ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

KHOA TOÁN - TIN
Faculty of Mathematics and Informatics

Decision Support Systems

Research Topic: Technology Sector Stock Price Prediction

Supervisor: Dr. Tran Ngoc Thang

Group 18:

Thai Duong Quach 20210253
Thai Duong Nguyen 20216806

Preface

In today's era of digitalization, the stock market is not merely a platform for trading equities, but a complex ecosystem where economic, political, and investor-sentiment factors collectively shape price volatility. Stock price forecasting therefore becomes a critical task—not only enabling investors to make more informed decisions, but also contributing to the stability and sustainable development of the financial market.

In this report, we present advanced models applied to equity price prediction, including traditional approaches such as VAR, as well as modern deep learning architectures such as LSTM, BiLSTM, and Transformer. The report explores the theoretical foundations behind each model, evaluates their strengths and limitations, and applies them to forecasting stock prices within the technology sector.

This report is structured into four main chapters:

- **Chapter 1: Theoretical Framework** provides an overview of the stock market, the key determinants of stock price movements, and the theoretical background of forecasting models including VAR, LSTM, BiLSTM, and Transformer.
- **Chapter 2: Technology Stock Forecasting Problem** focuses on the practical application of the aforementioned models for price prediction, including data analysis and preprocessing techniques.
- **Chapter 3: Experiments and Results** presents the experimental setup, evaluation metrics such as RMSE and MAPE, and the performance outcomes of the models. The chapter also includes model-error analysis and comparative assessment.
- **Chapter 4: Model Deployment** discusses the packaging and implementation of the forecasting models, including user interface design and execution scripts, to facilitate ease of use and real-world applicability.

With the aim of providing a comprehensive and in-depth perspective on stock price forecasting techniques within the technology sector, we hope that this report will serve as a valuable reference for researchers, investors, and other stakeholders interested in exploring and applying modern predictive methodologies in the stock market.

In addition, we would like to express our sincere gratitude to our academic supervisor, Dr. Tran Ngoc Thang, for his dedicated guidance and valuable insights throughout the development of this report.

Contents

1 Theoretical Foundations

1.1 Stock Market

1.1.1 Definition

Securities are tradable financial instruments that represent an ownership stake in a company or a debt obligation issued by a corporation or government entity. Securities encompass equities, bonds, and various types of derivative instruments.

- **Stocks (Equities)**: These are securities that represent fractional ownership in a company. Shareholders are entitled to receive dividends and have voting rights in major corporate decisions through participation in shareholder meetings.
- **Bonds**: These are debt securities issued by governments or corporations to raise capital. Bondholders earn fixed interest payments and receive the principal amount upon maturity.
- **Derivatives**: These are financial instruments whose value is derived from underlying assets such as stocks, bonds, commodities, interest rates, or market indices. Common derivative instruments include futures contracts, options, and swaps.

1.1.2 Stock Market

The stock market is a platform where securities are bought, sold, and exchanged. It plays a vital role in mobilizing capital for corporations and governments, while simultaneously offering an investment vehicle for individuals and institutions.

- **Primary Market**: This is where newly issued securities are offered for the first time by corporations or government entities to raise capital. New issuances are typically conducted through Initial Public Offerings (IPOs).

- **Secondary Market**: This is where previously issued securities are traded among investors. The secondary market provides liquidity, enabling investors to freely buy and sell securities.

1.1.3 Stock Indices

A stock market index is a performance metric that measures the price movement of a selected group of stocks, typically representing a specific market segment or industry. Key stock indices include:

- **Dow Jones Industrial Average (DJIA)**: Represents 30 of the largest U.S. corporations across various industrial sectors.
- **S&P 500 (Standard & Poor's 500 Index)**: Represents the 500 largest U.S. companies by market capitalization.
- **NASDAQ Composite Index**: Comprises all stocks listed on the NASDAQ exchange, with a high concentration of major technology companies.

1.1.4 Factors Influencing Stock Prices

Stock prices are influenced by a variety of factors, including:

- **Economic Factors**: Macroeconomic conditions such as GDP growth, interest rates, inflation, and exchange rates.
- **Firm-specific Factors**: Corporate financial performance, earnings, revenue, business strategies, and internal events such as leadership changes, mergers, and acquisitions.
- **Market Factors**: Supply–demand dynamics of the stock, investor sentiment, and overall market trends.
- **Political and Regulatory Factors**: Policy decisions, legal changes, and political events that may impact financial markets.

1.1.5 Stock Market Analysis

Stock analysis is the process of examining and evaluating information related to securities in order to make investment decisions. There are two primary analytical approaches:

- **Fundamental Analysis**: Focuses on assessing the intrinsic value of a security through economic, financial, and corporate factors. This method includes analyzing financial statements, evaluating management quality, and studying industry conditions.
- **Technical Analysis**: Concentrates on the study of price charts and trading volume to forecast future price movements. It employs technical tools and indicators such as moving averages, RSI, and MACD.

1.2 VAR Model

1.2.1 Introduction

The Vector Autoregression (VAR) model is a statistical framework used to capture dynamic relationships among multiple variables. Developed by Christopher Sims in 1980, this model introduced a new approach to time–series analysis. VAR enables effective forecasting and impact assessment of economic shocks. Owing to its flexibility and capacity to handle large datasets, VAR has become an indispensable tool in modern economic research.

1.2.2 Architecture

VAR model [5] has the architecture consisting of the following components:

a. Data: The input used in a VAR model typically consists of stationary time–series data in which all variables are endogenous. The value of each variable is determined by its own past values as well as the past values of other variables in the system.

b. Equations: A VAR model with lag order p and k variables is expressed as:

$$X_t = Z_t + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots + \phi_p X_{t-p}$$

- X_t: The vector of endogenous variables at time t.
- Z_t: A white noise vector with zero mean and covariance matrix Σ.
- ϕ_i: The coefficient matrix corresponding to lag i.

The lag order p is selected based on AIC and BIC criteria, where the optimal p is the value that minimizes both indicators. The two information criteria are defined as follows:

- AIC:

$$\text{AIC} = -2\ln(\hat{L}) + 2k$$

  where:
  - $\hat{L}$ is the likelihood evaluated at its estimated parameters.
  - k is the total number of estimated parameters in the model.
- BIC:

$$\text{BIC} = -2\ln(\hat{L}) + k\ln(n)$$

  where:
  - $\hat{L}$ is the likelihood evaluated at estimation.
  - k is the number of model parameters.
  - n is the number of observations in the sample.

c. Model Estimation: The coefficients of the VAR system are estimated using the Ordinary Least Squares (OLS) method.

d. Model Diagnostics:

- Stationarity Test: The Augmented Dickey–Fuller (ADF) test is employed to ensure stationarity of the time series. The ADF specification is as follows:

$$\Delta X_t = \alpha + \beta t + \gamma X_{t-1} + \sum_{i=1}^{p} \delta_i \Delta X_{t-i} + Z_t$$

  where:
  - ΔY_t denotes the first difference of Y_t.
  - α is the constant term.
  - βt represents the deterministic trend.
  - γ is the coefficient of Y_{t-1}, and is the key determinant of ADF rejection.
  - δ_i are the lagged difference coefficients.
  - Z_t denotes white noise.
  - Null hypothesis: The series Y_t contains a unit root (non-stationary).
- Autocorrelation Test: The Durbin–Watson statistic is used to verify that residuals do not exhibit autocorrelation. The test statistic is formulated as:

$$\mathrm{DW} = \frac{\sum_{t=2}^{n} (e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

  where:
  - e_t is the residual at time t.
  - n is the number of observations.
- Granger Causality Test: Determines whether one variable helps predict another — an essential check for VAR applicability. The test statistic is given by:

$$F = \frac{(RSS_r - RSS_{ur})/m}{RSS_{ur}/(n-k)}$$

  where:
  - RSS_r denotes the restricted residual sum of squares.
  - RSS_{ur} denotes the unrestricted residual sum of squares.
  - m is the number of restrictions.
  - n is the sample size.
  - k is the number of estimated parameters in the unrestricted model.

1.2.3 Advantages and Disadvantages of VAR

- Advantages:
  - The model is simple but can still provide highly accurate forecasts, making it widely used in economic and financial forecasting.
  - Does not require a specific theoretical model.
- Disadvantages:
  - Requires a large amount of historical data to accurately estimate the parameters.
  - The time series must be stationary.
  - Sensitive to the choice of lag length, variables, and estimation methods.

1.3 LSTM Model

1.3.1 Introduction

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed to process and predict sequential data, particularly effective in addressing the vanishing gradient problem commonly found in traditional RNNs. LSTM was proposed by Hochreiter and Schmidhuber in 1997 and has since become one of the most widely used models in time series processing and forecasting.

1.3.2 Architecture

LSTM consists of memory cells capable of retaining information over long periods, with its architecture illustrated in Figure 1. Each memory cell contains three main gates: the forget gate, the input gate, and the output gate. These gates regulate the flow of information through the memory cell [1].

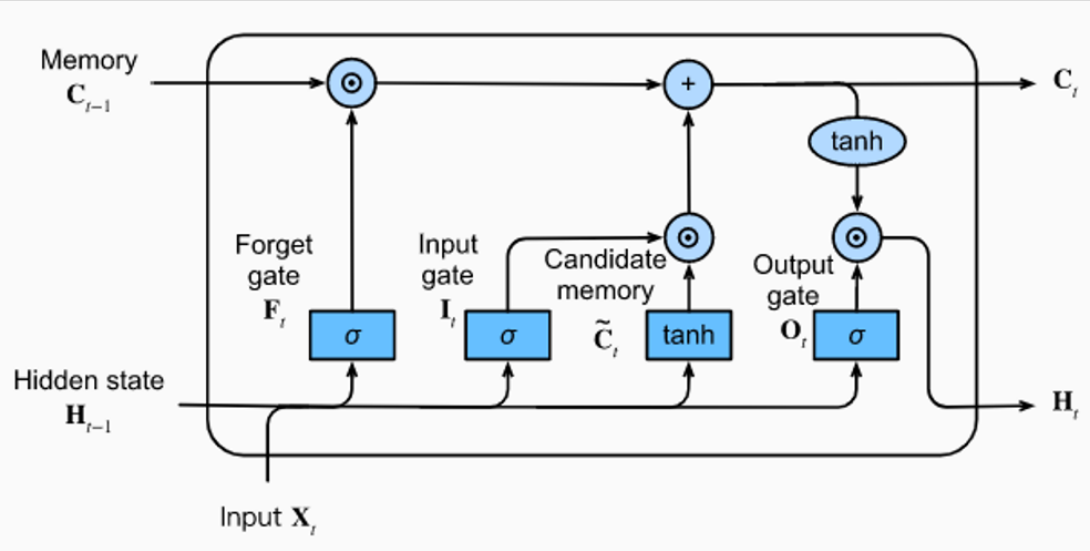

Figure 1: The architecture of LSTM.

a. Forget Gate

The forget gate determines which information from the previous state should be discarded. The calculation is given by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

where:

- σ is the sigmoid activation function, producing values in the range (0, 1).
- W_f is the weight matrix of the forget gate.
- h_{t-1} is the output from the previous time step (t-1).
- x_t is the current input (value at time t in the time series).
- b_f is the bias term of the forget gate.

- $[h_{t-1}, x_t]$ denotes the concatenation of the two vectors into a single vector. This concatenation allows the LSTM to utilize both past information (t-1) and current input information (t).

b. Input Gate

The input gate determines which information from the current input will be stored in the cell state. The computations are as follows:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

where:

- i_t is the output of the input gate.
- W_i and W_C are the weight matrices of the input gate and cell update.
- $\tilde{C}_t$ represents the candidate cell state.
- b_i and b_C are the bias terms for the input gate and cell update.

After computing the values from the forget and input gates, the cell state is updated as:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

where:

- C_{t-1} is the previous cell state.
- C_t is the updated (current) cell state.

c. Output Gate

The output gate determines which information from the current cell state will be used to produce the final output:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

where:

- W_o is the weight matrix of the output gate.
- b_o is the bias term of the output gate.
- h_t is the output of the LSTM cell at time step t.

1.3.3 Advantages and Disadvantages of LSTM

- Advantages:
  - Ability to retain information over long periods.
  - Effectively addresses the vanishing gradient problem.
  - Widely applied in fields such as time series forecasting, natural language processing, and speech recognition.
- Disadvantages:
  - More computationally complex compared to traditional RNNs.
  - Requires a large and diverse amount of data to perform efficiently.

1.4 BiLSTM Model

1.4.1 Introduction

BiLSTM, short for Bidirectional Long Short-Term Memory, is a variant of the traditional LSTM architecture. BiLSTM employs two LSTM layers operating in opposite directions when processing sequential data: one layer processes the sequence from left to right (forward direction), while the other processes it from right to left (backward direction). This structure enables the BiLSTM model to capture contextual information from both past and future time steps.

1.4.2 Architecture

A BiLSTM combines the outputs of both the forward LSTM and the backward LSTM [3]. The architecture consists of two separate LSTM layers, and is illustrated in Figure 2.

