ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

A Major Project Report on

Revolutionizing Stock Price Prediction Using LSTM

Submitted in Partial Fulfilment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

Artificial Intelligence and Machine Learning

Submitted by

Tudimilla Dheeraj Kumar Chary 20EG107150
K. Venkata Kavya 20EG107127
N. Nagarjun Reddy 20EG107133

Under the Guidance of

R. Sathya Prakash
Assistant Professor
Department of Artificial Intelligence
Anurag University
Venkatapur (V), Ghatkesar (M), Medchal-Malkajgiri (Dt)-500088

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

DEPARTMENT OF ARTIFICIAL INTELLIGENCE CERTIFICATE

This is to certify that the project report titled "Revolutionizing Stock Price Prediction" is being submitted by Tudimilla Dheeraj Kumar Chary, K. Venkata Kavya, N. Nagarjun Reddy bearing roll number 20EG107150, 20EG107127, 20EG107133 in IV Year B.Tech II semester Artificial Intelligence and Machine Learning is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Student's Name	Student's Signature
1. Tudimilla Dheeraj Kumar Chary	1.
2. K. Venkata Kavya	2.
3. N. Nagarjun Reddy	3.

Internal Guide
Mr. R. Sathya Prakash

Head of the Department
Dr. A. Mallikarujana Reddy

External Examiner

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

ACKNOWLEDGMENT

We owe our gratitude to Prof. S. Ramchandram, Vice-Chancellor, Anurag University, for extending the University facilities to the successful pursuit of our project so far and his kind patronage.

We acknowledge our deep sense of gratitude to Prof. Balaji Utla, Registrar, Anurag University, for being a constant source of inspiration and motivation.

We wish to record our profound gratitude Dr. V. Vijaya Kumar, Dean School of Engineering, for his motivation and encouragement.

We sincerely thank Dr. A. Mallikarjuna Reddy, Associate Professor and the Head of the Department of Artificial Intelligence, Anurag University, for all the facilities provided to us in the pursuit of this project.

We owe a great deal to our project coordinator Mrs. Neetha Reddy, Assistant Professor, Department of Artificial Intelligence, Anurag University for supporting us throughout the project work.

We are indebted to our project guide Mr. R. Sathya Prakash, Assistant Professor, Department of Artificial Intelligence, Anurag University. We feel it's a pleasure to be indebted to our guide for his valuable support, advice, and encouragement and we thank him for his superb and constant guidance towards this project.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

TABLE OF CONTENTS

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

ABSTRACT

The advent of deep learning techniques, particularly Long short-term memory (LSTM) networks, has sparked a revolution in the realm of stock price prediction. This paper proposes a novel approach to revolutionize stock price prediction by harnessing the power of LSTM networks. Traditional methods of predicting stock prices have often relied on simplistic models or technical indicators, which may struggle to capture the intricate dynamics of financial markets. In contrast, LSTM networks offer the capability to effectively capture temporal dependencies and nonlinear relationships in time series data, making them well-suited for stock price prediction tasks. In this study, we leverage LSTM networks to develop a robust and accurate model for predicting stock prices. We employ a comprehensive dataset comprising historical stock prices, trading volumes, and other relevant financial indicators to train and evaluate our LSTM model. Through extensive experimentation and evaluation, we demonstrate the superior predictive performance of our proposed LSTM-based approach compared to conventional methods. Furthermore, we explore various techniques to enhance the robustness and generalization capability of our model, including feature engineering, hyperparameter tuning, and ensemble methods. Our findings highlight the effectiveness of LSTM networks in capturing complex patterns inherent in stock price data, thereby offering valuable insights for investors, traders, and financial analysts. Overall, this research contributes to the ongoing advancement of stock price prediction methodologies and underscores the potential of LSTM networks in revolutionizing predictive analytics in financial markets. By harnessing the power of deep learning techniques, we aim to empower stakeholders with more accurate and reliable forecasts, ultimately facilitating informed decision-making and driving positive outcomes in the realm of finance.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

LIST OF FIGURES

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

LIST OF TABLES

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

SYMBOLS AND ABBREVIATIONS

Symbols and Abbreviations	Description
LSTM	Long Short-Term Memory
API	Application Program Interface
GUI	Graphical User Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER ONE
INTRODUCTION

➢ *Understanding Stocks:*

A Cornerstone of Financial Markets Stocks, known interchangeably as equities or shares, serve as tangible representations of ownership in a company. When investors, spanning from individual traders to large institutional entities, purchase stocks, they effectively acquire a stake in the underlying company. This ownership stake grants them certain privileges, including the right to a proportionate share of the company's assets and earnings.

The allure of stocks lies in their dynamic nature, serving as essential instruments within the framework of capitalism. Traditionally exchanged on stock exchanges, stocks offer investors a gateway to engage with the growth and prosperity of companies operating across diverse sectors and industries. By investing in stocks, individuals and institutions alike become integral participants in the economic ecosystem, contributing to the expansion and evolution of businesses on a global scale.

Moreover, stocks embody the core principles of capitalism, facilitating the efficient allocation of resources and capital within the economy. Through the issuance and trading of stocks, companies gain access to vital funding necessary for expansion, innovation, and development. Simultaneously, investors have the opportunity to deploy their capital strategically, seeking avenues for potential growth and wealth accumulation.

In essence, stocks represent more than mere financial instruments; they encapsulate the spirit of entrepreneurship, innovation, and investment. As investors navigate the complexities of financial markets, stocks serve as vehicles for wealth creation, risk diversification, and portfolio growth. In this symbiotic relationship between companies and investors, stocks emerge as conduits for fostering economic growth, prosperity, and opportunity.

➢ *Deciphering Stock Prices:*

The Pulse of Market Dynamics Stock prices serve as the barometer of a company's perceived worth within the financial markets, encapsulating the collective assessment of investors regarding its present and future prospects. They represent the culmination of myriad influences and interactions within the complex ecosystem of financial markets, where the forces of supply and demand converge to determine their trajectory.

The dynamics of stock prices are driven by an intricate interplay of various factors, each exerting its influence on market sentiment and investor behavior. Chief among these influences are the fundamental performance metrics of the company itself. Factors such as revenue growth, earnings per share, profit margins, and overall financial health play a pivotal role in shaping investor perceptions and, consequently, stock prices. Positive earnings reports, robust sales figures, and strategic business developments often result in upward pressure on stock prices, reflecting optimism about the company's prospects.

In addition to internal company factors, macroeconomic indicators wield significant influence over stock prices. Economic data points such as gross domestic product (GDP) growth, inflation rates, unemployment figures, and interest rate movements can impact investor sentiment and market sentiment. For instance, a strong GDP growth report may bolster investor confidence, leading to increased demand for stocks and upward price movements. Conversely, rising inflation or interest rates may dampen investor enthusiasm, triggering selloffs and downward pressure on stock prices.

Moreover, investor sentiment, often characterized by emotions such as fear, greed, and optimism, plays a crucial role in driving stock price movements. Market psychology and sentiment can be influenced by a myriad of factors, including news headlines, corporate scandals, geopolitical tensions, and broader market trends. Positive news such as product launches, strategic partnerships, or favorable regulatory developments can instill confidence and fuel buying interest, driving stock prices higher. Conversely, negative events or uncertainties may trigger panic selling and lead to sharp declines in stock prices.

Furthermore, regulatory developments and global geopolitical events can introduce volatility and uncertainty into financial markets, impacting stock prices across sectors and regions. Changes in regulatory policies, trade agreements, or geopolitical tensions can disrupt supply chains, alter business dynamics, and influence investor perceptions of risk, leading to fluctuations in stock prices.

In this dynamic and interconnected landscape, stock prices are subject to constant flux, reflecting the ebb and flow of market dynamics and the collective wisdom of investors. Understanding the myriad factors that shape stock prices is essential for investors, traders, and financial analysts seeking to navigate the complexities of financial markets and make informed investment decisions.

➢ *Unraveling the Art of Stock Price Prediction*

Stock price prediction stands as a cornerstone within the financial landscape, embodying the relentless pursuit of forecasting future price movements driven by historical data, prevailing market trends, and pertinent contextual factors. However, this pursuit is fraught with complexities arising from the inherent volatility and unpredictability that characterize financial markets. Despite

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

their informative nature, conventional methods of stock price prediction— encompassing statistical models, technical analysis, and fundamental analysis—often grapple with capturing the nuanced intricacies and nonlinear dynamics inherent in stock price movements.

At its core, stock price prediction represents a formidable challenge that necessitates grappling with the multifaceted nature of financial markets. These markets, shaped by the collective actions of myriad participants, exhibit a propensity for rapid fluctuations and unforeseen developments. The confluence of factors such as investor sentiment, economic indicators, corporate performance metrics, regulatory changes, and geopolitical events introduces a degree of complexity that eludes simplistic prediction models.

Conventional methods of stock price prediction, grounded in established financial principles and analytical frameworks, offer valuable insights into market dynamics. Statistical models leverage historical data to identify patterns and trends, providing a basis for forecasting future price movements. Technical analysis delves into market sentiment and price patterns, utilizing tools such as charts, indicators, and oscillators to identify potential buy and sell signals. Fundamental analysis, on the other hand, delves into the intrinsic value of stocks, assessing factors such as earnings growth, dividends, market share, and competitive positioning.

However, despite their utility, these conventional methods often fall short in capturing the full spectrum of complexities inherent in financial markets. The dynamic and adaptive nature of markets, coupled with the emergence of new information and unforeseen events, renders static prediction models inadequate in providing accurate forecasts. Moreover, the reliance on historical data may overlook emerging trends or structural shifts within the market, leading to inaccuracies in predictions.

Enterprises such as LSTM networks, a form of deep learning technology, have emerged as potent tools in addressing the shortcomings of conventional prediction methods. By leveraging neural networks capable of learning from sequential data and capturing longterm dependencies, LSTM networks offer a more nuanced approach to stock price prediction. Unlike traditional models, which rely on predefined rules and assumptions, LSTM networks possess the ability to adapt and evolve in response to changing market conditions, thereby enhancing their predictive accuracy.

In conclusion, stock price prediction represents a formidable challenge within the financial domain, requiring a multifaceted approach that transcends conventional methods. While statistical models, technical analysis, and fundamental analysis offer valuable insights into market dynamics, they often fall short in capturing the intricacies and nonlinear dynamics inherent in stock price movements. As financial markets continue to evolve and grow increasingly complex, innovative technologies such as LSTM networks hold the potential to revolutionize stock price prediction, offering more accurate and reliable forecasts in an everchanging landscape.

- *Enter the Era of LSTM:*

A Paradigm Shift in Predictive Analytics in recent years, the field of stock price prediction has witnessed a transformative revolution with the emergence of deep learning techniques, notably the Long short-term memory (LSTM) networks. This advancement has fundamentally altered the way we approach and understand the dynamics of financial markets. LSTM networks, a specialized variant of recurrent neural networks (RNNs), have emerged as a powerful tool specifically designed to tackle the challenges inherent in modeling sequential data, making them particularly wellsuited for predicting stock prices.

Unlike traditional methods that often rely on linear models or statistical approaches, LSTM networks offer a fundamentally different paradigm for understanding and forecasting stock price movements. These networks are uniquely equipped to handle the temporal dependencies and nonlinear patterns present in time-series data, which are characteristic of financial markets.

One of the key advantages of LSTM networks lies in their ability to capture long-term dependencies within sequential data. Traditional RNNs often struggle with the vanishing gradient problem, where gradients become increasingly smaller as they propagate back through time, making it difficult for the network to learn long-term dependencies. LSTM networks address this issue by introducing a gating mechanism that allows them to selectively remember or forget information over time, enabling them to capture and retain relevant patterns over longer sequences.

Moreover, LSTM networks excel at discerning intricate patterns within time-series data. Their architecture includes specialized memory cells that maintain a memory state over time, allowing them to capture subtle variations and trends within the data. This capability is particularly advantageous in financial markets, where stock prices are influenced by a myriad of factors ranging from microeconomic fundamentals to macroeconomic indicators and investor sentiment.

