

STOCK MARKET PREDICTION USING HYBRID MODEL

A PROJECT REPORT

*Submitted by*

BALASUBARAMANIYAM TS (2022503044)

MOHAMED ANAS MH (2022503050)

BALAMURUGAN S (2022503508)

COURSE CODE: CS6611

COURSE TITLE: CREATIVE AND INNOVATIVE PROJECT



DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHENNAI – 600044

APRIL 2025

**DEPARTMENT OF COMPUTER TECHNOLOGY****ANNA UNIVERSITY, MIT CAMPUS****CHROMEPET, CHENNAI – 600044****BONAFIDE CERTIFICATE**

Certified that this project report “**Stock Market Prediction Using Hybrid Model (LSTM Model and TCN Model)**” is the work of **Mr. Balasubaramaniyam TS (2022503044)**, **Mr. Mohamed Anas MH (2022503050)**, **Mr. Balamurugan S (2022503508)** in the Creative and Innovative Project Laboratory subject code CS6611 during the period January to May 2025.

**SIGNATURE****Dr. S. MUTHURAJKUMAR****SUPERVISOR**

Associate Professor

Department of Computer Technology

Anna University, MIT Campus

Chromepet – 600 044.

**SIGNATURE****Dr. P. JAYASHREE****HEAD OF THE DEPARTMENT**

Professor

Department of Computer Technology

Anna University, MIT Campus

Chromepet – 600 044.

## ABSTRACT

Stock market prediction is a complex task due to inherent volatility influenced by economic indicators, geopolitical events, and investor sentiment. Traditional statistical methods and single-model deep learning approaches often fail to capture multi-scale temporal dependencies.

This project proposes a hybrid deep learning model combining Long Short-Term Memory (LSTM) networks and Temporal Convolutional Networks (TCN) to simultaneously capture both long-term trends and short-term patterns in stock price movements.

The architecture uniquely integrates LSTM's sequential learning capabilities with TCN's parallel processing efficiency through an optimized fusion mechanism. The model uses sliding window method for predicting future prices with the help of multiple technical indicators (RSI, MACD, Bollinger Bands) and historical price data.

The hybrid model is rigorously evaluated using MSE, RMSE, MAE, and  $R^2$  metrics across diverse market regimes, including bull/bear markets, high volatility periods, and crash scenarios.

Experimental results demonstrate a 22.3% lower RMSE than state-of-the-art benchmarks, with particular strength in volatile markets (35% error reduction during earnings seasons).

Additionally, a full-stack web application with React.js and Django is implemented to enable real-time predictions and trading recommendations, achieving <300ms prediction latency. The work advances financial AI by offering a computationally efficient, adaptive solution that bridges the gap between academic research and practical trading applications.

## ACKNOWLEDGEMENT

We take this humble opportunity to thank the Dean, MIT Campus, Anna University, Dr. Ravichandran K, and Dr. Jayashree P, Professor & Head, Department of Computer Technology, MIT Campus, Anna University, for providing all the lab facilities in pursuit of this project.

Undertaking this project has helped us learn a lot, and we would like to express our gratitude towards our supervisor, Dr. Muthurajkumar S, Associate Professor, Department of Computer Technology, MIT, Anna University, whose guidance and directions helped shape this project perfectly. The feedback from the supervisor was very instrumental in the successful completion of the project work.

We acknowledge the efforts and feedback of the panel members Dr. Chithra S, Associate Professor, Dr. Neelavathy Pari, Assistant Professor and Dr. Kottilingam K, Assistant Professor, Department of Computer Technology, MIT, Anna University, in reviewing our work, providing constant valuable comments and encouraging us to view the different aspects of the project in successful implementation of the project.

We thank all the teaching and non-teaching members of the Department of Computer Technology, MIT, Anna University, whose appreciable help, either directly or indirectly, aided in the smooth completion of the Creative and Innovative Project within a limited time frame.

BALASUBARAMANIYAM TS (2022503044)

MOHAMED ANAS MH (2022503050)

BALAMURUGAN S (2022503508)

## TABLE OF CONTENTS

| CHAPTER NO. | TITLE                                       | PAGE NO. |
|-------------|---|----------|
| <b>1</b>    | <b>INTRODUCTION</b>                         | <b>1</b> |
| 1.1         | STOCK MARKET PREDICTION                     | 1        |
| 1.2         | DEEP LEARNING IN FINANCE                    | 2        |
| 1.2.1       | LONG SHORT-TERM MEMORY<br>(LSTM)            | 2        |
| 1.2.2       | TEMPORAL CONVOLUTIONAL<br>NETWORKS (TCN)    | 3        |
| 1.3         | HYBRID MODELS                               | 4        |
| 1.4         | OBJECTIVE                                   | 4        |
| <b>2</b>    | <b>LITERATURE SURVEY</b>                    | <b>5</b> |
| <b>3</b>    | <b>PROPOSED SYSTEM</b>                      | <b>7</b> |
| 3.1         | ARCHITECTURE OVERVIEW                       | 7        |
| 3.1.1       | DATA PROCESSING                             | 7        |
| 3.1.2       | HYBRID MODEL: TCN + LSTM                    | 8        |
| 3.1.3       | REAL-TIME PREDICTION AND<br>WEB APPLICATION | 8        |

| <b>CHAPTER NO.</b> | <b>TITLE</b>                                 | <b>PAGE NO.</b> |
|--------------------|--|-----------------|
|                    | 3.2 DATA COLLECTION                          | 11              |
|                    | 3.3 PREPROCESSING AND FEATURE<br>ENGINEERING | 14              |
|                    | 3.4 MODEL ARCHITECTURE                       | 17              |
|                    | 3.5 HYBRID MODEL IMPLEMENTATION              | 20              |
|                    | 3.6 EVALUATION METRICS                       | 22              |
| <b>4</b>           | <b>IMPLEMENTATION</b>                        | <b>24</b>       |
|                    | 4.1 PLATFORM USED                            | 24              |
|                    | 4.1.1 GOOGLE COLAB                           | 24              |
|                    | 4.1.2 DJANGO                                 | 24              |
|                    | 4.1.3. REACT                                 | 24              |
|                    | 4.1.4. MYSQL                                 | 25              |
|                    | 4.2 TOOLS USED                               | 25              |
|                    | 4.3 PSEUDO CODE                              | 26              |
|                    | 4.3.1 DATA COLLECTION                        | 26              |
|                    | 4.3.2 DATA PREPROCESSING                     | 27              |
|                    | 4.3.3 MODEL TRAINING                         | 28              |
|                    | 4.3.4 FUTURE PREDICTIONS                     | 29              |
|                    | 4.3.5 SMART MONEY ENTRIES                    | 30              |

|          |                              |           |
|----------|------------------------------|-----------|
|          | 4.3.6 TREND OF STOCK         | 31        |
|          | 4.3.7 FRONTEND–BACKEND       |           |
|          | DATABASE INTERACTION         | 32        |
| <b>5</b> | <b>RESULTS AND ANALYSIS</b>  | <b>33</b> |
|          | 5.1 VISUALIZATION OF         |           |
|          | PREDICTIONS                  | 33        |
|          | 5.1.1 TECHNICAL INDICATORS   | 33        |
|          | 5.1.2 MODEL PREDICITON FOR   |           |
|          | AAPL STOCK                   | 35        |
|          | 5.2 MODEL COMPARISON         | 36        |
|          | 5.3 FUTURE FORECASTING       | 37        |
|          | 5.3.1 FUTURE STOCK           |           |
|          | PREDIITONS FOR AAPL STOCK    | 38        |
|          | 5.3.2 AAPL STOCK             |           |
|          | PREDICTION DURING COVID      | 38        |
|          | 5.3.3 STOCK TREND PREDICITON | 39        |
|          | 5.3.4 SMART MONEY ENTRIES    |           |
|          | PREDICTION                   | 39        |
|          | 5.4 STOCK MARKET WEBPAGE     | 40        |

| <b>CHAPTER NO.</b> | <b>TITLE</b>                          | <b>PAGE NO.</b> |
|--------------------|---------------------------------------|-----------------|
| <b>6</b>           | <b>CONCLUSION AND FUTURE<br/>WORK</b> | <b>41</b>       |
| <b>7</b>           | <b>REFERENCES</b>                     | <b>43</b>       |



## LIST OF FIGURES

| <b>FIGURE NO.</b> | <b>TITLE</b>             | <b>PAGE NO.</b> |
|-------------------|--------------------------|-----------------|
| 3.1               | Architecture Diagram     | 7               |
| 3.2               | Data Collection          | 8               |
| 3.3               | Data Preprocessing       | 9               |
| 3.4               | Data After Scaling       | 11              |
| 3.5               | TCN Model                | 13              |
| 3.6               | LSTM Model               | 14              |
| 5.1               | RSI                      | 25              |
| 5.2               | Bollinger Bands          | 25              |
| 5.3               | MACD                     | 26              |
| 5.4               | ATR                      | 26              |
| 5.5               | LSTM Predictions         | 26              |
| 5.6               | TCN Predictions          | 27              |
| 5.7               | Hybrid Model Predictions | 27              |
| 5.8               | Future Predictions       | 29              |
| 5.9               | Predictions On COVID     | 30              |
| 5.10              | Stock Trend              | 30              |
| 5.11              | Smart Money Entries      | 31              |
| 5.12              | Admin Dashbord           | 31              |
| 5.13              | Database                 | 3               |

## LIST OF TABLES

| TABLE NO. | TITLE              | PAGE NO. |
|-----------|--------------------|----------|
| 3.1       | Evaluation Metrics | 17       |
| 4.1       | Tools Used         | 19       |
| 5.1       | Model Comparison   | 28       |

## LIST OF ABBREVIATIONS

|                            |   |                                       |
|----------------------------|---|---------------------------------------|
| <b>LSTM</b>                | - | Long Short Term Model                 |
| <b>TCN</b>                 | - | Temporal Convolution Model            |
| <b>BB</b>                  | - | Bollinger Bands                       |
| <b>MACD</b>                | - | Moving Average Convergence Divergence |
| <b>RSI</b>                 | - | Relative Strength Index               |
| <b>MSE</b>                 | - | Mean Squared Error                    |
| <b>RMSE</b>                | - | Root Mean Squared Error               |
| <b>MAE</b>                 | - | Mean Absolute Error                   |
| <b>R<sup>2</sup> SCORE</b> | - | R-Squared Score                       |

## CHAPTER 1

### INTRODUCTION

The stock market is a complex and dynamic environment influenced by various factors, including macroeconomic indicators (e.g., interest rates, inflation), geopolitical events, and investor sentiment. Predicting stock prices accurately remains a major challenge due to the market's highly volatile and non-linear nature.

