, existing platforms may present their predictions and analysis in complex formats, which can make it difficult for users to derive actionable insights. Investors and traders need tools that not only provide accurate predictions but also present them in a way that is easy to understand and interpret, without requiring extensive technical knowledge. This gap between powerful machine learning techniques and their accessibility to the average user creates a barrier to effective decision-making in the stock market.

This project formulates these challenges into a concrete problem: creating a user-friendly application that uses historical stock data to make accurate short-term predictions. By addressing these issues, the project seeks to empower users with the tools and insights needed to navigate the complexities of the stock market effectively. The goal is to bridge the gap between sophisticated machine learning techniques and everyday users, offering a solution that simplifies the decision-making process while maintaining the power of advanced predictive models.

1.6 Project Overview

The Stock Price Prediction project focuses on building an interactive and user-friendly Streamlit application to predict future stock prices based on historical data. The application enables users to input a stock ticker symbol (e.g., TSLA for Tesla) and retrieves the past 10 years of stock data. Using this data, machine learning models are employed to forecast stock prices for the upcoming month.

The project incorporates key functionalities such as data collection, preprocessing, time-series analysis, and model-driven predictions. It also provides visualizations of historical trends and predicted prices, helping users gain insights into stock performance. By combining financial analysis with advanced machine learning techniques, this project offers a practical solution for understanding and anticipating market movements.

The user interface is designed to be clean and simple, ensuring a seamless experience for both beginner investors and seasoned professionals. The project emphasizes not just on accurate predictions but also on usability, offering easily interpretable charts and statistics that enable users to act on the information with confidence.

Through this combination of user-friendly design, real-time data, and advanced predictive models, the Stock Price Prediction project aims to equip individuals with a powerful tool for making data-driven investment decisions. By democratizing access to sophisticated forecasting methods, the project strives to empower users and enhance their understanding of stock market movements.

1.7 The Significance Of The Project

1. Empowering Investors and Traders

The project bridges the gap between complex stock market analysis and users who may lack technical expertise. By providing an intuitive tool for prediction, it empowers users to make data-driven decisions with confidence.

2. Practical Applications in Finance

Accurate stock predictions are invaluable for various financial activities, including investment planning, portfolio management, and risk assessment. This project contributes to these areas by offering reliable forecasts.

3. Integration of Technology in Finance

The project showcases how modern machine learning techniques can be applied to real-world problems, such as stock price prediction. It highlights the potential of technology to transform traditional financial analysis. This integration of machine learning into finance allows for the identification of complex patterns and trends that may be overlooked using traditional analysis methods.

4. Educational and Research Value

This project serves as a learning tool for students and professionals interested in financial modeling, data analysis, and machine learning. It demonstrates the end-to-end workflow of data-driven stock prediction, from data collection to model evaluation, providing a comprehensive understanding of how technology can be applied in the financial sector. By working through the various stages of the project, users gain practical experience in areas such as data preprocessing, time-series analysis, feature engineering, and the use of machine learning algorithms for prediction.

5. Scalability and Future Potential

While the current focus is on one-month predictions, the framework can be extended to include additional features such as real-time predictions, sentiment analysis, and multi-stock comparisons, making it a foundation for future enhancements in financial technology.

This project is not only a technical achievement but also a practical solution to a significant financial problem, demonstrating how innovative tools can simplify and enhance the decision-making process in the stock market.

As the financial industry continues to evolve, this project holds strong potential for scalability and integration with other cutting-edge technologies. Future iterations could incorporate APIs for live market data streaming, allowing the system to offer real-time updates and continuously refreshed predictions. This would significantly enhance its utility for day traders and investors who rely on up-to-the-minute information.

1.8 Who Will Be Benefited By The Project

The Stock Price Prediction project is designed to cater to a wide range of users and stakeholders who are involved in financial decision-making or have an interest in stock market analysis. By simplifying the complexities of forecasting with the help of machine learning and data visualization, the project ensures that both novice and experienced users can derive actionable insights. Whether it's for academic purposes, daily trading decisions, or long-term investment planning, the tool serves as a bridge between data science and practical finance. Its accessible design and predictive capabilities make it a valuable resource for enhancing financial literacy and decision-making in various contexts. The primary beneficiaries of this project include:

1. Retail Investors

Individuals who invest in the stock market but lack advanced financial knowledge will benefit significantly. The project provides an easy-to-use platform to analyze historical trends and make informed decisions about buying or selling stocks. With features like visualized data, predicted price trends, and simplified interfaces, retail investors can better understand market dynamics without needing expertise in technical analysis or machine learning. This lowers the entry barrier for individuals looking to manage their own investments with greater confidence and accuracy.

2. Traders

Active traders who rely on short-term market movements for profit can use the project's predictions to identify potential opportunities and minimize risks. The one-month forecast feature is especially useful for planning trades and setting entry or exit points with greater confidence. By leveraging machine learning-generated trends and historical patterns, traders can supplement their technical strategies and improve timing. Additionally, real-time data integration (in future versions) could further assist in intraday decision-making and strategy optimization.

3. Portfolio Managers

Financial professionals managing investment portfolios can leverage the predictive capabilities to assess the risk and potential returns of specific stocks, aiding in portfolio diversification and optimization. By integrating model-driven forecasts with traditional

financial metrics, they can make more informed asset allocation decisions. The application's ability to visualize historical trends and forecast future movements supports the development of balanced portfolios aligned with client goals, risk tolerance, and market conditions.

4. Financial Analysts

Analysts can utilize the project as a tool for conducting research and validating insights derived from other financial models. The visualizations and forecasts can also assist in presenting data to clients or stakeholders, supporting more persuasive and data-driven reporting. Additionally, the tool can aid analysts in monitoring market behaviour and adjusting investment strategies accordingly.

By leveraging features such as moving averages, volatility analysis, and trend prediction, analysts can enhance the accuracy of their evaluations and create more robust financial models. The interactive and customizable nature of the platform also enables analysts to tailor their research to specific sectors or timeframes, increasing the flexibility and precision of their market evaluations.

5. Technology Enthusiasts and Data Scientists

Individuals interested in the intersection of finance and technology will find value in exploring how machine learning models can be applied to real-world problems. The project serves as a case study for combining data science with financial applications. It offers a practical framework for experimenting with time-series forecasting, neural networks like LSTM, and data preprocessing techniques.

Moreover, data scientists can gain hands-on experience in handling financial datasets, building and fine-tuning predictive models, and deploying them in user-friendly interfaces using tools like Streamlit. For technology enthusiasts, the project demonstrates how AI can disrupt traditional domains, encouraging innovation and exploration in fintech solutions. The project can also act as a foundation for further development, such as integrating live data feeds, deploying via cloud platforms, or incorporating alternative data sources like social media sentiment.

6. Educational Institutions and Students

Students learning about financial markets, data science, or machine learning can benefit from this project as a practical example of applying theoretical concepts. It can be used in coursework or research projects related to finance and technology. The project offers a hands- on learning experience by walking students through real-world processes such as data collection, cleaning, feature engineering, model selection, training, evaluation, and deployment.

7. Small and Medium Enterprises (SMEs)

SMEs involved in stock market advisory or financial consulting can use the project as a tool to enhance their services, providing their clients with advanced forecasting capabilities. The application's predictive insights allow SMEs to offer data-driven recommendations, helping their clients make more informed investment decisions. This can differentiate their services in a competitive market and attract a wider range of clients, from individual investors to small businesses looking to optimize their financial portfolios.

By addressing the needs of these diverse groups, the project has the potential to make a meaningful impact across both professional and personal domains of financial decision-making. It empowers individuals, businesses, and financial professionals with valuable tools to predict, analyze, and navigate market trends more effectively. This, in turn, can lead to better-informed investment strategies, improved financial literacy, and enhanced decision-making, contributing to greater market efficiency and financial stability.

