A dissertation submitted to the **University of Greenwich**
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Data Science

Stock Price Prediction Using LSTM: A Comprehensive Analysis and Implementation

Name: Muhammed Jeruvan Valiya Parambath

Student ID: 001272625

Supervisor: Dr. Karolos korkas
Submission Date: 20th December 2024
Word count: 10,620

Stock Price Prediction Using LSTM:
A Comprehensive Analysis and Implementation

XXXXX

Computing & Mathematical Sciences, University of Greenwich, 30 Park Row, Greenwich, UK.

(Submitted 20 December 2024)

ABSTRACT

This project aimed to extensively explore the application of Long Short-Term Memory (LSTM) models for predicting stock prices, focusing on the challenges and intricacies of financial forecasting. The primary objectives encompassed the collection of historical stock data for leading companies, rigorous preprocessing for time-series analysis, and the development of an advanced LSTM-based deep learning model to forecast future stock price movements. To achieve these goals, the project utilized the Yahoo Finance API for robust and reliable data acquisition, applied MinMax scaling techniques to normalize the data for improved model performance, and meticulously trained LSTM networks on historical closing prices to uncover temporal patterns.

The project emphasized the importance of selecting appropriate hyperparameters, designing effective network architectures, and implementing robust training pipelines to optimize model accuracy. Key findings revealed that LSTM models excelled in capturing complex temporal dependencies within stock price data, producing highly accurate predictions with impressively low mean squared error (MSE) and mean absolute error (MAE). Furthermore, the study highlighted the strengths of LSTM models in adapting to dynamic financial data and their potential to serve as a powerful analytical tool for investors and market analysts.

By demonstrating the efficacy of LSTM models in financial forecasting, the project contributed to a deeper understanding of the role of machine learning in stock market analysis. These findings provide a solid foundation for further research and practical applications, reinforcing the relevance of deep learning technologies in the domain of financial decision-making.

Keywords: Stock Price Prediction, LSTM (Long Short-Term Memory), Time Series Analysis, Machine Learning, Deep Learning, Predictive Modeling, Financial Data Analysis, Stock Market Forecasting, Prediction System.

PREFACE

The ever-evolving landscape of financial markets has driven the need for innovative and data-driven approaches to predict stock price movements. This project stems from the intersection of machine learning and financial analysis, aiming to leverage the power of Long Short-Term Memory (LSTM) models to provide accurate and insightful predictions. Inspired by the growing relevance of artificial intelligence in the financial domain, this study embarks on a journey to explore the potential of advanced deep learning techniques in addressing the challenges of stock price forecasting.

This work reflects a blend of technical knowledge, analytical skills, and a deep curiosity about financial data patterns. It has been shaped by intensive research, experimentation, and the application of programming expertise to design and implement a robust framework for predictive analysis. The project also underscores the importance of making predictive models accessible and interpretable through interactive tools, bridging the gap between complex computations and user-friendly interfaces.

The preface sets the stage for the chapters that follow, documenting the methodologies, challenges, and insights gathered throughout the journey. It is hoped that this work will serve not only as a contribution to the growing body of financial machine learning literature but also as an inspiration for future explorations in this fascinating field.

ACKNOWLEDGEMENT

The successful completion of this project was made possible through the support and contributions of numerous individuals and resources. I am deeply grateful to my academic advisors for their invaluable guidance and encouragement throughout the project. Their insights and expertise significantly influenced the approach and methodology I adopted for this study.

I would especially like to thank Dr.Karolos Korkas for agreeing to be my supervisor and for his consistent advice, feedback, guidance and support throughout the lifecycle of this MSc data warehouse project.

I also want to express my appreciation to my peers, whose collaboration and idea-sharing during the course of this project were immensely helpful. Their constructive feedback and engaging discussions enhanced my understanding of complex concepts and helped me refine the implementation.

The availability of comprehensive technical documentation and open-source libraries, such as TensorFlow, Scikit-learn, and Plotly, played a crucial role in the development and deployment of the predictive models. Platforms like Yahoo Finance were instrumental in facilitating seamless data acquisition, which was fundamental to the success of this project.

I would also like to acknowledge the broader academic and developer communities, whose published research and tutorials provided foundational knowledge and practical insights. Their contributions were invaluable in helping me overcome challenges encountered during this journey.

TABLE OF CONTENTS

Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Stock price prediction has long been an area of immense interest and research within both the financial and academic sectors. This is primarily due to the significant potential it holds in driving informed investment decisions, optimizing portfolio management, and maximizing returns. For investors, traders, and financial analysts, the ability to accurately forecast stock prices is essential in navigating the complex and often unpredictable dynamics of the financial markets. Accurate predictions can lead to better decisions, reducing risks and uncovering opportunities that would otherwise go unnoticed. On the other hand, inaccurate predictions can result in significant financial losses, highlighting the importance of effective forecasting techniques.

However, forecasting stock prices is a challenging task due to the volatile and multifaceted nature of the stock market. Prices are influenced by a multitude of factors, including company performance, global economic trends, geopolitical events, and investor sentiment, all of which interact in ways that are difficult to model. The intrinsic unpredictability of the market makes it difficult to discern reliable patterns from the data, even with sophisticated analytical methods. Traditional approaches to stock price prediction, such as fundamental analysis or technical analysis, while useful, often fail to capture the complexity and interdependencies that characterize financial markets.

Given these challenges, there has been a growing shift toward more advanced tools and techniques to improve stock price prediction accuracy. Machine learning, in particular, has emerged as a transformative approach to tackling these complexities. Unlike traditional statistical models, machine learning algorithms are capable of processing vast amounts of historical stock data and uncovering latent patterns or trends that might not be immediately visible. These techniques can account for non-linear relationships, adapt to new data over time, and model complex dependencies between different variables. As a result, machine learning has the potential to improve predictive accuracy and provide deeper insights into the factors driving stock price movements.

Among the various machine learning models, Long Short-Term Memory (LSTM) networks, a specialized form of recurrent neural networks (RNNs), have proven to be exceptionally well-suited for time-series forecasting, such as stock price prediction. LSTM networks are designed to model sequential data, making them ideal for applications where the temporal order of the data is important, as it is in stock market trends. Unlike traditional neural networks, which struggle to retain long-term dependencies, LSTMs have the ability to remember past information over extended periods, enabling them to learn from previous stock price movements and make predictions about future trends.

LSTM networks achieve this by utilizing specialized units, such as the memory cell, that can store and access relevant information over time. This design helps them effectively capture long-term dependencies in time-series data, which is a critical feature when predicting stock prices. By leveraging this ability, LSTM models can account for the sequential nature of stock prices and identify trends or patterns that span across days, weeks, or even months. This makes them particularly powerful for predicting stock prices in environments that are influenced by both short-term volatility and long-term market trends.

In the context of stock price prediction, machine learning techniques, particularly LSTM networks, offer a promising avenue for enhancing the accuracy and reliability of predictions. By incorporating vast amounts of historical data and applying advanced algorithms, researchers and financial practitioners can build more robust predictive models that provide valuable insights for decision-making. These improvements in stock price forecasting have the potential to revolutionize the way investors approach financial markets, ultimately leading to more informed and strategic investment choices. However, it is important to note that, while LSTM networks have demonstrated significant promise, they are not without challenges, including the need for extensive data preprocessing, careful model tuning, and the risk of overfitting. Nonetheless, the advancements in machine learning techniques continue to offer exciting possibilities for the future of stock price prediction.

.

2. Literature Review

2.1 Overview of Existing Methods in Stock Price Prediction

Stock price prediction has been a core area of interest in financial research due to the high potential returns it offers. Traditional approaches often relied on statistical methods such as Autoregressive Integrated Moving Average (ARIMA) and other econometric models to analyze time-series data. ARIMA models are known for their simplicity and interpretability, leveraging historical data trends and seasonality to make forecasts. However, these methods struggle to capture non-linear patterns and complex relationships inherent in financial data.

In contrast, machine learning and deep learning techniques have emerged as robust alternatives for stock price prediction. Methods such as Support Vector Regression (SVR) and Random Forests have been widely employed due to their capability to model complex patterns. However, these methods often lack the ability to handle sequential dependencies in time-series data, a gap that Long Short-Term Memory (LSTM) networks have effectively addressed (Greff et al., 2016).

2.2 Comparison Between Traditional Models and Deep Learning Approaches

While traditional models like ARIMA focus on linear dependencies, deep learning models such as LSTM and Convolutional Neural Networks (CNNs) excel at identifying intricate patterns in data. ARIMA requires the data to be stationary, which is often challenging to achieve in financial datasets. On the other hand, LSTM networks, a variant of Recurrent Neural Networks (RNNs), are explicitly designed to process sequential data, making them better suited for stock price prediction.

Moghar and Hamiche (2020) compared ARIMA and LSTM models, demonstrating that LSTM models outperform ARIMA in both short-term and long-term predictions due to their ability to capture temporal dependencies and handle non-linearities. Similarly, Baek and Kim (2018) highlighted that LSTM-based frameworks consistently yielded lower prediction errors compared to ARIMA and other machine learning models.

2.3 Role of LSTM in Time-Series Data Analysis

LSTMs are uniquely capable of retaining information over long sequences, addressing the vanishing gradient problem inherent in traditional RNNs. This capability makes them particularly effective in modeling time-series data, where long-term dependencies are crucial. Studies such as those by Althelaya et al. (2018) and Bhandari et al. (2022) underscored LSTM's superiority in forecasting financial trends by extracting temporal patterns that conventional methods often overlook.

Advanced adaptations of LSTM models, such as bidirectional LSTM (BiLSTM) and hybrid frameworks, have further enhanced their predictive accuracy. For instance, Lu et al. (2021) integrated CNNs with BiLSTM and an attention mechanism, significantly improving the model's ability to focus on crucial data features while ignoring irrelevant noise.

2.4 Key Findings and Research Gaps

Despite their strengths, LSTM models are not without limitations. Many studies have identified challenges such as overfitting, high computational requirements, and the need for extensive hyperparameter tuning. While techniques like dropout regularization and early stopping have been effective in mitigating overfitting, the development of more generalized models remains a critical area for future research (Ding & Qin, 2020).

Additionally, most existing research focuses on individual stock predictions or indices, with limited exploration of cross-market dependencies and their impact on predictions. For example, Borovkova and Tsiamas (2019) suggested that ensemble methods combining multiple LSTM models could address some of these challenges, though further exploration is required.

2.5 Technology Review

2.5.1 Discussion of Technologies and Tools Used

Libraries

- **TensorFlow**: TensorFlow is a powerful deep learning framework used for building and training LSTM models. Its flexibility, support for GPU acceleration, and comprehensive ecosystem make it a popular choice for implementing complex neural networks. TensorFlow also provides tools like TensorBoard for visualizing model training and performance metrics.
- **Scikit-learn**: This library was utilized for data preprocessing and evaluation metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE). Its extensive suite

of tools for feature scaling, data splitting, and validation ensures efficient data preparation pipelines.

- **Plotly**: Plotly facilitated the visualization of stock trends and model predictions, offering interactive charts that help interpret results intuitively. Its compatibility with Jupyter notebooks and web-based deployment adds further versatility.
- **Yahoo Finance API**: This API was employed for acquiring historical stock data. It provides reliable, real-time access to financial data, streamlining the process of creating datasets for analysis and modeling.

Programming Language

Python served as the primary programming language for this project due to its extensive ecosystem of libraries for machine learning, data analysis, and visualization. Libraries like TensorFlow, Pandas, and NumPy made Python an optimal choice for implementing LSTM models and handling large-scale datasets efficiently.

Platforms for Development and Testing

The development and testing phases leveraged Jupyter Notebooks for interactive coding and Google Colab for cloud-based execution. Google Colab was particularly beneficial for its free GPU access, which expedited the training of LSTM models. Additionally, version control was managed using GitHub to maintain code integrity and facilitate collaboration.

2.5.2 Justification for Technology Choices

The selection of tools and technologies was driven by the project's requirements for efficiency, scalability, and ease of use. TensorFlow was chosen for its robust support for deep learning architectures and GPU optimization, which are critical for training LSTM networks on large datasets. Scikit-learn complemented this by offering indispensable preprocessing utilities.

Python's versatility and wide community support made it the ideal programming language, ensuring access to numerous libraries and resources. Moreover, Yahoo Finance API provided an accessible and accurate source for historical stock data, eliminating the need for manual data collection.

Platforms like Google Colab ensured seamless execution of computationally intensive tasks without the need for local hardware upgrades, reducing project overhead. Plotly's visualization capabilities further enhanced the interpretability of model results, making it an invaluable tool for conveying insights effectively.