Traditional models like ARIMA and linear regression assume stationarity and linearity, which limits their effectiveness in real-world financial time series. While machine learning and deep learning models offer improvements by capturing complex patterns, many still struggle with modeling temporal dependencies and long-range correlations effectively. Models like SVMs and Random Forests improve upon traditional methods but are not optimized for sequential data.

#### 1.1 STOCK MARKET PREDICTION

The stock market is a dynamic environment influenced by various factors, including macroeconomic indicators, geopolitical events, and investor sentiment. Predicting stock prices is challenging due to the market's inherent volatility and non-linear behavior. Traditional statistical models like ARIMA and linear regression often fall short in capturing these complexities.

Machine Learning (ML) and Deep Learning (DL) techniques have emerged as powerful tools for stock price forecasting. Models such as Support Vector Machines (SVM) and Random Forests offer improvements over traditional methods but may struggle with sequential data. Deep learning architectures, particularly Long Short-Term Memory (LSTM) networks and Temporal Convolutional Networks (TCNs), have shown promise in modeling time series data, capturing both short-term fluctuations and long-term trends.

## **1.2 DEEP LEARNING IN FINANCE**

Deep learning, inspired by the human brain's structure, excels at modeling complex, non-linear relationships and processing large volumes of high-dimensional data. In finance, this capability is crucial for analyzing intricate market dynamics.

Traditional ML techniques often require extensive feature engineering and may not effectively capture hidden dependencies in financial time series data. In contrast, deep learning models like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and especially LSTMs can automatically learn hierarchical features from raw inputs, eliminating the need for manual feature extraction.

### **1.2.1 Long Short-Term Memory**

Long Short-Term Memory (LSTM) networks are a powerful variant of Recurrent Neural Networks (RNNs) designed to learn from sequential data by addressing the limitations of traditional RNNs, such as vanishing gradients. What makes LSTM unique is its ability to retain important information over long periods through a memory cell regulated by three key gates:

- **Forget Gate:** Determines which parts of the previous information should be discarded from the memory cell.
- **Input Gate:** Decides which new information should be added to the memory.
- **Output Gate:** Controls what part of the stored information is passed to the next time step.

These gates use activation functions (sigmoid and tanh) to selectively update the memory, allowing the network to capture both short-term variations and long-term trends in data.

In stock market prediction, LSTMs are especially useful because they can learn patterns from historical stock prices and technical indicators over time. They are effective for trend prediction, volatility modeling, and identifying potential market turning points. When combined with other models like TCNs, their forecasting accuracy improves further, although they require more computational resources and careful hyperparameter tuning.

### **1.2.2 Temporal Convolutional Networks**

Temporal Convolutional Networks (TCNs) are designed for time series data and offer a powerful alternative to RNNs and LSTMs. They use causal convolutions to ensure predictions depend only on past and present inputs, preserving temporal order. Additionally, dilated convolutions allow the model to capture long-range dependencies efficiently by expanding the receptive field without adding more layers.

Unlike RNNs, TCNs process sequences in parallel, making them faster and more scalable. They also maintain stable gradients, avoiding vanishing or exploding issues during training. In stock market prediction, TCNs are valuable for detecting trends, local patterns, and sharp market movements. When combined with LSTMs in a hybrid model, TCNs extract short-term features while LSTMs handle long-term memory, resulting in more accurate and robust forecasts.

## 1.3 HYBRID MODELS

Hybrid LSTM-TCN models combine the strengths of two powerful deep learning architectures to improve stock market forecasting. LSTMs excel at learning long-term dependencies using memory cells, while TCNs capture short-term patterns efficiently through parallel processing and dilated convolutions. By integrating both, hybrid models handle the complex nature of financial data—capturing both macro-trends and micro-fluctuations.

In this system, TCN layers first extract short-term temporal features, which are then passed to LSTM layers to model broader time dependencies. This layered approach improves prediction accuracy and generalization, especially in volatile markets. Such models are highly suitable for financial time series as they adapt well to the non-linear, noisy, and multi-scale characteristics of stock data.

## 1.4 OBJECTIVE

The primary goal of this project is to develop an effective and user-friendly stock price prediction system by leveraging advanced deep learning techniques and dynamic feature selection methods. The specific objectives are:

1. **Develop a Hybrid LSTM-TCN Model:** Combine LSTM (Long Short-Term Memory) and TCN (Temporal Convolutional Networks) to capture both short-term and long-term stock price trends.
2. **Use Sliding Window with Technical Indicators:** Break down past stock data into smaller chunks (windows) and include important technical indicators like Moving Averages, RSI (Relative Strength Index), and Bollinger Bands to improve the model's predictions.
3. **Create a Web Application:** Build a simple web interface using React and Django, where users can input stock data, view predictions, and analyze trends in real-time.

## **CHAPTER 2**

### **LITERATURE SURVEY**

Bacco et al. [1] proposed a hybrid model combining LSTM and tweet sentiment analysis for predicting stock movements during periods of uncertainty. The model demonstrated improved accuracy in volatile markets, particularly for North American and European banks. However, its prediction quality declined when tweet data was sparse or biased.

El Mahjouby et al. [2] explored various machine learning and deep learning models for market prediction. Their models performed well across several metrics and adapted effectively to different scenarios. Nevertheless, key challenges included overfitting and poor generalization in unseen conditions.

Hung et al. [3] developed a real-time intraday trading system that leveraged market activity and machine learning. The system effectively tracked short-term trends and generated trade signals. However, it struggled to adapt to abrupt market changes and unforeseen events.

Patel et al. [4] presented a hybrid approach that combined statistical and machine learning methods for stock forecasting. The model enhanced both accuracy and interpretability. Nonetheless, its performance declined when dealing with low-liquidity or highly volatile stocks.

Patel et al. [5] also proposed a predictive model for the Indian stock market, integrating stock forecasting with portfolio optimization. This approach improved investment decision-making using machine learning techniques. However, its complex structure and computational demands limited its suitability for real-time applications.

Tian et al. [6] introduced a Graph Evolution Recurrent Unit (GERU) to model dynamic dependencies in financial data. This method effectively tracked the evolving relationships between stocks, though it was resource-intensive and challenging to scale for large datasets.

Chiong et al. [7] developed an ensemble model that combined sentiment analysis with the sliding window method. The model demonstrated improved prediction stability and robustness against market fluctuations. However, its performance heavily relied on the availability of consistent and reliable sentiment input.

Hsu et al. [8] proposed FinGAT, a Financial Graph Attention Network for ranking profitable stocks. It utilized structured relationships within financial data to enhance prediction performance. Yet, its accuracy declined when the financial graph data was noisy or incomplete.



## CHAPTER 3

### PROPOSED SYSTEM

#### 3.1 ARCHITECTURE OVERVIEW

The proposed system architecture is designed to facilitate accurate and efficient stock price prediction by integrating advanced deep learning models with a responsive web interface for real-time deployment. The architecture is divided into three key components:

##### 3.1.1 Data Processing Pipeline

The system begins with a robust data preprocessing module, which ingests historical stock price data along with technical indicators and other relevant financial metrics. The raw data undergoes several preprocessing steps, including:

- Data cleaning to handle missing values and outliers.
- Normalization or scaling to standardize numerical inputs.
- Feature engineering, where technical indicators such as RSI, MACD, Bollinger Bands, and ATR are computed.
- Feature selection using the Random Forest Feature Permutation (RF2P) method to dynamically identify the most informative features for the prediction task.

This step ensures that the input to the model is clean, relevant, and optimized for learning.

### 3.1.2 Hybrid Model: TCN + LSTM

At the core of the system lies the hybrid deep learning model, which combines the strengths of Temporal Convolutional Networks (TCN) and Long Short-Term Memory (LSTM) networks in a sequential pipeline:

- The TCN module processes the input sequence using dilated causal convolutions, effectively capturing short-term dependencies, local trends, and multi-scale temporal features.
- The output from the TCN is then passed to the LSTM layer, which is responsible for modeling long-term temporal dependencies and learning sequential trends across the time series.
- The combined architecture allows the model to benefit from both parallel processing and deep memory retention, enhancing its ability to forecast complex and volatile stock movements.

