SE-GAN : Sentiment-Enhanced GAN for Stock Price Forecasting

A Comprehensive Analysis of Short-Term Prediction

Phrugsa Limbunlom 2311569

Supervisor: Dr. Michael Fairbank

School of Computer Science and Electronic Engineering
University of Essex

A thesis submitted for the degree of
Master of Science in Artificial Intelligence and Its Application

August 2024

Declaration

I affirm that this dissertation represents my original work, except where explicitly noted. The content has not been previously submitted, in part or in full, for any degree or qualification at university of Essex or other academic institution. All work presented is my own, conducted independently, unless otherwise stated in the text. This dissertation adheres to the prescribed limits, containing less than 15,000 words, including appendices, bibliography, tables, and equations, and fewer than 100 figures.

Phrugsa Limbunlom 2311569
August 2024

Abstract

Stock price prediction is a challenging task due to the inherent volatility of the market and the complexity of price movement. Traditional models still encounter limitations when faced with restricted historical data, which leads to over-fitting problems and poor performance on unseen data. Furthermore, market sentiment is a crucial factor for price fluctuations, yet several models still fail to adequately capture this important factor.

To address the issues, the research proposes a novel deep learning network architecture. The approach leverages generative adversarial networks (GANs) to expand the training set and generalize on unseen data by generating synthetic historical data. As the GAN model generates prediction of closing prices, the generated prices can be compared with actual market movements to assess its performance.

To capture the impact of market sentiment, the model also integrates sentiment analysis from financial news. This integration allows a more comprehensive technique for price forecasting, considering both quantitative historical data and qualitative sentiment indicators.

The model was trained on historical data of Microsoft stock (MSFT) from January 2013 to December 2023. To conduct market sentiment analysis on financial news, FINBERT will be utilized to extract sentiment scores from headlines. Additionally, various index of Microsoft stock (MSFT) are integrated in the training data.

The proposed model, *Sentiment-Enhanced GAN (SE-GAN)*, integrates sentiment analysis with generative adversarial networks (GANs) to generate more robust and accurate stock price predictions. The SE-GAN model demonstrates the lowest Root Mean Square Error (RMSE) for overall experiment setting compared to baselines models including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), as well as TimeGPT as another generative model.

The improvement suggests that the SE-GAN approach offers a more effective method for predicting stock prices, leveraging the combined strengths of sentiment analysis and generative adversarial networks (GANs).

Keywords: Stock Price Prediction, Sentiment Analysis, Generative Adversarial Networks (GANs), FINBERT

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The stock market plays a crucial role for modern investment by providing a platform for trading shares of public companies and other financial instruments. It serves two purposes: allowing companies to raise capital through share issuance and providing individuals with opportunities to create wealth through investment. This complex ecosystem encompasses various securities and stock exchanges worldwide, forming a vital component of the global financial landscape [1]. Accurately predicting stock prices in this dynamic environment presents significant challenges, as numerous factors, including market sentiment and economic indicators, influence price movements. While traditional approaches have attempted to model these complexities, the advent of deep learning and generative models has opened new approach to enhance the predictive analysis for financial context. This dissertation will study the intersection of sentiment analysis and Generative Adversarial Networks (GANs) to develop an innovative framework for stock price forecasting. The study is motivated by the recognition that stock price volatility is not solely determined by numerical indicators but is also significantly influenced by trader sentiments, opinions, and perceptions. Financial news can substantially impact market movements, often in ways that deviate from conventional modeling approaches. By integrating sentiment analysis into stock price prediction models, this study acknowledges the crucial role of market sentiment in shaping financial dynamics. Furthermore, to address the potential for over-fitting when relying solely on historical data, the proposed approach leverages GANs to generate synthetic data, thereby increasing dataset diversity and potentially improving model generalization. The goal of this research is developing a comprehensive model combining historical stock price data with sentiment analysis from financial news. By harnessing generative adversarial networks (GANs) architecture, the study aims to create a more robust and accurate model for stock price forecasting in today's complex market.

1.1 Problem Statement and Research Questions

The limitation of the data diversity can lead to an over-fitting problem that has a negative impact on the model prediction for the current stock price in the market. When trained on limited historical data, the model becomes too specific to the data, highlighting this problem. This is because the model fails to generalize to different patterns in unseen data. Furthermore, the high volatility of stock prices makes it difficult for the model to provide precision due to the influence of numerous complex factors, unpredictable market sentiment, unexpected events, and economic changes. Therefore, this study aims to address two primary research questions. Firstly, it seeks to determine whether utilizing closing prices generated by a Generative Adversarial Network (GAN) model can improve the overall accuracy of the model and mitigate the issue of overfitting. Secondly, the research investigates whether incorporating headline sentiment analysis of stock data can enhance the precision of short-term forecasts for future closing prices. By exploring these questions, the study aims to contribute to the advancement of stock price prediction methodologies.

1.2 Goal

The primary goal of this research is to implement a robust predictive model for stock price forecasting. The model aims to adopt a Generative Adversarial Networks (GANs) architecture integrating sentiment analysis from financial headlines to predict Microsoft stock prices. The objectives and scope of the project are described in the following sections:

1.2.1 Objectives

This study sets out to achieve four main objectives. First, it aims to develop a Generative Adversarial Networks (GANs) model that incorporates sentiment analysis of financial headlines. Second, the research aims to predict Microsoft stock's daily closing price using input data from previous dates. Third, the study will evaluate the closing price prediction results using input data from various sequential date ranges, including 7, 15, 30, and 60 days. Finally, the research will assess and compare the impact on prediction results when integrating sentiment analysis for short-term forecasts. Through these objectives, the study aims to enhance the accuracy and reliability of stock price prediction models.

1.2.2 Scope and Limitation

This study focuses on three key areas of its approach to stock price prediction. Specifically, the model developed will exclusively predict the daily closing price of *Microsoft stocks (MSFT)*. The predictions will be limited to daily intervals, rather than intraday or longer-term forecasts. Additionally, the model will utilize data from six major stock market indices as input for its predictions: *S&P 500, Dow Jones, NASDAQ 100, Nikkei 225, FTSE 100, and DAX 30*. By concentrating on these specific parameters, the study aims to create a focused and targeted approach to predicting Microsoft's stock performance within the context of global market trends.

1.3 Thesis structure

This thesis includes six main chapters. It begins with the **introduction**, presenting the problem statement, research questions, goals, and objectives. This chapter also outlines the scope and limitations of the study and provides an overview of the thesis structure.

The second chapter is **Related Works**. This chapter reviews existing literature and research relevant to the topic, establishing the context for the current study.

The **Background** chapter follows. This chapter offers detailed explanations of key concepts and technologies used in the research, including various neural network architectures and specialized models such as FINBERT, GANs, and TimeGPT.

The fourth chapter is **Methodology**, which is the core of the thesis. It describes the research approach in detail, starting with the data ingestion phase, including data collection, sentiment analysis, and data pre-processing. It then delves into the model training phase, covering the implementation of different models, including LSTM, GRU, GANs, and TimeGPT. This chapter concludes with the presentation of experimental results.

The **Evaluation** chapter follows. This chapter highlights the analysis of results using specific metrics, including Root Mean Squared Error (RMSE), Accuracy, and Sharpe Ratio to assess the performance of the models.

The thesis concludes with a final chapter, **Conclusion**, to summarize the key findings and focus on comparing the performance of SE-GAN with other generative models. This chapter also suggests directions for further research, providing a springboard for future studies in the field.

Throughout its structure, the thesis proposes a flow from introducing the problem to presenting the model development and evaluating its effectiveness

Chapter 2

Related works

Recent research has explored the application of deep learning techniques in financial forecasting, particularly for stock price prediction. Several studies have focused on addressing the limitations of traditional supervised learning methods, which can struggle with the complexity and diversity of trading strategies.

The first study is Wu et al. [2] suggesting an approach to predict daily trading actions using generative adversarial networks (GANs) along with a piecewise linear representation (PLR) method. This model demonstrated superior performance compared to long short-term memory networks in experimental evaluations.

In a different study, Zhang et al. [3] used a GAN architecture with "an MLP as the discriminator and an LSTM as the generator (Zhang et al.)" to forecast the future closing price. This approach showed promising results when testing on real data from the S&P 500 Index and various stocks across different trading periods, outperforming several other machine learning and deep learning models.

Furthermore, due to a significant impact of news and feelings on how investors react to the market, Priyank et al. [4] suggested using an ensemble method that combines sentiment analysis with BERT and a GAN to predict Apple Inc. stock prices. This approach integrated technical indicators, stock indexes, commodities, and historical price data, and was benchmarked against models such as Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), vanilla GAN, and ARIMA.

Similarly, Rahul et al. [5] created a hybrid model combining a Naive Bayes classifier for analyzing the tone of financial news with a GAN model using "LSTM as the generator and MLP as the discriminator (Rahul et al.)". This approach aimed to leverage both sentiment data and financial metrics for more accurate stock value predictions.

Lastly, research conducted by Hung Chun Lin et al. [6] exhibits a GAN model incorporating sentiment analysis from FINBERT for stock prices prediction. The model employed "a GRU network as the generator and a CNN architecture as the discriminator (Hung Chun Lin et al.)". The result shows that the proposed GAN model can produce lower RMSE scores compared to baseline models such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs).

In summary, these studies show that advanced deep learning techniques, especially GANs and hybrid models, can help make stock price predictions more accurate and reliable by capturing complex market dynamics and using a variety of data sources.

Chapter 3

Background

This section outlines the key components of the model architecture and the technical approaches employed in this research. It aims to provide context for both readers with and without background knowledge, laying the groundwork for the more detailed discussion in the methodology section. The components include Long short-term memory (LSTM), Gated recurrent units (GRUs), Multilayer Perceptron (MLP), Generative Adversarial Network (GANs), Transformer, FINBERT, and TimeGPT.

3.1 Long short-term memory (LSTM)

Long Short-Term Memory (LSTM) networks [7] are a type of recurrent neural network (RNN) architecture. It was designed to solve the vanishing gradient problem that occurs in typical RNNs when dealing with long-term dependencies in sequential data. LSTM becomes a popular usage in sequence modeling, techniques such as natural language processing (NLP) and time series analysis. The key component of LSTMs is the structure of the memory cell, which allows the network to select information to forget and memorize. This concept contains three main components : The forget gate determines which information to remove from the cell state. The input gate decides what data to store in the cell state. The output gate regulates the information that is utilized for the output. Furthermore, these gates employ sigmoid and tanh activation functions to regulate the information flow that allows LSTMs to capture both short-term and long-term dependencies in the data.

3.2 Gated recurrent units (GRUs)

Gated Recurrent Units (GRUs) [8] are one of recurrent neural network (RNN) architectures developed to improve Long Short-Term Memory (LSTM) networks. An update gate, which determines the amount of past information from previous time steps to pass forward to the future, and a reset gate, which determines the amount of past information to forget, are the primary features of GRUs. The difference between GRUs and LSTM is that GRUs do not have a separate memory cell, unlike LSTM. Instead, they use the hidden state to transfer information, making GRUs computationally more efficient and more lightweight than LSTM. Another advantage of GRUs is having fewer parameters, leading to faster training and being able to capture long-term dependencies.

3.3 Multilayer Perceptron (MLP)

Multi-Layer Perceptrons (MLPs) [9] are feedforward neural networks consisting of three layers at the minimum. The layers are an input layer, one or more hidden layers, and an output layer. Each node, excluding the input nodes, is a neuron using a nonlinear activation function. Fully connected layers, determining the connection of every neuron in one layer to every neuron in the next layer, are key features of MLPs. Backpropagation and nonlinear activation functions help the network learn complex and nonlinear mappings between inputs and outputs. Backpropagation is used to train the network by changing weights to reduce the error between predicted and actual outputs. Combining these components enables the MLPs network to learn intricate patterns, making it suitable for various tasks like regression and classification.

3.4 Generative Adversarial Network (GANs)

Two neural networks are comprised of Generative Adversarial Networks (GANs) [10], a generator and a discriminator. These two models are trained against each other at the same time. Two key components of GANs are the generator model, which generates fake data samples, and the discriminator model, which distinguishes the difference between real samples from the dataset and fake samples created by the generator. During the iterative adversarial training steps, two networks compete. The generator uses fake data to attempt to fool the discriminator, whereas the discriminator attempts to discover the real samples. The core concept of GANs is to generate new and synthetic instances of data that are difficult to differentiate from real data. The GAN model has been successful in image generation tasks but is also evident in applications from various domains, such as text, audio, and video generation.

3.5 Transformer

The Transformer [11] is a deep learning architecture that has revolutionized natural language processing (NLP) and is finding increasing applications in other domains.The Transformer leverages self-attention to capture relationships between all elements in a sequence in parallel, allowing for faster training and better performance on long sequences. This technique makes the transformer model different from traditional recurrent neural networks (RNNs) that process sequences sequentially. The key component of the transformer is self-attention, which is the core concept of this model, allowing one word to have more weight over different words in a sequence when encoding or decoding a specific word. This enables the model to capture long-range dependencies of the input data and contextual information efficiently. Another component is the encoder-decoder structure. The transformer employs an encoder-decoder structure, in which the encoder processes the input sequence and creates a contextualized representation, and the decoder then uses this representation to generate the output sequence. Both the encoder and decoder consist of multiple identical layers. Furthermore, positional encoding is used in the transfer model to inject the position of each word, since the model does not inherently capture the order of words in a sequence. The last component of the model is multi-head attention. Transformers do not use a single attention mechanism. Instead, they use multi-head attention, which lets the model pay attention to different parts of the input sequence with different heads. This gives the model a more accurate picture of the input. This model is suitable for long-range dependencies context since it can effectively capture relationships between distant words in a sequence. Although the model has various usages in related-language handling tasks, many applications from different domains also adopted this model to tackle their issues.

3.6 FINBERT

FINBERT [12] is a pre-trained language model designed to handle financial text. It takes advantage of the *BERT (Bidirectional Encoder Representations from Transformers)* architecture ability to understand context in both directions. FINBERT has fine-tuned a large corpus of financial data, enabling it to capture the nuances and jargon specific to the financial domain. FINBERT's main feature is that it has already been trained on a huge amount of financial text, such as 10-K reports, earnings call transcripts, and financial news articles.

This allows the model to learn specific words, phrases, and grammar, as well as semantic relationships, which are commonly seen in financial language. Additionally, the model leverages the BERT architecture by using a transformer network with self-attention to get two-way context, enabling the model's ability to understand word relationships and meaning in a deep way. One of the key uses of FINBERT model is sentiment analysis, which involves analyzing the sentiment expressed in financial news, tweets, and earnings calls to understand market perception and predict stock movements.

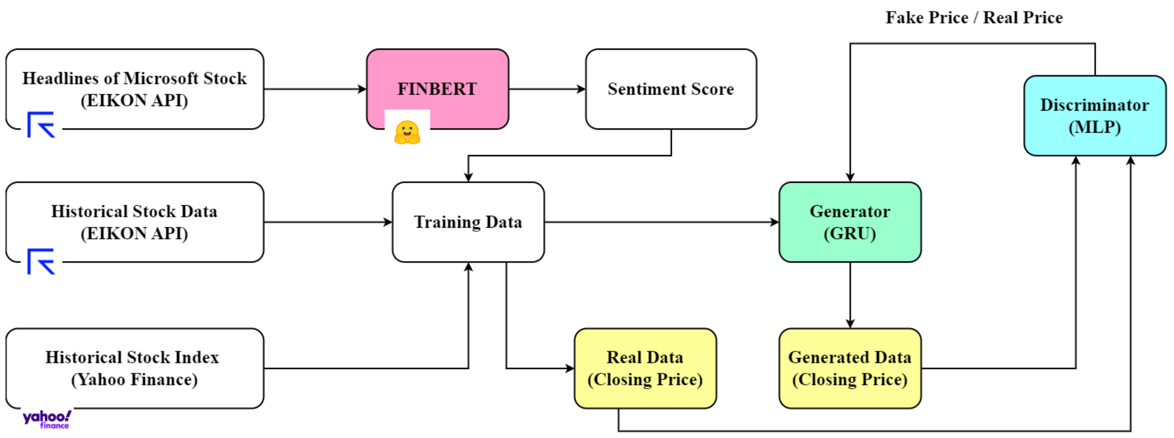
3.7 TimeGPT

TimeGPT, first introduced by Garza et al. (2024) [14], represents the first foundation model for time-series forecasting. TimeGPT employed a Transformer architecture to train an extensive dataset of over 100 billion data points from various domains, such as finance, healthcare, and IoT. The model demonstrates remarkable zero-shot inference capabilities and outperforms established statistical, machine learning, and deep learning methods. Also it exhibits superior performance across various time frequencies in experimental evaluations. TimeGPT's architecture enables it to handle complex time series characteristics, including multiple seasonalities, various trend types, and different noise levels. Moreover, the model offers rapid computation, averaging 0.6 milliseconds per series on GPU, and incorporates uncertainty quantification through conformal prediction. Additionally, TimeGPT can be fine-tuned for specific datasets to enhance its performance on specific data. By simplifying the forecasting pipeline, TimeGPT presents an opportunity to democratize access to sophisticated time series predictions, potentially reshaping current forecasting practices across industries.

Chapter 4

Methodology

This section outlines the methodology employed in the project SE-GAN as depicted in Figure 4.1, encompassing data collection, sentiment-analysis, integration of data from multiple sources, data preprocessing, model training, and experimental results compared to other models.



Fake Price / Real Price
Headlines of Microsoft Stock (EIKON API)
FINBERT
Sentiment Score
Discriminator (MLP)
Historical Stock Data (EIKON API)
Training Data
Generator (GRU)
Historical Stock Index (Yahoo Finance)
Real Data (Closing Price)
Generated Data (Closing Price)

Fig. 4.1 The diagram illustrates model development process of SE-GAN

4.1 Data Ingestion

This section delineates the data collection and pre-processing techniques employed to generate the final dataset used for model training. The process encompasses multiple stages: gathering data from diverse sources, conducting sentiment analysis using FINBERT, integrating various data streams, validating data correlations, and performing necessary pre-processing steps.

4.1.1 Data Collection

A dataset has been collected from the EIKON database in LSEG Workspace using the available API. Historical prices of Microsoft stock (MSFT.O) have been collected from 2013-01-01 to 2023-12-31. The price movement of Microsoft stock (MSFT) has been plotted in Figure 4.2. The dataset includes Date, HIGH, CLOSE, LOW, OPEN, COUNT, and VOLUME columns. Moreover, headlines about Microsoft stock have also been collected from 2023-03-29 to 2023-12-01 for sentiment analysis. For the headlines, there are 11,012 records in total. This is the dataset that will be processed as the training data later.

For the dataset used as the testing data, the historical prices have been collected from 2024-01-01 to 2024-05-31, and the headlines have been collected from 2024-01-01 to 2024-05-31, which are 5,243 records in total. However, the collection of the date of the headlines has been limited by the data provided from the API from the EIKON database. Therefore, the headlines collected from the data source includes only the date range from 2023 to 2024.

Furthermore, Index prices have also been collected from Yahoo Finance. The dates of the training and testing set are the same as the historical stock prices. The indexes include S&P 500, Dow Jones, NASDAQ 100, Nikkei 225, FTSE 100, and DAX 30.

