STOCK PREDICTION USING LSTM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

MANOJ SREENIVAS [RA2111003010153]
NALLURI RAGHUNANDANA TEJA [RA2111003010141]
NISHAANTH [RA2111003010144]
PRANAV N [RA2111003010158]

Under the guidance of
Dr. LUBIN BALASUBRAMANIAN

Assistant Professor,

Department of Computer Science and Engineering

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled " **STOCK PREDICTION USING LSTM**" is the bonafide work of **Manoj Sreenivas (RA2111003010153),Nalluri Raghunandana Teja (RA2111003010141), Nishaanth (RA2111003010144), Pranav.N (RA2111003010158)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

FACULTY INCHARGE	**HEAD OF DEPARTMENT**
Dr. Lubin Balasubramanian	Dr.M.Pushpalatha
Assistant Professor	Professor & Head
Department of Computing Technologies	Department of Computing Technologies

ABSTRACT

This project explores the application of Long Short-Term Memory (LSTM) neural networks for stock price prediction, addressing the challenges posed by the dynamic and complex nature of financial markets. Historical stock price data, enriched with relevant financial indicators, is preprocessed and fed into a multi-layered LSTM architecture designed to capture temporal dependencies effectively. Through meticulous training and hyperparameter tuning, the LSTM model learns to forecast future stock prices. Evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are employed to assess the model's performance. Results indicate that the LSTM model demonstrates competitive predictive accuracy compared to baseline models and traditional forecasting methods, offering valuable insights for investors in navigating the volatile stock market landscape. Future research may delve into further optimization of the LSTM architecture and incorporation of additional features to enhance prediction robustness.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
SVM	Support Vector Machine
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
MSE	Mean Square Error
MA	Moving Average
AM	Attention Mechanism

CHAPTER 1
INTRODUCTION

Predicting stock market trends is crucial for investors, traders, and financial institutions seeking to optimize investment strategies and mitigate risks. However, the dynamic and volatile nature of financial markets poses significant challenges to accurate prediction. In recent years, advancements in machine learning, particularly in deep learning techniques such as Long Short-Term Memory (LSTM) networks, have shown promise in improving the accuracy of stock market predictions.

LSTM is a type of recurrent neural network (RNN) architecture specifically designed to model sequential data. Unlike traditional RNNs, LSTM networks utilize a more sophisticated memory cell structure that enables them to capture long-term dependencies and handle issues like the vanishing gradient problem. This makes LSTM particularly well-suited for time-series data analysis tasks such as stock market prediction.

One of the critical steps in building a successful stock market prediction model using LSTM is data preparation. This involves collecting historical stock market data from reliable sources such as Yahoo Finance or Quandl and preprocessing it to remove noise and irrelevant information. Common preprocessing techniques include data normalization, differencing, and feature engineering, which help improve the model's performance and generalization ability.

Building an LSTM-based stock market prediction model involves designing the network architecture, selecting appropriate hyperparameters, and training the model using historical data. The architecture of an LSTM network typically consists of multiple layers of LSTM cells interconnected in a sequential fashion. The choice of hyperparameters such as the number of LSTM layers, the size of the hidden state, and the learning rate significantly impacts the model's predictive performance.

During the training phase, the LSTM model learns to map input sequences of historical stock prices to output sequences representing future price movements. This process involves iteratively adjusting the model's parameters using optimization algorithms like stochastic gradient descent (SGD) or Adam. To prevent overfitting, techniques such as dropout regularization and early stopping are commonly employed.

Once the model is trained, it is essential to evaluate its performance using appropriate metrics. Common evaluation metrics for stock market prediction models include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). These metrics provide insights into the accuracy and reliability of the model's predictions and help assess its suitability for real-world applications.

After evaluating the model's performance, it's crucial to analyze the prediction results to understand its strengths, weaknesses, and potential areas for improvement. Visualizations comparing predicted and actual stock prices over time can provide valuable insights into the model's behavior and performance under different market conditions. Additionally, conducting sensitivity analyses and scenario testing can help assess the robustness of the model to various input parameters and market dynamics.

LSTM-based stock market prediction models have numerous real-world applications beyond individual investment decision-making. They are widely used in algorithmic trading systems, where automated trading strategies are executed based on real-time market predictions generated by LSTM models. Additionally, financial institutions leverage LSTM models for portfolio optimization, risk management, and hedging strategies to enhance returns and minimize exposure to market volatility.

CHAPTER 2

LITERATURE SURVEY

Lu, W., Li, J., Wang, J. et al.(2021). - A CNN-BiLSTM-AM method for stock price prediction:

The paper "A CNN-BiLSTM-AM Method for Stock Price Prediction" introduces a novel and innovative approach to forecasting stock prices by synergistically integrating three powerful components: Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and Attention Mechanism (AM). In this hybrid framework, each component plays a crucial role in enhancing the model's predictive capabilities. Convolutional Neural Networks (CNNs) are employed for their ability to effectively extract intricate patterns and features from historical stock data. By leveraging the hierarchical structure of CNN layers, the model can capture spatial relationships within the data, enabling it to identify significant market trends and patterns. Bidirectional Long Short-Term Memory (BiLSTM) networks are utilized to capture the long-term dependencies inherent in sequential data such as stock prices. By processing the input sequence both forwards and backwards, BiLSTMs excel at capturing complex temporal relationships, thereby enabling the model to make more accurate predictions based on historical price movements. Furthermore, the incorporation of an Attention Mechanism (AM) allows the model to dynamically focus on the most relevant features within the input sequence. By assigning varying degrees of importance to different temporal features, the attention mechanism enhances the model's ability to discern subtle patterns and trends that may significantly impact future stock prices. By combining these three components into a unified framework, the proposed method aims to provide a robust and effective solution for stock price prediction. The synergistic integration of CNNs, BiLSTMs, and attention mechanisms enables the model to leverage the complementary strengths of each component, resulting in improved prediction accuracy and robustness. This approach holds great promise for investors and financial analysts seeking more accurate and reliable forecasting tools in today's dynamic and volatile markets.

Yu, P., Yan, X.(2020) - Stock price prediction based on deep neural networks.