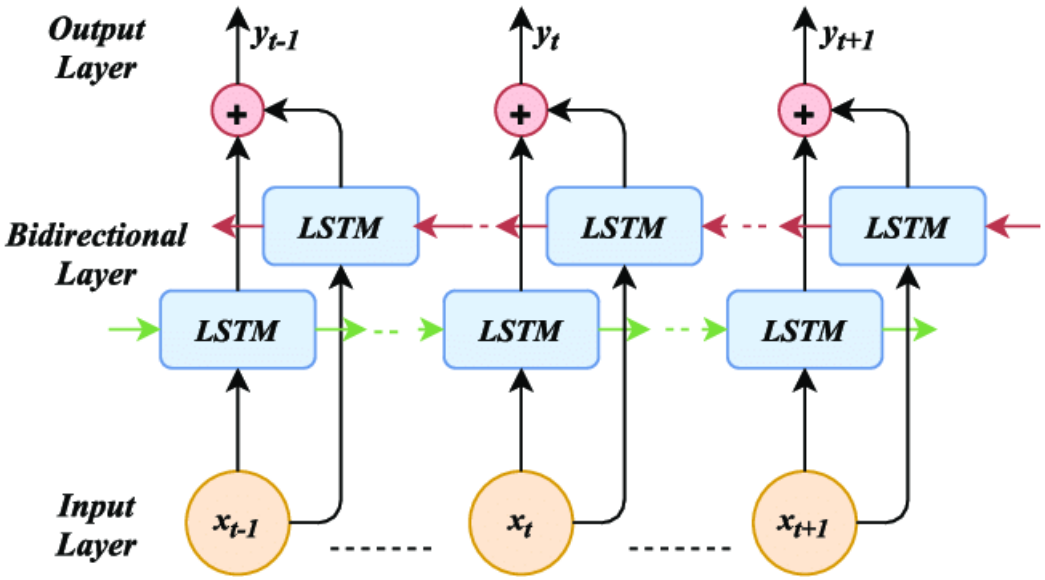

Figure 2: The architecture of BiLSTM

a. Forward LSTM

The forward LSTM processes the time sequence from the first time step to the final one. At each time step t, the forward LSTM receives the input x_t and the hidden state from the previous time step $\overrightarrow{h_{t-1}}$, then computes the current hidden state $\overrightarrow{h_t}$. The computation steps are as follows:

- Receive the input at time step t: x_t.
- Compute the forget gate:

$$f_t^{\text{fwd}} = \sigma(W_f^{\text{fwd}} \cdot [\overrightarrow{h_{t-1}}, x_t] + b_f^{\text{fwd}})$$

- Compute the input gate:

$$i_t^{\text{fwd}} = \sigma(W_i^{\text{fwd}} \cdot [\overrightarrow{h_{t-1}}, x_t] + b_i^{\text{fwd}})$$

$$\tilde{C}_t^{\text{fwd}} = \tanh(W_C^{\text{fwd}} \cdot [\overrightarrow{h_{t-1}}, x_t] + b_C^{\text{fwd}})$$

- Update the cell state:

$$C_t^{\text{fwd}} = f_t^{\text{fwd}} * C_{t-1}^{\text{fwd}} + i_t^{\text{fwd}} * \tilde{C}_t^{\text{fwd}}$$

- Compute the output gate:

$$o_t^{\text{fwd}} = \sigma(W_o^{\text{fwd}} \cdot [\overrightarrow{h_{t-1}}, x_t] + b_o^{\text{fwd}})$$
$$\overrightarrow{h_t} = o_t^{\text{fwd}} * \tanh(C_t^{\text{fwd}})$$

b. Backward LSTM

The backward LSTM processes the time sequence in the reverse order, from the final time step to the first. At each time step t, the backward LSTM receives the input x_t and the hidden state from the subsequent time step $\overleftarrow{h_{t+1}}$, then computes the current hidden state $\overleftarrow{h_t}$.

- Receive the input at time step t: x_t.
- Compute the forget gate:

$$f_t^{(\text{bwd})} = \sigma(W_f^{(\text{bwd})} \cdot [\overleftarrow{h_{t+1}}, x_t] + b_f^{(\text{bwd})})$$

- Compute the input gate:

$$i_t^{(\text{bwd})} = \sigma(W_i^{(\text{bwd})} \cdot [\overleftarrow{h_{t+1}}, x_t] + b_i^{(\text{bwd})})$$
$$\tilde{C}_t^{(\text{bwd})} = \tanh(W_C^{(\text{bwd})} \cdot [\overleftarrow{h_{t+1}}, x_t] + b_C^{(\text{bwd})})$$

- Update the cell state:

$$C_t^{(\text{bwd})} = f_t^{(\text{bwd})} * C_{t+1}^{(\text{bwd})} + i_t^{(\text{bwd})} * \tilde{C}_t^{(\text{bwd})}$$

- Compute the output gate:

$$o_t^{(\text{bwd})} = \sigma(W_o^{(\text{bwd})} \cdot [\overleftarrow{h_{t+1}}, x_t] + b_o^{(\text{bwd})})$$
$$\overleftarrow{h_t} = o_t^{(\text{bwd})} * \tanh(C_t^{(\text{bwd})})$$

c. Output Combination

The outputs from the forward and backward LSTM are typically combined by concatenating the output vectors at each time step. This yields an output vector h_t whose dimension is doubled compared to that of a single LSTM:

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$$

1.4.3 Advantages and Disadvantages of BiLSTM

- Advantages:
  - Captures a more comprehensive contextual representation.
  - Improves performance in various applications such as time series forecasting and natural language processing.
- Disadvantages:
  - More computationally complex compared to a standard LSTM.
  - Requires a larger and more diverse dataset for the BiLSTM to operate effectively.

1.5 Transformer Model

1.5.1 Introduction

Transformer is a powerful deep learning model originally developed for natural language processing tasks, but it has since been extended and successfully applied to various other domains, including time series forecasting and stock price prediction.

1.5.2 Architecture

The Transformer model is designed to process sequential data using encoder layers that learn representations from the input. Each encoder layer consists of key components such as Multi-Head Attention, a Feedforward Neural Network, and Layer Normalization [6]. The architecture of the Transformer Encoder is illustrated in Figure 3.

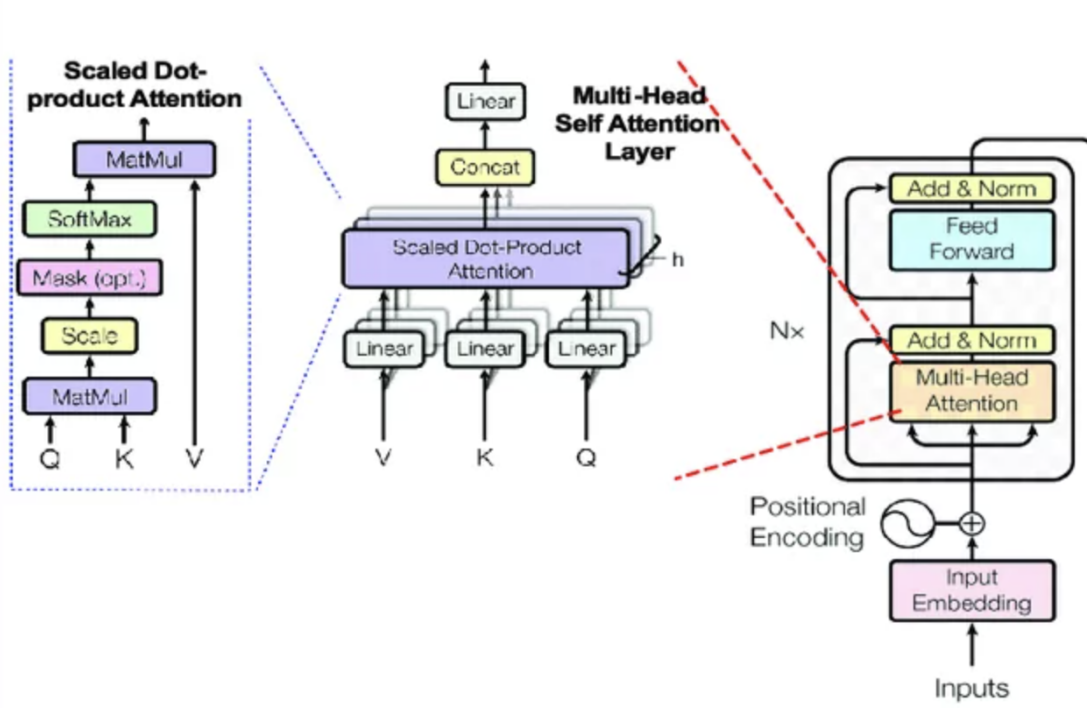

Figure 3: The architecture of Transformer Encoder.

a. Positional Encoding

Positional Encoding is used to capture information about the order of elements within a sequence. It is defined by the following formulas:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

where:

- pos is the position of the element in the sequence.
- i is the dimension index of the positional vector.
- d_{model} is the dimensionality of the model's hidden representations.

The positional encodings are added to the input vectors to enable the model to incorporate positional information.

b. Multi-Head Attention

Multi-Head Attention allows the Transformer to simultaneously focus on different parts of the input sequence, enabling the model to learn complex relationships in the data. Attention is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where:

- Q (Query): Query matrix.
- K (Key): Key matrix.
- V (Value): Value matrix.
- d_k: Dimensionality of the query/key vectors.

Instead of using a single attention mechanism, Multi-Head Attention employs h parallel attention heads to capture different types of relationships within the sequence:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_O$$

where:

- $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
- W_i^Q, W_i^K, W_i^V: Learnable weight matrices for the queries, keys, and values in each attention head.
- W_O: Output projection matrix for combining the attention heads.

c. Feedforward Neural Network (FFN)

After the output from Multi-Head Attention is computed, it is passed through a feedforward neural network to learn nonlinear transformations. This network consists of two dense layers with a nonlinear activation function (ReLU) in between, expressed as:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where:

- W_1 and W_2 are weight matrices.
- b_1 and b_2 are bias terms.
- $\max(0, x)$ is the ReLU activation function.

d. Layer Normalization

Layer Normalization is applied after both the Multi-Head Attention and the FFN components to stabilize training and improve model performance. Given an input $x = [x_1, x_2, \ldots, x_n]$, Layer Normalization is computed as:

$$\text{LayerNorm}(x) = \gamma \frac{x - \mu}{\sigma + \epsilon} + \beta$$

where:

- γ and β are learnable parameters with the same dimensionality as x, where γ is a scaling factor and β is a shifting factor.
- μ is the mean, computed as:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- σ is the standard deviation, computed as:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}$$

- ϵ is a small constant added to avoid division by zero.

d. Residual Connections

To improve performance and facilitate learning, the Transformer employs Residual Connections, which allow the output of a layer to be added directly to its input.

- The output of Multi-Head Attention is added to its input, followed by Layer Normalization.
- The output of the FFN is added to its input, followed by Layer Normalization.

1.5.3 Advantages and Disadvantages of Transformer

- Advantages:
  - Ability to learn long-range dependencies through the Self-Attention mechanism.
  - Transformers do not require data to be processed sequentially as in LSTMs, allowing the model to perform computations in parallel.
- Disadvantages:
  - High computational resource requirements.
  - Increased implementation complexity.

2 Technological Stock Price Forecasting Problem

2.0.1 Business Problem

Problem Description: This problem aims to forecast the stock prices of major technology companies in the stock market. Accurate stock price prediction enables investors to make informed decisions regarding buying, selling, or holding stocks, thereby optimizing profits and minimizing risks.

Business Requirements:

- Data Collection:
  - Collect historical stock price data for selected technology companies over a specified period (3 years).
  - Include information such as opening price (Open), highest price (High), lowest price (Low), and closing price (Close).
- Data Analysis:
  - Use statistical methods to analyze and understand the data.
  - Identify factors that influence stock prices, such as market news, company financial indicators, and economic events.
- Stock Price Forecasting:
  - Apply statistical and deep learning models to forecast future stock prices.
- Evaluation:
  - Use evaluation metrics such as RMSE (Root Mean Squared Error) and MAPE (Mean Absolute Percentage Error) to assess the accuracy of the model.

2.0.2 Technical Problem

Problem Description:

Develop and implement statistical and deep learning models to forecast stock prices in the technology sector. This problem is addressed through data processing techniques, model design, model training, and model evaluation.

Technical Requirements:

- Data Preparation:
  - Collect stock price data from the NASDAQ-100 Technology Sector.
  - Perform data preprocessing, including handling missing values, normalizing the data, and generating technical indicators such as Moving Average, RSI (Relative Strength Index), and MACD (Moving Average Convergence Divergence).
- Model Design and Construction:

- Use libraries such as TensorFlow, Keras, or PyTorch to build deep learning models.

- Model Training:
  - Split the data into training, validation, and test sets.
  - Use optimization techniques such as the Adam optimizer to train the models.

- Model Evaluation and Fine-Tuning:
  - Evaluate the models on the test set using performance metrics such as RMSE and MAPE.
  - Fine-tune the model hyperparameters to improve accuracy.

- Model Deployment:
  - Package the trained models.
  - Build a user interface (UI) to allow investors to interact with the system and view prediction results.

2.1 Proposed Models for Stock Price Prediction

2.1.1 VAR Model

After the dataset is prepared, the VAR model is executed through the following stages:

1. The dataset is first examined using the Granger Causality test. Variables that do not satisfy causality relevance for forecasting are removed.

2. The stationarity of the series is evaluated using the Augmented Dickey-Fuller (ADF) test. Non-stationary variables are differenced and/or log-transformed where necessary.

3. If only some of the variables require differencing, linear backward forecasting is applied to maintain data continuity for those series.

4. The processed dataset is then tested again using the Granger Causality approach.

5. The VAR model is trained and the optimal lag order is selected.

6. Residual autocorrelation is evaluated using the Durbin–Watson test. If residual correlation exists, the lag order is increased accordingly.

7. The closing price for future periods is predicted based on previous values corresponding to the selected lag length.

A detailed schematic representation of the model is illustrated as follows:
In this stage, after determining the optimal lag order by selecting the model with the lowest AIC and BIC values, the main dataset is fitted using VAR(6), the technical dataset is modeled using VAR(2), and the economic dataset is specified as VAR(1).

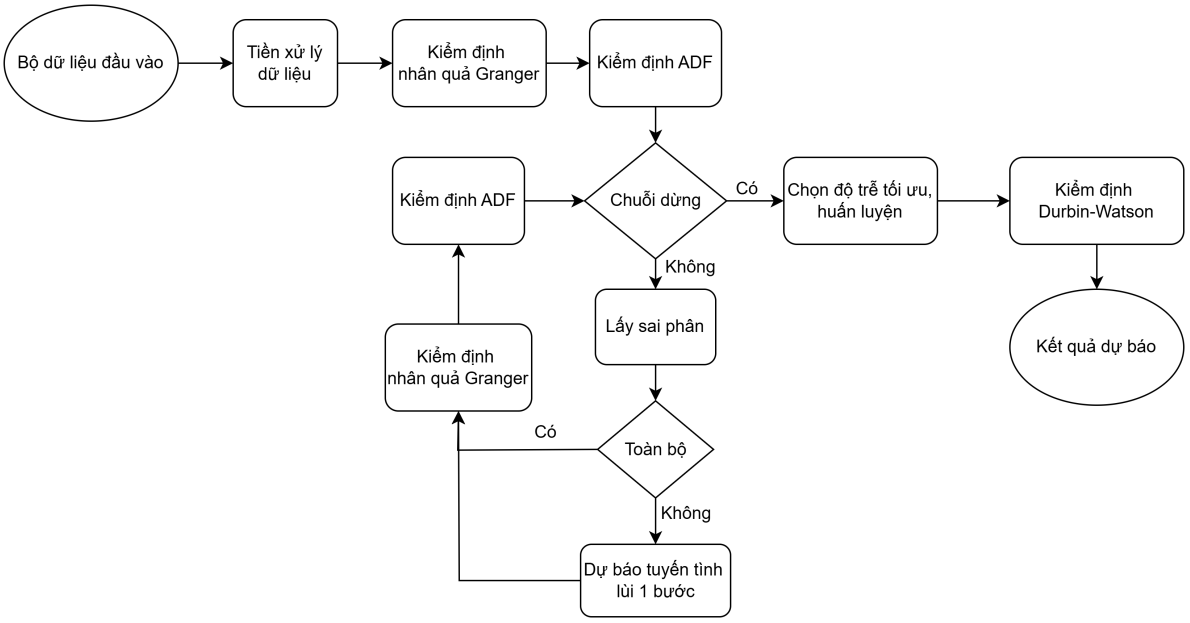

Figure 4: VAR Diagram

2.1.2 LSTM Model

With a 30-day stock time series corresponding to the values $x_1, x_2, \ldots, x_{30}$. The operating process of LSTM proceeds as follows:

1. Day 1:
   - x_1 is fed into the LSTM.
   - The forget gate f_1, input gate i_1, temporary cell state $\tilde{C}_1$, cell state C_1, and output gate o_1 are computed.
   - The output h_1 is generated from o_1 and C_1.
2. Day 2 to Day 29:
   - The same procedure as Day 1, with input ranging from x_2 to x_{29}.
   - The values $f_t, i_t, \tilde{C}_t, C_t, o_t$ and h_t are continuously updated for each day.
3. Day 30:
   - Input x_{30} is fed into the LSTM.
   - The forget gate f_{30}, input gate i_{30}, temporary cell state $\tilde{C}_{30}$, cell state C_{30}, and output gate o_{30} are computed.
   - The output h_{30} is generated from o_{30} and C_{30}.
4. Predicting Day 31:
   - The output of the LSTM at Day 30 (h_{30}) is fed into the Dense layer to predict the value for Day 31. The Dense layer performs a linear transformation to produce the final predicted value.