Furthermore, LSTM networks are inherently flexible and adaptable, allowing them to learn complex relationships and adapt to changing market conditions. This adaptability is crucial in financial markets, where conditions can evolve rapidly, and historical patterns may not always hold true in the future. By continuously learning from new data and adjusting their internal representations, LSTM networks can provide more accurate and robust predictions of stock price movements.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

Overall, the advent of LSTM networks has ushered in a new era of stock price prediction, marked by unprecedented accuracy and reliability. As researchers and practitioners continue to explore the capabilities of deep learning techniques, LSTM networks are poised to play an increasingly pivotal role in understanding and navigating the complexities of financial markets, ultimately empowering investors and analysts with valuable insights for making informed decisions.

- *Pioneering the Future of Stock Price Prediction with LSTM*

Embarking on this pioneering journey signifies our commitment to revolutionizing stock price prediction by harnessing the formidable prowess of LSTM networks. At the core of our mission lies the aspiration to construct a robust and precise model capable of prognosticating future stock prices with unprecedented accuracy, thereby furnishing invaluable insights for stakeholders spanning from individual investors and seasoned traders to seasoned financial analysts.

Our endeavor is propelled by the recognition of the limitations inherent in conventional prediction methodologies, which often struggle to capture the intricate nuances and dynamic complexities of financial markets. By embracing advanced deep learning techniques, particularly LSTM networks, we aim to transcend these constraints and chart new frontiers in predictive analytics within the realm of financial markets.

The cornerstone of our approach lies in the judicious application of LSTM networks, which are uniquely equipped to handle the temporal dependencies and nonlinear patterns prevalent in timeseries data. By leveraging the inherent capabilities of LSTM networks to discern and assimilate complex patterns, we seek to unlock deeper insights into the underlying dynamics driving stock price movements.

Furthermore, we are committed to fostering innovation and pushing the boundaries of predictive analytics through rigorous experimentation and empirical validation. By meticulously finetuning model parameters, optimizing performance metrics, and exploring novel methodologies, we endeavor to develop a model that not only predicts stock prices with unparalleled accuracy but also exhibits robustness and reliability across diverse market conditions.

Our aspiration extends beyond mere prediction; we envision our model as a valuable decisionmaking tool, empowering stakeholders with actionable insights and foresight into market trends. Whether it's identifying lucrative investment opportunities, managing portfolio risks, or formulating strategic trading strategies, our model aims to serve as a trusted ally for investors and analysts navigating the complex terrain of financial markets.

In essence, our journey represents a bold leap forward in the quest for enhanced understanding and mastery of stock price prediction. By embracing innovation and leveraging the transformative potential of LSTM networks, we strive to reshape the landscape of predictive analytics within financial markets, ultimately empowering stakeholders with the tools and knowledge needed to thrive in an ever-evolving economic ecosystem.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER TWO
LITERATURE SURVEY

➢ *Existing System:*

- *Convolutional Neural Networks (CNN):*

CNNs have gained widespread recognition for their efficacy in various domains, including image recognition and natural language processing. In the realm of stock price prediction, CNNs have emerged as a promising tool for feature extraction from time-series data. By leveraging convolutional filters, CNNs can effectively capture local patterns and dependencies within sequential financial data, offering a novel approach to modeling stock price movements.

- *Recurrent Neural Networks (RNN):*

RNNs have garnered considerable attention for their ability to model sequential data, owing to their inherent capacity to capture temporal dependencies. In stock price prediction, RNNs have been employed to analyze historical trends and forecast future price movements. However, traditional RNNs are susceptible to the vanishing gradient problem, which impedes their ability to capture long-term dependencies effectively.

- *Stacked Autoencoders:*

Stacked autoencoders have emerged as a prominent technique for unsupervised feature learning and dimensionality reduction. In the context of stock price prediction, stacked autoencoders have been explored for their potential to extract meaningful representations from raw financial data. By compressing input sequences into a lowerdimensional latent space, stacked autoencoders facilitate feature extraction and enhance the predictive capabilities of subsequent models.

- *Long Short-Term Memory (LSTM):*

LSTM networks represent a significant advancement in sequential data modeling, specifically designed to overcome the limitations of traditional RNNs. In the domain of stock price prediction, LSTM networks have garnered widespread attention for their ability to capture temporal dynamics and nonlinear relationships in financial time-series data. Equipped with specialized memory cells and gating mechanisms, LSTM networks excel at retaining relevant information over extended sequences, making them well-suited for forecasting stock prices.

➢ *Limitation of Existing System:*

- *Limitations of CNNs:*

While CNNs excel in capturing spatial features in images, they may struggle to capture long-term dependencies in sequential data, such as stock price movements. This limitation arises from their design, which is primarily geared towards extracting local patterns through convolutional filters. Consequently, CNNs may not effectively capture the complex temporal relationships inherent in financial time series data, especially over extended time horizons.

- *Challenges with Traditional RNNs:*

Traditional RNNs, including vanilla RNN architectures, encounter the vanishing gradient problem, which hampers their ability to learn and retain information over long sequences. As a result, these models may struggle to capture and utilize long-term dependencies in sequential data, leading to suboptimal predictive performance. Additionally, traditional RNNs may exhibit difficulty in capturing subtle patterns and trends within noisy financial data, further diminishing their effectiveness in stock price prediction tasks.

- *Issues with Stacked Autoencoders:*

Stacked autoencoders, while effective in unsupervised feature learning and dimensionality reduction tasks, may face challenges in effectively encoding complex temporal patterns present in raw financial data. This limitation arises from the inherent difficulty of compressing high-dimensional sequential data into a lowerdimensional latent space while preserving relevant information. In the presence of noise and variability, stacked autoencoders may struggle to extract meaningful representations, limiting their utility in stock price prediction tasks.

- *Limited Robustness and Generalization Capability:*

Existing deep learning models for stock price prediction may exhibit limited robustness and generalization capability when confronted with diverse market conditions and unseen data. The complexity and volatility of financial markets introduce inherent uncertainties and non-stationarities, which pose significant challenges for predictive modeling. Consequently, models trained on historical data may fail to generalize effectively to new market conditions or unforeseen events, resulting in diminished predictive performance and reliability.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

➢ *Gaps Identified:*

Despite the advancements in stock price prediction using deep learning techniques, there are still several critical gaps that need to be addressed to enhance the effectiveness and reliability of predictive models:

- *Limited Understanding of Market Dynamics:*

One of the fundamental challenges in stock price prediction is the limited understanding of the underlying mechanisms driving market dynamics. Financial markets are influenced by a multitude of factors, including economic indicators, investor sentiment, geopolitical events, and regulatory changes. However, the complex interactions and interdependencies among these factors make it challenging to accurately model and predict stock price movements. Further research is needed to gain deeper insights into the dynamics of financial markets and identify the key drivers of stock price changes.

- *Challenges in Capturing Long-Term Dependencies:*

Financial time-series data often exhibit long-term dependencies and nonlinear relationships that are difficult to capture using traditional modeling techniques. Deep learning models, such as LSTM networks, have shown promise in capturing temporal dynamics and extracting patterns from sequential data. However, challenges remain in effectively modelinglong-term dependencies and nonlinear relationships, particularly in the presence of noisy and volatile market conditions. Developing more robust deep learning architectures and training methodologies that can effectively capture and exploit long-term dependencies is essential for improving the accuracy of stock price predictions.

- *Lack of Robustness and Generalization Capability:*

Existing deep learning models for stock price prediction may lack robustness and generalization capability when faced with unseen data and diverse market conditions. Financial markets are inherently dynamic and subject to rapid changes, making it challenging for predictive models to adapt and generalize to new scenarios. Improving the robustness and generalization capability of deep learning models requires the development of more sophisticated regularization techniques, ensemble methods, and model evaluation frameworks. Additionally, incorporating techniques such as transfer learning and domain adaptation can help improve the performance of models across different market conditions and data distributions.

- *Need for Interpretable and Explainable Models:*

The lack of interpretability and explainability in deep learning models poses challenges for stakeholders in the financial industry, who require transparency and trust in predictive analytics. While deep learning models may achieve high predictive accuracy, understanding the underlying factors driving predictions is often challenging. Interpretable and explainable models are essential for gaining insights into the rationale behind predictions and identifying potential sources of bias or error. Developing interpretable deep learning architectures and model explanation techniques can help enhance transparency and trust in predictive analytics within financial markets.

Addressing these gaps requires interdisciplinary collaboration between researchers, practitioners, and domain experts in the fields of finance, machine learning, and computational modeling. By leveraging advances in deep learning techniques, data analytics, and domainspecific knowledge, researchers can develop more robust and reliable predictive models for stock price prediction, ultimately empowering stakeholders with actionable insights for informed decision-making in financial markets.

➢ *Problem Statement:*

The challenge lies in revolutionizing stock price prediction by leveraging LSTM networks to overcome the limitations of traditional methods. The objective is to develop robust models capable of accurately forecasting stock prices amidst market volatility. This entails harnessing LSTM's capacity to capture long-term dependencies and adapt to changing market conditions. The goal is to empower stakeholders with actionable insights for informed decisionmaking in dynamic financial landscapes. Thus, the urgent need is to advance predictive analytics through innovative methodologies tailored to the complexities of financial time-series data.

➢ *Objectives*

The primary objective is to pioneer a paradigm shift in stock price prediction by leveraging the power of Long short-term memory (LSTM) networks. We aim to develop and implement a cutting-edge predictive model that transcends the limitations of traditional methods, offering unparalleled accuracy, reliability, and adaptability in forecasting stock price movements.

Enhance Predictive Accuracy and Reliability: Our goal is to significantly improve the predictive accuracy and reliability of stock price prediction models by harnessing the capabilities of LSTM networks. By effectively capturing temporal dependencies and nonlinear relationships in financial time-series data, we aim to develop a model that can provide more precise and robust forecasts of stock prices across various market conditions.

Adapt to Changing Market Dynamics: We aim to develop a model that can adapt dynamically to changing market dynamics and evolving trends. LSTM networks are well-suited for capturing long-term dependencies and temporal patterns, enabling our model to effectively adapt to shifts in market sentiment, economic indicators, and other relevant factors that influence stock price movements.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

Enhance Generalization and Robustness: We recognize the importance of developing a model that can generalize effectively across diverse market conditions and datasets. To achieve this, we will train our LSTM-based predictive model on a comprehensive dataset containing a wide range of historical stock price data, spanning different market regimes and scenarios. By exposing the model to diverse datasets and market conditions, we aim to enhance its generalization capability and robustness, ensuring reliable performance in real-world settings.

Provide Transparent and Interpretable Predictions: In addition to predictive accuracy, we aim to provide transparent and interpretable predictions to users. We will integrate explainability mechanisms into our LSTM-based predictive model, enabling users to understand the rationale behind the model's predictions and gain insights into the factors driving stock price movements. By enhancing transparency and interpretability, we aim to build trust and confidence among users, facilitating informed decision-making in financial markets.

Comprehensive Evaluation and Validation: We will conduct comprehensive evaluations to assess the effectiveness, robustness, and generalization capability of our LSTMbased predictive model. This includes quantitative assessments using standard evaluation metrics such as mean absolute error, root mean square error, and correlation coefficients, as well as qualitative analysis of the model's performance on real-world datasets and scenarios. Through rigorous evaluation and validation, we aim to demonstrate the superiority of our LSTM-based predictive model compared to existing methods and establish its efficacy in revolutionizing stock price prediction.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER THREE
PROPOSED SYSTEM

A. Architecture

➢ *Data Collection and Preprocessing:*

- *Data Sources:*
  Historical stock price data and relevant financial indicators are collected from various sources, such as financial databases, APIs, and market data providers.

- *Data Preprocessing:*
  The collected data undergoes preprocessing steps, including cleaning, normalization, and feature engineering. Missing values are handled, and features are scaled to ensure uniformity across the dataset.