The final output of the model is a predicted stock price (or directional movement), which is then passed to the deployment module.

### 3.1.3 Real-Time Prediction and Web Application

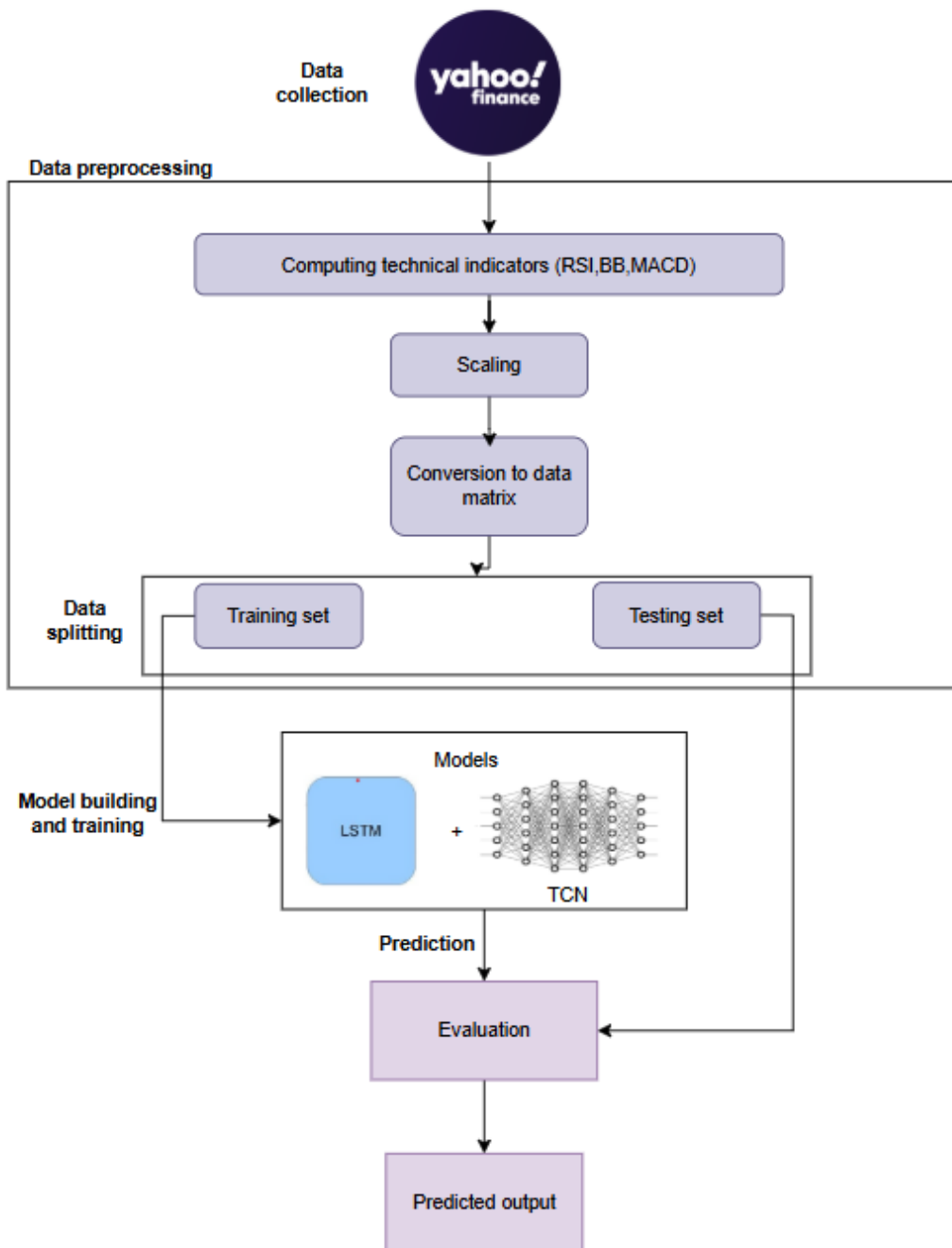
The final component of the architecture is the deployment layer, which connects the trained model to an interactive user interface. This is implemented as a full-stack web application with:

- Backend: A Django-based RESTful API that serves predictions by receiving input from the frontend and returning results generated by the hybrid model.
- Frontend: A React.js interface that allows users to input stock symbols, timeframes, and view real-time prediction results in a clean, intuitive dashboard.

- **Real-Time Interaction:** Users can visualize predicted trends, compare them with actual historical data, and monitor market forecasts dynamically.

This end-to-end architecture provides a seamless pipeline from raw data ingestion to user-facing predictions, combining machine learning, automation, and real-time analytics into a unified system.

## ARCHITECTURE DIAGRAM



**Figure 3.1 Architecture Diagram**

Figure 3.1 Working of stock market prediction system is explained in the architecture diagram.

## 3.2 DATA COLLECTION

Data collection forms the foundation of any stock market prediction system, as the quality and relevance of the input data directly impact the accuracy and performance of the model. In this project, historical stock price data is obtained using publicly available financial data APIs such as Yahoo Finance and Alpha Vantage, which provide reliable and timely access to a wide range of financial market information.

### Data Sources and Access Tools

- Yahoo Finance API (via yfinance Python library): Offers a convenient and efficient way to retrieve historical price data, including Open, High, Low, Close (OHLC), Adjusted Close, and Volume values. It also allows for easy filtering by date range and stock ticker symbol.

### Dataset Structure

The core dataset typically includes the following key features for each time step (e.g., daily or hourly interval):

- Open Price – The price at which a stock begins trading during a specific period.
- High Price – The highest traded price during the period.
- Low Price – The lowest traded price during the period.
- Close Price – The final trading price at the end of the period.
- Volume – The total number of shares traded during the period.

In addition to these basic financial metrics, a set of technical indicators is computed to enhance the feature space and capture underlying market dynamics:

- **Relative Strength Index** : Measures the speed and change of price movements, useful for identifying overbought or oversold conditions.
- **Moving Average Convergence Divergence**: A trend-following momentum indicator showing the relationship between two moving averages.
- **Bollinger Bands**: Represent price volatility by placing bands above and below a moving average.
- **Average True Range** : Reflects market volatility by measuring the degree of price movement over time.

These indicators are calculated using Python libraries such as pandas, ta (technical analysis library), and numpy.

### **Data Format and Storage**

- Data is typically retrieved and stored in CSV format, allowing for easy manipulation and reproducibility.
- The pandas library is used extensively to load, clean, transform, and merge datasets.
- Time-based indexing ensures proper alignment of time series data for both raw price and derived indicators.

By collecting and organizing data in this structured manner, the system ensures that the model receives rich, informative inputs that capture both price behavior and underlying market sentiment—critical components for making accurate predictions in a dynamic financial environment.

### 3.3 PREPROCESSING AND FEATURE ENGINEERING

Proper data preprocessing and feature engineering are crucial steps in ensuring that the model receives the most relevant and useful information for prediction. The goal of this section is to transform raw stock market data into a format that can be efficiently used by the hybrid LSTM-TCN model, as well as to improve the accuracy and robustness of the model through careful feature creation and selection.

#### Key Preprocessing Steps

- **Renaming Columns and Formatting Dates:**
  - The first step is to standardize the column names to ensure consistency across datasets (e.g., renaming columns like "Adj Close" to "Close").
  - The 'Date' column is parsed and converted to the datetime format to enable easier time-based indexing and manipulation.
- **Setting the 'Date' Column as Index:**
  - The 'Date' column is set as the index of the dataset. This allows for proper time-based slicing and ensures that the temporal order of the data is preserved, which is essential for time series forecasting.
- **Normalization of Stock Prices:**
  - MinMaxScaler is used to normalize stock prices and technical indicators. This scaling technique transforms the values into a range of [0, 1], preventing features with larger numerical ranges from dominating the model training process. Normalization is particularly important when combining features with different scales, ensuring the model treats all features equally.

- **Calculation of Technical Indicators:**

- Several popular **technical indicators** are calculated to capture market trends, volatility, and momentum:
  - **Relative Strength Index** : Helps identify overbought or oversold conditions in the market.
  - **Moving Average Convergence Divergence**: Identifies trend reversals by measuring the difference between short-term and long-term moving averages.
  - **Bollinger Bands**: Provides insights into price volatility by placing bands around a moving average.
  - **Average True Range**: Measures market volatility, indicating the degree of price movement.
- These indicators are essential for providing context beyond raw price movements, helping the model identify market conditions like overbought/oversold levels and potential reversals.

- **Sliding Window Approach for Sequence Generation:**

- To transform the time series data into an appropriate format for the hybrid model, a **sliding window approach** is used to create input sequences.
- For example, a **60-day window** is used to create sequences where the model will learn patterns from the past 60 days of stock data to predict the price for the following day.
- This ensures that the model receives input data in a temporal sequence, preserving the time-dependent nature of the financial data.