The diagram and detailed information of the LSTM models used in this report are as follows:

1. One-layer LSTM model:

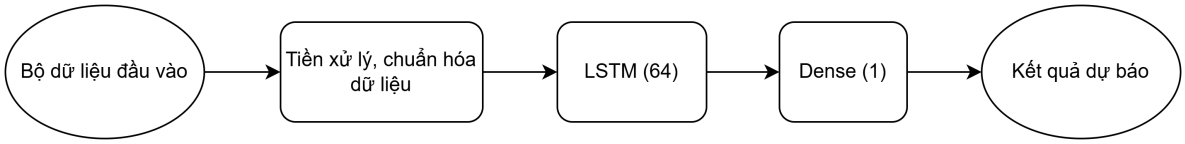

Figure 5: Diagram of the 1-layer LSTM model

The model is trained with a batch size of 32 over 20 epochs. The model weights are optimized during training using Adaptive Moment Estimation, and the loss function applied is Mean Squared Error (MSE).

2. Two-layer LSTM model:

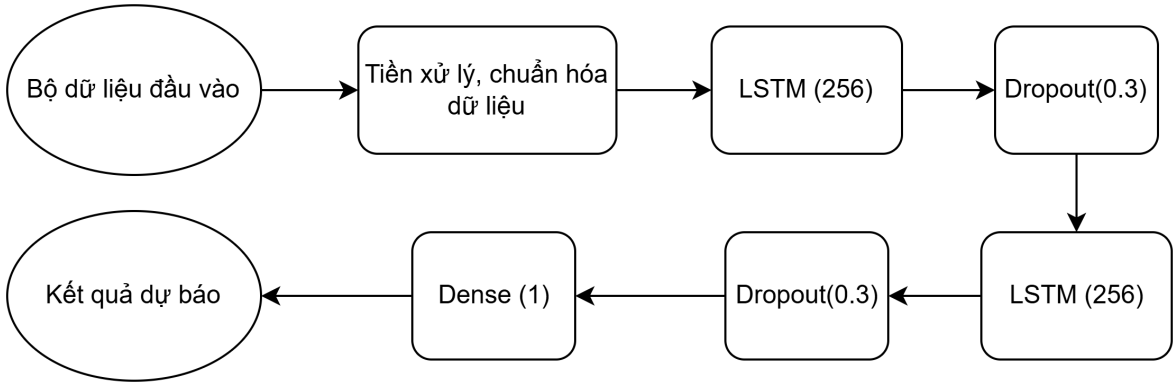

Figure 6: Diagram of the 2-layer LSTM model

This model is taken from the paper *Robust Portfolio Design and Stock Price Prediction Using an Optimized LSTM Model* published by Sen et al. in 2021 [4]. It is trained with a batch size of 64 over 100 epochs. The model weights are optimized using Adaptive Moment Estimation, with the loss function being Huber for the training set and Mean Absolute Error for the validation set.
Additionally, this report provides another model based on the same two-layer LSTM architecture but trained with a batch size of 128 over 100 epochs. The model weights are also optimized using Adaptive Moment Estimation, with Mean Squared Error (MSE) as the loss function. This alternative model demonstrates an improved performance and offers a better predictive result.

2.1.3 BiLSTM Model

With a 30-day stock time series corresponding to the values $x_1, x_2, \ldots, x_{30}$. The operational process of BiLSTM proceeds as follows:

1. Day 1:
    - Input x_1 is fed into both the forward LSTM and backward LSTM.
    - The forward LSTM processes x_1 and updates the hidden state $\overrightarrow{h_1}$.
    - The backward LSTM processes from the end of the sequence (Day 30) backward, but at this stage, only x_1 is processed.
2. Day 2 to Day 29:
    - Similar to Day 1, inputs x_t from Day 2 to Day 29 are fed into both forward and backward LSTM.
    - The forward LSTM continuously processes the sequence from x_2 to x_{29} and updates the hidden state $\overrightarrow{h_t}$ at each step.
    - The backward LSTM also begins processing from Day 29 backward and updates the hidden state $\overleftarrow{h_t}$.
3. Day 30:
    - Input x_{30} is fed into both forward and backward LSTM.
    - The forward LSTM completes processing from Day 1 to Day 30, and the backward LSTM completes processing from Day 30 to Day 1.
    - The final output of Day 30 is the concatenation of the hidden states from both forward and backward LSTM:

$$h_{30} = [\overrightarrow{h_{30}}, \overleftarrow{h_{30}}]$$

4. Forecasting Day 31:
    - Pass through Dropout:
        - The output h_{30} is passed through a Dropout layer with a rate of 0.1 to prevent overfitting.
    - Pass through ReLU layer:
        - The output from Dropout is passed through the ReLU activation function to introduce non-linearity.
    - Pass through Dense layer:
        - It is then passed through a Dense layer with 16 units and ReLU activation.
    - Final prediction:
        - Finally, it is passed through a Dense layer with 1 unit to generate the stock price prediction for Day 31.

BiLSTM Model:

The model is trained with a batch size of 32 over 20 epochs. The model weights are optimized using Adaptive Moment Estimation during training, with Mean Squared Error (MSE) used as the loss function.

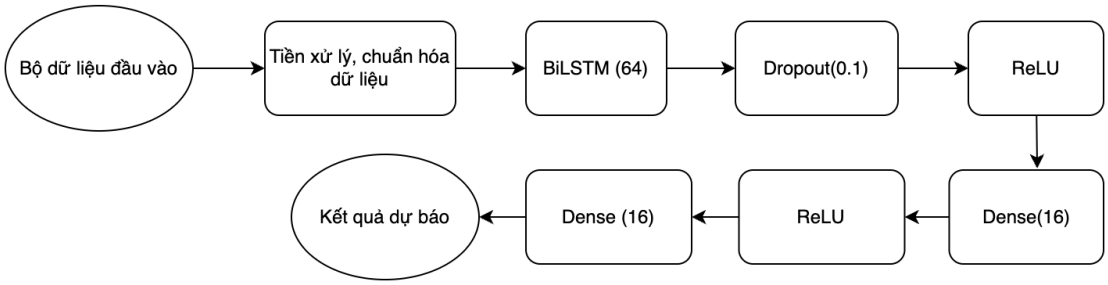

Figure 7: Diagram for the BiLSTM model

2.1.4 Transformer Model

With a 30-day stock price time series dataset corresponding to the values $x_1, x_2, \ldots, x_{30}$. In the Transformer model, three encoder layers are stacked, where the output of each encoder layer becomes the input to the next one. The operational flow of an encoder layer is described as follows:

1. Embedding và Positional Encoding

   - Embedding: Each input vector x_i is transformed into a hidden vector with dimension $d_{model} = 64$ through a Linear layer.

   $$\text{Encoded Input} = \text{Linear}(x_i) \cdot d_{model}$$

   - Positional Encoding: Positional information is added to indicate the position of each element in the sequence. Positional Encoding uses sine and cosine functions with varying frequencies to encode time-step information.

   $$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

   $$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

   - Then, Positional Encoding is added to the hidden vectors:

   $$\text{Encoded Input} += \text{Positional Encoding}(pos)$$

2. Multi-Head Attention

   - The input is split into 8 attention heads, each with dimension $d_k = \frac{d_{model}}{nhead} = \frac{64}{8} = 8$.
   - Attention is computed for each head:

   $$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

   where: $Q = K = V = \text{Linear}(\text{Encoded Input})$

- Outputs from all 8 heads are concatenated and projected through a Linear layer:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_8)W_O$$

3. Residual Connection và Layer Normalization

- The output of Multi-Head Attention is added to the original input.

$$\text{Output}_1 = x + \text{MultiHead}(Q, K, V)$$

- Layer Normalization is then applied to stabilize the output.

$$\text{Output}_1 = \text{LayerNorm}(\text{Output}_1)$$

4. Feedforward Neural Network (FFN)

- The output from the previous step is passed through a feedforward neural network with two fully connected layers and a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

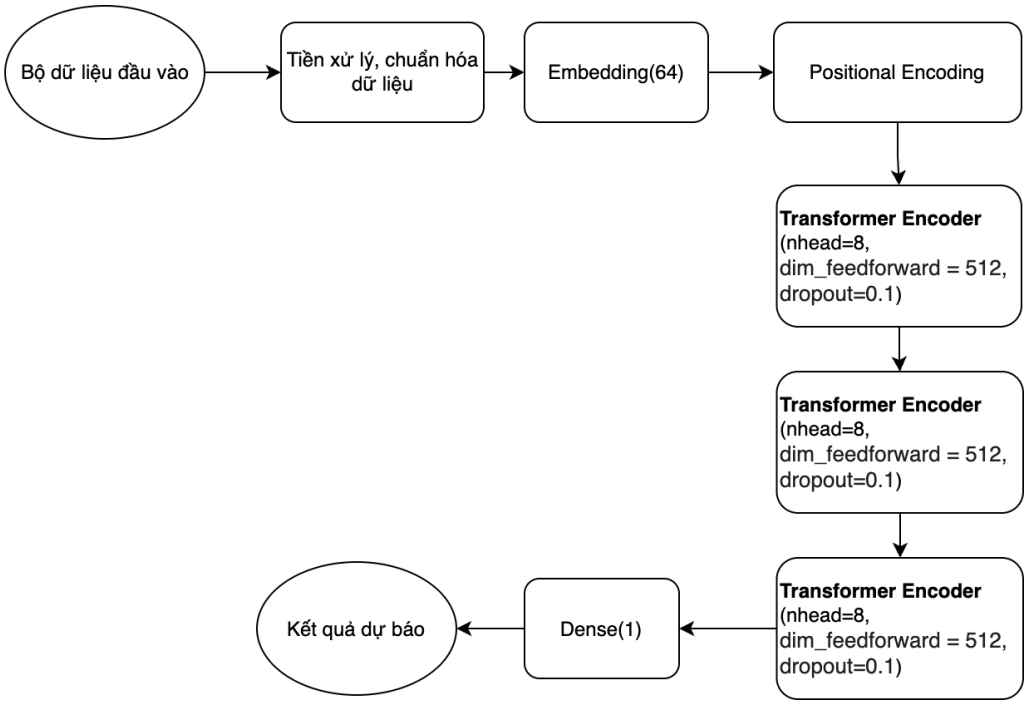

Figure 8: Diagram of the Transformer model

5. Residual Connection và Layer Normalization

- The output of the FFN is added to the input of the FFN:

$$\text{Output}_2 = \text{Output}_1 + \text{FFN}(\text{Output}_1)$$

- Layer Normalization is applied once again:

$$\text{Output}_2 = \text{LayerNorm}(\text{Output}_2)$$

Three encoder layers are then stacked, where the output of each layer becomes the input to the next. After passing through all encoder layers, the final output corresponding to x_{30} is fed into a fully connected layer to produce the final prediction for day 31 stock price:

$$\text{Output} = \text{Linear}(\text{Output}_{\text{final}(x_{30})})$$

The model is trained with a batch size of 32, 20 epochs, learning rate = 0.0001. Model weights are optimized using Adaptive Moment Estimation (Adam), and the loss function is Mean Squared Error (MSE).

3 Experiments and Results

3.1 Dataset

3.1.1 Dataset Overview

he report employs three different datasets for implementation across experimental models. The datasets are collected over the period from 03/06/2019 to 30/05/2024, consisting of **1258** observations. Based on these datasets, the report forecasts the **Close/Last** price — the daily closing price of a selected NASDAQ-100 technology stock. For Dataset 2, since the technical indicators are derived through separate computational formulas, the date column only represents the timeline corresponding to the **Close/Last** values. The details of the three datasets are as follows:

1. Primary Dataset: The dataset consists of 4 attributes: **Close/Last**, **Open**, **High**, **Low**. These attributes represent trading price information of a NASDAQ-100 technology stock. The dataset is constructed based on the market capitalization of the 100 largest non-financial companies listed on the Nasdaq Stock Exchange.

2. Technical Dataset [2]: This dataset includes 5 components: **Close/Last**, **SMA**, **MACD**, **Signal_Line**, **RSI**. These attributes are technical indicators derived from the **Close/Last** price. Their computation is described as follows:

   - SMA (Simple Moving Average): The arithmetic mean of closing prices over a predefined window. For example, a 20-day SMA is computed as the sum of the 20 most recent closing prices divided by 20.

$$\text{SMA}_n = \frac{\sum_{i=1}^{n} \text{Close/Last}_i}{n}$$

   - MACD (Moving Average Convergence Divergence): Computed as the difference between the 12-day EMA (Exponential Moving Average) and the 26-day EMA.

$$\text{MACD} = \text{EMA}_{12} - \text{EMA}_{26}$$

- Signal Line: Defined as the 9-day EMA of MACD and used for buy/sell signal identification.

$$\text{Signal Line} = \text{EMA}_9(\text{MACD})$$

- RSI (Relative Strength Index): A momentum oscillator that quantifies price movements based on average gains and losses within a specific window (commonly 14 days).

$$\text{RSI} = 100 - \frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}}$$

3. Economic Dataset: This dataset comprises 4 attributes: **Close/Last**, **COMP**, **NATS**, **MSCI**. These components represent the closing price of correlated technology indices and the **Close/Last** price. Descriptions of the attributes are given below:

   - COMP: Closing price of the NASDAQ Composite Index, which includes over 3,000 equities listed on NASDAQ across multiple sectors, the majority of which are technology-oriented. The NASDAQ Composite acts as a major benchmark indicating market conditions and sentiment, particularly in the technology sector.
   - NATS: Closing price of the S&P North American Technology Software Index, comprising North American firms (primarily U.S. and Canada) operating in the software industry. This index offers insights into leading technological firms and highlights the developmental trajectory of the regional software sector.
   - MSCI: Closing price of the MSCI USA Software&Services Index, representing U.S. firms specializing in software and technology services. The index is designed to measure the performance of mid- and large-cap corporations within the software and technology services industry.