➢ *Feature Selection and Engineering:*

- *Feature Selection:*
  Relevant features such as stock prices, trading volume, price-to-earnings ratio, moving averages, and technical indicators are selected based on their significance in predicting stock price movements.

- *Feature Engineering:*
  Additional features may be engineered to capture specific patterns or relationships in the data. This could include lagged values, rolling statistics, and transformations to enhance the model's predictive capabilities.

➢ *Model Architecture:*

- *Input Layer:*
  The input layer receives the pre-processed historical data and financial indicators as input.

- *LSTM Layers:*
  Multiple LSTM layers are stacked sequentially to capture temporal dependencies and long-term patterns in the data. Each LSTM layer consists of memory cells and gating mechanisms that allow the model to retain and utilize information over extended sequences.

- *Dense Layers:*
  Following the LSTM layers, dense (fully connected) layers are used to transform the learned features into predictions of future stock prices.

- *Output Layer:*
  The output layer produces the predicted stock prices based on the learned representations from the preceding layers.

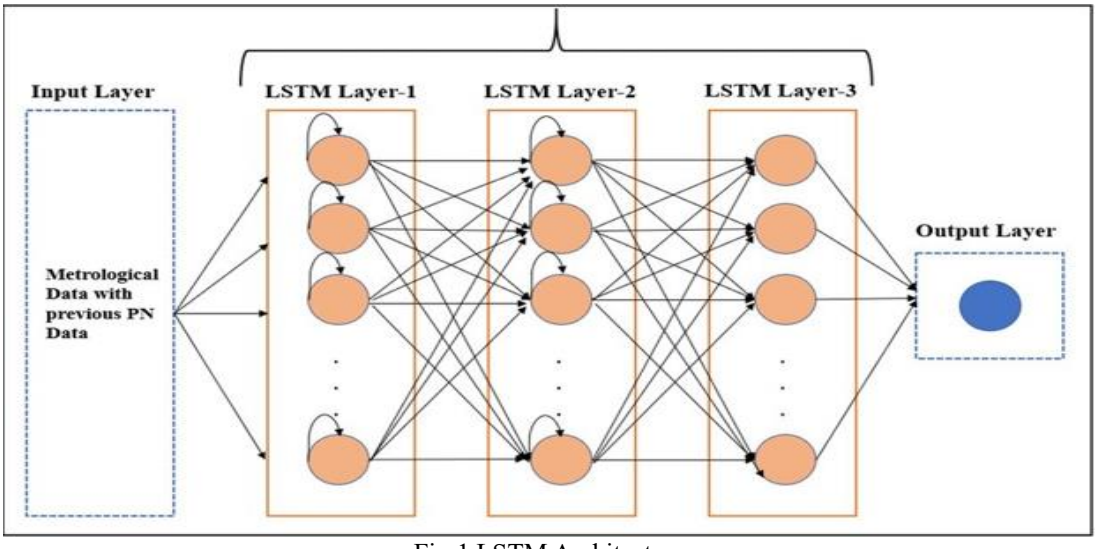

Fig 1 LSTM Architecture

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

➢ *Training Process:*

- *Loss Function:*

A suitable loss function, such as mean squared error (MSE) or mean absolute error (MAE), is chosen to quantify the disparity between the predicted and actual stock prices. Optimization Algorithm: Gradient-based optimization algorithms like stochastic gradient descent (SGD) or Adam are employed to adjust the model parameters and minimize the loss function iteratively.

- *Regularization Techniques:*

Techniques like dropout and L2 regularization may be employed to prevent overfitting and improve the generalization capability of the model.

- *Hyperparameter Tuning:*

Hyperparameters such as learning rate, batch size, and number of LSTM layers are finetuned to optimize the model's performance.

➢ *Evaluation and Validation:*

- *Validation Split:*

The dataset is split into training and validation sets to evaluate the model's performance during training and prevent overfitting.

- *Performance Metrics:*

Standard evaluation metrics such as mean absolute error (MAE), root mean square error (RMSE), and accuracy are used to assess the model's predictive accuracy and generalization capability.

- *Cross Validation:*

Cross-validation techniques may be employed to validate the model's performance across different subsets of the data and ensure robustness.

➢ *Integration:*

- *Frontend Integration:*

The trained model is integrated into a frontend application using frameworks like Streamlit. This allows users to interactively input parameters and receive predictions through a user-friendly interface, enabling seamless integration of the predictive model into decision-making processes without extensive technical knowledge.

By leveraging the architecture outlined above, the proposed system aims to revolutionize stock price prediction by providing accurate, reliable, and interpretable forecasts that empower stakeholders in the financial industry to make informed decisions.

B. Requirements & Specifications

➢ *Software Requirements*

Software Requirements specify the logical characteristics of each interface and software components of the system. The following are some software requirements.

Table 1 Software Requirements

Software	Version	Description
Python	3.11.6	Programming language for system development
TensorFlow	2.14.0	Deep learning framework for building LSTM model
Streamlit	1.31.1	Web application framework for frontend development
Plotly	5.18.0	An open-source library that can be used for data visualization and understanding data simply and easily.

➢ *Hardware Requirements*

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

Table 2 Hardware Requirements

Hardware	Description
CPU/GPU	Computational resources for model training Minimum 8GB
Memory	RAM for processing large datasets and model training 16GB And Above
Storage	Solid-state drive (SSD) for storing historical data, trained model weights, and application files

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER FOUR
DESIGN

A. DFD

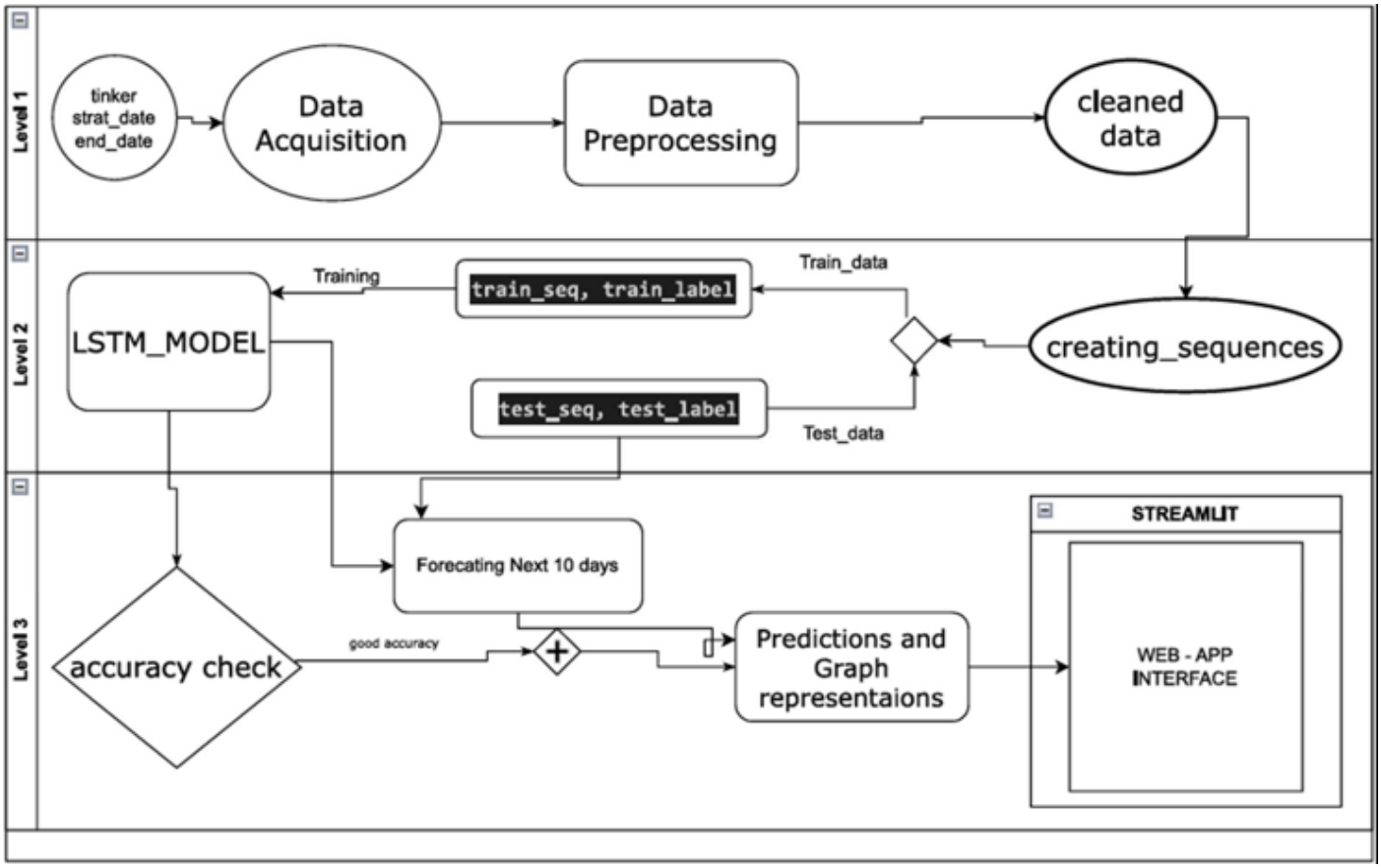

Fig 2 Data Flow Diagram

B. Module Design and Organization

➢ *Data Preprocessing Module:*

- Responsible for preprocessing raw stock price data before feeding it into the LSTM model. Includes functions for data cleaning, normalization, feature engineering, and sequence generation.
- Ensures that the input data is in the appropriate format for training the LSTM model.

➢ *LSTM Model Architecture Module:*

- Defines the architecture of the LSTM neural network for stock price prediction.
- Includes functions for building the LSTM model with configurable parameters such as several layers, units, activation functions, and dropout rates.
- Allows for customization of the model architecture based on specific requirements and optimization strategies.

➢ *Training Module:*

- Handles the training of the LSTM model using historical stock price data.
- Utilizes the pre-processed data to train the LSTM model with configurable training parameters such as batch size, learning rate, and number of epochs.
- Implements mechanisms for monitoring training progress, evaluating performance metrics, and saving model checkpoints.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

➢ *Prediction Module:*

- Facilitates the use of the trained LSTM model for making stock price predictions.
- Includes functions for loading the trained model weights and generating predictions for future stock price movements.
- Provides options for customizing prediction intervals and visualizing predicted trends.

➢ *Evaluation Module:*

- Evaluates the performance of the LSTM model using standard evaluation metrics.
- Calculates metrics such as mean absolute error, root mean square error, accuracy, precision, recall, and score.
- Generates comprehensive evaluation reports and visualizations to assess the model's accuracy, robustness, and generalization capability.

➢ *Frontend Interface Module (Using Streamlit):*

- Develops an interactive web interface for users to interact with the stock price prediction system.
- Utilizes Streamlit framework for building the frontend webpage with intuitive widgets and controls.
- Integrates functionalities for inputting data, selecting model parameters, displaying predictions, and visualizing results in real-time.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER FIVE
IMPLEMENTATION & TESTING

A. Technology Used

Python: Python is chosen as the primary programming language for its simplicity, readability, and extensive ecosystem of libraries and frameworks. It is widely adopted in the data science and machine learning communities due to its ease of use and versatility. Python's rich library ecosystem enables developers to efficiently implement various aspects of the stock price prediction system, from data preprocessing to model training and deployment.

Fig 3 Python

➢ *TensorFlow:*

TensorFlow is a powerful deep learning framework developed by Google, known for its flexibility, scalability, and performance in building and training neural networks. It provides high-level abstractions for defining complex neural network architectures, including LSTM networks used in time series prediction tasks like stock price forecasting. TensorFlow's computational graph execution model allows for efficient training and inference on both CPU and GPU architectures, making it an ideal choice for developing the predictive model in the system.