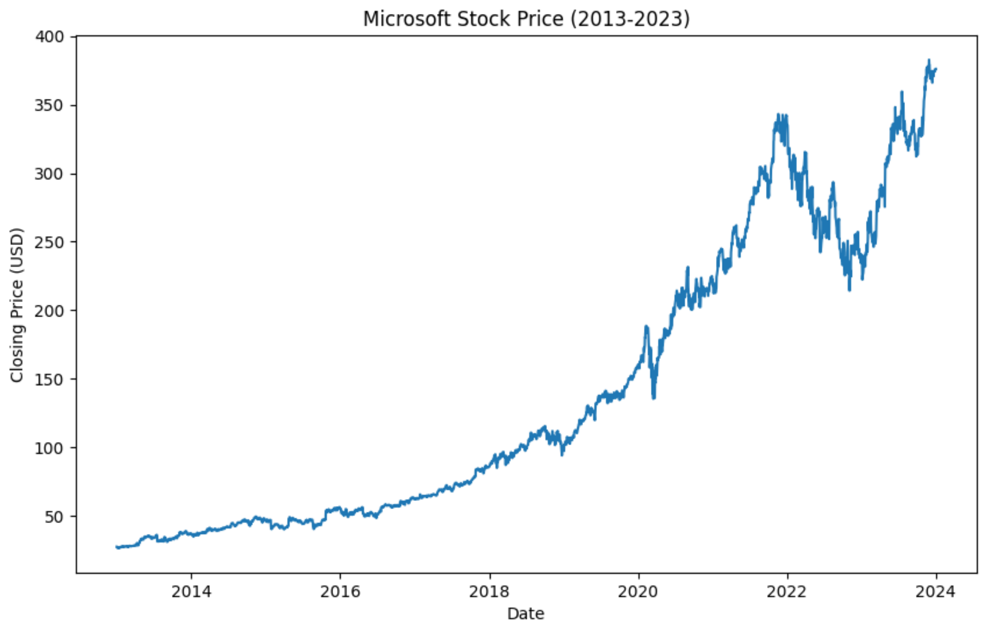


Fig. 4.2 The movement of the Microsoft stock prices from 2013 to 2023

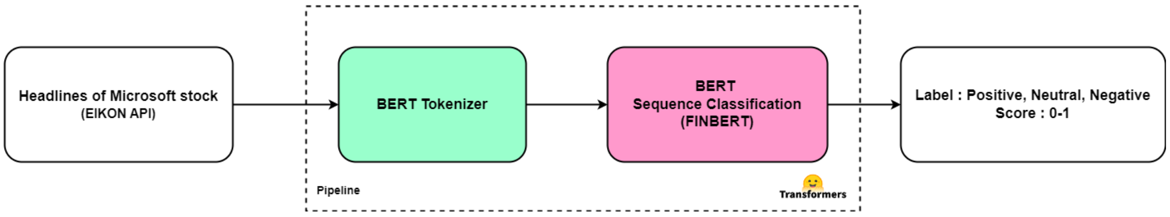


Fig. 4.3 Sentiment analysis pipeline of Microsoft stock headlines

4.1.2 Sentiment Analysis Using FINBERT

The headlines of the training and testing sets were used for sentiment analysis. The pre-trained model, FINBERT, was employed to convert the headlines to sentiment scores as illustrated in Figure 4.3. The model received headlines as the input data before feeding them to the pipeline to convert the text data, by using BERT tokenization, into encoded numerical representations that the model can interpret. The encoded data is then fed into the FINBERT model, a BERT-based classifier fine-tuned for financial text, to classify the label of input text. Finally, the model outputs a label (Positive, Neutral, or Negative) and a corresponding confidence score. The example of input and output data represented in Table 4.1.

To aggregate these sentiment scores from minute-level to daily-level, the following transformations were applied:

1. **Positive Sentiment**: The original score from the model output is retained.
2. **Neutral Sentiment**: The score is converted to zero.
3. **Negative Sentiment**: The score is converted to the negative of the original score from the model output.

After these transformations, the average score for each date was calculated to produce a daily sentiment indicator. This approach preserves the magnitude of the model's confidence while creating a continuous scale, ranging from -1 to 1, where positive values indicate positive sentiment, negative values indicate negative sentiment, and values near zero represent neutral sentiment.

This aggregation method allows for a nuanced daily sentiment score that reflects both the direction and strength of the sentiment expressed in the news headlines throughout each day.

FINBERT Implementation

The pre-trained model, `yiyanghkust/finbert-tone` [13], for sentiment analysis was employed to analyze the sentiment of headlines data. This model was called by using the

FINBERT Sentiment Analysis	
Input Text	**Output**
CallMiner Collaborates with Microsoft to Enhance AI and Machine Learning Capabilities	label: Positive score: 0.9842389822006226
5 things Microsoft co-founder Bill Gates wishes he knew in his early 20s	label: Neutral score: 0.9726649522781372
MICROSOFT STATUS PAGE SHOWS TEAMS AND OUTLOOK.COM ARE HAVING PROBLEMS	label: Negative score: 0.9999890327453613

Table 4.1 Example input and output data for sentiment analysis using FINBERT

library *Transformer* as described in Appendix B.0.1. The source code of FINBERT implementation can be accessed in the repository mentioned in Appendix A and inside the folder `/sentiment-analysis/sentiment-analysis.py` provided in Appendix B.

4.1.3 Data Incorporation

Historical stock prices, daily sentiment scores, and index prices were combined based on their corresponding dates. This combined dataset then underwent further pre-processing steps.

4.1.4 Data Correlation

After the data incorporation, the correlation between the prices of Microsoft stock (MSFT) and sentiment scores has been visualized to verify the correlation between *daily returns* and *sentiment scores* at the same date (Figure 4.4), the correlation between daily returns and the sentiment scores of the next dates (Figure 4.5), and the correlation between the current dates of sentiment scores and the next dates of daily returns (Figure 4.6). According to the graphs, the correlation between the daily returns and sentiment scores of the same dates is around 0.147, the correlation between the daily returns and the sentiment scores of the next dates is around 0.03, and the correlation between the sentiment scores and the daily returns of the next dates is around 0.078. It can be seen that stock prices and sentiment scores are *correlated* at some extent, suggesting that sentiment data can provide valuable insights for predicting future stock price movements.

According to the correlation results, the analysis reveals the strongest relationship exists between daily returns and sentiment scores on the same date, compared to correlations with current return on next-day sentiment and current sentiment on next-day returns. This suggests that news released on a given day is more likely to influence market sentiment and trading decisions immediately. While news from the current day still impacts the market to some degree, and today's daily returns have a minor influence on buying and selling decisions, these effects are less pronounced than the same-day correlation. This finding underscores the immediacy of market reactions to news and sentiment in the context of daily stock trading.

Thus, by integrating sentiment scores, the system may be able to more accurately forecast future stock prices and help the model identify complex patterns and relationships that may not be readily apparent from the price data alone.

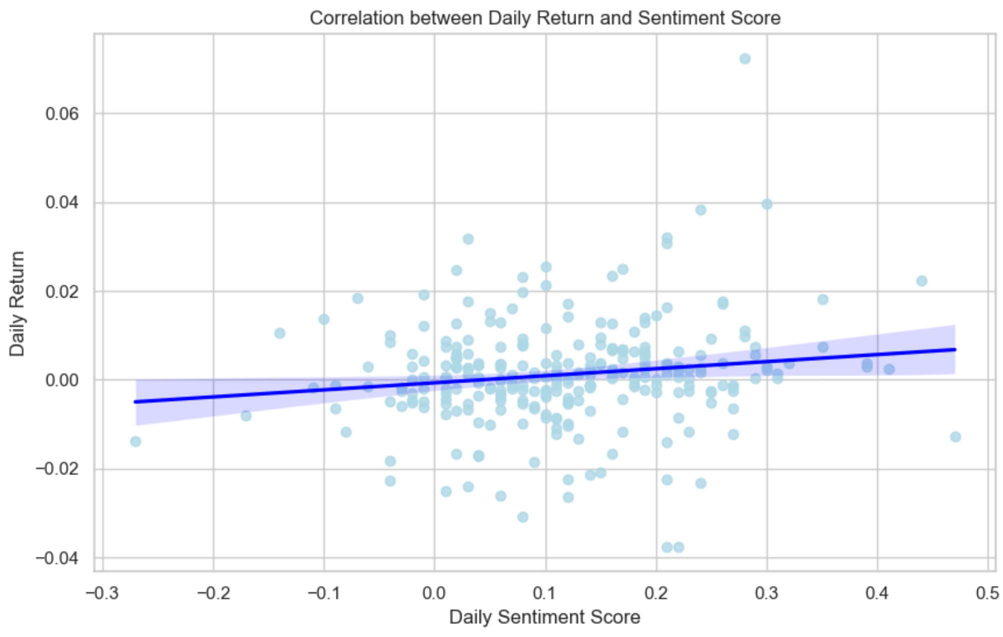


Fig. 4.4 The correlation between daily returns and sentiment scores at the same date

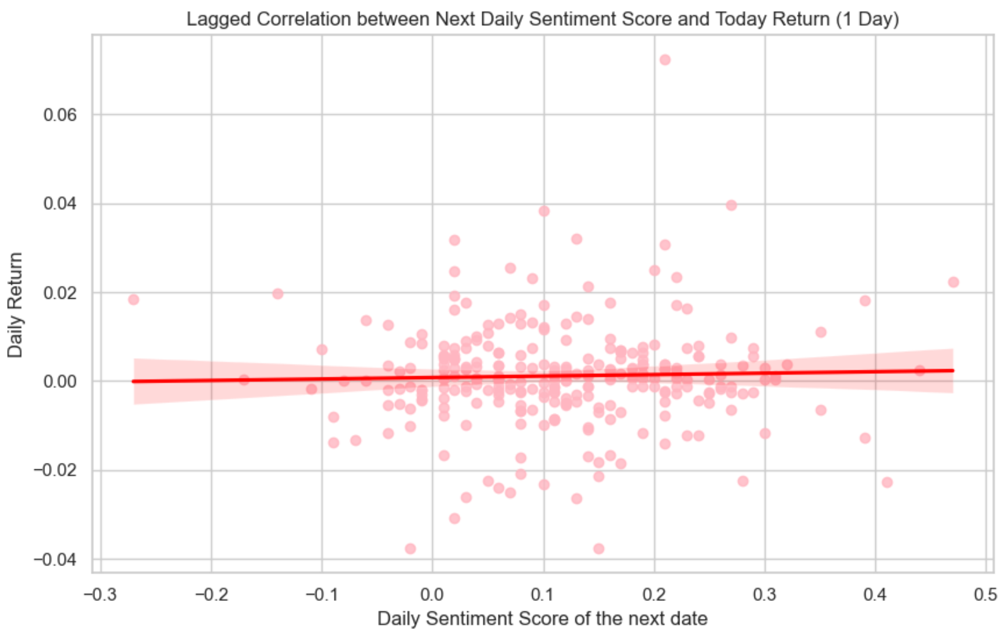


Fig. 4.5 The correlation between daily returns and the sentiment scores of the next dates

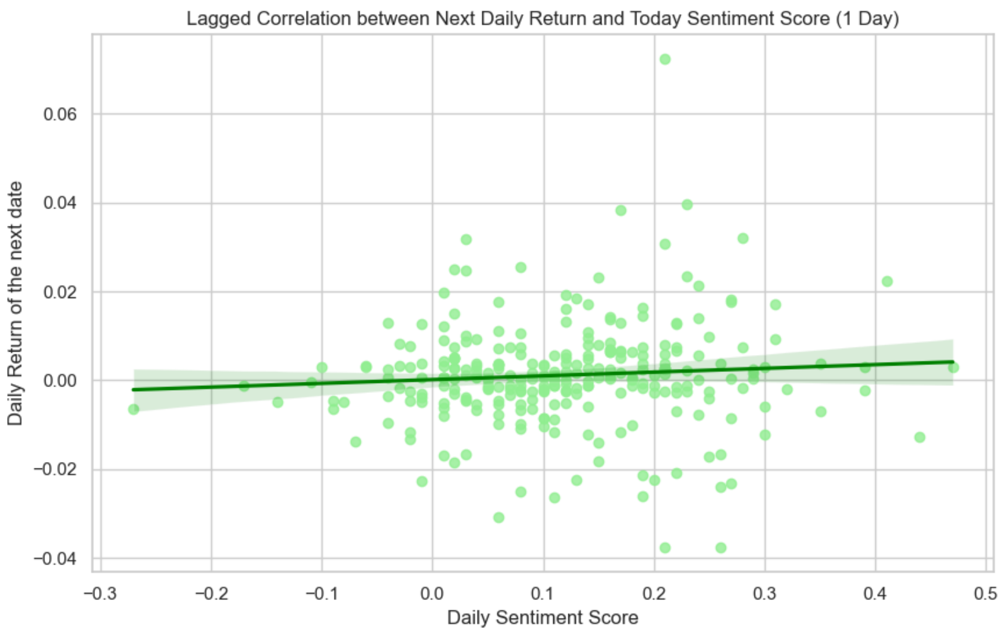


Fig. 4.6 The correlation between the current dates of sentiment scores and the next dates of daily returns

	ALL	**With Sentiment Score**
Training Set	4017	282
Testing Set	152	150

Table 4.2 The table represents the final dataset used to train and evaluate the models

	Columns	**Description**
Timestamp	Date	Date of the trading day of the Microsoft stock
Features	HIGH	Highest price of the stock during the day
	LOW	Lowest price of the stock during the day
	OPEN	Opening price of the stock for the day
	COUNT	Number of trades executed
	VOLUME	Total number of shares traded
	Daily Sentiment Score	Aggregate sentiment score from news and social media
	S&P 500	Value of the S&P 500 index
	Dow Jones	Value of the Dow Jones Industrial Average
	NASDAQ 100	Value of the NASDAQ 100 index
	Nikkei 225	Value of the Nikkei 225 index
	FTSE 100	Value of the FTSE 100 index
	DAX 30	Value of the DAX 30 index
Label	CLOSE	Closing price of the stock for the day

Table 4.3 Features included in the dataset and description

4.1.5 Data Pre-processing

The data pre-processing steps commenced with an assessment of null values within the dataset, followed by the time interpolation technique to address any missing dates after data incorporation. Subsequently, a Min-Max scaler was employed to normalize the data ranging from -1 to 1.

4.1.6 Final Dataset

Following the aforementioned procedures, the final composition of the training and testing datasets is detailed in the table 4.2. In summary, the features included in the datasets for further training are listed in table 4.3.

4.2 Model Training

This section elucidates the models utilized for training the final dataset, which was generated through the techniques outlined in the Data Ingestion section. The methodology employs a range of models, including Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Generative Adversarial Network (GAN), and TimeGPT.

4.2.1 Long short-term memory (LSTM)

The first baseline model is LSTM. The architecture begins with an LSTM layer comprising 50 units (neurons). The input shape is defined as (sequence length, features), representing the sequence of data and all features used in the model. This initial layer utilizes the `return_sequences` parameter to output for each time step.

Following the LSTM layer, a dropout layer with a rate of 0.2 is implemented. This dropout mechanism serves as a regularization technique to mitigate overfitting. The subsequent layers follow a similar pattern to enhance the model's complexity. These consist of another LSTM layer without returning sequences, followed by another dropout layer with the same rate.

The final layer of the model employed to predict the closing price (CLOSE). It is a dense layer with an output size of 1 and no activation function.

The model architecture is structured as Figure 4.7:

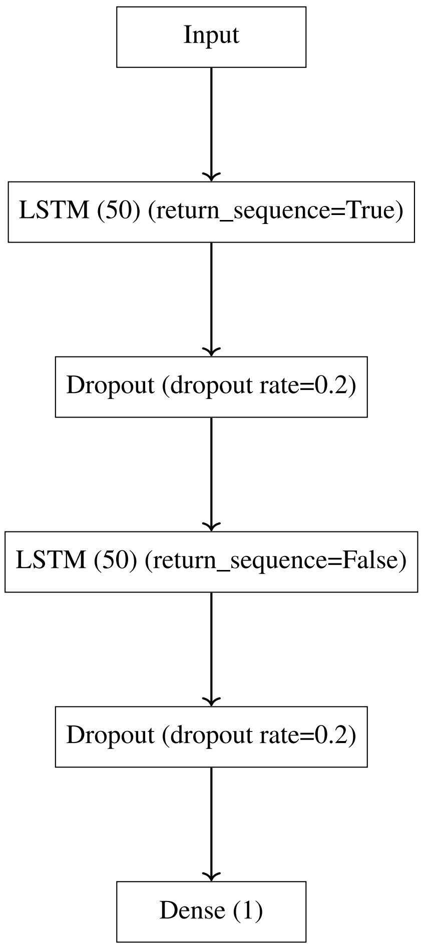

Input
LSTM (50) (return_sequence=True)
Dropout (dropout rate=0.2)
LSTM (50) (return_sequence=False)
Dropout (dropout rate=0.2)
Dense (1)

Fig. 4.7 LSTM Model Architecture

The training process is configured as follows:

- Data split: The training dataset is partitioned into 80% for training set and 20% for validation set.
- Hyperparameters:

$$\begin{aligned}
\text{Sequence Length} &= 7, 15, 30, 60 \\
\text{Epochs} &= 50 \\
\text{Batch Size} &= 32 \\
\text{Optimizer} &= \text{Adam} \\
\text{Loss Function} &= \text{Mean Squared Error (MSE)}
\end{aligned} \tag{4.1}$$

This configuration enforces effective model training while evaluating performance on a validation set. The Adam optimizer is chosen for its adaptive learning rate capabilities, while MSE serves as an appropriate loss function for predicting the closing price (CLOSE).

LSTM Implementation

The program for LSTM model implementation can be accessed in the project repository provided in Appendix A, specifically in the file `/model/Baseline/training/LSTMModel.py`. For model training, the program can be found in `/model/Baseline/training/BaseModel.py`, as per the project structure detailed in Appendix B.

4.2.2 Gated recurrent units (GRUs)

Another baseline model is GRU. Similar to LSTM, the first layer begins with GRU layer containing 50 units. The input shape represents the sequence of data and all features used to train the model, defined as (sequence length, features). This layer utilizes the `return_sequences` parameter to output for each time step. The subsequent layer is a dropout layer with a rate of 0.2. This dropout layer is applied for regularization, helping to prevent overfitting. Two additional layers are employed to learn more complex sequential patterns in the data. These layers follow the same structure as the previous ones, a GRU layer followed by a dropout layer. However, this GRU layer does not return sequences to output only the final state. The final layer is a dense layer with output size 1 and no activation function. This layer produces the model's output, which is the closing price (CLOSE).