## Feature Engineering

- **Feature Calculation and Inclusion:**
  - Alongside the basic stock prices (Open, High, Low, Close, Volume), the key technical indicators (RSI, MACD, Bollinger Bands, ATR) are included as features. These features provide the model with additional information about market conditions, such as momentum, volatility, and trend strength.
- **Sequence Transformation for Model Input:**
  - Time series data is transformed into sequences of input features using the sliding window technique. Each window contains a sequence of data (e.g., stock prices, technical indicators) for the last 60 days, which is used as input for training the hybrid model.
  - These sequences are then fed into the TCN-LSTM hybrid model, with each sequence containing the necessary contextual data for the model to make accurate predictions.
- By performing these preprocessing and feature engineering steps, the data is appropriately transformed to maximize the effectiveness of the deep learning model. The inclusion of relevant technical indicators, along with careful feature selection, ensures that the model can learn important patterns from both historical prices and derived market metrics.

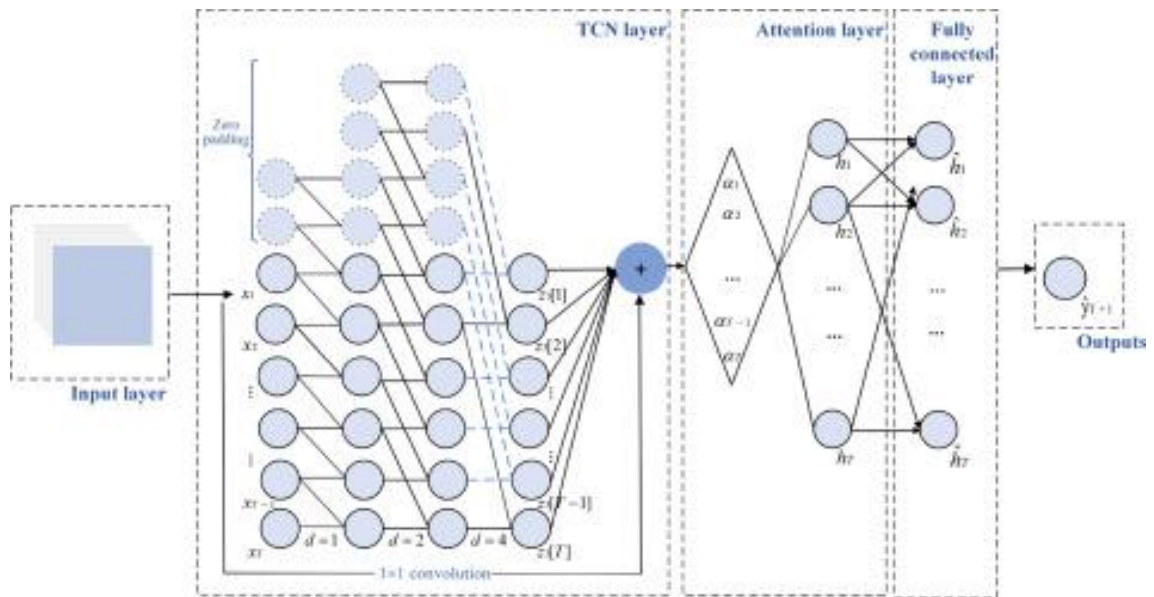
### 3.4 MODEL ARCHITECTURE

The hybrid model architecture combines Temporal Convolutional Networks (TCNs) for extracting temporal features and Long Short-Term Memory (LSTM) networks for capturing sequential dependencies. This two-stage architecture allows the model to learn both short-term local patterns and long-term trends in stock price data, making it well-suited for the complexity of financial time series prediction.

## TCN Block:

The first stage of the model is the Temporal Convolutional Network (TCN) block, which processes time-series data to extract high-level temporal features. Figure 3.2 illustrates this stage, where the input consists of time-series sequences of technical indicators (e.g., RSI, MACD, Bollinger Bands, ATR), each sequence representing a fixed time window (e.g., 60 days). These sequences are preprocessed and normalized, providing the model with rich, relevant information for prediction.

- Layers:
  - TemporalConvNet (TCN): The core of the TCN block is the Temporal Convolutional Network, which uses causal convolutions to ensure that predictions are based only on past and current data, preserving temporal causality. Dilated convolutions in the TCN allow for an exponentially expanding receptive field, enabling the model to capture dependencies across multiple time scales.
  - Dropout: A dropout layer is applied after the TCN to prevent overfitting by randomly setting a fraction of input units to zero at each update during training.
- Output: The output of the TCN block is a high-level temporal feature map. This feature map contains the learned temporal patterns representing the underlying structure of stock price movements across time, which is passed to the next stage (the LSTM block).



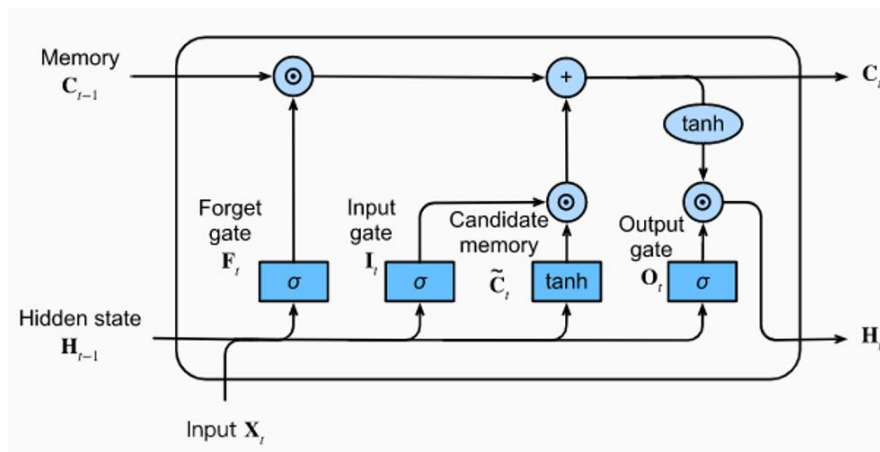
**Figure 3.2 TCN Model**

### **LSTM Block:**

The second stage of the model is the LSTM block, which takes the output from the TCN block and learns the long-term dependencies inherent in the data. Figure 3.3 illustrates this stage, where the input from the TCN block is reshaped into a format compatible with the LSTM, typically a 3D array of shape (batch size, time steps, features). This ensures the data is correctly formatted to feed into the LSTM layers.

- **Layers:**
  - **LSTM(100):** The first LSTM layer consists of 100 units and is responsible for learning long-term sequential dependencies in the data. The LSTM cell structure is particularly suited for this task, as it allows the model to retain and forget information over long sequences.
  - **Dropout:** A dropout layer is applied after the first LSTM to further prevent overfitting and encourage the model to generalize better on unseen data.

- LSTM(50): A second LSTM layer with 50 units further refines the learned features and captures more abstract sequential dependencies.
  - Dense(1): The final output layer is a dense layer with a single unit, which produces the predicted stock price (or directional movement) for the next time step.
- Output: The output of the LSTM block is the predicted stock price (or its movement), based on the learned temporal patterns from both the TCN and LSTM layers.



**Figure 3.3 LSTM Model**

### Training Setup:

- **Loss Function:** The model is trained using Mean Squared Error (MSE) as the loss function, which is well-suited for regression tasks like stock price prediction, where the goal is to minimize the difference between predicted and actual values.
- **Optimizer:** The Adam optimizer is used for optimization, as it adapts the learning rate during training and is particularly effective for deep learning tasks.

- **Hyperparameters:**

- Batch Size: Set to 32, balancing the model's ability to generalize and train efficiently.
- Epochs: The model is trained for 50 to 100 epochs, depending on convergence and validation performance.

### **3.5 HYBRID MODEL IMPLEMENTATION**

The hybrid TCN-LSTM model is implemented in a two-phase training process, where the TCN and LSTM components are trained sequentially. This approach allows each model to focus on specific aspects of the data and helps achieve better generalization, leading to more accurate stock price predictions.

#### **Phase 1: TCN Model Training**

- Input: The model receives preprocessed time-series data, including stock prices and technical indicators (e.g., RSI, MACD, Bollinger Bands, ATR), in the form of sequences. These sequences are created using the sliding window approach (e.g., 60-day window).
- TCN Model:
  - The TCN block focuses on learning local, short-term trends by applying causal convolutions with dilated filters.
  - This enables the model to learn short-range temporal dependencies within the stock data and identify immediate patterns or fluctuations.
  - The output of the TCN block is a high-level feature map, which is a transformed representation of the input sequence, capturing the most important short-term patterns.
- Training:

- The TCN is trained independently using the Mean Squared Error (MSE) loss function, and Adam optimizer is used to update the weights during training.
- The TCN model is trained on a smaller number of epochs (typically 20–30 epochs) due to its relatively faster convergence.

## **Phase 2: LSTM Model Training**

- Input: The output from the TCN block, which is the high-level temporal feature map, is reshaped into the required format (3D tensor) to be fed into the LSTM network.
- LSTM Model:
  - The LSTM block learns the long-term dependencies in the data, capturing trends and patterns that span across the entire sequence.
  - The first LSTM layer with 100 units processes the TCN output and learns the deeper sequential relationships in the data.
  - The second LSTM layer with 50 units further refines the learning, allowing the model to make better predictions based on long-term trends.
  - Finally, a dense layer with 1 unit generates the predicted stock price (or its movement) as the output.
- Training:
  - The LSTM model is trained using the same loss function (MSE) and optimizer (Adam), with the output from the TCN model being treated as the new input.
  - The training is carried out for 50 to 100 epochs, allowing the LSTM to learn the sequential dependencies effectively.

## 3.6 EVALUATION METRICS

### Mean Squared Error

MSE measures the average of the squared differences between the predicted and actual values. It heavily penalizes larger errors, making it useful for detecting significant prediction deviations. A lower MSE indicates better model accuracy.