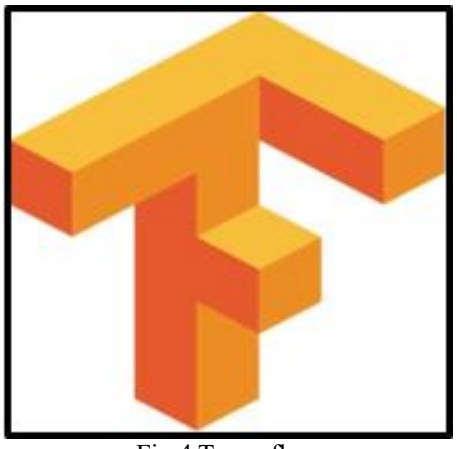

Fig 4 Tensorflow

➢ *Git:*

Git is a distributed version control system widely used in software development for tracking changes to the codebase, facilitating collaboration among developers, and managing project history. With Git, developers can work on different features or branches simultaneously and merge changes seamlessly, ensuring code integrity and version management. By adopting Git, the development process of the stock price prediction system becomes more organized, transparent, and efficient.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

Fig 5 Git

➢ *Scikitlearn:*

Scikitlearn is a popular machine-learning library in Python, renowned for its simplicity, consistency, and ease of use. It provides a wide range of algorithms for classification, regression, clustering, and dimensionality reduction, along with utilities for model evaluation, preprocessing, and cross-validation. While the system primarily focuses on deep learning with TensorFlow, Scikitlearn complements the deep learning components by providing additional machine learning capabilities for tasks such as feature engineering, data preprocessing, and model evaluation.

Fig 6 Scikit Learn

➢ *NumPy:*

NumPy is a fundamental library for numerical computing in Python, offering support for multidimensional arrays, mathematical functions, linear algebra operations, and random number generation. It provides the foundation for numerical operations and data manipulation in the system, enabling efficient processing of large datasets and matrix operations required for building and training machine learning models.

Fig 7 Numpy

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

➢ *Pandas:*

Pandas is a versatile data manipulation library in Python, widely used for handling structured data and timeseries data structures. It provides powerful data structures like DataFrame and Series, along with functions for data cleaning, transformation, indexing, and aggregation. In the stock price prediction system, Pandas facilitates the preprocessing of raw financial data, handling missing values, scaling features, and organizing data into suitable formats for model training and evaluation.

Fig 8 Pandas

➢ *Plotly:*

Plotly is a versatile and interactive visualization library that provides a wide range of chart types, including line plots, scatter plots, bar charts, and more. It offers interactive features such as zooming, panning, and hover tooltips, making it ideal for visualizing complex data and trends. Plotly can enhance the system by providing rich and dynamic visualizations of stock price data, model predictions, and evaluation metrics.

Fig 9 Plotly

➢ *Tiingo API:*

Tiingo API is a financial data API that provides access to real-time and historical stock price data, company fundamentals, and market information. It enables the stock price prediction system to fetch relevant financial data and market indicators necessary for training the predictive model and making informed predictions. By integrating tiingo API into the system, developers can access a wealth of financial data, enriching the analysis and forecasting capabilities of the system.

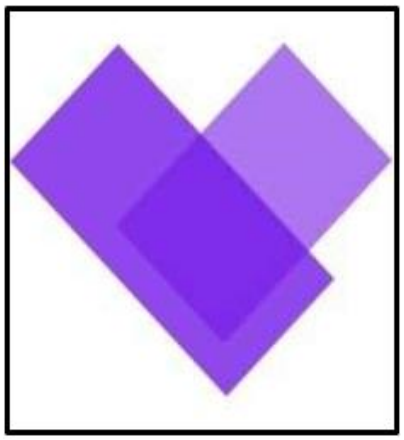

Fig 10 Tiingo

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

➢ *VSCode:*

Visual Studio Code (VSCode) is a popular code editor developed by Microsoft, known for its lightweight, extensible, and feature-rich environment for software development. It offers built-in support for Python development, along with a wide range of extensions for debugging, version control, and project management. With its intuitive user interface and powerful features, VS-Code serves as the integrated development environment (IDE) for developing the stock price prediction system, providing developers with a productive and customizable environment for coding, testing, and debugging.

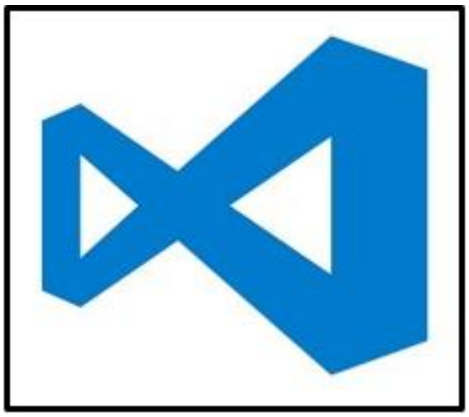

Fig 11 VSCode

B. Procedures

➢ *Project Setup:*

Establishing a new project directory and version control repository using Git provides a structured framework for organizing code and facilitates collaboration among team members. It ensures that changes to the codebase are tracked, managed, and documented effectively.

Creating a virtual environment using tools like virtual env or conda helps manage dependencies and ensures that the project's requirements are isolated from other system environments. This prevents conflicts between different versions of libraries and packages.

Installing necessary packages such as TensorFlow or PyTorch, Streamlit, Plotly, Pandas, NumPy, Matplotlib, and Scikitlearn sets the foundation for system development. These packages provide essential functionalities for data processing, machine learning, visualization, and web development.

➢ *Data Acquisition and Preprocessing:*

Acquiring historical stock price data from reliable sources like tiingo API is critical for training and testing the system. Access to accurate and up-to-date financial data ensures the robustness and effectiveness of the predictive model.

Preprocessing the acquired data involves several steps, including cleaning, normalization, and transformation. Cleaning the data involves handling missing values, removing duplicates, and correcting errors to ensure data integrity. Normalizing the data scales the features to a consistent range, facilitating convergence during model training. Transformation techniques such as feature engineering extract relevant information and create input sequences suitable for training the LSTM model.

➢ *Model Development:*

Designing and developing the architecture of the LSTM model is a crucial step in the development process. This involves defining the structure of the neural network, including the number of layers, units, activation functions, and dropout rates. The architecture is tailored to capture the temporal dependencies and nonlinear relationships present in the stock price data. Compiling the model with appropriate loss functions, optimizers, and evaluation metrics prepares it for training. Choosing suitable loss functions, such as mean squared error or binary cross-entropy, and optimizers, such as Adam or RMSprop, affects the model's ability to learn from the data and make accurate predictions.

Training the LSTM model involves feeding the pre-processed stock price data into the model and iteratively adjusting its parameters to minimize the loss function. This process requires careful tuning of hyperparameters such as batch size, learning rate, and number of epochs to achieve optimal performance.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

- *Evaluation and Validation:*

Evaluating the performance of the trained LSTM model using standard evaluation metrics provides insights into its accuracy, robustness, and generalization capability. Metrics such as mean absolute error, root mean square error, accuracy, precision, recall, and score quantify the model's predictive performance and guide further refinement.

Validating the model's predictions against real-world data and scenarios ensures its effectiveness in practical applications. This step involves comparing predicted stock prices with actual values and assessing the model's ability to capture market trends and dynamics accurately.

- *Frontend Development with Streamlit:*

Developing an interactive frontend interface using Streamlit enhances the user experience and facilitates easy interaction with the stock price prediction system. Streamlit provides intuitive widgets and controls for data input, parameter selection, prediction display, and result visualization, enabling users to explore and analyze the predictions effectively.

Integrating Streamlit into the frontend interface involves designing layout elements, defining callback functions, and incorporating interactive components such as sliders, dropdown menus, and plot widgets. Customizing the interface to meet user requirements and preferences enhances usability and engagement.

- *Integration and Testing:*

Integrating all system modules, including data preprocessing, model development, evaluation, and frontend interface, ensures seamless communication and data flow between components. This step involves linking individual modules, handling dependencies, and verifying the overall functionality of the system.

Conducting thorough testing, including unit tests, integration tests, and end-to-end tests, identifies and resolves any bugs, errors, or inconsistencies in the system. Testing ensures that each module performs as expected and that the system operates reliably under different conditions and scenarios.

- *Documentation and Maintenance:*

Providing comprehensive documentation for the stock price prediction system aids users in understanding its architecture, functionality, and usage. Documentation includes detailed instructions on data preprocessing, model development, frontend interface usage, and testing procedures, enabling users to navigate and utilize the system effectively.

Ensuring ongoing maintenance and support for the system involves monitoring performance, addressing user feedback, and implementing updates and improvements as needed. Regular maintenance activities, such as code refactoring, bug fixing, and performance optimization, help sustain the reliability and effectiveness of the system over time.

C. Testing & Validation

- *Data Splitting:*

The dataset is divided into three subsets: training, validation, and testing. The training set is used for training the LSTM model, the validation set for tuning hyperparameters and preventing overfitting, and the testing set for evaluating the final model's performance.

- *Training the LSTM Model:*

The LSTM model learns patterns and relationships in historical stock price data using the training dataset. During training, the model adjusts its parameters to minimize prediction error based on an optimization algorithm such as gradient descent.

- *Validation Set Evaluation:*

The LSTM model's performance is evaluated on the validation set during training. This helps monitor its generalization ability and prevent overfitting. Metrics like loss function values and evaluation metrics are computed to assess performance.

- *Hyperparameter Tuning:*

Hyperparameters, such as the learning rate and several LSTM layers, significantly impact the model's performance. Tuning involves systematically varying these parameters and evaluating performance on the validation set to find the optimal configuration.

- *Final Model Selection:*

After hyperparameter tuning, the model with the best performance on the validation set is selected. This model configuration is then used for testing on the unseen testing dataset to evaluate its real-world performance.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

➢ *Testing Set Evaluation:*

The final model is evaluated on the testing dataset to measure its predictive accuracy and generalization capability. Predictions are compared against actual stock prices, and evaluation metrics like mean absolute error and root mean square error are computed.

➢ *Analysis of Results:*

Results from testing and validation are analyzed to understand the model's strengths, weaknesses, and areas for improvement. Visualization techniques can be used to compare predicted and actual stock prices visually.

➢ *Iterative Refinement:*

Based on the analysis, the model may undergo iterative refinement to improve performance. This may involve revisiting data preprocessing steps, adjusting hyperparameters, or exploring alternative architectures.

➢ *Design Test Cases and Scenarios*

- *Data Preprocessing Test Cases:*

✓ Test Case 1: Verify that missing values in the dataset are handled appropriately during preprocessing.
✓ Test Case 2: Validate that data normalization is performed correctly, ensuring that all features are scaled to a consistent range.
✓ Test Case 3: Ensure that feature engineering techniques capture relevant patterns and relationships in the data effectively.
✓ Test Case 4: Verify that the train validation test split is done correctly, maintaining data integrity and randomness across subsets.

date	open	close	high	low
2009-03-26	107.830	109.87	109.98	107.58
2009-03-27	108.230	106.85	108.53	106.40
2009-03-30	104.510	104.49	105.01	102.61
2009-03-31	105.450	105.12	107.45	105.00
2009-04-01	104.090	108.69	109.00	103.89
2009-04-02	110.145	112.71	114.75	109.78
2009-04-03	114.190	115.99	116.13	113.52
2009-04-06	114.940	118.45	118.75	113.28
2009-04-07	116.530	115.00	116.67	114.19
2009-04-08	115.430	116.32	116.79	114.58