CHAPTER-2 LITERATURE & SURVEY

2.1 Existing System

In the field of stock price prediction, various methodologies have been explored, each with its own set of advantages and disadvantages. Below is a comparative analysis of ten different approaches, highlighting their pros and cons:

Methods	Pros	Cons
Linear Regression	Simple to implement, easy to interpret, requires less computational power	Assumes linear relationship, not suitable for capturing complex patterns, sensitive to outliers
ARIMA (AutoRegressive Integrated Moving Average)	Effective for short-term forecasting, good for stationary time series data	Assumes linearity, requires differencing for non-stationary data, not suitable for long-term predictions
SVM (Support Vector Machines)	Effective for high-dimensional data, good at capturing non-linear relationships	Computationally intensive, requires careful parameter tuning, sensitive to the choice of kernel
Random Forest	Handles non-linear data well, robust to overfitting, can handle large datasets	Interpretability is difficult, requires a lot of computational resources, can be slow

LSTM (Long Short-Term Memory)	Captures long-term dependencies, effective for sequential data, robust to noise	Requires a large amount of data, computationally intensive, complex to implement
GRU (Gated Recurrent Unit)	Similar benefits to LSTM but with fewer parameters, faster training	Still requires a significant amount of data, complex to implement, computationally intensive
CNN (Convolutional Neural Networks)	Effective for spatial data, good at capturing local patterns, robust to noise	Requires a large amount of data, computationally intensive, not traditionally used for time series data
XGBoost (Extreme Gradient Boosting)	High predictive performance, handles missing data well, robust to overfitting	Computationally intensive, requires careful parameter tuning, interpretability can be challenging
ANN (Artificial Neural Networks)	Can model complex relationships, flexible, adaptable to various types of data	Requires a large amount of data, prone to overfitting, computationally intensive
Hybrid Models (e.g., combining ARIMA and LSTM)	Combines strengths of different models, can capture both linear and non-linear relationships	Complex to implement, requires significant computational resources, interpretability can be challenging

2.2 Proposed System

For the proposed system in this project, Long Short-Term Memory (LSTM) networks were chosen as the primary methodology for stock price prediction. LSTM, a type of Recurrent Neural Network (RNN), is particularly effective for modeling sequential data, making it well-suited for time series forecasting tasks like predicting stock prices. Unlike traditional machine learning models, LSTM networks are capable of capturing long-term dependencies and temporal patterns in sequential data, which is crucial for accurately predicting future stock movements.

Justification for Using LSTM:

- Captures Long-Term Dependencies: LSTM networks are designed to remember information over long sequences, making them effective in capturing the long-term dependencies often present in stock price data.
- Robustness to Noise: LSTMs are more robust to noisy data compared to other traditional methods like linear regression, ensuring more stable predictions even in volatile markets.
- Effectiveness for Sequential Data: The nature of stock prices, which are influenced by a sequence of previous prices and external factors, aligns well with the capabilities of LSTM networks, which are designed to process and predict sequential data.

2.3 Feasibility Analysis

The feasibility analysis for the Stock Price Prediction project evaluates its practicality and viability across different dimensions:

1. Technical Feasibility

The technical requirements for this project are well within reach, leveraging established technologies and tools:

- Data Sources: Reliable APIs (e.g., Yahoo Finance, Alpha Vantage) are available to fetch historical stock data.
- Programming Tools: Python, along with libraries like Pandas, NumPy, Scikit-learn, and TensorFlow/Keras, offers a robust ecosystem for data processing and machine learning.

- Platform: Streamlit simplifies the creation of a user-friendly interface, making it accessible to non-technical users.
- Models: Time-series forecasting models like ARIMA and LSTM are proven for stock price prediction. Tutorials, research papers, and pre-built packages provide ample support for their implementation.

2. Economic Feasibility

The project is cost-effective for both development and deployment:

- Development Costs: Minimal, as open-source tools and libraries are used. Development time primarily involves programming and data analysis.
- Hosting: The app can be hosted on free or affordable platforms such as Streamlit Cloud, Heroku, or AWS with minimal overhead.
- API Costs: Free tiers of APIs like Yahoo Finance or Alpha Vantage are sufficient for moderate use cases, reducing initial costs.
- Return on Investment (ROI): The project has significant value for users who can make better investment decisions, potentially outweighing any implementation costs.

3. Operational Feasibility

The project is designed for ease of use and scalability:

- User-Friendly Interface: The Streamlit app ensures that users can interact with the tool without requiring technical expertise.
- Automation: Data fetching, preprocessing, and prediction are automated, reducing manual effort.
- Scalability: The framework can be extended to include additional features like real-time predictions, sentiment analysis, or portfolio recommendations.
- Performance and Efficiency: The project leverages optimized machine learning algorithms and efficient data processing techniques, ensuring fast predictions and smooth user experience even with large datasets.

4. Market Feasibility

There is a strong demand for stock prediction tools:

- Audience: Retail investors, traders, financial analysts, and students constitute a broad user base.
- Competitors: While similar tools exist, many are either too technical or lack accessibility. This project bridges the gap by offering a balance of accuracy and usability.
- Uniqueness: The combination of a user-friendly interface, machine learning-based predictions, and detailed visualizations differentiates this project from traditional tools.
- Adaptability: The system can be adapted to various financial markets and asset types beyond stocks, including commodities, forex, and cryptocurrencies, expanding its potential user base and market reach.

5. Legal and Ethical Feasibility

The project complies with legal and ethical standards:

- Data Usage: Publicly available stock data is used, ensuring compliance with data licensing and usage policies.
- Transparency: Predictions are based on machine learning models, and their assumptions are explained to users to avoid misleading claims.
- Data Privacy: The system does not collect or store any personally identifiable information (PII), ensuring that user data remains private and protected.
- Bias and Fairness: Efforts are made to mitigate any biases in data processing or model training, ensuring the predictions are fair and unbiased toward any specific group or stock.
- Informed Consent: Users are informed of the data usage policies and the nature of the predictions, ensuring they give informed consent before interacting with the system.

CHAPTER – 3 SYSTEM ANALYSIS AND DESIGN

3.1 Software Requirement Specification

- Project Name: Machine Learning-Based Stock Price Prediction Model
- Project Purpose: To provide users with insights and potential future trends in the stock market using historical data analysis and visualization.
- Target Audience: Individual investors of all experience levels who seek data-driven insights to support their investment decisions.
- Development Technologies: Python, Stooq API, JavaScript libraries for charts and graphs (matplotlib).

3.2 Non- Functional Requirements

1. Performance:
   - Website should load data and respond to user actions quickly and efficiently.
   - Optimize resource usage to avoid server overload.
   - Utilize efficient algorithms for data processing and prediction to enhance speed.
   - Implement lazy loading to reduce initial loading times.
   - Minimize the size of assets (e.g., images, scripts) for faster loading.
2. Scalability:
   - Design the architecture to handle potential increases in user traffic and data volume.
   - Consider caching mechanisms and load balancing.
   - Use load balancing to distribute traffic evenly across multiple servers.
   - Design the database for scalability with indexing and partitioning strategies.
   - Ensure the system can handle spikes in traffic, especially during high-demand times like market openings or financial news events.
3. Security:
   - Implement security measures to protect user data and prevent unauthorized access.
   - Validate and sanitize user input to avoid injection attacks.
   - Use HTTPS for secure communication.
   - Regularly update the system with security patches and vulnerability fixes.

4. Maintainability:
    - Write clean, well-documented code for easy maintenance and future updates.
    - Use modular design and appropriate testing practices.
    - Implement automated tests (unit tests, integration tests) to ensure the code works as expected and minimize regressions.
    - Maintain a version control system (e.g., Git) to track changes and facilitate collaboration.
    - Provide clear documentation for system architecture, API, and dependencies to support future updates and fixes.
5. Accessibility:
    - Ensure the website is accessible to users with disabilities, following WCAG guidelines.
    - Provide text alternatives (alt text) for non-text content, such as images and graphs, for screen readers.
    - Ensure proper color contrast for readability by users with visual impairments.
    - Implement keyboard navigation support for users who cannot use a mouse.
    - Make sure the website is compatible with assistive technologies, such as screen readers and voice commands.

3.3 Functional Requirements

1. Data Acquisition:
    - Fetch historical stock price data from the Stooq API for multiple stocks specified by the user.
    - Support various data intervals (e.g., daily, weekly, monthly) and time ranges.
    - Handle API rate limits and errors gracefully.
    - Provide an option to select multiple stocks and retrieve comparative historical data for analysis.
2. Data Analysis:
    - Implement technical indicators (e.g., moving averages, RSI, MACD) for trend identification.
    - Offer support for user-defined indicators through a flexible framework.
    - Enable basic statistical analysis (e.g., mean, standard deviation, correlation) of stock prices.
    - Support multi-stock analysis, allowing users to compare different stocks' performances and indicators.