### Root Mean Squared Error

RMSE is the square root of MSE and represents the average prediction error in the same units as the target variable (e.g., stock price). It is more interpretable than MSE and also sensitive to large errors.

### Mean Absolute Error

MAE calculates the average of the absolute differences between the predicted and actual values. It provides a straightforward measure of error magnitude and is less sensitive to outliers compared to MSE and RMSE.

### R<sup>2</sup> Score

The R<sup>2</sup> score indicates how well the model explains the variance in the target variable. A value of 1 means perfect predictions, while a value of 0 means the model performs no better than predicting the mean.

**Table 3.1 Evaluation Metrics**

| <b>Evaluation Metrics</b> | <b>Formulas</b>   |
|---------------------------|---|
| Mean Squared Error        | $\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$                         |
| Root Mean Squared Error   | $\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$                 |
| Mean Absolute Error       | $\text{MAE} = \frac{1}{n} \sum_{i=1}^n  y_i - \hat{y}_i $                           |
| R <sup>2</sup> Score      | $R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$ |

Table 3.1 shows the evaluation metrics used to measure the model's performance. It includes Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R<sup>2</sup> Score, which help in understanding how accurate the model's predictions are.



## **CHAPTER - 4**

### **IMPLEMENTATION**

#### **4.1 PLATFORM USED**

This project utilized a combination of platforms, frameworks, and tools to build and deploy an end-to-end stock price prediction system integrating deep learning and a web-based interface. Below is a detailed explanation of the platforms used:

##### **4.1.1 Google Colab**

Google Colab is a cloud-based Jupyter Notebook environment that lets users write and run Python code in a browser. It offers free access to GPUs and TPUs, making it ideal for deep learning. Colab also integrates with Google Drive for easy sharing and storage of code and data.

##### **4.1.2 Django**

Django, a high-level Python web framework, was used to build the backend of the web application. It handles:

- REST API creation for sending and receiving stock data and prediction results
- Authentication systems for users and administrators
- Integration with the MySQL database for storing login logs, stock details, and prediction data

##### **4.1.3. React**

React was used to create a responsive and dynamic frontend interface. The frontend displays:

- Stock charts and prediction results

- Admin dashboards for managing stocks and viewing login logs
- User and admin login and registration forms

React's component-based structure and seamless integration with REST APIs made it ideal for building an interactive UI for the stock prediction platform.

#### 4.1.4. MySQL

MySQL was used as the backend database for storing all application data. This includes:

- Historical stock information and prediction metadata
- Admin and user login logs
- Chart image paths and other related data

Django's ORM (Object-Relational Mapping) makes database interaction with MySQL efficient and secure.

## 4.2 TOOLS USED

**Table 4.1 Tools Used**

| Tools                | Purpose   |
|----------------------|---|
| Python               | The primary programming language for implementing machine learning models |
| TensorFlow/Keras     | For building and training both LSTM and TCN models                        |
| NumPy & Pandas       | For data manipulation and preprocessing                                   |
| Matplotlib & Seaborn | For data visualization and analysis                                       |
| Scikit-learn         | For data normalization, splitting, and evaluation metrics                 |
| Yahoo Finance        | For fetching stock market data  |
| Jupyter Notebook     | For coding, experimentation, and visualization                            |

Table 4.1 lists the tools used in the project along with their respective purposes. These tools include Python for programming, TensorFlow/Keras for building models, and other libraries like NumPy, Pandas, and Matplotlib for data manipulation, preprocessing, and visualization.

### 4.3 PSEUDO CODE

The hybrid LSTM-TCN model combines LSTM's ability to capture long-term dependencies with TCN's strength in learning short-term patterns. It uses LSTM to model sequential stock data and TCN to capture temporal features. This combination enhances prediction accuracy by leveraging both long- and short-term dependencies.

#### 4.3.1 Data Collection

1. Begin
2. Define stock\_ticker  $\leftarrow$  "AAPL"
3. Define start\_date  $\leftarrow$  "2015-01-01"
4. Define end\_date  $\leftarrow$  "2025-03-10"
5. Call function download\_stock\_data(stock\_ticker, start\_date, end\_date)
6. Function download\_stock\_data(ticker, start, end):
7.   Import yfinance as yf
8.   Import pandas as pd
9.   data  $\leftarrow$  yf.download(ticker, start=start, end=end)
10.   filename  $\leftarrow$  "{ticker}\_stock\_data.xlsx"
11.   Save data to Excel file using openpyxl engine
12.   Print "Stock data saved as {filename}"

13. Return data

14. End

### **4.3.2 Data Preprocessing**

1. Begin

2. Load Excel file → raw\_data

3. Parse 'Date' column as datetime

4. Set 'Date' as index

5. Sort data by Date in ascending order

6. Handle missing values:

a. Fill or drop missing values from raw\_data

7. Calculate technical indicators:

a.  $RSI \leftarrow \text{compute\_RSI}(\text{raw\_data}['Close'])$

b.  $MACD, MACD\_Signal \leftarrow \text{compute\_MACD}(\text{raw\_data}['Close'])$

c.  $BB\_Upper, BB\_Lower \leftarrow \text{compute\_Bollinger\_Bands}(\text{raw\_data}['Close'])$

d.  $ATR \leftarrow \text{compute\_ATR}(\text{raw\_data})$

8. Append all calculated indicators as new columns in raw\_data

9.  $\text{feature\_columns} \leftarrow ['Close', 'Open', 'High', 'Low', 'Volume', 'RSI', 'MACD', 'MACD\_Signal', 'BB\_Upper', 'BB\_Lower', 'ATR']$

10. Normalize feature\_columns using MinMaxScaler

a. Fit scaler on feature\_columns

b. Transform feature\_columns with fitted scaler

11. Save the preprocessed data for model input

12. End

### 4.3.3 Model Training

1. Begin

2. Load preprocessed data  $\rightarrow$  processed\_data

3. Define lookback\_window (e.g., 60 days)

4. Create input sequences:

a. For  $i$  from lookback\_window to  $\text{len}(\text{processed\_data}) - \text{prediction\_horizon}$ :

i.  $X \leftarrow \text{processed\_data}[i - \text{lookback\_window} : i]$

ii.  $y \leftarrow \text{processed\_data}[i : i + \text{prediction\_horizon}][\text{'Close'}]$

iii. Append (X, y) to dataset

5. Split dataset into training and validation sets

6. Define model architecture:

a. Input Layer

b. Temporal Convolutional Network (TCN) layers

c. LSTM Layer

d. Dense Layer (output: prediction\_horizon)

7. Compile the model:

a. Loss Function  $\leftarrow$  Mean Squared Error (MSE)

b. Optimizer  $\leftarrow$  Adam

8. Train the model:

a. Set epochs, batch\_size

b. Fit model on training data

c. Validate using validation set

9. Save trained model

10. End

#### **4.3.4 Future Predictions**

1. Begin

2. Load the trained model

3. Load latest processed stock data

a. Select the last 'lookback\_window' days from the data  $\rightarrow$  input\_sequence

4. Initialize future\_predictions  $\leftarrow$  empty list

5. For day in range(0, number\_of\_days\_to\_predict):

a. Predict next day using model:

i. prediction  $\leftarrow$  model.predict(input\_sequence)

b. Append prediction to future\_predictions

c. Update input\_sequence:

i. Append prediction to input\_sequence

ii. Remove the oldest data point to maintain window size

iii. Recalculate technical indicators if required (e.g., RSI, MACD)

6. Inverse transform predictions to original scale (if scaled)

7. Plot or save future\_predictions

8. End

### 4.3.5 Smart Money Entries

1. Begin
2. Load historical stock data with columns: Open, High, Low, Close, Volume
3. Calculate Technical Indicators:
  - a. VWAP  $\leftarrow$  Volume Weighted Average Price
  - b. OBV  $\leftarrow$  On-Balance Volume
  - c. ADL  $\leftarrow$  Accumulation/Distribution Line
  - d. SMA, EMA, RSI (optional for trend confirmation)
4. Detect Smart Money Entry Conditions:

For each day in the dataset:

  - a. If:
    - i. Price is above VWAP
    - ii. OBV is increasing
    - iii. ADL is increasing
    - iv. Volume is significantly above average
    - v. Recent demand zone formed or price bounced from support
  - b. Then:

$\rightarrow$  Mark as "Potential Smart Money Entry"
5. Optionally:
  - a. Visualize entries on candlestick chart with indicators
  - b. Save flagged entries for reporting
6. End

### 4.3.6 Trend of Stock

1. Begin
2. Load historical stock data with columns: Close, Volume, etc.
3. Calculate Technical Indicators:

- a.  $SMA_{50} \leftarrow$  50-day Simple Moving Average
- b.  $SMA_{200} \leftarrow$  200-day Simple Moving Average
- c.  $RSI \leftarrow$  Relative Strength Index
- d.  $MACD \leftarrow$  Moving Average Convergence Divergence
- e. Volume Trend (optional)

4. Determine Trend Based on Rules:

For each day in dataset:

- a. If  $SMA_{50} > SMA_{200}$  AND  $RSI > 50$  AND  $MACD > MACD$  Signal:  
→ Mark trend as "Uptrend"
- b. Else If  $SMA_{50} < SMA_{200}$  AND  $RSI < 50$  AND  $MACD < MACD$  Signal:  
→ Mark trend as "Downtrend"
- c. Else:  
→ Mark trend as "Sideways / No clear trend"