The paper titled "Stock Price Prediction Based on Deep Neural Networks" authored by Yu and Yan offers a meticulous investigation into the application of deep neural networks (DNNs) for the purpose of stock price prediction. The study represents a significant contribution to the field of financial forecasting, as it delves deep into the intricacies of employing cutting-edge machine learning techniques to tackle the inherently complex and volatile nature of stock markets. Throughout their research, Yu and Yan meticulously explore the potential of DNNs in capturing and learning from the vast array of patterns and dynamics present within historical stock data. By harnessing the hierarchical structure of deep neural networks, the model is adept at discerning nuanced relationships and dependencies that may not be readily apparent through traditional statistical methods. Moreover, the authors carefully assess various architectures and configurations of deep neural networks to identify the most effective approach for stock price prediction. Through rigorous experimentation and analysis, they elucidate the strengths and limitations of different DNN architectures, providing valuable insights into their applicability and performance in real-world financial forecasting tasks. By presenting their findings in a comprehensive and detailed manner, Yu and Yan offer readers a deeper understanding of the capabilities and nuances of employing deep neural networks for stock price prediction. Their research not only advances the state-of-the-art in financial forecasting but also provides practical guidance for investors and financial professionals seeking to leverage advanced machine learning techniques to gain a competitive edge in the ever-evolving landscape of financial markets.

Title	Author	Methods	Evaluation Metrics	Observation
A CNN-BiLSTM-AM method for stock price prediction	Wenjie Lu, Jiazheng Li, Jingyang WangLele Qin	MLP, CNN,RNN, LSTM, BiLSTM, CNN-LSTM,CNN-BiLSTM,BiLSTM-AM, CNN-BiLSTM-AM	MAE, RMSE, R2	The experimental results show that the CNN-BiLSTM-AM has the highest prediction accuracy and the best performance compared to other proposed models.
Stock price prediction based on deep neural networks	Pengfei Yu, Xuesong Yan	Deep Neural Network	MRSE, MAPE, R	DNN-based prediction model displaysa higher prediction capacity than the other models

Summary:

The papers by Lu et al. and Yu and Yan present innovative approaches to stock price prediction using deep learning techniques. Lu et al. propose a novel method that combines Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and Attention Mechanism (AM) to enhance prediction accuracy. Their model leverages CNNs for feature extraction, BiLSTMs for capturing long-term dependencies, and attention mechanisms for focusing on relevant temporal features. On the other hand, Yu and Yan explore the application of deep neural networks (DNNs) for stock price prediction, investigating various architectures and configurations to model and forecast stock prices based on historical data. Both papers contribute significantly to the field of financial forecasting, offering valuable insights and methodologies that can aid investors and financial analysts in making informed decisions in dynamic market environments.

CHAPTER 3
EXISTING SYSTEM

Random Forests, a popular ensemble learning method, constructs numerous decision trees to predict stock movements based on historical data. Gradient Boosting Machines (GBM) sequentially build models to correct errors made by preceding ones, effectively capturing complex relationships in stock data. Support Vector Machines (SVM) find hyperplanes in high-dimensional spaces to separate classes, aiding in stock prediction by identifying patterns in historical data.

Bayesian Networks, probabilistic graphical models, incorporate various factors and their dependencies, providing a means to model uncertainty in stock prediction. Time-series analysis techniques like ARIMA and SARIMA capture trends, seasonality, and noise in historical price data, enabling forecasts of future stock prices. Ensemble learning methods, such as bagging, boosting, and stacking, combine the predictions of multiple models like decision trees, SVMs, or neural networks, enhancing prediction accuracy by leveraging diverse model strengths.

Each model has its own set of advantages and limitations, with their effectiveness influenced by factors like data characteristics and feature selection. Successful stock prediction often involves careful consideration of these models alongside thorough feature engineering to extract meaningful insights from the data.

Advantages:

- Random Forests: They are robust to overfitting, handle high-dimensional data well, and provide feature importance scores, aiding in interpretability.
- Gradient Boosting Machines: GBM can handle complex interactions between features and have high predictive accuracy, often outperforming other models.
- Support Vector Machines: SVMs are effective in high-dimensional spaces, suitable for datasets with many features, and can handle non-linear relationships with appropriate kernel functions.
- Bayesian Networks: They offer probabilistic reasoning, aiding in uncertainty modeling and decision-making, and allow for transparent representation of dependencies between variables.

- Time-Series Analysis (ARIMA, SARIMA): These models capture temporal patterns, trends, and seasonality in data, making them suitable for time-series forecasting tasks like stock prediction.

Disadvantages:

- Random Forests: They may not perform well with highly imbalanced data, struggle with extrapolation outside the training range, and can be computationally expensive with large datasets.
- Gradient Boosting Machines: GBM training can be time-consuming, especially with large datasets, and they are sensitive to hyperparameter tuning.
- Support Vector Machines: SVMs can be sensitive to the choice of kernel function and parameters, require careful preprocessing of data, and might not scale well to large datasets.
- Bayesian Networks: Constructing a Bayesian Network requires prior knowledge or assumptions about variable dependencies, and they can become computationally expensive as the number of variables increases.
- Time-Series Analysis (ARIMA, SARIMA): These models may struggle with capturing non-linear relationships and may not perform well with data exhibiting sudden changes or irregular patterns.

Summary:

Various models are employed in stock prediction, each with its own set of advantages and disadvantages. Random Forests provide robustness to overfitting and interpretability but may struggle with imbalanced data. Gradient Boosting Machines offer high accuracy but require extensive tuning and training time. Support Vector Machines handle high-dimensional data but are sensitive to kernel choice and preprocessing. Bayesian Networks aid in uncertainty modeling but require prior knowledge of dependencies. Time-Series Analysis captures temporal patterns but may struggle with non-linear relationships. Ensemble Learning combines model strengths but can be complex to implement. Careful consideration of these factors is crucial in selecting the most suitable model for effective stock prediction.

CHAPTER 5
PROBLEM STATEMENT

Predicting stock prices accurately is a challenging task due to the inherent complexity and volatility of financial markets. Traditional statistical methods often struggle to capture the non-linear patterns and dependencies present in historical stock data. As a result, investors, traders, and financial analysts face difficulties in making informed decisions about buying, selling, or holding stocks.

4.1 Problem Definition

The problem faced by stakeholders in the financial domain is the unreliable and inconsistent nature of stock price predictions. Given the dynamic nature of financial markets, accurately forecasting stock prices requires advanced analytical techniques that can capture complex patterns and trends in historical data. Traditional methods often fall short in providing reliable predictions, leading to suboptimal investment decisions and missed opportunities for maximizing returns.

