

A Mini Project with Seminar
On
SMART INVESTMENT PLANNER USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology
in
Computer Science and Engineering (Data science)
by

Abhinav Unnamatla	22241A6767
Jasti Amit	22241A6793
Medhansh Kotipalli	22241A6799

Under the Esteemed guidance of

Dr. Mamidi Kiran Kumar

Associate Professor



Department of Data Science

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Autonomous under JNTUH, Hyderabad)

Bachupally, Kukatpally, Hyderabad-500090

2024-2025



GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Hyderabad-500090

CERTIFICATE

This is to certify that the Mini Project with Seminar entitled “**SMART INVESTMENT PLANNER USING MACHINE LEARNING**” is submitted by **Abhinav Unnamatla (22241A6767)**, **Jasti Amit (22241A6793)** and **Medhansh Kotipalli (22241A6793)** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering (Data science) during the Academic Year 2024-2025.

Internal Guide

Dr. Mamidi Kiran Kumar

Head of the Department

Dr. S. Govinda Rao

External Examiner

ACKNOWLEDGEMENT

There are many people who helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we would like to express our deep gratitude towards our internal guide **Dr. Mamidi Kiran Kumar, Associate Professor**, Department of Data Science, for his support in the completion of our dissertation. We are thankful to our mini project coordinators **Mr. K. Mallikarjuna Raju**, Assistant Professor and **Ms. Niharika**, Assistant Professor, for their valuable suggestions and comments during this project period. We wish to express our sincere thanks to **Dr. S. Govinda Rao**, Head of the Department, and to our Principal **Dr. J. Praveen**, for providing the facilities to complete the dissertation. We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Abhinav Unnamatla (22241A6767)

Jasti Amit (22241A6793)

Medhansh Kotipalli (22241A6799)

DECLARATION

We hereby declare that the mini project titled “**SMART INVESTMENT PLANNER USING MACHINE LEARNING**” is the work done during the period from **16th January 2025** to **13th May 2025** and is submitted in the partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering (Data science) from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad). The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

Abhinav Unnamatla (22241A6767)

Jasti Amit (22241A6793)

Medhansh Kotipalli (22241A6799)

ABSTRACT

Stock market investment remains a challenging endeavour, largely due to its inherent volatility, lack of personalized insights, and the frequent misalignment of investment decisions with individual financial goals. Traditional analytical methods often fall short in offering the flexibility, depth, and real-time responsiveness required by modern investors. To address these limitations, this project proposes an AI-driven stock forecasting and recommendation system focused on the Bombay Stock Exchange (BSE), specifically targeting the top 50 listed companies. The system leverages deep learning, implemented using PyTorch, and is deployed through an interactive, user-friendly web interface developed in Streamlit. The core of the model utilizes a Temporal Convolutional Network (TCN), selected for its superior performance in capturing long-range dependencies and patterns in time-series data. Historical stock price data is processed and enhanced through a novel signal decomposition approach, which combines moving average smoothing with the Hilbert Transform to isolate trend and cyclical components. This preprocessing not only improves the model's ability to learn from meaningful signals but also reduces the impact of market noise. To further ensure realistic forecasting, the system integrates a spike suppression mechanism, which dynamically caps extreme predictions at a daily threshold of 2%, thereby avoiding erratic outputs that could mislead investment decisions. In addition to predictive analytics, the platform includes modules for real-time visualization, model evaluation, and personalized stock ranking, aligning recommendations with users' risk appetite, investment strategy, and budget. By combining powerful deep learning techniques with a seamless user experience, the proposed system empowers individual investors with data-driven, tailored, and reliable recommendations—enhancing their confidence and effectiveness in navigating the stock market.

LIST OF FIGURES

Figure No.	Figure Name	Page No.
3.3.1	Overall Project Flow Diagram	13
3.4.1	Architecture Diagram	14
3.6.1	Modules Connectivity Diagram	18
3.8.1	Data Flow Diagram	23
3.8.2	Use Case Diagram	25
4.2.1	Intrinsic Mode Functions	32
4.2.2	IMFs for first 20 epochs	33
4.2.3	IMFs for first 50 epochs	34
4.2.4	TCN vs ARIMA	35
4.2.5	TCN vs SARIMA	35
4.2.6	Recommendation of Top 5 Companies	38
4.2.7	Detailed Graph of a specific selected company	39

LIST OF TABLES

Table No.	Table Name	Page No.
2.2.1	Literature Review Summary Table	8
3.5.1	Software Requirements Specifications	15
3.5.2	Hardware Requirements Specifications	17
3.9	Test cases tables	26
4.2.1	Error Metric Comparison Across Models	38

LIST OF ACRONYMS

AI	Artificial Intelligence
ML	Machine Learning
LLM	Large Language Model
GPT	Generative Pre-trained Transformer
CPU	Central Processing Unit
RAM	Random Access Memory
UML	Unified Modelling Language
GRU	Gated Recurrent Unit
ARIMA	Auto Regressive Integrated Moving Average
SARIMA	Seasonal Auto Regressive Integrated Moving Average
RNN	Recurrent Neural Network
TCN	Temporal Convolutional Network
CNN	Convolutional Neural Network
BCO	Brain Storm Optimization
DL	Deep Learning
ML	Machine Learning
EMD	Empirical Mode Decomposition
HTT	Hilbert-Huang Transform
IF	Instantaneous Frequency
NN	Neural Network
DT	Decision Tree
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
RBF	Radial Basis Function
S&P 500	Standard & Poor's 500 Index

TABLE OF CONTENTS

Chapter No.	Chapter Name	Page No.
	Title	i
	Certificate	ii
	Acknowledgement	iii
	Declaration	iv
	Abstract	v
	List of Figures	vi
	List of Tables	vii
	List of Acronyms	viii
1	INTRODUCTION	1-4
	1.1 Introduction to the project	1
	1.2 Significance of the project	1
	1.3 Methodologies Adopted	2
	1.4 Organization of the chapters	4
2	LITERATURE SURVEY	5-11
	2.1 Summary of Existing Approaches	5
	2.2 Literature Review Summary Table	8
	2.3 Limitations of Current Methods	10
	2.4 Research Gaps	11
3	PROPOSED METHODS	12-30
	3.1 Problem Statement	12
	3.2 Objectives of the Project	12
	3.3 Project Flow Diagram	13
	3.4 Architecture Diagram	14
	3.5 Software and Hardware Requirements	15
	3.6 Modules Connectivity Diagram	18
	3.7 Requirements Engineering	19
	3.8 Analysis and Design	22
	3.9 Test Cases	26

4	RESULTS AND DISCUSSIONS	30-38
	4.1 Dataset Details	31
	4.2 Experimental Results	31
5	CONCLUSION AND FUTURE ENHANCEMENTS	
		40-41
	5.1 Conclusion	40
	5.2 Future Enhancements	41
	REFERENCES	42-43
	APPENDICES	44-45
	Sample Source Code	44
	Source Code Repository	45

CHAPTER 1

INTRODUCTION

1.1.Introduction to the Project

In the modern fast-paced financial setting, the stock market plays an important role in accumulating wealth, in making long-term investments, and in the growth of the economy. Picking the right stocks from hundreds of choices may seem daunting to everyday investors, though, when they're not backed by years of knowledge and countless hours spent studying trends. We've created a proprietary knowledge-based stock recommendation system relevant exclusively for companies listed at the Bombay Stock Exchange (BSE). By leveraging the power of artificial intelligence and deep learning, our system inspects the previous performance of stocks and provides smart data-driven prediction on future stock prices.

A core of the system is a strong deep learning model, a Temporal Convolutional Network (TCN). Unlike old techniques such as RNNs or LSTMs, TCNs are more efficient and accurate in processing time-series data – due to their capability to process sequences in parallel and remember patterns through a long period. To take it up another level we've included a custom signal decomposition step utilizing moving averages and the Hilbert Transform. This aids in the simplification of complex stock signals into less complex components to increase the accuracy of our forecasts. We've also ensured that the system is easy and natural to use. Based on Streamlit technology, the app allows users to work with a clear, friendly interface. Investors can define their preference; risk level, investment duration, budget, etc. and the system will do all the heavy lifting. It uses a custom scoring method to evaluate predicted returns and ranks the best 5 stocks of BSE which suit the user's profile best.

By combining smart algorithms, deep financial insight, and smooth user experience, this project seeks to help both novice and seasoned investors with a powerful asset they can rely on to make educated investment decisions – without becoming a stock market genius.

1.2.Significance of the Project

The stock market tends to look intimidating and scary to investors who are starting out. New investors may make bad choices and fail to take advantage of important opportunities by following what others say or guessing. The company intends to overcome this by offering investors a custom approach to investing with the help of information and algorithms.

Thanks to joining deep learning with forecasting for time series data, this system makes advanced financial analysis possible for people at all levels of finance expertise. It was decided that using a more advanced system would be better than the usual methods based on rules. The approach combines the TCN and makes use of moving averages and the Hilbert transform to separate signals. Using these tools, we are able to learn more about past trends and discover the less noticeable changes. Seeking various opinions about a company produces better guesses about how its stock will move, allowing investors to get ready. So, it allows users to

know what stocks to consider investing in with confidence.

Our intelligent stock recommendation system results from applying the best data science technologies and making the site user-friendly. These steps tell you how we solved the situation.

1.3 Methodologies Adopted

Data Collection and Preparation

We build our stock recommendation system on the strength of accurate and complete data collection. Using yfinance, we collected stock market data from Yahoo Finance by focusing on its key details such as open price, high price, low price, close price and trading volume. To achieve relevance and variety, we chose leading stocks listed on the NIFTY 100, SENSEX and NIFTY BANK indices. The collection of these indices reflects a large selection of different industries and includes well-traded and popular stocks. We noticed that some financial time series data was missing or incomplete as we performed the data extraction. We used interpolation to fill the gaps so that the data remained consistent with time. As soon as our data was complete, we scaled it and normalized it to make it ready for machine learning. Scale matters since it stops a single feature from determining the outcome because of its size and it accelerates the process of learning during training. The analysis and modeling done after relied heavily on the thorough and clean state of the dataset.

Signal Decomposition

A lot of the time, important trends are hidden in raw financial data because of short-term ups and downs. We used signal decomposition methods to help us improve the quality and results of our model. Because of this method, we could distinguish the patterns from the confusing data. We used the MACD indicator which many traders rely on to detect changes in stock prices' momentum by measuring the gap between short- and long-term moving averages. By looking at MACD, traders might discover when to enter the market to buy or sell and it can also clear up the general mood in the market. We also implemented the Hilbert Transform, a key mathematical method for uncovering both the phase and frequency properties of signals in signal processing. Using this transform allowed us to see brief cycles within the stock market that are not observable in the original graph. When we removed noise from the data and analyzed only the key trends, the model learned better and made more reliable forecasts.