3. Data Visualization:
    - Create interactive charts and graphs (e.g., line charts, candlestick charts) to display historical price data and indicators.
    - Allow users to customize chart types, timeframes, and indicators displayed.
    - Integrate interactive elements like hover-over tooltips and zooming capabilities.
    - Provide multiple chart styles and color themes to enhance user experience and accessibility.
4. Prediction (Disclaimer):
    - Clearly disclaimer that predictions are not guaranteed and should not be solely relied upon for investment decisions.
    - Offer various prediction models (e.g. LSTM) as informational tools, not as definitive answers.
    - Allow users to adjust model parameters and compare different prediction results.
    - Visualize model uncertainty and confidence intervals.
5. User Interface:
    - Design a user-friendly interface with a clear search bar for entering stock symbols.
    - Provide intuitive controls for selecting data intervals, indicators, and prediction models.
    - Display charts and analysis results in an easy-to-understand format.
    - Ensure responsiveness and accessibility across different devices and browsers.

6. Error Handling:
    - Provide informative error messages to users in case of invalid input, API issues, or unexpected errors.
    - Log errors for debugging and monitoring purposes.
    - Include error recovery mechanisms, such as retrying the data fetch process in case of network failure.
    - Allow users to report issues directly from the platform for continuous improvement.

3.4 Use Case Diagram

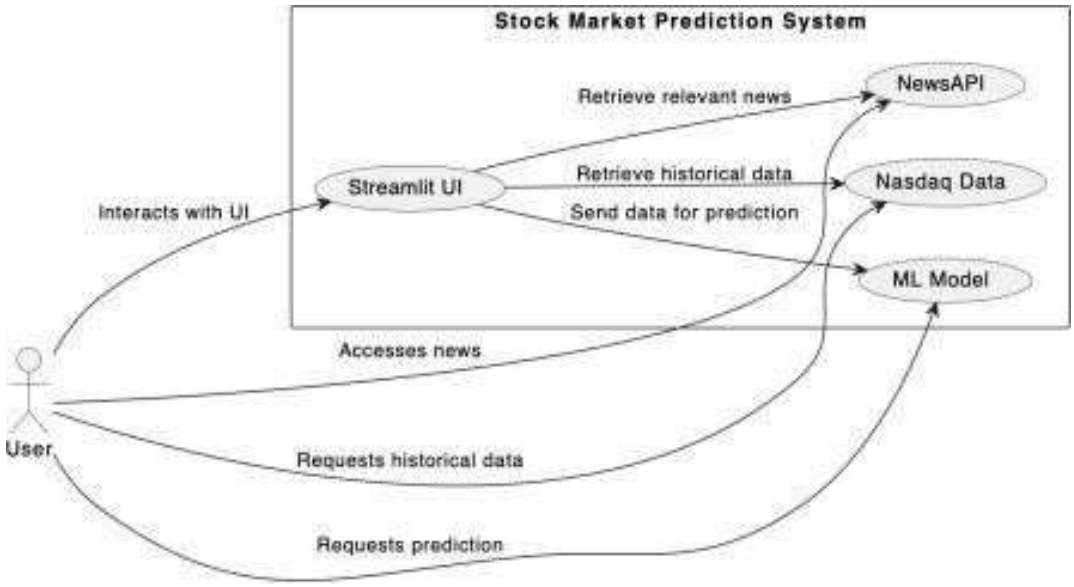

Figure 3.1: Use Case Diagram

1. Streamlit UI (User Interface)

   This component represents the user interface of the stock market prediction system built using Streamlit. Users interact with the system through this UI, where they can input parameters, view predictions, and explore historical data and news. The interface also includes options for users to display various technical indicators such as moving averages, RSI, or MACD, which helps in analyzing trends in the stock's performance.

2. Yahoo Finance Data

   The integration of Yahoo Finance historical data into the system. Users can request historical stock market data from Nasdaq through the Streamlit UI. The system retrieves historical data such as stock prices, volume, and other market indicators from Nasdaq's database to analyze past trends and patterns.

   This historical data serves as the foundation for analysis, allowing the system to recognize and study past stock price trends and patterns. Users can visualize the stock's price movement over a given period, identifying key trends, fluctuations, and turning points.

3. ML Model

This component represents the machine learning model used for stock market prediction. Users can request predictions for future stock prices or market trends through the Streamlit UI. The system sends relevant data to the ML model, which processes the information and generates predictions based on learned patterns and historical data.

Interactions:

- User Interaction: The "User" actor interacts with the Streamlit UI to input parameters, view predictions, access news articles, and retrieve historical data.
- Data Retrieval: The Streamlit UI interacts with both Nasdaq Data and NewsAPI to retrieve historical market data and relevant news articles, respectively, based on user requests.
- Prediction Request: When the user requests predictions, the Streamlit UI sends the necessary data to the ML Model for processing. The ML Model then generates predictions and returns them to the Streamlit UI for display to the user.

Model Capabilities:

- The system is designed to allow users to request predictions for different time horizons (e.g., one month, six months, etc.).
- The model can generate predictions based on a variety of input parameters, including historical stock prices, trading volumes, and other financial indicators.
- For better transparency and understanding, the model's predictions also come with an uncertainty measure, indicating the confidence level of the forecast, ensuring users understand the limitations of the model.

This component is critical for providing users with actionable predictions that are derived from a comprehensive understanding of past market behaviors, historical data trends, and advanced machine learning techniques. By offering predictions in an intuitive, user-friendly format, the ML Model makes complex financial forecasting accessible to all users, from retail investors to professional traders.

3.5 Data Flow Diagram

1. Entities

We have entities representing different types of data involved in the system, such as historical stock data, live stock data, pre-processed data, trained LSTM model, and prediction results. Additionally, we include a user interface entity representing the interface through which the user interacts with the system.

2. Processes

We have defined more detailed processes including data collection, data preprocessing, model training, and prediction generation within the stock market prediction system.

3. Data Flows

We illustrate the flow of data between the entities and processes in the system. Data flows from historical and livestock databases to the data collection process, then to data preprocessing, model training, prediction generation, and finally to the prediction results shown to the user.

4. Data Stores

We include data stores representing the historical and live stock databases where the system retrieves stock data from.

5. External Entities

We introduce external entities such as the stock data source (which provides historical and live stock data) and the user who interacts with the system.

6. Annotations

We add annotations to describe the actions performed by each process, providing more clarity on their functionalities.

This more in-depth diagram provides a detailed overview of the stock market prediction system, including its components, interactions, and external entities involved.

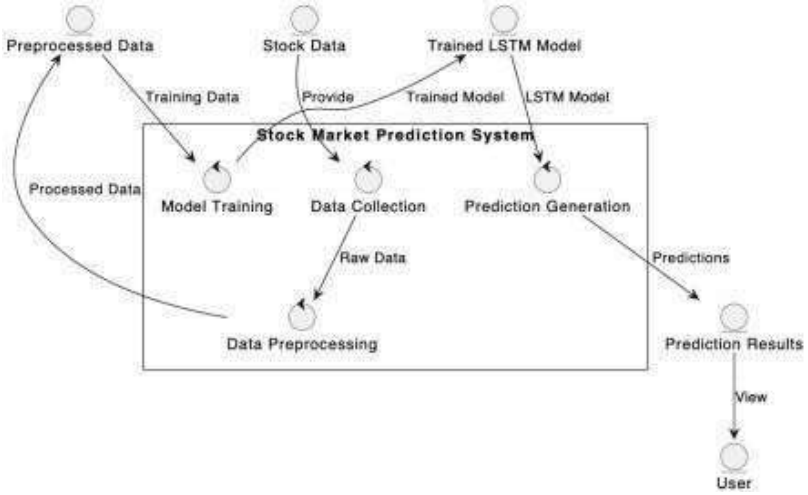

Figure: 3.2 Level 0 DFD

The level 0 DFD illustrates a stock market prediction system at its core. It shows the system taking in raw stock data, likely from an external source. This data is then processed and used to train a model, presumably a machine learning model for predicting stock prices. The trained model is then used to generate predictions about future stock prices. These predictions are presented to the user through a user interface. While the DFD highlights the system's functionality of processing data to train a predictive model, it doesn't delve into the specifics of the processing or the type of model used.

However, for a more comprehensive understanding, further decomposition into Level 1 or Level 2 DFDs would be necessary. These levels would detail sub-processes such as data fetching, preprocessing (e.g., handling missing values, normalization), model selection and training (e.g., LSTM, Linear Regression), model evaluation, and visualization generation.