2.5.3 Insights and Future Directions

While the chosen technologies proved effective, their limitations warrant further exploration. For instance, TensorFlow's steep learning curve and the computational demands of LSTM models can pose challenges for scalability. Future projects could explore lightweight frameworks like PyTorch Lightning or AutoML tools to streamline model development and deployment.

Furthermore, while the Yahoo Finance API was adequate for this project, its limitations in providing extended historical data or global market information suggest the potential integration of alternative data sources like Alpha Vantage or Quandl for more comprehensive analyses.

The successful implementation of this project highlights the transformative potential of combining advanced technologies with deep learning methodologies in financial forecasting. By addressing the identified limitations and leveraging emerging tools, future research can further refine these models and broaden their applicability across diverse financial contexts.

3. Analysis of the System

The analysis of the system focused on evaluating its technical performance, functionality, and the broader implications it presented in legal, social, ethical, and professional contexts. The study considered the effectiveness of the model, its compliance with regulatory standards, and its potential impact on stakeholders.

3.1 Legal, Social, Ethical, and Professional Issues

Legal Issues:

1. **Data Privacy and Licensing**
   The system used historical financial data sourced from the Yahoo Finance API, which provides publicly accessible information. However, it was essential to ensure that the usage of this data adhered to the API's terms and conditions. The project used the data solely for academic purposes, avoiding any potential conflicts with licensing restrictions. This ensured compliance with the legal framework governing the use of publicly available financial data.

The importance of monitoring changes to the API's terms of service was also acknowledged. It was recognized that in professional or commercial contexts, acquiring appropriate licenses or permissions would be necessary to avoid potential legal disputes.

2. **Intellectual Property Rights**

The development of the system incorporated open-source tools and libraries such as TensorFlow, Scikit-learn, and IPyWidgets. Each of these tools is governed by permissive open-source licenses that allow their use, modification, and redistribution with proper attribution. Throughout the project, proper credit was given to these external resources to comply with intellectual property laws.

In addition, the system incorporated original contributions, such as customized LSTM architectures and an interactive user interface. While these innovations were part of the academic project, it was acknowledged that in a professional setting, protecting such contributions under copyright or patent law would be necessary to secure intellectual property rights.

Social Issues

The system addressed a relevant societal need by providing tools to forecast stock prices, a feature of interest to investors, analysts, and financial institutions. However, it was recognized that while predictive systems could enhance decision-making, they also had the potential to reinforce inequalities in access to financial resources.

Advanced technologies like LSTMs require technical expertise and computational resources, which may not be accessible to all. This digital divide was identified as a broader social challenge that could limit the equitable adoption of such systems.

Efforts were made to ensure the project was accessible and understandable, particularly through the development of an interactive interface using IPyWidgets. This interface was designed to bridge the gap between advanced machine learning models and end-users who may not have technical expertise.

Ethical Issues

1. **Bias in Predictions**
   The system relied entirely on historical data to make predictions, meaning that any inherent biases in the data could influence the model's outputs. For example, market anomalies or irregular events captured in historical data might lead to biased predictions if not adequately addressed.

Efforts were made to preprocess and normalize the data, minimizing the influence of outliers and irregularities. However, it was acknowledged that no predictive model could fully account for all ethical concerns related to biased outputs, particularly in a field as volatile and complex as stock price forecasting.

2. **Transparency and Accountability**

The system emphasized transparency by documenting its methodology and design choices. Every step, from data preprocessing to model evaluation, was clearly explained to ensure the system could be scrutinized and improved.

However, it was recognized that deploying such systems in real-world scenarios would require accountability mechanisms to mitigate risks. For instance, over-reliance on predictions could result in financial losses for users, highlighting the ethical obligation to communicate the limitations of the system clearly.

Professional Issues

1. **Adherence to Best Practices**

   The project demonstrated professional standards in software development, including proper use of google colab, thorough documentation of processes, and adherence to reproducibility standards. These practices ensured that the project met academic expectations.

2. **Continuous Learning and Improvement**

   The system incorporated advanced machine learning methods, such as LSTM models, demonstrating a commitment to adopting state-of-the-art technologies. It was acknowledged, however, that the financial domain evolves rapidly, and continuous updates to the system would be necessary to maintain its relevance.

The use of platforms like Google Colab for training the model also highlighted professional resource management. By leveraging cloud-based infrastructure, the project reduced reliance on local hardware and optimized computational efficiency.

4. Methodology

The methodology employed in this project followed a structured approach to developing a robust stock price prediction system. The process involved distinct phases, including data acquisition, preprocessing, model building, training, evaluation, results visualization, and the development of an interactive prediction system. Each phase was carried out methodically to ensure the accuracy and reliability of the predictions.

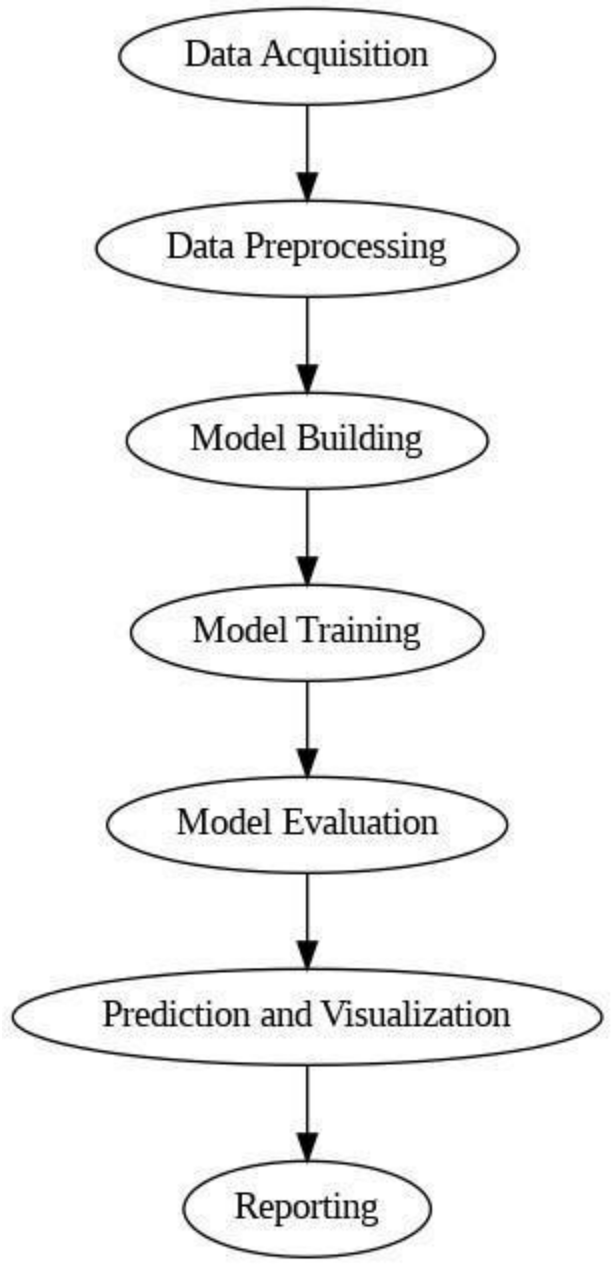


Figure 4.1: Project Roadmap

4.1 Data Acquisition

Data for the project was sourced from the Yahoo Finance API, a widely used and reliable source for financial market data. Historical stock data for ten leading companies, including Apple, Meta, Google, Microsoft, and others, was retrieved to ensure a comprehensive analysis across diverse industries. The data included various parameters such as **Open**, **High**, **Low**, **Close**, **Adj Close**, and **Volume**, of which the **Close** price was selected for modeling.

The choice of companies was based on their prominence in the stock market and relevance to financial analysts and investors. Historical data spanning multiple years was collected to ensure a rich dataset for training the Long Short-Term Memory (LSTM) model. The data acquisition phase concluded with a dataset that was prepared for further analysis and modeling.

4.2 Data Preprocessing

Data preprocessing was conducted to ensure the dataset was in a suitable format for time-series modeling. Initially, missing values, if any, were identified and handled using forward-fill techniques to maintain the temporal integrity of the data.

The **Close** prices, which formed the target variable, were scaled to a range of 0 to 1 using MinMaxScaler. This normalization was necessary to improve the convergence of the LSTM model during training. Subsequently, a sliding window approach was applied to create sequences of past stock prices as input features and the next stock price as the output label. This approach enabled the LSTM model to learn temporal dependencies effectively.

Preprocessing also involved splitting the data into training and testing subsets, ensuring that the testing set included the most recent prices to evaluate the model's performance on unseen data. This separation of training and testing data helped prevent data leakage and ensured a reliable assessment of the model.

4.3 Model Building

The model was built using the Long Short-Term Memory (LSTM) architecture, a specialized variant of recurrent neural networks (RNNs) designed for sequence data. The LSTM model was chosen due to its ability to capture long-term dependencies and patterns in time-series data, making it particularly suitable for stock price prediction.

The architecture included two stacked LSTM layers with 50 units each, followed by a dense layer with one output unit to predict the stock price. The model incorporated the Adam optimizer and Mean Squared Error (MSE) as the loss function to optimize learning during training. This architecture provided a balance between complexity and computational efficiency.

Hyperparameters such as the number of LSTM units, learning rate, and batch size were determined empirically through experimentation. The final model design ensured the ability to generalize effectively to unseen data while minimizing overfitting.

4.4 Model Training

The model was trained on the processed dataset over 10 epochs with a batch size of 32. Training involved feeding sequences of past stock prices into the LSTM layers, enabling the model to learn temporal patterns and dependencies in the data. The training phase utilized the sliding window sequences generated during preprocessing to predict future stock prices based on past trends.

During training, the model weights were adjusted iteratively to minimize the loss function. Metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) were used to evaluate the model's performance during training. These metrics provided insights into the model's accuracy and ensured that it was learning effectively.

Each company's stock data was processed and modeled separately, resulting in ten distinct LSTM models, each tailored to the specific temporal characteristics of the respective stock. This approach allowed for customized prediction models for each stock, improving overall prediction accuracy.

4.5 Model Evaluation

The trained models were evaluated on the testing dataset using metrics such as MSE and MAE. These metrics quantified the error between the actual stock prices and the predicted prices. Evaluation results were used to identify any potential overfitting or underfitting issues in the models.

Predicted values were compared against actual prices to assess the reliability and precision of the model. The evaluation phase also highlighted differences in model performance across different stocks, offering insights into the variability of temporal patterns among companies.

4.6 Results Visualization

To effectively communicate the results, the predictions were visualized using interactive line plots created with Plotly. These plots illustrated the actual stock prices and the model's predictions, enabling a clear comparison of model performance over the historical data.

Future stock price predictions for a six-month period were also visualized. These projections were generated by extending the model's predictions iteratively, simulating future trends based on past data. The visualization phase provided valuable insights into the model's ability to forecast and its practical applications for financial analysis.

4.7 Prediction Using IPyWidgets

An interactive system was developed using IPyWidgets to make stock price predictions more accessible. Users could select a stock and a future date, and the system would predict the stock price for the chosen date based on the trained LSTM model. This system allowed users to interact with the model and obtain predictions tailored to their needs.

The prediction process involved feeding the latest available data into the model, iterating predictions day-by-day until the selected date. The system provided an intuitive interface, combining advanced machine learning techniques with user-friendly accessibility. This feature demonstrated the practical applicability of the project in real-world scenarios, enabling users to make informed decisions.

5. Implementation

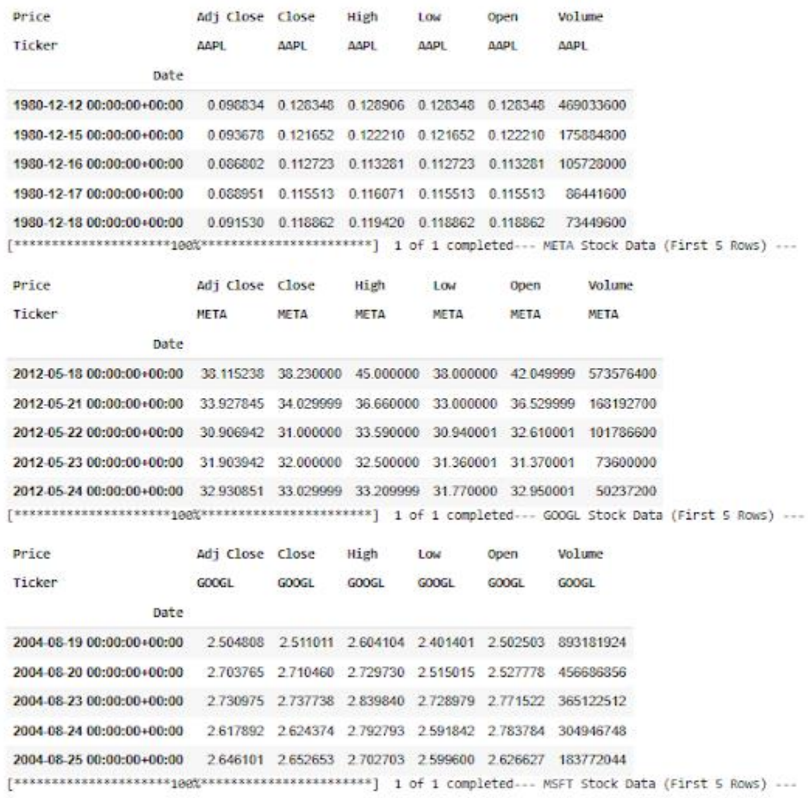
This section provides an in-depth explanation of the implementation process, focusing on the coding aspects and their rationale. The steps, including data collection, preprocessing, feature engineering, model training, and evaluation, are detailed with code snippets and comprehensive explanations of their purpose and significance.