The architecture is defined as Figure 4.8:

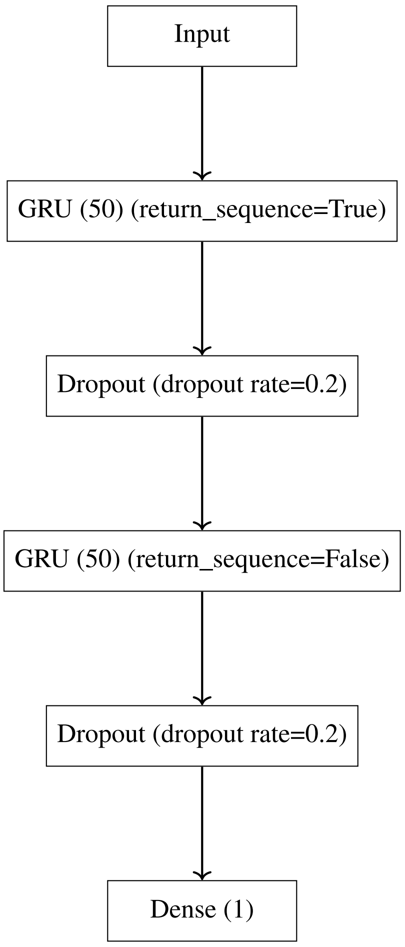

Input
GRU (50) (return_sequence=True)
Dropout (dropout rate=0.2)
GRU (50) (return_sequence=False)
Dropout (dropout rate=0.2)
Dense (1)

Fig. 4.8 GRU Model Architecture

The GRU model employs training parameters identical to those used for the LSTM model:

$$\begin{aligned}
\text{Sequence Length} &= 7, 15, 30, 60 \\
\text{Epochs} &= 50 \\
\text{Batch Size} &= 32 \\
\text{Optimizer} &= \text{Adam} \\
\text{Loss Function} &= \text{Mean Squared Error (MSE)}
\end{aligned} \tag{4.2}$$

This configuration ensures a consistent training approach across both the LSTM and GRU models, which facilitates a fair comparison of their respective performances.

GRU Implementation

The program for GRU model implementation can be accessed in the project repository provided in Appendix A, specifically in the file `/model/Baseline/training/GRUModel.py`. For model training, the program can be found in `/model/Baseline/training/BaseModel.py`, as per the project structure detailed in Appendix B.

4.2.3 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) model has been trained through an iterative feedback mechanism. The process commences with employing generator model to generate closing prices, which are subsequently input into the discriminator model to receive the feedback regarding the authenticity of the prices generated. Once the generator model received the feedback from the discriminator, the model will undergo optimization to generate more synthetic prices to fool the discriminator, aiming to generate the prices that closely approximate actual market values.

The generator model and the discriminator model have been implemented in this study are described below.

Generator Model

The generator model is designed to generate the closing price (CLOSE). It employs a GRU-based architecture, which processes sequential data and its associated features as input and utilizes a dense layer with linear activation as the output to generate continuous price values. The architecture mirrors that of the baseline model described in Figure 4.8.

The Generator component of the model processes input data, which includes historical stock prices, sentiment scores, and stock indexes. It learns the underlying patterns and relationships among these features for each data point. Using this combined information, the generator attempts to produce a realistic future closing price for a *one-day* time horizon based on sequences of input data. These input sequences vary in length, including 7, 15, 30, and 60 days prior to the target prediction date.

The generator (G) can be mathematically represented by the following equation:

$$G(X) = \hat{x}_{t+1} = \delta(\mathbf{W}_h^T \mathbf{h}_t + \mathbf{b_h}) \tag{4.3}$$

where:

- X represents the input data

- $\hat{x}_{t+1}$ is the predicted future closing price after time t
- δ is an activation function
- $\mathbf{W}_h^T$ represents the weight matrix at time t
- $\mathbf{h}_t$ is the hidden state at time t
- $\mathbf{b_h}$ is the bias term

This equation describes functionality of the generator model when receiving input data from the previous days and outputs a prediction of the closing price of the next day by using a GRU architecture.

The training configuration for the generator model is as follows:

$$\begin{aligned}
\text{Sequence Length} &= 7, 15, 30, 60 \\
\text{Epochs} &= 180 \\
\text{Optimizer} &= \text{Adam with:} \\
\text{Learning rate} &= 0.001 \\
\beta_1 &= 0.9 \\
\beta_2 &= 0.999 \\
\varepsilon &= 1 \times 10^{-7} \\
\text{Loss Function} &= \text{Mean Squared Error (MSE)}
\end{aligned} \tag{4.4}$$

This configuration enables extensive training with 180 epochs and utilizes the Adam optimizer with carefully hyperparameters tuning. The method will effectively facilitate the model to learn complexity of the price patterns.

The loss function of the generator model is Mean Squared Error (MSE) as detailed below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.5}$$

where y_i represents the actual closing price and $\hat{y}_i$ denotes the predicted closing price of the model.

The loss function is able to guide the generator model to produce realistic closing prices, which approximately close to actual market values. This will make the model susceptible to generate outputs that are as close as possible to the real closing prices, which can be achieve by minimizing the squared difference between predicted and actual prices.

Discriminator Model

The discriminator is another component of the GAN model. This model acts as an adversary to the generator, which will evaluates the actual prices and the synthetic prices produced by the generator. The method aims to distinguish between generated and real closing prices from the input data. To illustrate, this adversarial steps compels the generator to continuously refine its output to produce more realistic closing prices that closely resemble to the historical data. At the end of the day, the generator aims to create promising future price sequences capable of deceiving the discriminator.

The discriminator (D) can be represented as the equations below:

$$D(X_{fake}) = \sigma(d(X_{fake})) \tag{4.6}$$

$$D(X_{real}) = \sigma(d(X_{real})) \tag{4.7}$$

where:

- X_{fake} denotes the generated closing price
- X_{real} denotes the real closing price
- $d(\cdot)$ is a discriminator function yielding a score of the input data being real or fake
- $\sigma(\cdot)$ is the sigmoid activation function mapping the output score to a probability between 0 and 1.

These equations represent functionality of the discriminator taking a closing price as input and output a probability of that input being real.

The model utilizes a multi-layer perceptron (MLP) architecture including multiple layers for creating model complexity. To be more precise, a series of four dense layers with units of 256, 128, 64, excluding the final layer, is constructed , along with utilizing the Leaky ReLU activation function with an alpha parameter of 0.2, helping to mitigate the vanishing gradient problem. This model can provide more efficient training for deeper networks. Furthermore, the alpha parameter introduces a small non-zero slope for negative inputs that further enhance the ability of the model for learning complex relationships in the data.

The training configuration for the discriminator model is as follows:

$$\begin{aligned}
&\text{Sequence Length} && = 7, 15, 30, 60 \\
&\text{Epochs} && = 180 \\
&\text{Optimizer} && = \text{Adam with:} \\
& && \text{Learning rate} = 0.001 \\
& && \beta_1 = 0.9 \\
& && \beta_2 = 0.999 \\
& && \varepsilon = 1 \times 10^{-7} \\
&\text{Loss Function} && = \text{Binary Cross Entropy}
\end{aligned} \tag{4.8}$$

The loss function of the discriminator model is binary cross-entropy that is used to update the weight of the model because of its ability for classification tasks. This loss is used for classification tasks since its ability to measure the dissimilarity between the classification and the actual labels of the input data, encouraging to distinguish between generated and real prices.

The binary cross-entropy loss is defined as following equations :

$$\text{Binary cross-entropy} = -(y \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{4.9}$$

where y_i yi denotes the ground truth labels (1 for real, 0 for fake) and $\hat{y}_i$ denotes probability of the input prices classified to be real or fake.

This method contributes to overall efficiency of the GAN architecture, by encouraging it learning optimal probability of the distribution for accurate classification. In other words, the discriminator model aims to classify actual prices and generated prices by aiming to minimize the loss, which will improve performance of the GAN framework.

In summary, the discriminator architecture as depicted in Figure 4.9 can provide a robust framework to effectively distinguish between generated and real closing prices. Moreover, this designed architecture can contribute to ability of the model to accurately assess the authenticity of generated closing price

GAN optimization

Training a GAN involves a competition between two neural networks. The generator aims to deceive the discriminator that is it quantifies a success by achieving a lower G_loss. A lower loss of the generator model indicates how the generated data closely resembles the real data, for this case is closing prices, making it difficult for the discriminator to distinguish between them. Conversely, the Discriminator Loss (D_loss) reflects the ability of the discriminator model to differentiate between real and generated data. A higher D_loss means the discriminator is effectively identifying fake samples (generated prices), which can be translated that the discriminator is completely fooled by the generator model.

This competitive dynamic is captured in the following mini-max equations, which represents the objective function of the GAN training:

$$\min_G \max_D V(G,D) = \mathbb{E}[\log D(X_{real})] + \mathbb{E}[\log(1 - D(G(X)))] \tag{4.10}$$

$$\min_G \max_D V(G,D) = \mathbb{E}[\log D(X_{real})] + \mathbb{E}[\log(1 - D(X_{fake}))] \tag{4.11}$$

GAN Implementation

The source code provided for the GAN implementation can be accessed in the project repository from Appendix A inside the folder `/model/GAN` as detailed in Appendix B. Moreover, the implementation of the GAN model has been refereed from the repository of the paper *Stock price prediction using Generative Adversarial Networks* by HungChun Lin et al [6]. During the implementation, it was found that the paper used binary cross entropy for calculating the loss of the generator model. This loss was used to calculate from assuming 1 as the real price and 0 as the fake price. Due to this, the loss has been modified from the reference paper by using Mean square error (MSE) loss to calculate the loss of generated price with the real price for the generator model. The reason behind this idea is that the generator model aiming to generate the price which is closet to the real price. After modification, it can help in a significant reduction in the Root mean square error (RMSE) loss of the dataset for this research. Apart from this, it was detected that the training loss was calculated by by using the generated price for each epoch to calculate the final RMSE. After followed this approach, it was figured out that the training RMSE was significantly high at certain level compared to the testing RMSE. Additionally, the graphs between real and generated price look compromise, which is conflict to the result receiving from the training loss. For this reason, the issue was solved by using the saved weights after model training in order to calculate RMSE for training set, which is the same as method used when calculating the RMSE of the testing set.

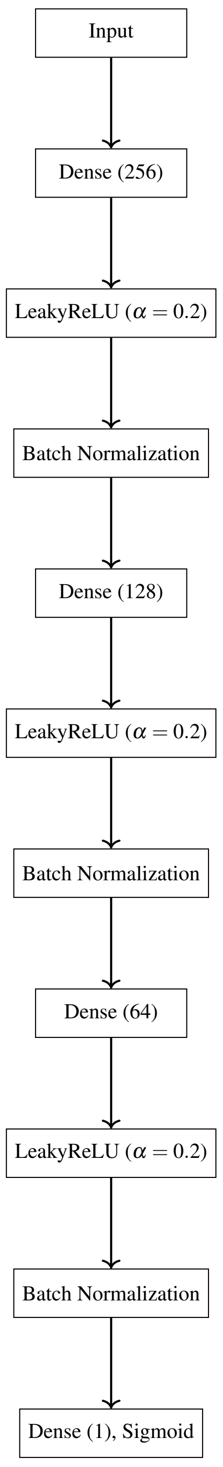


Fig. 4.9 MLP Model Architecture

4.2.4 TimeGPT

TimeGPT, a pre-trained model optimized for time-series data analysis, offers a robust framework for forecasting future price movements. This study employs the *Nixtla* [14] library, specifically its forecast function, to generate closing prices based on historical data and exogenous variables. The model fine tunes on two parameters including training the input data for 10 times (steps = 10) and using MSE loss.

The primary input variables are:

- **Temporal indicator**: Date (timestamp)
- **Target variable**: CLOSE price

Exogenous variables incorporated in the model include:

- **Price indicators**: HIGH, LOW, OPEN
- **Trading metrics**: COUNT, VOLUME
- **Market sentiment**: Daily Sentiment Score
- **Major market indices**: S&P 500, Dow Jones, NASDAQ 100, Nikkei 225, FTSE 100, DAX 30

By integrating these diverse data points, the model aims to capture complex market dynamics and generate accurate prices forecasts for specified time horizons. This methodology leverages the predictive power of the GPT model trained on extensive historical data from multiple time-series data sources, aiming to offer insights into future trends of unseen data without training the dataset or fine tuning the dataset provided by leveraging transfer learning technique.

TimeGPT Implementation

The TimeGPT model was developed by using the library from *Nixtla* as described in the Appendix B.0.1 to employ the pre-trained model to train and fine-tune the given dataset. The source code of TimeGPT usage can be accessed by the project repository mentioned in Appendix A inside the project folder detailed in Appendix A.

4.3 Experimental Results

After the training process, the loss of each model during training has been plotted. The models were trained using different sequence lengths determined previous dates of input data, including 7 days, 15 days, 30 days, and 60 days.

Figures 4.10-4.13 illustrate the training losses and validation losses of LSTM models, whereas Figures 4.14-4.17 represent the training losses and validation losses of GRU models. For the GAN models, Figures 4.18-4.21 demonstrate the losses between generator and discriminator models.

According to the results, the analysis of the training and validation losses reveals distinct patterns across the different models. The LSTM and GRU models exhibit stable training losses, starting around 0.02-0.03 and gradually declining to approximately 0.001. However, the validation losses for these models are particularly unstable.

In depth, for the LSTM model, the validation losses demonstrate dramatical fluctuation for each epoch during training. This can be seen when the volatility increasing as the sequence length of the input data extending. Specifically, the validation losses for sequence lengths of 30 days and 60 days input are more volatile compared to the smaller sequence lengths like 7 days and 15 days.

The GRU model demonstrates an identical trend. The validation losses are also volatile, yet small instability occurred and having no significant differences among the various sequence lengths.

In contrast to the LSTM and GRU models, the GAN model exhibits more stable behavior. To illustrate, the losses of the generator model start in the range of 0.2-0.6 and steadily decline to nearly 0.001, while the losses of the discriminator begin between 1.6-2.0 and drop to around 1.3-1.4.

These results highlight the relative effectiveness of the models. It indicates how efficiency each model is to capture and learn the underlying complexity in the data. The stable losses of the GAN model suggests its ability to capture the signal of the input data as well as converge to the optimal points during the training phrase more efficiently than other models.

This comparative analysis suggests that the GAN model yield the better performance in term of its ability to consistently learn the patterns in the financial data, especially historical stocks. This is because the GAN model produces more desirable results and stable trends compared to the LSTM and GRU models where the volatile behavior observed during training.