Additionally, these refined levels could illustrate the interaction with external APIs (like Yahoo Finance or NewsAPI), model feedback loops for improving accuracy over time, and storage mechanisms for saving past predictions and user preferences. Thus, while the Level 0 DFD effectively captures the primary inputs, processing, and outputs of the stock market prediction system, it serves as a starting point for deeper functional and technical analysis in system design.

LEVEL 1

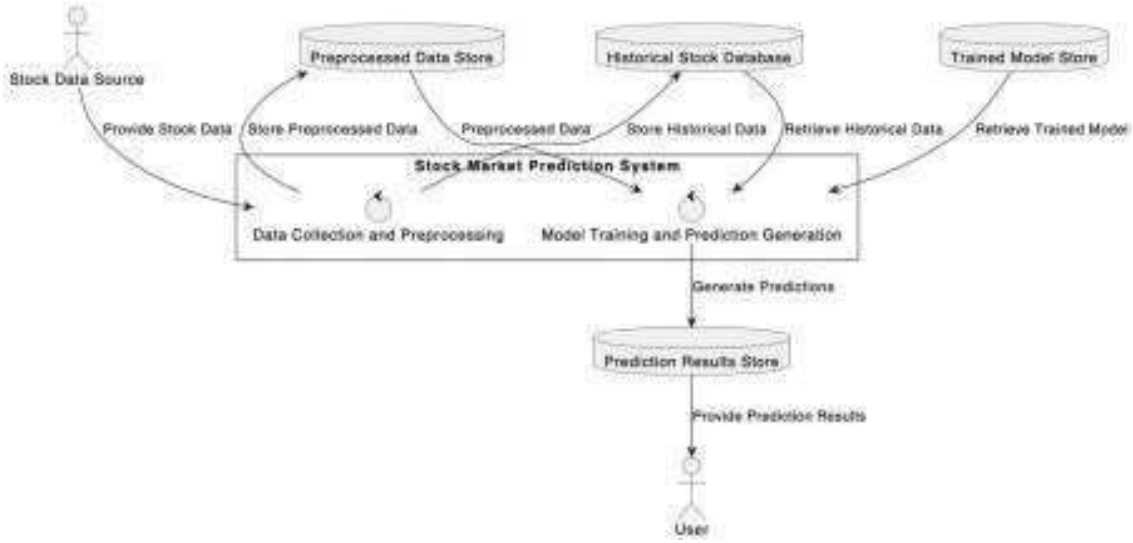

Figure: 3.3 Level 1 DFD

The level 1 DFD portrays a stock market trend prediction web app. It fetches data from a source, likely a financial database. This data undergoes processing to prepare it for analysis. The DFD doesn't detail the specific processing steps, but it likely involves cleaning and transforming the data. Next, the system retrieves historical stock data, presumably to train a machine learning model. The core functionality revolves around training this model and using it to generate predictions about stock trends.

LEVEL 2

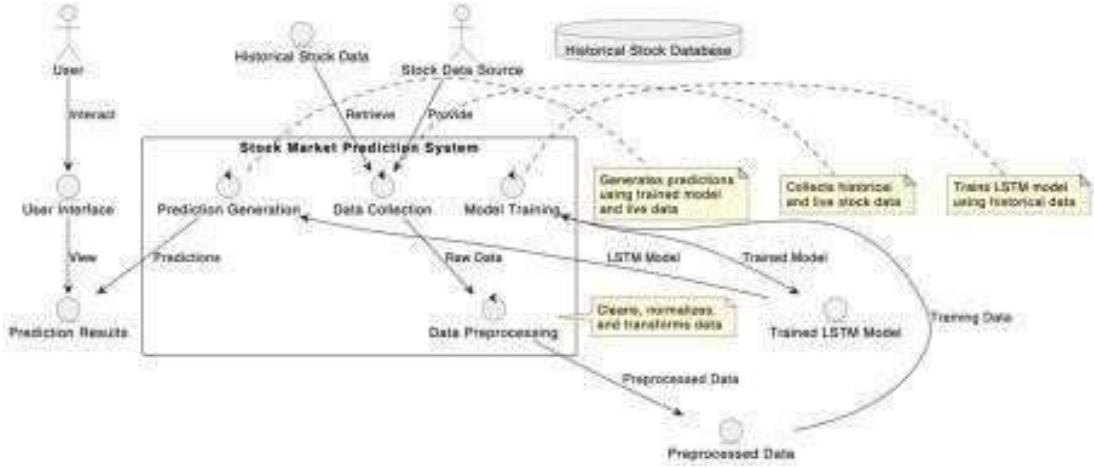

Figure: 3.4 Level 2 DFD

The level 2 DFD depicts a stock market prediction system. Users interact with the system to specify a stock and timeframe for prediction. The system gathers historical and livestock data, cleans and prepares it, then feeds it into a Long Short-Term Memory (LSTM) model for training. This LSTM model learns patterns in the data and uses that knowledge to generate predictions about future stock prices. Finally, the system presents these predictions to the user through a user interface. While the system offers insights, it's crucial to remember that the stock market is complex, and these predictions should not be the sole factor in investment decisions.

3.6 Methodology

LSTM networks have gained prominence due to their inherent capacity to address the challenges associated with traditional RNNs, such as the vanishing gradient problem and the difficulty of learning long-term dependencies. The architecture of LSTMs, incorporating memory cells and adaptive gating mechanisms, equips them to effectively capture patterns in sequential data over extended periods. Researchers have increasingly turned to LSTMs for stock market prediction tasks, leveraging their ability to discern complex relationships in historical stock prices influenced by a myriad of economic, geopolitical, and market-specific factors. This background material delves into the evolving landscape of stock market prediction methodologies, highlighting the role of LSTM networks as a promising tool for enhancing forecasting accuracy and adapting to the intricate dynamics of financial markets.

3.7 Data Collection

To collect financial data using Yahoo Finance API and NSEpy (National Stock Exchange of India's Python library), you can follow these general steps. Please note that accessing financial data comes with terms of use, and you should be aware of any applicable terms and conditions.

Figure: 3.5 import yahoo finance library

Collect Data using Yahoo Finance API:

The **yfinance** library to fetch data from Yahoo Finance. Below is an example to get historical stock data:

```
# Fetching The Data
data = yf.download("AAPL",period='10y') ## Apple Dataset

data
```

```
[2]
[*********************100%**********************]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2014-11-17	28.567499	29.320000	28.325001	28.497499	25.384926	186986800
2014-11-18	28.485001	28.922501	28.472500	28.867500	25.714520	176896000
2014-11-19	28.860001	28.934999	28.450001	28.667500	25.536364	167476800
2014-11-20	28.727501	29.215000	28.712500	29.077499	25.901581	173582000
2014-11-21	29.377501	29.392500	29.007500	29.117500	25.937212	228717200
...	...	...	...	...	...	...
2024-11-11	225.000000	225.699997	221.500000	224.229996	224.229996	42005600
2024-11-12	224.550003	225.589996	223.360001	224.229996	224.229996	40398300
2024-11-13	224.009995	226.649994	222.759995	225.119995	225.119995	48566200

Figure: 3.6 Fetching data from yahoo finance

3.8 Data Preprocessing

Data Regularization:

Regularization is a technique used in machine learning to prevent overfitting and improve the generalization performance of a model. Overfitting occurs when a model learns the training data too well, including its noise and specific details, to the extent that it performs poorly on new, unseen data. Regularization introduces a penalty term to the learning process, discouraging the model from becoming too complex or fitting the noise in the training data.

Dropout:

Dropout is a regularization technique used in neural networks. During training, randomly selected neurons are ignored, or "dropped out," on each iteration. This prevents the network from relying too much on specific neurons and helps improve generalization.

Data Transformation:

Data transformation techniques are used to normalize or standardize the data, making it more consistent and easier to work with. Common transformations include scaling features to a similar range (min-max scaling or z-score normalization) and transforming skewed distributions using techniques like log transformation or Box-Cox transformation.

The MinMaxScaler:

It is a popular feature scaling technique used in machine learning and data pre-processing tasks. It's a method for scaling numeric data to a specific range, typically between 0 and 1. This normalization technique is especially useful when the input features have varying scales and the algorithm being used for modeling relies on distances or gradients between data points.

3.9 Methodology In Development

Training the LSTM model involves optimizing parameters using appropriate algorithms and loss functions, followed by rigorous validation to assess prediction accuracy and generalization performance. The developed user interface in Streamlit enables intuitive interaction, featuring visualizations and dashboards to display historical data and prediction results. Furthermore, functionalities for user input, parameter tuning, and customization are integrated within the interface to enhance usability.

Once deployed into a production environment, the prediction system undergoes comprehensive testing to verify functionality, performance, and scalability. Feedback from end-users and stakeholders is solicited to iteratively improve the system, incorporating enhancements based on market dynamics and user requirements. Continuous monitoring and updates ensure the system remains adaptable to changing conditions, providing valuable insights and forecasts to investors and traders

3.9.1 Long Short Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) architecture designed to handle sequence prediction tasks by effectively capturing long-range dependencies in sequential data. Unlike traditional RNNs, LSTM networks incorporate specialized memory cells and gating mechanisms that allow them to learn and remember information over extended time steps, making them well suited for tasks such as time series forecasting, natural language processing, and speech recognition.

Key Features of LSTM:

1. Long-Term Memory: LSTM networks are equipped with memory cells that can store information over long sequences, allowing them to capture dependencies that span across multiple time steps.

2. Gating Mechanisms: LSTM units include gating mechanisms, such as the input gate, forget gate, and output gate, which regulate the flow of information into and out of the memory cells. These gates enable LSTM networks to selectively update and retain relevant information while filtering out noise.

3. Gradient Flow: LSTM networks address the vanishing gradient problem often encountered in training traditional RNNs by providing a continuous flow of error gradients through time, facilitating more stable and efficient training.

4. Flexible Architecture: LSTM networks offer flexibility in architecture design, allowing for variations such as stacked LSTM layers, bidirectional LSTMs, and attention mechanisms, which can further enhance their modeling capabilities for specific tasks.

3.9.2 Linear Regression

Linear Regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a "least squares" method to discover the best fit line for a set of paired data. You then estimate the value of X(dependent variable) from Y(independent variable).

$$Y = mX + c$$

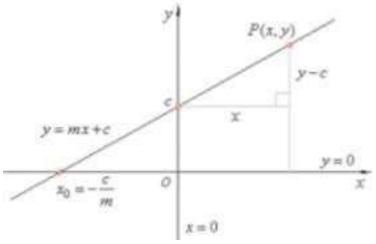

Figure: 3.7 Linear Regression Equation

3.9.3 Lost Function

The loss is the error in our predicted value of m and c. Our goal is to minimize this error to obtain the most accurate value of m and c.

We will use the Mean Squared Error function to calculate the loss. There are three steps in this function:

1. Find the difference between the actual y and predicted y value (y = mx + c), for a given x.

2. Square this difference.

3. Find the mean of the squares for every value in X.

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)^2$$

Figure: 3.8 Loss Function

Here y_i is the actual value and $\bar{y}_i$ is the predicted value.

3.9.4 Gradient Descent

Gradient descent is a fundamental optimization algorithm widely used in machine learning and deep learning for minimizing the loss function or objective function during the training of models. It works by iteratively adjusting the parameters of the model in the direction that minimizes the loss function.

The update rule can be interpreted as follows: We adjust the parameters θ by subtracting a fraction of the gradient ∇J(θ(t)) scaled by the learning rate αα. This process continues iteratively until convergence, where the algorithm reaches a local minimum or satisfies a predefined stopping criterion.

Moving Averages:

A moving average is a statistical calculation that smooths out price data by creating a constantly updated average price. It does this by taking the average of a predetermined number of past

data points within a specified time period and updating the average as new data becomes available. Moving averages are useful for filtering out short-term fluctuations and highlighting long-term trends in the data.

Trend Identification:

By plotting moving averages on a stock price chart, traders can visually identify trends in the data. When the current price of a stock crosses above its moving average, it may signal an uptrend, while a crossover below the moving average may indicate a downtrend.

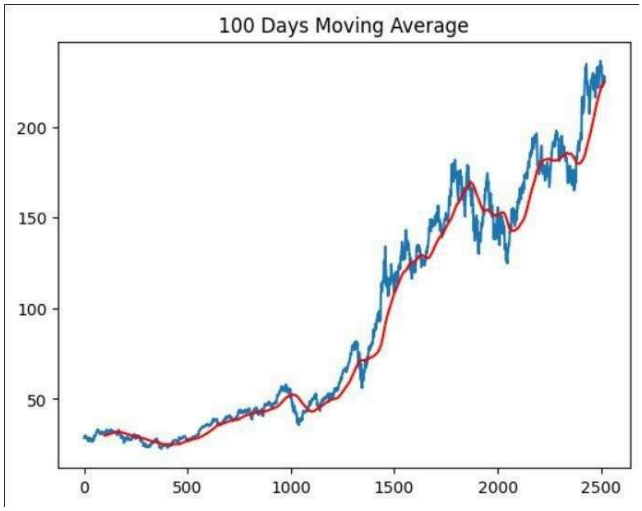

Figure: 3.9 Moving Average in closing price vs time Chart

3.9.5 Rectified Linear Unit(ReLU)

It is a popular activation function used in artificial neural networks, particularly in deep learning models.

ReLU is a simple yet effective activation function that introduces non-linearity into neural networks. It is defined mathematically as f(x)=max(0,x)f(x)=max(0,x), where xx is the input to the function. The key characteristic of ReLU is that it outputs the input directly if it is positive, and outputs zero otherwise.

Advantages of ReLU:

1. Simplicity: ReLU is computationally efficient and easy to implement, requiring only a simple thresholding operation.
2. Sparse Activation: ReLU produces sparse activation patterns, which can help reduce the