4.2 Problem Description

Investors, traders, and financial analysts encounter significant challenges in predicting stock prices with precision and confidence. The process involves analyzing vast amounts of historical market data, including price movements, trading volumes, and external factors such as economic indicators and geopolitical events. Traditional statistical models often struggle to account for the non-linear relationships and sudden shifts in market sentiment, resulting in inaccurate predictions and increased financial risks.

Additionally, the inherent volatility and uncertainty in financial markets further exacerbate the difficulty of accurately forecasting stock prices. Factors such as sudden market shocks, regulatory changes, and unexpected news events can cause rapid fluctuations in stock prices, making it challenging for predictive models to adapt quickly and provide reliable forecasts.

A solution that leverages advanced machine learning techniques, such as Long Short-Term Memory (LSTM) neural networks, to effectively capture the complex patterns and dependencies in historical stock data is needed to address this problem. By providing accurate and timely predictions, such a solution can empower investors, traders, and financial analysts to make more informed decisions and navigate volatile market conditions with confidence.

CHAPTER 5
METHODOLOGY

The methodology for utilizing LSTM in stock market prediction involves gathering and preprocessing historical stock price data, selecting and engineering relevant features such as technical indicators and market sentiment data, organizing the data into sequential input-output pairs representing windows of historical data, designing and training an LSTM model while optimizing hyperparameters to prevent overfitting, evaluating the model's performance on a testing set using metrics like Mean Absolute Error, and finally deploying the trained LSTM model for real-time prediction while continuously monitoring its performance to ensure accuracy and effectiveness in capturing stock market trends and movements.

Goals:

1. Enhance Predictive Accuracy: Utilize LSTM networks to develop models that can accurately forecast future stock prices or trends. The primary goal is to improve prediction accuracy compared to traditional statistical methods or other machine learning approaches.

2. Capture Temporal Dependencies: Leverage the ability of LSTM networks to capture long-term dependencies in sequential data. The goal is to exploit the temporal relationships present in historical stock price data to make more informed predictions about future price movements.

3. Handle Non-linear Relationships: Take advantage of LSTM's capability to model complex, non-linear relationships in data. The goal is to capture the intricate dynamics of financial markets, where price movements are influenced by a multitude of factors and exhibit non-linear patterns.

4. Adaptability to Irregular Time Intervals: Utilize LSTM networks to handle irregularly spaced time intervals in stock market data. The goal is to develop models that can effectively process data with varying time gaps between observations, accommodating the asynchronous nature of financial market data.

5. Real-time Prediction: Develop LSTM models capable of making real-time predictions, enabling timely decision-making for traders and investors. The goal is to create systems that can provide up-to-date forecasts of stock prices, allowing users to react quickly to market changes.

Objectives:

1. Data Preparation: Gather and preprocess historical stock price data, ensuring it is formatted appropriately for input into the LSTM model. Objective: Ensure the data is clean, consistent, and aligned with the model's input requirements.

2. Model Development: Design and train LSTM architectures tailored to the characteristics of stock market data. Objective: Develop LSTM models that can effectively capture temporal dependencies and non-linear relationships present in the data.

3. Hyperparameter Optimization: Fine-tune model hyperparameters such as the number of LSTM units, layers, learning rate, and dropout rate to optimize performance. Objective: Identify the optimal configuration that maximizes prediction accuracy and generalization ability.

4. Validation and Testing: Evaluate the trained LSTM models on validation and testing datasets to assess their performance. Objective: Ensure that the models generalize well to unseen data and exhibit robust predictive capabilities.

5. Performance Monitoring: Monitor the performance of deployed LSTM models over time and assess their effectiveness in real-world scenarios. Objective: Continuously evaluate model performance and identify opportunities for improvement or recalibration as market conditions change.

6. Interpretability and Explainability: Enhance the interpretability of LSTM models to provide insights into the factors driving predictions. Objective: Develop methods to interpret model decisions and communicate them to users, increasing confidence in the predictions.

Methodology Steps:

1. Data Collection and Preprocessing:

- Gather historical stock price data from reliable sources such as financial databases or APIs like Yahoo Finance or Alpha Vantage.

- Preprocess the data to handle missing values, outliers, and inconsistencies. This may involve techniques like interpolation, normalization, or scaling to ensure uniformity and stability in the dataset.

2. Sequence Formation:

- Structure the preprocessed data into sequential input-output pairs suitable for LSTM training.

- Define a sequence length, which determines the number of past time steps used to predict the future stock price.

- Create overlapping sequences by sliding a window of fixed size over the historical data.

3. Data Splitting:

- Split the sequential data into training, validation, and testing sets. Typically, the majority of the data is allocated for training, while smaller portions are reserved for validation and testing to evaluate model performance.

4. Model Architecture Design:

- Define the architecture of the LSTM neural network, including the number of LSTM layers, the number of neurons in each layer, and any additional layers such as dropout or dense layers.

5. Model Training:

- Train the LSTM model using the training data prepared in step 3.

- Define a loss function (e.g., Mean Squared Error) and an optimization algorithm (e.g., Adam optimizer) to minimize the difference between predicted and actual stock prices.

- Specify the number of epochs (training iterations) and batch size for training the model.

- Monitor the model's performance on the validation set to prevent overfitting and adjust hyperparameters accordingly.

6. Model Evaluation:

- Evaluate the trained LSTM model on the testing set to assess its predictive performance.
- Calculate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), etc., to quantify the accuracy of the predictions.
- Visualize the predicted stock prices alongside the actual prices to qualitatively assess the model's performance.

7. Prediction:

- Prepare the input sequences for prediction by incorporating the latest available data.
- Generate predictions for future stock prices using the trained LSTM model.

8. Post-processing and Analysis:

- Analyze the predicted stock prices in the context of market trends, and other relevant factors.
- Adjust the model or incorporate additional features as necessary based on the analysis and feedback.

CHAPTER 6
SYSTEM ARCHITECTURE AND DESIGN

6.1 Architecture Overview

The system architecture for LSTM-based stock prediction involves several interconnected components: Firstly, historical stock data is collected and preprocessed, including feature engineering and normalization. This processed data is then fed into LSTM neural networks, which are designed to capture temporal dependencies in sequential data. The model is trained using a portion of the data, validated for performance, and fine-tuned through hyperparameter tuning. Evaluation metrics are calculated using a separate testing dataset to assess the model's accuracy. Once validated, the trained model is deployed for real-time predictions, integrated into a production environment, and continuously monitored and maintained for updates and improvements, forming a robust end-to-end system for stock prediction.