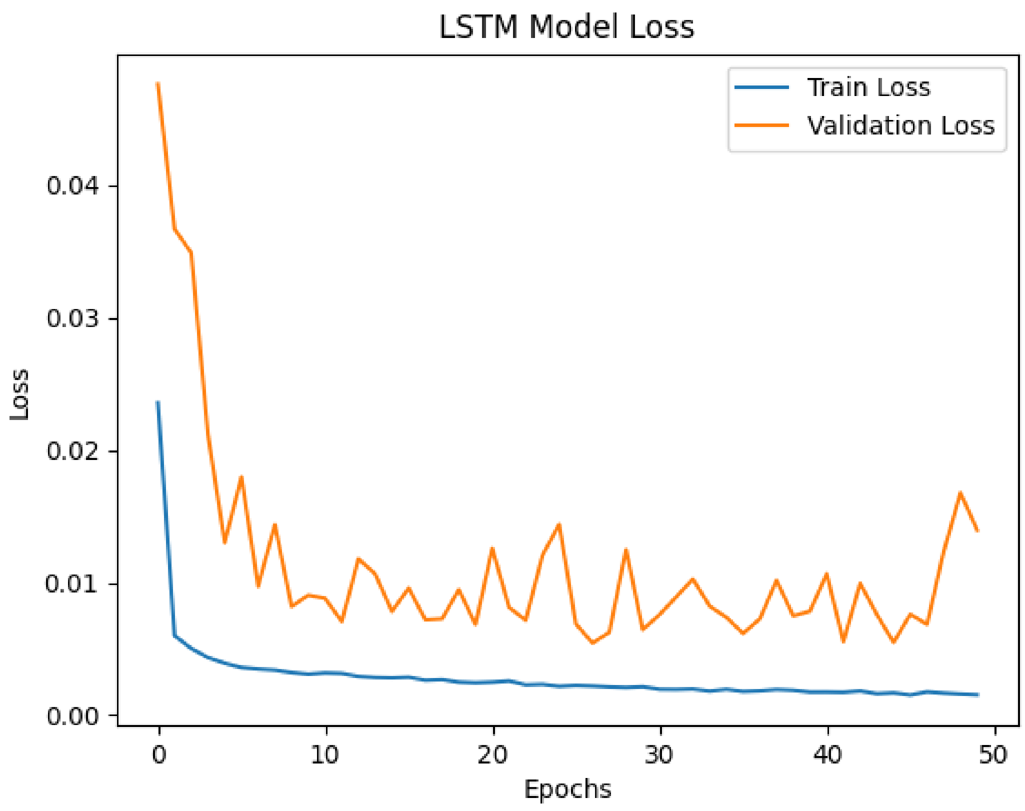


Fig. 4.10 The loss of LSTM model for sequence length of 7 days

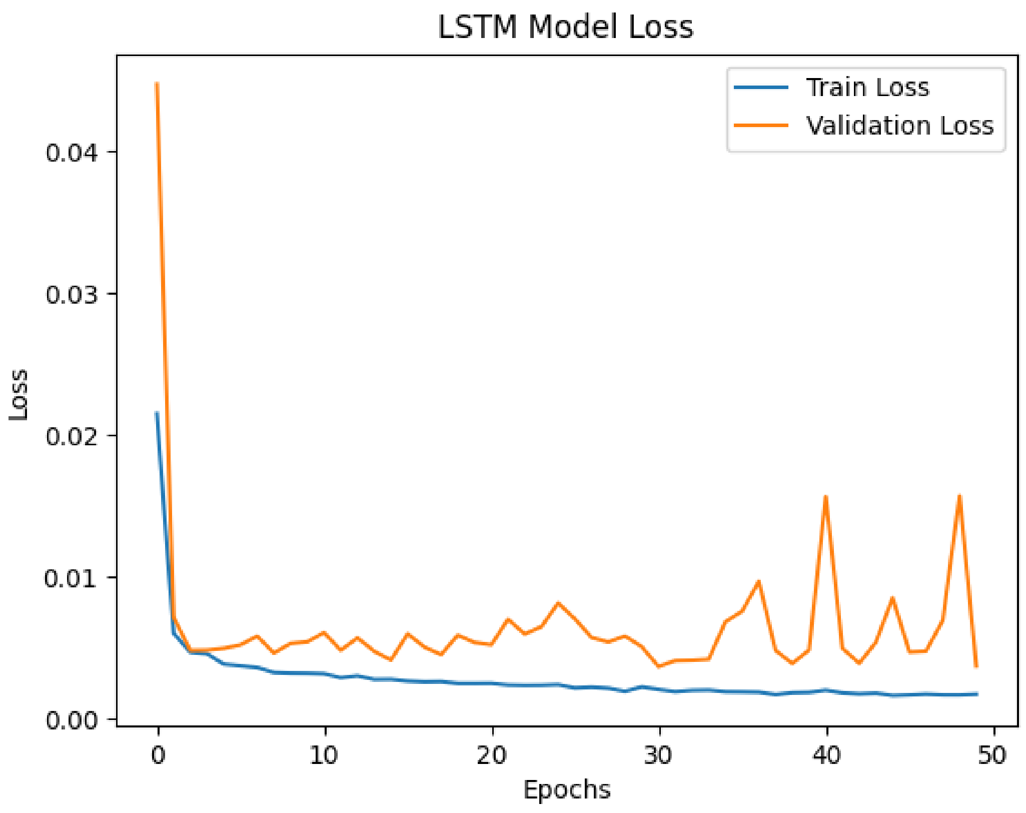


Fig. 4.11 The loss of LSTM model for sequence length of 15 days

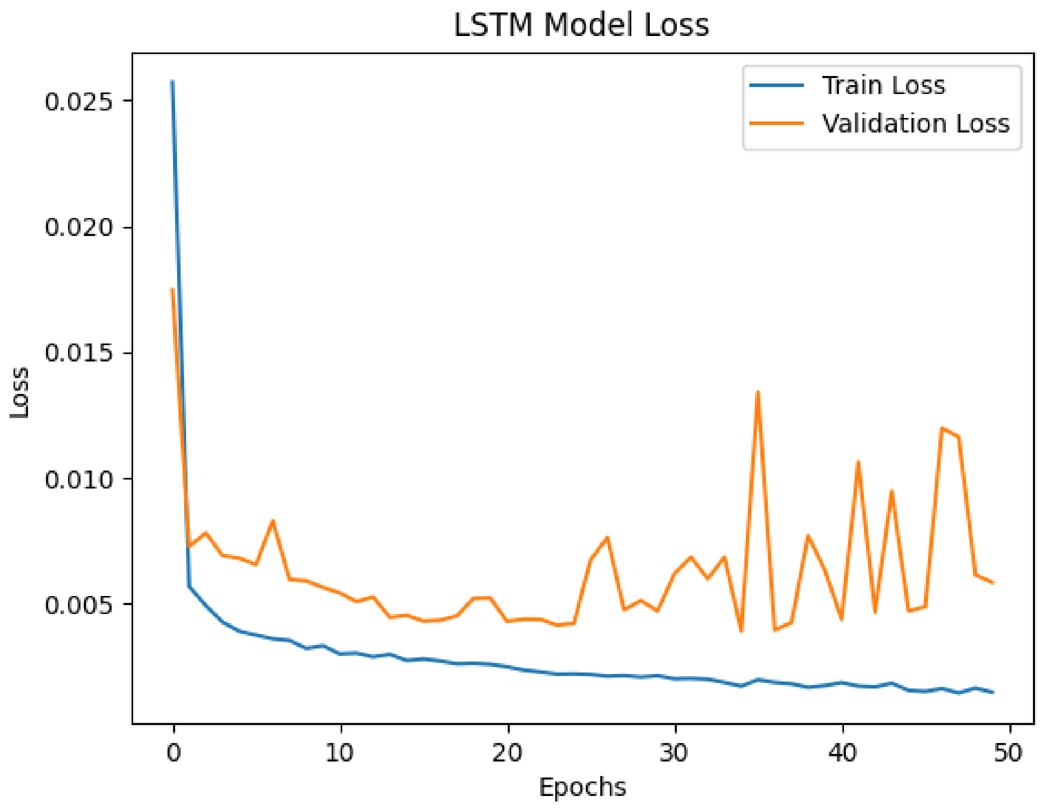


Fig. 4.12 The loss of LSTM model for sequence length of 30 days

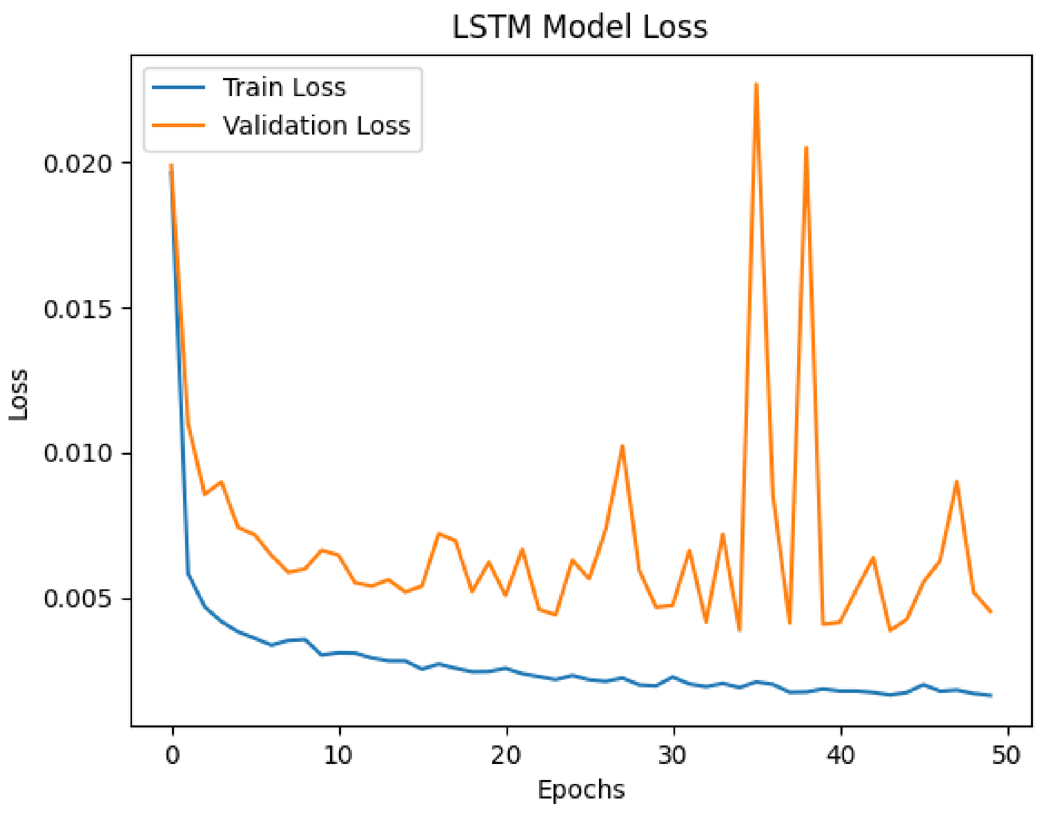


Fig. 4.13 The loss of LSTM model for sequence length of 60 days

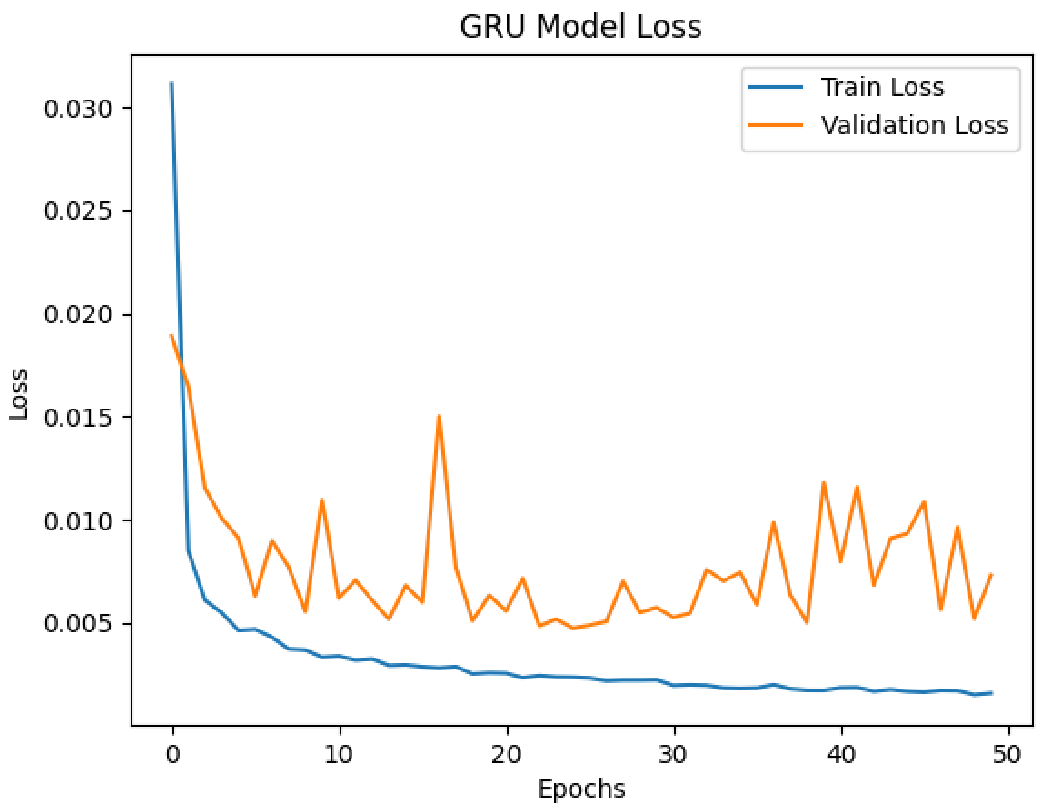


Fig. 4.14 The loss of GRU model for sequence length of 7 days

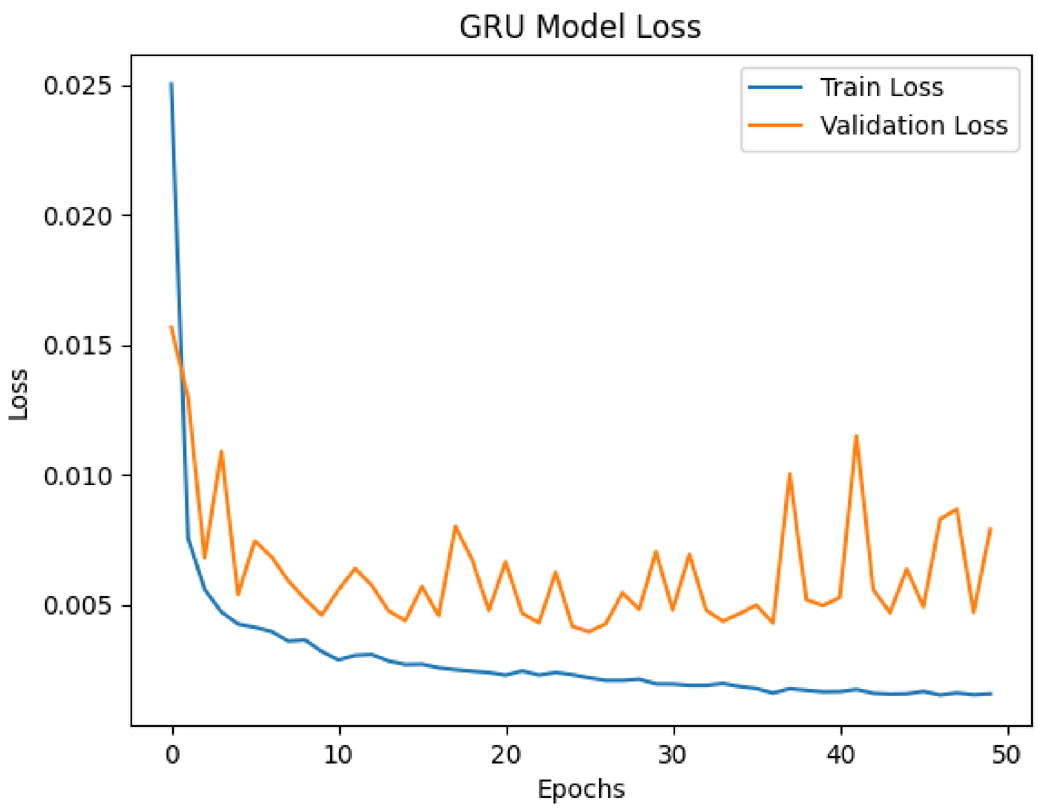


Fig. 4.15 The loss of GRU model for sequence length of 15 days

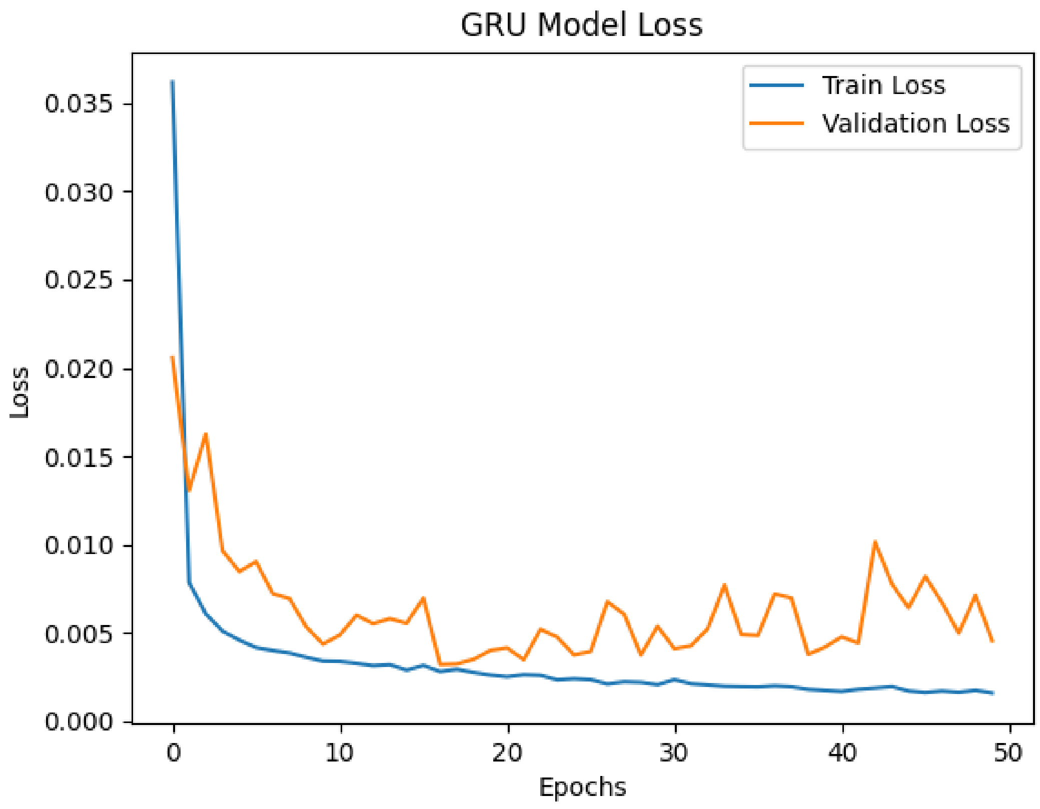


Fig. 4.16 The loss of GRU model for sequence length of 30 days

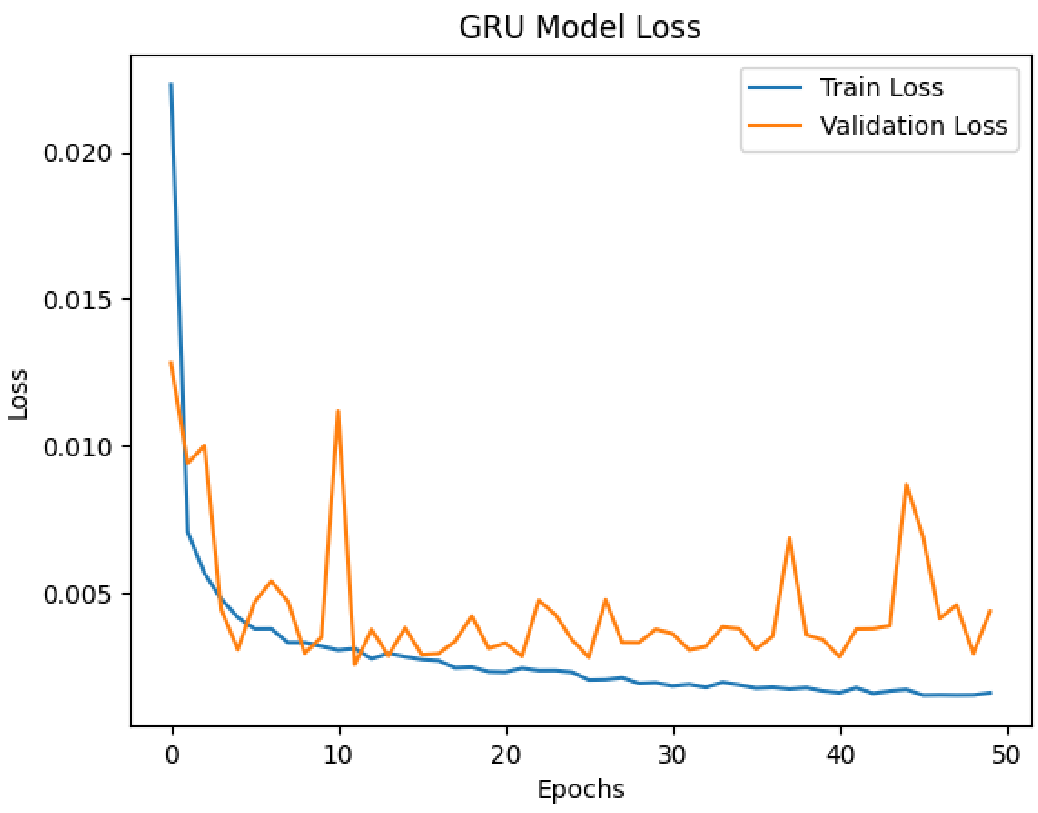


Fig. 4.17 The loss of GRU model for sequence length of 60 days

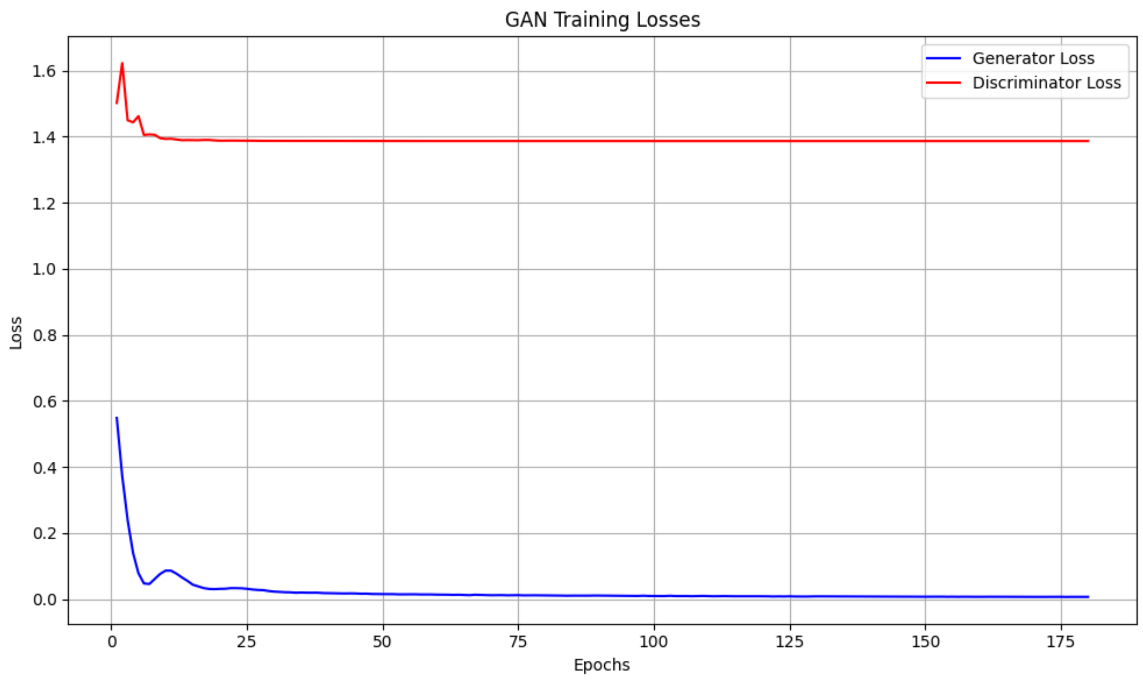


Fig. 4.18 The loss of GAN model for sequence length of 7 days

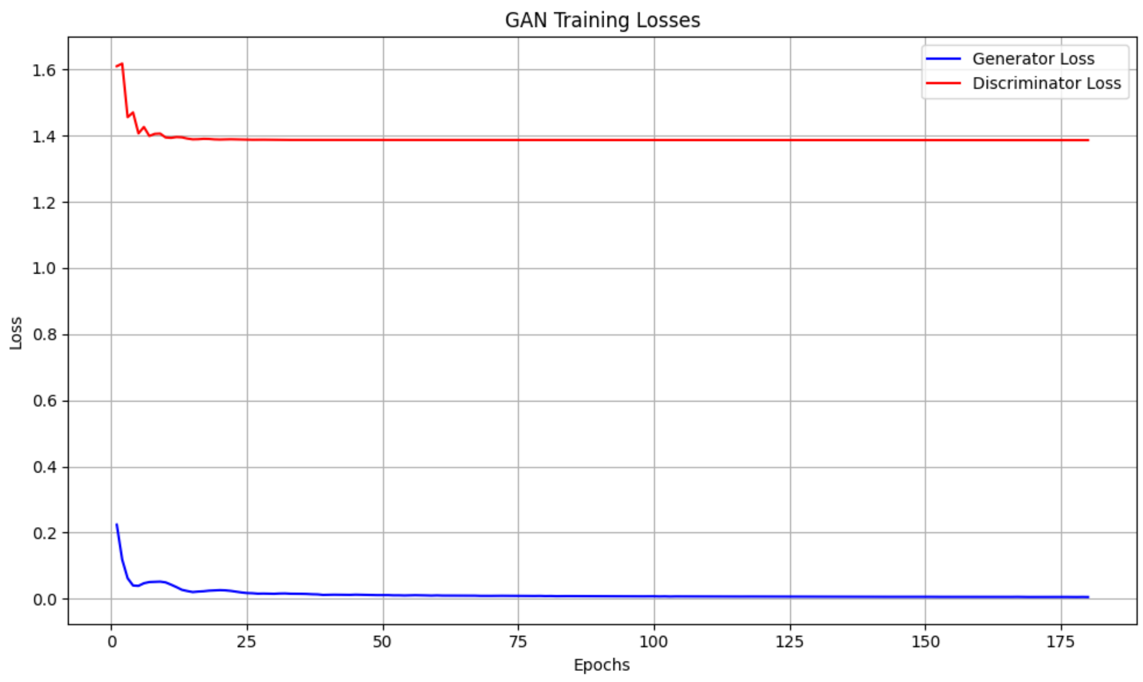


Fig. 4.19 The loss of GAN model for sequence length of 15 days

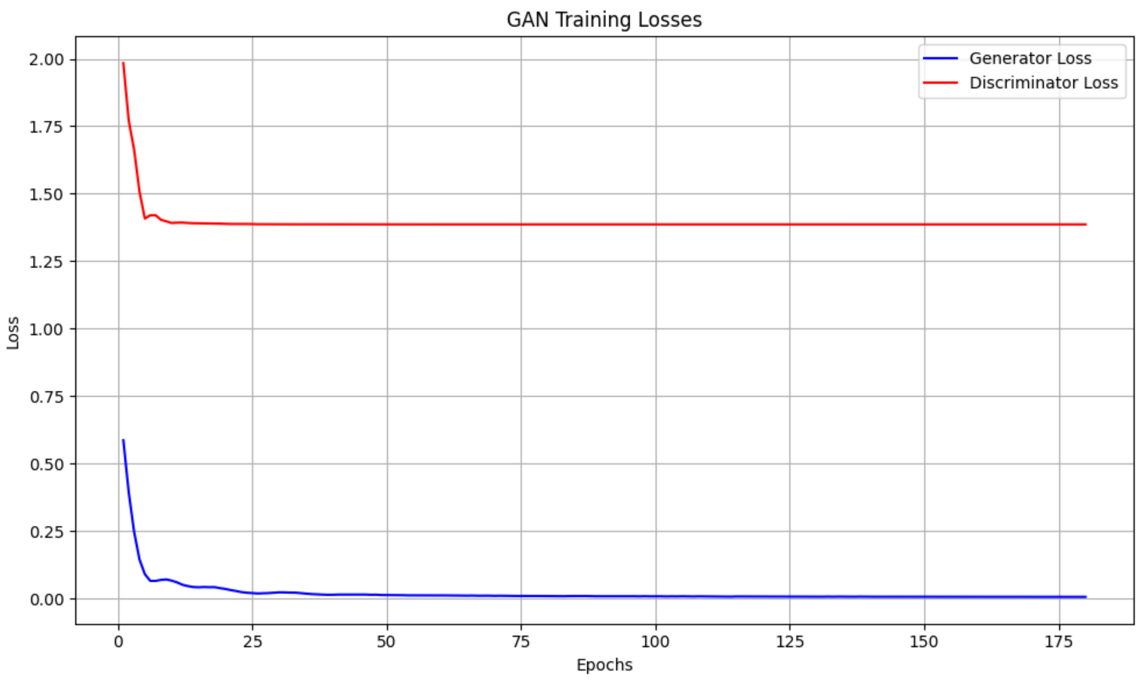


Fig. 4.20 The loss of GAN model for sequence length of 30 days

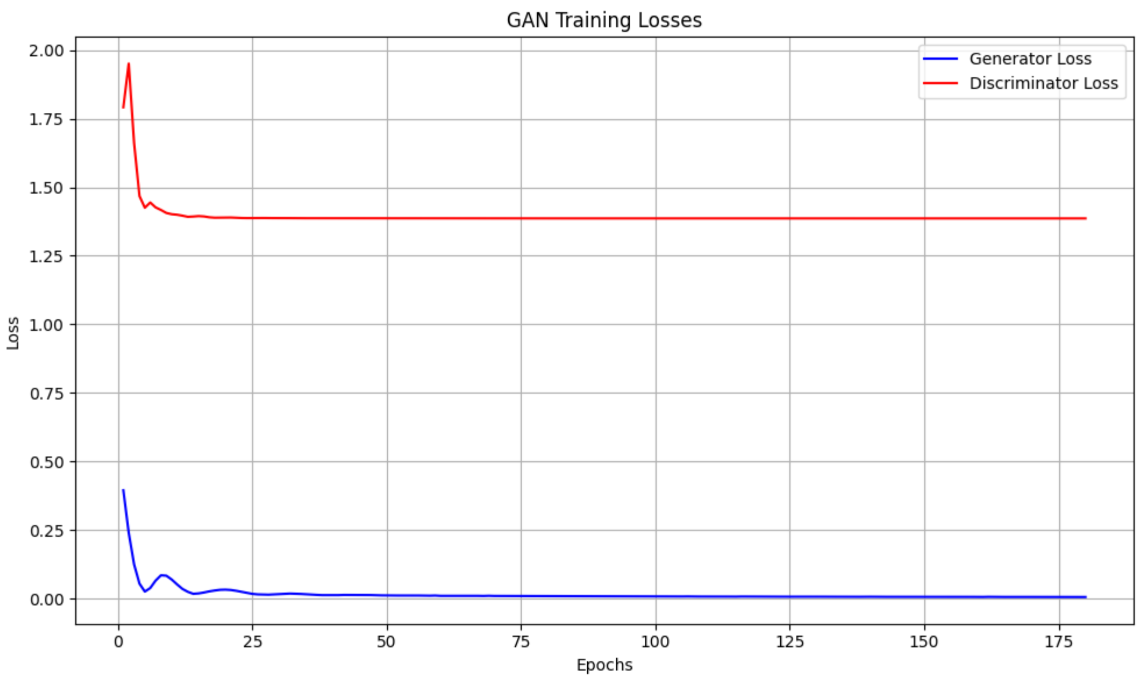


Fig. 4.21 The loss of GAN model for sequence length of 60 days

Chapter 5

Evaluation

In this section, two primary evaluation methods are discussed. The Root Mean Square Error (RMSE) validates the proximity of the predicted prices of each model and the real closing prices. For the Sharpe ratio, the model will be assessed for its performance when converting the prices generated from the model to a trading system.

5.1 Root Mean Squared Error (RMSE)

The models have been evaluated using Root mean squared error (RMSE) of the training set and the testing set. The RMSE defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5.1}$$

where y_i represents the actual closing price and $\hat{y}_i$ denotes the predicted closing price of the model and n denoted the number of the dates of the closing prices observed.

The results of the LSTM and GRU model are described in Table 5.1 and Table5.2, while the results of the GAN model are depicted in Table 5.3.

	LSTM			
Sequence Length	7	15	30	60
Training RMSE	9.827	5.68	6.803	6.276
Testing RMSE	8.83	7.695	7.904	7.774

Table 5.1 Performance of the LSTM model with varying sequence lengths

	GRU			
Sequence Length	7	15	30	60
Training RMSE	7.37	7.279	6	5.895
Testing RMSE	7.37	6.625	5.794	6.448

Table 5.2 Performance of the GRU model with varying sequence lengths

	GAN			
Sequence Length	7	15	30	60
Training RMSE	4.729	4.004	4.293	4.326
Testing RMSE	7.286	6.403	6.3	6.975

Table 5.3 Performance of the GAN model with varying sequence lengths

The results have been evaluated by the dataset included both the dates having sentiment scores and without sentiment scores. The total records of this dataset are 4017 for training set and 152 for testing set.

Moreover, the dataset excluded the dates having sentiment scores has been used to evaluate and validate the effect of the sentiment towards the performance of the GAN model. The records of both training and testing sets are the same as the dataset having both dates (4017 records and 152 records). The results are described in Table 5.4 compared to the RMSE scores of the dataset having both dates.

	GAN							
	With/Without sentiment score				Without sentiment score (4017 Records)			
Sequence Length	7	15	30	60	7	15	30	60
Training RMSE	4.729	4.004	4.293	4.326	5.005	3.93	3.857	4.528
Testing RMSE	7.286	6.403	6.3	6.975	6.32	6.707	5.926	6.38

Table 5.4 Comparison of GAN model performance with and without sentiment score (4017 records)

According to Table 5.4, it can be seen that the average RMSE scores of training and testing set of the dataset without sentiment score is lower than the dataset having both dates. For further evaluation, the model has been evaluated by the dataset having sentiment score and the dataset without sentiment score. The records of both datasets have been equally used around 221 for fair evaluation, and the results are described in Table 5.5, which can clearly seen that the sentiment score has an impact on model performance at some extent, since the dataset having sentiment score yields the lower RMSE scores in all aspects, both training and testing set and all sequence lengths.

	GAN							
	With sentiment score				Without sentiment score (221 Records)			
Sequence Length	7	15	30	60	7	15	30	60
Training RMSE	4.028	3.904	3.834	3.803	4.08	3.929	3.93	3.898
Testing RMSE	7.444	7.634	7.435	8.163	22.807	27.351	30.554	21.637

Table 5.5 Comparison of GAN model performance with and without sentiment score (221 records)

The TimeGPT model was evaluated on its ability to generate future closing prices using unseen data that was not included in the training set. The results indicate that the model was able to learn the complex patterns in the training data very well. However, it struggled significantly to generalize that domain knowledge to generate realistic future prices for the unseen data in the testing set, as illustrated in Table 5.6.

This is evident from the stark contrast between the low root mean squared error (RMSE) scores for the training set, and the much higher RMSE scores for the testing set across all prediction horizons. This suggests the model had difficulty extrapolating the learned patterns to make accurate forecasts on new and unfamiliar data.

The disparity in performance between the training and testing sets highlights the model's limitations in generalizing its learning beyond the data it was trained on. This points to a need for further improvements in the model's ability to transfer its knowledge to make reliable predictions on unseen market data.

	TimeGPT		
Frequency/Time Horizon	Daily/152 days	Weekly/21 weeks	Monthly/5 months
Training RMSE	2.429	2.085	8.333
Testing RMSE	39.298	44.416	73.170

Table 5.6 The performance of TimeGPT model generating the closing prices

To further validate performance of the models, the predicted prices were plotted against the actual prices, allowing for a visual comparison of their proximity. Figures 5.1 -5.4 display prediction of the LSTM model, Figures 5.5 -5.8 represent forecasts of the GRU model, and Figures 5.9 -5.12 illustrate results of the GAN model. These graphs, depicting real and predicted closing prices for each model, reveal a consistent trend: *The shorter the input data sequence used for prediction, the closer the predicted prices align with the actual prices*. This pattern is particularly evident in results of the GAN model, where the prediction of the 7-day input sequence (Figure 5.9) demonstrates noticeably closer alignment with real prices compared to the prediction of 60-day input (Figure 5.12).

Additionally, the predicted prices and the real closing prices have also been plotted from the prediction result receiving from TimeGPT model as depicted in Figure 5.17-5.19. These results demonstrate a substantial discrepancy between the actual and forecasted closing prices, resulting in poor RMSE scores during model evaluation and indicating sub-optimal predictive performance.

Based on the analysis of all the graphs presented, the Generative Adversarial Network (GAN) model demonstrates the closest proximity between the real and forecasted prices, outperforming the Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and TimeGPT models, respectively.

This suggests that the GAN model produces the best overall performance in terms of the closeness or distance between the expected and predicted data points (closing prices). The GAN model's ability to generate the most similar forecasted prices to the actual prices indicates its superior capability in capturing the underlying patterns and dynamics of the data.