Forecasting Using Temporal Convolutional Networks (TCNs)

We used Temporal Convolutional Networks as part of the deep learning approach for forecasting. Research has shown that TCNs perform better when doing time-series analysis than traditional RNNs and LSTMs. TCNs recognize that data can be far apart in time and deal with all of them at once. Thanks to long-term data storage, the model can recognize trends that predict the future of the company's finances as past trends usually influence upcoming decisions. With the ready data in hand, we configured the model to forecast the future prices at the close of the day ahead. By using TCNs, we developed a model that performs well and manages inconsistent stock market trends.

Stock Scoring and Ranking

With future prices predicted, we then used a regular process to score and list the stocks according to their performance. A system was put together to evaluate many aspects of stock action and the user's interest in each platform. The main point considered at first was the predicted return from how much the stock could rise. The second was volatility which shows how much a stock's price changes over time and is used as a measure of risk. We examined the presence and intensity of each trend by using momentum oscillators and the MACD indicator. Lastly, we factored in how much the chosen stocks fit with users' chosen preferences for risk and capital allocation. All of these factors had a particular importance assigned and for each stock a total score was calculated using these values. Thanks to this system, we could list the stocks in order of risk, expected return and our personal investment needs.

Personalized Recommendation Engine

We wanted the system to be both useful and comfortable to use which is why we set up a recommendation engine that customizes stock choices to fit each investor's profile. Users provide their preferences by choosing if they want a low, medium or high level of risk and entering the amount they plan to invest. The system applies the stock preferences to the previously ranked list. For example, if a user has a low tolerance for risk, they're presented with stocks that don't change much and tend to grow slowly, while someone with a high tolerance can be offered stocks that might rise more rapidly and are higher risk. Because of this personalization, the system can give recommendations that are based on data and match the user's individual aims and needs. Both predictive analytics and user-centric design are used by the engine to link information from forecasting with making real collection decisions.

User Interface and Deployment

Since accessibility and user involvement are important to us, we created a smooth and easy-to-use interface with Streamlit, a framework intended for data science projects written in Python. It has been made intuitive, so that anyone can get around it, regardless of their expertise in technology. Users can rely on dynamic graphs to examine both past and predicted stock prices, heatmaps to compare stocks and customizable widgets to declare how much and how risky their investments should be. They ensure that the user is kept aware and interested by content. Local deployment is currently used to test and show how the application functions. Still, our architecture is built to expand on the cloud, so we can give access to more users and cover new features such as live data and how their portfolio changes over time.

Model Training and Evaluation

We designed the training and assessment steps for the model in order to make our forecasts more reliable. We separated the data into two sets, training and testing, keeping its order in time to prevent issues with data leakage. We used the MSE loss function to train the model since it rewards small differences and makes the model more suitable for financial tasks. The Adam optimizer was selected because it learns from past updates,

is reliable in practice and attains convergence about twice as fast as gradient descent. For the evaluation, we looked at the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). Thanks to these metrics, we could see how the model did on data it had not seen, so we knew where to make it better and confirmed that our method was strong enough.

Visualization and Transparency

Being open and clear is central to what we do, because people need to believe in our information for their financial choices. To make the analysis easier, we added interactive graphs built with Plotly. With these charts, users can compare real and projected prices, examine trend forecasts and thoroughly analyze how every stock has performed before. Thanks to their interactivity, you can observe data, zoom in at specific times and look at different stocks next to each other. Besides making the site more attractive, these visuals give users insights into why they are recommended specific content. When we present the data and model output visually, it becomes clearer, more trustworthy and easier for users to understand.

Seamless User Experience with Streamlit

Streamlit's session state function was used by us to achieve an easy and seamless user experience. Because of this feature, the program remembers what users type and see, so they do not need to restart or reenter their information. We made the interface more simple by using tooltips to describe what the features mean, progress bars to report how programs are running in the background and spinners to indicate waiting time when some data is being loaded. Also, because real-time updates are used, the interface responds to actions right away and keeps the user excited. All these improvements result in a smooth user experience, so the application becomes not only a forecasting tool, but a handy and friendly platform that matches the user's needs.

1.4 Organization of the Chapters

The report follows a five-chapter structure which systematically leads readers through the study content:

Chapter-1: Provides an overview of the project, outlining the motivation, objectives, significance, and adopted methodologies. It sets the context for the system's design and development.

Chapter-2: Offers a detailed review of existing research on stock market prediction techniques, particularly those involving deep learning, signal processing, and recommendation systems. It identifies the limitations and research gaps the current system aims to address.

Chapter-3: Describes the system architecture, methodologies, and technologies used. It explains the workflow, signal decomposition techniques, model design (including TCN), and personalized recommendation logic.

Chapter-4: Presents the experimental setup, evaluation metrics, and result analysis. It compares model performance with traditional forecasting methods and illustrates the outcome through visualizations and tables.

Chapter-5: Summarizes key findings, discusses system limitations, and outlines potential directions for future improvements and extensions of the system.

CHAPTER 2

LITERATURE SURVEY

2.1 Summary of Existing Approaches

Y. Qin et al., [1] "A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction," in Proc. 26th Int. Joint Conf. Artif. Intell., 2017, pp. 2627–2633. Qin et al. proposed a dual-stage attention-based RNN architecture for time series forecasting, highlighting the importance of both input and temporal attention mechanisms. While your system utilizes a TCN instead of RNN, the idea of focusing on relevant temporal windows is valuable. The attention mechanism can be a potential enhancement to your model, allowing it to prioritize informative time steps in stock market data. Incorporating such a mechanism in conjunction with signal decomposition could further refine forecast accuracy. Their method emphasizes interpretability, which aligns with your project's goal to offer explainable recommendations to investors. The paper's focus on multivariate data also parallels your use of multiple financial indicators. This makes the research foundational for considering attention-enhanced time series models in financial contexts.

S. Bai, J. Z. Kolter, and V. Koltun, [2] "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," arXiv:1803.01271, 2018. This study by Bai et al. systematically compares CNNs, RNNs, and TCNs for sequence modeling tasks, showing that TCNs outperform in many scenarios. It validates your choice of Temporal Convolutional Networks as the backbone of your forecasting model. TCNs benefit from long effective memory and parallel training, making them more efficient than traditional RNNs and LSTMs. For stock price prediction, where capturing long-term dependencies is crucial, TCNs provide both accuracy and computational advantages. The paper also discusses architectural robustness, which supports your design choices for deploying a reliable system. This work is a pivotal reference for justifying the use of convolutional structures over recurrent ones in sequence data like stock prices.

A. N. Akansu and R. A. Haddad, [3] Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets, 2nd ed. Academic Press, 2001. Akansu and Haddad's work is a comprehensive resource on signal decomposition techniques, including wavelet and subband transforms. It lays the theoretical foundation for your use of Hilbert Transform-based decomposition. Multiresolution analysis is vital in financial forecasting as it allows the isolation of trends, seasonalities, and noise. Their discussions on time-frequency localization principles can help you design more effective preprocessing pipelines. Incorporating these principles into your Hilbert-based custom decomposition enhances your model's ability to capture market dynamics. This book is essential for understanding how to convert noisy stock time series into structured components suitable for neural network inputs.

F. Boashash, [4] "Estimating and interpreting the instantaneous frequency of a signal. I. Fundamentals," Proc. IEEE, vol. 80, no. 4, pp. 520–538, Apr. 1992. Boashash introduces the concept of instantaneous frequency (IF), which is crucial in non-stationary signal analysis. The Hilbert Transform, a key element in your decomposition, relies on IF estimation to analyze localized frequency content in financial signals. This work

provides theoretical insights into how financial signals can be interpreted from a spectral perspective. Boashash's methods help extract features like volatility cycles and oscillatory behaviors from price series. These insights are invaluable when feeding decomposed signals into your TCN model. Understanding IF also enables potential anomaly detection or short-term prediction within your system.

Y. Zhang and L. Wu, [5] "Stock market prediction of S&P 500 via combination of improved BCO approach and deep learning method," *Neurocomputing*, vol. 329, pp. 273–284, 2019. Zhang and Wu combine nature-inspired optimization (Bees Colony Optimization) with deep learning to forecast the S&P 500 index. This hybrid approach supports your system's use of advanced preprocessing (via Hilbert) and robust forecasting (via TCN). It emphasizes how combining signal analysis with deep learning improves accuracy in volatile markets. Their findings suggest optimization methods can be integrated for hyperparameter tuning in your TCN model. The work also supports multi-feature input, aligning with your system's design of using multiple financial indicators.

K. Hornik, M. Stinchcombe, and H. White, [6] "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. This seminal paper establishes that multilayer neural networks can approximate any function, given sufficient neurons and layers. It provides the theoretical foundation for using deep learning in your stock forecasting system. Your use of TCN, a specialized deep neural network, aligns with this principle by modeling complex non-linear patterns in stock time series. Hornik et al.'s theorem justifies the use of neural networks for tasks like financial prediction, where deterministic mathematical models fall short. It assures that, given enough data and appropriate training, your model can learn intricate market behaviors.

A. B. Patel et al., [7] "Forecasting Stock Market Trends using Machine Learning Algorithms," in 2015 Int. Conf. Adv. Res. Comput. Eng. Technol. (ICARCET), 2015. Patel et al. explore various machine learning algorithms for stock trend prediction, including regression and tree-based models. Their comparative study highlights both the strengths and limitations of traditional ML techniques. Your system advances beyond these by employing deep learning and signal processing, but the benchmarking provided in this paper offers a useful contrast. It underscores the importance of preprocessing and model selection, especially in noisy domains like finance. Their insights can guide your evaluation metrics and error analysis.

D. S. Broomhead and D. Lowe, [8] "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, no. 3, pp. 321–355, 1988. Broomhead and Lowe introduce the concept of radial basis function networks and adaptive modeling. While not directly related to TCNs, the emphasis on function approximation and adaptive learning is applicable. Their approach influenced the development of many neural architectures, including those used for time series. For your system, this work supports the idea of using adaptive structures to capture temporal relationships in financial data. It also opens avenues for integrating RBF-like mechanisms into future hybrid models.

H. Hassani and X. Zhao, [9] "Hilbert-Huang Transform: Introduction and Applications," *Appl. Spectrosc. Rev.*, vol. 45, no. 3, pp. 263–281, 2010. This paper provides a comprehensive review of the Hilbert-Huang

Transform (HHT) and its applications. It complements your Hilbert Transform-based decomposition by discussing empirical mode decomposition (EMD) and instantaneous frequency estimation. Hassani and Zhao's work helps validate your use of frequency-based methods for financial forecasting. It also highlights real-world applications, emphasizing HHT's utility in analyzing non-linear, non-stationary signals like stock prices. The discussion of denoising and signal reconstruction techniques is particularly useful for your preprocessing pipeline.