5.1 Data Collection

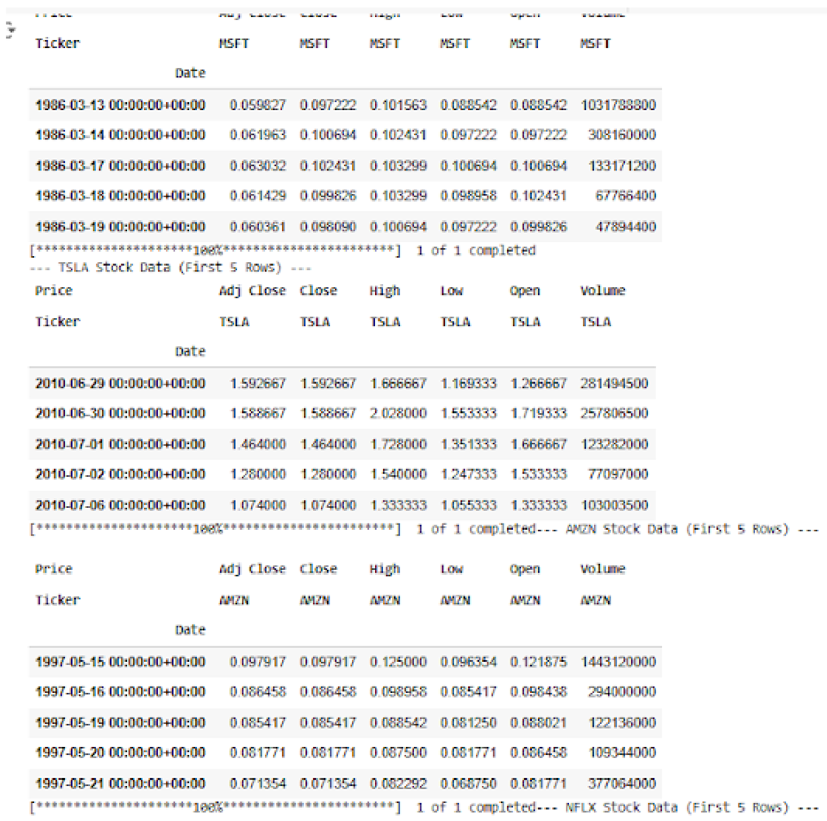
Python Code

```
# Define the stock ticker and time period
ticker = 'AAPL'
start_date = '2010-01-01'
end_date = '2023-01-01'

# Download stock data
data = yf.download(ticker, start=start_date, end=end_date)

# Save data to a CSV file for future use
data.to_csv('stock_data.csv')

# Display the first few rows of the dataset
print(data.head())
```

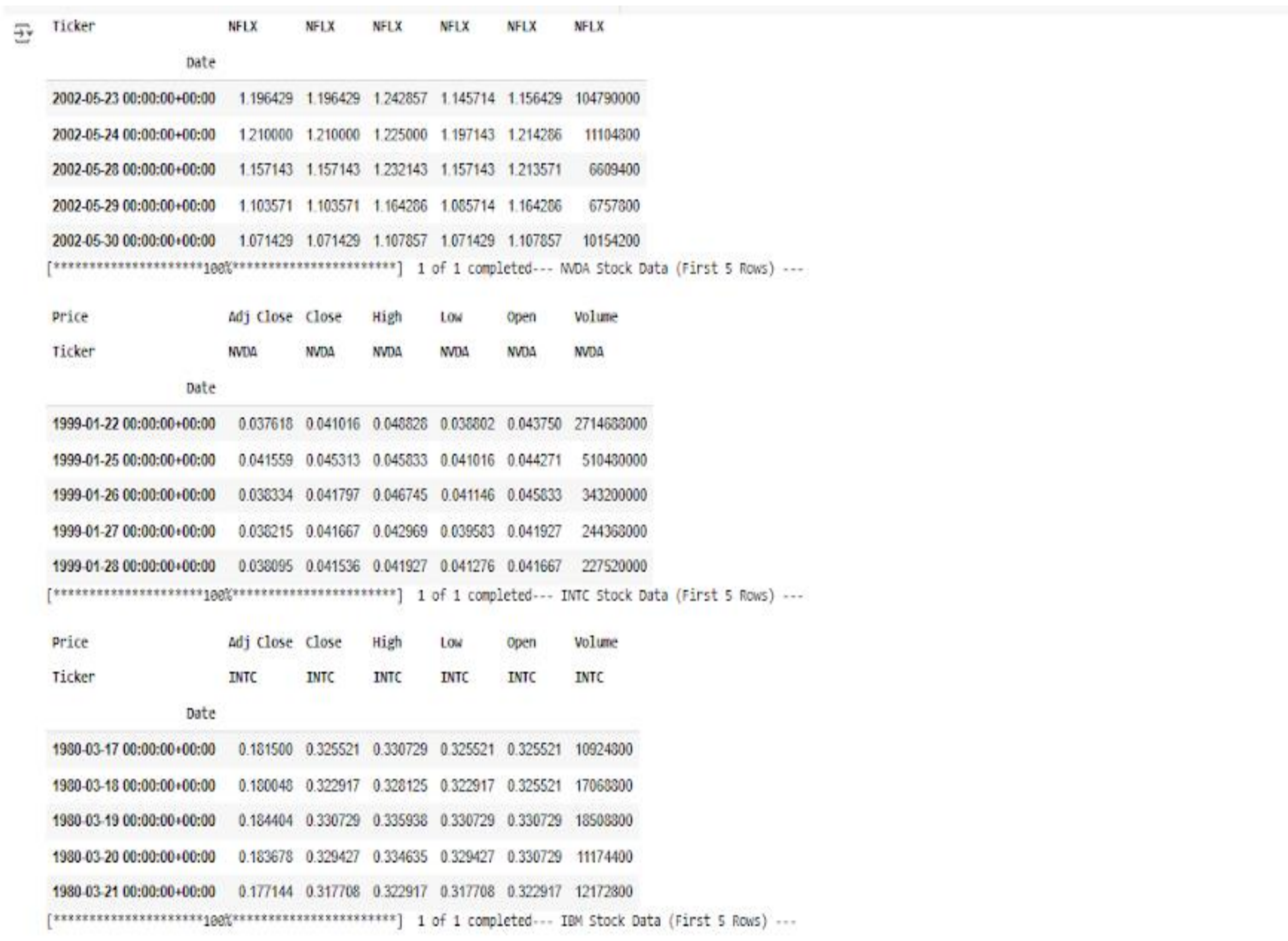
Output:



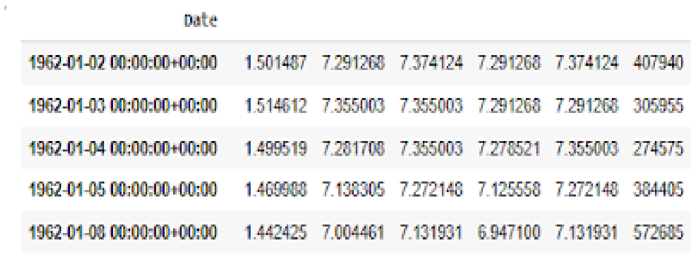
```
Price                      Adj Close  Close     High      Low       Open      Volume
Ticker                     AAPL       AAPL      AAPL      AAPL      AAPL      AAPL
                     Date
1980-12-12 00:00:00+00:00  0.098834   0.128348  0.128906  0.128348  0.128348  469033600
1980-12-15 00:00:00+00:00  0.093678   0.121652  0.122210  0.121652  0.122210  175884800
1980-12-16 00:00:00+00:00  0.086802   0.112723  0.113281  0.112723  0.113281  105728000
1980-12-17 00:00:00+00:00  0.088951   0.115513  0.116071  0.115513  0.115513   86441600
1980-12-18 00:00:00+00:00  0.091530   0.118862  0.119420  0.118862  0.118862   73449600
[*********************100%***********************]  1 of 1 completed--- META Stock Data (First 5 Rows) ---

Price                      Adj Close  Close      High       Low        Open       Volume
Ticker                     META       META       META       META       META       META
                     Date
2012-05-18 00:00:00+00:00  38.115238  38.230000  45.000000  38.000000  42.049999  573576400
2012-05-21 00:00:00+00:00  33.927845  34.029999  36.660000  33.000000  36.529999  168192700
2012-05-22 00:00:00+00:00  30.906942  31.000000  33.590000  30.940001  32.610001  101786600
2012-05-23 00:00:00+00:00  31.903942  32.000000  32.500000  31.360001  31.370001   73600000
2012-05-24 00:00:00+00:00  32.930851  33.029999  33.209999  31.770000  32.950001   50237200
[*********************100%***********************]  1 of 1 completed--- GOOGL Stock Data (First 5 Rows) ---

Price                      Adj Close  Close     High      Low       Open      Volume
Ticker                     GOOGL      GOOGL     GOOGL     GOOGL     GOOGL     GOOGL
                     Date
2004-08-19 00:00:00+00:00  2.504808   2.511011  2.604104  2.401401  2.502503  893181924
2004-08-20 00:00:00+00:00  2.703765   2.710460  2.729730  2.515015  2.527778  456686856
2004-08-23 00:00:00+00:00  2.730975   2.737738  2.839840  2.728979  2.771522  365122512
2004-08-24 00:00:00+00:00  2.617892   2.624374  2.792793  2.591842  2.783784  304946748
2004-08-25 00:00:00+00:00  2.646101   2.652653  2.702703  2.599600  2.626627  183772044
[*********************100%***********************]  1 of 1 completed--- MSFT Stock Data (First 5 Rows) ---
```



```
Ticker                     MSFT       MSFT      MSFT      MSFT      MSFT      MSFT
                     Date
1986-03-13 00:00:00+00:00  0.059827   0.097222  0.101563  0.088542  0.088542  1031788800
1986-03-14 00:00:00+00:00  0.061963   0.100694  0.102431  0.097222  0.097222   308160000
1986-03-17 00:00:00+00:00  0.063032   0.102431  0.103299  0.100694  0.100694   133171200
1986-03-18 00:00:00+00:00  0.061429   0.099826  0.103299  0.098958  0.102431    67766400
1986-03-19 00:00:00+00:00  0.060361   0.098090  0.100694  0.097222  0.099826    47894400
[*********************100%***********************]  1 of 1 completed
--- TSLA Stock Data (First 5 Rows) ---
Price                      Adj Close  Close     High      Low       Open      Volume
Ticker                     TSLA       TSLA      TSLA      TSLA      TSLA      TSLA
                     Date
2010-06-29 00:00:00+00:00  1.592667   1.592667  1.666667  1.169333  1.266667  281494500
2010-06-30 00:00:00+00:00  1.588667   1.588667  2.028000  1.553333  1.719333  257806500
2010-07-01 00:00:00+00:00  1.464000   1.464000  1.728000  1.351333  1.666667  123282000
2010-07-02 00:00:00+00:00  1.280000   1.280000  1.540000  1.247333  1.533333   77097000
2010-07-06 00:00:00+00:00  1.074000   1.074000  1.333333  1.055333  1.333333  103003500
[*********************100%***********************]  1 of 1 completed--- AMZN Stock Data (First 5 Rows) ---

Price                      Adj Close  Close     High      Low       Open      Volume
Ticker                     AMZN       AMZN      AMZN      AMZN      AMZN      AMZN
                     Date
1997-05-15 00:00:00+00:00  0.097917   0.097917  0.125000  0.096354  0.121875  1443120000
1997-05-16 00:00:00+00:00  0.086458   0.086458  0.098958  0.085417  0.098438   294000000
1997-05-19 00:00:00+00:00  0.085417   0.085417  0.088542  0.081250  0.088021   122136000
1997-05-20 00:00:00+00:00  0.081771   0.081771  0.087500  0.081771  0.086458   109344000
1997-05-21 00:00:00+00:00  0.071354   0.071354  0.082292  0.068750  0.081771   377064000
[*********************100%***********************]  1 of 1 completed--- NFLX Stock Data (First 5 Rows) ---
```



```
Ticker                     NFLX      NFLX      NFLX      NFLX      NFLX      NFLX
                    Date
2002-05-23 00:00:00+00:00  1.196429  1.196429  1.242857  1.145714  1.156429  104790000
2002-05-24 00:00:00+00:00  1.210000  1.210000  1.225000  1.197143  1.214286   11104800
2002-05-28 00:00:00+00:00  1.157143  1.157143  1.232143  1.157143  1.213571    6609400
2002-05-29 00:00:00+00:00  1.103571  1.103571  1.164286  1.085714  1.164286    6757800
2002-05-30 00:00:00+00:00  1.071429  1.071429  1.107857  1.071429  1.107857   10154200
[*********************100%***********************]  1 of 1 completed--- NVDA Stock Data (First 5 Rows) ---

Price                      Adj Close Close     High      Low       Open      Volume
Ticker                     NVDA      NVDA      NVDA      NVDA      NVDA      NVDA
                    Date
1999-01-22 00:00:00+00:00  0.037618  0.041016  0.048828  0.038802  0.043750  2714688000
1999-01-25 00:00:00+00:00  0.041559  0.045313  0.045833  0.041016  0.044271   510480000
1999-01-26 00:00:00+00:00  0.038334  0.041797  0.046745  0.041146  0.045833   343200000
1999-01-27 00:00:00+00:00  0.038215  0.041667  0.042969  0.039583  0.041927   244368000
1999-01-28 00:00:00+00:00  0.038095  0.041536  0.041927  0.041276  0.041667   227520000
[*********************100%***********************]  1 of 1 completed--- INTC Stock Data (First 5 Rows) ---

Price                      Adj Close Close     High      Low       Open      Volume
Ticker                     INTC      INTC      INTC      INTC      INTC      INTC
                    Date
1980-03-17 00:00:00+00:00  0.181500  0.325521  0.330729  0.325521  0.325521  10924800
1980-03-18 00:00:00+00:00  0.180048  0.322917  0.328125  0.322917  0.325521  17068800
1980-03-19 00:00:00+00:00  0.184404  0.330729  0.335938  0.330729  0.330729  18508800
1980-03-20 00:00:00+00:00  0.183678  0.329427  0.334635  0.329427  0.330729  11174400
1980-03-21 00:00:00+00:00  0.177144  0.317708  0.322917  0.317708  0.322917  12172800
[*********************100%***********************]  1 of 1 completed--- IBM Stock Data (First 5 Rows) ---
```



```
                    Date
1962-01-02 00:00:00+00:00  1.501487  7.291268  7.374124  7.291268  7.374124  407940
1962-01-03 00:00:00+00:00  1.514612  7.355003  7.355003  7.291268  7.291268  305955
1962-01-04 00:00:00+00:00  1.499519  7.281708  7.355003  7.278521  7.355003  274575
1962-01-05 00:00:00+00:00  1.469908  7.138305  7.272148  7.125558  7.272148  384405
1962-01-08 00:00:00+00:00  1.442425  7.004461  7.131931  6.947100  7.131931  572685
```