In contrast, the LSTM and TimeGPT models exhibit greater distances between the real and predicted prices, suggesting they are struggle more in accurately forecasting future price movements compared to the GAN model.

The comparative analysis of the graph-based results highlights the GAN model as the most effective in generating forecasts that closely match the real-world price behavior, making it a potentially preferred choice for systems requiring accurate price predictions.

However, the GAN model was also evaluated to extract the underlying insights of the model prediction. The changes of the real prices against the forecasted prices have been visualized to validate the price changes over the specified previous trading dates (Sequence length of 7 days, 15 days, 30 days, and 60 days) as represented in Figure 5.13-5.16. According to the results, these graphs highlight the mode's learning ability to forecast future prices that closely match the prices from yesterday.

Due to this, the GAN model has been compared to the baseline of the "*predict same price as yesterday*" model. The baseline was created from the *naive* calculation between distance of current price and the price of yesterday. Figure 5.20 depicts the graph of real and predicted closing prices from Naive model. The comparative results of GAN and Naive models are represented in the Table 5.7 .

The comparative results between the Naive and GAN models presented in Table 5.7 suggest that although the GAN model may exhibit better performance compared to other predictive models like GRU and LSTM, it has not yet surpassed the Naive model in overall performance.

	Naive	GAN			
Sequence Length	-	7	15	30	60
Training RMSE	2.346	4.729	4.004	4.293	4.326
Testing RMSE	4.024	7.286	6.403	6.3	6.975

Table 5.7 Comparison of model performance between Naive and GAN models

However, the GAN model's performance is not drastically behind the Naive model, indicating there is a relatively small margin for improvement. This highlights potential areas for further optimization of the GAN model to enhance its capabilities.

Although the GAN model cannot outperform the Naive model, the performance of the GAN model is still sufficiently desirable to be considered as a useful assistant. This can be done by receiving the signal from the model plus taking input from human experts to make a comprehensive decision before purchasing and selling the stock in a trading system.

Overall, the comparative analysis suggests the GAN model holds promise yet still requires additional refinement and tuning in order to fully capitalize on its advantages over simpler Naive forecasting approaches. Continuing efforts to improve generalization ability of the GAN model is more likely to lead the model to be more compelling option for trading applications.

5.2 Accuracy

Apart from the RMSE evaluation, effectiveness of the GAN model for forecasting the change of the prices was evaluated. This evaluation focused on the ability of the model to correctly predict whether the price would increase or decrease. The performance was quantified using an accuracy metric, defined as follows:

$$\text{Accuracy (\%)} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100 \tag{5.2}$$

This metric provides a distinct measure of proficiency of the model to anticipate directional movement of stock prices, which offers insights of its potential utility in trading decisions considering future trends of predicted prices.

The accuracy of the GAN model for forecasting the change of the price over specified period was represented in Table 5.8.

According to the table, shorter sequence length like 7, 15, and 30 days yield identical accuracy around 61-65 %, while the 15-day sequence length produce the highest accuracy at *69.853 %*. On the other hand, the longest sequence length, like 60 days, provide the most inferior accuracy at 21.978 %. These results highlight effectiveness of the GAN model in predicting price changes when considering shorter historical data sequences to forecast future prices, while *15-day* sequence is the most powerful setting to predict the next closing prices, followed by 7 days and 30 days.

	GAN			
Sequence Length	7	15	30	60
Correct forecasting ratio	0.653	0.699	0.620	0.220
Incorrect forecasting ratio	0.347	0.301	0.380	0.780
Accuracy (%)	65.278	69.853	61.983	21.978

Table 5.8 Accuracy performance for forecasting price change from GAN model

5.3 Sharpe Ratio

Following comprehensive model evaluation, the optimal model was selected to convert predicted closing prices into profits and calculate the Sharpe ratio. The model is The GAN model that was trained on a dataset encompassing both dates with and without sentiment scores due to its overall performance.

To analyze the Sharpe Ratio score, a trading strategy was created. The strategy is that the system will purchase stock when the forecasted price change over specified periods (7, 15, 30, and 60 days) is positive and sell or maintain zero holdings otherwise. To be more precise, if the current date is 20/8/2024, the current price is 400$, and the day of price change is set to 7 days, the system will validate whether the future price forecasted by the model for the next date, 27/8/2024, is greater than 400$ (positive price change). If there is a positive price change, the system will buy the stock at the price of that day (27/8/2024).

The trading days are *252 days* and the risk-free rate is *2%* and *no risk (0%)*. The Sharpe ratio defined as:

$$\mathrm{S} = \frac{R\_p - R\_f}{\sigma\_p} \tag{5.3}$$

where R_p is the portfolio return, R_f is the risk-free rate, and σ_p is the portfolio standard deviation, was calculated to assess risk-adjusted performance.

The results, as represented in Table 5.9 and Table 5.10 reveal a concerning trend. For shorter sequence lengths (7 and 15 days), the Sharpe ratios are significantly negative while making a greater profits compared to the longer sequence length (60 days). The 60-day sequence length can generate the smaller profit yet producing the compromised Sharpe ratio at *1.776* for 2% risk-free rate and *1.785* when no risk considered. However, the Sharpe ratio of 30-day sequence length is negative while the loss occurred. The results from both 2% risk-free rate and no risk are similar and represent a small improvement for sequence length of 7 and 60 days whereas the same results are given for both 15 and 30 days.

The results show the the shorter the sequence length is, the more the final profit increases dramatically. Given this, the relationship between the increasing profit and the more negative Sharpe Ratio can be explained, for the shorter sequence lengths (7 days and 15 days), the final profit is higher compared to the longer sequence lengths (30 days and 60 days). This suggests the trading strategy is able to generate larger profits in the short term since the trading strategy may be able to capitalize on market inefficiencies or temporary trends. Despite this, a more negative Sharpe Ratio indicates the strategy is generating returns that are not commensurate with the level of risk taken, as evidenced by the more negative Sharpe Ratio for the shorter sequence lengths. In contrast, as the sequence length increases, the strategy's ability to consistently generate high profits while managing risk effectively appears to be reasonable, resulting in lower final profits but a less negative Sharpe Ratio.

In essence, the trading strategy seems to be optimized for short-term profitability at the expense of long-term risk-adjusted performance. This is a common challenge in trading, where strategies that perform well in the short term (by making a higher profit) may not necessarily be sustainable or optimal in the long run (when considering the risk taken).

To improve the performance of this trading strategy, further analysis and optimization would be required to find the right balance between short-term profits and long-term risk-adjusted returns.