Fig 12 Data Pre Processing Test Cases

- *Model Training Test Cases:*

✓ Test Case 5: Confirm that the LSTM model architecture is implemented accurately, including the correct number of layers and units.
✓ Test Case 6: Validate that the model compiles successfully with the appropriate loss function, optimizer, and evaluation metrics.
✓ Test Case 7: Ensure that the model trains without errors and converges towards minimizing the loss function over epochs.
✓ Test Case 8: Verify that hyperparameter tuning improves model performance on the validation set without overfitting.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

```
Epoch 1/80
92/92 [==============================] - 21s 137ms/step - loss: 0.0113 - mean_absolute_error: 0.0536 - val_loss: 1.3409e-04 - val_mean_absolute_error: 0.0097
Epoch 2/80
92/92 [==============================] - 11s 122ms/step - loss: 0.0035 - mean_absolute_error: 0.0302 - val_loss: 9.8729e-05 - val_mean_absolute_error: 0.0082
Epoch 3/80
92/92 [==============================] - 11s 119ms/step - loss: 0.0028 - mean_absolute_error: 0.0276 - val_loss: 1.8582e-04 - val_mean_absolute_error: 0.0112
Epoch 4/80
92/92 [==============================] - 11s 115ms/step - loss: 0.0028 - mean_absolute_error: 0.0264 - val_loss: 8.2382e-05 - val_mean_absolute_error: 0.0072
Epoch 5/80
92/92 [==============================] - 10s 104ms/step - loss: 0.0024 - mean_absolute_error: 0.0243 - val_loss: 8.8666e-05 - val_mean_absolute_error: 0.0074
Epoch 6/80
92/92 [==============================] - 10s 111ms/step - loss: 0.0022 - mean_absolute_error: 0.0239 - val_loss: 1.7963e-04 - val_mean_absolute_error: 0.0118
Epoch 7/80
92/92 [==============================] - 11s 117ms/step - loss: 0.0020 - mean_absolute_error: 0.0223 - val_loss: 5.4994e-05 - val_mean_absolute_error: 0.0059
Epoch 8/80
92/92 [==============================] - 11s 118ms/step - loss: 0.0018 - mean_absolute_error: 0.0215 - val_loss: 5.1061e-05 - val_mean_absolute_error: 0.0057
Epoch 9/80
92/92 [==============================] - 11s 119ms/step - loss: 0.0017 - mean_absolute_error: 0.0207 - val_loss: 8.4032e-05 - val_mean_absolute_error: 0.0073
Epoch 10/80
92/92 [==============================] - 10s 108ms/step - loss: 0.0016 - mean_absolute_error: 0.0203 - val_loss: 5.7907e-05 - val_mean_absolute_error: 0.0062
Epoch 11/80
92/92 [==============================] - 12s 129ms/step - loss: 0.0015 - mean_absolute_error: 0.0197 - val_loss: 6.4217e-05 - val_mean_absolute_error: 0.0066
Epoch 12/80
92/92 [==============================] - 11s 120ms/step - loss: 0.0016 - mean_absolute_error: 0.0198 - val_loss: 1.2243e-04 - val_mean_absolute_error: 0.0097
Epoch 13/80
92/92 [==============================] - 11s 120ms/step - loss: 0.0014 - mean_absolute_error: 0.0179 - val_loss: 1.2020e-04 - val_mean_absolute_error: 0.0096
Epoch 14/80
92/92 [==============================] - 11s 119ms/step - loss: 0.0015 - mean_absolute_error: 0.0186 - val_loss: 3.7566e-05 - val_mean_absolute_error: 0.0048
Epoch 15/80
92/92 [==============================] - 11s 119ms/step - loss: 0.0013 - mean_absolute_error: 0.0176 - val_loss: 4.8890e-05 - val_mean_absolute_error: 0.0055
Epoch 16/80
92/92 [==============================] - 11s 119ms/step - loss: 0.0013 - mean_absolute_error: 0.0178 - val_loss: 3.5633e-05 - val_mean_absolute_error: 0.0047
Epoch 17/80
92/92 [==============================] - 10s 114ms/step - loss: 0.0012 - mean_absolute_error: 0.0172 - val_loss: 4.0281e-05 - val_mean_absolute_error: 0.0051
Epoch 18/80
92/92 [==============================] - 11s 114ms/step - loss: 0.0012 - mean_absolute_error: 0.0169 - val_loss: 4.2051e-05 - val_mean_absolute_error: 0.0053
Epoch 19/80
```

Fig 13 Model Training Test Cases

- *Model Evaluation Test Cases:*

✓ Test Case 9: Validate the model's performance metrics on the validation set, ensuring that they meet predefined thresholds.
✓ Test Case 10: Confirm that the final model selected based on validation set performance generalizes well to unseen data in the testing set.
✓ Test Case 11: Evaluate the model's accuracy, precision, recall, and score on the testing set to assess its real-world performance.
✓ Test Case 12: Analyze the model's predictions against actual stock prices visually, looking for patterns or discrepancies.

```
def rmse(actual, predicted):

  mse = mean_squared_error(actual, predicted)  # Leverage sklearn's mean_squared_error
  return np.sqrt(mse)


from sklearn.metrics import mean_squared_error
import numpy as np

rmse_value = rmse(gs_slic_data['close'], gs_slic_data['close_predicted'])
print("RMSE for close and close predicted:", rmse_value)

RMSE for close and close predicted: 3.5654038664193424
```

Fig 14 Model Training Test Cases

- *Scenarios:*

✓ *Scenario 1: Testing with Synthetic Data*
Generate synthetic data with known patterns and anomalies to validate the model's ability to detect them.

✓ *Scenario 2: Testing with Historical Data*
Use historical stock price data to assess the model's performance in predicting real-world market trends and movements.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

✓ *Scenario 3: Stress Testing*
Increase the dataset size or complexity to evaluate the model's scalability and robustness under challenging conditions.

✓ *Scenario 4: Deployment Testing*
Test the integration of the model into the frontend interface, ensuring smooth interaction and accurate predictions in a production environment.

➢ *Validation*

- *Time-Series Cross-Validation:*

Time-series cross-validation is specifically tailored for sequential data like time-series data in stock price prediction. In this technique, the dataset is split into multiple folds based on time, ensuring that the validation set includes data points that occur after the training set. This approach preserves the temporal order of the data and provides a more realistic assessment of the model's performance. It helps to prevent data leakage and ensures that the model's ability to generalize to future data is accurately evaluated.