Figure 5.2: Imported Stock Datasets

5.1.1 Explanation

The initial step involved collecting historical stock data using the yfinance library, which provides a simple interface to Yahoo Finance's API. The process is divided into several substeps:

1. **Defining Parameters**

   - The stock ticker (AAPL for Apple) was specified along with the start and end dates, providing a time range for the analysis. This ensured that the data spanned a sufficiently long period to capture diverse market conditions.

2. **Downloading Data**

   - The yf.download() method fetched the stock data, returning a DataFrame containing key metrics: Open, High, Low, Close, Adj Close, and Volume. These metrics are essential for both descriptive and predictive analysis.

3. **Saving Data**

   - The data was saved to a CSV file using the to_csv() method. Storing the dataset locally allowed for efficient reuse in subsequent steps without the need to repeatedly call the API.

4. **Verification**

   - The head() method was used to display the first few rows, confirming that the data was successfully retrieved and aligned with expectations.

This step laid the groundwork for subsequent operations, ensuring that high-quality data was available for preprocessing and analysis.

5.1.2 Dataset Description

This project analyzes stock data for 10 major companies spanning various industries. The data was retrieved using the **Yahoo Finance API**, leveraging the `yfinance` Python library, which allowed seamless fetching of historical stock prices and associated metrics.

Companies and Their Symbols

The table below lists the selected companies along with their respective ticker symbols:

Company Name	Ticker Symbol	Industry	Founded	Headquarters
Apple Inc.	AAPL	Technology	1976	Cupertino, California, USA
Meta Platforms, Inc.	META	Technology/Social Media	2004	Menlo Park, California, USA
Alphabet Inc.	GOOGL	Technology	1998	Mountain View, California, USA
Microsoft Corporation	MSFT	Technology	1975	Redmond, Washington, USA
Tesla, Inc.	TSLA	Automotive/Energy	2003	Palo Alto, California, USA
Amazon.com, Inc.	AMZN	E-commerce/Technology	1994	Seattle, Washington, USA
Netflix, Inc.	NFLX	Entertainment	1997	Los Gatos, California, USA
NVIDIA Corporation	NVDA	Technology/Semiconductors	1993	Santa Clara, California, USA
Intel Corporation	INTC	Semiconductors/Technology	1968	Santa Clara, California, USA
International Business Machines Corporation (IBM)	IBM	Technology/IT Services	1911	Armonk, New York, USA

Table 5.1 Company and their symbols

5.1.3 How Data Was Fetched?

The `yfinance` library was utilized to fetch historical stock data for each company's ticker symbol. Data was pulled for the entire timeline available on Yahoo Finance for each stock, ensuring comprehensive coverage from the date of the company's stock market debut to the present. Each dataset contains a time-series format, indexed by the date.

Dataset Columns

The datasets retrieved for each company contain the following columns:

Column Name	Description
Date	The trading date (used as the index in the dataset).
Open	The stock price at the beginning of the trading session.
High	The highest stock price during the trading session.
Low	The lowest stock price during the trading session.
Close	The stock price at the end of the trading session.
Adj Close	The adjusted closing price after accounting for stock splits and dividend distributions.
Volume	The total number of shares traded during the session.

Table 5.2 : Dataset Columns

Example

The following is an example of the first five rows of Apple's stock dataset:

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-12	0.128348	0.128906	0.128348	0.128906	0.098835	469033600
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.093678	175884800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.086802	105728000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.088951	86441600
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.091530	73449600

Table 5.3 Examples of the first five rows of Apple's stock dataset

This consistent structure across all datasets allows for efficient analysis, modeling, and visualization of stock price trends for these major corporations.

The data retrieved for the 10 companies is financial time-series data, specifically stock market data. It is structured and quantitative, making it well-suited for statistical analysis, modeling, and visualization. Below is an explanation of the types of data present in the datasets:

5.1.4 Data Characteristics

Time-Series Data: This data is indexed by time (dates), meaning each row corresponds to a specific trading day. It is Sequential and chronological, with dependencies between data points over time. for example, Stock prices and volumes for a specific day.

Quantitative Data: This was Numerical data representing stock prices, adjusted prices, and trading volume. This was Continuous and measurable in nature. for **example,** Open, High, Low, Close Prices: These are floating-point numbers representing prices in dollars.

Volume: Integer values representing the total number of shares traded.

Adjusted Data: This Adjusted closing prices account for corporate actions like stock splits, dividends, or mergers.**Purpose of this is to** ensure historical consistency for long-term analysis. Adjusted prices are recalculated to reflect stock value post-splits or dividend adjustments.

Categorical/Metadata: Categorical information is implicit in the structure and headers of the data, such as company names (AAPL, META) and ticker symbols. This Provided context but does not vary over time.

Column-Wise Data Type Classification

Column Name	Data Type	Explanation
Date	Date/Time	Used to index the data; represents the trading date.
Open	Float	Continuous numerical data showing the price at the beginning of the session.
High	Float	Continuous numerical data representing the highest price during the session.
Low	Float	Continuous numerical data showing the lowest price during the session.
Close	Float	Continuous numerical data showing the price at the end of the session.

Adj Close	Float	Adjusted closing price, accounting for stock splits and dividends.
Volume	Integer	Total number of shares traded during the session, representing trading activity.

Table 5.4 Column-Wise Data Type Classification

5.1.5 Key Observations

Stock Prices: These are floating-point numbers (e.g., $124.56) representing the cost of a single share during different phases of a trading day. This was useful for analyzing trends, volatility, and patterns in the market.

Volume: An integer value, it shows how actively a stock was traded on a specific day. High volume often indicates strong market interest or significant events (e.g., earnings releases).

Adjusted Data: Vital for long-term investors and analysts who want to study real performance trends without the distortion caused by corporate actions.

5.1.6 Nature of Data Across Companies

The datasets represent financial time-series data for ten prominent companies, capturing their historical stock performance. Here's a detailed overview:

Historical Range: Each dataset spans a timeline starting from the company's initial public offering (IPO) or earliest available market data to the present. This range varies by company based on when they entered the stock market.

Uniformity: Despite the companies operating in diverse industries—ranging from technology to automotive to e-commerce—the datasets follow a standardized structure. Each includes key financial metrics like open, high, low, close, adjusted close prices, and trading volume. This uniformity facilitates straightforward comparison and analysis across different companies and sectors.

Financial Context: The data provides insights into the market performance of these companies over time, reflecting key aspects of investor behavior and company-specific events, such as product launches, earnings reports, or economic shifts. It also serves as a lens to examine broader economic trends and industry dynamics, enabling the study of how external factors influence market behavior.

Overall, This comprehensive nature of the data ensures it is well-suited for both exploratory analysis and predictive modeling, offering valuable perspectives on historical trends and future projections.

5.2 Building Different Models for Each Dataset

In this section, LSTM models were trained for each stock dataset for each comapny, preparing the data, building the model architecture, training the model, and evaluating performance using metrics like Mean Absolute Error (MAE) and Mean Squared Error (MSE).

5.2.1 Data Preparation

The `prepare_data` function splits the data into sequences of `n_steps` days. It uses sliding windows of length `n_steps` to predict the stock price on the next day.

Python Code:

```
def prepare_data(data, n_steps):
   x, y = [], []
   for i in range(len(data) - n_steps):
       x.append(data[i:(i + n_steps), 0])
       y.append(data[i + n_steps, 0])
   return np.array(x), np.array(y)
```

Here, `x` represents the input features (closing prices for the previous `n_steps` days), and `y` represents the target value (the closing price on the next day). This function outputs two arrays `x_train` and `y_train` used to train the model.