	GAN			
Sequence Length	7	15	30	60
Sharpe Ratio	-2.496	-2.129	-1.945	1.775
Profit ($)	73.19	46.44	-1.57	14.02

Table 5.9 The Sharpe ratio of the GAN model with varying sequence lengths for 0.02 risk-free rate

	GAN			
Sequence Length	7	15	30	60
Sharpe Ratio	-2.495	-2.129	-1.945	1.786
Profit ($)	73.19	46.44	-1.57	14.02

Table 5.10 The Sharpe ratio of the GAN model with varying sequence lengths for no risk

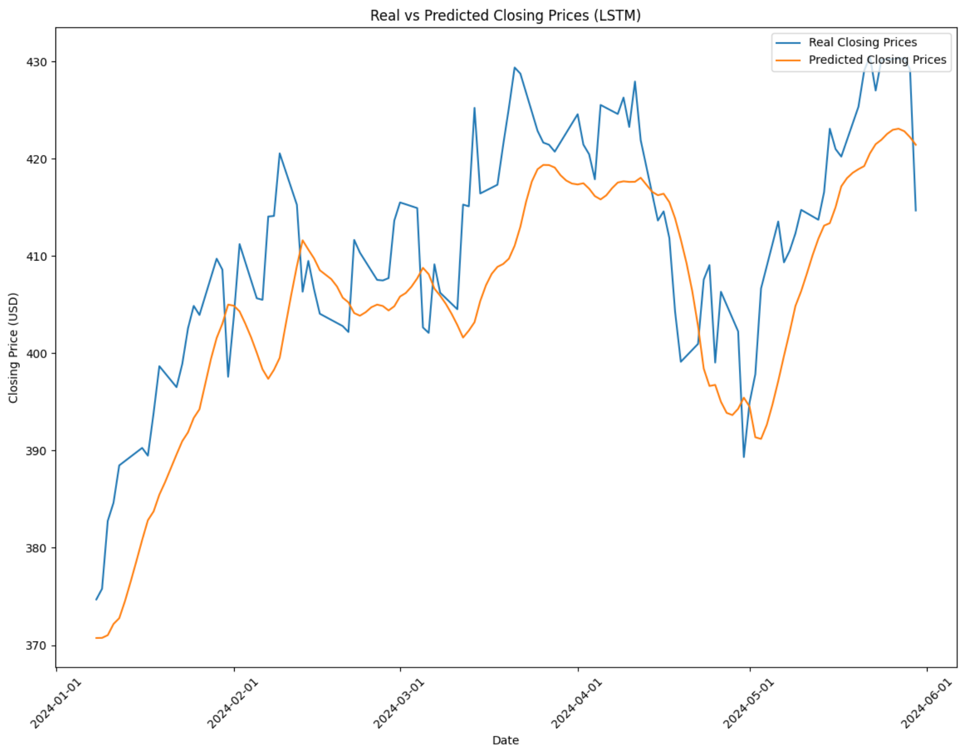


Fig. 5.1 The closing price of real price and predicted price from the LSTM model for the sequence length of 7 days

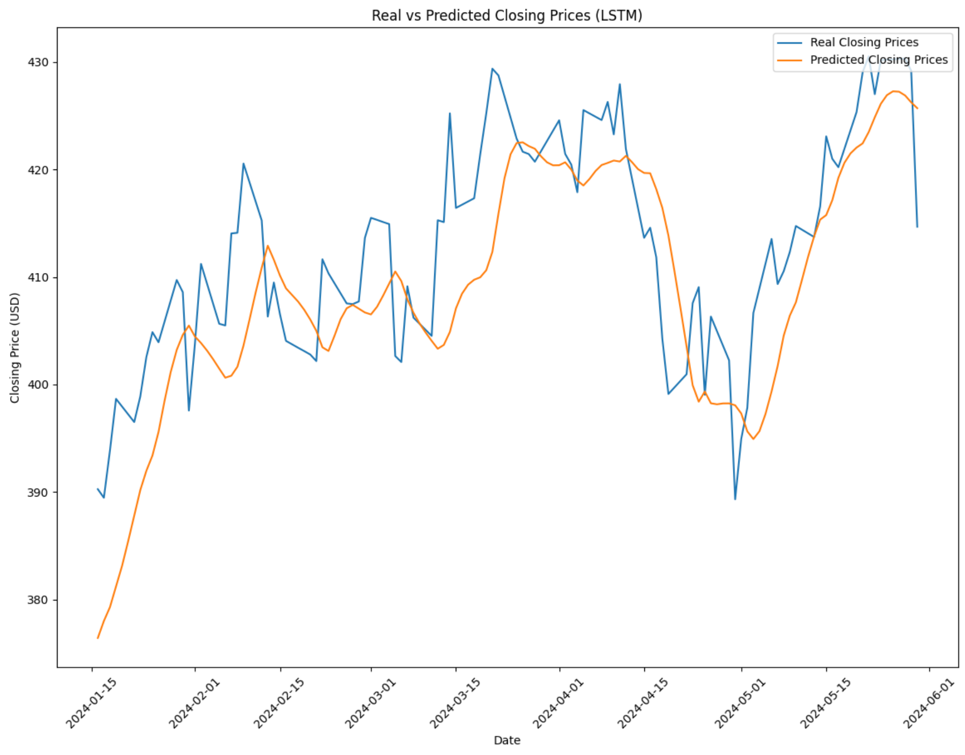


Fig. 5.2 The closing price of real price and predicted price from the LSTM model for the sequence length of 15 days

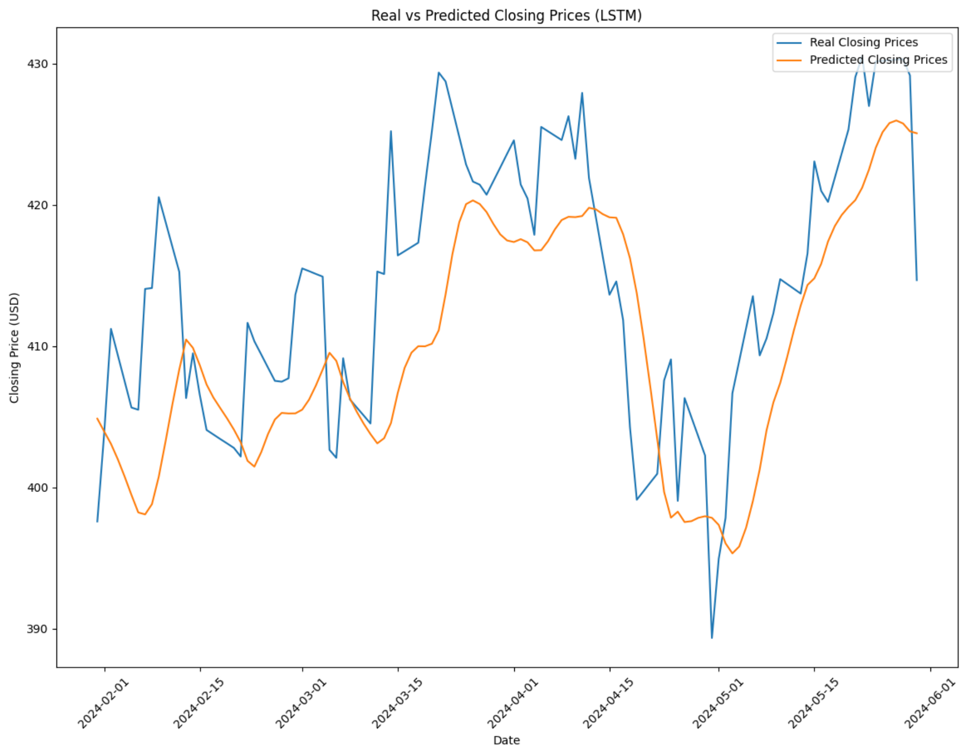


Fig. 5.3 The closing price of real price and predicted price from the LSTM model for the sequence length of 30 days

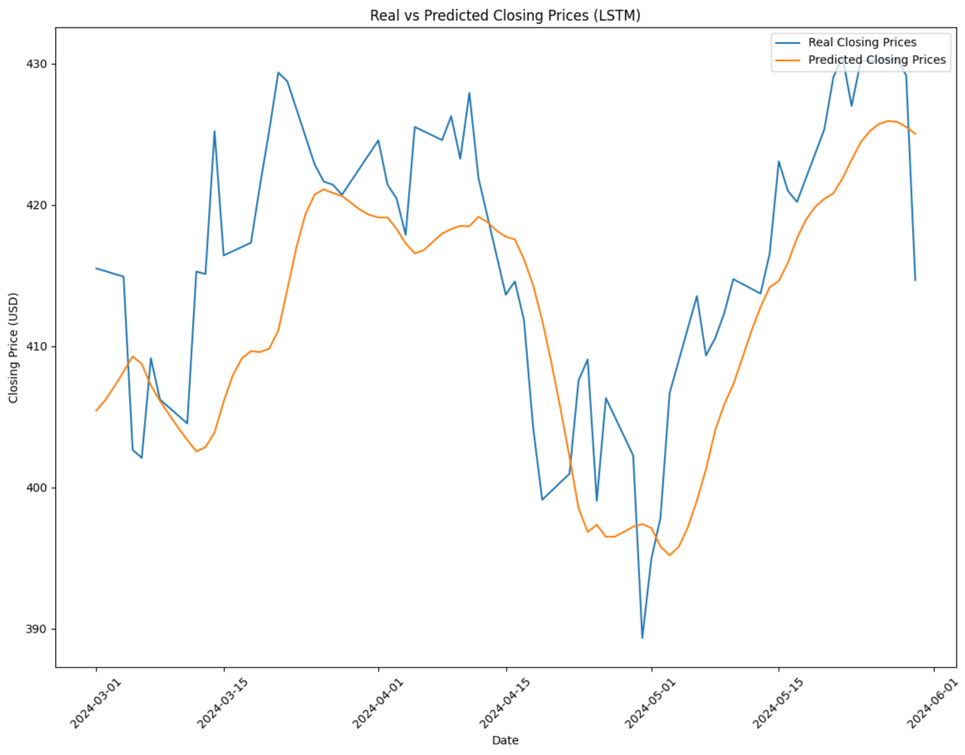


Fig. 5.4 The closing price of real price and predicted price from the LSTM model for the sequence length of 60 days

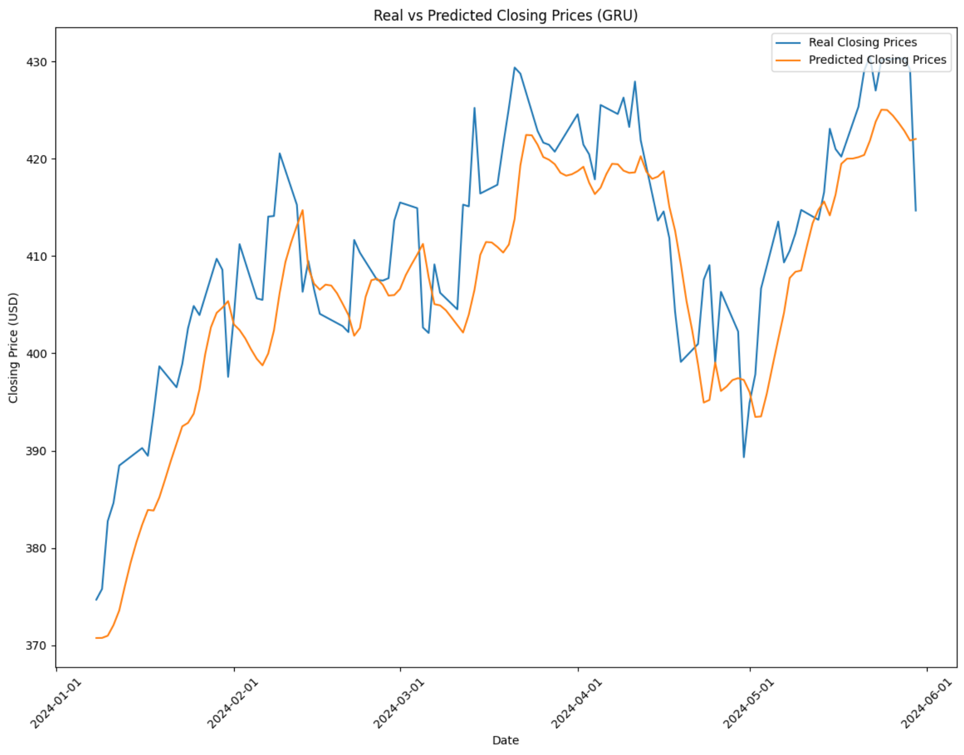


Fig. 5.5 The closing price of real price and predicted price from the GRU model for the sequence length of 7 days

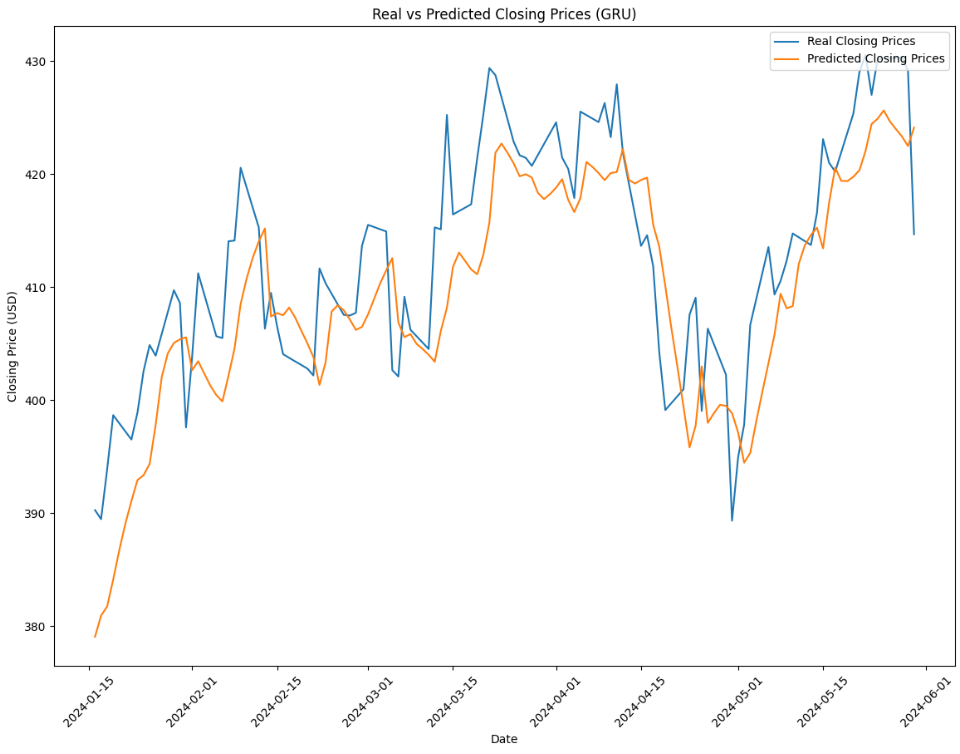


Fig. 5.6 The closing price of real price and predicted price from the GRU model for the sequence length of 15 days

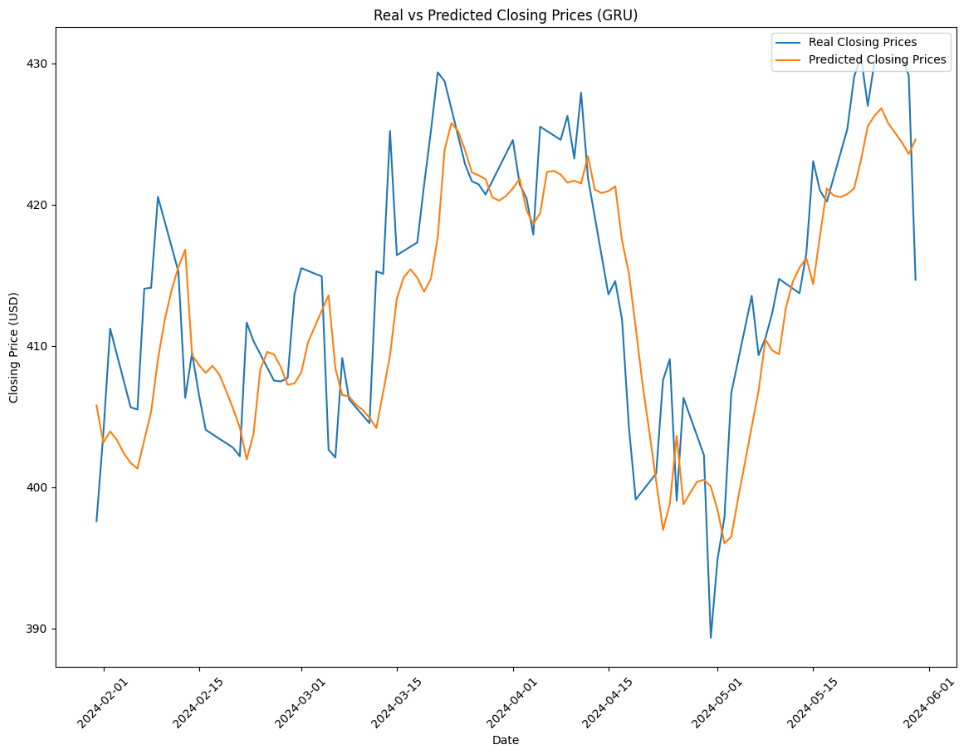


Fig. 5.7 The closing price of real price and predicted price from the GRU model for the sequence length of 30 days

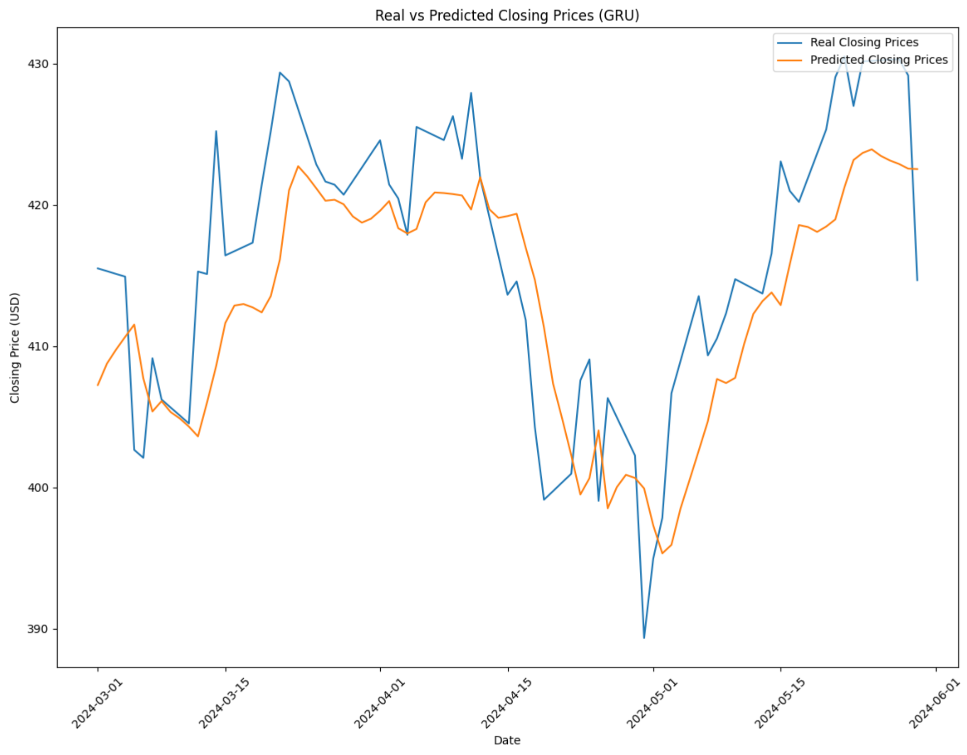


Fig. 5.8 The closing price of real price and predicted price from the GRU model for the sequence length of 60 days