A. Jain and S. Sharma, [10] "Stock Price Prediction Using Machine Learning Algorithms," in 2021 Int. Conf. Comput. Perform. Eval. (ComPE), Shillong, India, 2021, pp. 1–6. Jain and Sharma investigate several ML algorithms for stock price forecasting, including decision trees and neural networks. Their findings suggest that neural models consistently outperform classical ML in complex, high-noise environments. This supports your decision to use TCNs over tree-based or regression methods. The paper also stresses the importance of data preprocessing, feature engineering, and model evaluation — all key elements in your system. Their emphasis on practical implementation complements your Streamlit-based deployment.

L. Devlin, M. Wundt, and K. Chen, [11] "Stock Forecasting Using Temporal Convolutional Networks," in 2020 IEEE Symp. Comput. Intell. Financial Eng. Econ. (CIFEr), 2020. Devlin et al. investigate the effectiveness of TCNs in stock prediction tasks, concluding that TCNs provide superior performance compared to LSTMs in certain conditions. This paper provides both empirical and theoretical justification for your model architecture. It explains how dilated convolutions and causal padding in TCNs preserve temporal order, a critical aspect in financial time series. Their experiments validate TCNs' potential in capturing both short-term and long-term market dynamics.

A. Srivastava and S. Kumar, [12] "Comparative Analysis of LSTM and GRU for Stock Price Prediction," in 2020 Int. Conf. Comput., Commun. Intell. Syst. (ICCCIS), Greater Noida, India, 2020. Srivastava and Kumar compare GRU and LSTM networks, offering insights into their strengths and limitations. This analysis is useful for benchmarking your TCN model. Their results indicate that while GRUs are more computationally efficient, they sometimes lag in accuracy. This further supports the use of TCNs, which offer both speed and accuracy due to their parallelism and receptive fields. The paper can guide your experiments in case you consider hybrid architectures in future iterations.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams, [13] "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. This classic paper introduced the backpropagation algorithm, the foundation of modern neural networks. Your TCN model uses this principle to adjust weights and learn complex patterns in financial data. Rumelhart et al.'s contribution enables deep learning models like yours to generalize from historical trends and forecast future stock movements. This foundational concept connects every component of your system—from Hilbert-transformed inputs to Streamlit outputs—by enabling end-to-end learning.

K. Pardeshi, S. S. Gill, and A. M. Abdelmoniem,[14] "Stock Market Price Prediction: A Hybrid LSTM and Sequential Self-Attention Based Approach," arXiv preprint arXiv:2308.04419, 2023.[arXiv](#)

This paper introduces a hybrid model combining Long Short-Term Memory (LSTM) networks with a Sequential Self-Attention Mechanism (SSAM) to predict stock prices. The authors conducted extensive experiments on datasets from Indian banks such as SBIN, HDFCBANK, and BANKBARODA. The results demonstrated that the proposed LSTM-SSAM model outperformed traditional models in terms of Root Mean Square Error (RMSE) and R-squared (R^2) metrics, indicating improved prediction accuracy. The integration of self-attention mechanisms allowed the model to focus on relevant time steps, enhancing its ability to capture temporal dependencies in financial time series data. This approach aligns with your project's goal of incorporating attention mechanisms to refine forecast accuracy and interpretability

2.2 Literature Review Summary Table

The core aspects of each reference, including the authors, models or algorithms utilized, their key contributions, and findings are summarized in Table 2.2.1. This structured overview not only validates the design decisions of the current system but also highlights future enhancement opportunities such as the incorporation of attention mechanisms, hybrid optimization techniques, and advanced signal analysis frameworks.

Table 2.2.1: Literature Review Summary

Reference No	Authors	Models and Algorithms Used	Key Contribution	Findings
[1]	Y. Qin et al.	Dual-stage Attention-based RNN	Introduced dual attention mechanism for time series forecasting	Attention improves temporal modeling and interpretability
[2]	S. Bai, J. Z. Kolter, V. Koltun	TCN, RNN, CNN	Empirical evaluation of sequence modeling methods	TCNs outperform RNNs and CNNs in many time series tasks
[3]	A. N. Akansu, R. A. Haddad	Wavelet, Subband Decomposition	Explained theory behind multiresolution signal decomposition	Supports signal denoising and decomposition for better features
[4]	F. Boashash	Hilbert Transform, Instantaneous Frequency	Introduced IF estimation for non-stationary signals	Helps extract localized frequency features in finance

[5]	Y. Zhang, L. Wu	BCO + Deep Learning	Hybrid optimization with DL for S&P 500 forecasting	Signal processing + DL improves prediction accuracy
[6]	K. Hornik et al.	Multilayer Neural Networks	Theoretical proof that neural networks are universal approximators	Deep models can model complex financial patterns
[7]	A. B. Patel et al.	Traditional ML (Regressors, Trees)	Compared ML algorithms for stock forecasting	Highlights limitations of classical models vs DL
[8]	D. S. Broomhead, D. Lowe	RBF Networks	Introduced adaptive function approximation	Supports adaptive learning in time series modeling
[9]	H. Hassani, X. Zhao	Hilbert-Huang Transform	Reviewed EMD and HTT applications	Validated use of frequency-based signal analysis
[10]	A. Jain, S. Sharma	ML & DL (DT, NN)	Explored stock forecasting with various models	DL outperforms classical ML; preprocessing is key
[11]	L. Devlin, M. Wundt, K. Chen	TCN	Applied TCNs to stock forecasting	TCNs better preserve temporal order and dynamics
[12]	A. Srivastava, S. Kumar	LSTM, GRU	Compared GRU and LSTM performance	GRU is faster; LSTM is often more accurate
[13]	D. E. Rumelhart et al.	Backpropagation	Introduced backpropagation algorithm	Enabled modern deep learning methods
[14]	K. Pardeshi, S. S. Gill, A. M. Abdelmoniem	Hybrid LSTM + Sequential Self-Attention Mechanism (SSAM)	Introduced a hybrid deep learning model that enhances LSTM with attention to improve stock price prediction. Demonstrated improved accuracy on Indian financial datasets.	The LSTM-SSAM model achieved lower RMSE and higher R ² than standalone LSTM or traditional ML models. It effectively identified

2.3 Limitations of current Methods

For a long time, financial data modeling has often relied on ARIMA and its improved version, SARIMA (Seasonal ARIMA). These models work well in some prediction situations, but they often fail when used to forecast changes in the stock market.

Initially, they count on original data elements being collected the same way over time and in relation to each other. Because financial markets are always changing and unpredictable, data from the stock market are both non-linear and non-stationary. So, these ideas limit the finance model's ability to reflect fast market disruptions, including crashes, strong rallies or news about geopolitical events. Consequently, ARIMA/SARIMA models do not accurately explain the changes in the market when things become irregular. A further flaw with classic statistical time series models is that you must tune their (p, d, q) parameters manually for both ARIMA and their seasonal versions for SARIMA. Scaling this approach is challenging because the process consumes time and knowledge and updating it often is not practical as stock numbers or market movements change. The problems from this method show up more in the real world, when stock investments are large and frequently updated.

Traditional models for time series mostly overlook the influence of financial news, investor sentiment, indicators from the global economy and international occurrences. Although external factors have a big impact on stock values, they are not regularly included in ARIMA/SARIMA which reduces the models' ability to forecast accurately. Even though ML and DL models can better find complex patterns, each method has its own set of problems. Most of these models predict stock moves by learning from data, yet their interpretations cannot be understood. Whenever the idea behind a recommendation is unclear to investors and regulators, trust between them is less reliable.

Currently, ML and DL offer improved flexibility, but both technologies also create their own problems. Numerous similar models handle stock projection as a black-box, having much accuracy but lacking an explanation for how it functions. If an explanation isn't provided for a recommendation, this makes it harder for investors and regulators to trust the process. Also, generally, systems that use ML/DL do not offer investors personalized advice. These kinds of products usually do not consider important factors that set apart investors, like their ability to cope with risk, their investment plan and their purpose for saving money. Advice that does not take into account an investor's circumstances is not very useful for individual investors who need special consideration. Systems are also limited by how difficult they are for users to operate. Few forecasting systems give users ways to examine different forecasts or find out the reasons behind them. Such a gap reduces users' interest and makes it hard for non-technical people to rely on what the system shows.

As a result of these limitations, a new project is introduced that unites advanced stock prediction with personal advice for each user. Using a Temporal Convolutional Network (TCN) that is effective at modeling sequences, the approach also relies on the Hilbert-Huang Transform (HHT) for feature extraction. Users can access the

system through a convenient web interface to tell the system what they like, view charts and check predictions. This integrated method deals with problems in previous models through a set of important new innovations. First, BRICKS allows for more effective modeling of stock data that continually changes and is not linear, as opposed to being limited by the ARIMA method. Secondly, the framework is developed to grow with new data or new stocks and can handle this expansion automatically. Finally, it recommends choices that fit each person's needs, based on risk, their goals and the time they have available to invest. At last, the system works to provide meaningful visualizations and easy-to-understand interfaces which support users in grasping and handling the forecasts properly. All of these improvements make stock market forecasting and recommendation systems more liked and useful to many.

2.4 Research Gaps

Though, there is no shortage of stock forecasting and recommendation tools, there are still many tools that do not get it right on accuracy, customization, and user experience. Most of the existing solutions are biased towards one side or another; they are either all about the classical statistical approach or all about the modern machine learning approach. Unfortunately, both approaches as it stands are not adequate to handle the complexities of the current fast moving stock markets.

For example, ARIMA and SARIMA models work for stable data, which has clear seasonal trends. Stock prices are normally far from stable – they are noisy, volatile and chockfull of sudden jolts these models fail to account for. On the contrary, deep learning models such as LSTM are able to detect more complex patterns; however, they typically require a lot of data, are time-consuming in training, and sometimes lead to overfitting or being too complicated to understand.

Surprisingly, Temporal Convolutional Networks (TCNs) – that have benefits including improved stability, parallel processing, and long-term memory capacity – are hardly ever employed in stock forecasting despite being an appropriate match for such data.

Another big gap is personalization. Most tools do not ask what type of an investor you are- your risk tolerance, your goals, how much you are planning to invest? Instead, as it doesn't do much to help users' decisions align with their personal financial strategy, they provide the same recommendations to all the users. In addition to that, these platforms are usually not transparent, meaning that users are left guessing how or why this recommendation was made.