3.1.2 Data Visualization and Exploration

To better support subsequent processing and forecasting stages, the dataset is visualized. This report provides the following visual results:

- Missing data statistics:

```
Date          0   Date          0   Date          0
Close/Last    0   Close/Last    0   Close/Last    0
Open          0   SMA           0   COMP          0
High          0   MACD          0   NATS          0
Low           0   Signal_Line   0   MSCI          0
                  RSI           0
```

- Close/Last data:

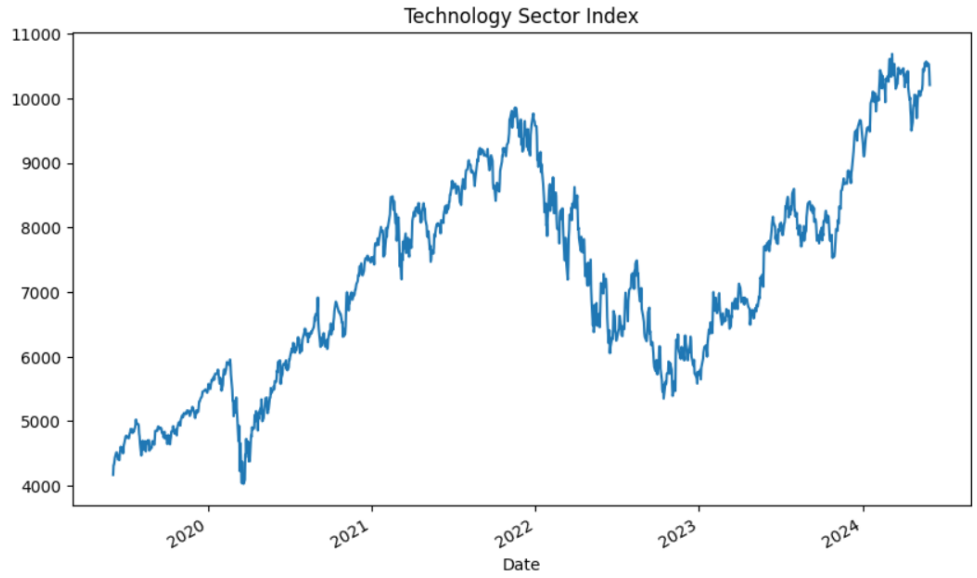

Figure 9: Close price chart

- Main dataset: Since the attributes Open, High, and Low have relatively similar values, they are visualized separately to improve readability.
  - Open:

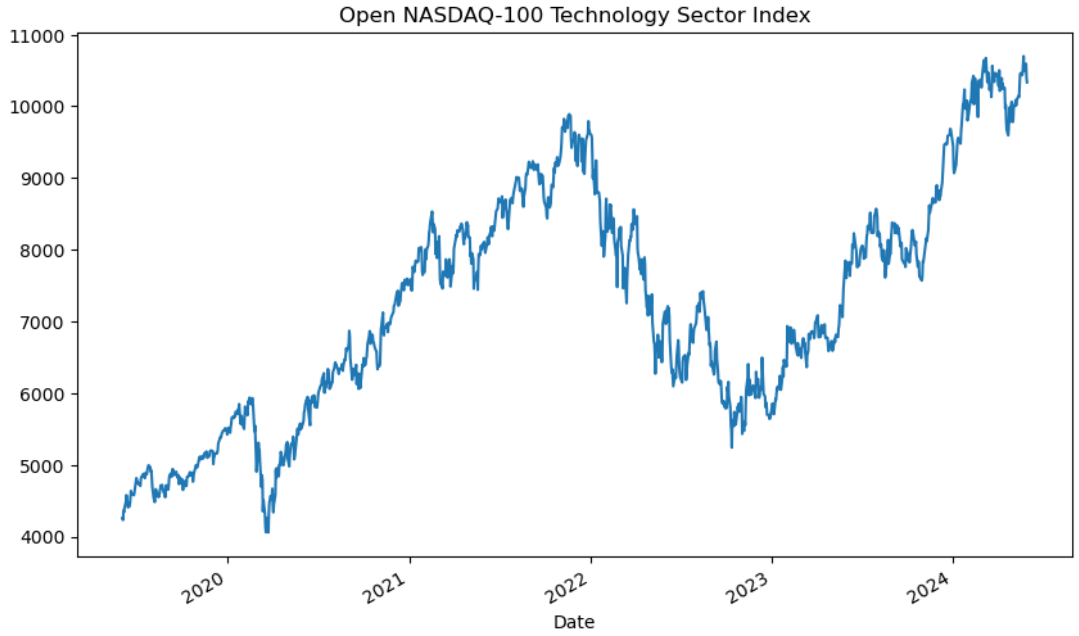

Figure 10: Open price chart

– High:

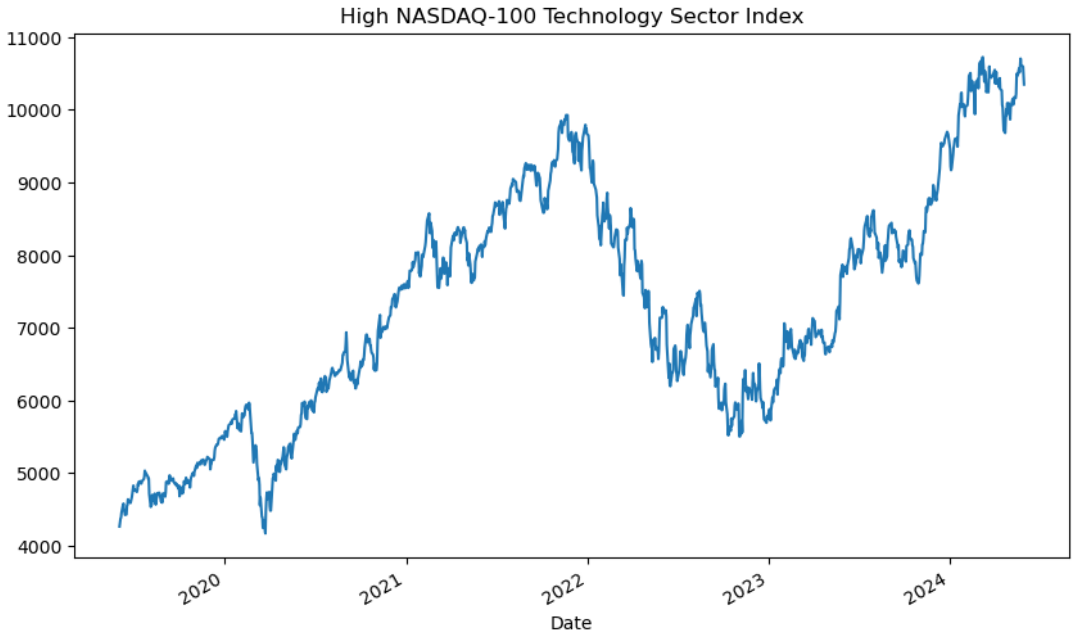

Figure 11: High price chart

– Low:

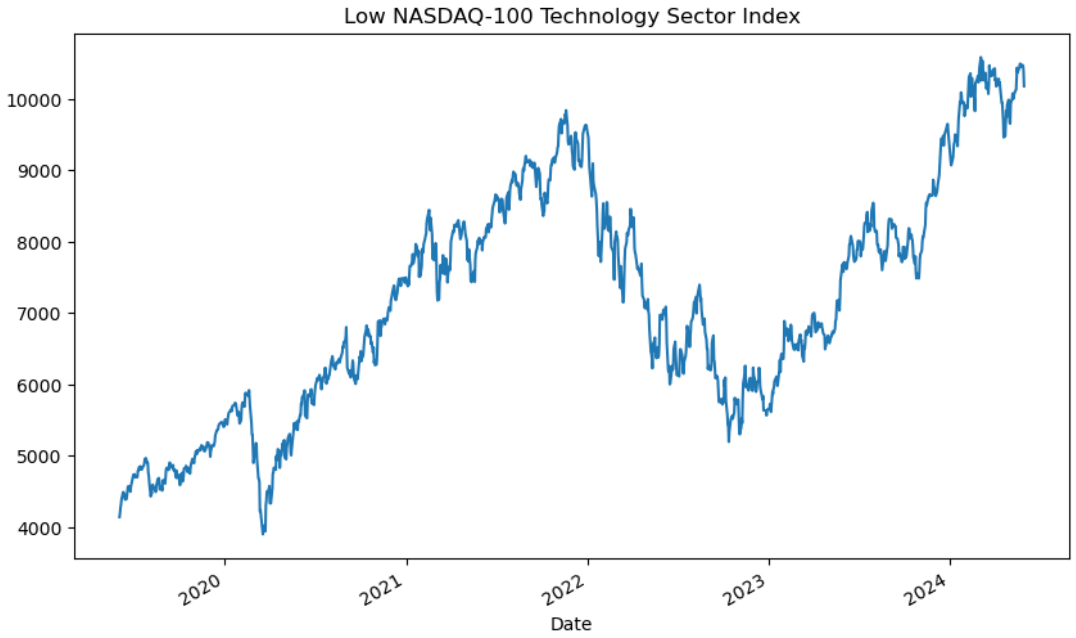

Figure 12: Low price chart

– Moving average and rolling standard deviation with a window size of 30 time units:

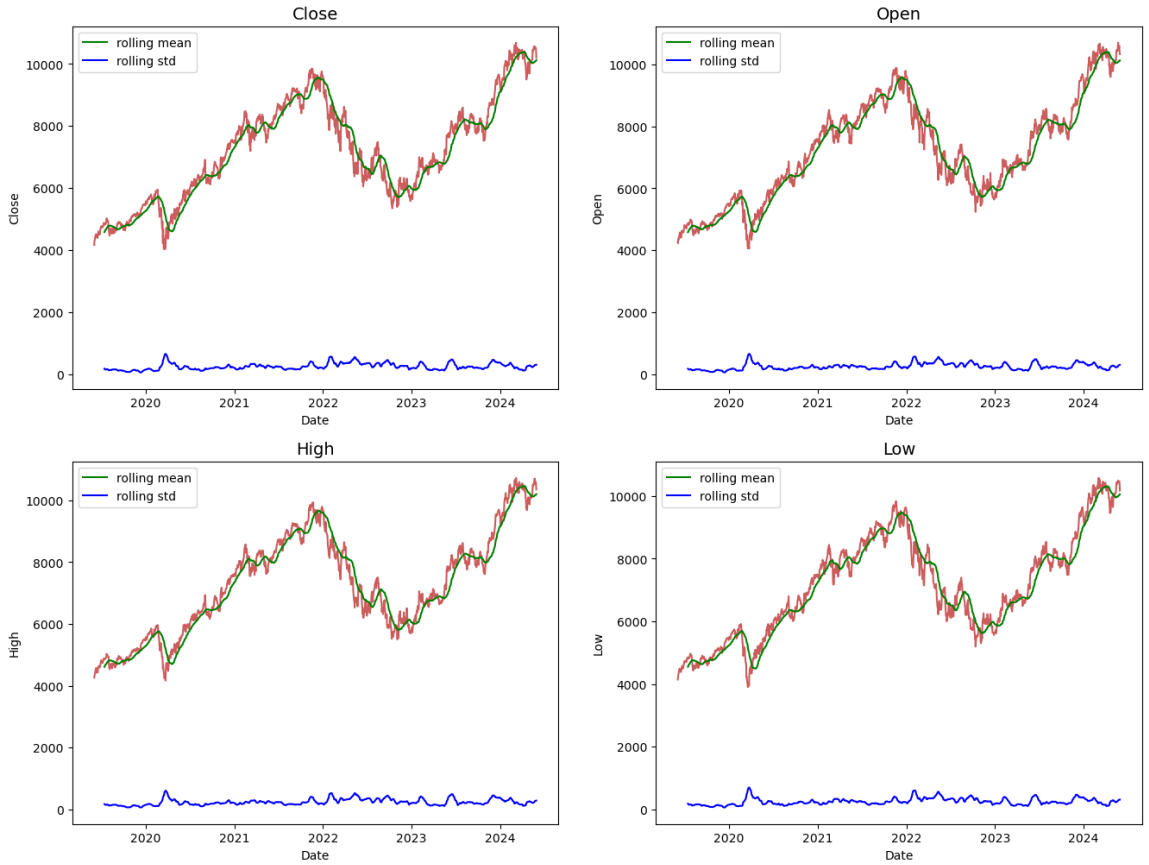

Figure 13: Rolling Mean and Rolling Std chart

Moving averages help identify long-term or short-term trends in the data. When the price consistently stays above the moving average, it indicates an upward trend. Conversely, when the price remains below the moving average, it signals a downward trend. Meanwhile, the rolling standard deviation reflects the degree of price fluctuation. A high rolling standard deviation typically indicates substantial volatility, which corresponds to higher risk, and vice versa.