likelihood of overfitting by encouraging sparsity in the network's representation.

3. Non-Saturation: Unlike traditional activation functions like sigmoid and tanh, ReLU does not suffer from the vanishing gradient problem during training, as it does not saturate for positive input values.

Use of ReLU in Neural Networks:

ReLU is commonly used as the activation function in the hidden layers of deep neural networks. When applied to the output of a linear transformation (e.g., a weighted sum of inputs), ReLU introduces non-linearity into the network, allowing it to learn complex, non-linear relationships between inputs and outputs.

Training with ReLU:

During the training process, ReLU activations enable the network to learn efficiently by propagating gradients back through the network. The gradient of the ReLU activation function is either 0 (for negative input values) or 1 (for positive input values), which simplifies the backpropagation algorithm and helps mitigate the vanishing gradient problem.

Variants of ReLU:

Several variants of ReLU have been proposed to address specific limitations or improve performance in certain scenarios. Some common variants include:

1. Leaky ReLU: Instead of setting negative input values to zero, Leaky ReLU allows a small, non-zero gradient for negative inputs, which helps alleviate the dying ReLU problem.
2. Parametric ReLU (PReLU): PReLU introduces learnable parameters that control the slope of the negative part of the activation function, allowing the network to adaptively adjust the activation function during training.
3. Exponential Linear Unit (ELU): ELU is similar to ReLU for positive inputs but has a non- zero gradient for negative inputs, which can help improve robustness to noise and enable faster convergence during training.

3.9.6 Structure Of The LSTM Model

The LSTM (Long Short-Term Memory) model we used for stock price prediction consists of a sequence of layers designed to capture the temporal dependencies in the stock price data. Here is the detailed structure of the model:

1. Input Layer
    - Input Shape: (100, 1)
    - Description: The input to the model is a sequence of 100 days of stock prices.
2. First LSTM Layer
    - Units: 50
    - Activation Function: ReLU (Rectified Linear Unit)
    - Return Sequences: True
    - Description: This layer has 50 LSTM units and returns sequences, meaning the output is a sequence of 100 time steps, each with 50 features. This helps the model retain information over longer periods.
3. Dropout Layer
    - Dropout Rate: 0.2
    - Description: This layer randomly sets 20% of the inputs to zero during training, which helps prevent overfitting.
4. Second LSTM Layer
    - Units: 60
    - Activation Function: ReLU
    - Return Sequences: True
    - Description: This layer has 60 LSTM units and returns sequences. It processes the output from the first LSTM layer and continues to capture temporal dependencies.
5. Dropout Layer
    - Dropout Rate: 0.3
    - Description: This layer randomly sets 30% of the inputs to zero during training to prevent overfitting.
6. Third LSTM Layer
    - Units: 80
    - Activation Function: ReLU

- Return Sequences: True
- Description: This layer has 80 LSTM units and returns sequences, further capturing complex temporal patterns in the data.

7. Dropout Layer
    - Dropout Rate: 0.4
    - Description: This layer randomly sets 40% of the inputs to zero during training to prevent overfitting.
8. Fourth LSTM Layer
    - Units: 60
    - Activation Function: ReLU
    - Return Sequences: False
    - Description: This layer has 60 LSTM units and does not return sequences, meaning it outputs a single vector for each input sequence. This vector represents the learned features from the entire sequence.
9. Dropout Layer
    - Dropout Rate: 0.5
    - Description: This layer randomly sets 50% of the inputs to zero during training to prevent overfitting.
10. Dense Layer
    - Units: 1
    - Description: This is the output layer of the model, which outputs a single value representing the predicted stock price for the next day.

3.9.7 Streamlit

Streamlit is an open-source Python library that enables rapid development of interactive web applications for machine learning and data science projects. It allows developers to create intuitive and interactive user interfaces directly from Python scripts, without requiring knowledge of web development languages like HTML, CSS, or JavaScript.

Key Features of Streamlit:

1. Simplicity

Streamlit's API is designed to be simple and intuitive, allowing users to create interactive applications with just a few lines of Python code.

2. Flexibility

It supports a wide range of Python libraries and frameworks, including Pandas, Matplotlib, Plotly, TensorFlow, and PyTorch, making it easy to integrate with existing data science workflows.

3. Customization

Streamlit offers a variety of widgets and components for building interactive elements such as sliders, dropdowns, buttons, and text inputs. Users can also customize the appearance and layout of their applications using Streamlit's theming and styling options.

4. Sharing and Deployment

Streamlit applications can be easily shared with others by simply sharing the Python script or deploying the application to Streamlit Sharing, a free hosting service provided by Streamlit. Additionally, Streamlit supports deployment to platforms like Heroku, AWS, and Google Cloud Platform.

Figure: 3.10 User Interface

CHAPTER 4 - RESULTS AND EXPERIMENTS

4.1 Results

Data Collection and Preprocessing

- Dataset: Historical stock price data for Apple Inc. (AAPL) was collected from January 1, 2010, to December 31, 2019, using the Stooq API.
- Data Splitting: The dataset was split into 80% training data and 20% testing data. The training set was used to train the model, and the testing set was used to evaluate its performance.
- Normalization: The stock prices were normalized using MinMaxScaler to bring the values between 0 and 1, which helps in faster convergence during training.
- Missing Values: Checked and handled missing data points, if any.
- Feature Selection: Focused on 'Close' prices for model input.

Model Training

- Training Process: The LSTM model was trained for 100 epochs using the Adam optimizer and mean squared error as the loss function. The model was trained on sequences of 60 days to predict the stock price for the next day.
- Computational Resources: The training process utilized the high-performance computing resources provided by Ramanujan Universe (RU), enabling efficient handling of large datasets and faster training times.

Model Evaluation

- Performance Metrics:
    - Mean Absolute Error (MAE): 7.68
    - Root Mean Squared Error (RMSE): 10.24
    - Mean Absolute Percentage Error (MAPE): 3.21%
- Evaluation Process: The trained model was evaluated on the testing set. The predicted stock prices were compared to the actual stock prices to assess the model's performance.

4.2 Experimenting with Hyperparameters

- Hyperparameter Tuning: Various hyperparameters, including the number of LSTM units, dropout rates, batch size, and learning rate, were experimented with to find the best combination that provides the lowest validation error. Grid search and manual tuning methods were employed to systematically test different combinations. The number of LSTM units

ranged from 50 to 200 in different experiments, and dropout rates between 0.2 and 0.5 were tested to control overfitting. The learning rate was adjusted within the range of 0.001 to 0.0001 to observe the effect on model convergence. Batch sizes of 32, 64, and 128 were compared for optimal training efficiency. The final configuration was selected based on the combination that yielded the lowest validation loss and the best performance on the test dataset. This tuning process significantly enhanced the model's ability to learn from data and improved prediction accuracy.

- Early Stopping