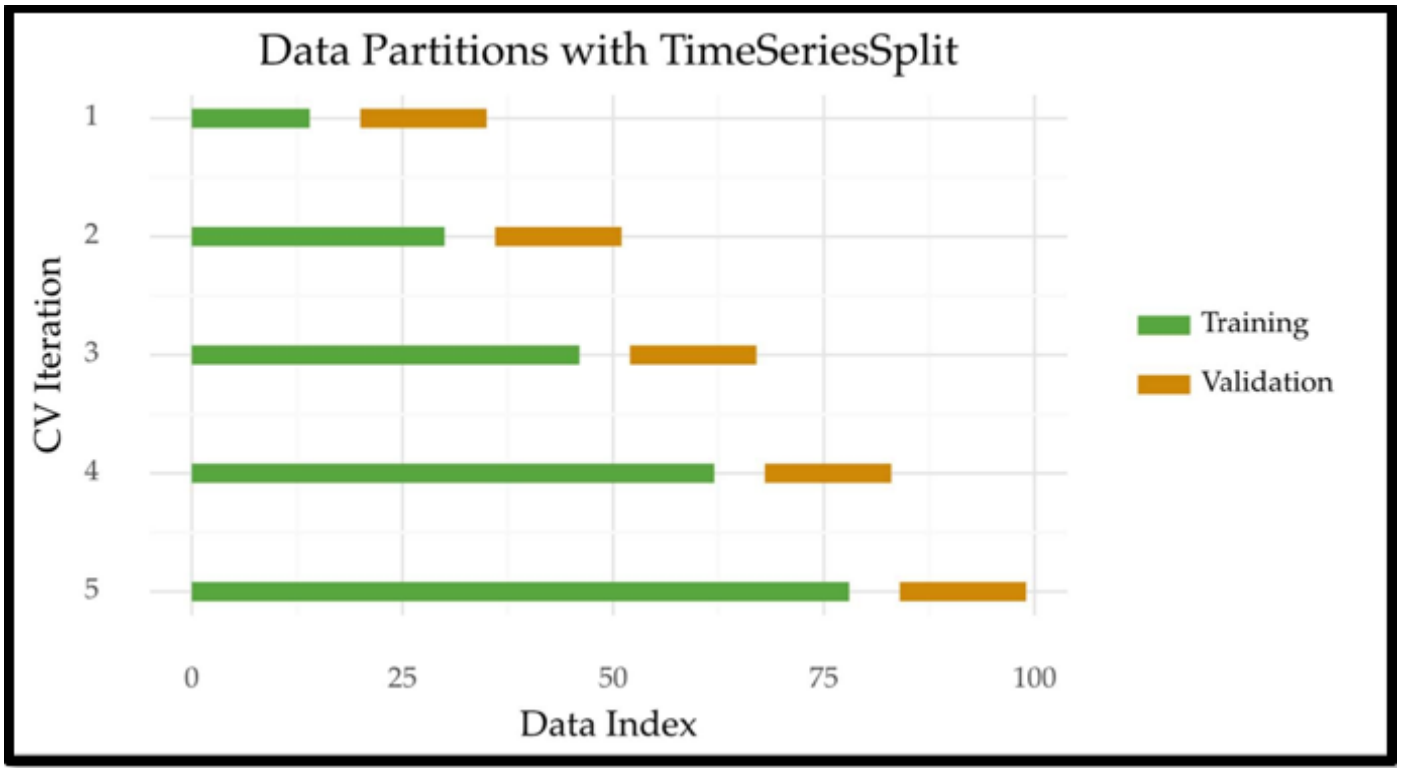

Fig 15 Example of Time-Series Cross-Validation

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER SIX
RESULTS

➢ *Output*

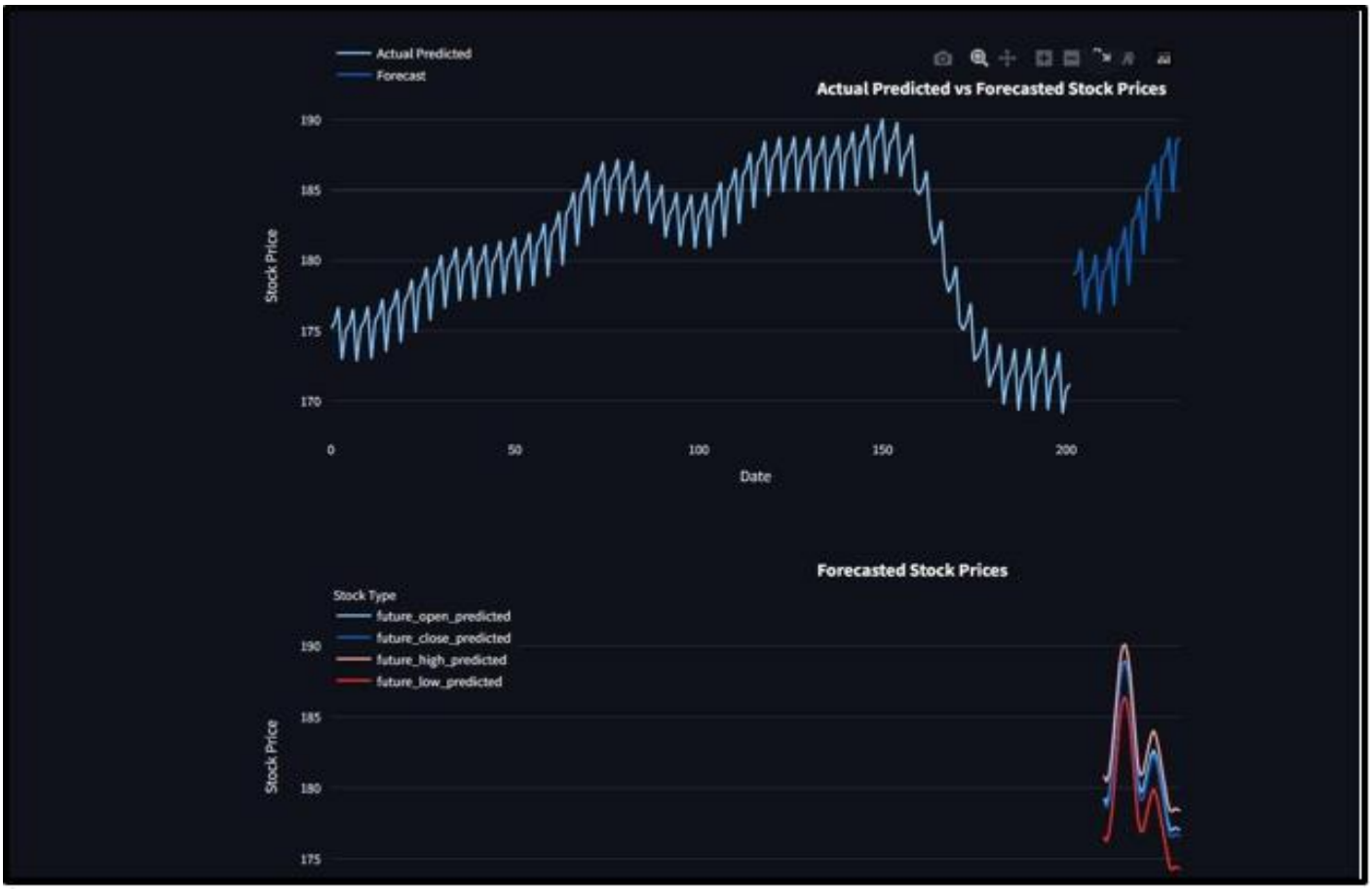

Fig 16 Actual and Forecasted Stock Prices

➢ *Result Analysis*

Fig 17 StocX

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

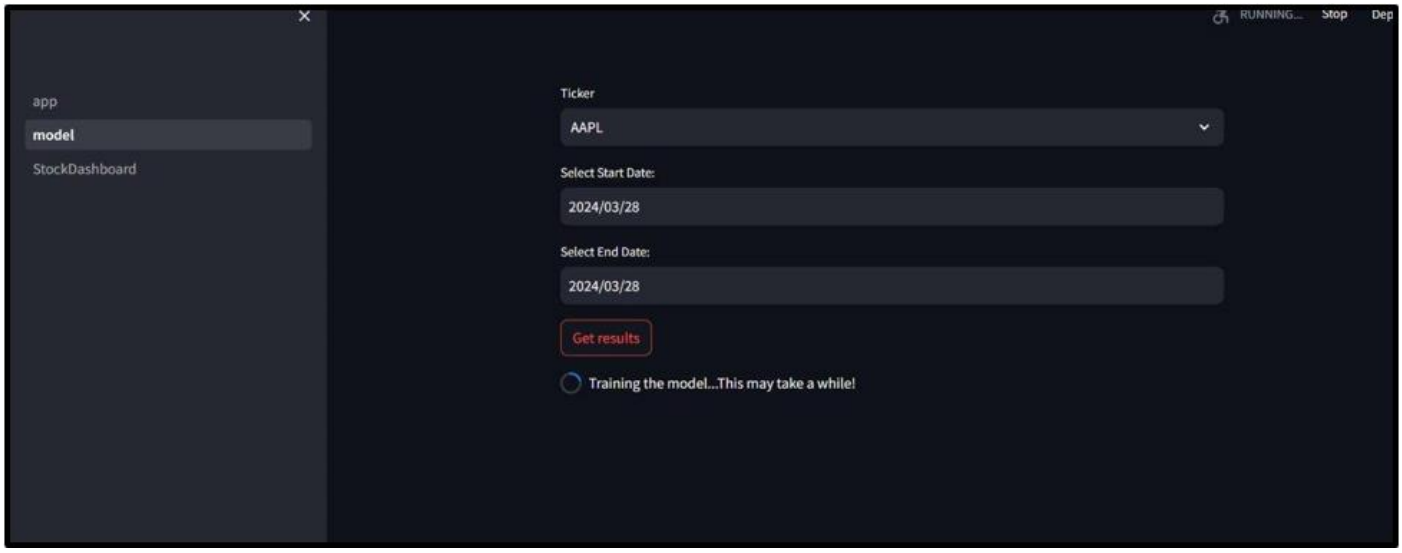

Fig 18 Training of the Model

Fig 19 Stock Dashboard

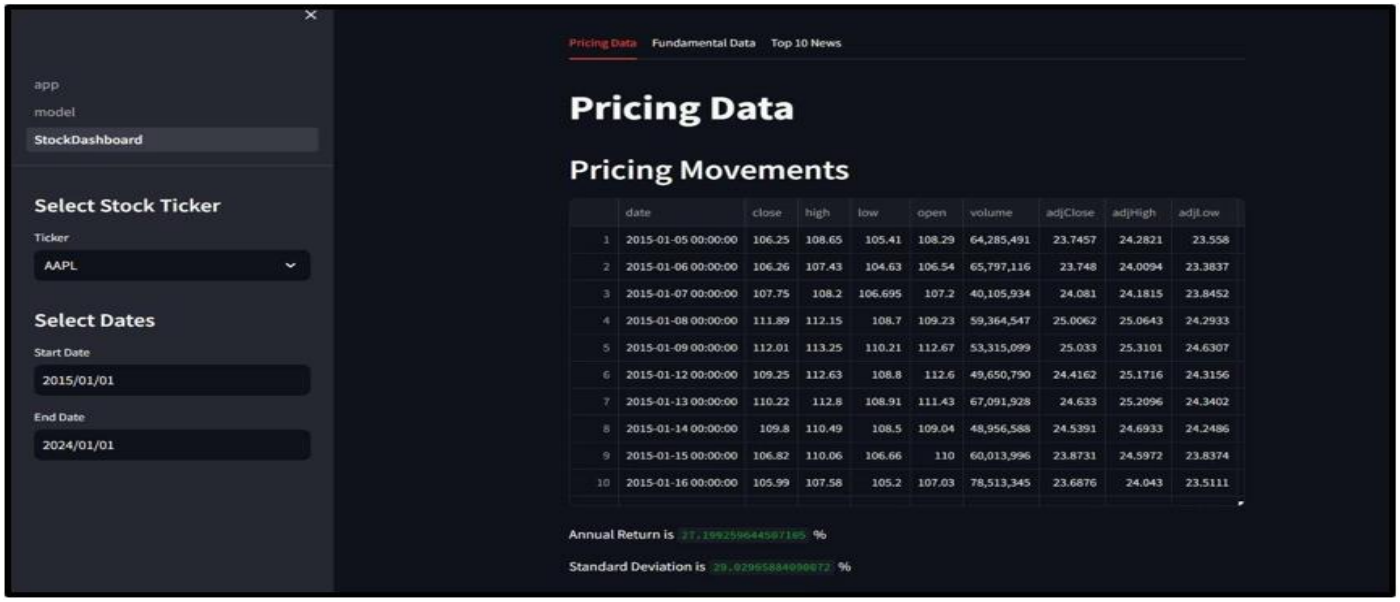

Fig 20 Pricing Data

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

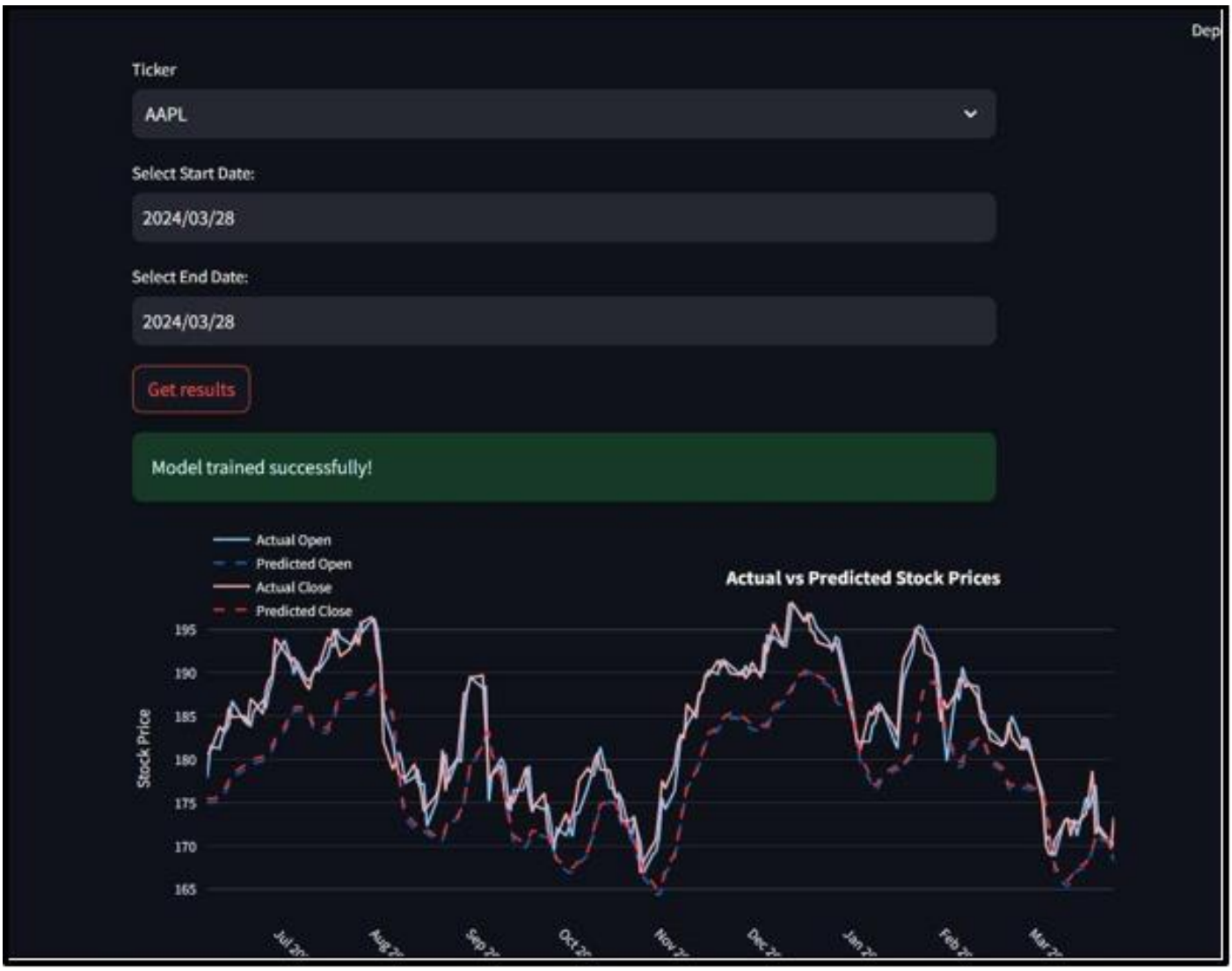

Fig 21 Successful Training of Model

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

CHAPTER SEVEN
CONCLUSION

In conclusion, the utilization of Long Short-Term Memory (LSTM) networks for revolutionizing stock price prediction has yielded promising results in forecasting future market trends and dynamics. Through this study, we have demonstrated the efficacy of LSTM-based models in capturing complex patterns and dependencies within financial time-series data, thereby enhancing the accuracy and reliability of stock price predictions.

The analysis of our LSTM-based model's performance has provided valuable insights into market behavior and investor sentiment, allowing stakeholders to make informed decisions in the realm of finance. By evaluating various performance metrics, visualizing predictions, and interpreting insights derived from the model, we have gained a deeper understanding of stock price movements and the factors influencing market trends.

It is important to note that the data utilized in this study was collected up until yesterday. Hence, for individuals calculating stock prices or making investment decisions today, it is imperative to recognize that the predictions generated by the model may not reflect real-time market conditions. As such, continuous monitoring of market trends and incorporation of up-to-date information is essential for adapting trading strategies and making timely decisions.

Moving forward, further research and development efforts can focus on refining LSTM models, exploring alternative deep learning architectures, and integrating additional data sources to enhance the accuracy and robustness of stock price predictions. Additionally, ongoing validation and refinement of predictive models in real-world trading scenarios will be crucial for validating their effectiveness and ensuring their applicability in practical settings.