– Trend and seasonality decomposition for features:

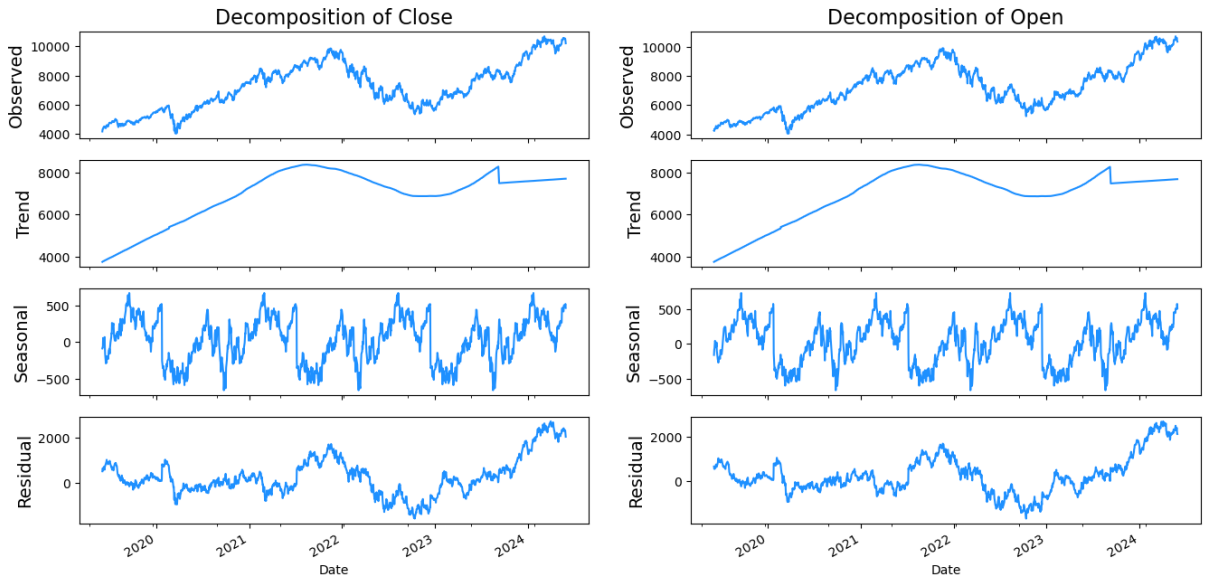

Figure 14: Decomposition plots for Close and Open

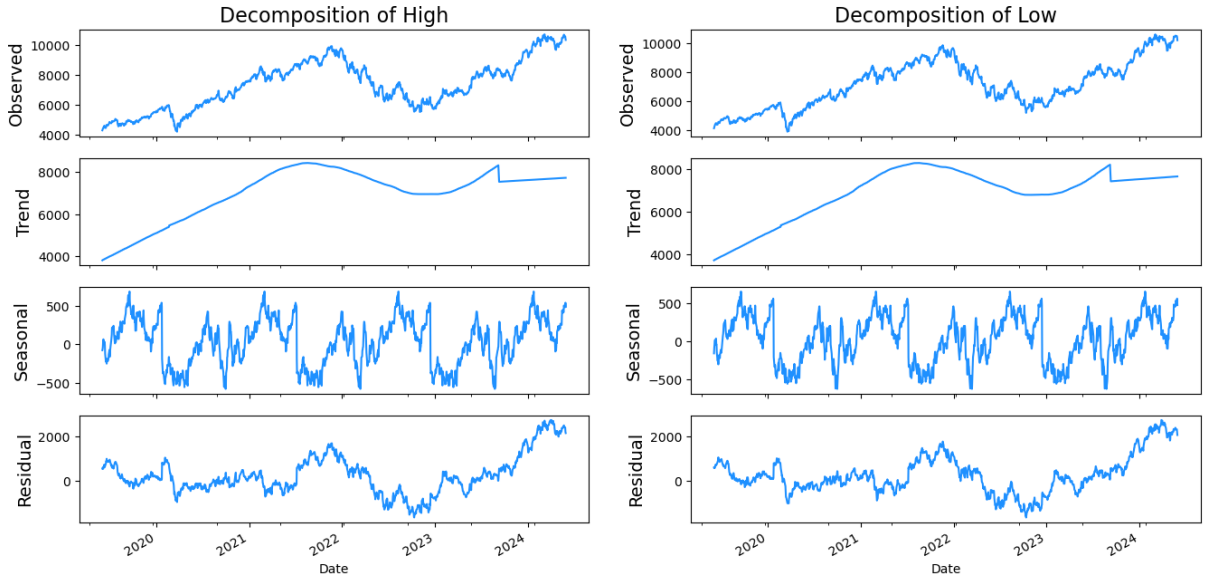

Figure 15: Decomposition plots for High and Low

– Correlation matrix of the dataset:

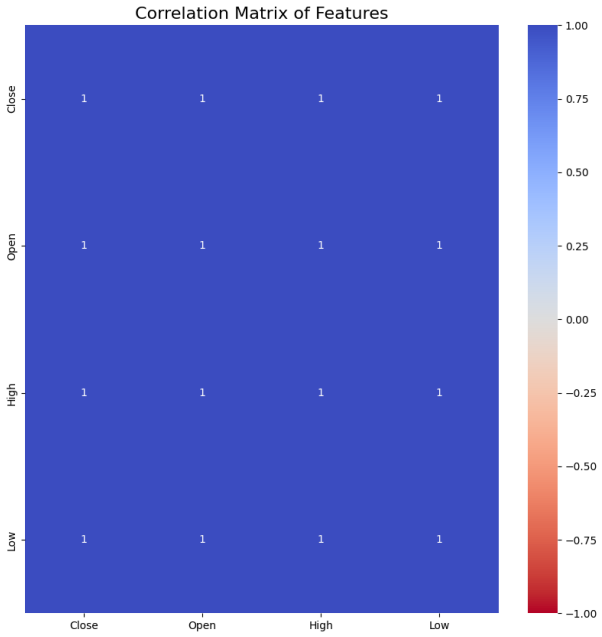

Figure 16: Correlation matrix

Since the data consist of opening, closing, high, and low prices of the same stock, there is no significant difference in their overall trends. Therefore, the examined attributes exhibit very high correlation with one another.

- Technical indicator dataset:
  - Overview of the dataset:

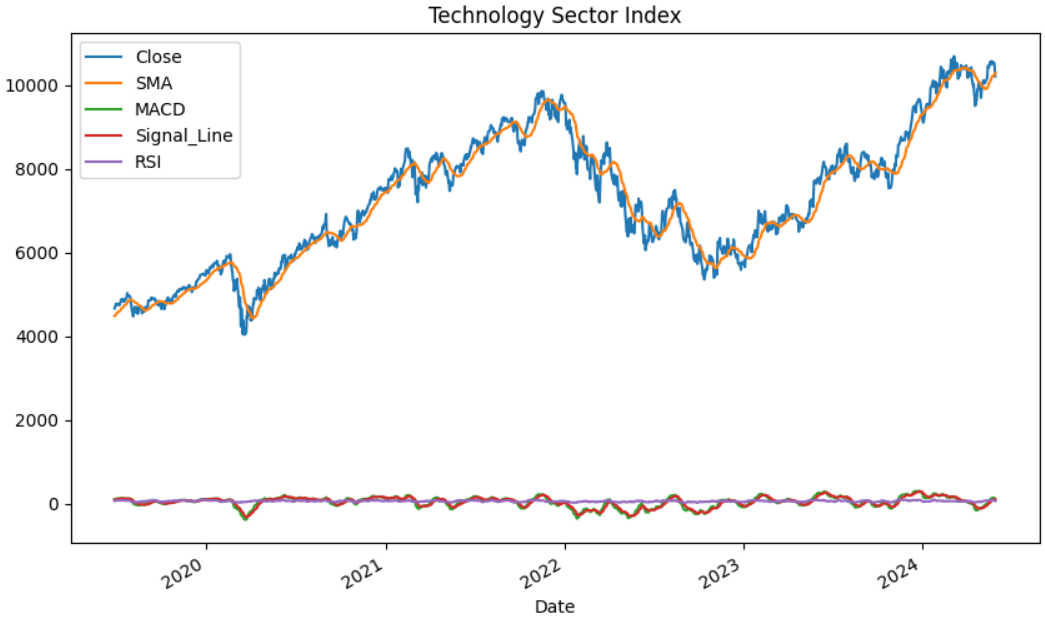

Figure 17: Plot of technical indicators

  - Rolling mean and rolling standard deviation with a window size of 30 time units. Here, the plot does not include Close since it has already been presented

in the previous dataset.

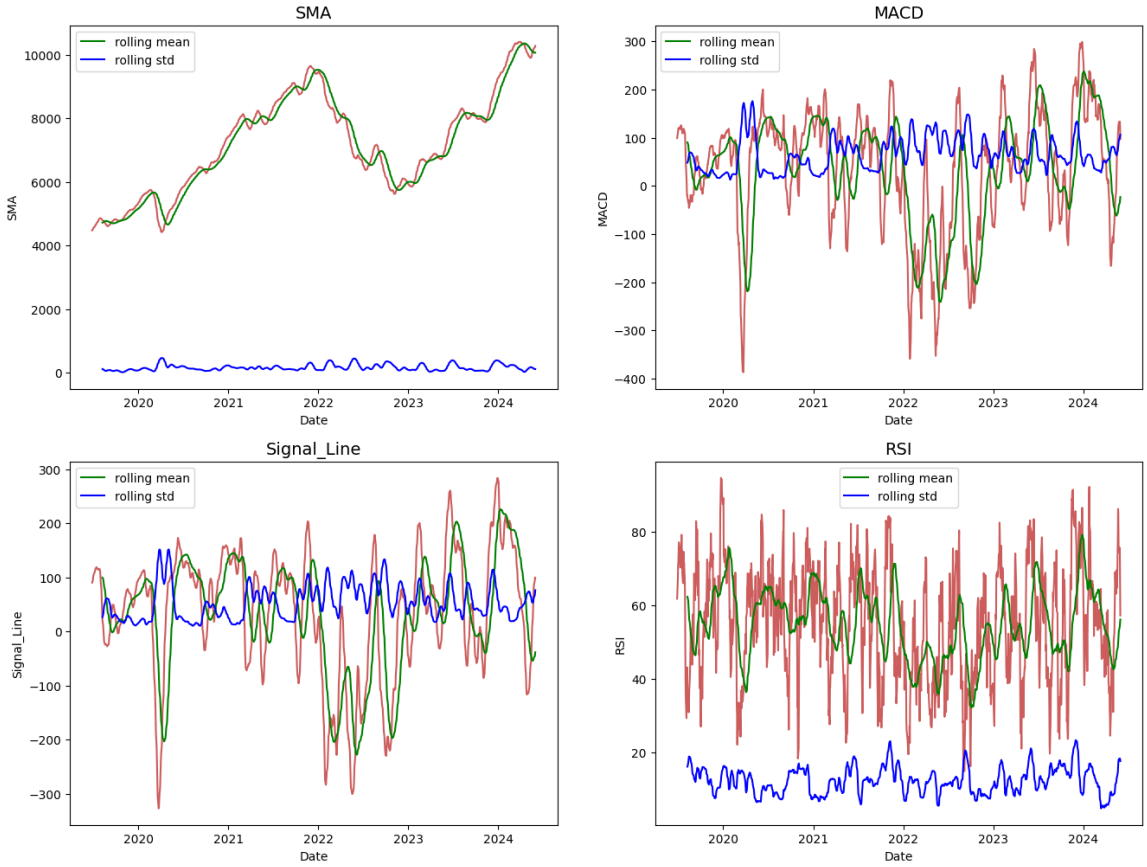

Figure 18: Rolling Mean and Rolling Std plots

– Trend and seasonality decomposition for attributes. Again, *Close* is excluded as it has been visualized earlier.

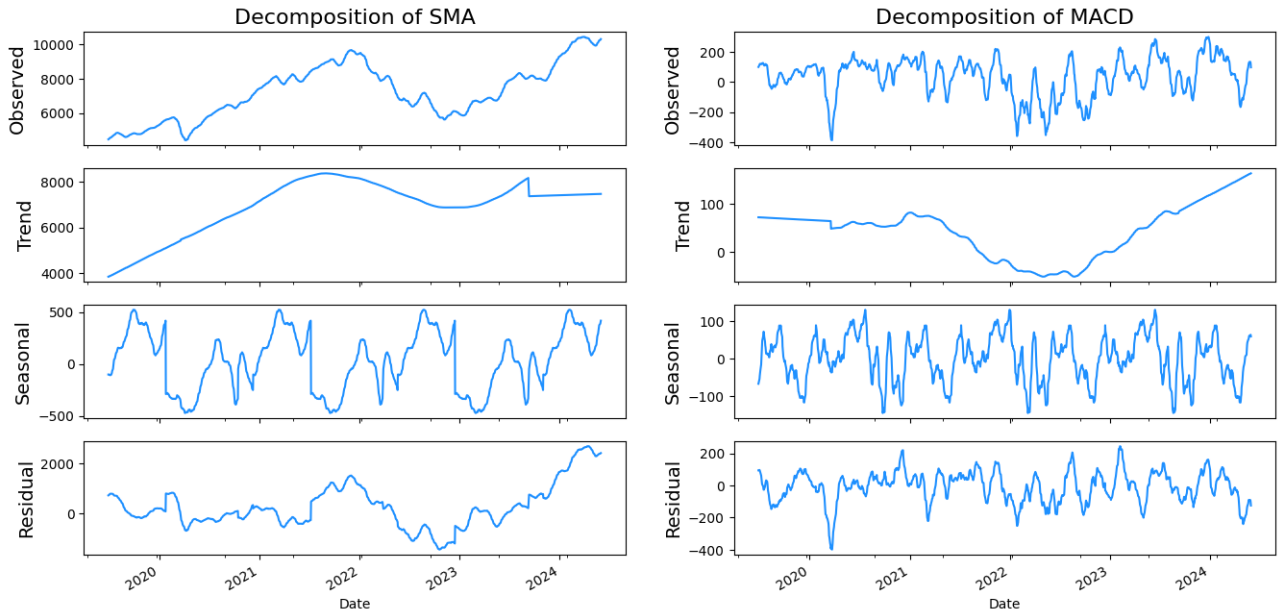

Figure 19: Decomposition plots for SMA and MACD

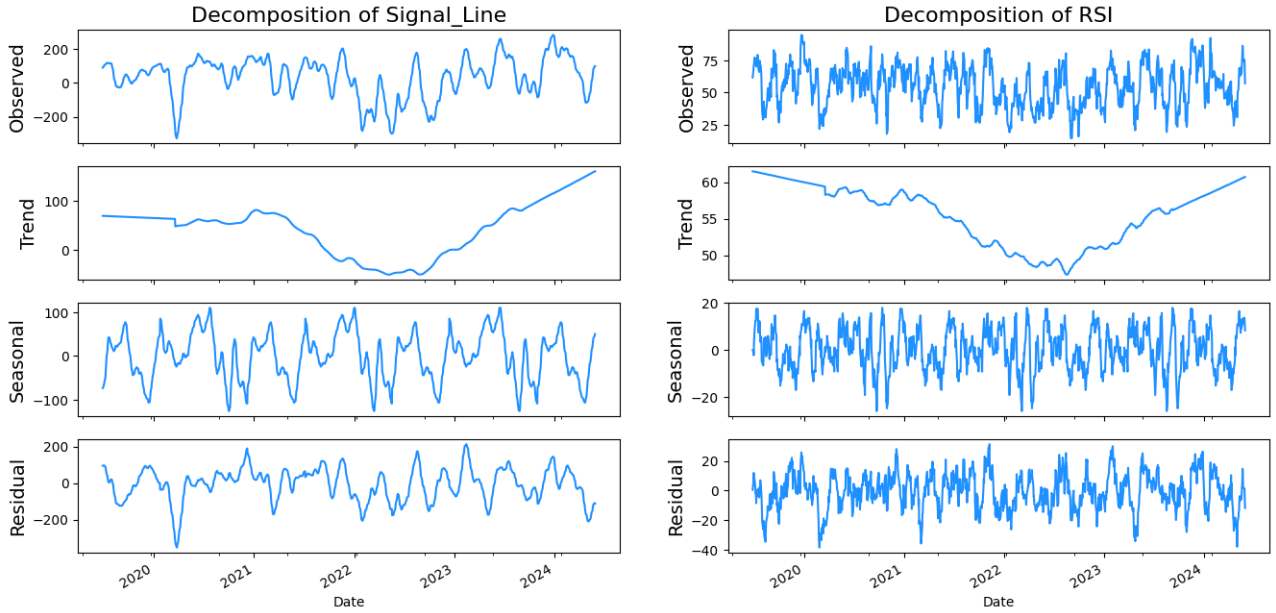

Figure 20: Decomposition plots for Signal_Line and RSI

– Correlation matrix of the dataset:

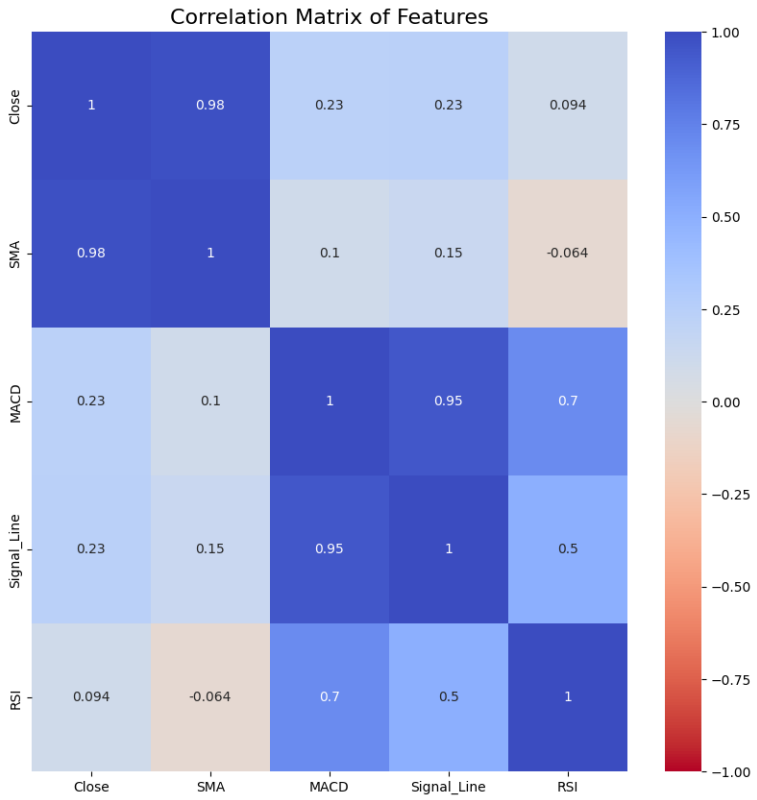

Figure 21: Correlation matrix

From the correlation matrix, it can be observed that except for SMA—which exhibits very high correlation—the remaining indicators show relatively low correlation with the Close price.

- Economic dataset:

– Overview of the dataset:

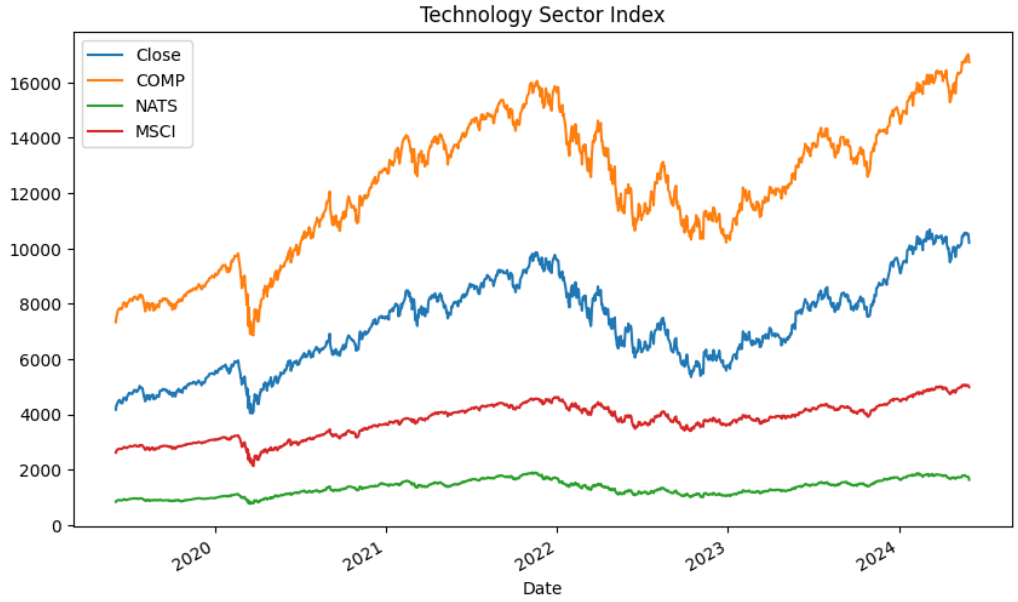

Figure 22: Plot of economic indicators

– Rolling mean and rolling standard deviation with a 30-period window:

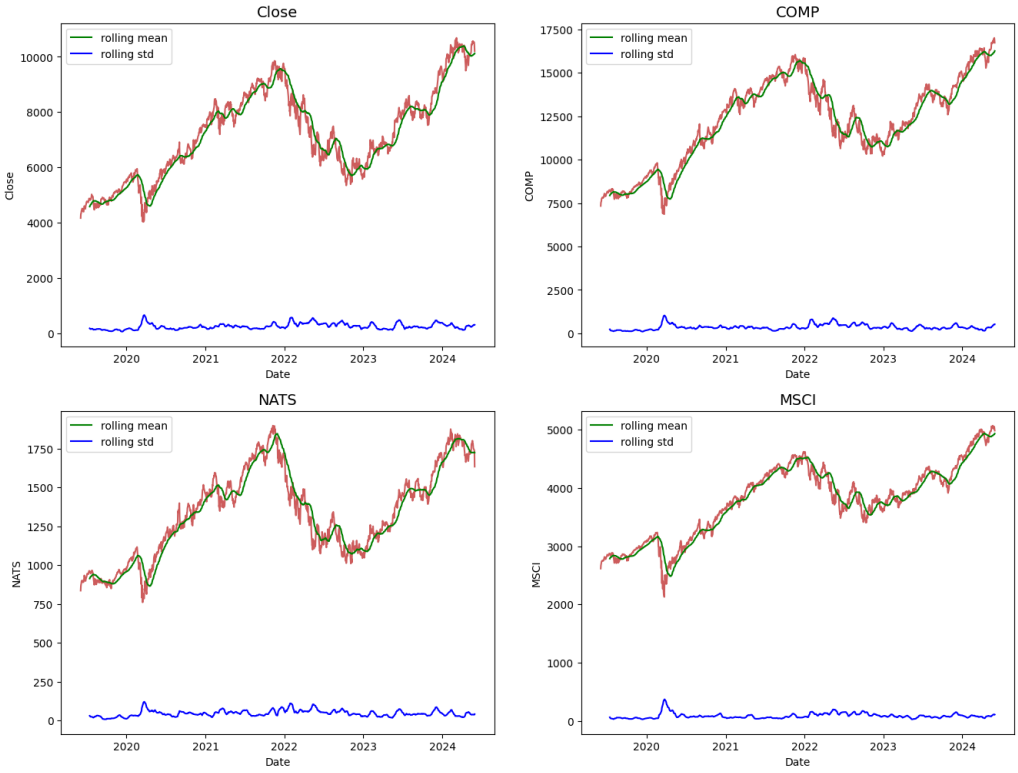

Figure 23: Rolling Mean and Rolling Std plots

– Trend and seasonality decomposition for the attributes:

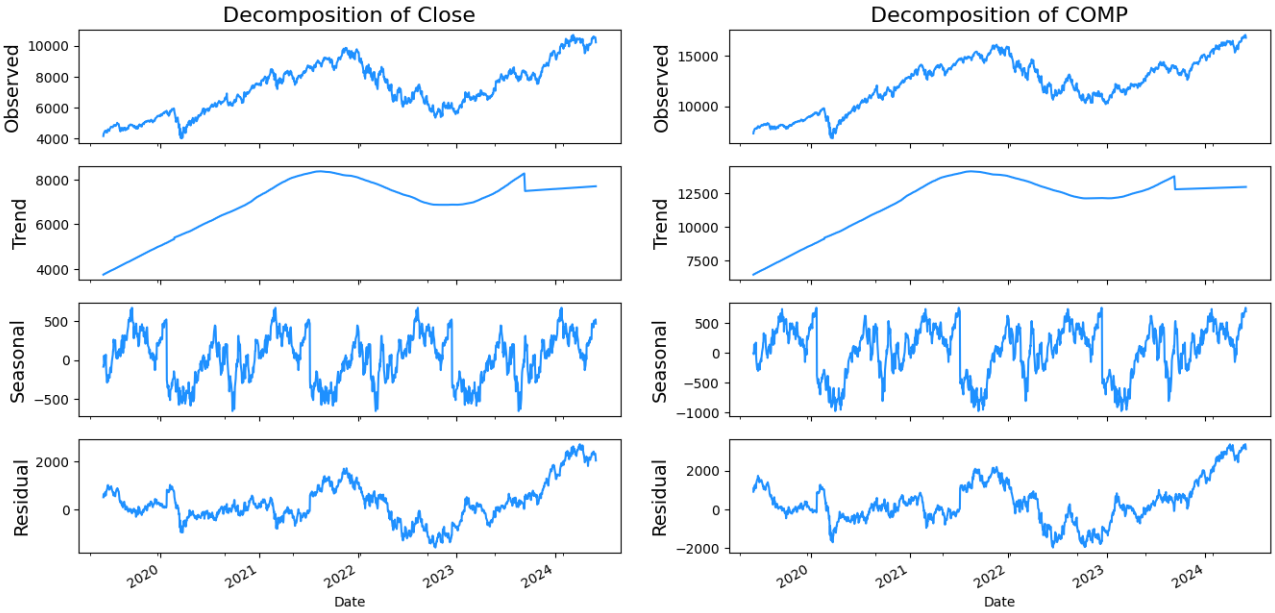

Figure 24: Decomposition plots for Close and COMP

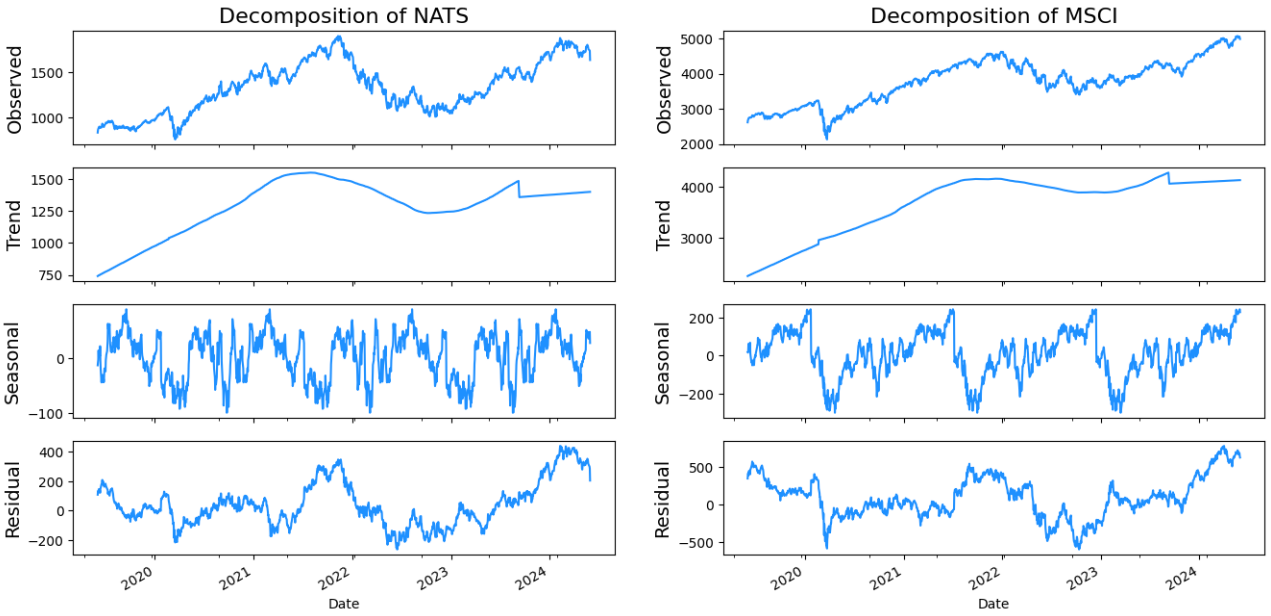

Figure 25: Decomposition plots for NATS and MSCI

– Correlation matrix of the dataset:

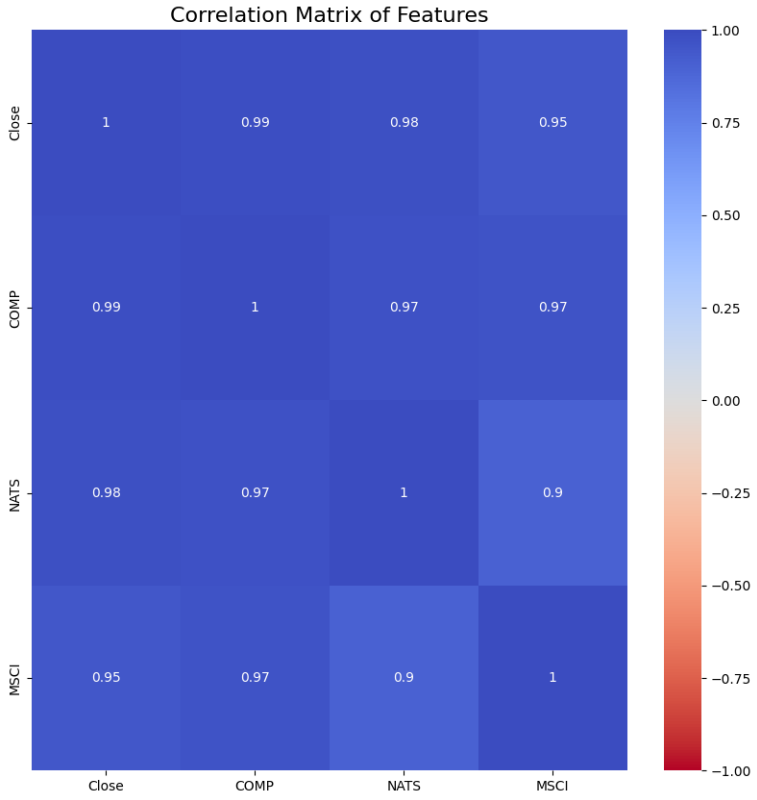

Figure 26: Correlation matrix

From the correlation matrix, it can be observed that the attributes exhibit very high correlation with the Close price.