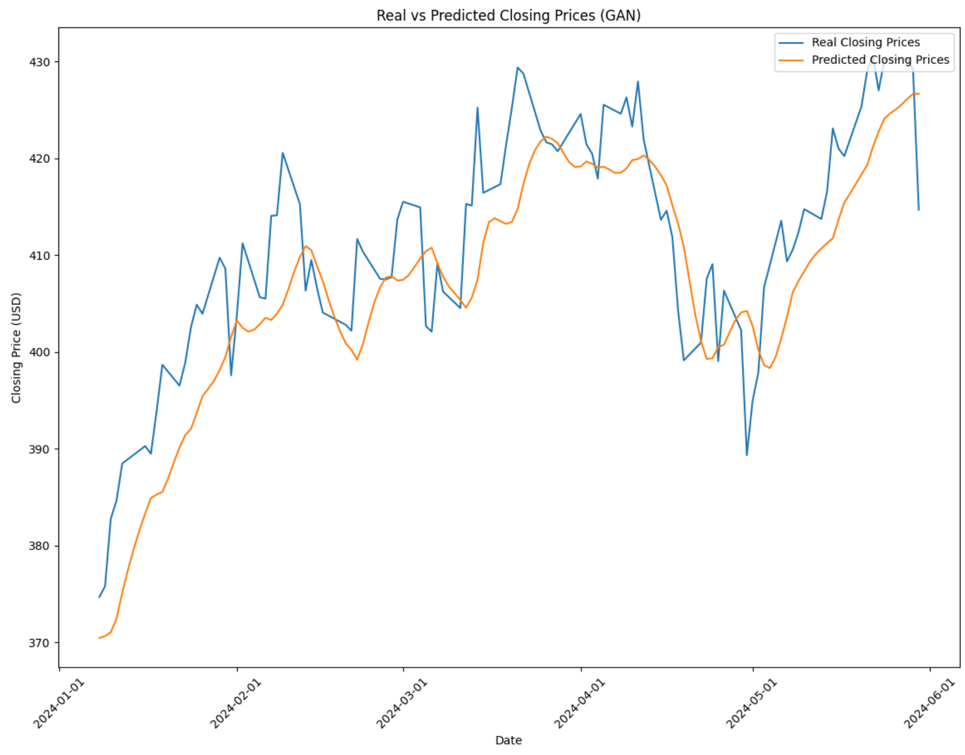


Fig. 5.9 The closing price of real price and predicted price from the GAN model for the sequence length of 7 days

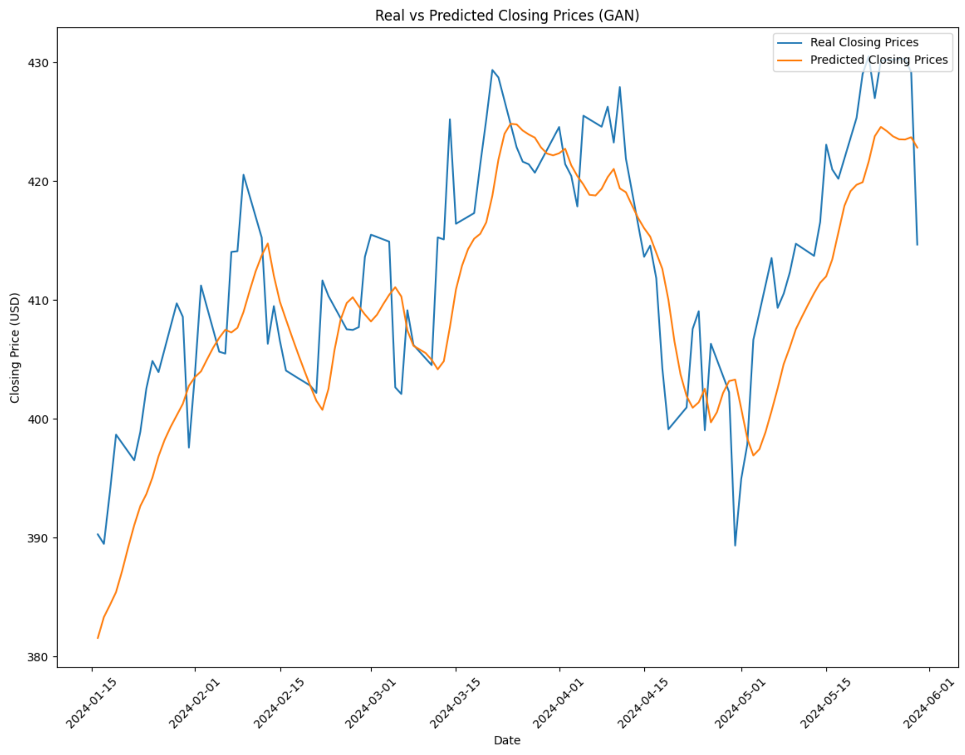


Fig. 5.10 The closing price of real price and predicted price from the GAN model for the sequence length of 15 days

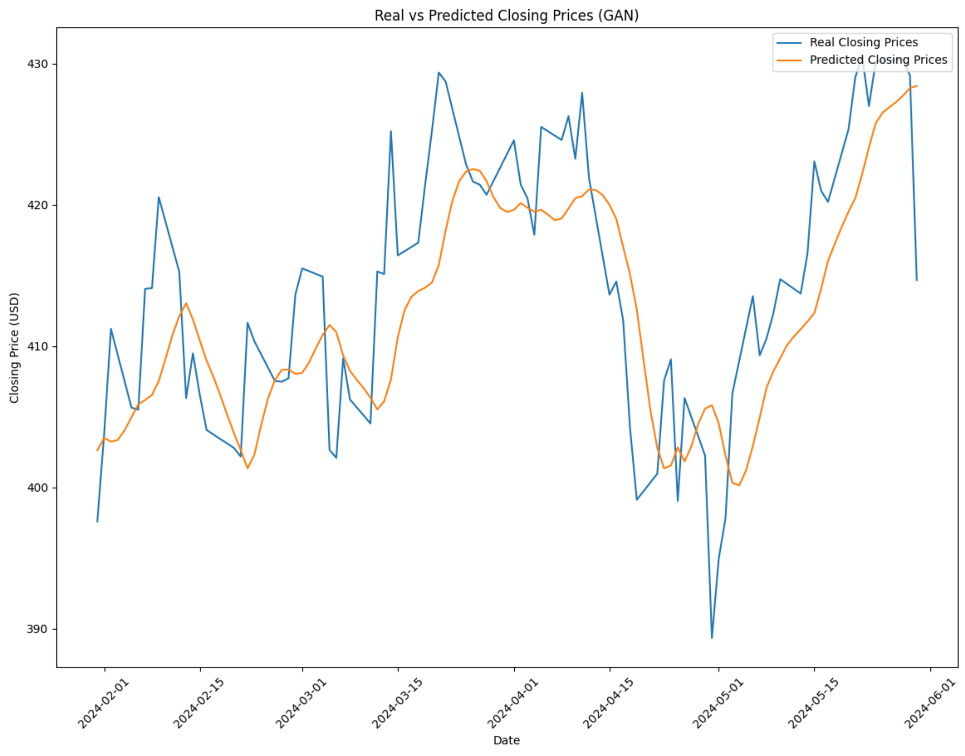


Fig. 5.11 The closing price of real price and predicted price from the GAN model for the sequence length of 30 days

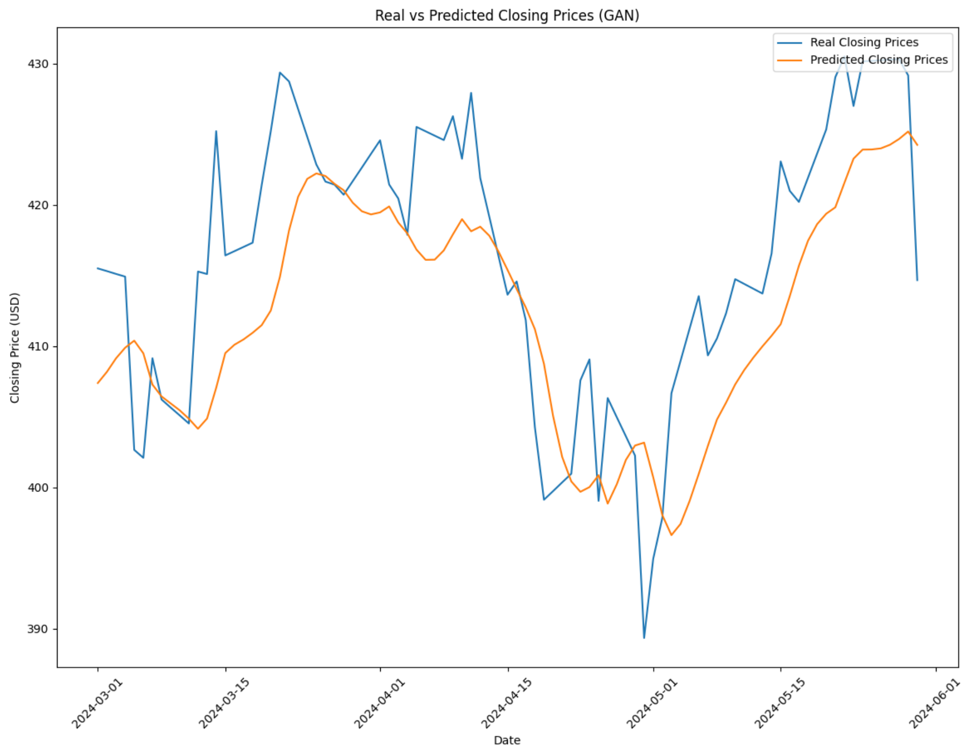


Fig. 5.12 The closing price of real price and predicted price from the GAN model for the sequence length of 60 days

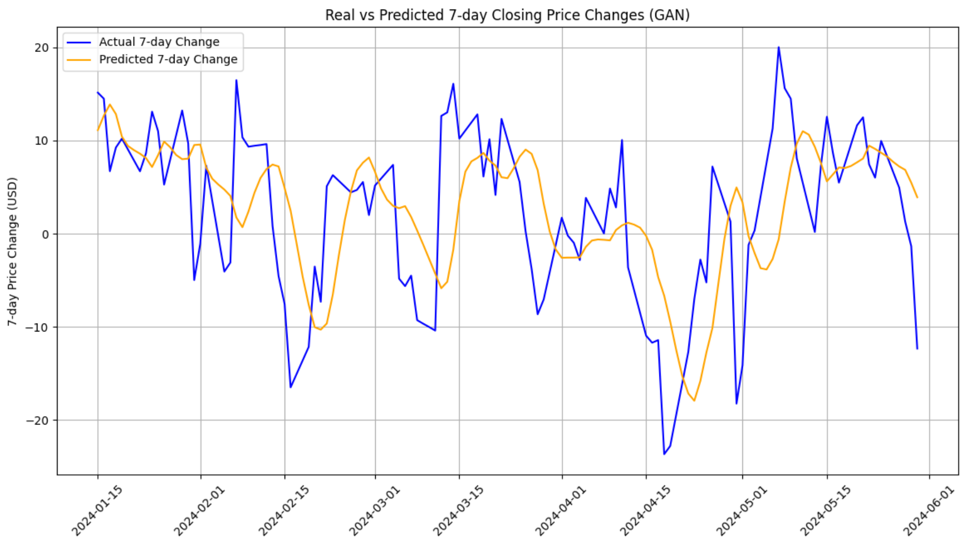


Fig. 5.13 The change of the real closing prices and the forecasted closing prices from the GAN model for the sequence length of 7 days

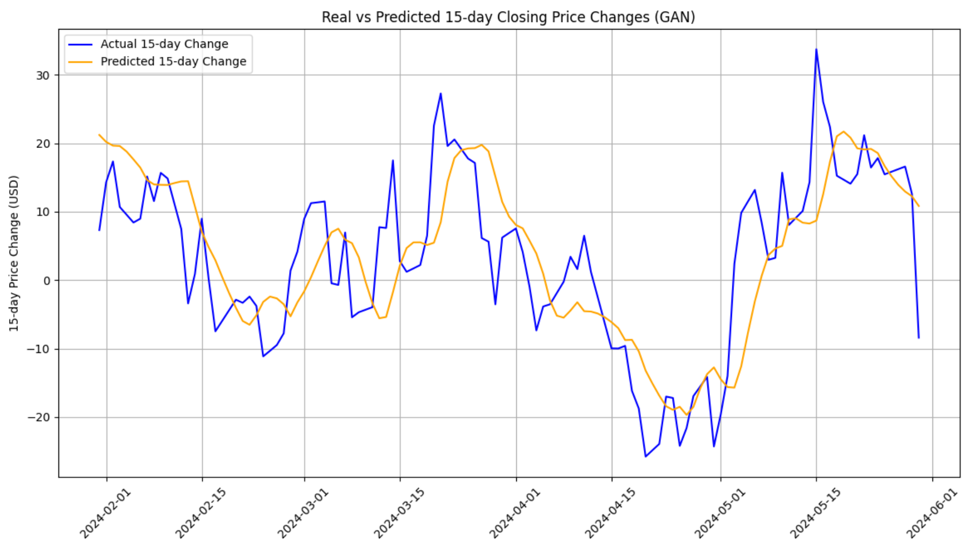


Fig. 5.14 The change of the real closing prices and the forecasted closing prices from the GAN model for the sequence length of 15 days

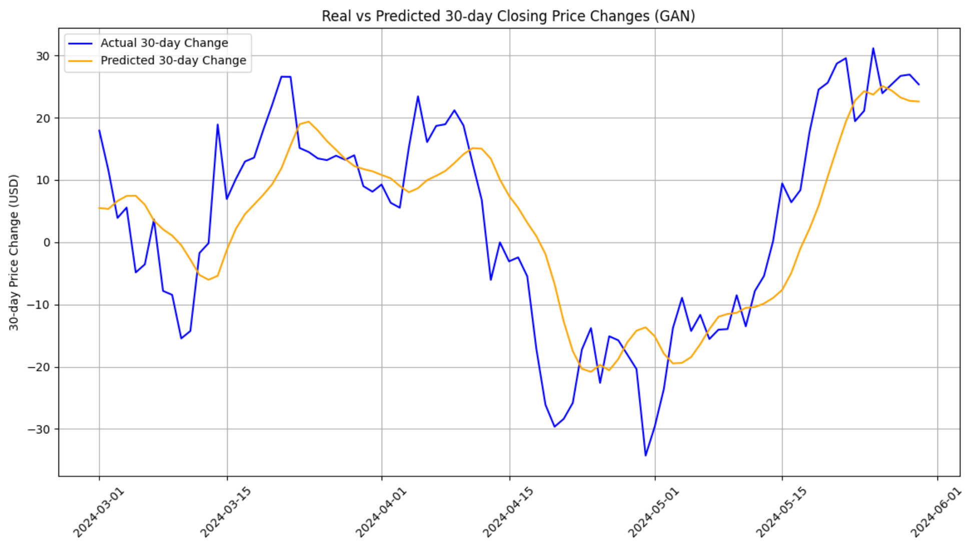


Fig. 5.15 The change of the real closing prices and the forecasted closing prices from the GAN model for the sequence length of 30 days

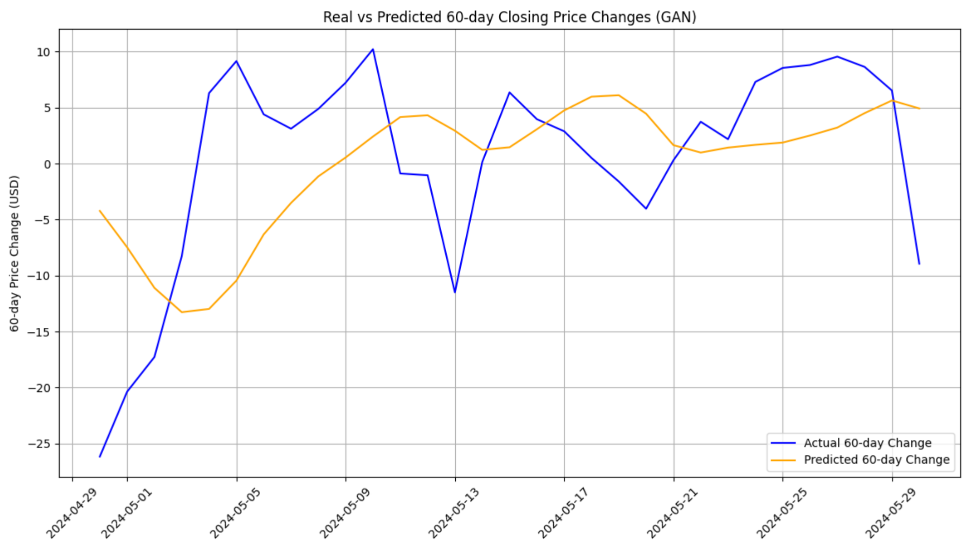


Fig. 5.16 The change of the real closing prices and the forecasted closing prices from the GAN model for the sequence length of 60 days

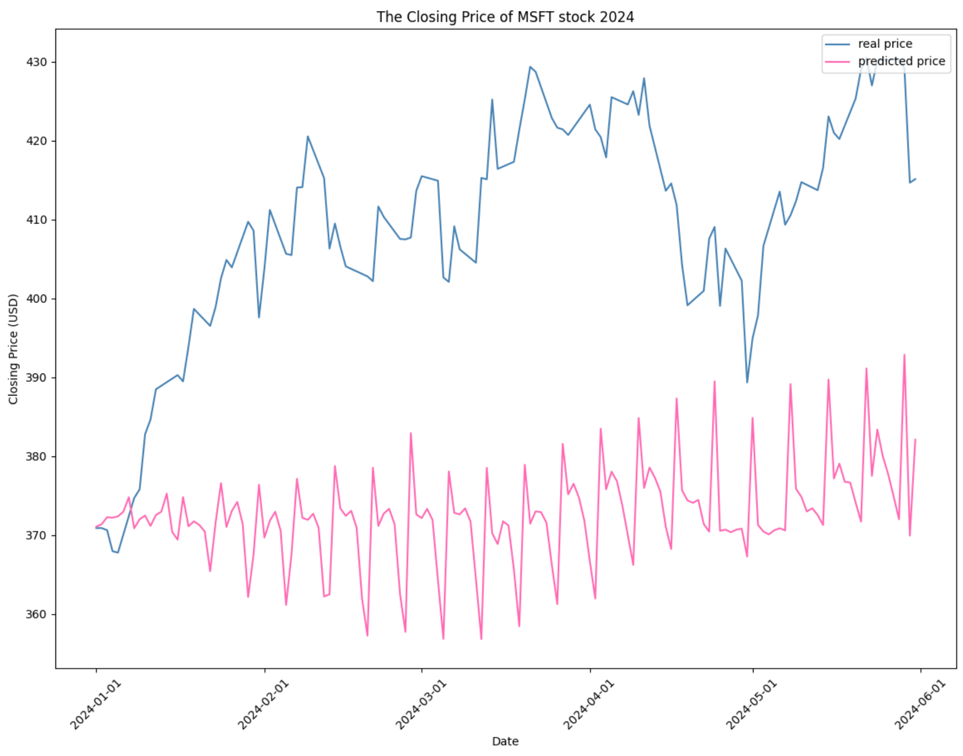


Fig. 5.17 The closing price of real price and predicted price from the TimeGPT model for the daily prediction of 152 days ahead