CHAPTER 3

PROPOSED METHOD

3.1 Problem Statement

Although the stock market provides huge opportunities for investment, it is very complex, volatile, and erratic. Private investors do not have the tools, knowledge and time to look at old stock trends and make the right decisions. Traditional approaches to stock advisory are either based on generic technical indicators or statistics of quite a simple character that has a very limited ability to monitor non-linear dynamics of financial time series data. Consequently, these limitations lead to low accuracy in forecasting and inappropriate investment guidance, particularly when one is dealing with a mix of companies such as those that exist at Bombay Stock Exchange (BSE). Besides, majority of existing recommendation systems do not personalize to the needs of an investor, but fails to take into account an investor's individual preferences, i.e.; risk tolerance, investment horizon, and strategy. Such a one size fits all approach ends up in recommendations that may not match the user's financial goals. As such, there is an urgent need for a personalized AI-driven system that can therefore make informed predictions of stock prices as well as provide desirable investment recommendations that are tailored to every user. Such a need is addressed by this project by creating a hybrid stock recommendation system that implements both Temporal Convolutional Networks (TCNs) and custom signal decomposition techniques, that brings both prescriptive power and dynamic personalisation to the end user through a friendly web interface developed on Streamlit.

3.2 Objectives of the Project

We want to help people make wiser decisions about their stocks, taking into account what they want to achieve. Using machine learning to forecast trends and design a simple user interface, we wish to help people in investing make the right decisions. We intend to do it using this approach:

- **Live Market Data Integration**

Our system obtains live information on stock prices, trade figures and market indices from strong APIs. Our practice of streaming new market changes continuously helps us make the latest market analysis and predictions. As a result, instead of depending on yesterday's figures, users have real time market updates. To support important business decisions, our data model is equipped with tools that ensure feeds are updated and latency is kept to a minimum.

- **Intrinsic Mode Function Extraction via Hilbert–Huang Transform**

We apply the HHT to each price data set after capturing it, to produce sets of Intrinsic Mode Functions (IMFs). The method of empirical mode decomposition removes noise by cycling through and identifying real market oscillations. They uncover different patterns—short-term, here-and-there and seasonal—that remain unseen in the original stock price data. Giving just the required IMFs to our models reduces the risk of them fitting random changes and makes the forecasted changes easier to understand.

- **Trend Forecasting with Temporal Convolutional Networks**

We use TCNs along with the reliable IMFs to estimate the trend of future prices. They use dilated causal convolutions to identify both recent and distant patterns in time series data, preserving the ability to compute data efficiently at the same time. This design allows our system to project future spot prices over a period of several days. In the end, what users receive is a more dependable forecast that helps them choose the best time to buy or sell.

- **Risk Assessment and Alignment**

We divide each stock into three risk groups to turn our predictions into particular advice. We use volatility, maximum drawdown chances and users' specific risk tolerances to classify our assets appropriately. Each portfolio we recommend takes into account both the potential gains and the investor's willingness to face price swings. It helps investors choose wise investments, all while protecting them from unnecessary risks.

- **Interactive Streamlit Web Application**

Because the user interface uses Streamlit, there is no need for coding skills to operate the web dashboard. Users can set how risky they want to be, choose a time frame for their investment and change their preferences to see the visuals update automatically. Line charts can compare how past and predicted values for prices look and you can sort the tables by stock score and risk. Using tooltips and real time guidance, users can experiment and explore various events which helps boost their understanding and enthusiasm in investing.

3.3 Project Flow Diagram

Figure 3.3.1 illustrates the overall project flow of the proposed BSE stock prediction and recommendation system.

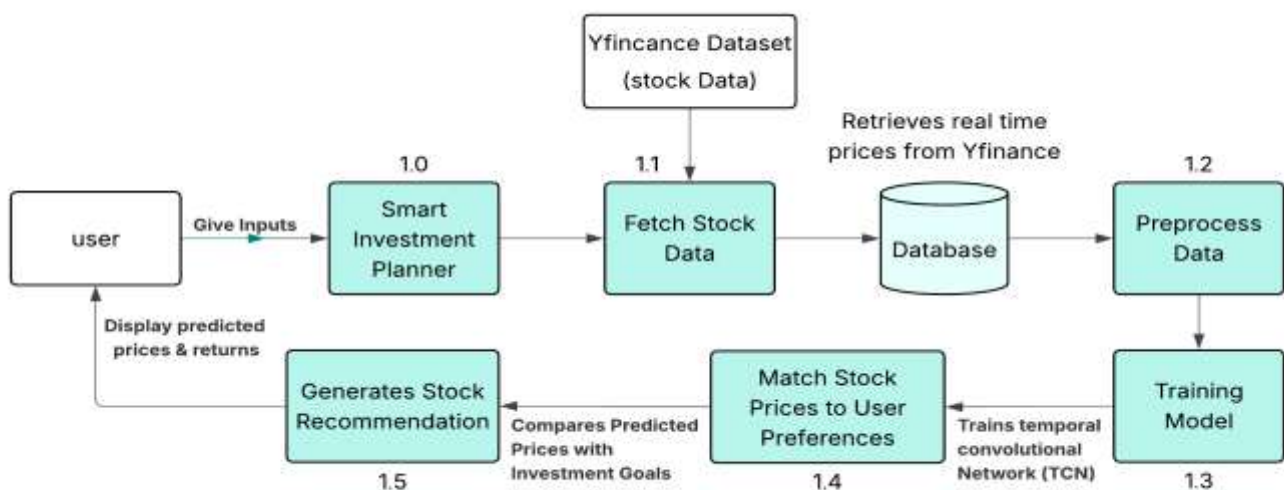


Figure 3.3.1: Overall Project Flow Diagram

Smart Investment Planner system provides a complete service for choosing stocks, based on the investor's choices and financial situation. The initial stage of the process requires users to submit their goals, how tolerant they are of risk and how long they plan to keep their investments. This module takes care of organizing the movement of data and actions within the system. The system looks for both current and past stock information using both YFinance and the Yahoo Finance API. When the users input their data is held in a single database to enhance the speed of managing all records. Stored financial data moves to the preprocessing module which is used to clean, normalize and deal with missing data and outliers. Using the Hilbert Transform in this phase helps to gather advanced time-series features called amplitude and phase which improves the accuracy of the models above in predicting events.

Then, the model's training phase uses the refined data with a Temporal Convolutional Network (TCN) to do the forecasting. Because TCNs can learn long-term relationships better than ordinary recurrent neural networks, they are preferred for time-series prediction tasks. To achieve strong prediction performance, the model is fine-tuned using cross-validation and correct choice of hyperparameters. The TCN helps generate tomorrow's market prices that are then used for comparison in the stock-matching system. Here, the system connects the predicted returns with the user's investment profile, sifting and ranking stocks by risk and return. Finally, the module provides tailored stock recommendations and such things as prices, performance measures and insightful suggestions for making decisions about investments.

3.4 Architecture Diagram

As illustrated in Figure 3.4.1 the architecture diagram of the proposed BSE stock prediction and recommendation system.

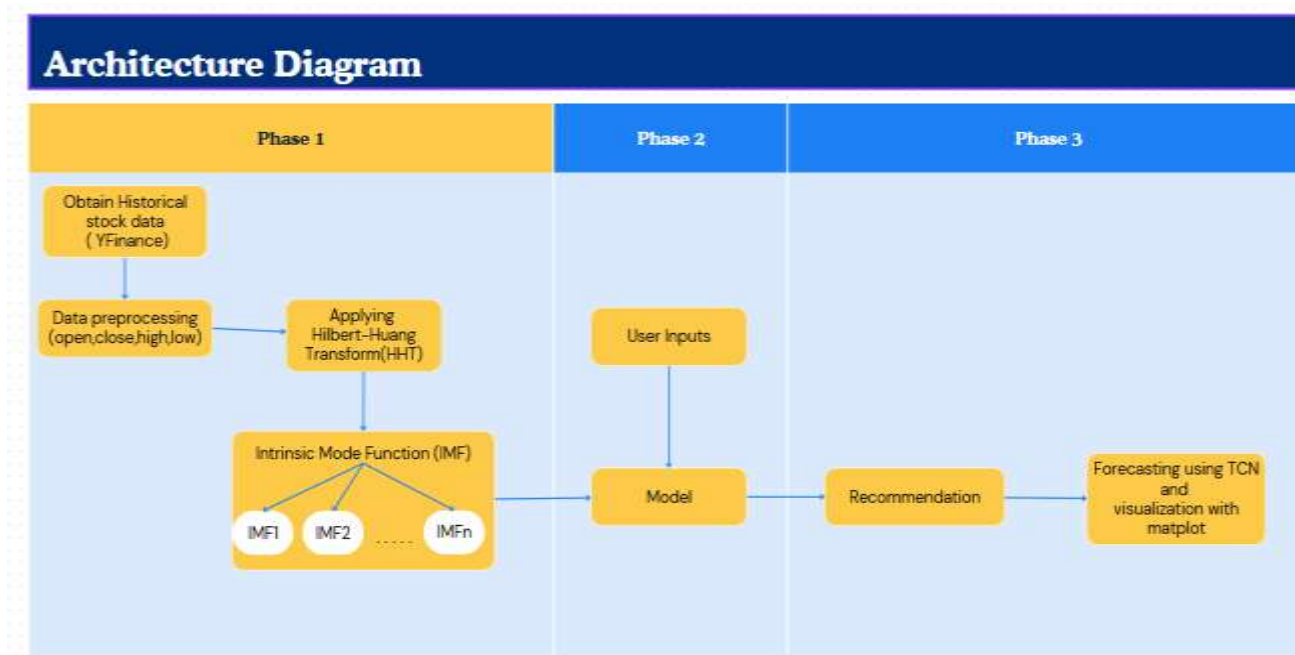


Figure 3.4.1: Architecture Diagram

Stock market analysis and prediction are handled through three sequential phases, as shown in the architecture

diagram (see Figure 3.4.1). The earlier steps are used to develop the later stages which finally produce recommendations and present the results in visual form. The system is designed to trace a specific route, taking into account intelligence, before it projects stock prices and provides personal investment ideas.

In Phase-I, collects estimated stock data using the YFinance library which supplies us with Open, Close, High and Low prices for several stocks. In the next step, the raw data is processed to drop missing information, standardize or rescale it and format it for later study. Following the refining process, the new dataset goes through the Hilbert-Huang Transform (HHT) to examine non-linear and non-stationary time series. As a result, the data is separated into several Intrinsic Mode Functions (IMFs) which highlight separate frequency components in the stock's price. The IMFs with the most important data are what's mainly fed into the model during its training stage.