3.1.3 Data Preprocessing

The dataset would be augmented in the event of missing observations. However, in this report, none of the three datasets contain missing values, so this step is omitted. To prepare the data for model training, each dataset undergoes different preprocessing procedures depending on the type of model. The process is conducted as follows:

1. Classical Statistical Models:

- Main Dataset: A Granger causality test is performed on the dataset, yielding the following results:

	Close_x	Open_x	High_x	Low_x
Close_y	1.0	0.0326	0.0582	0.0002
Open_y	0.0	1.0000	0.0000	0.0000
High_y	0.0	0.0000	1.0000	0.0000
Low_y	0.0	0.0000	0.0000	1.0000

Based on the results, the attribute *High* is removed since 0.0582 ¿ 0.05. The dataset is then tested for stationarity using the Augmented Dickey–Fuller (ADF) test:

```
=> P-Value = 0.6388. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.5645. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.5823. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.
```

Since all the time series are non-stationary, a natural logarithm transformation and first differencing are applied to the entire dataset. The resulting series become stationary and satisfy the Granger causality requirements:

```
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

	Close_x	Open_x	Low_x
Close_y	1.0	0.0037	0.0019
Open_y	0.0	1.0000	0.0000
Low_y	0.0	0.0000	1.0000

After model selection, this report employs a VAR(6) model, using six lagged observations (i.e., data from the six preceding days) as input features for forecasting.

- Technical Dataset: A Granger causality test is performed on this dataset as well:

	Close_x	SMA_x	MACD_x	Signal_Line_x	RSI_x
Close_y	1.000	0.0313	0.0091	0.0223	0.0394
SMA_y	0.000	1.0000	0.0000	0.0000	0.0000
MACD_y	0.001	0.0007	1.0000	0.0000	0.0000
Signal_Line_y	0.000	0.0000	0.0000	1.0000	0.0000
RSI_y	0.000	0.0000	0.0000	0.0000	1.0000

The dataset is then tested for stationarity using the ADF test:

```
=> P-Value = 0.6967. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.6548. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0003. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

The first two series are non-stationary and therefore require differencing. However, differencing reduces the length of these two series by one observation relative to the others. To address this imbalance, a one-step backward linear prediction is performed to impute the missing initial values. Afterward, the dataset is standardized by subtracting the mean:

$$\tilde{x}_i = x_i - \text{mean}$$

After that, the following steps are performed to compute the coefficients Θ for the backward prediction model:

$$\Theta = \Gamma^{-1}\gamma$$

where:

$$\Gamma = \begin{pmatrix} \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_2 & \gamma_1 & \gamma_0 \end{pmatrix}$$

$$\gamma = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix}$$

$\gamma_0 = \frac{1}{n}\sum_{i=0}^{n-1}(\tilde{x}_i)^2$; $\gamma_1 = \frac{1}{n}\sum_{i=0}^{n-2}(\tilde{x}_i \cdot \tilde{x}_{i+1})$; $\gamma_2 = \frac{1}{n}\sum_{i=0}^{n-3}(\tilde{x}_i \cdot \tilde{x}_{i+2})$;
$\gamma_3 = \frac{1}{n}\sum_{i=0}^{n-4}(\tilde{x}_i \cdot \tilde{x}_{i+3})$
The resulting prediction is computed as:

$$\text{predict} = \theta_1\tilde{x}_0 + \theta_2\tilde{x}_1 + \theta_3\tilde{x}_2 + \text{mean}$$

From this, the obtained series becomes stationary and satisfies the Granger causality requirements:

```
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0003. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

	Close_x	SMA_x	MACD_x	Signal_Line_x	RSI_x
Close_y	1.0000	0.0392	0.0166	0.0171	0.0482
SMA_y	0.0001	1.0000	0.0000	0.0000	0.0000
MACD_y	0.0000	0.0000	1.0000	0.0000	0.0000
Signal_Line_y	0.0000	0.0000	0.0000	1.0000	0.0000
RSI_y	0.0000	0.0001	0.0000	0.0000	1.0000

After the model selection process, this report employs a VAR(2) model for this dataset, using two lagged observations corresponding to the two preceding days as input features for forecasting.

- Economic Dataset: A Granger causality test is conducted on the dataset:

	Close_x	COMP_x	NATS_x	MSCI_x
Close_y	1.0000	0.0247	0.0264	0.0176
COMP_y	0.2764	1.0000	0.0305	0.0007
NATS_y	0.0505	0.0094	1.0000	0.0006
MSCI_y	0.0073	0.0059	0.0088	1.0000

The dataset is then tested for stationarity using the ADF test:

```
=> P-Value = 0.6388. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.5818. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.5297. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.


=> P-Value = 0.6867. Weak evidence to reject the Null Hypothesis.
=> Series is Non-Stationary.
```

A natural logarithm transformation is applied to *NATS*, and differencing is applied to the entire dataset. The resulting series are stationary and satisfy the Granger causality requirements:

```
=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.