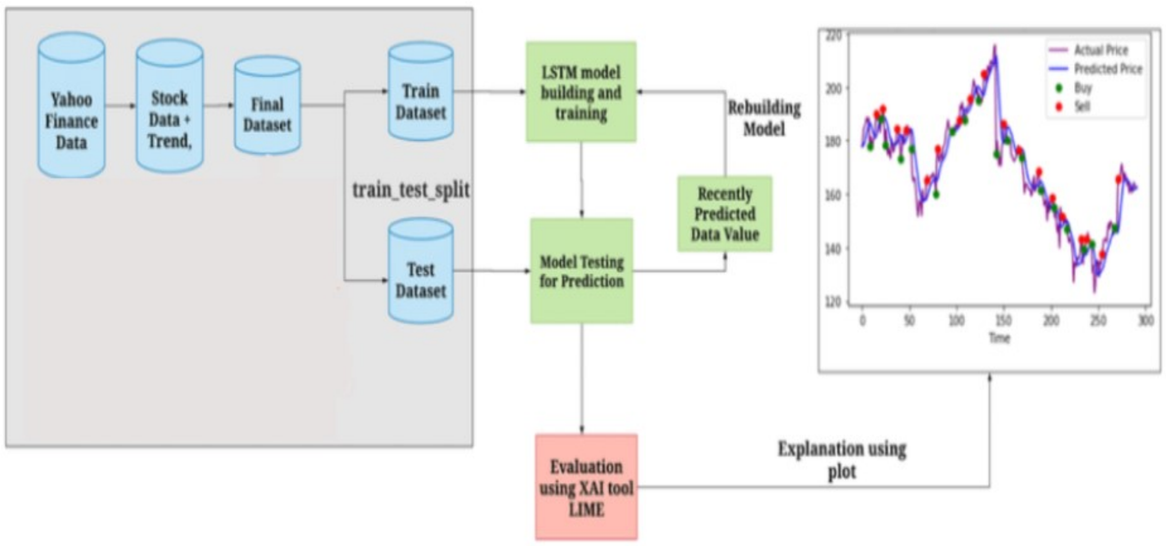

Fig 6.1.1: Architecture Diagram of project

This machine learning model architecture comprises two Long Short-Term Memory (LSTM) layers followed by dense layers, designed to learn and predict stock prices based on historical data. The input layer receives data shaped by the number of time steps in the input sequence and one feature per time step, assuming univariate input data. The first LSTM layer consists

of 128 LSTM units, capable of capturing long-term dependencies in sequential data.
With `return_sequences=True`, this layer provides the entire sequence of outputs for each input sequence, enabling subsequent layers to receive comprehensive LSTM outputs. The second LSTM layer, with 64 units, further processes the LSTM outputs, returning only the output at the last time step due to `return_sequences=False`. The dense layers following the LSTM layers, with 25 units in the first and a single unit in the output layer, facilitate learning complex patterns from the LSTM outputs and produce the final prediction, representing the predicted stock price.

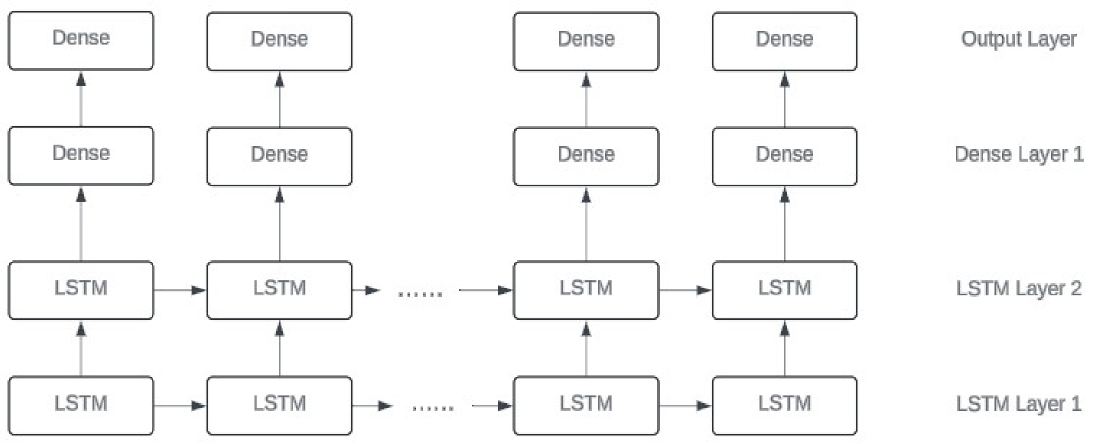

Fig 6.1.2: Architecture of Machine Learning Model

Through training, the model aims to minimize the disparity between predicted and actual stock prices, leveraging the sequential nature of LSTM units to capture temporal dependencies effectively and make informed predictions.

6.2 Description of Modules:

Pandas: Pandas is a versatile Python library used for data manipulation and analysis. It offers data structures like DataFrame and Series, along with functions to clean, transform, and analyze data efficiently. With its intuitive syntax and powerful capabilities, Pandas is widely used in data science and analysis tasks.

NumPy: NumPy is the fundamental package for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy is essential for scientific computing and forms the foundation for many other libraries in the Python ecosystem.

Matplotlib: Matplotlib is a comprehensive library for creating static, interactive, and animated visualizations in Python. It offers a wide range of plotting functions to generate various types of plots, including line plots, scatter plots, histograms, and more. Matplotlib is highly customizable and widely used for data visualization tasks in fields such as data science, engineering, and finance.

Seaborn: Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Seaborn simplifies the process of generating complex visualizations like heatmaps, violin plots, and pair plots by providing built-in functions with sensible default settings.

yfinance: yfinance is a Python library that facilitates fetching historical market data from Yahoo Finance. It allows users to retrieve stock prices, trading volumes, and other financial data easily. yfinance is commonly used in financial analysis, algorithmic trading, and quantitative research projects.