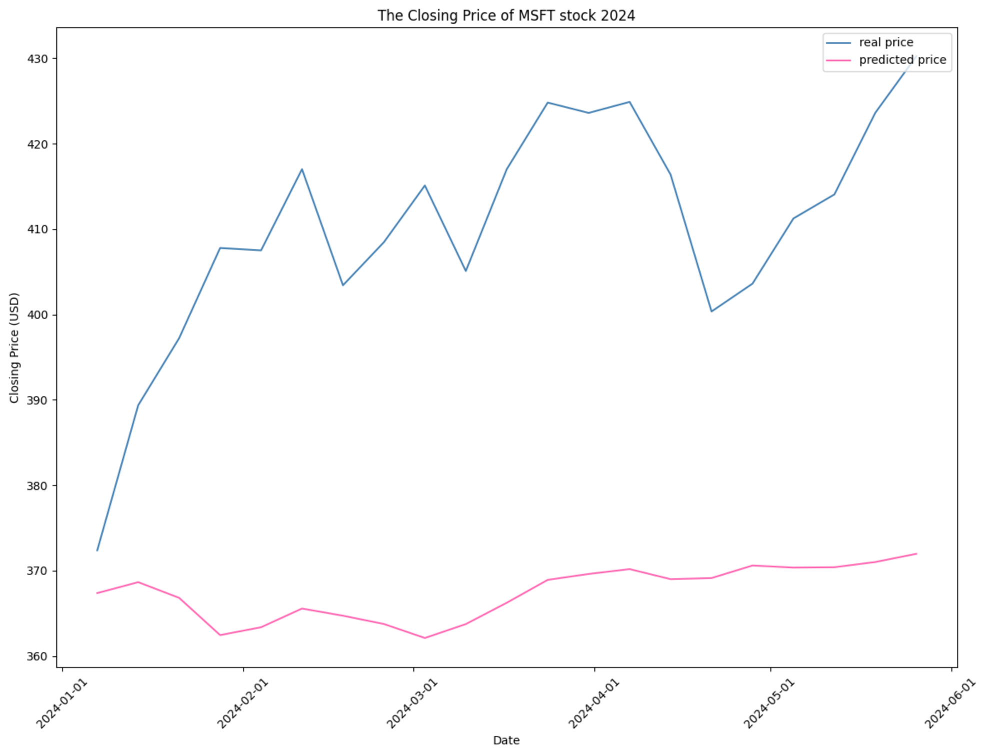


Fig. 5.18 The closing price of real price and predicted price from the TimeGPT model for the weekly prediction of 21 weeks ahead

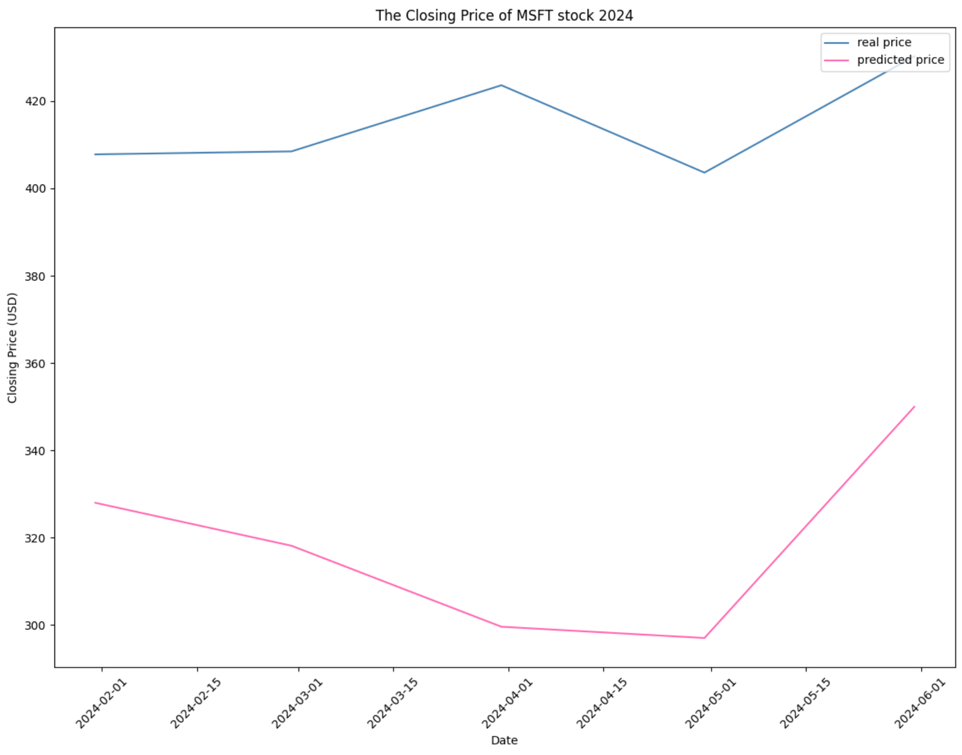


Fig. 5.19 The closing price of real price and predicted price from the TimeGPT model for the monthly prediction of 5 months ahead

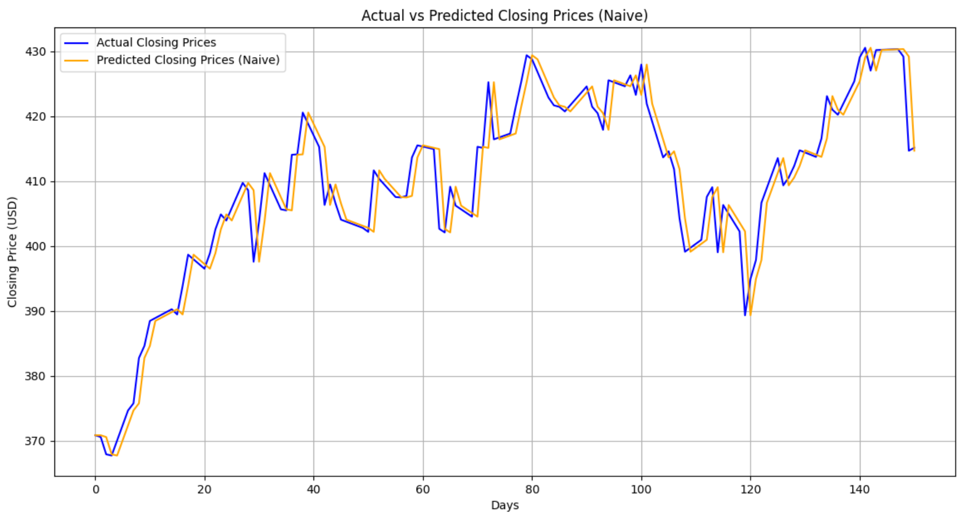


Fig. 5.20 The closing price of real price and predicted price from the Naive model

Chapter 6

Conclusion

The research proposed a novel *SE-GAN (Sentiment-Enhanced Generative Adversarial Network)* framework for stock price prediction. The architecture employs a GRU (Gated Recurrent Unit) as the generator and an MLP (Multi-Layer Perceptron) as the discriminator, while also incorporating market sentiment scores. A comprehensive evaluation of this model's performance is presented in Chapter 5, with key findings summarized below.

The GAN model consistently outperforms the LSTM model across all sequence lengths, as evidenced by lower root mean squared error (RMSE) for both training and testing sets. This suggests the GAN model exhibits better performance in all experimental settings than those of the LSTM model.

Nevertheless, compared to the GRU model, the GAN model introduces the mixed results. The GAN model achieves lower RMSE scores across all sequence lengths for the training set, which demonstrates its ability to learn complex patterns better than that of the GRU model. However, performance of the GAN model varies depending on the sequence lengths setting for the testing set.

Precisely, the GAN model maintains the lower RMSE scores for shorter sequences like 7 days and 15 days, indicating the better generalization of the model on unseen data. However, the RMSE scores of the GAN model are slightly higher than those of the GRU mode for longer sequences like 30 days and 60 days input.

Overall, these results highlight ability of the GAN model at excelling to capture and learn complex patterns in the training data, while maintaining strong generalized capabilities on unseen data, particularly for short-term sequences like 7 and 15 days input.

This performance indicates the effective memory of the model and pattern recognition for shorter time horizons. The slight decrease in performance for longer sequences in the testing set suggests further optimization may be required to capture extended temporal dependencies in regard to the effectiveness of the GAN model.

Another study has set up an experiment to validate the impact of market sentiment on the model prediction.

An equal set of data has been allocated both for the dataset having sentiment scores and the dataset having no sentiment scores. Due to the limited number of headlines provided for the intended date of 2013-01-01 to 2023-12-31. The experimental setting utilized a dataset comprising 221 records.

The results demonstrate the incorporation of sentiment scores significantly enhances the learning capability of the model. The research findings exhibits improved performance of the proposed model, as evidenced by lower RMSE scores across both the training and testing datasets. Importantly, this enhancement in model performance is achieved by the inclusion of sentiment data as an additional input feature, alongside the various sequence lengths considered in the analysis.

The combination of lower RMSE values suggests the proposed approach coupled with the incorporation of sentiment data is better able to capture the relevant signals and patterns in the stock data, leading to more accurate predictions compared to models that do not leverage this sentiment information. This indicates the value of integrating sentiment data as a complementary signal to enhance the predictive capabilities of the financial forecasting model, beyond relying solely on the historical stock data and sequence-based features.

Furthermore, the study also proposed another evaluation approach to analyze the accuracy of the model in term of forecasting positive and negative price change, which will be further employed in trading strategy to evaluate the Sharpe ratio scores.

According to the result, shorter sequence lengths yielded comparable accuracy rates, ranging from approximately 60% to 70%. Notably, the 15-day sequence achieved the highest accuracy at exceeding 69%. This suggests that the model excels for forecasting the change of the prices based on small windows of historical data inputs. In contrast, performance of the model declined significantly for longer windows, as evidenced by the 60-day sequence, producing the lowest accuracy score of around 21%.

After RMSE and accuracy evaluation, the *SE-GAN (Sentiment-enhanced GAN)* model is the final proposed framework that was used to make comparisons against the Naive model and evaluate the resulting performance using the Sharpe ratio metric. The analysis shows that the SE-GAN model is capable of generating higher profitability when considering shorter trading signal windows.

The ability for generating higher profits with shorter sequences like 7 days and 15 days demonstrates capacity of the model in capturing more granular market dynamics and immediately react to the market conditions. However, this comes with the trade off regarding to accelerating higher associated risk, as indicated by the negative Sharpe ratio scores. Due to this concern, the elevated risk levels associated with the shorter-term trading requires careful evaluation and risk management from human traders.

Conversely, the ability of the SE-GAN model to produce desirable profit levels, while maintaining a more acceptable risk profile, as reflected in the positive Sharpe ratio, exhibits the sign of viable and promising approach for financial forecasting and trading application. However, the results also highlight the need of human decision, rather than relying solely on the output from the model.

In conclusion, *sentiment-enhanced GAN (SE-GAN)* model demonstrates the significant promise, especially for short-term price forecasting using historical data spanning from 7 days to 15 days. However, an effective trading strategy necessarily requires long-term sequence for the change of the prices, such as 60-day period. This is because the model still requires involvement of the balanced approach to deploy model in real-world trading systems. Due to this, the model has to leverage both predictive capabilities and human judgment for the optimization of risk-adjusted returns as well as to ensure the robustness of the overall trading strategy. While output of the model and the proposed trading strategy providing valuable guidance, the judgment of human input is still unavoidable. The synergy between both of them can foster the optimization of the outcome from actual investment setting. For this reason, this collaboration between model forecasting and human insight is crucial for effective real-world application.

6.0.1 Comparison of SE-GAN with other generative model

Apart from evaluation of the *SE-GAN* model, this research also leverages transfer learning techniques by utilizing weights from the pre-trained generative model, *TimeGPT*.

This approach was adopted in order to assess the generative capability of the GAN model in comparison with other alternative model. The results of this investigation reveal some useful insights.

Despite the ability of sophisticated pre-trained model trained on the large corpus of various time-series data to specifically design for time-series forecasting, the TimeGPT model, purposefully generalized for various time series tasks, exhibits a sign of overfitting problem. This can be seen that the model demonstrates a strong ability to classify pattern or seasonality in the training set, indicating effective learning during the training phase, struggles significantly for generalizing to future trends. Specifically, performance of the TimeGPT model on the training set suggests a high degree of data fitting, translating into the model learns all possible data points including the noise in the dataset. However, this proficiency does not generalize well to the testing data, in which the predictive accuracy of the model for future prices decreases dramatically. This disparity between training and testing performance shows a remarkable indicator of overfitting, suggesting the model is too closely customized to the specific pattern of the training data at the expense of broader generalizing capabilities. These findings highlight the ongoing challenges to develop generative models equally effectively learning from historical data and maintaining robust predictive power once facing with new and unseen market conditions. Also, this finding underscores the need of further refinement for transfer learning approach and model architecture to achieve a better balance between pattern recognition and generalization for stock price prediction.

6.0.2 Further Research

Further research direction aims to expand and refine capability of the SE-GAN model.

The model will be developed to predict stock prices across various time horizons, from on daily basis to forecasting prices on weekly, monthly, and yearly basis. This expansion allow to understand performance of the model across different prediction timescales more comprehensively, potentially uncovering its strength in long-term trend analysis as well as short-term fluctuation.

Furthermore, to assess versatility and robustness of the model, it will apply a wide range of stock dataset. This will include stocks from various sectors, market capitalization, and geographical regions. Such diversity will help evaluate generalizability of the model and identify patterns of any sector-specific or market-specific performance.

Advanced techniques for hyper-parameter tuning will be explored. The techniques are Bayesian optimization, genetic algorithms, or neural architecture search. This study plan aims to find optimal point of the SE-GAN model, which can improve performance of the model across various timescales prediction.

Additionally, a new transformer model will be built using the existing dataset without relying on pre-trained weights from the pre-trained model like TimeGPT. This approach will also introduce a fair comparison between a custom-built transformer model and the SE-GAN architecture on the specific dataset. This can provide insight of relative strength of different generative models for the domain of stock prediction.

In summary, this research direction will enable a comprehensive evaluation of predictive capabilities, specifically for stock price forecasting, of deep learning models across various time scales. The finding of the investigation will contribute significantly to the field of financial modeling, which potentially leads to a more robust and accurate predictive tool for stock market analysis.

References

[1] K. Tretina and B. Curry, “What Is The Stock Market? How Does It Work,” Forbes Advisor, 2023. [Online]. Available: https://www.forbes.com/advisor/investing/what-is-the-stock-market/

[2] J.-L. Wu, X.-R. Tang, and C.-H. Hsu, “A prediction model of stock market trading actions using generative adversarial network and piecewise linear representation approaches,” *Soft Computing*, vol. 27, pp. 8209–8222, 2023.

[3] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, “Stock Market Prediction Based on Generative Adversarial Network,” *Procedia Computer Science*, vol. 147, pp. 400–406, 2019.

[4] P. Sonkiya, V. Bajpai, and A. Bansal, “Stock price prediction using BERT and GAN,” 2021.

[5] R. Jadhav, Sh. Sinha, S. Wattamwar, P. Kosamkar, *Leveraging Market Sentiment for Stock Price Prediction using GAN*, 2021.

[6] H. Lin, C. Chen, G. Huang, and A. Jafari, “Stock price prediction using Generative Adversarial Networks,” *Journal of Computer Science*, vol. 17, no. 3, pp. 188–196, 2021.

[7] S. Hochreiter and J. Schmidhuber, ’Long short-term memory,’ *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[10] I. Goodfellow *et al.*, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.

[11] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[12] D. Araci, “FinBERT: Financial sentiment analysis with pre-trained language models,” *arXiv preprint arXiv:1908.10063*, 2019.

[13] Huang, Allen H., Hui Wang, and Yi Yang. "FinBERT: A Large Language Model for Extracting Information from Financial Text." *Contemporary Accounting Research (2022)*

[14] A. Garza, C. Challu, and M. Mergenthaler-Canseco. "TimeGPT-1",” *arXiv preprint arXiv:2310.03589v3*, 2024.

Appendix A

Project Repository

The repository of this research can be accessed in *23-24_CE901-CE911-CF981-SU_limbunlom_phrugsa*.

This research has been used the repository from the paper "*Stock price prediction using Generative Adversarial Networks*" by HungChun Lin et al [6]. as a reference to train and develop the GAN model. The repository can be accessed in *Stock-price-prediction-using-GAN*.

Appendix B

Project Structure

Structure of the project repository containing the source code and related files in the project repository.

```
- config
    - config.yaml
    - requirment.txt
- data-ingestion
    - historical-stock
        - collect_historical_prices.py
    - index
        - collect_index_prices.py
    - news
        - extract_news.py
- file
    - dataset
    - incorporate
    - index
    - news
    - preprocessed
    - sentiment
    - stock
- incorporate
    - incorporate_index_prices.py
    - sentiment_score_to_stock_daily.py
- model
    - BaseLine
        - testing
            - figure
            - GRU_model_test.log
            - GRU_test.py
            - LSTM_model_test.log
            - LSTM_test.py
        - training
            - figure
            - weight
            - base_model.log
            - BaseModel.py
            - DataPreprocessor.py
            - GRUModel.py
            - LSTMModel.py
            - main.py
        - baseline_model.log
        - baseline_model.py
    - GAN
        - sharpe_ratio
            - GAN_sharpe_ratio_comparison2.log
            - Sharpe_Ratio2.py
        - testing
            - figure
            - prediction
            - GAN_accuracy_test.log
            - GAN_accuracy_test.py
            - GAN_test.log
            - GAN_test.py
         - training
            - figure
            - weight
            - DataPreprocessor.py
            - Discriminator.py
            - EarlyStopping.py
            - GAN.log
            - GAN.py
            - GANTrainer.py
            - Generator.py
            - main.py
      - TimeGPT
         - figure
         - result
         - test
         - TimeGPT.log
         - TimeGPT.py
         - TimeGPT_test.log
         - TimeGPT_test.py
         - TimeGPT_visualization.py
- preprocessing
      - preprocessing.py
- sentiment-analysis
      - sentiment_analysis.py
- visualization
      - visualization.py
- dataset.py
```

B.0.1 Library

The libraries were used for developing and implementing the project.

- **pandas**: Data manipulation and analysis for handling structured data.
- **numpy**: The library used for scientific computation of multi-dimensional arrays and matrices.
- **eikon**: API for accessing financial data from Refinitiv used for retrieving stock data and headlines.
- **pyyaml**: YAML parser and emitter used for reading and writing YAML configuration files.
- **transformers**: Natural language processing library supported by Hugging Face. This library is used for retrieving FINBERT model for sentiment analysis task.
- **scikit-learn**: Machine learning library used for data normalization and model evaluation.
- **tensorflow**: Open-source machine learning framework developed by Google used for deep learning applications.
- **keras**: High-level neural networks API, running on top of TensorFlow, facilitating rapid prototyping of deep learning models.
- **nixtla**: Time series forecasting library applied for using TimeGPT model for price forecasting.
- **seaborn**: Statistical data visualization library built on top of Matplotlib, providing a high-level interface for drawing attractive statistical graphics.
- **matplotlib**: A library used for creating static and interactive visualization of the statistics data.
- **yfinance**: Library to download historical market data from Yahoo Finance API used for retrieving stock index data.
- **torch**: Open-source machine learning library (PyTorch) developed by Facebook's AI Research Lab used for deep learning and neural networks application.