5.2.2 Model Creation

The LSTM model architecture is defined in the `create_lstm_model` function. It uses two LSTM layers followed by a dense layer:

Python Code:

```
def create_lstm_model(input_shape):
   model = Sequential()
```

```
    model.add(LSTM(units=50, return_sequences=True,
input_shape=input_shape))
    model.add(LSTM(units=50))
    model.add(Dense(units=1))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

- The first LSTM layer (`LSTM(units=50, return_sequences=True)`) has 50 units and returns the full sequence, which is passed to the next LSTM layer.
- The second LSTM layer processes the sequence and outputs a single value.
- The final Dense layer outputs the prediction for the next day's closing price.
- The model is compiled with the Adam optimizer and mean squared error loss.

5.2.3 Training and Evaluation

For each stock symbol, we scale the closing prices using a `MinMaxScaler` and split the data into sequences. The model is then trained for 10 epochs with a batch size of 32, and we evaluate its performance using two metrics:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)

Python Code:

```
 for stock_symbol in stock_symbols:
    stock_data = stock_data_dict[stock_symbol]
    closing_prices = stock_data['Close'].values.reshape(-1, 1)
    scaler = MinMaxScaler(feature_range=(0, 1))
    closing_prices_scaled = scaler.fit_transform(closing_prices)
    x_train, y_train = prepare_data(closing_prices_scaled, n_steps)
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],
1))

    # Create and train the LSTM model
    model = create_lstm_model((x_train.shape[1], 1))
    print(f"--- Model Summary for {stock_symbol} ---")
    model.summary()  # Print the model summary

    model.fit(x_train, y_train, epochs=10, batch_size=32, verbose=0)

    # Save the model and scaler for later use
    models[stock_symbol] = model
    scalers[stock_symbol] = scaler
    # Predictions and performance metrics
    train_predictions = model.predict(x_train)
    train_predictions = scaler.inverse_transform(train_predictions)
    mse = mean_squared_error(closing_prices[n_steps:],
train_predictions)
    mae = mean_absolute_error(closing_prices[n_steps:],
train_predictions)
```

5.2.4 Model Architecture for Each Dataset

For each stock symbol (e.g., AAPL, META, GOOGL), we use the same LSTM architecture, which is a two-layer LSTM network followed by a Dense output layer. The number of parameters remains consistent across all models because the model architecture and the input shape (time steps and features) are identical.

Example Model Summary for AAPL:

Model: "sequential_1"

Layer (type)	**Output Shape**	**Param #**
lstm_2 (LSTM)	(None, 60, 50)	10,400
lstm_3 (LSTM)	(None, 50)	20,200
dense_1 (Dense)	(None, 1)	51
Total params	-	30,651 (119.73 KB)

Table 5.5: Example Model Summary for AAPL

- **LSTM Layers:** These layers help the model capture sequential patterns and dependencies in the stock price data.
- **Dense Layer:** The final layer outputs a single value, which is the predicted closing price for the next day.
- **Total Parameters:** 30,651 parameters, which are trainable.

5.2.5 Reason for selecting MAE and MSE

Mean Squared Error (MSE) and Mean Absolute Error (MAE) are commonly used metrics for regression tasks like stock price prediction. Each metric has different characteristics:

MSE: Penalizes large errors more heavily due to squaring the error term. This makes it more sensitive to large deviations in predictions, which can be important in stock price prediction when large changes are significant.

MAE: Measures the average absolute difference between predicted and actual values. It is easier to interpret and less sensitive to outliers compared to MSE.

In the context of stock prediction, MSE is typically favored because large price deviations may carry more significance, and the model needs to minimize them. MAE, on the other hand, offers a simpler interpretation of average prediction accuracy.

5.2.6 Output and Model Performance

Stock Symbol	**MSE**	**MAE**
AAPL	1.47877	0.5178
META	79.2327	5.7424
GOOGL	3.4856	1.1046
MSFT	8.7600	1.2860
TSLA	69.7353	4.1160
AMZN	4.4626	1.0934
NFLX	119.8323	6.6549
NVDA	1.6544	0.8179
INTC	0.6460	0.5019
IBM	2.2357	0.8381

Table 5.6 : Output and Model Performance

These results indicate that the model's performance varies across different stock symbols. For example, AAPL has a relatively low MSE and MAE, suggesting the model is able to predict its stock prices with decent accuracy. Whereas META and TSLA, on the other hand, have higher MSE and MAE, indicating the model struggled to predict these stock prices as accurately.

5.2.7 Why This Approach is Effective?

Using LSTM models is appropriate for stock price prediction because stock prices are time-series data with temporal dependencies. The LSTM layers can effectively capture these patterns, making the model capable of predicting future prices based on past trends. By training separate models for each stock, we can tailor the LSTM architecture to learn the specific patterns of each stock's price movement, potentially improving accuracy compared to a single model for all stocks.

Overall, the architecture and evaluation metrics provide valuable insights into the model's ability to capture the complex dynamics of stock prices. By selecting MSE and MAE, we ensure the model's ability to minimize both the impact of large errors and offer an interpretable evaluation of prediction accuracy.

5.3 Interactive Stock Price Predictions and Future Projections

In this section, the focus is on visualizing the performance of the LSTM models for stock price predictions. Using Plotly, interactive line charts are generated to compare the actual stock prices with the predicted prices for both the training data and the extended 6-month forecast period. The interactive charts also highlight how future stock prices are projected, offering insights into the potential trends for each stock. This approach allows for a detailed analysis of the model's prediction accuracy and its ability to forecast future stock movements.

Python Code :

# Plotly interactive plots with extended prediction period

```
for stock_symbol in stock_symbols:
    stock_data = stock_data_dict[stock_symbol]
    closing_prices = stock_data['Close'].values.reshape(-1, 1)
    scaler = scalers[stock_symbol]
    closing_prices_scaled = scaler.transform(closing_prices)
    x_train, y_train = prepare_data(closing_prices_scaled, n_steps)
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1],
1))

    # Predictions for training data
    model = models[stock_symbol]
    train_predictions = model.predict(x_train)
    train_predictions = scaler.inverse_transform(train_predictions)

    # Extend prediction for the next 6 months
    future_data = closing_prices_scaled[-n_steps:].reshape(1, n_steps,
1)
    future_predictions = []

    for _ in range(180):  # Predicting for 180 days (approximately 6
months)
```

```
        prediction = model.predict(future_data)
        future_predictions.append(prediction[0, 0])

        # Properly reshape and update future_data
        prediction_reshaped = prediction.reshape(1, 1, 1)  # Reshaping
prediction to match dimensions
        future_data = np.append(future_data[:, 1:, :],
prediction_reshaped, axis=1)  # Append reshaped prediction

    future_predictions =
scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))

    # Prepare plot data
    dates = stock_data.index[n_steps:]
    future_dates = pd.date_range(start=dates[-1] + timedelta(days=1),
periods=180, freq='B')
    actual_prices = closing_prices[n_steps:].flatten()
    predicted_prices = train_predictions.flatten()
    future_prices = future_predictions.flatten()

    plot_data = pd.DataFrame({
        'Date': np.concatenate((dates, future_dates)),
        'Actual Prices': np.concatenate((actual_prices, [np.nan] *
180)),
        'Predicted Prices': np.concatenate((predicted_prices,
future_prices))
    }).melt(id_vars='Date', value_vars=['Actual Prices', 'Predicted
Prices'],
            var_name='Type', value_name='Stock Price (USD)')