Datetime: Datetime is a module in Python's standard library that provides classes for manipulating dates and times. It offers functionalities to create, format, and perform arithmetic operations on dates and times. Datetime is essential for working with time-series data and handling temporal information in various applications.

Keras: Keras is a high-level neural networks API written in Python and capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit. It allows for easy and fast experimentation with deep learning models, providing a user-friendly interface for building and training neural networks. Keras is widely adopted in both research and industry for tasks such as image classification, natural language processing, and reinforcement learning.

CHAPTER 7
IMPLEMENTATION AND TESTING

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
from pandas_datareader.data import DataReader
import yfinance as yf
from pandas_datareader import data as pdr
yf.pdr_override()
from datetime import datetime
tech_list = ['AAPL', 'GOOG', 'MSFT', 'AMZN']
end = datetime.now()
start = datetime(end.year - 1, end.month, end.day)
for stock in tech_list:
    globals()[stock] = yf.download(stock, start, end)
company_list = [AAPL, GOOG, MSFT, AMZN]
company_name = ["APPLE", "GOOGLE", "MICROSOFT", "AMAZON"]
for company, com_name in zip(company_list, company_name):
    company["company_name"] = com_name
df = pd.concat(company_list, axis=0)
df.tail(10)
AAPL.describe()
AAPL.info()
plt.figure(figsize=(15, 10))
plt.subplots_adjust(top=1.25, bottom=1.2)
for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Adj Close'].plot()
    plt.ylabel('Adj Close')
    plt.xlabel(None)
    plt.title(f"Closing Price of {tech_list[i - 1]}")
plt.tight_layout()
plt.figure(figsize=(15, 10))
plt.subplots_adjust(top=1.25, bottom=1.2)
for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Volume'].plot()
    plt.ylabel('Volume')
    plt.xlabel(None)
    plt.title(f"Sales Volume for {tech_list[i - 1]}")
plt.tight_layout()
ma_day = [10, 20, 50]
for ma in ma_day:
    for company in company_list:
        column_name = f"MA for {ma} days"
        company[column_name] = company['Adj Close'].rolling(ma).mean()
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)
AAPL[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[0,0])
axes[0,0].set_title('APPLE')
GOOG[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50
days']].plot(ax=axes[0,1])
axes[0,1].set_title('GOOGLE')
MSFT[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50 days']].plot(ax=axes[1,0])
axes[1,0].set_title('MICROSOFT')
AMZN[['Adj Close', 'MA for 10 days', 'MA for 20 days', 'MA for 50
days']].plot(ax=axes[1,1])
axes[1,1].set_title('AMAZON')
fig.tight_layout()
for company in company_list:
    company['Daily Return'] = company['Adj Close'].pct_change()
```

```
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(15)
AAPL['Daily Return'].plot(ax=axes[0,0], legend=True, linestyle='--', marker='o')
axes[0,0].set_title('APPLE')
GOOG['Daily Return'].plot(ax=axes[0,1], legend=True, linestyle='--', marker='o')
axes[0,1].set_title('GOOGLE')
MSFT['Daily Return'].plot(ax=axes[1,0], legend=True, linestyle='--', marker='o')
axes[1,0].set_title('MICROSOFT')
AMZN['Daily Return'].plot(ax=axes[1,1], legend=True, linestyle='--', marker='o')
axes[1,1].set_title('AMAZON')
fig.tight_layout()
plt.figure(figsize=(12, 9))
for i, company in enumerate(company_list, 1):
    plt.subplot(2, 2, i)
    company['Daily Return'].hist(bins=50)
    plt.xlabel('Daily Return')
    plt.ylabel('Counts')
    plt.title(f'{company_name[i - 1]}')
plt.tight_layout()
closing_df = pdr.get_data_yahoo(tech_list, start=start, end=end)['Adj Close']
tech_rets = closing_df.pct_change()
tech_rets.head()
sns.jointplot(x='GOOG', y='GOOG', data=tech_rets, kind='scatter', color='seagreen')
sns.jointplot(x='GOOG', y='MSFT', data=tech_rets, kind='scatter')
sns.pairplot(tech_rets, kind='reg')
return_fig = sns.PairGrid(tech_rets.dropna())
return_fig.map_upper(plt.scatter, color='purple')
return_fig.map_lower(sns.kdeplot, cmap='cool_d')
return_fig.map_diag(plt.hist, bins=30)
returns_fig = sns.PairGrid(closing_df)
returns_fig.map_upper(plt.scatter,color='purple')
returns_fig.map_lower(sns.kdeplot,cmap='cool_d')
returns_fig.map_diag(plt.hist,bins=30)
```

```
plt.figure(figsize=(12, 10))
plt.subplot(2, 2, 1)
sns.heatmap(tech_rets.corr(), annot=True, cmap='summer')
plt.title('Correlation of stock return')
plt.subplot(2, 2, 2)
sns.heatmap(closing_df.corr(), annot=True, cmap='summer')
plt.title('Correlation of stock closing price')
rets = tech_rets.dropna()
area = np.pi * 20
plt.figure(figsize=(10, 8))
plt.scatter(rets.mean(), rets.std(), s=area)
plt.xlabel('Expected return')
plt.ylabel('Risk')
for label, x, y in zip(rets.columns, rets.mean(), rets.std()):
    plt.annotate(label, xy=(x, y), xytext=(50, 50), textcoords='offset points', ha='right',
va='bottom',
                 arrowprops=dict(arrowstyle='-', color='blue', connectionstyle='arc3,rad=-0.3'))
df = pdr.get_data_yahoo('AAPL', start='2012-01-01', end=datetime.now())
plt.figure(figsize=(16,6))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
data = df.filter(['Close'])
dataset = data.values
training_data_len = int(np.ceil( len(dataset) * .95 ))
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
train_data = scaled_data[0:int(training_data_len), :]
x_train = []
y_train = []
```