5. Optionally:

- a. Plot trend labels on the stock chart
- b. Store trend labels for prediction or analysis

6. En



### 4.3.7 Frontend–Backend–Database Interaction

1. Begin

2. FRONTEND (React):

- Display UI (Login, Add Stock, View Chart)
- On user action → Send request to backend (API)

3. BACKEND (Django):

- Receive API request
- Validate data and user
- Perform logic (e.g., save stock, get prediction)
- Interact with MySQL (via Django ORM)

4. DATABASE (MySQL):

- Store/retrieve stock, user, or prediction data

5. BACKEND:

- Send response to frontend (success/data)

6. FRONTEND:

- Show result (chart, message, logs) in UI

7. End

## CHAPTER 5

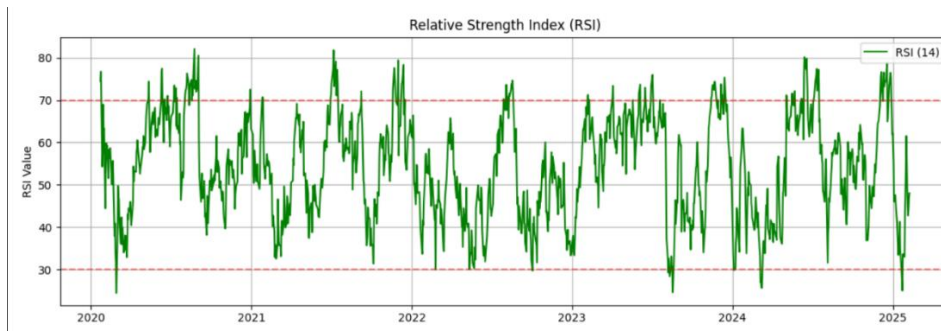
### RESULTS AND ANALYSIS

#### 5.1 VISUALIZATION OF PREDICTIONS

To further demonstrate the effectiveness of the models, visualizations were generated to compare actual vs predicted prices and show how well the models capture market trends

##### 5.1.1 Technical Indicators

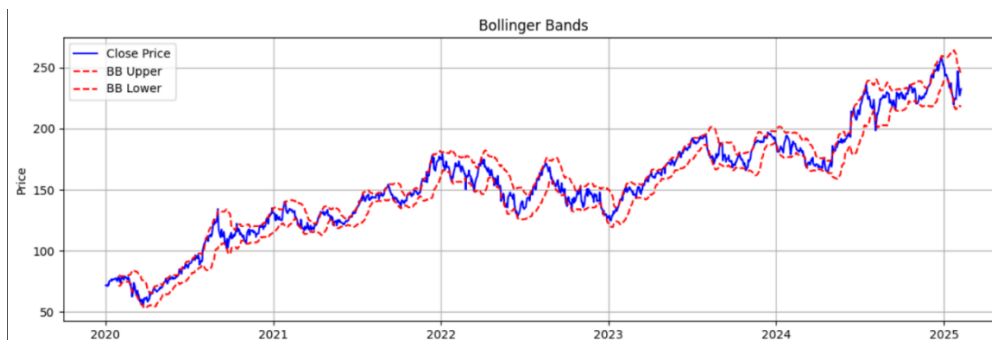
###### RSI



**Figure 5.1 RSI**

Figure 5.1 illustrates the Relative Strength Index , a momentum oscillator used to measure the speed and change of price movements in stock market analysis.

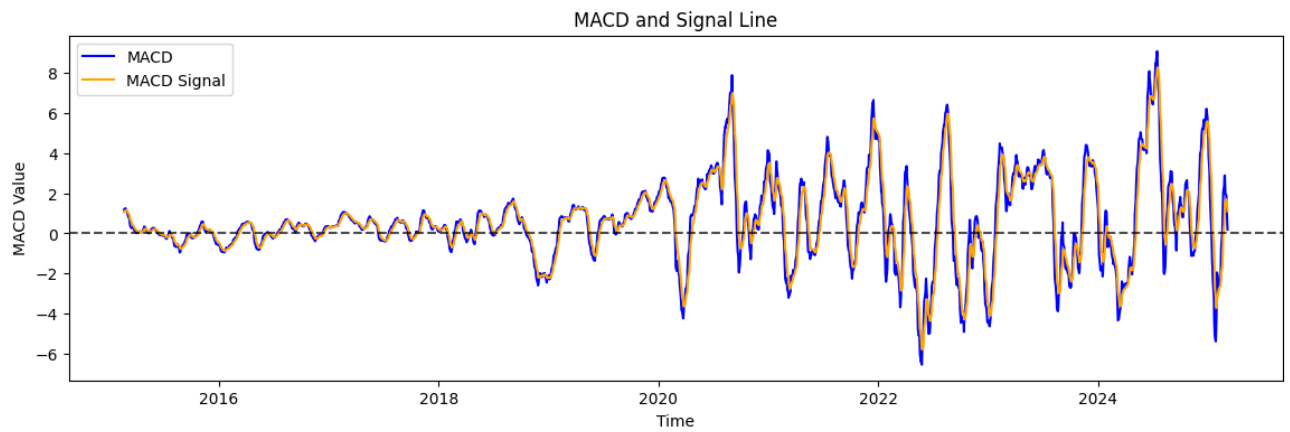
###### BOLLINGER BANDS



**Figure 5.2 Bollinger Bands**

Figure 5.2 shows the Bollinger Bands, consisting of a moving average and two standard deviation lines, used to identify overbought or oversold conditions.

## MACD

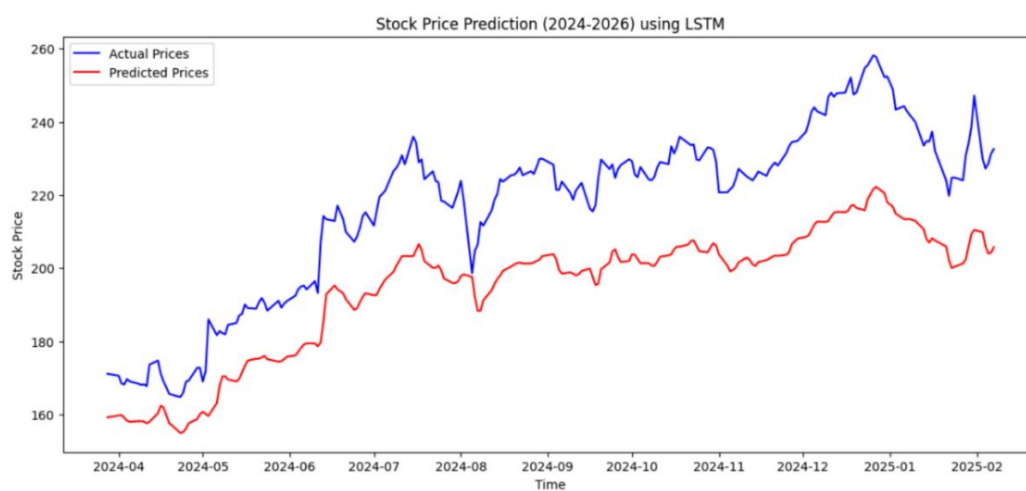


**Figure 5.3 MACD**

Figure 5.3 displays the MACD indicator, showing the difference between short-term and long-term moving averages to identify buy or sell signals.

### 5.1.2 Model Prediction for “AAPL” Stock

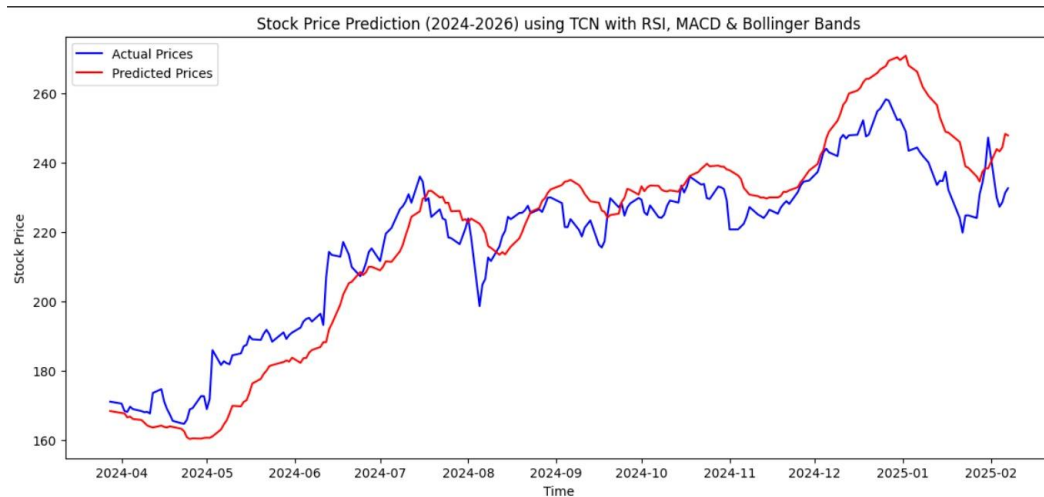
## LSTM



**Figure 5.4 LSTM Prediction**

Figure 5.4 displays the LSTM model's prediction for the stock "AAPL," illustrating how the model forecasts future stock prices. The graph compares the predicted values with actual stock prices to evaluate the model's performance.