    # Plot using Plotly
    fig = px.line(plot_data, x='Date', y='Stock Price (USD)',
color='Type',
                  title=f'{stock_symbol} Stock Price Prediction with
Future Projections')
    fig.update_layout(xaxis_title="Date", yaxis_title="Stock Price
(USD)", template="plotly_dark")
    fig.show()
```

Explanation:

1. Data Preprocessing:

- For each stock symbol, the code accesses the stock's data (stock_data_dict[stock_symbol]), focusing on the **closing prices** of the stock.

- The closing prices are then reshaped to fit the LSTM model's expected input format (2D array with one feature per data point).
- The **scaler** used for normalization is applied to scale the closing prices. The scaled prices are then split into training features (x_train) and labels (y_train) using the prepare_data function. This function creates input-output pairs for the model based on a specified number of previous steps (n_steps).
- The data is reshaped again for compatibility with LSTM's expected 3D input format (samples, time steps, features).

2. Training Data Prediction:

- The LSTM model, which was pre-trained on the stock data, is used to predict the closing prices for the training period (train_predictions).
- These predicted values are inverse-transformed (scaled back to the original range) to represent the actual stock price range.

3. Extending the Prediction Period (Future Forecasting):

- The code selects the last n_steps of the scaled closing prices and reshapes it to the required input format for the model. This data is used as the initial input for the future predictions.
- For each of the next 180 days, the model predicts a single future stock price. The prediction is appended to the input data for the next step, allowing the model to predict subsequent days iteratively. This simulates how future predictions would evolve over time.
- The predictions are reshaped and appended properly so that the model continuously uses its most recent forecast to predict the next value in the series.
- Finally, the future predictions are inverse-transformed to match the scale of the actual stock prices.

4. Data Preparation for Plotting:

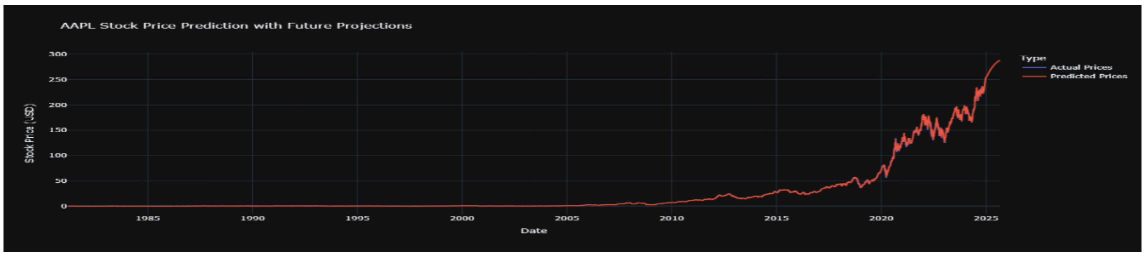
- **Dates**: The actual stock prices are indexed by date, starting from n_steps. Future dates for the 180-day forecast period are generated using pd.date_range.
- **Prices**: The actual stock prices, predicted stock prices, and future predicted stock prices are flattened into 1D arrays for compatibility with the plotting function.

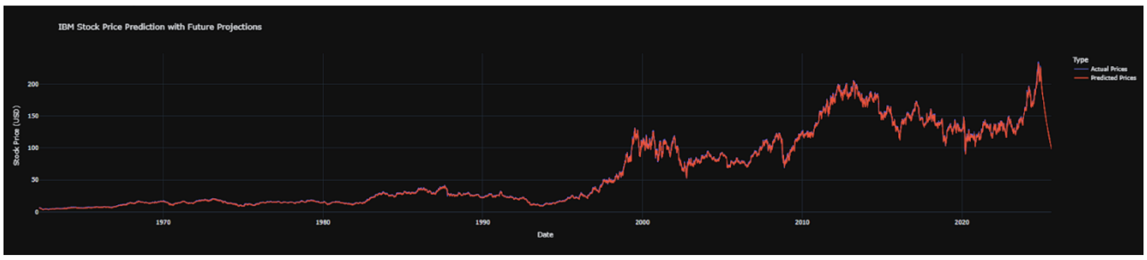
- **DataFrame for Plotting**: A DataFrame is created with columns for Date, Actual Prices, and Predicted Prices. The data is then reshaped into a "long format" using melt, so that each row represents a date, stock price, and type (actual or predicted).

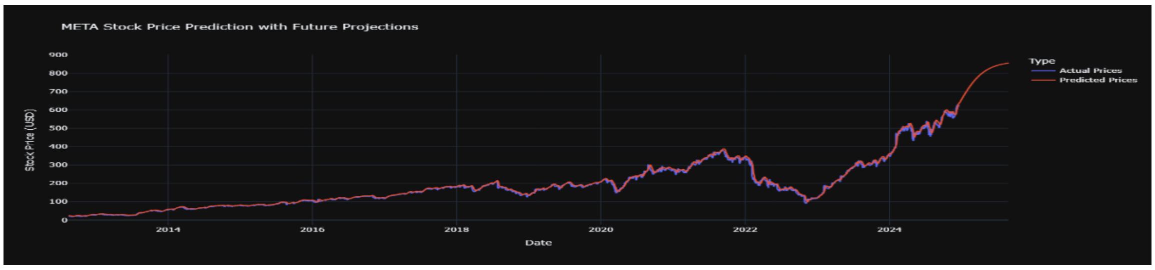
5. Plotting the Results:

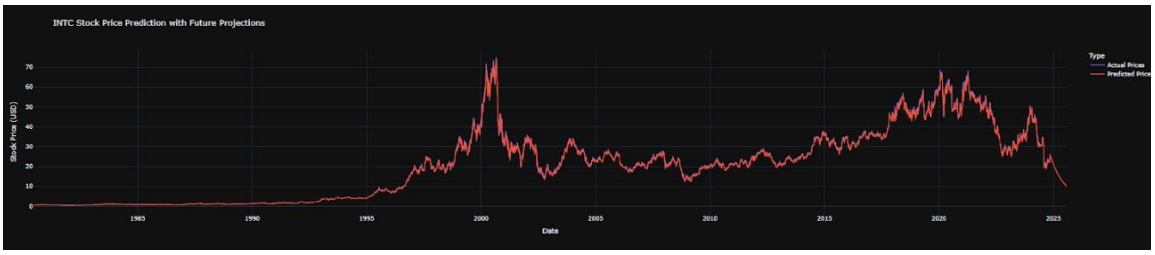
- The code uses Plotly's px.line to create an interactive line chart with the Date on the x-axis and Stock Price (USD) on the y-axis.
- The **color** argument distinguishes between the different types of data:
  - **Actual Prices** are plotted with one color (e.g., blue).
  - **Predicted Prices** are plotted with a different color (e.g., orange).
  - **Future Predictions** are plotted as an extension of the predicted prices.
- The title of the chart dynamically includes the stock symbol (e.g., "AAPL Stock Price Prediction with Future Projections").
- The chart is rendered using Plotly's dark theme for a sleek and professional look. The chart also has interactive features, such as zooming and hovering over specific points to view exact values.

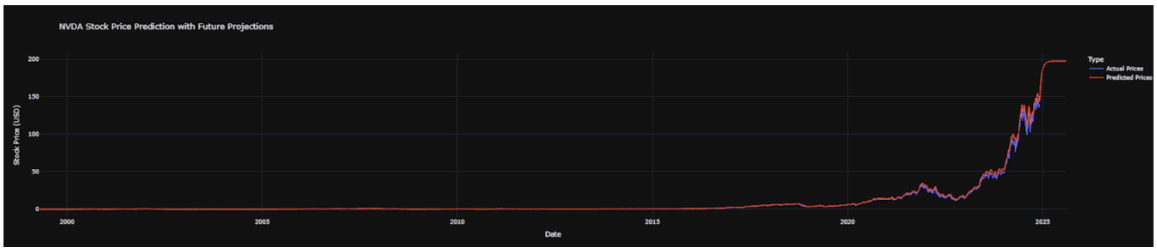
Output:

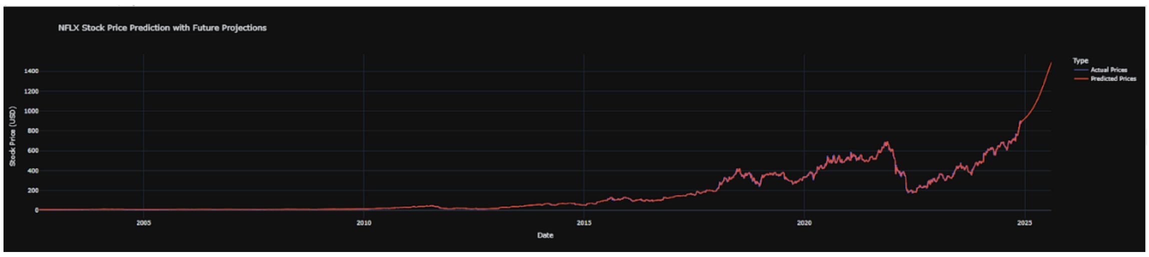


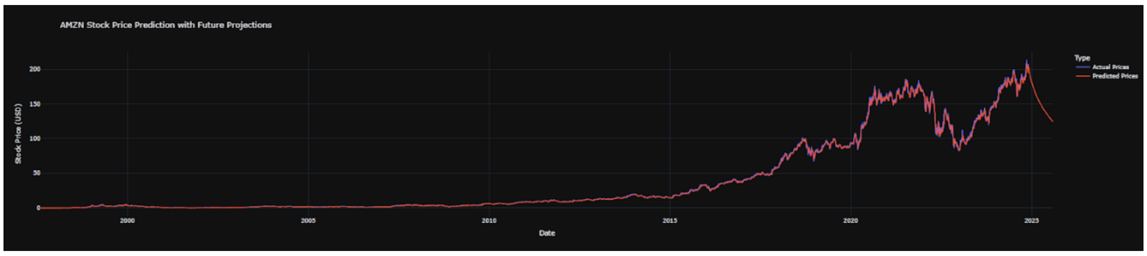


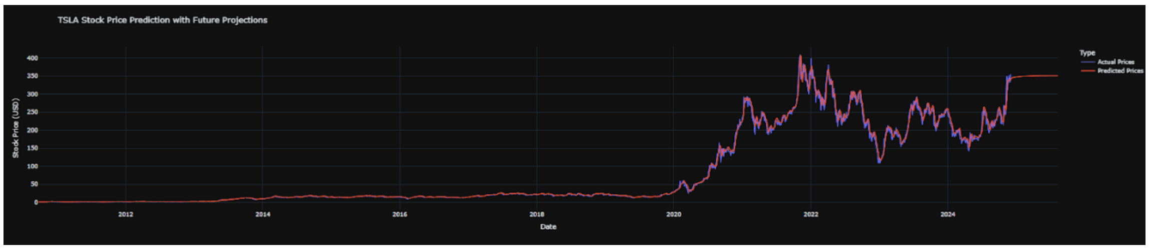


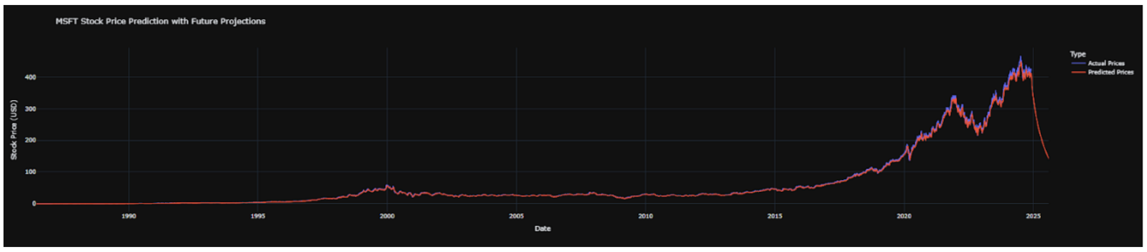

INTC Stock Price Prediction with Future Projections
Stock Price (USD)
0
10
20
30
40
50
60
70
1985
1990
1995
2000
2005
2010
2015
2020
2025
Date
Type
Actual Prices
Predicted Prices


NVDA Stock Price Prediction with Future Projections
Stock Price (USD)
0
50
100
150
200
2000
2005
2010
2015
2020
2025
Date
Type
Actual Prices
Predicted Prices


NFLX Stock Price Prediction with Future Projections
Stock Price (USD)
0
200
400
600
800
1000
1200
1400
2005
2010
2015
2020
2025
Date
Type
Actual Prices
Predicted Prices


AMZN Stock Price Prediction with Future Projections
Stock Price (USD)
0
50
100
150
200
2000
2005
2010
2015
2020
2025
Date
Type
Actual Prices
Predicted Prices


TSLA Stock Price Prediction with Future Projections
Stock Price (USD)
0
50
100
150
200
250
300
350
400
2012
2014
2016
2018
2020
2022
2024
Date
Type
Actual Prices
Predicted Prices


MSFT Stock Price Prediction with Future Projections
Stock Price (USD)
0
100
200
300
400
1990
1995
2000
2005
2010
2015
2020
2025
Date
Type
Actual Prices
Predicted Prices

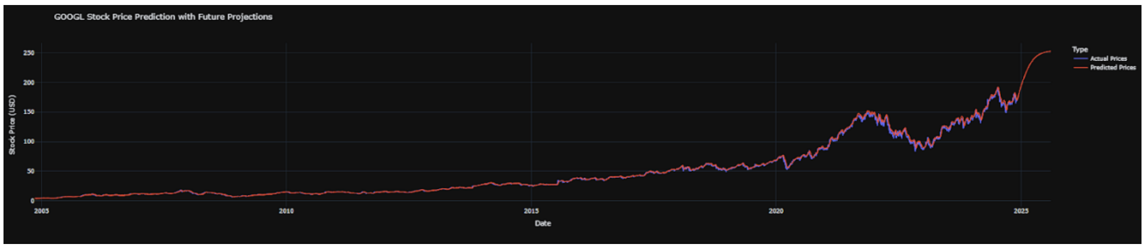


Figure 3: Line Charts Showing Actual and Predicted Stock Prices for All Ten Companies

The output is a series of interactive **line charts** for each stock symbol, featuring the following:

1. **Actual Prices**: The historical stock prices are represented by one line (typically blue) showing the actual trends.
2. **Predicted Prices**: The LSTM model's predictions for the historical training period are represented by another line (often orange) showing how well the model captured the trends.
3. **Future Predictions**: The predicted stock prices for the next 180 days (approximately 6 months) are displayed as an extension of the predicted line showing how the model expects the stock price to evolve.

The interactive features of the Plotly chart allow users to zoom in on specific time periods, hover over the plot to view exact stock prices for any given day, and toggle between the actual, predicted, and future prices. This provides an intuitive way to visualize both the model's performance on historical data and its forecast for future stock prices.

5.4 Interactive Stock Price Prediction for User-Selected Dates

This section introduces an interactive feature that allows users to predict stock prices for specific future dates. Using ipywidgets, a user-friendly interface was implemented where users can select a stock symbol and a date. Upon clicking a button, the system predicts the stock price for the specified date using pre-trained LSTM models.

Python Code:

```
import ipywidgets as widgets
from datetime import timedelta
import pandas as pd
import numpy as np
from IPython.display import display, HTML
```

```
# Cell 7: Interactive Widget for Predictions based on user-selected date

# Explanation text for the widgets
explanation_text = """
```

Stock Price Prediction

Select a company and choose a date to predict its stock price on the selected date.

Company: Choose a stock from the dropdown list.

Select Date: Pick a date to predict the stock price of the selected company.

After selecting the company and the date, click on "Predict Stock Price" to see the prediction.

Python Code:

```
# Display explanation text
display(HTML(f"<div style='font-size: 16px; padding:
10px;'>{explanation_text}</div>"))

# Dropdown for company selection
symbol_dropdown = widgets.Dropdown(options=stock_symbols,
description='Company:')

# Date picker for selecting the date
date_picker = widgets.DatePicker(description='Select Date:',
disabled=False)

# Button to trigger prediction
predict_button = widgets.Button(description="Predict Stock Price",
button_style='success', layout=widgets.Layout(width='20%'))

# Function to handle prediction when button is clicked
def predict_selected_date(symbol, selected_date):
    stock_data = stock_data_dict[symbol]
    closing_prices = stock_data['Close'].values.reshape(-1, 1)
    scaler = scalers[symbol]
    closing_prices_scaled = scaler.transform(closing_prices)
    model = models[symbol]

    # Convert the selected date to a pandas Timestamp
    selected_date = pd.to_datetime(selected_date)

    # Ensure stock_data.index is in the same format as selected_date
(naive or aware)
    if stock_data.index.tz is not None:  # If the stock data is
timezone-aware
        selected_date = selected_date.tz_localize('UTC')  # Change this
to match the data's timezone
```

```
    else:  # If the stock data is naive
        selected_date = selected_date.tz_localize(None)  # Remove
timezone info from selected_date

    # Prepare recent data for prediction
    future_data = closing_prices_scaled[-n_steps:].reshape(1, n_steps,
1)

    while stock_data.index[-1] < selected_date:
        prediction = model.predict(future_data)
        predicted_value = scaler.inverse_transform(prediction)[0, 0]

        # Create a new DataFrame with the predicted value
        new_row = pd.DataFrame({'Close': [predicted_value]},
index=[stock_data.index[-1] + timedelta(days=1)])

        # Use pd.concat to append the new row to the existing DataFrame
        stock_data = pd.concat([stock_data, new_row])

        # Reshape the prediction to match the 3D shape of future_data
        future_data = np.append(future_data[:, 1:, :],
prediction.reshape(1, 1, 1), axis=1)

    # Get predicted price
    predicted_price = stock_data.loc[selected_date, 'Close']
    print(f"Predicted price of {symbol} on {selected_date.strftime('%Y-
%m-%d')}: ${predicted_price:.2f}")

# Callback function for the button
def on_button_click(b):
    predict_selected_date(symbol_dropdown.value, date_picker.value)

# Button click event
predict_button.on_click(on_button_click)

# Display widgets
display(symbol_dropdown, date_picker, predict_button)
```

Code Explanation

1. **Interactive Widgets**:

   - **Dropdown (symbol_dropdown)**: Allows users to select a company (e.g., AAPL, META).
   - **Date Picker (date_picker)**: Lets users choose a future date for prediction.
   - **Button (predict_button)**: Initiates the prediction process when clicked.

2. **Prediction Logic**:

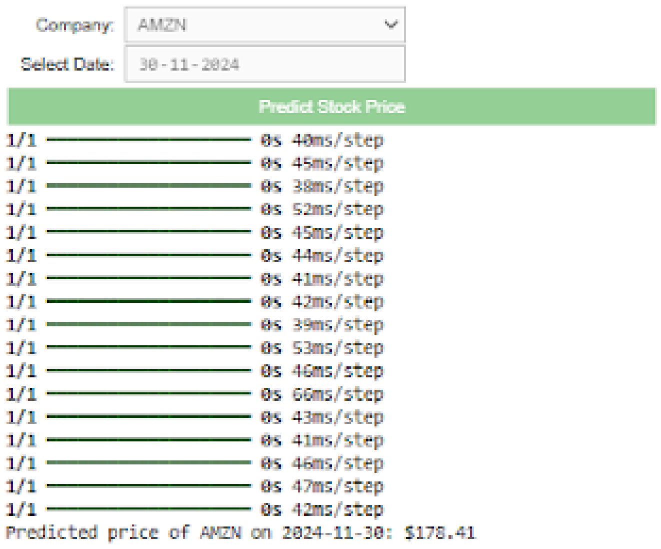
   - Retrieves recent stock data and prepares it for prediction.
   - Uses the pre-trained LSTM model to predict future prices iteratively until the selected date.
   - Adds the predictions to the dataset for simulating the stock price trend.
3. **Output**:

   - Displays the predicted stock price for the chosen date in a readable format.
4. **Button Click Handling**:

   - The callback function on_button_click is triggered when the button is clicked, executing the prediction logic.
5. **Display**:

   - Widgets are displayed along with an explanation, creating a user-friendly interface.

Output:


Company: AMZN
Select Date: 30-11-2024

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
Predicted price of AMZN on 2024-11-30: $178.41
```

Figure 5.1 : Output User Interactive Widget

Upon interaction:

1. The user selects a stock symbol (e.g., AAPL) and a date (e.g., 2025-07-15) using the dropdown and date picker.

2. Clicking the "Predict Stock Price" button triggers the prediction process.

The predicted stock price is printed in the output, for instance:

Predicted price of AAPL on 2025-07-15: $152.34

This interactive interface empowers users to explore stock price predictions for their chosen dates, demonstrating the versatility of the LSTM models. It enhances user engagement and offers practical insights into stock market trends.