```
for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()
x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
from keras.models import Sequential
from keras.layers import Dense, LSTM
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=1, epochs=1)
test_data = scaled_data[training_data_len - 60: , :]
x_test = []
y_test = dataset[training_data_len:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])
x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1 ))
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(((predictions - y_test) ** 2)))
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
plt.figure(figsize=(16,6))
plt.title('Model')
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
plt.show()
```

CHAPTER 8
SCREENSHOTS AND RESULTS

8.1 Data Acquisition

The graph displays the adjusted closing prices of four major technology companies (Apple, Google, Microsoft, and Amazon) over the past year. Each company's stock prices are shown in a separate subplot within a 2x2 grid.

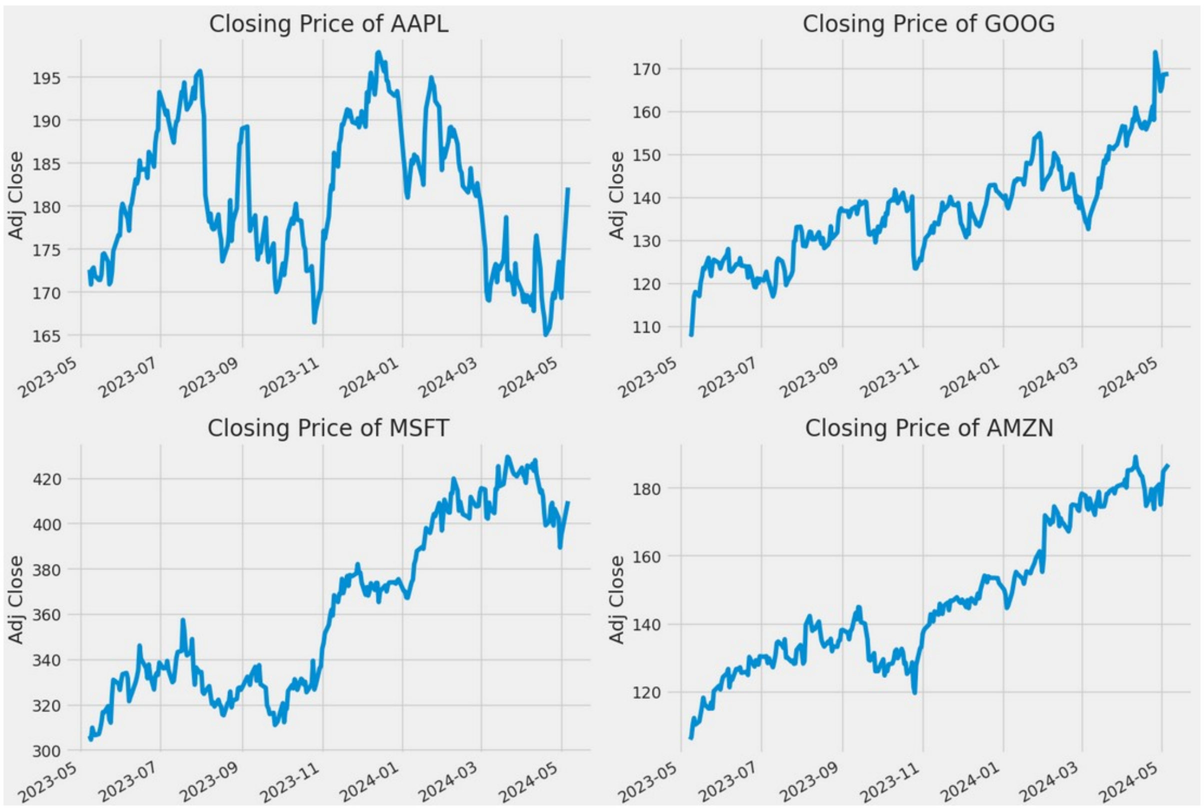

Fig 8.1: Data from YFinance

The x-axis represents time, while the y-axis represents the adjusted closing price of the stock. By visually comparing the plots, you can analyze the relative performance of these companies' stocks over the specified time period.

8.2 Exploratory Data Analysis

The graph exhibits exploratory data analysis (EDA) techniques such as moving averages and correlation mapping to gain insights into the stock prices of four prominent technology companies (Apple, Google, Microsoft, and Amazon).

8.2.1 Moving Average Extrapolation

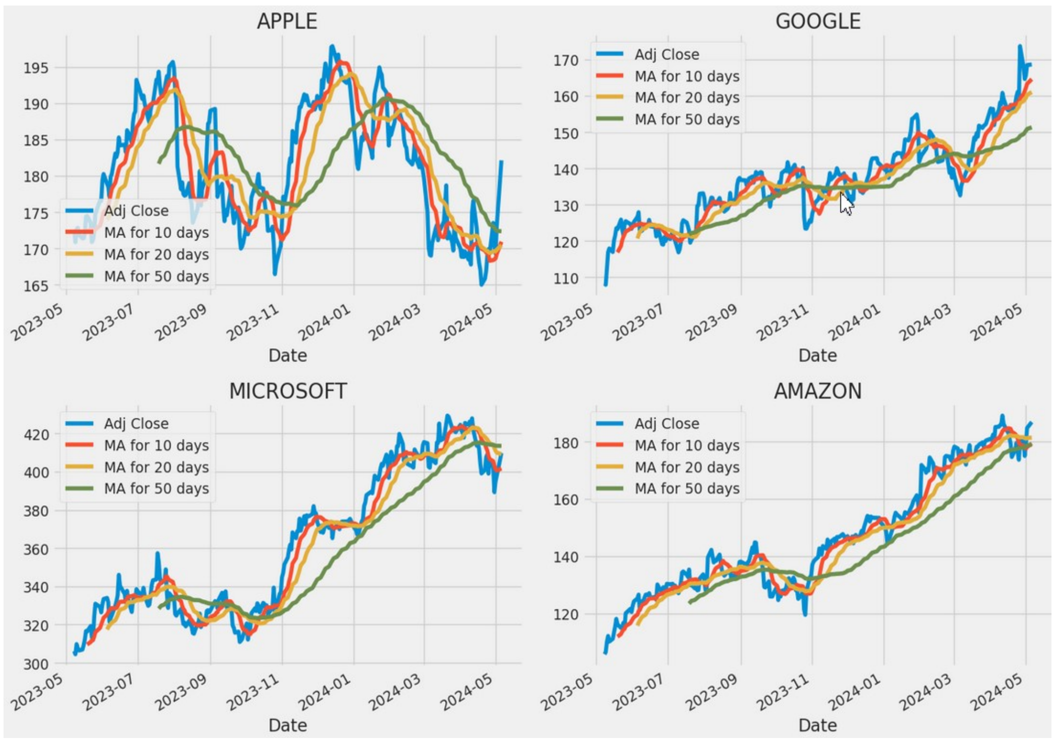

Fig 8.2.1: Moving Average of Data

Moving Averages: One common EDA technique used in financial analysis is the calculation and visualization of moving averages. A moving average smoothens out price data by creating a constantly updated average price over a specified time period. By plotting moving averages alongside actual stock prices, analysts can identify trends and patterns more easily, helping to filter out noise and fluctuations in the data.

8.2.2 Regression Analysis

Regression analysis is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It aims to understand how changes in the independent variables are associated with changes in the dependent variable.

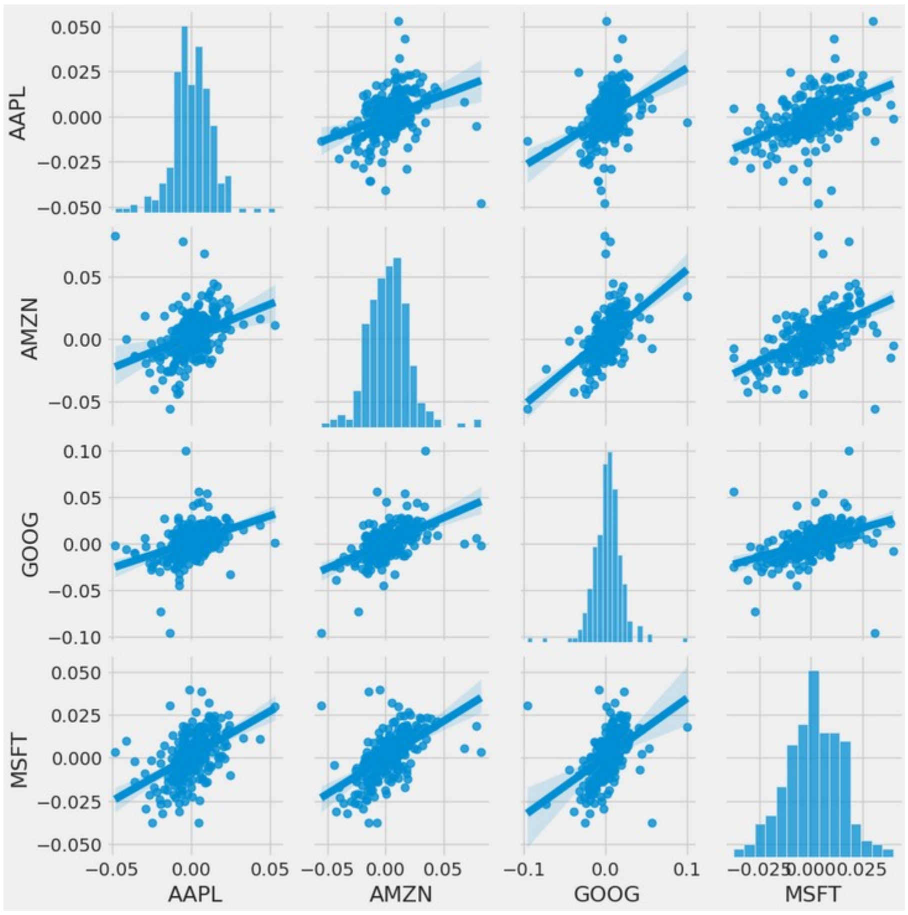

Fig 8.2.2: Regression Analysis of Data

Linear regression models the relationship between two variables by fitting a linear equation to observed data points. In this context, we can use linear regression to understand how changes in the price of one company's stock (independent variable) relate to changes in the price of another company's stock (dependent variable). A positive coefficient suggests a positive relationship, while a negative coefficient indicates an inverse relationship.

8.2.3 Risk Return Analysis