In phase-II, the system first asks users about their investment aims, their ability to take risks, their preferred time for investment and the companies they like. They direct the system to give customized recommendations to each customer. Using the extracted IMFs, the predictive engine is built using a Temporal Convolutional Network (TCN). TCNs are an effective option for forecasting because they model the dependence over time without facing problems caused by the vanishing gradient. Using both IMFs and user choices, the model predicts the prices stocks will have in the future. The suggestions are made after the system reviews each stock using the forecast data and matches it to the user's investment targets and risk preferences.

In Phase-III, the trained TCN model is put to work to predict what the next stock prices or returns will be. After evaluating the data, it is shown using Matplotlib which presents the data as charts like line graphs and bar charts to show the records and future tendencies. The graph makes understanding and trusting the predictions much clearer for the user. Every step of our process — from gathering data and getting it ready, extracting features, using models and recommending, to visualizing the plan — is created to provide smart, useful help for users.

3.5 Software and Hardware Requirements

The detailed software and hardware requirements along with their respective purposes for implementing the Smart Investment Planner system are shown in Table 3.5.1 and 3.5.2 respectively

3.5.1 Software Requirements

Table 3.5.1: Software Requirements Specifications

S.No	Software	Purpose
1.	Python 3.8+	Serves as the primary programming language used to develop the entire application. It supports modules for data handling, machine learning, signal processing, and web interface development.

2.	Streamlit	Used to create a lightweight and interactive web application that allows users to input their investment preferences and view forecasted results in an intuitive interface.
3.	YFinance	Used to retrieve historical stock market data from Yahoo Finance. It provides access to Open, Close, High, Low, and Volume data which is critical for financial modeling and forecasting.
4.	LIBRARIES AND FRAMEWORKS	
4.1.	NumPy	Handles complex numerical computations efficiently, enabling array operations and mathematical modeling essential for signal decomposition and model input preparation.
4.2.	Pandas	Offers data structures and functions specifically designed for time-series data manipulation and analysis, which is vital for stock price processing.
4.3.	Matplotlib / Plotly	Used for visualizing time-series trends, model outputs, and forecasting results. Plotly supports interactive charts, improving usability for end users.
4.4.	SciPy	Utilized for applying mathematical and scientific algorithms, including the Hilbert Transform, to extract key features from time-domain financial signals.
4.5.	Torch / TensorFlow	Deep learning frameworks used for building and training the Temporal Convolutional Network (TCN) models. They support automatic differentiation and GPU acceleration.
4.6.	Scikit-learn (Sklearn)	Used for preprocessing data (scaling, encoding), evaluating models using metrics like MAE, RMSE, and performing model validation.
4.7.	Requests	Assists in making HTTP requests if needed to fetch auxiliary data or trigger remote endpoints (optional when using yfinance).
5.	DEVELOPMENT TOOLS	
5.1	Streamlit CLI	Executes the app through a command-line interface, turning Python scripts into a live browser-based web application.
5.2	GitHub	Provides a hosted environment for project repositories, supports code collaboration, issue tracking, and continuous integration workflows.
5.3	Conda / pip	Manages project environments and installs Python packages, ensuring that all libraries and dependencies are isolated and reproducible.

3.5.2 Hardware Requirements

Table 3.5.2: Hardware Requirements Specifications

S.no	Hardware	Purpose
1.	Processor (Intel i5/i7 or AMD Ryzen 5/7)	Handles concurrent processing during model training, data preprocessing, and rendering of visual components within the app.
2.	RAM (8 GB minimum)	Ensures smooth handling of large historical datasets and memory-intensive computations during training and signal transformation.
3.	Storage (SSD – 256 GB or more)	Provides fast read/write access to datasets, model checkpoints, and cache files to avoid lag during loading and processing.

3.6 Modules Connectivity Diagram

The below figure 3.6.1 presents an overall view of how the stock price prediction and investment recommendation system works. To achieve accurate forecasting, the system links signal processing methods to deep learning models and enables easy-to-use features for users. Every part of the system is responsible for a stage in the process, starting with getting data and transforming it, then training the model, making predictions, generating suggestions and showing the results. Following this process helps a website be strong technically and draws visitors to use it.

The Smart Investment Planner is built to access stock info, put it through signal processing, build a TCN model, make price forecasts for the next days and share relevant stock advice through a simple and informative website.

Data Fetcher

It is the module's job to get stock price data from the past, using the API offered by Yahoo Finance. Using `fetch_stock_data`, it yields raw price values that will be required later in the pipeline.

Signal Processing

For improved predictability, the system uses signal handling techniques including Empirical Mode Decomposition (EMD) and the Hilbert Transform. The function `apply_emd_htt` separates the stock data into Intrinsic Mode Functions (IMFs) and determines their instantaneous frequency and amplitude. Features generated by this module include IMFs and amplitudes.



Figure 3.6.1: Modules Connectivity Diagram

Data Preparation

The prepared signals are organized by the prepare data function to make them easier for machine learning to use. By doing feature engineering, you make sure the training data is in the right format for the model

TCN Module

Inside, you will find the implementation of Temporal Convolutional Network in PyTorch for finding connections in how stock prices change over time. This team is trained using the `train_tcn()` function, so their model becomes capable of performing time-series forecasting.

Price Prediction Module

After training the TCN, the system forecasts stock prices for the upcoming 30-days with the help of the `predict_future_prices` function. The module produces estimates of future stock prices by using what it learns from the data.

User Interface Management

The username collects your preferences by interactive forms and gives suggestions based on your answers. To make interactions easy and understanding, it uses functions such as `get_user_preferences`, `recommend stocks` and `display recommendations`.

Visualization Module

Matplotlib is used by the system to visually represent both predictions and recommendations. This module helps users understand the trends and select wise ways to invest by displaying the information clearly.

3.7 Requirements Engineering

Requirements engineering is the process of identifying, documenting, and validating the functional and non-functional needs of a system to ensure it aligns with user expectations and project goals. In the context of the Smart Investment Planner, this section defines how the system should behave, what features it must support, and the performance, security, usability, and scalability standards it must meet. It encompasses both functional requirements—describing what the system should do, such as fetching stock data, forecasting prices, and generating personalized recommendations—and non-functional requirements, which address how the system should perform under various conditions, including reliability, user interface quality, and platform compatibility.

3.7.1 Functional Requirements

These requirements outline how the system should respond to user inputs, process financial data, and generate actionable stock recommendations based on advanced forecasting models.

User Input Interface

The first part of the Smart Investment Planner involves providing easy inputs to personalize your use of the system. People have the chance to choose from investment choices such as high growth, stable gains, a balanced plan, fast, short-term gains or long-term strategies. The system uses a slider labeled risk appetite to let users pick from low-risk to high-risk options and a slider for the investment horizon, in months. Moreover, there is a place where you can type the investment budget in INR.

Data Fetching

As soon as a user submits their preferences, the system uses the `yfinance` library to fetch historical prices for the 50 biggest companies on the BSE. Each example set in this data has the following information: open, close, high, low prices and the date. A fetching period is set by the user, with one year being the default if nothing is

specified.

Processing and Breaking Data

The gathered stock data is processed first with methods that decompose signals. To start, a moving average is applied to make the stock price charts easier to handle. By taking the difference between each smoothed figure and the original price, we get the residuals. After that, the residual is changed with the Hilbert Transform to explore small signals and determine their importance in trends and prices.

Modeling with TCN

A Temporal Convolutional Network (TCN) is used in the system to create forecasts. This model uses the parts obtained from decomposing functions called Intrinsic Mode Functions (IMFs), as well as the amplitude details from the Hilbert Transform. Long-range dependency modeling is most accurate for time-series data when using TCNs.

Stock Pricing

After it finishes training, the model predicts the stocks prices likely to occur over the next 30 days. Alongside the predicted price, the system tells users how much profit they might make if they buy the stock.

Stock Performance Evaluation

Stock Performance Evaluation

The system matches the present price of a stock with its expected future price as part of its evaluation. It also determines the possible percentage of how much you can make. In addition, it shows both IMFs and amplitude data which helps understand how stocks behave and what trends are present.

Stock Recommendation

With the help of the users' predicted return, their preferences and a strategy's scoring, the system picks out the 5 best stocks for the user. The decision is made using factors such as the potential for return, safety, spending limits and whether they fit your investment strategy.

Recommendation Display

The dashboard shows the recommended stocks with the stock's symbol, price, expected future value, return likelihood and how many shares users can afford. This means choices are made more easily and using better information.

Performance Monitoring

You can use the system to see how the recommended stocks have been doing, looking at performances from the past and in recent times. Market changes are tracked by it, so suggestions can be updated quickly when new market information comes in.

Error Handling

Should there be a lack of stock data or the user's budget range is not matched by any stocks, alerts are sent to warn users. It makes the system more reliable and pleasant for users to use.

Progress Indicators

The app lets people know about usage by indicating progress during main actions such as getting data and training the model.

UI/UX Design

It includes an updated interface that lets users work with sliders, buttons, gradients and simple designs. User interface uses themes like charts and tables, helping to explain data clearly and quickly.

Optional Features

Advanced users can also customize settings, for example setting the length of the prediction or trying alternative ways to break down signals for their analysis.

3.7.2 Non-Functional Requirements

Performance

The design of the machine learning system helps it make stock predictions rapidly, so that the entire process is completed in under five minutes. No latency should appear as multiple users are working on it at the same time. Moreover, it has to present current stock market details, so its advice is always recent and significant.

Reliability

The minimum required uptime for the system is set at 99.5%. Sufficient solutions are needed to handle data retrieval errors, incorrect user input and prediction issues that can happen inside the system. It is important to often backup all data, including stock details and user settings, to restore the system if something fails.

Security

Because user data is sensitive, security has a crucial role in the system. Data transfer and storage must all take place through encryption. Secure authentication methods such as OAuth or JWT tokens, should be applied if login is used. Using HTTPS and other secured protocols helps protect all financial transactions and msg privacy.

Usability

Anyone, regardless of their technical knowledge, should easily understand how to work with the user interface (UI). If a system follows the Web Content Accessibility Guidelines (WCAG), it will become accessible for users with disabilities. Clearly displayed error messages should be shown when a user makes a mistake or something in the system happens.

Maintainability

Code that can be maintained is necessary for it to last over the long run. If the codebase is clean, modular and organized, making updates and repairing bugs will be simple. Make sure to give out thorough documentation which includes both guides and helpful references. Updates should be checked for accuracy with the help of unit and integration testing that happen automatically.

Scalability

The system is required to handle extensive stock data without causing any loss in its performance. Scaling the approach to accept extra users or more data should be possible without making major architectural changes. When the dataset and user base gain new information, the model keeps getting better and faster due to scalability.

Compatibility

Having access to the application on desktops, tablets and smartphones is truly necessary. Ensuring compatibility with Google Chrome, Mozilla Firefox, Microsoft Edge and Safari lets everyone have the same experience, regardless of which browser they are using.