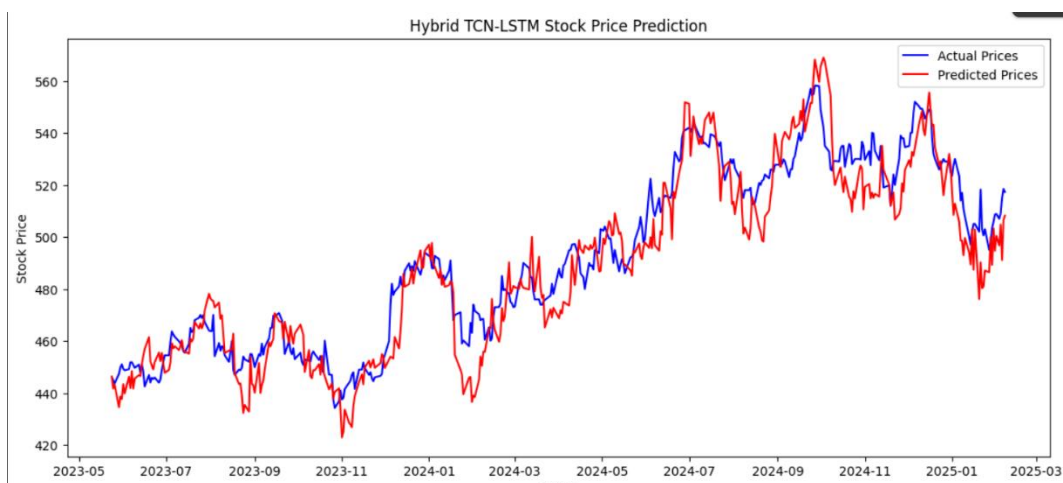
## TCN



**Figure 5.5 TCN Prediction**

Figure 5.5 shows the TCN model's prediction for stock prices, comparing the predicted values with the actual data to assess the model's accuracy.

## TCN – LSTM



**Figure 5.6 Hybrid Model Prediction**

Figure 5.6 displays the prediction results of the hybrid model, combining TCN and LSTM. The graph demonstrates how the hybrid model enhances stock price forecasting by leveraging the strengths of both architectures.

## 5.2 MODEL COMPARISON

The performance of the LSTM, TCN, and Hybrid LSTM-TCN models was evaluated using standard regression metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and  $R^2$  Score.

The Hybrid model consistently outperformed the standalone models, indicating its superior ability to capture both short-term and long-term dependencies in stock price trends.

**Table 5.1 Model Comparison**

| <b>METRICS</b> | <b>LSTM</b> | <b>TCN</b> | <b>HYBRID</b> |
|----------------|-------------|------------|---------------|
| MSE            | 0.0042      | 0.0038     | 0.0029        |
| RMSE           | 0.0648      | 0.0616     | 0.0538        |
| MAE            | 0.517       | 0.051      | 0.0423        |
| R-Squared      | 0.89        | 0.91       | 0.94          |

Table 5.1 compares the performance of different models based on evaluation metrics. The hybrid model outperforms others, achieving the lowest MSE and RMSE, the smallest MAE, and an impressive 94%  $R^2$  score, indicating its high accuracy in stock price prediction.

**MSE & RMSE:** Hybrid model achieves the lowest values, indicating minimal prediction error.

**MAE:** Reflects the smallest average deviation from actual prices.

**R<sup>2</sup> Score:** The Hybrid model explains 94% of the variance, demonstrating excellent overall performance.

### 5.3 FUTURE STOCK PREDICITON

The Hybrid LSTM-TCN model was also evaluated for its ability to predict stock prices for the next 90 days. This was done using an iterative forecasting strategy, where each new prediction is based on the most recent data.

#### Approach

##### 1. Input Preparation

- The model uses the last 60 days of stock prices and technical indicators (e.g., RSI, MACD, Bollinger Bands) as input.

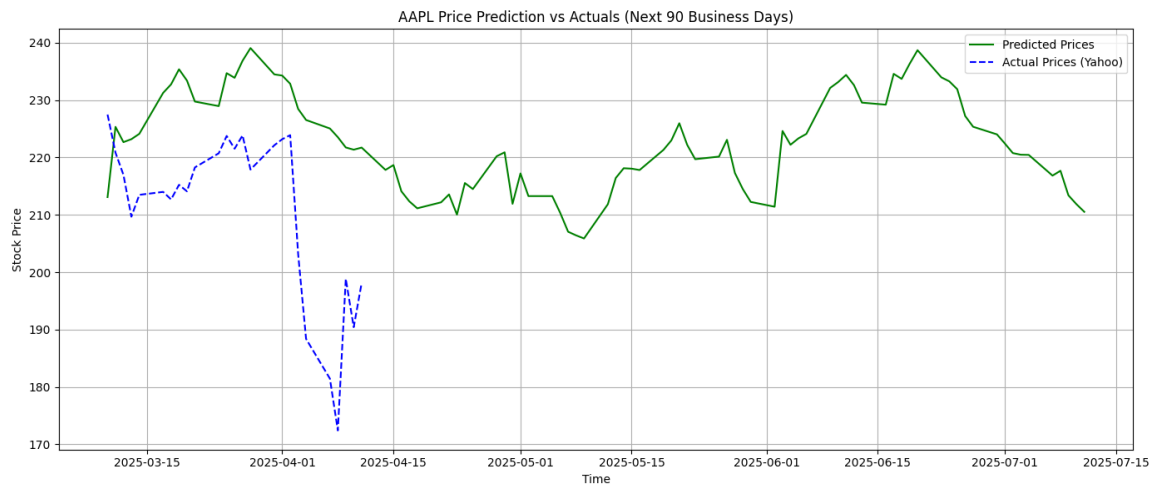
##### 2. Day-by-Day Forecasting

- The model predicts the price for the next day. This prediction is then added to the input sequence, and the model uses the updated data to predict the following day.
- This process is repeated for 90 days, generating a future price trend step by step.

##### 3. Forecast Visualization

- A graph is plotted showing the 90-day predicted stock prices, illustrating how the model captures both short-term fluctuations and long-term trends.

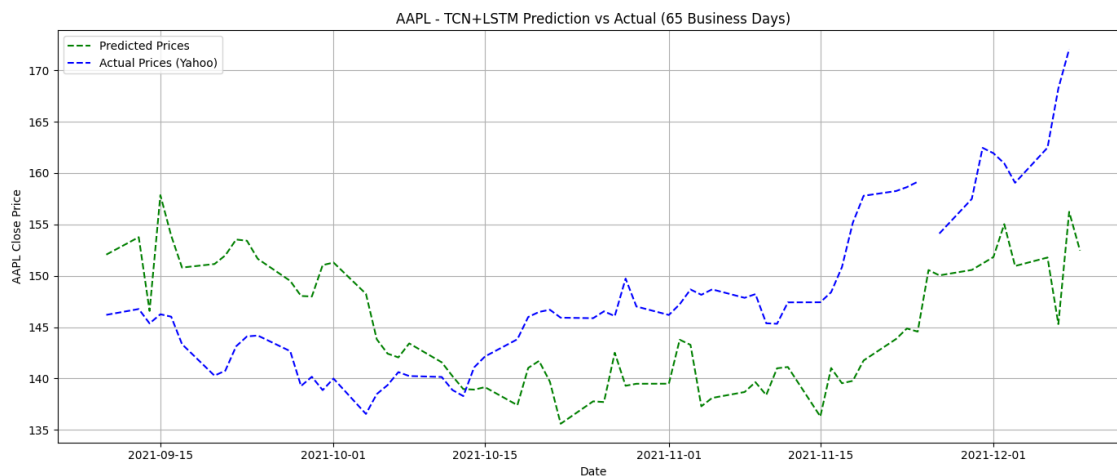
### 5.3.1 Future Stock Predictions for “AAPL” Stock



**Figure 5.7 Future Predictions**

Figure 5.7 Future Predictions shows the predicted stock price for Apple Inc. (AAPL) based on the model's forecasting capabilities, providing insight into future stock trends and potential price movements.

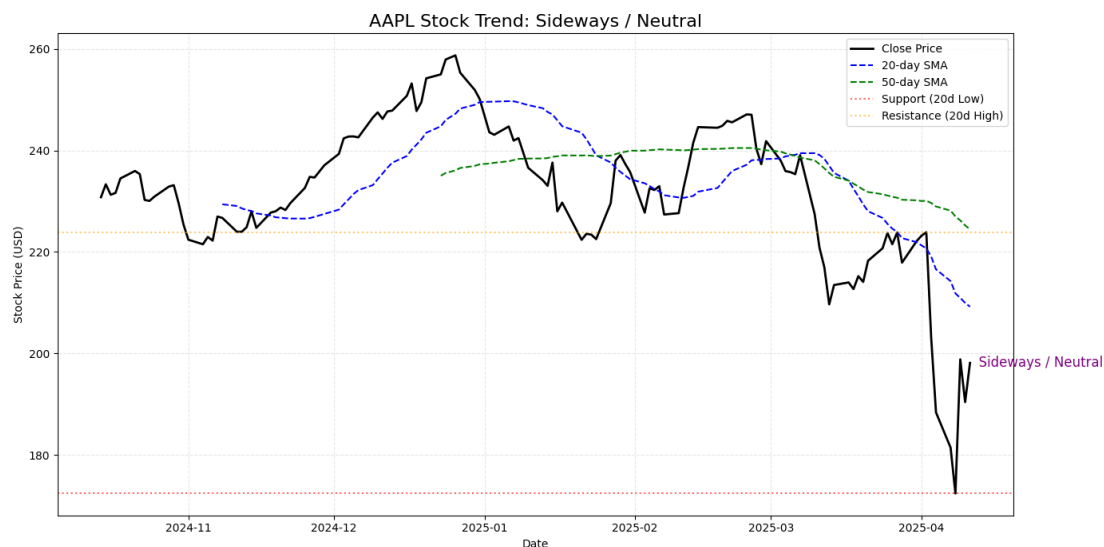
### 5.3.2 AAPL Stock Prediction During Covid



**Figure 5.8 Predictions on Covid**

Figure 5.8 Predictions on COVID illustrates the predicted AAPL stock price trends during the COVID-19 pandemic, highlighting the impact of the pandemic on market fluctuations and potential future trends.