=> P-Value = 0.0. Rejecting Null Hypothesis.
=> Series is Stationary.
```

	Close_x	COMP_x	NATS_x	MSCI_x
Close_y	1.0000	0.0208	0.0101	0.0047
COMP_y	0.1546	1.0000	0.0752	0.0007
NATS_y	0.1879	0.1700	1.0000	0.0021
MSCI_y	0.0052	0.0092	0.4805	1.0000

After model selection, this report employs a VAR(1) model for this dataset, with one lagged observation corresponding to the preceding day serving as the input variable for forecasting.

2. Deep Learning Models:

   (a) Normalization of the dataset: All attributes from the three datasets are normalized using a Min–Max Scaler to ensure that all values fall within the same numerical range. This helps deep learning models learn the relationships between variables more effectively.

(b) Transformation to cross-sectional format: The time series data is transformed into a cross-sectional (supervised learning) format using a sliding window technique. In this report, the model predicts future values based on the preceding 30 days; therefore, the window size is set to 30 with a step size of 1. In other words, the model receives 30 input variables for each prediction.

3.2 Model Evaluation Methods

Evaluating the model is a crucial step in the deep learning model development workflow, as it helps determine the accuracy and effectiveness of stock price forecasting models. The group selected evaluation methods based on criteria related to accuracy and practical applicability. Two widely used evaluation metrics employed in this report are RMSE and MAPE. These metrics are used to assess both the predictive accuracy and the usability of the models.

3.2.1 RMSE

Root Mean Squared Error (RMSE) is a statistical metric that measures the difference between the predicted values generated by a model and the actual observed values. RMSE is computed by taking the square root of the average of the squared errors. An error is defined as the difference between the predicted value and the actual value. The formula for RMSE is as follows:

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

where:

- n is the number of samples.
- y_i is the actual value.
- $\hat{y}_i$ is the model's predicted value.

RMSE is a popular metric because it is intuitive and has the same units as the target variable, making it easy to compare performance across different models. A lower RMSE indicates that the model has better predictive performance, while a higher RMSE suggests poorer forecasting capability.

3.2.2 MAPE

Mean Absolute Percentage Error (MAPE) is another statistical metric used to evaluate forecasting accuracy. MAPE is calculated by taking the average percentage of absolute errors between the predicted and actual values. The formula for MAPE is:

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

where:

- n is the number of samples.
- y_i is the actual value.
- $\hat{y}_i$ is the model's predicted value.

MAPE provides the average percentage deviation between the forecasted and actual values. This metric allows for evaluating the forecasting error relative to the magnitude of the true values, making it useful for comparing model accuracy across different datasets or different models. A lower MAPE value indicates a more accurate forecasting model.

3.3 Results

The dataset is divided according to the specific requirements of each model. For statistical models, the test set spans from May 1, 2024 to May 30, 2024, while the remaining data is used for training. For deep learning models, the training set, validation set, and test set are split following an 80:10:10 ratio based on the chronological order of the time series. After the training process, the models generate results that are evaluated against the test set as follows:

- One-day-ahead forecasting based on previous days
  - Main dataset:

	RMSE	MAPE
VAR	291.54	2.33%
1-layer LSTM	162.04	1.3%
2-layer LSTM	**145.33**	**1.16%**
2-layer LSTM (*)	202.38	1.67%
BiLSTM	163.04	1.34%
Transformer	309.12	2.69%

Table 1: Results on the main dataset

The 2-layer LSTM (*) corresponds to the model used in the referenced research paper.

– Technical dataset:

	RMSE	MAPE
VAR	322.87	2.55%
1-layer LSTM	210.76	1.69%
2-layer LSTM	**150.36**	**1.24%**
2-layer LSTM (*)	303.27	2.64%
BiLSTM	300.88	2.58%
Transformer	417.59	3.72%

Table 2: Results on the technical dataset

– Economic dataset:

	RMSE	MAPE
VAR	308.93	2.45%
1-layer LSTM	275.81	2.39%
2-layer LSTM	280.9	2.45%
2-layer LSTM (*)	**185.81**	**1.54%**
BiLSTM	364.76	3.29%
Transformer	406.09	3.69%

Table 3: Results on the economic dataset

Based on the results, the 2-layer LSTM model using the main dataset delivers the best forecasting performance. A potential reason is that this model was trained for more epochs and is inherently better suited to the characteristics of the dataset. Furthermore, the findings highlight that the improved 2-layer LSTM model performs better than the reference model.

- Multi-step forecasting based on previous days

	RMSE	MAPE
VAR (*)	274.54	2.17%
VAR (**)	274.44	2.16%
VAR (***)	279.04	2.19%

Table 4: Results of multi-step forecasting

VAR(*), VAR(**), VAR(***) correspond to the VAR models trained on the main dataset, the technical dataset, and the economic dataset, respectively. The results show that the VAR model performs reasonably well under long-term forecasting conditions. Additionally, the findings indicate that the VAR model achieves better performance in long-term forecasting compared to short-term forecasting, a point that will be analyzed in later sections.

Below are the plots illustrating the one-day-ahead forecasts compared to the actual values for each model and dataset:

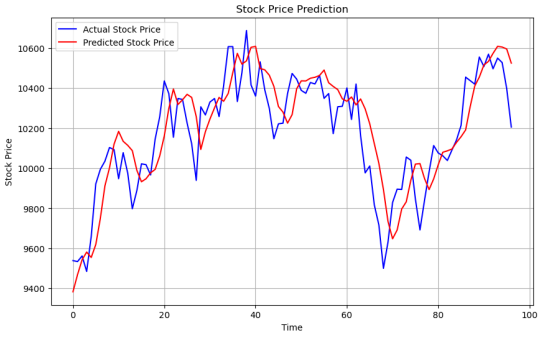

Figure 27: 1-layer LSTM with the main dataset

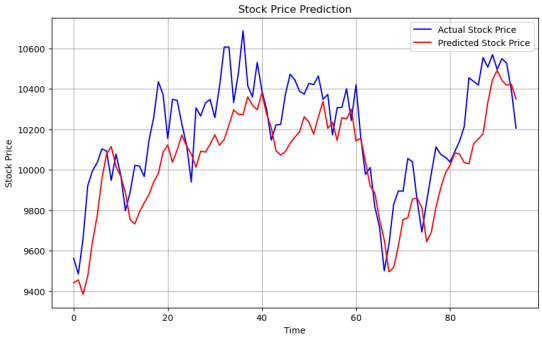

Figure 28: 1-layer LSTM with the technical dataset

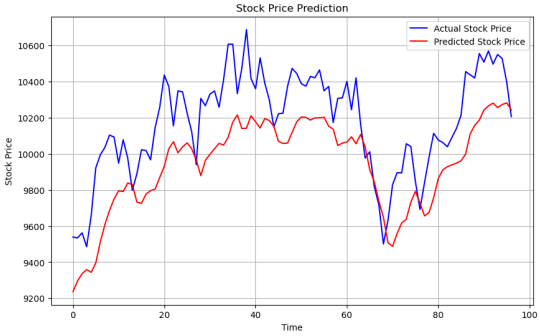

Figure 29: 1-layer LSTM with the economic dataset

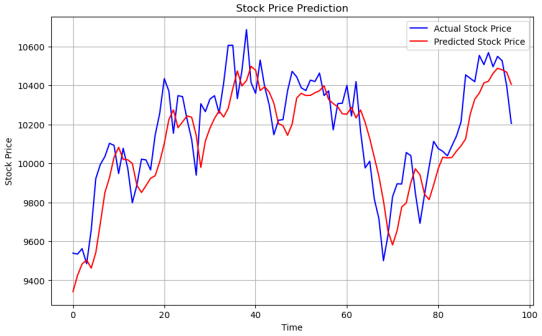

Figure 30: BiLSTM with the main dataset

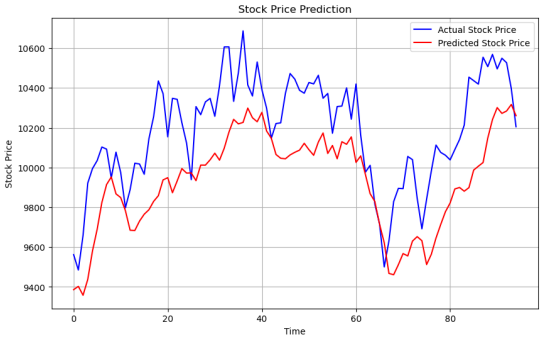

Figure 31: BiLSTM with the technical dataset

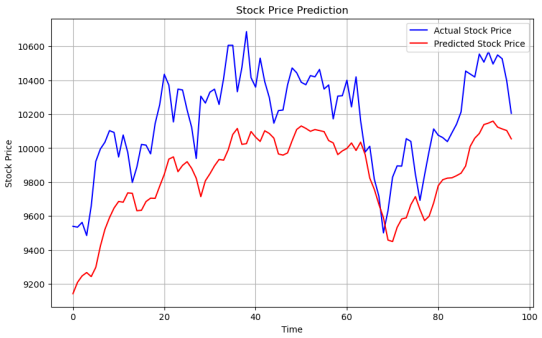

Figure 32: BiLSTM with the economic dataset

3.4 Error Statistics and Analysis

Based on the performance results of the models, several of them encountered notable issues as follows:

- Transformer Model: The fact that the Transformer model yields higher MAPE

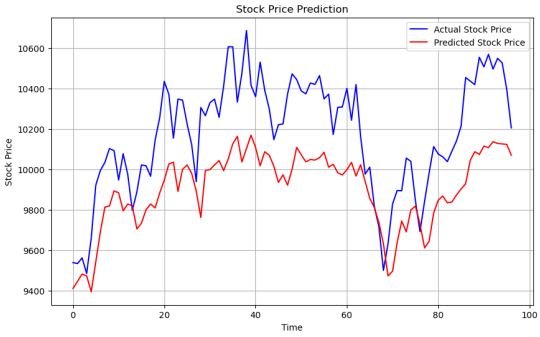

Figure 33: Transformer with the main dataset

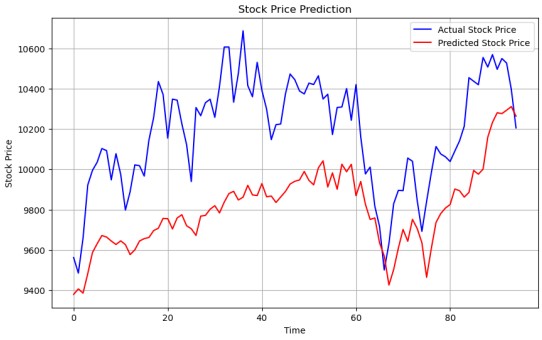

Figure 34: Transformer with the technical dataset

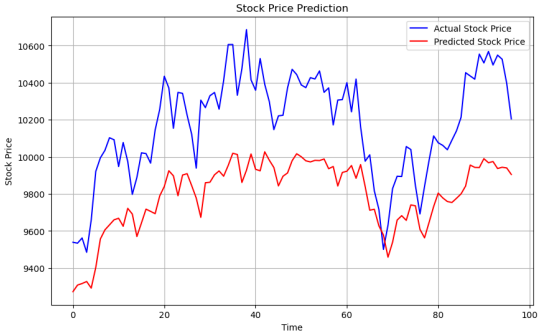

Figure 35: Transformer with the economic dataset

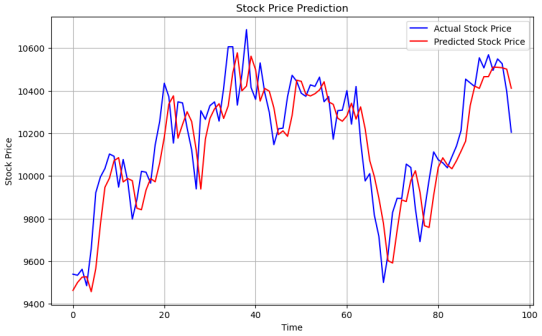

Figure 36: 2-layer LSTM with the main dataset

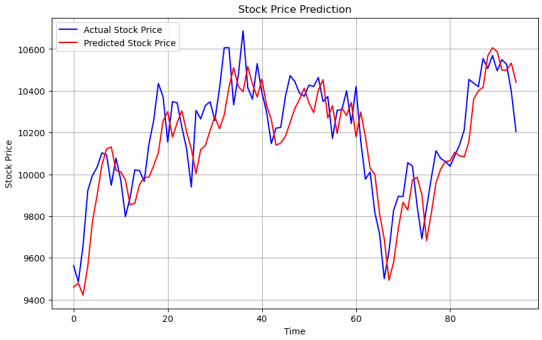

Figure 37: 2-layer LSTM with the technical dataset

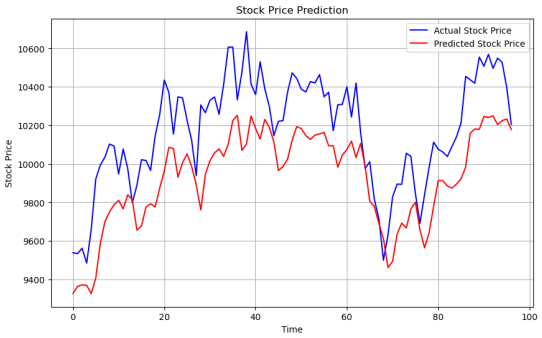

Figure 38: 2-layer LSTM with the economic dataset

(Mean Absolute Percentage Error) and MSE (Mean Squared Error) in predicting stock closing prices despite its more complex architecture may stem from several reasons:

– The Transformer requires a large and diverse dataset to perform effectively, whereas the current dataset may be insufficient in size or not properly cleaned.

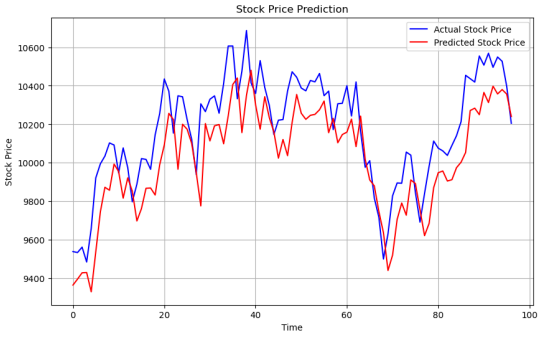

Figure 39: Reference 2-layer LSTM with the main dataset

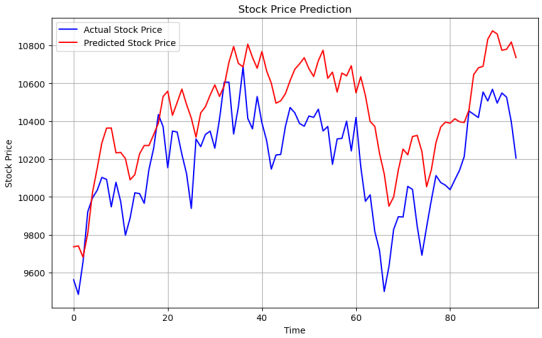

Figure 40: Reference 2-layer LSTM with the technical dataset

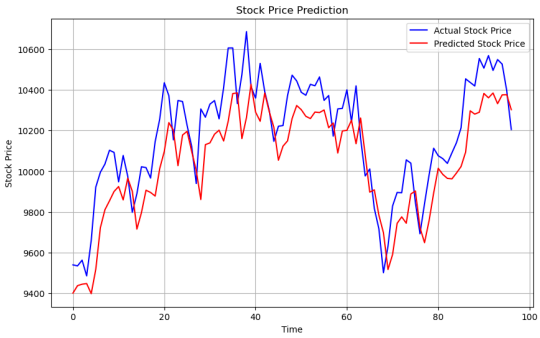

Figure 41: Reference 2-layer LSTM with the economic dataset

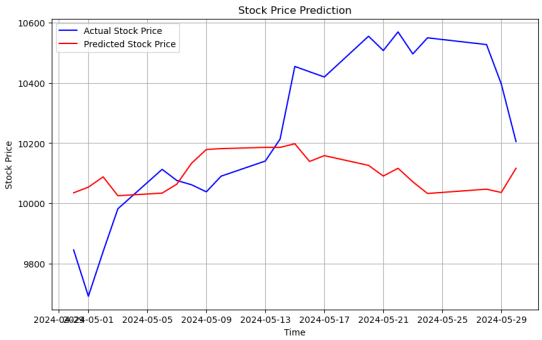

Figure 42: VAR with the main dataset

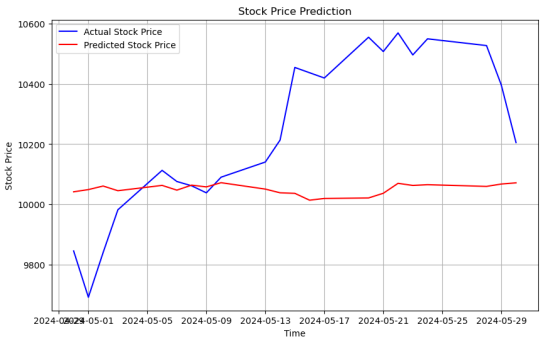

Figure 43: VAR with the technical dataset

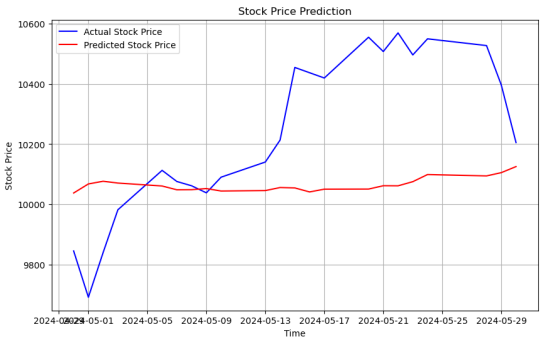

Figure 44: VAR with the economic dataset

– The Transformer contains many hyperparameters that need to be fine-tuned (e.g., number of heads, number of layers, feed-forward dimension, etc.). Suboptimal configurations can prevent the model from learning effectively.

– An insufficient number of epochs or an inappropriate learning rate may prevent the model from fully converging.

  - The Transformer is prone to overfitting if regularization techniques are not properly applied.
  - The Transformer requires significantly more computational resources compared to LSTM and BiLSTM. Limited computational resources can result in incomplete training, affecting prediction performance. Additionally, because the training data only spans 30 past days, the Transformer may require a much longer time window (e.g., 100 past days) to function effectively.
  - Stock market data typically contains substantial noise and complex nonlinear patterns. The Transformer may struggle to handle such noise if the data is not thoroughly preprocessed.

  Some potential solutions to address these issues include:

  - Collecting more stock market data.
  - Using Grid Search, Random Search, or Bayesian Optimization to identify the optimal hyperparameter configuration.
  - Increasing the number of epochs and experimenting with different learning rates.

- VAR Model: The fact that the VAR model performs worse in short-term forecasting compared to long-term forecasting may be attributed to the following factors:

  - The VAR model may lack sufficient complexity to capture the structure of the dataset.
  - Stock market data often contains cyclical patterns and irregular fluctuations. In the short term, these fluctuations can be large and difficult to predict accurately. In the long term, underlying trends tend to become clearer and easier to forecast.
  - Short-term fluctuations are often influenced by unexpected economic events or new financial policies, which the VAR model is unable to capture. These factors tend to exert a more significant impact on short-term predictions than on long-term ones.

  Potential solutions include:

  - Employing other statistical models such as VARMA, VARX, or deep learning models to improve predictive accuracy.
  - Incorporating additional exogenous variables or other important factors that may have been omitted, thereby capturing a more complete set of short-term influencers.

4 Model Deployment

4.1 Program Interface

The program begins with the following interface:

This program uses 12 deep learning models (excluding the 2-layer LSTM model referenced from the previous study) to perform short-term forecasting, and 3 VAR models for

Figure 45: Interface

long-term forecasting over a 30-day period from May 1, 2024, to May 30, 2024. Since the final forecasting date has already passed, both the short-term and long-term forecasting models are able to predict for all days under consideration.

After selecting any specific date, for example **May 9, 2024**, and any model, for example **BiLSTM with Main Data**—which refers to using the BiLSTM model with the main dataset—the output will be as follows:

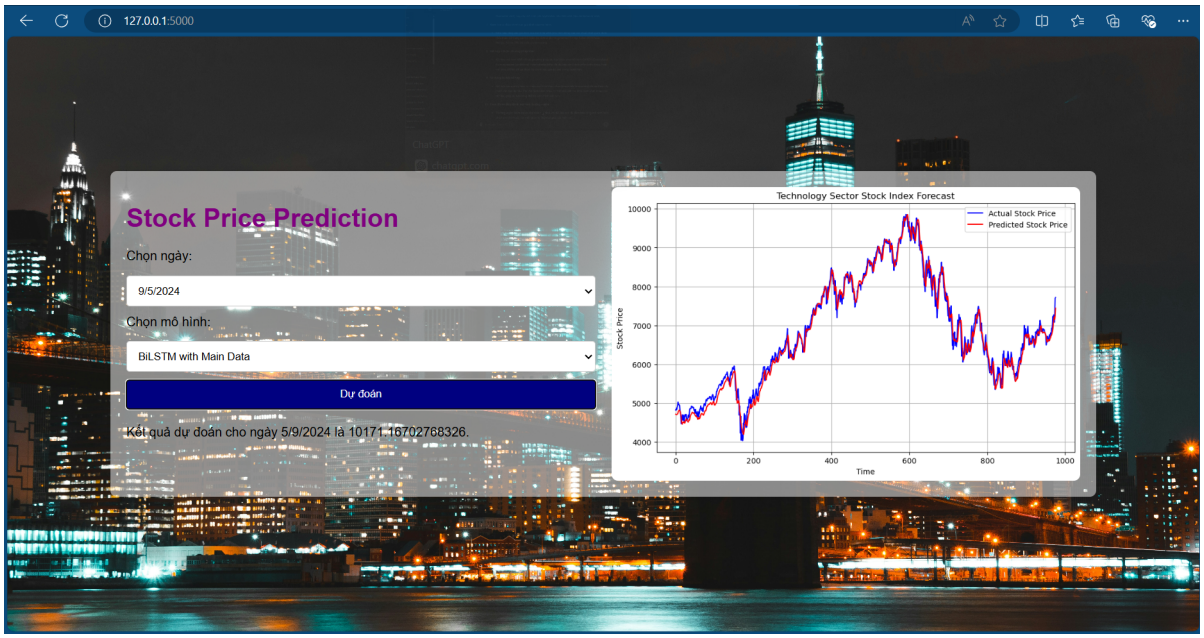

Figure 46: Result

The returned output includes the *Close* price predicted by the model for the selected date. To the right of the result is a chart illustrating the predicted and actual time series on the training set, providing a more intuitive visual representation for the user.

4.2 Program Scenarios

A is a financial investor who wants to forecast the stock price of the technology sector for the next day, assumed to be May 15, 2024. A has access to an online tool capable of predicting stock prices based on statistical and deep learning models using various datasets. A proceeds as follows:

1. Access the web application: Run the command **flask run** in the terminal, then navigate to **http://127.0.0.1:5000**.
2. Select the prediction date as May 15, 2024.
3. Test all available models to determine a reasonable prediction range.
4. Based on the obtained results, A can make a more informed investment decision. If the predicted stock price is significantly higher than the current price, A may consider buying. Conversely, if the predicted price is lower, A may consider selling or choosing not to invest at that time.

Conclusion

Above is our group's report on the topic Stock Forecasting in the Technology Sector. By applying the knowledge learned from the course Decision Support Systems, as well as combining additional insights from various references and articles, we have presented content on machine learning models, deep learning models, and classical statistical methods, along with their application to stock forecasting and the web interface after integrating the models used.

During the process of preparing this report, our team members have acquired many new skills, such as teamwork, research, and information retrieval.

Due to the limited time available for preparing the report and our current level of knowledge, the report inevitably contains some shortcomings. We sincerely welcome any feedback and suggestions from you, and once again, we would like to express our gratitude to Dr. Tran Ngoc Thang for his dedicated teaching and guidance throughout this course.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[2] Tran Phuoc et al. "Applying machine learning algorithms to predict the stock price trend in the stock market–The case of Vietnam". In: *Humanities and Social Sciences Communications* 11.1 (2024), pp. 1–18.

[3] Mike Schuster and Kuldip K. Paliwal. "Bidirectional Recurrent Neural Networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[4] Jaydip Sen, Saikat Mondal, and Gourab Nath. "Robust portfolio design and stock price prediction using an optimized LSTM model". In: *2021 IEEE 18th India Council International Conference (INDICON)*. IEEE. 2021, pp. 1–6.

[5] Christopher A. Sims. "Macroeconomics and Reality". In: *Econometrica* 48.1 (1980), pp. 1–48.

[6] Ashish Vaswani et al. "Attention is All You Need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.