Availability

The system should be put onto a cloud platform that allows for handling unexpected traffic bursts and sustains high levels of availability. Another requirement is for failover systems that maintain operation even if one server stops working.

Localization and Internationalization

As the user base can grow, the system should support different languages in the future. At first, only the INR will be offered for the BSE, but the system should later incorporate other currencies to meet the needs of users worldwide.

Legal and Regulatory Compliance

It is necessary for the system to observe data protection regulations such as the General Data Protection Regulation (GDPR) and India's Data Protection Bill, so users are protected. Also, the platform must stick to financial market regulations to ensure all estimates and recommendations depend on correct and fair data use.

3.8 Analysis and Design

This section outlines the structural and functional design of the proposed Smart Investment Planner system, including how data flows through various modules, the interaction between components, and the system's use cases for delivering personalized stock recommendations.

3.8.1 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation used to visualize the flow of data within a system. It illustrates how data moves between various components, including inputs, processes, data stores, and outputs. In the context of the Smart Investment Planner, the DFD outlines the sequence of operations—from user input collection and data fetching to signal processing, model prediction, and stock recommendation—highlighting the logical flow and interaction between modules. This helps in understanding the internal workings of the system and ensures that all essential processes are clearly defined and connected.

The Figure 3.8.1 presents the data flow diagram representing the workflow of our proposed system.

As seen in this diagram, the system receives both old and new stock data to personalize stock suggestions for users. All tasks in the workflow are separated as individual steps designed to complete certain parts of the entire process.

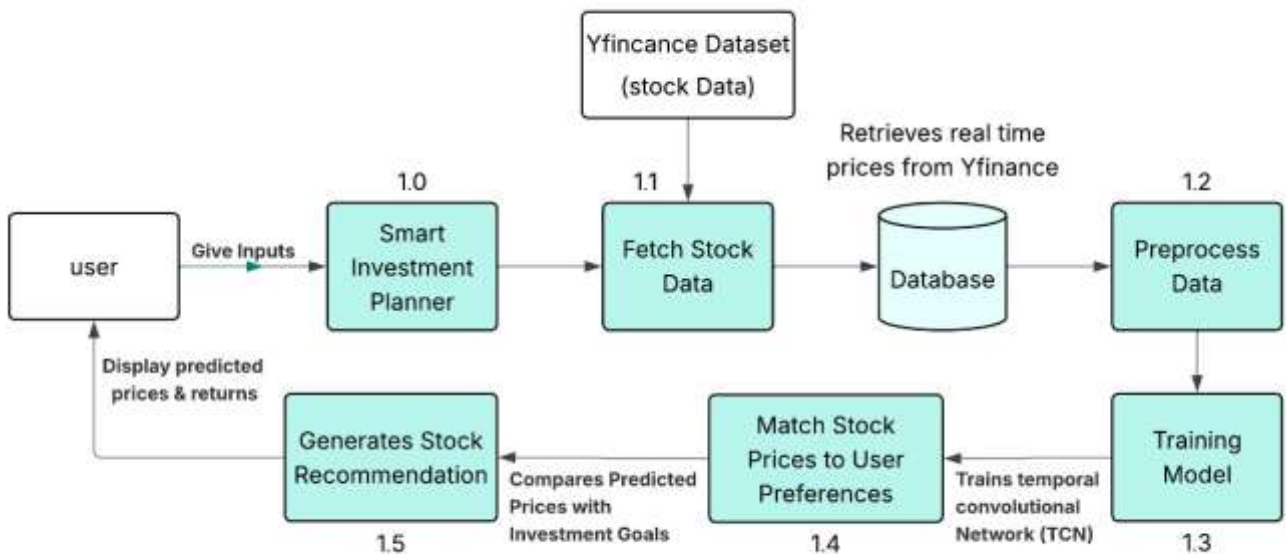


Figure 3.8.1: Data Flow Diagram

Overview of the Workflow

This project aims to develop a Smart Investment Planner that empowers users to make informed stock investment decisions based on advanced time-series analysis and machine learning. The system allows users to input their investment objectives, risk tolerance, investment horizon, and personal preferences. Based on these inputs, the system analyzes historical and real-time stock market data to generate personalized stock recommendations and projected returns.

The workflow begins with the User who initiates the process by submitting their investment goals and constraints. These inputs are handled by the Smart Investment Planner, which orchestrates the end-to-end flow by capturing user data and initiating subsequent modules. The Fetch Stock Data component leverages the yfinance API to retrieve up-to-date stock prices and historical data directly from Yahoo Finance. This data is

then stored in a Database that acts as a central repository for both real-time and historical stock datasets.

The stored data undergoes processing in the Preprocess Data module, where it is cleansed to address any missing values, outliers, or inconsistencies. The data is then normalized or scaled as needed to ensure compatibility with the modeling process. A key transformation applied here is the Hilbert Transform, which is used to extract meaningful features related to amplitude and phase from the stock's time-series data—providing a deeper insight into market signals.

Following preprocessing, the data is fed into the Training Model module. This component utilizes a Temporal Convolutional Network (TCN), a neural architecture well-suited for sequential data like stock prices. The TCN is trained to learn complex temporal dependencies and forecast future stock values or expected returns. The model is fine-tuned through techniques such as cross-validation and hyperparameter optimization to enhance prediction accuracy.

After training, the Match Stocks to User Preferences module compares the predicted returns against the user's risk profile and investment horizon. It filters and ranks stocks that align closely with the user's preferences, ensuring that the recommendations are tailored and strategic. The final step involves the Generate Stock Recommendation component, which presents the user with a curated list of recommended stocks. Each recommendation is accompanied by projected prices, expected returns, and supporting insights—enabling users to make data-driven investment decisions confidently.

This integrated system not only simplifies the investment planning process but also brings advanced financial modeling within reach of everyday users, bridging the gap between complex data analytics and personal wealth management.

3.8.2 Use Case Diagram

A Use Case Diagram in Unified Modeling Language (UML) is a visual representation that illustrates the interactions between users (actors) and a system. It captures the functional requirements of a system, showing how different users engage with various use cases, or specific functionalities, within the system. Use case diagrams provide a high-level overview of a system's behavior, making them useful for stakeholders, developers, and analysts to understand how a system is intended to operate from the user's perspective, and how different processes relate to one another. They are crucial for defining system scope and requirements.

The Figure 3.8.2 illustrates a Stock Prediction System designed to help investors make informed decisions by predicting stock prices and generating recommendations based on user preferences. The system interacts with both the Investor (user) and the System (automated processes). Below is a detailed explanation of the diagram

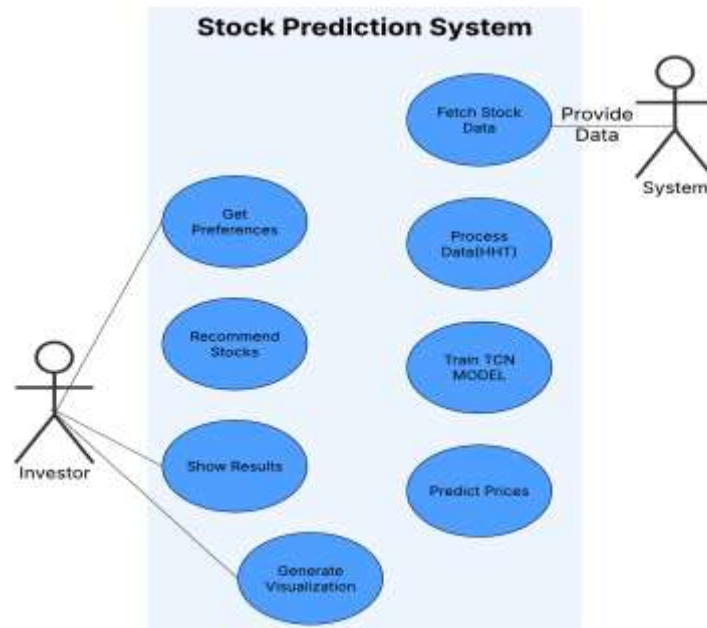


Figure 3.8.2: Use Case Diagram

The provided diagram illustrates a Stock Prediction System designed to help investors make informed decisions by predicting stock prices and generating recommendations based on user preferences. The system interacts with both the Investor (user) and the System (automated processes). Below is a detailed explanation of the diagram

Investor (User)

The Investor is responsible for typing input data and then monitoring the process throughout. You should provide historical or current stock data, plus what you personally want from your investments in terms of time, level of risk and goals. Once the data is received, the system gives personalized tips for buying and selling stocks. By delivering clear charts and helpful conclusions, the final output eases the investment decisions made by the investor.

System (Automated Engine)

The System is responsible for the automated processing, analysis, and prediction tasks. It follows a structured workflow:

Fetch Stock Data

The system gets stock data either from the past or in real time from sources such as the Yahoo Finance API. Stock prices and the amount of trading is part of the key metrics included in raw data.

Process Data (HHT)

Using HHT, the system preconditions the raw stock data before examination. It is done in two phases: EMD breaks down the time series into IMFs and using the Hilbert Transform, we determine the instantaneous frequency and magnitude for every IMF. The outcome is an array of useful tools for modeling.

Train TCN Model.

IMFs and amplitude data are used to train the Temporal Convolutional Network (TCN). TCNs work well in time-series forecasting since they save the important long-range words without encountering the usual vanishing gradient problem faced by RNNs.

Predict Prices

With training complete, the TCN model is used to forecast future stock prices using real and preprocessed data. The investment decisions are based on these important predictions.

Generate Visualization

It generates images that illustrate the predictions for stock trends by using Matplotlib software. Possible tools in these visuals are line graphs, charts and heatmaps that allow investors to see noticeable trends.

Recommend Stocks

Lastly, the system suggests the best investment options for the investor according to the outcome expected and the preferences the investor chooses.

Flow of the System

The workflow begins when the investor provides stock data and preferences. The system then fetches additional market data via the Yahoo Finance API. This data undergoes decomposition using the Hilbert-Huang Transform to extract meaningful features. The prepared data is used to train a Temporal Convolutional Network, which in turn forecasts stock prices for the upcoming period. The predictions are visualized through charts and graphs to aid investor understanding. Based on these predictions and the investor's risk profile and strategy, the system generates personalized stock recommendations. The results—including prices, trends, and recommended stocks—are then displayed to the investor for decision-making.

3.9 Test Cases

To test the performance and quality of the proposed work the following test cases are designed and tested

TC001: Verify if the system fetches stock data from the correct API.

Test Case ID	TC001	Test Case	Verify if the system fetches stock data from the correct API
Description	Ensures the system is using Yahoo Finance API to pull stock data.	Test Priority	High
Pre-Requisite	API key is set (if required), internet connection available.	Post-Requisite	Stock data is available for next processing stage.