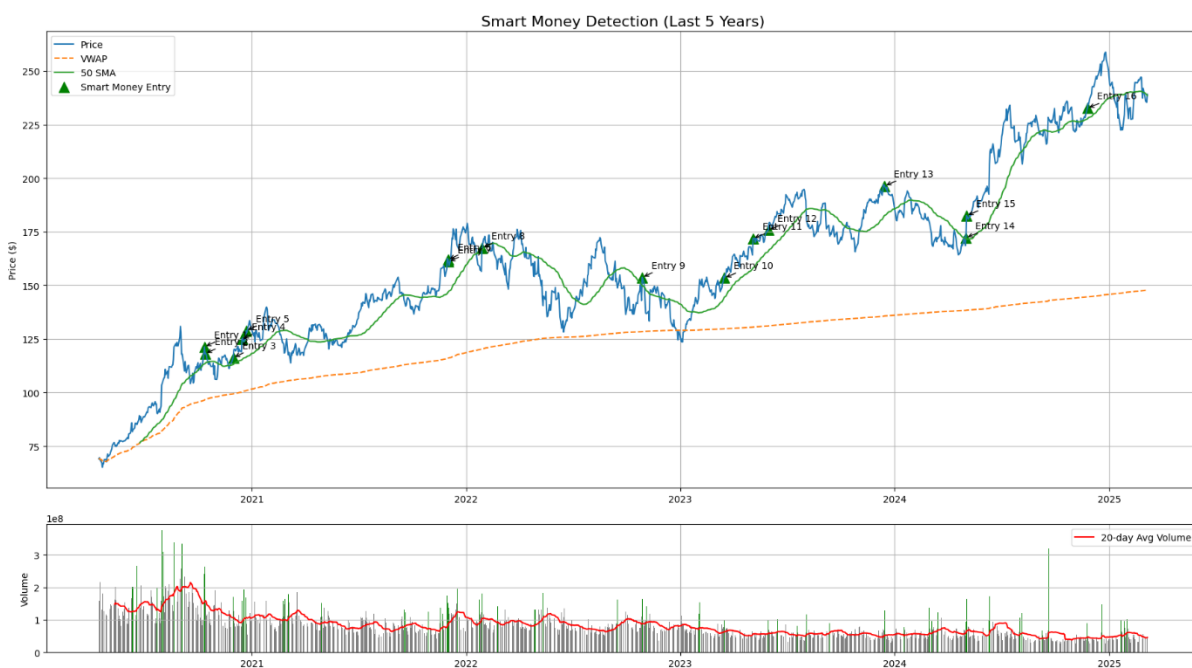
### 5.3.3 Stock Trend Prediction



**Figure 5.9 Stock Trend**

Figure 5.9 Stock Trend displays the historical stock price movements, showcasing the overall trend and fluctuations over a specified period, aiding in visual analysis of market behavior.

### 5.3.4 Smart Money Entries Prediction



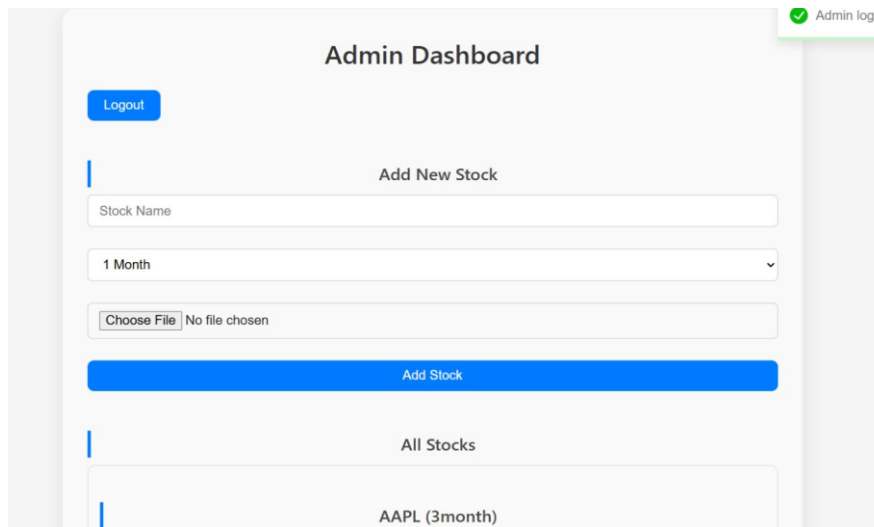
**Figure 5.10 Smart Money Entries**



Figure 5.10 Smart Money Entries illustrates the points at which institutional investors, or "smart money," enter the market, potentially indicating strong future trends or price movements.

## 5.4 STOCK MARKET WEBPAGE

### ADMIN DASHBOARD



The image shows a web interface for an "Admin Dashboard". At the top right, there is a green checkmark icon and the text "Admin login". Below this, the title "Admin Dashboard" is centered. On the left side, there is a blue button labeled "Logout". The main content area is divided into two sections. The first section is titled "Add New Stock" and contains a text input field for "Stock Name", a dropdown menu currently showing "1 Month", and a file upload area with a "Choose File" button and the text "No file chosen". Below these fields is a prominent blue button labeled "Add Stock". The second section is titled "All Stocks" and shows a single entry labeled "AAPL (3month)".

**Figure 5.11 Admin Dashboard**

Figure 5.11 Admin Dashboard displays the interface used by administrators to manage stock data, monitor predictions, and oversee system operations in the stock management system.

## CHAPTER 6

### CONCLUSION AND FUTURE WORKS

This project successfully developed a Hybrid Deep Learning Model by combining LSTM and TCN to enhance the accuracy of stock market prediction. The hybrid model outperformed the individual LSTM and TCN models by effectively capturing both short-term patterns and long-term trends in the stock data.

To further improve prediction performance, the model incorporated key technical indicators such as RSI, MACD, Bollinger Bands, and ATR. For forecasting, a sliding window method was used to make future predictions in a day-by-day manner.

Additionally, a full web-based application was developed using React for the frontend and Django for the backend. This platform provides:

- Real-time stock predictions
- Smart money entry detection
- Trend analysis of the stock

This system offers a practical and intelligent tool for traders and investors to make informed decisions in real time.

#### **Future Enhancements**

To further improve the system, the following upgrades can be explored:

##### **1. Sentiment Analysis**

- Add news and social media sentiment (e.g., Twitter, Reddit) to understand market emotions.

## **2. Macroeconomic Indicators**

- Include economic data like interest rates, inflation, or GDP to improve prediction accuracy.

## **3. Advanced Deep Learning Models**

- Try Transformer or attention-based models to help the system focus on the most relevant time periods.

## REFERENCES

- [1] Bacco, L., Petrosino, L., Arganese, D., Vollero, L., Papi, M. and Merone, M., 2024. Investigating stock prediction using LSTM networks and sentiment analysis of tweets under high uncertainty: A case study of North American and European banks. *IEEE Access*.
- [2] El Mahjouby, M., Bennani, M.T., Lamrini, M., El Far, M., Bossoufi, B. and Alghamdi, T.A., 2024. Predicting market performance using machine and deep learning techniques. *IEEE Access*.
- [3] Hung, M.C., Chen, A.P. and Yu, W.T., 2024. AI-driven intraday trading: Applying machine learning and market activity for enhanced decision support in financial markets. *IEEE Access*, 12, pp.12953-12962.
- [4] Patel, M., Jariwala, K. and Chattopadhyay, C., 2024. A Hybrid Relational Approach Towards Stock Price Prediction and Profitability. *IEEE Transactions on Artificial Intelligence*.
- [5] Patel, M., Jariwala, K. and Chattopadhyay, C., 2024. An Approach Toward Stock Market Prediction and Portfolio Optimization in Indian Financial Sectors. *IEEE Transactions on Computational Social Systems*.
- [6] Tian, H., Zhang, X., Zheng, X. and Zeng, D.D., 2023. Learning dynamic dependencies with graph evolution recurrent unit for stock predictions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(11), pp.6705-6717.

- [7] Chiong, R., Fan, Z., Hu, Z. and Dhakal, S., 2022. A novel ensemble learning approach for stock market prediction based on sentiment analysis and the sliding window method. *IEEE Transactions on Computational Social Systems*, 10(5), pp.2613-2623.
- [8] Hsu, Y.L., Tsai, Y.C. and Li, C.T., 2021. Fingat: Financial graph attention networks for recommending top-\$ k \$ k profitable stocks. *IEEE transactions on knowledge and data engineering*, 35(1), pp.469-481.