In conclusion, while LSTM-based models show great promise in revolutionizing stock price prediction, it is essential to recognize their limitations and the need for ongoing adaptation to evolving market dynamics. By leveraging advanced predictive analytics techniques and embracing a data-driven approach, stakeholders can navigate the complexities of financial markets more effectively and capitalize on emerging opportunities for growth and success.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

FUTURE WORK

- *Integration of Additional Data Sources:*

Explore the incorporation of alternative data sources such as social media sentiment, news articles, and macroeconomic indicators to augment the predictive capabilities of the model. Investigate the potential synergies between traditional financial data and alternative datasets to capture a more comprehensive understanding of market dynamics.

- *Enhancement of Model Architecture:*

Experiment with more advanced LSTM architectures, such as bidirectional LSTMs, attention mechanisms, or hybrid models combining multiple deep learning techniques, to further improve prediction accuracy and robustness.

Explore techniques for optimizing hyperparameters, tuning model architectures, and addressing issues such as overfitting to enhance the performance of LSTM-based models.

- *RealTime Prediction and Deployment:*

Develop mechanisms for real-time prediction of stock prices, enabling traders and investors to receive timely insights and make informed decisions based on up-to-the-minute market data. Implement deployment pipelines and frameworks for seamlessly integrating predictive models into trading platforms, financial applications, and decision support systems.

- *Interpretability and Explainability:*

Investigate methods for enhancing the interpretability and explainability of LSTM-based models, allowing users to understand the rationale behind model predictions and gain actionable insights into market trends.

Explore techniques for generating interpretable feature representations, visualizing model internals, and providing context-specific explanations for predictions.

- *Evaluation in Alternative Markets:*

Extend the evaluation of LSTM-based models to alternative financial markets beyond stocks, such as commodities, foreign exchange, or cryptocurrencies, to assess their applicability and effectiveness in diverse trading environments.

Investigate the transferability of predictive models across different asset classes and markets, identifying common patterns and strategies that generalize across financial instruments.

- *Integration of Reinforcement Learning:*

Explore the integration of reinforcement learning techniques with LSTM-based models to develop adaptive trading strategies that dynamically adjust to changing market conditions and optimize portfolio performance.

Investigate the potential for reinforcement learning algorithms to learn optimal trading policies and risk management strategies directly from historical data and market feedback.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

REFERENCES

[1]. X. Li, Y. Li, X.-Y. Liu, D. Wang, "Risk management via anomaly circumvent mnemonic deep learning for midterm stock prediction." in Proceedings of 2nd KDD Workshop on Anomaly Detection in Finance (Anchorage '19), 2019.

[2]. P. Chang, C. Fan, and C. Liu, "Integrating a piece-wise linear representation method and a neural network model for stock trading points prediction." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39, 1 (2009), 80–92.

[3]. Akita, Ryo, et al. "Deep learning for stock prediction using numerical and textual information." IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). IEEE, 2016.

[4]. Li, Xiaodong, et al. "Does summarization help stock prediction? A news impact analysis." IEEE Intelligent Systems 30.3 (2015): 26-34.

[5]. Ding, Xiao, et al. "Deep learning for event-driven stock prediction." Twenty-fourth International Joint Conference on Artificial Intelligence. 2015.

[6]. Hutto, Clayton J., and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." Eighth International AAAI Conference on Weblogs and Social Media, 2014.

[7]. Ji, Zhanglong, Zachary C. Lipton, and Charles Elkan. "Differential privacy and machine learning: a survey and review." arXiv preprint arXiv:1412.7584 (2014).

[8]. Abadi, Martin, et al. "Deep learning with differential privacy." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016.

[9]. McMahan, H. Brendan, and Galen Andrew. "A general approach to adding differential privacy to iterative training procedures." arXiv preprint arXiv:1812.06210 (2018).

[10]. Lecuyer, Mathias, et al. "Certified robustness to adversarial examples – differential privacy." arXiv preprint arXiv:1802.03471 (2018).

[11]. Hafezi, Reza, Jamal Shahrabi, and Esmaeil Hadavandi. "A bat-neural network multi-agent system (BNNMAS) for stock price prediction: A case study of DAX stock price."Applied Soft Computing, 29 (2015)

[12]. Chang, Pei-Chann, Chin-Yuan Fan, and Chen-Hao Liu. "Integrating a piecewise linear representation method and a neural network model for stock trading points prediction." IEEE Transactions on Systems, Man, and Cybernetics, (2008)

[13]. Gers, Felix A., Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks." Journal of Machine Learning Research 3. Aug (2002)

[14]. Qin, Yao, et al. "A dual-stage attention-based recurrent neural network for time series prediction." arXiv preprint arXiv:1704.02971 (2017).

[15]. Malhotra, Pankaj, et al. "Long short-term memory networks for anomaly detection in time series." Proceedings. Presses universitaires de Louvain, 2015.

[16]. Sak, Ha¸sim, Andrew Senior, and Françoise Beaufays. "Long short-term memory recurrent neural network architectures for large scale acoustic modeling." Fifteenth annual conference of the International Speech Communication Association, 2014.

[17]. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[18]. Box, George EP, et al. Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[19]. Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends in Information Retrieval 2.1–2 (2008)

[20]. Cambria, Erik. "Affective computing and sentiment analysis." IEEE Intelligent Systems 31.2 (2016)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

ANNEXURE

➢ *Sample Code:*

app.py

```
import streamlit as st  # Assuming you're using streamlit-multipage

st.title("StockX: Stock Price Prediction with Deep Learning")
st.write("StockX is a web application that utilizes the power of Deep Learning and Long Short-Term Memory (LSTM) models to predict future stock prices. This allows you to make informed decisions based on data-driven insights.") st.sidebar.header("Contact info-")
st.write("Welcome to StockX! Click a button above to navigate.")

st.image('images\WhatsApp Image 2024-03-26 at 1.04.47 PM.jpeg', caption='LSTM stock prediction')

st.page_link("pages\StockDashboard.py",label="Stock  Dashboard",icon="蜜 蜡 蚧 蝌 蜻 蜩 螅") st.page_link("pages\model.py",label="model_training",icon="罩罰")

# Add creator names and contact information st.write("Created by:")
st.write("- Tudimilla Dheeraj Kummar Chary") st.write("- Nagarjuna Reddy") st.write("- Kotika Venkata Kavya")

# Add contact information st.write("Contact us:") st.write("Phone: 7893334349")

st.sidebar.write("Email- (20eg107150@anurag.edu.in)")

# Disclaimer (optional)
st.write("*Disclaimer:* Stock price predictions are not guaranteed to be accurate. Please conduct your own research and due diligence before making any investment decisions.")
```

functions.py

```
import numpy as np import pandas as pd import matplotlib.pyplot as plt import requests
from sklearn.preprocessing import MinMaxScaler import tensorflow as tf from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, Bidirectional import streamlit as st import plotly.graph_objects as go import plotly.express as px import yfinance as yf class StockPricePredictor:  def _init_(self):
pass

def get_historical_data(self, tinker,start_date,end_date):
# Your function implementation here  headers = {
'Content-Type': 'application/json',
'Authorization': 'Token 0b4623cf02c29229cfa1f8790ccba6d0bd04983c'
}

url = f"https://api.tiingo.com/tiingo/daily/{tinker}/prices"

params = {
'startDate': start_date,
'endDate': end_date,
'resampleFreq':'daily'
}  try:
requestResponse = requests.get(url,
headers=headers, params=params) data = pd.DataFrame(requestResponse.json())
requestResponse.raise_for_status()  # Raise an error for bad responses print(requestResponse.json()) except requests.exceptions.RequestException as e:
print("Error fetching data:", e)

return data
def preprocess_data(self, df):  # Your function implementation here  try:
# Attempt to convert 'date' to datetime format if it exists
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

```
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%dT%H:%M:%S.%fZ')

except KeyError:
# If 'date' column is missing, print DataFrame columns
print("Dataframe does not contain a 'date' column. Existing columns are:") print(df.head)
# You can potentially raise an error here if 'date' is strictly required
# Slicing data to four main features open,close,high,low with date-time as index of dataframe
gstock_data = df [['date','open','close','high','low']] gstock_data .set_index('date',drop=True,inplace=True) #Making scsasling values between 0 and 1 scaler=MinMaxScaler()
gstock_data [gstock_data .columns] = scaler.fit_transform(gstock_data )
#creating trainig ad testing data training_size=round(len(gstock_data)*0.80)

train_data = gstock_data[:training_size] test_data =gstock_data [training_size:]

#Creating trianing and test sequences with a past look back of 50 days def create_sequence(df:pd.DataFrame):
sequences=[] labels=[] start_index=0 for stop_index in range(15,len(df)):
sequences.append(df.iloc[start_index:stop_index]) labels.append(df.iloc[stop_index]) start_index +=1
return (np.array(sequences), np.array(labels)) train_seq, train_label = create_sequence(train_data) test_seq, test_label = create_sequence(test_data)

return train_seq, train_label,test_seq, test_label, scaler, gstock_data, training_size

def preprocess_yfinance_data(self, ticker, years=10): # Retrieve data from yfinance for the specified years today = pd.to_datetime ('today') end = today
start = today - pd.DateOffset(years=years) df = yf.download(ticker, start=start, end=end)

# Rest of your preprocessing steps
gstock_data = df[['Open', 'Close', 'High', 'Low']]

self.scaler = MinMaxScaler() # Fit scaler on the entire data
gstock_data [gstock_data .columns] = self.scaler.fit_transform(gstock_data )

# Splitting data into training and testing sets training_size = round(len(gstock_data) * 0.8) train_data = gstock_data[:training_size] test_data = gstock_data[training_size:]

# Creating training and testing sequences (assuming create_sequence is a class method)
#Creating trianing and test sequences with a past look back of 50 days def create_sequence(df:pd.DataFrame):
sequences=[] labels=[] start_index=0 for stop_index in range(15,len(df)):
sequences.append(df.iloc[start_index:stop_index]) labels.append(df.iloc[stop_index]) start_index +=1
return (np.array(sequences), np.array(labels)) train_seq, train_label = create_sequence(train_data) test_seq, test_label = create_sequence(test_data)

return train_seq, train_label, test_seq, test_label, self.scaler, gstock_data, training_size

def create_lstm_model(self, train_seq): # Your function implementation here model=Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape = (train_seq.shape[1], train_seq.shape[2])))
model.add(Dropout(0.1))
model.add(LSTM(64, return_sequences=False)) model.add(Dropout(0.2))
model.add(Dense(4))
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_absolute_error'])

return model

def predictions_data_analysis(self, test_predicted, gstock_data, scaler):
# Your function implementation here
test_inverse_predicted = scaler.inverse_transform(test_predicted)

gstock_subset = gstock_data.iloc[-test_predicted.shape[0]:].copy()