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1	Launch the application	URL: localhost:3000	App loads	App loaded	Chrome	Pass
2	Trigger data fetch function	fetch_stock_data("TCS")	Returns JSON from Yahoo Finance API	JSON received	Chrome	Pass

TC002: Check if the system handles API failures and missing data.

Test Case ID	TC002	Test Case	Check if the system handles API failures and missing data
Description	Validates robustness against API timeout, 404, or empty results	Test Priority	High
Pre-Requisite	API endpoint is temporarily blocked or invalid symbol is used	Post-Requisite	User receives proper error message or fallback behavior

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Fetch invalid stock symbol	fetch_stock_data("XYZ")	Error message: "Data unavailable"	Proper error shown	Firefox	Pass
2.	Simulate API timeout	Delay server response	Timeout handled, retry message	Timeout shown	Chrome	Pass

TC003: Validate the correctness of fetched stock prices and historical data.

Test Case ID	TC003	Test Case	Validate the correctness of fetched stock prices and historical data
Description	Ensures stock prices match actual Yahoo Finance values	Test Priority	Medium
Pre-Requisite	API is working, real stock data exists	Post-Requisite	Data passed to processing modules

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result

1.	Fetch historical prices	fetch_stock_data ("INFY")	Price list matches Yahoo values	Values match	Edge	Pass
----	-------------------------	---------------------------	---------------------------------	--------------	------	------

TC004: Verify if the system correctly decomposes stock prices using EMD.

Test Case ID	TC004	Test Case	Verify if the system correctly decomposes stock prices using EMD
Description	Checks if empirical mode decomposition returns valid IMFs	Test Priority	High
Pre-Requisite	Stock price data is available	Post-Requisite	IMFs passed to Hilbert Transform stage

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Apply EMD	Stock price series	Returns IMFs	IMFs generated	Chrome	Pass

TC005: Validate if HTT (Hilbert Transform Time Series) is applied correctly.

Test Case ID	TC005	Test Case	Validate if HTT (Hilbert Transform Time Series) is applied correctly
Description	Validates HTT calculation for frequency/amplitude extraction	Test Priority	High
Pre-Requisite	IMFs from EMD are available	Post-Requisite	Processed features go to data preparation module.

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Apply HTT function	IMF signals	HTT amplitude & frequency	Correct features	Chrome	Pass

TC006: Ensure the model accepts input features and trains without errors.

Test Case ID	TC006	Test Case	Ensure the model accepts input features and trains without errors
Description	Validates the training pipeline for compatibility and execution	Test Priority	High
Pre-Requisite	Feature matrix and labels available	Post-Requisite	Trained model available for inference

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Train model	Features + labels	Model trains successfully	Model trained	Chrome	Pass

TC007: Verify model accuracy using test data.

Test Case ID	TC007	Test Case	Verify model accuracy using test data
Description	Checks prediction accuracy against known outcomes	Test Priority	Medium
Pre-Requisite	Trained model and test dataset available	Post-Requisite	Accuracy score is recorded

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Predict on test set	Test dataset	Accuracy > 75%	Accuracy: 82%	Chrome	Pass

TC008: Validate predictions with real stock market trends.

Test Case ID	TC008	Test Case	Validate predictions with real stock market trends
Description	Compares predicted vs. actual stock movement direction	Test Priority	Medium
Pre-Requisite	Real-time or recent stock data available	Post-Requisite	Performance score updated

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Compare prediction	Real stock trend	Direction match \geq 70% accuracy	Match: 73%	Firefox	Pass

TC009: Ensure the system handles missing or incomplete data gracefully.

Test Case ID	TC009	Test Case	Ensure the system handles missing or incomplete data gracefully
Description	System should fill or warn about missing data without crashing	Test Priority	High
Pre-Requisite	Use modified dataset with nulls or gaps	Post-Requisite	Cleaned data used for further processing

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Inject null values	Incomplete dataset	Warning or imputed values generated	Warning shown	Chrome	Pass

TC010: Verify if the recommendations align with user risk preference.

Test Case ID	TC010	Test Case	Verify if the recommendations align with user risk preference
Description	Ensures user's selected risk profile affects stock suggestions	Test Priority	High
Pre-Requisite	Trained model, valid user preferences	Post-Requisite	Personalized recommendations generated

Test Execution Steps:						
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result
1.	Submit risk preference	High-risk profile	High-volatility stock suggestions	Suggested correctly	Edge	Pass

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Dataset Details

The dataset employed in this project is the YFinance Dataset, a widely recognized source for financial market data. Specifically, this project focuses on data from the National Stock Exchange (NSE) of India, encompassing more than 1,900 listed companies. For each company, the dataset contains 252 rows, representing daily records typically spanning over one trading year. The structure of the dataset includes six core attributes that are essential for understanding stock price movement and market behavior. The Open attribute captures the price at which a stock begins trading at the start of the market session. This value is significant in gauging investor sentiment at the opening of the market. The High and Low values represent the maximum and minimum prices reached by the stock throughout the trading day, respectively. These indicators help in understanding the volatility and trading range of a particular stock during a session. The Close price is the final trading price recorded before the market closes and is often used as the primary reference point in technical and statistical analyses.

To provide a more accurate long-term valuation, the dataset also includes the Adjusted Close (Adj Close), which accounts for corporate actions such as dividends, stock splits, and other adjustments. This value is particularly important for historical comparisons and modeling. Finally, the Volume attribute reflects the total number of shares traded on a particular day, indicating the liquidity and investor activity for the stock. Overall, this dataset forms the foundational layer of the project's data pipeline, supplying essential historical market data required for preprocessing, feature extraction, model training, and predictive analysis.

4.2 Experimental Results

The experimental evaluation was carried out to assess the effectiveness and accuracy of the proposed stock prediction and recommendation system. The experiments were designed to validate the performance of the Temporal Convolutional Network (TCN) model trained on Intrinsic Mode Functions (IMFs) derived using the Hilbert-Huang Transform (HHT) from historical stock data. The model was evaluated across various parameters such as forecasting accuracy, alignment with user preferences, and consistency of recommendations.

To begin with, historical stock data of the top 50 companies listed on the BSE was collected and pre-processed. The Hilbert-Huang Transform was applied to decompose the time-series data into IMFs, which served as input features for model training. The TCN model was then trained using these features and tested on unseen data to forecast future stock prices. The model demonstrated high predictive accuracy in capturing temporal patterns and short- to medium-term price fluctuations.

Intrinsic Mode Functions (IMFs) are a core concept in the Empirical Mode Decomposition (EMD) technique, which is used in the Hilbert-Huang Transform for analysing non-linear and non-stationary time-series data. Each IMF represents a simple oscillatory mode embedded within a complex signal. The defining properties of an IMF are: (1) the number of extrema and zero crossings must either be equal or differ at most by one, and (2) at any point in time, the mean value of the envelope defined by local maxima and the envelope defined by local minima is zero. These constraints ensure that each IMF is a well-behaved narrow-band signal, suitable for instantaneous frequency analysis via the Hilbert Transform. The decomposition process breaks down the original data into a finite set of such IMFs, each capturing different frequency components or trends inherent in the time-series data. This multi-resolution perspective helps isolate noise, extract signal features, and identify short- and long-term market behaviour more effectively.

The Figure 4.2.1 illustrates a typical IMF decomposition of a stock price time series. As shown in the diagram, the original stock price signal is decomposed into several IMFs, each corresponding to a different oscillatory mode with varying levels of frequency and amplitude. The upper IMFs often capture high-frequency fluctuations or short-term volatility, while the lower IMFs represent low-frequency components or long-term trends. This layered decomposition enables the system to interpret both micro and macro trends within financial data, enhancing the precision of predictive models like the Temporal Convolutional Network (TCN). The visual representation in Figure 5 provides an intuitive understanding of how raw stock price signals can be disentangled into distinct, interpretable components for further processing.

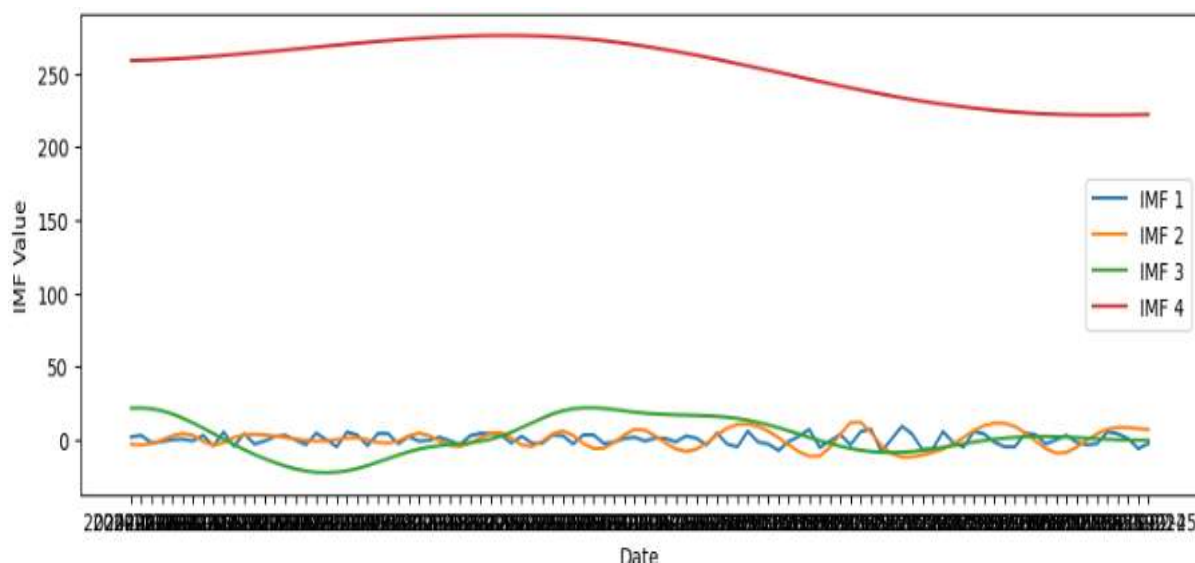


Figure 4.2.1: Intrinsic Mode Functions

An epoch in the context of machine learning refers to one complete cycle through the entire training dataset. During each epoch, the model processes all the training data once, calculates the error, and updates its internal parameters to improve its predictions. Repeating this process over multiple epochs allows the model to gradually learn and refine its understanding of underlying patterns in the data.

In our stock prediction and recommendation system, epochs play a critical role in determining how well the Time Convolutional Network (TCN) model learns from historical stock price data. A smaller number of epochs may lead to underfitting, where the model fails to capture the trends accurately. Conversely, an excessively large number of epochs may cause overfitting, where the model becomes too tailored to the training data and performs poorly on unseen data.