Risk-return analysis evaluates the trade-off between potential gains and the level of uncertainty or volatility inherent in an investment. It helps investors determine if the potential return justifies the associated risk, aiding in portfolio management and decision-making. By balancing risk and return, investors aim to optimize their investment strategy to achieve their financial goals while managing exposure to potential losses.

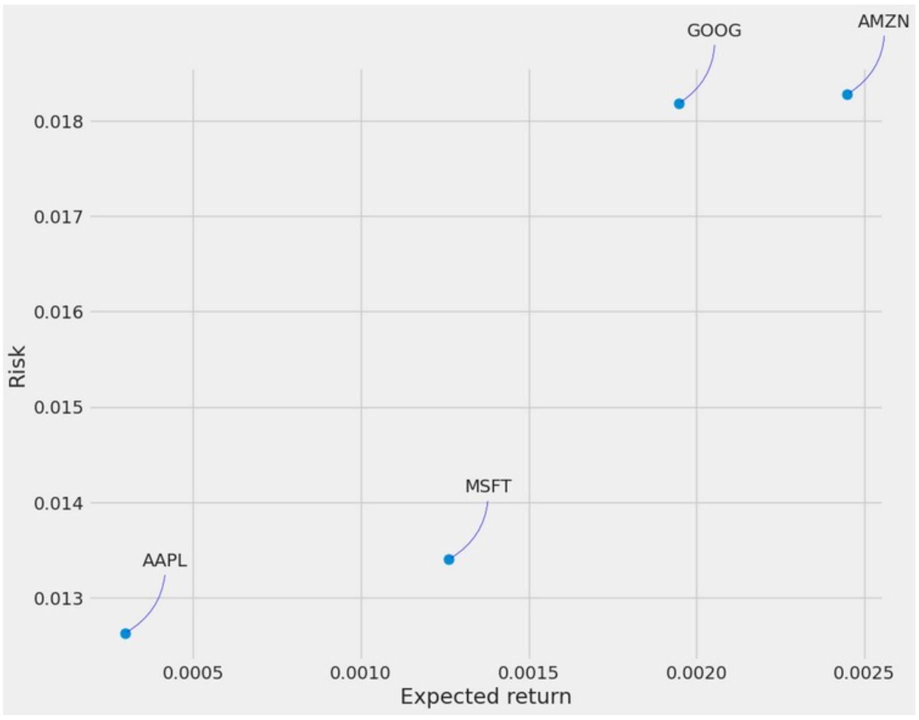

Fig 8.2.3: Risk Return Analysis of Data

In stock market contexts, risk signifies the degree of unpredictability or fluctuation in investment returns, reflecting the potential for losses. Return denotes the profit or loss realized from an investment within a defined timeframe, serving as a measure of its financial performance. Understanding the relationship between risk and return is crucial for investors to gauge the trade-offs involved in pursuing higher potential gains against the likelihood of incurring losses.

8.3 Model Performance

The graph illustrates the predictive performance of a model for forecasting stock prices. By contrasting the actual closing prices from both the training and validation datasets with the model's predicted prices, it enables a direct comparison of the model's accuracy in capturing stock price movements. Investors can use this visual representation to assess the reliability of the model's predictions and make informed decisions regarding their investment strategies.

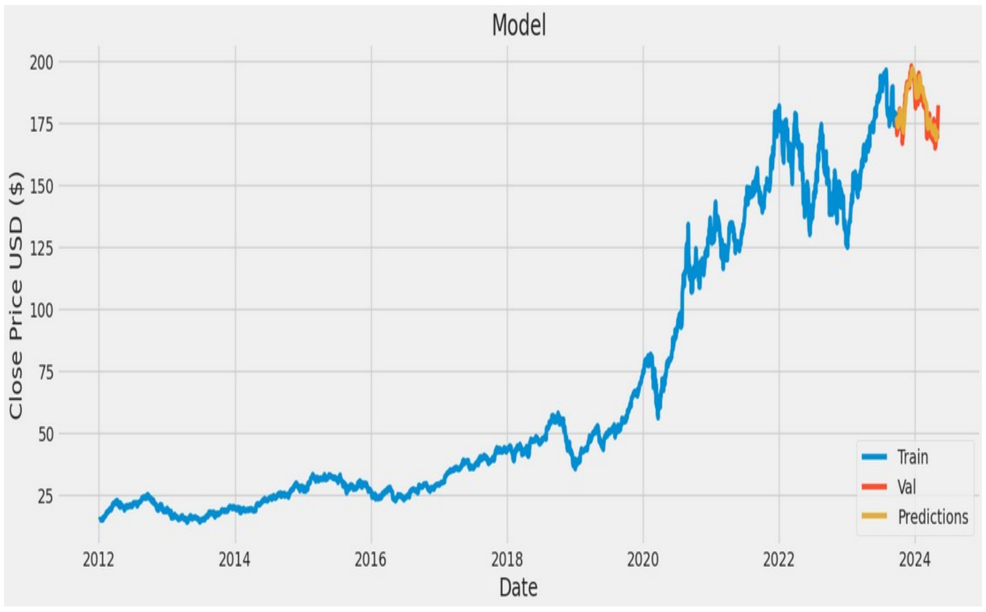

Fig 8.3: Model Performance

Additionally, discrepancies between the predicted and actual prices highlight areas for model refinement, guiding efforts to improve the model's predictive capabilities. Ultimately, the graph serves as a valuable tool for evaluating the effectiveness of the predictive model in the context of stock market analysis and decision-making.

CHAPTER 9
CONCLUSION AND FUTURE ENHANCEMENTS

8.1 Conclusion

In conclusion, LSTM models represent a powerful tool in the realm of stock market prediction, offering the capability to analyze and learn from intricate patterns embedded within historical price data. While the utilization of LSTM networks has showcased promising outcomes in certain scenarios, it's imperative to acknowledge the inherent volatility and unpredictability intrinsic to financial markets. Despite the advancements in machine learning methodologies, the quest for precise and reliable stock market predictions remains an ongoing challenge.

The efficacy of LSTM models in stock market prediction hinges on various factors, including the quality and quantity of input data, the feature engineering process, and the robustness of the model architecture. Furthermore, the dynamic and evolving nature of financial markets necessitates continuous adaptation and refinement of predictive models to account for emerging trends and unforeseen events.

Moreover, it's essential to exercise caution and skepticism when interpreting the results generated by LSTM models or any predictive algorithm in the context of stock market forecasting. While these models can offer valuable insights and aid in decision-making processes, they should be viewed as one of many tools in the arsenal of investors and financial analysts, rather than a definitive oracle of future market behavior.