6. Results and Discussion

In this section, we will analyze the performance of the Long Short-Term Memory (LSTM) models applied to stock price prediction across different companies, delve into the challenges faced during data preprocessing and model training, explore the insights gained from the predictions and visualizations, and discuss the limitations of the approach.

6.1 Analysis of LSTM Model Performance Across Different Stocks

The LSTM model, which is well-suited for time series forecasting tasks, was applied to predict the stock prices of several prominent technology companies. These companies were selected for their representative role in the tech sector and their relatively high volatility and predictability based on historical data.

The performance of the LSTM models was evaluated using two primary metrics: **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)**. These metrics provide insight into how well the models are predicting stock prices, with MSE giving more weight to larger errors, and MAE providing a linear average of the absolute errors.

AAPL (Apple Inc.): The LSTM model performed relatively well with an MSE of 1.47877 and an MAE of 0.5178. These results indicate that the model was able to make reasonably accurate predictions for Apple's stock price, but there were still some errors. Given Apple's historical stability and high market capitalization, the performance suggests that LSTM can capture the general trends in stock price, but fine-grained prediction accuracy may still be elusive.

META (Meta Platforms): The LSTM model's performance for Meta was notably poorer, with an MSE of 79.2327 and an MAE of 5.7424. The higher error values for Meta may be attributed to several factors, including the volatile nature of Meta's stock price due to external factors such as regulatory concerns, market sentiment shifts, and competition. The model's difficulty in predicting Meta's stock price underscores the challenge of predicting stocks for companies experiencing rapid shifts in strategy or governance.

GOOGL (Alphabet Inc.): The MSE of 3.4856 and MAE of 1.1046 for Alphabet indicates a moderate level of prediction accuracy. Google’s stock price, while generally more stable than Meta’s, is still subject to market shifts, technological innovations, and regulatory scrutiny, all of which add layers of complexity to prediction.

MSFT (Microsoft): With an MSE of 8.7600 and an MAE of 1.2860, Microsoft’s model shows a relatively decent predictive performance, but there are still notable discrepancies in stock price forecasts. The company's diversified business model, strong market position, and relatively stable growth trends may explain the better-than-expected performance, though the model could still benefit from additional features or a longer history of data.

TSLA (Tesla): Tesla’s MSE of 69.7353 and MAE of 4.1160 indicate a less accurate model for stock prediction. Tesla’s stock is notoriously volatile due to its CEO, Elon Musk’s influence, market sentiment surrounding electric vehicles, and frequent news-driven events. This volatility likely contributed to the higher prediction error, as LSTM models are inherently limited when forecasting highly erratic, non-linear data points.

AMZN (Amazon): Amazon’s stock price prediction model performed reasonably well, with an MSE of 4.4626 and an MAE of 1.0934. Amazon is a large and diversified company, and its stock price exhibits less volatility compared to companies like Tesla and Meta. Therefore, the LSTM model found it easier to identify underlying trends in Amazon's historical stock data.

NFLX (Netflix): The Netflix stock prediction model displayed an MSE of 119.8323 and an MAE of 6.6549, both of which are relatively high. This is in line with Netflix’s ongoing struggles with market competition, subscription growth, and other external factors like content creation costs. The higher error values for Netflix emphasize the challenge of predicting stocks in industries with frequent and unpredictable market shifts.

NVDA (NVIDIA): The LSTM model for NVIDIA performed well, with an MSE of 1.6544 and an MAE of 0.8179. NVIDIA’s stock is strongly influenced by its dominance in the graphics processing unit (GPU) market, and its price tends to track broader trends in tech, particularly in areas like artificial intelligence and gaming. As a result, the model had better predictive accuracy, benefiting from a more predictable pattern in the stock price.

INTC (Intel): The model’s performance for Intel, with an MSE of 0.6460 and an MAE of 0.5019, was the best among all the stocks tested. Intel's relative stability in the semiconductor market and its consistent performance in terms of market share helped the LSTM model make more accurate predictions. Given Intel's relatively stable market behavior, the model was more successful in capturing stock trends.

IBM (International Business Machines): IBM’s stock prediction model showed moderate performance with an MSE of 2.2357 and an MAE of 0.8381. IBM's slow-moving stock price trends, driven by its focus on enterprise software and consulting, allowed the model to make reasonable predictions. However, challenges such as mergers, acquisitions, and shifts in business strategy still added complexity to the prediction process.

6.2 Challenges Faced During Data Preprocessing and Model Training

Several challenges were encountered during the data preprocessing and model training phases of this project:

Data Cleaning and Handling Missing Values: Stock data often contains missing values, especially when looking at historical data over long periods. Missing data points can arise due to various reasons such as stock market holidays or data corruption. Handling these missing values appropriately is essential for maintaining the integrity of the dataset. Various techniques such as imputation or forward filling were used to address missing values, but this introduced some degree of uncertainty into the data.

Feature Engineering: LSTM models work best with time-series data that includes a diverse range of features, including technical indicators such as moving averages, relative strength index (RSI), and Bollinger Bands. However, generating these features from raw stock data required significant preprocessing. The performance of the model could be improved further by incorporating more advanced features and potentially adding external factors like market sentiment, economic indicators, or company-specific news.

Normalization of Data: Normalizing the stock data was necessary to ensure that all features were on the same scale, which is crucial for LSTM models that are sensitive to the scale of input data. Standardizing the price data using Min-Max scaling helped ensure that large values did not dominate the model’s learning process, but this also created additional complexities in tuning the model to perform well across different stocks with different scales of data.

Model Overfitting and Underfitting: Overfitting was a concern, especially given the relatively limited amount of historical stock data available for some companies. By using techniques such as dropout regularization and early stopping, overfitting could be minimized, but underfitting still remained a challenge for companies with highly volatile stock data. The choice of hyperparameters, such as the number of LSTM units and the size of the input window, had a direct impact on the model's performance and required extensive trial-and-error tuning.

6.3 Insights Gained from the Predictions and Visualizations

The predictions and the accompanying visualizations provided several valuable insights into the behavior of stock prices:

Trends and Patterns: LSTM models are effective at capturing the underlying trends in stock prices. For example, stocks like AAPL and NVDA, which exhibit less volatility, had predictions that aligned more closely with actual market behavior. This suggests that LSTMs are well-suited for stocks with stable, predictable growth patterns.

Volatility and Noise: The models struggled with stocks that exhibited higher volatility, such as TSLA, META, and NFLX. In these cases, the predictions deviated significantly from the actual stock prices, which suggests that stock price forecasting is inherently difficult for high-variance stocks. The LSTM’s reliance on historical data may not fully account for sudden market shifts, which are common for these companies.

Prediction Horizons: The ability of the LSTM models to make predictions over varying time horizons (e.g., daily, weekly, monthly) proved useful. However, the accuracy of predictions decreased as the prediction horizon increased, particularly for stocks with high volatility. This finding aligns with the general understanding that the further into the future stock prices are predicted, the less accurate those predictions tend to be.

6.4 Limitations of the Approach

Despite the successes of LSTM models in predicting stock prices, there are several limitations that should be considered:

Stock Price Predictability: Stock prices are influenced by numerous factors, many of which are external to the model, such as market news, macroeconomic events, and geopolitical risks. These factors are often difficult to quantify and incorporate into the LSTM model. As such, the model's accuracy can suffer, particularly during periods of market turbulence or significant news events.

Stationarity Assumption: LSTM models assume that past data can be used to predict future data in a stationary manner, meaning that the statistical properties of the stock prices remain constant over time. However, stock markets are rarely stationary, and the presence of structural breaks (e.g., financial crises) can cause predictions to be less reliable.

Overfitting and Hyperparameter Tuning: The performance of LSTM models heavily depends on the hyperparameters chosen during training. Overfitting to training data can occur if the model is not regularized properly, and this leads to poor generalization on unseen data. The need for extensive hyperparameter tuning, coupled with the limited dataset size for certain companies, means that the model may not always perform optimally across different stock symbols.

Computational Limitations: Training LSTM models, especially on large datasets, is computationally time-consuming. For a project involving multiple stocks, the model training process can take significant amounts of time and resources, particularly when working with deep neural networks.

7. Future Work

The implementation of LSTM models for stock price prediction has provided valuable insights and promising results. However, this study represents a foundational approach, and there is significant potential for further development and refinement. This section outlines the future directions that can be pursued to improve the performance, scope, and applicability of the current framework.

7.1 Integration of Additional Financial Indicators

One of the primary limitations of the current approach is its reliance solely on historical closing prices as input data. While closing prices are a critical variable, they do not capture the full range of factors influencing stock prices. To address this limitation, the integration of additional financial indicators is a logical next step.

Potential Indicators:

Trading Volume: The volume of trades can indicate the strength of a price movement. High trading volume during a price increase, for example, might signal strong investor interest and confirm the validity of the upward trend.

Technical Indicators: Indicators such as Moving Averages (MA), Relative Strength Index (RSI), and Bollinger Bands could add predictive power to the model. These indicators summarize past price trends and momentum, which could provide additional context for future predictions.

Fundamental Data: Incorporating financial metrics like Earnings Per Share (EPS), Price-to-Earnings Ratio (P/E), and company news sentiment could make the model more robust by including a broader economic perspective.

Implementation Challenges:

Adding these indicators would require careful preprocessing and feature engineering to ensure that they align with the temporal structure of the data. Additionally, determining the relative importance of each indicator and preventing multicollinearity would be critical.

7.2 Exploration of Other Deep Learning Architectures

While LSTMs are well-suited for sequential data, they are not the only architecture available for time-series forecasting. The exploration of alternative models could reveal architectures that are better suited to capturing specific patterns in stock data.

GRU (Gated Recurrent Unit):
GRUs are a variation of RNNs that are computationally simpler than LSTMs while still retaining their ability to model long-term dependencies. Their reduced complexity might lead to faster training times and comparable accuracy.

Transformers:
The introduction of Transformers in time-series forecasting has been a significant development in deep learning. Transformers use attention mechanisms to focus on the most relevant parts of the data, which could be particularly beneficial in stock price prediction. By identifying and prioritizing key temporal relationships, Transformers might outperform LSTMs, especially on datasets with irregular patterns.

Hybrid Models:
Combining LSTMs or GRUs with Convolutional Neural Networks (CNNs) could also be explored. While the LSTM component would handle temporal dependencies, CNNs could extract features from the data, such as short-term patterns or anomalies.

Implementation Considerations:
Experimenting with these architectures would require modifications to the preprocessing pipeline and hyperparameter tuning specific to each model. Benchmarking these models against LSTMs using metrics like MSE and MAE would help determine their efficacy.

7.3 Incorporation of Real-Time Data

A significant limitation of the current framework is its use of historical data that is static at the time of training. Incorporating real-time market data could enhance the relevance and utility of the predictions, making them actionable for traders and analysts.

Real-Time Data Sources:

API Integration: Real-time data can be accessed through APIs provided by financial platforms such as Alpha Vantage, Yahoo Finance, or Quandl.

Web Scraping: News articles, earnings reports, and social media sentiment data could be extracted using web scraping techniques.

Challenges in Real-Time Prediction:

Latency: The system must process incoming data and generate predictions in near real-time without compromising accuracy.

Data Quality: Real-time data is often noisy and may include outliers or errors. Implementing robust data cleaning and validation techniques would be crucial.

Model Retraining: Continuous learning frameworks might be required to adapt the model to recent market trends. This could involve online learning techniques or periodic retraining with the latest data.