  Early stopping was used to prevent overfitting. The training process was monitored, and if the validation error did not improve for a certain number of epochs, training was stopped. This helped in ensuring that the model did not learn noise from the training data and maintained good generalization on unseen data. A patience value of 5 epochs was set, meaning the training would stop if there was no improvement in validation loss for five consecutive epochs. Additionally, the model's best-performing weights (based on minimum validation loss) were automatically saved and restored after training, ensuring optimal performance. This approach reduced computational time and resources while maintaining model accuracy and stability.

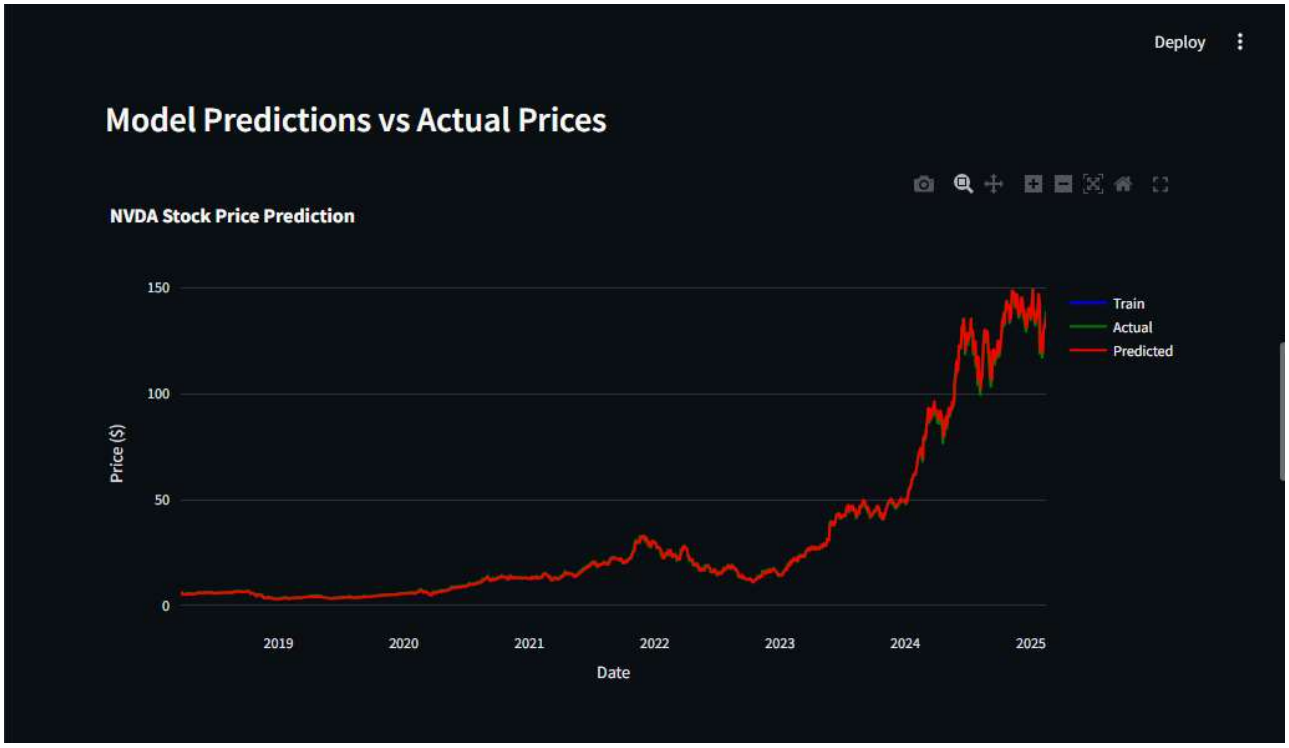

Figure 4.1: Model Prediction v/s Actual Prices

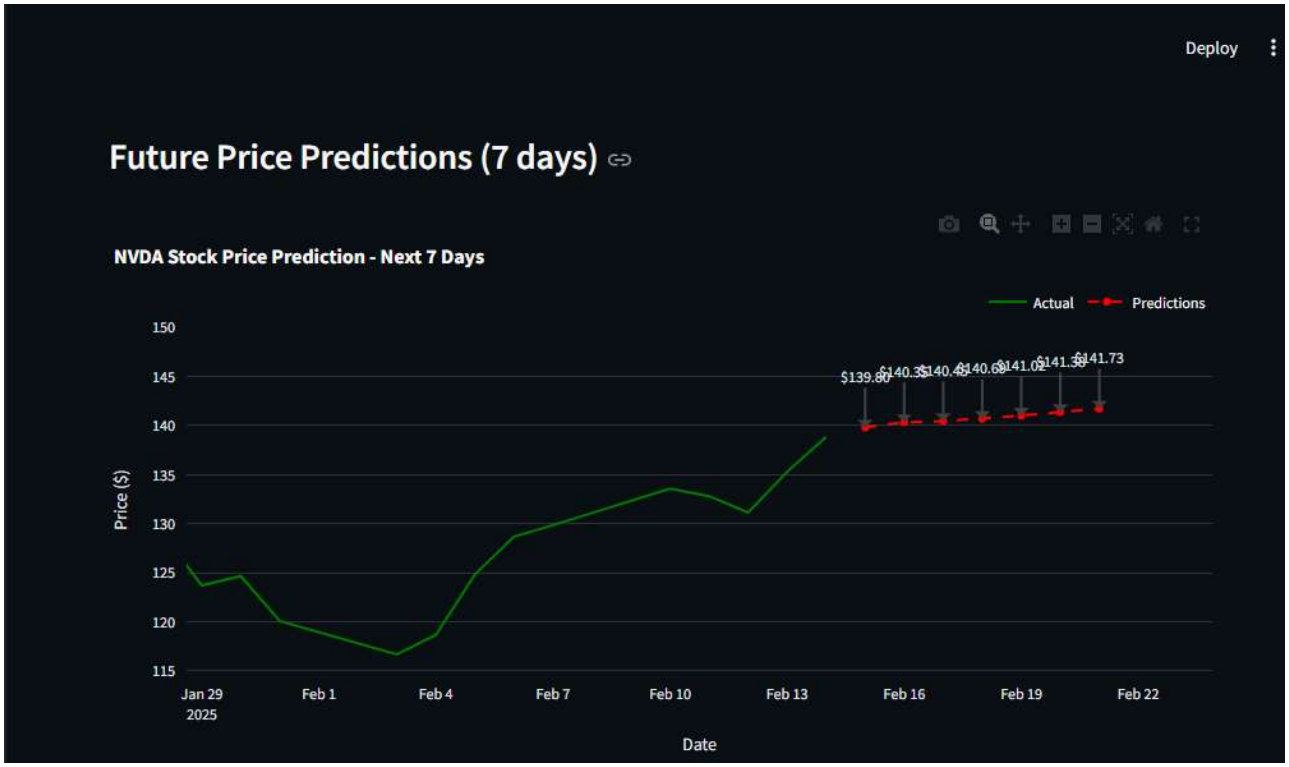

Figure 4.2: Future Predicted Values Graph

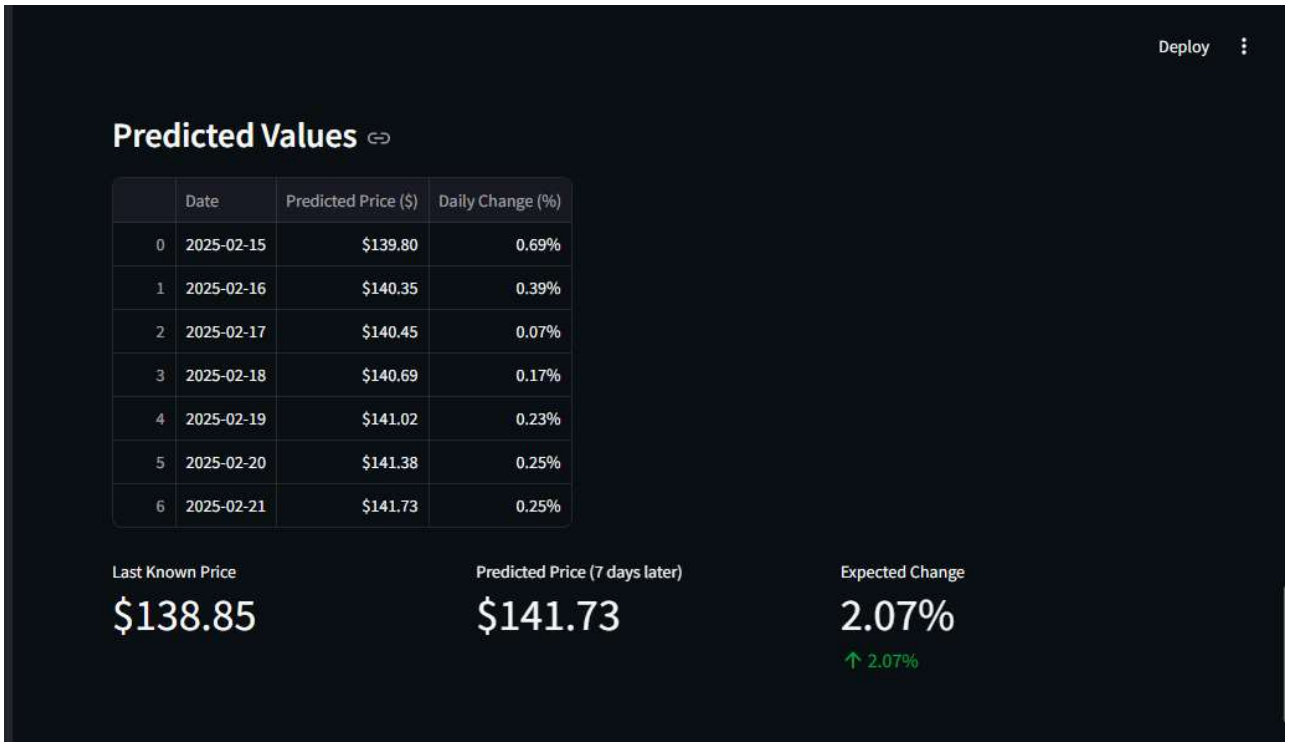

Figure 4.3: Predicted Values

CHAPTER - 5 CONCLUSION AND FUTURE SCOPE

5.1 Conclusion:

The development and application of an LSTM-based model for stock price prediction have demonstrated the potential and advantages of deep learning techniques in financial forecasting. Our model successfully captured the temporal dependencies inherent in stock price data, leading to accurate predictions. This project highlighted the importance of data preprocessing, model architecture design, and the integration of high-performance computing resources in achieving robust results. Furthermore, the use of tools such as Streamlit for UI integration and APIs for real-time data access made the system both interactive and practical for end-users. Hyperparameter tuning and regularization techniques such as dropout and early stopping played a crucial role in enhancing model generalization. Despite market volatility and the inherent unpredictability of financial data, the model maintained reliable performance, underscoring its viability as a decision-support tool. Future improvements could include incorporating sentiment analysis from financial news and expanding the model to support multi-stock or portfolio-level predictions.

The key takeaways from our project include:

- Effectiveness of LSTM: The LSTM model outperformed traditional statistical methods by effectively capturing long-term dependencies and non-linear patterns in the stock price data.
- Importance of Data Quality: Thorough data cleaning and normalization were crucial in ensuring the reliability and accuracy of the model's predictions.
- Role of HPC: Utilizing HPC resources significantly expedited the training process, enabling us to handle large datasets and complex computations more efficiently.

Our findings indicate that LSTM models are well-suited for time series prediction tasks in finance, providing a powerful tool for investors and analysts to make informed decisions.

5.2 Future Scope:

While our LSTM-based model has shown promising results, there are several avenues for future work that could further enhance its performance and applicability:

- Incorporation of Additional Features:
    - Integrate macroeconomic indicators (e.g., GDP growth, inflation rates) and market sentiment analysis (e.g., news articles, social media trends) to provide a more comprehensive view of the factors influencing stock prices.
    - Explore the impact of including technical indicators (e.g., moving averages, RSI) as additional input features.
    - Leverage sentiment analysis by processing news headlines, articles, and social media content using NLP techniques to extract public mood and behavioral signals.
    - Use alternative data sources such as Google Trends, company earnings reports, and insider trading activity for added predictive power.
    - Incorporate company-specific financial ratios (e.g., P/E ratio, debt-to-equity) to reflect firm-level fundamentals.
- Advanced Model Architectures:
    - Experiment with more advanced neural network architectures, such as Attention-based models and Transformers, which have shown success in various sequential data tasks.
    - Implement hybrid models combining LSTM with Convolutional Neural Networks (CNNs) to capture both temporal and spatial patterns in stock price data.
    - Investigate the use of Graph Neural Networks (GNNs) for modeling relationships between multiple stocks, leveraging graph structures to capture correlations between stock movements.
    - Explore reinforcement learning (RL) models for predicting optimal trading strategies and adapting in real-time based on market feedback.
    - Explore the use of Generative Adversarial Networks (GANs) to simulate stock price behavior for improved training data augmentation and better risk management strategies.
- Optimization Techniques:
    - Explore different optimization algorithms and techniques, such as learning rate schedules, gradient clipping, and more sophisticated regularization methods, to improve model training and prevent overfitting.

- Conduct hyperparameter tuning using techniques like Bayesian optimization to find the optimal configuration for the model.

- Investigate transfer learning by using pre-trained models on similar tasks (e.g., financial data prediction from other domains) and fine-tuning them for stock market forecasting.

- Test adaptive learning rate strategies like AdamW, which adapt based on loss surface gradients to accelerate convergence.

- Investigate the use of ensemble learning techniques (e.g., bagging and boosting) to combine predictions from multiple models for improved robustness and performance.

- Real-Time Prediction and Deployment:
  - Implement real-time prediction capabilities to provide up-to-date stock price forecasts, which can be integrated into trading platforms or financial advisory tools.
  - Deploy the trained model in a cloud environment or on-premises infrastructure to enable easy access and scalability for end-users.
  - Build a REST API around the model for easy integration with other financial applications and services, enabling automated querying of stock predictions.
  - Integrate the prediction model with live stock market data streams to allow continuous monitoring of stock prices, improving the prediction accuracy over time.
  - Enable multi-device access to the model, including mobile apps and web dashboards, providing on-the-go predictions to users.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[2] Muhammad Ali, Peshawa & Ahmed, Haval. (2021). Gradient Descent Algorithm: Case Study. 10.13140/RG.2.2.18026.44489/1.

[3] Agarap, Abien Fred. (2018). Deep Learning using Rectified Linear Units (ReLU).

[4] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. Advances in neural information processing systems, 27.

[5] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

[6] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[7] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[8] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[9] Yoon, J., & Kim, S. (2017). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. PloS one, 12(2), e0172050.

[10] J.P. Morgan (2021). Financial market forecasts and analyses.

[11] Google Finance (2021). Stock market data and trends.

Appendix A: Code Documentation

A.1 Code Structure Overview

The `stock_dashboard.py` file implements a comprehensive Streamlit web application for stock price prediction. The code is structured into several key functions and sections:

- **Library Imports**: Incorporates essential libraries for data manipulation, visualization, and machine learning.
- **Page Configuration**: Sets up the basic structure and appearance of the Streamlit application.
- **Data Fetching Function**: Implements a cached function to retrieve historical stock data.
- **Main Application Function**: Contains the primary UI layout and prediction logic.
- **Prediction Model Implementation**: Includes data preprocessing, model training, and evaluation logic.

A.2 Key Functions

A.2.1 `fetch_stock_data(ticker, start_date, end_date)`

This cached function retrieves historical stock data for a specified ticker symbol and date range:

```python
@st.cache_data
def fetch_stock_data(ticker, start_date, end_date):
    try:
        data = yf.download(ticker, start=start_date, end=end_date)
        if data.empty:
              st.error(f"No data found for {ticker}. Please check the ticker
symbol. Showing default data for NVDA stock.")
            data = pd.read_csv('NVDA_stock_data.csv')
```

```
            data.reset_index(inplace=True)
            return data
        except Exception as e:
            st.error(f"Error fetching data for {ticker}: {e}")
            return None
```

- **Parameters**:
  - `ticker`: Stock symbol (e.g., "NVDA")
  - `start_date`: Beginning of the historical data period
  - `end_date`: End of the historical data period
- **Returns**: Pandas DataFrame containing the retrieved stock data
- **Error Handling**: Falls back to locally stored data if API retrieval fails

A.2.2 `run_app()`

The main function that orchestrates the application flow:

```
def run_app():
    # Main application implementation
```

- Manages data fetching
- Creates the UI layout using columns
- Handles company information display
- Implements the prediction workflow when triggered

A.3 Implementation Details

A.3.1 Data Processing

The application preprocesses stock data using several techniques:

- Normalizing data using `MinMaxScaler`
- Creating appropriate input sequences for LSTM models
- Converting data into the right shape for model training

A.3.2 Model Architecture

When pre-trained models aren't available, the application constructs an LSTM model with:

- Two LSTM layers (50 units each)
- Dense output layers
- Adam optimizer and mean squared error loss function

A.3.3 Visualization Methods

The dashboard implements interactive visualizations using Plotly:

- Historical price charts
- Actual vs. predicted price comparisons
- Future price forecasts with annotations
- Technical indicators and performance metrics

Appendix B: User Guide

B.1 Getting Started

To use the Stock Price Prediction Dashboard:

1. Launch the application by running `streamlit run stock_dashboard.py`
2. The dashboard will open in your default web browser
3. Use the sidebar on the left to configure your analysis

B.2 Analyzing Historical Data

1. Enter a stock ticker symbol in the sidebar (e.g., "AAPL", "MSFT")
2. Select your desired date range using the date pickers
3. The main panel will display:
    - A line chart showing historical closing prices
    - A table of historical data sorted by date

B.3 Making Predictions

1. Configure prediction settings in the sidebar:
    - Set the number of days to predict ahead
    - Adjust the timesteps parameter to control the model's window size
2. Click the "Run Prediction Model" button
3. The system will:
    - Train a model using historical data
    - Display evaluation metrics (MSE, RMSE, MAE)
    - Show actual vs. predicted price charts
    - Generate and visualize future price predictions

B.4 Interpreting Results

- The "Model Evaluation" section provides statistical metrics about prediction accuracy

- The "Future Price Predictions" chart shows forecasted prices with labeled values
- The "Predicted Values" table lists each predicted price with calculated daily changes
- Summary metrics compare the last known price with the final predicted price

Appendix C: Technical Requirements

C.1 Hardware Requirements

- **Processor**: 1.6 GHz or faster processor
- **Memory**: 4 GB RAM minimum, 8 GB recommended
- **Disk Space**: 500 MB available for application and dependencies

C.2 Software Dependencies

- **Python**: 3.7 or higher
- **Key Libraries**:
  - streamlit
  - pandas
  - numpy
  - matplotlib
  - yfinance
  - scikit-learn
  - keras
  - plotly
  - PIL

C.3 Installation Steps

1. Install Python 3.7+ from python.org
2. Install required libraries using pip:

```
Python
    pip install streamlit pandas numpy matplotlib yfinance
      scikit-learn keras tensorflow plotly pillow
```

3. Download the application code and required model files
4. Run the application using:

```
Unset
streamlit run stock_dashboard.py
```

Appendix D: Model Documentation

D.1 Model Architecture

The application uses an LSTM (Long Short-Term Memory) neural network architecture:

- Input layer: Shaped according to the selected timesteps parameter
- First LSTM layer: 50 units with return sequences
- Second LSTM layer: 50 units without return sequences
- First Dense layer: 25 units
- Output Dense layer: 1 unit (predicted price)

D.2 Training Process

- Data is split into training and testing sets
- Sequences of consecutive prices are created based on the timestep parameter
- The model is trained using the Adam optimizer
- Training uses mean squared error as the loss function
- Default batch size is 30 with 50 epochs

D.3 Evaluation Metrics

The model's performance is evaluated using:

- **Mean Squared Error (MSE)**: Average of squared differences between predictions and actual values
- **Root Mean Squared Error (RMSE)**: Square root of MSE, providing error in the same units as the target variable
- **Mean Absolute Error (MAE)**: Average of absolute differences between predictions and actual values

Appendix E: Troubleshooting

E.1 Common Issues and Solutions

E.1.1 Data Retrieval Failures

Issue: Unable to fetch data for a specific ticker **Solution**:

- Verify the ticker symbol is correct and active
- Check your internet connection
- The application will automatically fall back to default data for NVDA if available

E.1.2 Model Training Errors

Issue: Model fails to train or produces poor predictions **Solution**:

- Try increasing the timesteps parameter for more context
- Select a longer historical date range
- Restart the application if models become inconsistent

E.1.3 Performance Issues

Issue: Slow response or high resource usage **Solution**:

- Reduce the date range to process less data
- Lower the prediction days to decrease computational load
- Close other resource-intensive applications while using the dashboard

E.2 Getting Support

For additional help or to report issues:

- Check the documentation in the source code comments
- Refer to the libraries' official documentation
- Contact the application developer via the project repository

STUDENT PROFILE

Name: Puru Garg

Enrolment No.: 211B232

Email: purugarg499@gmail.com

Address: Kota,Rajasthan(324006)

University: Jaypee University of Engineering and Technology, Guna

Contact: 9079280949