In essence, while LSTM models hold promise in deciphering the complexities of stock market dynamics, they are not devoid of limitations and uncertainties. The pursuit of more accurate and reliable stock market predictions remains an ongoing endeavor, necessitating interdisciplinary collaboration and continuous innovation. Ultimately, the integration of advanced machine learning techniques with domain expertise and sound financial principles offers the potential to unlock new avenues for understanding and navigating the intricacies of the stock market landscape.

8.2 Future Enhancements

Future enhancements for stock market prediction using LSTM models could involve several avenues:

1. **Incorporating External Data Sources:** Integrating additional data sources such as macroeconomic indicators, news sentiment analysis, social media trends, and geopolitical events could provide a more comprehensive view of market dynamics and enhance prediction accuracy.
2. **Feature Engineering:** Experimenting with novel feature engineering techniques to extract more informative features from raw data, such as technical indicators, market sentiment scores, and derived metrics, could improve model performance.
3. **Attention Mechanisms:** Investigating the integration of attention mechanisms within LSTM architectures to prioritize relevant information and focus on salient features within the input sequence could improve the model's ability to capture long-term dependencies and subtle patterns in the data.
4. **Adaptive Learning Strategies:** Developing adaptive learning strategies that dynamically adjust model parameters and learning rates based on evolving market conditions could enhance the model's adaptability to changing environments and improve prediction performance over time.
5. **Uncertainty Estimation:** Incorporating uncertainty estimation techniques, such as Bayesian inference or Monte Carlo dropout, to quantify prediction uncertainties and provide probabilistic forecasts could offer valuable insights for risk management and decision-making.
6. **Interpretability and Explainability:** Enhancing the interpretability and explainability of LSTM models by leveraging techniques such as attention visualization, feature importance analysis, and model-agnostic explanation methods could increase trust and usability in real-world applications.

REFERENCES

1. Lu, W., Li, J., Wang, J. et al. A CNN-BiLSTM-AM method for stock price prediction. Neural Comput & Applic 33, 4741–4753 (2021). https://doi.org/10.1007/s00521-020-05532-z
2. Yu, P., Yan, X. Stock price prediction based on deep neural networks. Neural Comput & Applic 32, 1609–1628 (2020). https://doi.org/10.1007/s00521-019-04212-x
3. Rebwar M. Nabi, Soran AB. Saeed, Abdulrahman M. W. Abdi. (2020). Feature Engineering for Stock Price Prediction . International Journal of Advanced Science and Technology, 29(12s), 2486-2496.