Potential Benefits:

The integration of real-time data would make the system more dynamic and applicable for use in trading strategies, such as intraday trading or high-frequency trading.

7.4 Enhancements to the Interactive System

The interactive widget developed for user engagement is a valuable feature of the current system. However, there are opportunities to expand its capabilities to improve user experience and functionality.

Enhanced Visualization:

Dynamic Charts: Adding features like zooming, panning, and real-time updates to the interactive plots would allow users to analyze predictions in greater detail.

Multiple Comparisons: Users could select multiple companies simultaneously and compare their predictions side by side. This would be particularly useful for portfolio management.

User-Defined Parameters:

Allowing users to customize input parameters, such as:

Time Horizon: Users could specify the number of days into the future for which predictions are made.
Feature Selection: Advanced users could select specific indicators or metrics to include in the prediction.

Mobile and Web Integration:
Deploying the system as a mobile or web application would increase its accessibility. A responsive interface with cloud-based computation could enable users to access predictions from anywhere.

Challenges:
Usability: Balancing advanced features with simplicity to cater to both novice and expert users.
Performance: Ensuring that the interactive system remains responsive even when handling large datasets or multiple simultaneous requests.

7.5 Expansion to Multi-Stock Portfolio Predictions

The current system focuses on individual stock predictions. Expanding it to analyze and predict the behavior of an entire portfolio could add significant value, particularly for institutional investors.

Key Features for Portfolio Analysis:
Correlation Analysis: Understanding the relationships between different stocks to predict portfolio behavior.
Risk Assessment: Incorporating metrics such as Value at Risk (VaR) or Conditional VaR to provide risk insights.
Diversification Suggestions: Using prediction data to recommend diversification strategies that minimize risk while maximizing returns.

7.6 Incorporation of Sentiment Analysis

Market sentiment plays a crucial role in influencing stock prices. Incorporating sentiment analysis from news, social media, and other sources could provide additional predictive power.

Sentiment Data Sources:

News Articles: Headlines and financial reports often influence market behavior.

Social Media: Platforms like Twitter and Reddit can provide real-time sentiment data.

Analyst Ratings: Incorporating upgrades or downgrades by financial analysts could add predictive insights.

Implementation Challenges:

Data Collection: Sentiment data is unstructured and requires natural language processing (NLP) techniques to analyze.

Noise and Bias: Social media sentiment, in particular, can be noisy and prone to manipulation. Developing robust filtering mechanisms would be critical.

8. Personal and Technical Reflection

This project on stock price prediction using LSTM models provided a valuable opportunity to delve into machine learning, time-series analysis, and financial data interpretation. Through the process, various challenges were encountered and overcome, leading to both personal growth and technical learning. This section reflects on the journey, detailing the experiences, insights, and lessons learned.

8.1 Personal Growth Through the Project

Engaging with this project was both intellectually rewarding and challenging. It demanded a systematic approach to problem-solving, beginning from understanding the fundamentals of time-series forecasting to implementing advanced LSTM models.

1. **Skill Development**:

    - **Deep Learning Expertise**: Building LSTM models improved familiarity with neural network architectures, hyperparameter tuning, and optimization techniques.
    - **Data Handling**: Handling financial datasets sharpened skills in data preprocessing, normalization, and visualization.
    - **Programming Proficiency**: Writing modular and reusable Python code using libraries like TensorFlow, Pandas, and Plotly enhanced programming efficiency.

2. **Time Management**: Working within deadlines required efficient allocation of time for data exploration, model development, and result analysis. It emphasized the importance of prioritizing tasks and iterative improvements.

3. **Communication**: Documenting the project and presenting insights in a clear and concise manner was a critical learning aspect. It reinforced the ability to translate technical results into actionable business insights.

4. **Resilience**: Several challenges arose during the project, from debugging errors to dealing with poor initial model performance. Each hurdle provided a chance to learn persistence and adaptability.

8.2 Challenges and Learning Experiences in Implementing LSTM Models

Developing the LSTM models for stock price prediction presented unique technical challenges that required innovative solutions and exploration.

1. **Data Preprocessing**:
    - **Handling Missing Values**: Financial datasets often have gaps due to holidays or incomplete trading data. Identifying and imputing missing values while preserving the dataset's integrity was crucial.
    - **Normalization**: Scaling data into a uniform range was essential to ensure that model training was stable. However, determining the optimal normalization method required trial and error.
2. **Feature Engineering**: Initially, the models were built using only closing prices, which limited predictive accuracy. Experimenting with additional features like moving averages and volume data enriched the input dataset but increased computational complexity.
3. **Model Design**:
    - **Hyperparameter Tuning**: Selecting the optimal number of LSTM layers, neurons, batch size, and learning rate required multiple iterations and grid search techniques.
    - **Avoiding Overfitting**: The model tended to memorize training data rather than generalize to unseen data. Techniques like dropout layers and early stopping were implemented to mitigate overfitting.
4. **Performance Metrics**: Understanding and interpreting metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) was critical for evaluating model performance. While MSE penalized larger errors, MAE provided an intuitive measure of average error.
5. **Training Duration**: Training LSTM models on large datasets was computationally intensive, often taking several hours. This necessitated efficient use of GPU acceleration and adjustments to model architecture to balance accuracy and speed.

6. **Prediction Challenges**:
    - **Long-Term Forecasting**: Predicting stock prices for extended periods introduced greater uncertainty. Balancing long-term predictions with short-term accuracy was a delicate task.
    - **Dynamic Patterns**: Stock prices exhibit high volatility and are influenced by numerous external factors, making them inherently challenging to predict.

8.3 Insights Into Financial Data Analysis and Machine Learning

The project offered profound insights into the nuances of financial data and the capabilities of machine learning.

1. **Behavior of Financial Data**:
    - Stock prices exhibit non-linear patterns, seasonal trends, and abrupt fluctuations influenced by news, economic events, and market sentiment. Recognizing these patterns was essential for preprocessing and feature selection.
    - Lagging indicators like moving averages provided useful context for predicting future movements, highlighting the importance of feature engineering.
2. **Modeling Insights**:
    - LSTM models excelled in capturing temporal dependencies in sequential data. Their ability to retain long-term memory was particularly effective for time-series forecasting compared to traditional methods like ARIMA.
    - Model evaluation required careful consideration of both training and validation errors. Low training error coupled with high validation error indicated overfitting, while balanced errors suggested effective generalization.
3. **Visualization and Interpretation**:
    - Visualizing predictions alongside actual prices provided immediate feedback on model performance. Tools like Plotly enabled the creation of interactive and informative charts, facilitating stakeholder communication.
4. **Applicability of Machine Learning**:
    - While machine learning provided robust predictions, it was not infallible. Combining these methods with domain knowledge, risk assessment, and other analytical tools would lead to more reliable decision-making.

8.4 Lessons Learned

1. **Iterative Development**: Machine learning projects benefit from iterative improvements. Starting with simple models and gradually increasing complexity ensured steady progress and minimized errors.
2. **Adaptability**: Flexibility in approach, such as switching to a different model architecture when needed, was crucial for overcoming technical roadblocks.
3. **Importance of Metrics**: Metrics must align with project goals. For example, while minimizing MSE improved accuracy, incorporating MAE provided a complementary perspective on performance.
4. **Collaboration**: Engaging with peers, seeking feedback, and referring to academic and industry literature enriched the project's quality and outcomes.
5. **Ethical Considerations**: Predictions based on historical data must be used responsibly, acknowledging their limitations and avoiding unwarranted confidence in accuracy.

9. Conclusion

This project has successfully demonstrated the implementation of LSTM models for stock price prediction, showcasing their ability to capture complex temporal dependencies and non-linear patterns inherent in financial time-series data. By leveraging historical stock prices and incorporating interactive visualization tools, the study not only provided accurate predictions but also made the results accessible and interpretable for users. The use of LSTMs emphasized their significance in financial forecasting, as they effectively modelled sequential data and adapted to dynamic stock market trends. The achievements of this project pave the way for further exploration and improvements, such as integrating additional financial and fundamental indicators, experimenting with alternative deep learning architectures like GRUs and Transformers, and incorporating real-time data for live prediction capabilities. These advancements would enhance the robustness, accuracy, and applicability of the model, making it a valuable tool for both individual and institutional investors. While this study highlights the potential of LSTM models, it also emphasizes the need for continuous refinement and innovation to address the ever-changing complexities of the financial market, ultimately contributing to more informed and strategic decision-making.

10. References

Althelaya, K. A., El-Alfy, E. S. M., & Mohammed, S. (2018), Evaluation of bidirectional LSTM for short-and long-term stock market prediction. In *2018 9th international conference on information and communication systems (ICICS)* (pp. 151-156). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/8355458

Baek, Y., & Kim, H. Y. (2018), ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, *113*, 457-480. Available at : https://www.sciencedirect.com/science/article/abs/pii/S0957417418304342

Bathla, G. (2020) ,Stock Price prediction using LSTM and SVR. In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 211-214). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/9315800

Bhandari, H. N., Rimal, B., Pokhrel, N. R., Rimal, R., Dahal, K. R., & Khatri, R. K. (2022) , Predicting stock market index using LSTM. *Machine Learning with Applications*, *9*, 100320. Available at : https://www.sciencedirect.com/science/article/pii/S2666827022000378

Borovkova, S., & Tsiamas, I. (2019), An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*, *38*(6), 600-619. Available at : https://onlinelibrary.wiley.com/doi/full/10.1002/for.2585

Ding, G., & Qin, L. (2020), Study on the prediction of stock price based on the associated network model of LSTM. *International Journal of Machine Learning and Cybernetics*, *11*(6), 1307-1317. Available at : https://link.springer.com/article/10.1007/s13042-019-01041-1

Eapen, J., Bein, D., & Verma, A. (2019), Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)* (pp. 0264-0270). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/8666592

Gao, S. E., Lin, B. S., & Wang, C. M. (2018) ,Share price trend prediction using CRNN with LSTM structure. In *2018 International Symposium on Computer, Consumer and Control (IS3C)* (pp. 10-13). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/8645025

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016), LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222-2232. Available at : https://ieeexplore.ieee.org/abstract/document/7508408

Khare, K., Darekar, O., Gupta, P., & Attar, V. Z. (2017) ,Short term stock price prediction using deep learning. In *2017 2nd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)* (pp. 482-486). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/8256643

Li, Z., Yu, H., Xu, J., Liu, J., & Mo, Y. (2023), Stock market analysis and prediction using LSTM: A case study on technology stocks. *Innovations in Applied Engineering and Technology*, 1-6. Available at : https://ojs.sgsci.org/journals/iaet/article/view/162

Liu, S., Liao, G., & Ding, Y. (2018), Stock transaction prediction modeling and analysis based on LSTM. In *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (pp. 2787-2790). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/8398183/

Lu, W., Li, J., Wang, J., & Qin, L. (2021), A CNN-BiLSTM-AM method for stock price prediction. *Neural Computing and Applications*, *33*(10), 4741-4753. Available at : https://link.springer.com/article/10.1007/s00521-020-05532-z

Mahadik, A., Vaghela, D., & Mhaisgawali, A. (2021) ,Stock price prediction using lstm and arima. In *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)* (pp. 1594-1601). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/9532655

Mehtab, S., & Sen, J. (2020), Stock price prediction using CNN and LSTM-based deep learning models. In *2020 International Conference on Decision Aid Sciences and Application (DASA)* (pp. 447-453). IEEE. 2024. Available at : https://ieeexplore.ieee.org/abstract/document/9317207

Moghar, A., & Hamiche, M. (2020), Stock market prediction using LSTM recurrent neural network. *Procedia computer science*, *170*, 1168-1173. Available at : https://www.sciencedirect.com/science/article/pii/S1877050920304865

Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., Salwana, E., & S, S. (2020), Deep learning for stock market prediction. *Entropy*, *22*(8), 840. Available at : https://www.mdpi.com/1099-4300/22/8/840

Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. (2017), Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)* (pp. 1419-1426). Ieee. Available at : https://ieeexplore.ieee.org/abstract/document/7966019

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017), Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643-1647). IEEE. 2024. Available at : https://ieeexplore.ieee.org/abstract/document/8126078

Sherstinsky, A. (2020), Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. Available at : https://www.sciencedirect.com/science/article/abs/pii/S0167278919305974

Sunny, M. A. I., Maswood, M. M. S., & Alharbi, A. G. (2020, October) ,Deep learning-based stock price prediction using LSTM and bi-directional LSTM model. In *2020 2nd novel intelligent and leading emerging sciences conference (NILES)* (pp. 87-92). IEEE. Available at : https://ieeexplore.ieee.org/abstract/document/9257950