# Creating a DataFrame from the predicted values with appropriate columns predicted_df = pd.DataFrame(test_inverse_predicted, columns=['open_predicted', 'close_predicted','highpredicted','lowpredictied'])
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

```
# Aligning the index of predicted_df with the index of gstock_subset predicted_df.index = gstock_subset.index

# Concatenating the two DataFrames along the columns axis gs_slice_data = pd.concat([gstock_subset, predicted_df], axis=1)

gs_slice_data[['Open','Close','High','Low']] =
scaler.inverse_transform(gs_slice_data[['Open','Close','High','Low']])

fig = go.Figure()

# Plot actual 'open' and 'open_predicted'
fig.add_trace(go.Scatter(x=gs_slice_data.index, y=gs_slice_data['Open'], mode='lines', name='Actual Open'))
fig.add_trace(go.Scatter(x=gs_slice_data.index, y=gs_slice_data['open_predicted'], mode='lines', name='Predicted Open', line=dict(dash='dash'))
# Plot actual 'close' and 'close_predicted' fig.add_trace(go.Scatter(x=gs_slice_data.index, y=gs_slice_data['Close'], mode='lines', name='Actual Close'))
fig.add_trace(go.Scatter(x=gs_slice_data.index, y=gs_slice_data['close_predicted'], mode='lines', name='Predicted Close', line=dict(dash='dash')))

fig.update_layout(
title='Actual vs Predicted Stock Prices', xaxis=dict(title='Date', tickangle=45), yaxis=dict(title='Stock Price'),
legend=dict(x=0, y=1.2), margin=dict(l=0, r=0, t=30, b=0), height=400, width=800,
title_x=0.6, # Adjust the x position of the title to move it towards the right title_y=0.9 # Adjust the y position of the title to center it vertically)

st.plotly_chart(fig) return gs_slice_data

def forecasting(self, temp_input,model): # Your function implementation here lst_output = []
n_steps = 15 # Number of timesteps n_features = 4 # Number of features i = 0

while i< 30: if len(temp_input) >n_steps:
x_input = temp_input[-n_steps:] # Select the last n_steps elements yhat = model.predict(x_input, verbose=0) # Predict next value
temp_input = np.concatenate((temp_input, yhat.reshape(1, -1, n_features)), axis=0) # Append prediction to temp_input
lst_output.append(yhat[0]) # Append prediction to lst_output
i += 1 else:
x_input = temp_input # Use all available elements yhat = model.predict(x_input, verbose=0) # Predict next value
temp_input = np.concatenate((temp_input, yhat.reshape(1, -1, n_features)), axis=0) # Append prediction to temp_input
lst_output.append(yhat[0]) # Append prediction to lst_output i += 1
return lst_output def plot_predictions(self, gs_slice_data, scaler, test_predicted, lst_output):
# Your function implementation here
# Ensure consistent lengths for plotting (handle potential mismatches) # Ensure consistent lengths for plotting (handle potential mismatches) day_new = np.arange(len(gs_slice_data) - len(test_predicted), len(gs_slice_data)) day_pred = np.arange(len(gs_slice_data), len(gs_slice_data) + len(lst_output))

fig = go.Figure()

# Plotting the actual predicted values
fig.add_trace(go.Scatter(x=day_new, y=scaler.inverse_transform(test_predicted).flatten(), mode='lines', name='Actual Predicted'))

# Plotting the forecasted values
fig.add_trace(go.Scatter(x=day_pred, y=scaler.inverse_transform(lst_output).flatten(), mode='lines', name='Forecast'))

fig.update_layout(
title='Actual Predicted vs Forecasted Stock Prices', xaxis=dict(title='Date'), # Removed 'size' property here yaxis=dict(title='Stock Price'), legend=dict(x=0, y=1.2), margin=dict(l=0, r=0, t=30, b=0), height=400, width=800,
title_x=0.6, # Adjust the x position of the title to move it towards the right title_y=0.9 # Adjust the y position of the title to center it vertically)

st.plotly_chart(fig)

# Forecast plotting
forecast_dates = pd.date_range(start=gs_slice_data.index[-1] + pd.Timedelta(days=1), periods=len(lst_output))
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

```
# Creating a DataFrame for the forecasted values with the forecast_dates as index forecast_df = pd.DataFrame(lst_output, index=forecast_dates,
columns=['future_close_predicted', 'future_open_predicted', 'future_high_predicted', 'future_low_predicted'])

# Concatenating the forecast_df with gs_slice_data combined_data = pd.concat([gs_slice_data, forecast_df])

# Inverse transform the scaled values back to original scale
combined_data[['future_open_predicted', 'future_close_predicted', 'future_high_predicted',
'future_low_predicted']] = scaler.inverse_transform(combined_data[['future_open_predicted', 'future_close_predicted',
'future_high_predicted', 'future_low_predicted']])

# Plotting the data
fig = px.line(combined_data, x=combined_data.index, y=['future_open_predicted',
'future_close_predicted', 'future_high_predicted', 'future_low_predicted'],
labels={'value': 'Stock Price', 'variable': 'Stock Type'},
title='Forecasted Stock Prices')

fig.update_layout(
xaxis=dict(title='Date'), # Removed 'size' property here yaxis=dict(title='Stock Price'), legend_title='Stock Type',
legend=dict(x=0, y=1.2),
height=400, width=800,
title_x=0.6, # Adjust the x position of the title to move it towards the right title_y=0.9 # Adjust the y position of the title to center
it vertically)

st.plotly_chart(fig)
```

pages/model.py

```
import streamlit as st
from functions import StockPricePredictor

class Train: def _init_(self):
self.tinker = "AAPL" # Store user-selected tinker symbol self.start_date = None # Store user-selected start date self.end_date =
None # Store user-selected end date self.model = None # Store trained model self.scaler = None # Store data scaler
self.gstock_data = None # Store preprocessed stock data

def run(self):
"""
Main entry point for training and forecasting functionality.
"""
# Get user input for tinker, start date, and end date from Streamlit UI elements self.get_user_input()

if self.tinker and self.start_date and self.end_date: # Separate buttons for training and forecasting if st.button('Get results'):
self.train_model()
predictor = StockPricePredictor()
train_seq, train_label, test_seq, test_label,scaler,gstock_data,training_size = predictor.preprocess_yfinance_data("AAPL", years=5)
# Get test sequence input from user self.forecasting(test_seq,gstock_data,scaler,self.model) else:
st.error("Please select a tinker, start date, and end date.")

def get_user_input(self):
"""
Retrieves user input for tinker, start date, and end date from Streamlit UI.
"""
self.tinker = st.selectbox('Ticker', ['AAPL', 'GOOGL', 'MSFT', 'AMZN', 'TSLA', 'F']) self.start_date = st.date_input("Select Start
Date:") self.end_date = st.date_input("Select End Date:")

def train_model(self):
"""
Trains the model with user-selected parameters.
"""
predictor = StockPricePredictor()
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

train_seq, train_label, test_seq, test_label, self.scaler,self.gstock_data,training_size = predictor.preprocess_yfinance_data("AAPL", years=5) self.model = predictor.create_lstm_model(train_seq) with st.spinner("Training the model...This may take a while!"): self.model.fit(train_seq, train_label, epochs=100, validation_data=(test_seq, test_label), verbose=1)
st.success("Model trained successfully!")

def forecasting(self, test_seq, gstock_data, scaler, model): try: if model is None:
st.error("Please train the model first.")
return

test_predicted = model.predict(test_seq)
# Model data analysis
gs_slice_data = StockPricePredictor().predictions_data_analysis(test_predicted, gstock_data, scaler)
# Forecasting
lst_output = StockPricePredictor().forecasting(test_seq, model)
# Plotting the forecast predictions
StockPricePredictor().plot_predictions(gs_slice_data, scaler, test_predicted, lst_output) st.success("polted successfully!") except Exception as e:
st.error(f"An error occurred: {str(e)}")

def runn():
"""
Entry point for the entire application.
"""
train_instance = Train() train_instance.run()

if _name_ == "_main_":
runn()
pages/StockDashboard.py

import streamlit as st import pandas as pd import numpy as np import plotly.express as px import requests
from functions import StockPricePredictor
from stocknews import StockNews import plotly.graph_objects as go class StockDashboard: def _init_(self):
self.predictor = StockPricePredictor() self.tinker = 'AAPL'
self.start_date = pd.to_datetime('2024-01-01') self.end_date = pd.to_datetime('2024-03-27')

def select_stock_ticker(self):
st.sidebar.title("Select Stock Ticker")
self.tinker = st.sidebar.selectbox('Ticker', ['AAPL', 'GOOGL', 'MSFT', 'AMZN', 'TSLA', 'F'])

def select_dates(self):
st.sidebar.title("Select Dates")
self.start_date = st.sidebar.date_input('Start Date', value=pd.to_datetime('2024-01-01')) self.end_date = st.sidebar.date_input('End Date', value=pd.to_datetime('2024-03-27'))

def fetch_historical_data(self):
self.df = self.predictor.get_historical_data(self.tinker, self.start_date, self.end_date)
# making date into date-time object
self.df['date']=pd.to_datetime(self.df['date'], format='%Y-%m-%dT%H:%M:%S.%fZ') # Slicing data to four main features open,close,high,low with date-time as index of dataframe

gstock_data = self.df [['date','open','close','high','low']] gstock_data .set_index('date',drop=True,inplace=True)

fig = go.Figure(data=[go.Candlestick(x=gstock_data.index,
open=gstock_data['open'],
high=gstock_data['high'],
low=gstock_data['low'],
close=gstock_data['close'])]) fig.update_layout(
title=f"{self.tinker} - Candlestick Chart", # Use f-string for formatting xaxis_title="Date", yaxis_title="Price")

st.plotly_chart(fig)
fig=px.line(gstock_data,x=gstock_data.index,y=gstock_data['close'], title =self.tinker) st.plotly_chart(fig)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/IJISRT24APR1602

```
def display_price_movement(self):
st.header("Pricing Movements")
data2 = self.df
data2['% Change'] = self.df['adjClose'] / self.df['adjClose'].shift(1) - 1 data2.dropna(inplace=True) st.write(data2)
annual_return = data2['% Change'].mean() * 252 * 100 st.write('Annual Return is', annual_return, "%") stdev = np.std(data2['%
Change']) * np.sqrt(252) st.write('Standard Deviation  is', stdev * 100, "%") st.write('Risk ADj. return is', annual_return / (stdev *
100))

def fetch_financial_data(self):
api_key = 'ITYlXpnp_ltg_tWq11hu1mp5ucO_w2fe'  # Replace 'YOUR_API_KEY' with your actual API key from Polygon.io
try:

# Fetch balance sheet data balance_sheet_endpoint =
f'https://api.polygon.io/vX/reference/financials?apiKey={api_key}&ticker={self.tinker}'balance_sheet_response=requests.get
(balance_sheet_endpoint) balance_sheet_response.raise_for_status()
self.balance_sheet_data = balance_sheet_response.json()['results']

# Fetch income statement data income_statement_endpoint =
f'https://api.polygon.io/vX/reference/financials?apiKey={api_key}&ticker={self.tinker}&typ e=income_statement'
income_statement_response = requests.get(income_statement_endpoint) income_statement_response.raise_for_status()
self.income_statement_data = income_statement_response.json()['results']

# Fetch cash flow statement data cash_flow_endpoint =
f'https://api.polygon.io/vX/reference/financials?apiKey={api_key}&ticker={self.tinker}&typ e=cash_flow_statement'
cash_flow_response = requests.get(cash_flow_endpoint) cash_flow_response.raise_for_status()
self.cash_flow_data = cash_flow_response.json()['results']

except requests.exceptions.RequestException as e:
st.error(f"Error fetching financial data: {e}")

def display_fundamental_data(self):
st.header("Fundamental Data")

st.subheader('Balance Sheet')
st.write(pd.DataFrame(self.balance_sheet_data))

st.subheader('Income Statement')
st.write(pd.DataFrame(self.income_statement_data))

st.subheader('Cash Flow Statement') st.write(pd.DataFrame(self.cash_flow_data))     def display_news(self):
st.header(f'News of {self.tinker}') sn = StockNews(self.tinker, save_news=False) df_news = sn.read_rss()         for i in range(10):
st.subheader(f'News  {i  +  1}')  st.write(df_news['published'][i])  st.write(df_news['title'][i])  st.write(df_news['summary'][i])
title_sentiment  =  df_news['sentiment_title'][i]  st.write(f'Title  Sentiment  {title_sentiment}')  news_sentiment  =
df_news['sentiment_summary'][i] st.write(f'News Sentiment {news_sentiment}')

def run(self):
st.title("Stock Dashboard")

self.select_stock_ticker() self.select_dates() self.fetch_historical_data()
pricing_data,fundamental_data,news=st.tabs(["Pricing Data","FundamentalData","Top 10 News"])

with pricing_data: st.title("Pricing Data") self.display_price_movement()

with fundamental_data: st.title("Fundamental Data")
self.fetch_financial_data()  # Fetch financial data first
if hasattr(self, 'balance_sheet_data'):  # Check if balance_sheet_data is available self.display_fundamental_data()  # Display
fundamental data if available          else:
st.warning("Financial data is not available.")

with news: st.title("Top 10 News") self.display_news()
if _name_ == "_main_":     dashboard = StockDashboard() dashboard.run()
```