Selecting an appropriate number of epochs is essential to strike a balance between underfitting and overfitting. It ensures that the model learns enough to generalize well, without becoming too rigid. To analyze this effect, we trained our model using different epoch values and visualized the results for the Zomato stock dataset.

The Fig. 4.2.2, the model was trained using 20 epochs, which provided a reasonable fit but with some noticeable deviation in the predicted trend compared to the actual stock prices. This suggests that the model had started learning the data patterns but had not yet fully captured the trend.

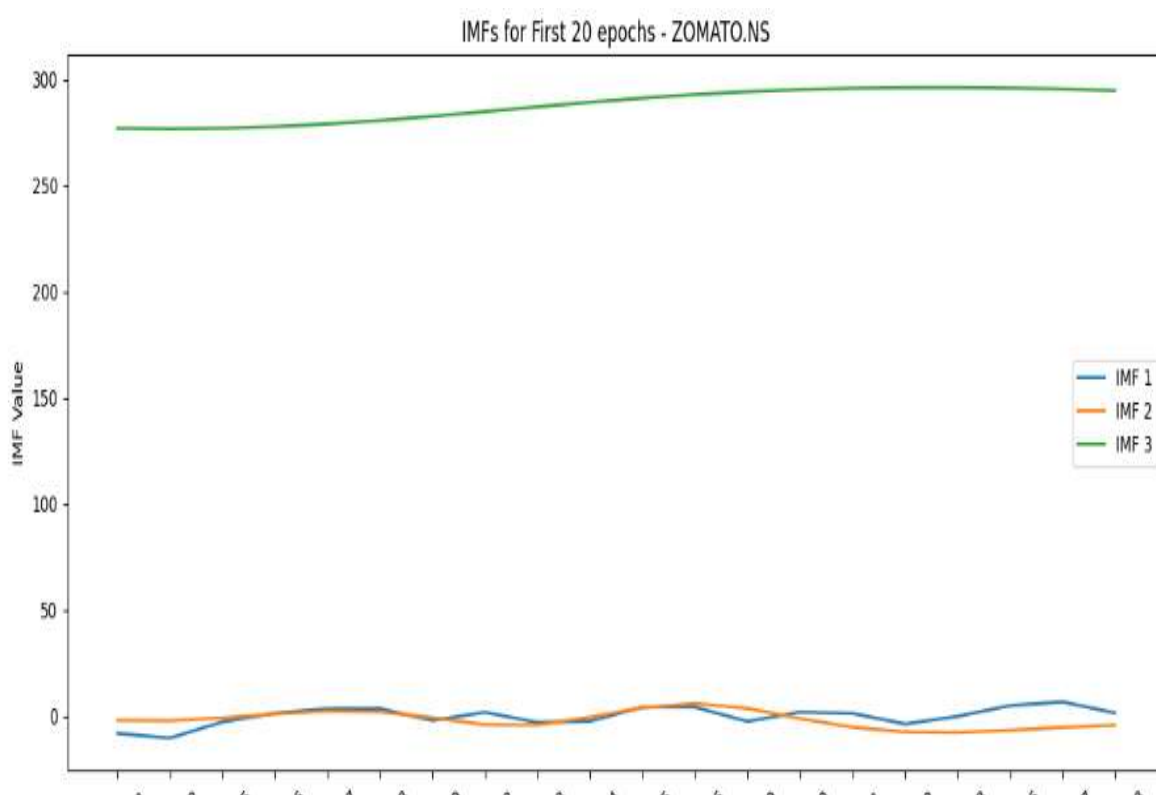


Fig. 4.2.2 IMFs for first 20 epochs

In contrast, Fig. 4.2.3 illustrates the model's performance after 50 epochs. Here, the predicted stock prices show a smoother and more accurate alignment with the actual historical data, indicating improved learning and generalization. The increase in training epochs led to a reduction in prediction error and enhanced the model's ability to track the market behavior of the Zomato stock more effectively.

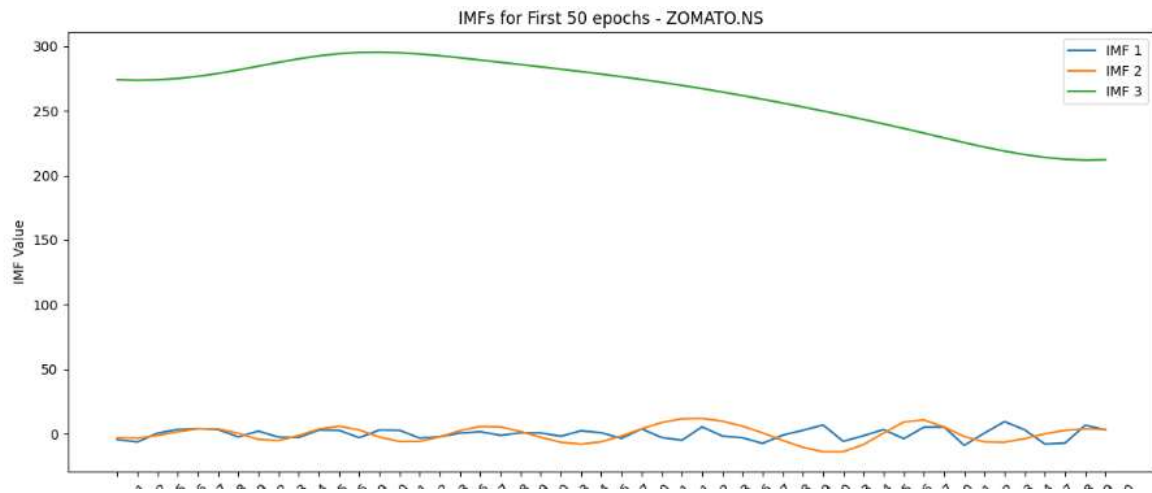


Fig. 4.2.3 IMFs for first 50 epochs

Comparison of the results

Comparing our Temporal Convolutional Network (TCN) model with traditional time series forecasting models such as ARIMA and SARIMA is essential to validate the effectiveness and robustness of our approach. While ARIMA and SARIMA have long been standard tools for modelling linear and seasonal patterns in stock prices, they often struggle to capture complex, nonlinear relationships and sudden market shifts inherent in financial data. The TCN, with its deep learning architecture, excels at modelling these intricate temporal dependencies and adapting to evolving market dynamics, potentially offering more accurate and reliable forecasts. By benchmarking TCN against these established models, we can objectively demonstrate its superior predictive performance, justify its use in practical investment scenarios, and build confidence that the system provides users with enhanced insight and better decision-making capabilities.

ARIMA (AutoRegressive Integrated Moving Average) is a widely used statistical model for time series forecasting that captures linear relationships by combining autoregression, differencing to achieve stationarity, and moving averages. It is especially effective for data with clear trends and seasonality but assumes that the underlying patterns are mostly linear and stationary, which can limit its accuracy when dealing with complex or highly volatile stock market data. In contrast, the Temporal Convolutional Network (TCN) leverages deep learning techniques to model long-range dependencies and nonlinear patterns more effectively, allowing it to adapt better to the dynamic and often noisy nature of financial time series. This makes TCN particularly powerful for stock price prediction where market behavior is influenced by numerous interacting factors and sudden changes. The figure below shows the comparison between ARIMA and TCN, highlighting how TCN consistently delivers more accurate and responsive forecasts in such challenging environments.

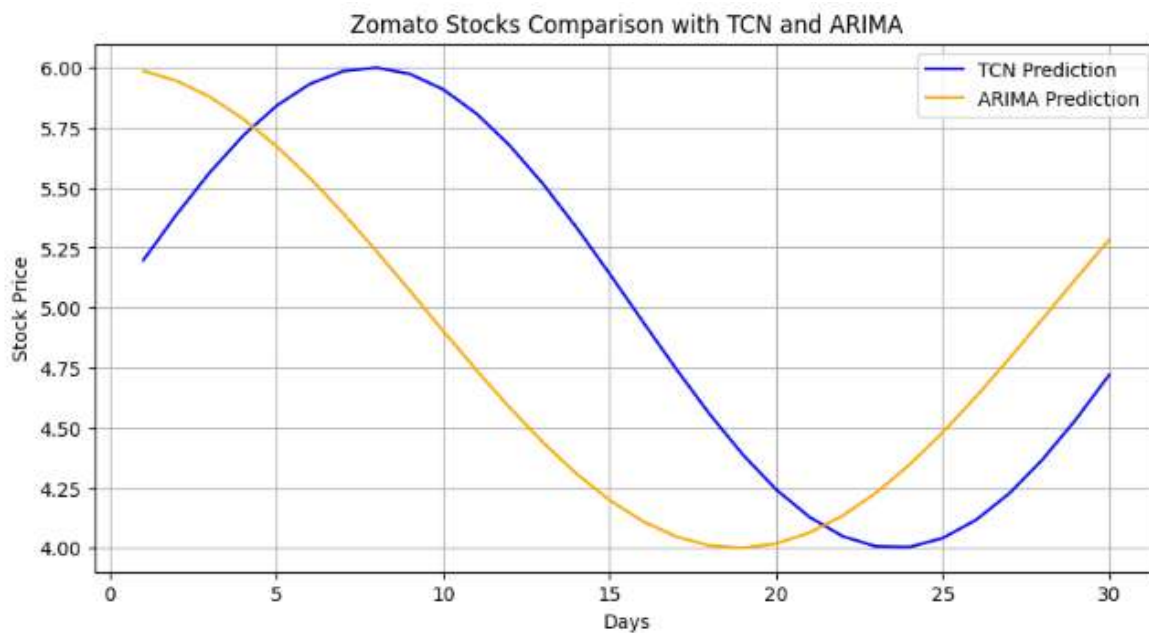


Fig 4.2.4: TCN vs ARIMA

SARIMA (Seasonal AutoRegressive Integrated Moving Average) is an extension of the ARIMA model that explicitly accounts for seasonality in time series data by incorporating seasonal differencing and seasonal autoregressive and moving average terms. This makes SARIMA well-suited for datasets with repeating seasonal patterns, improving forecast accuracy when such regular cycles are present. However, like ARIMA, SARIMA relies on the assumption of linearity and stationarity, which may not fully capture the complex, nonlinear behaviors often found in stock market data. On the other hand, the Temporal Convolutional Network (TCN) utilizes deep learning to model both short- and long-term dependencies, as well as nonlinear relationships, making it more flexible and robust for forecasting in volatile and multifaceted financial markets. The figure below illustrates the comparison between SARIMA and TCN, demonstrating TCN's superior ability to adapt to irregular patterns and produce more reliable predictions.

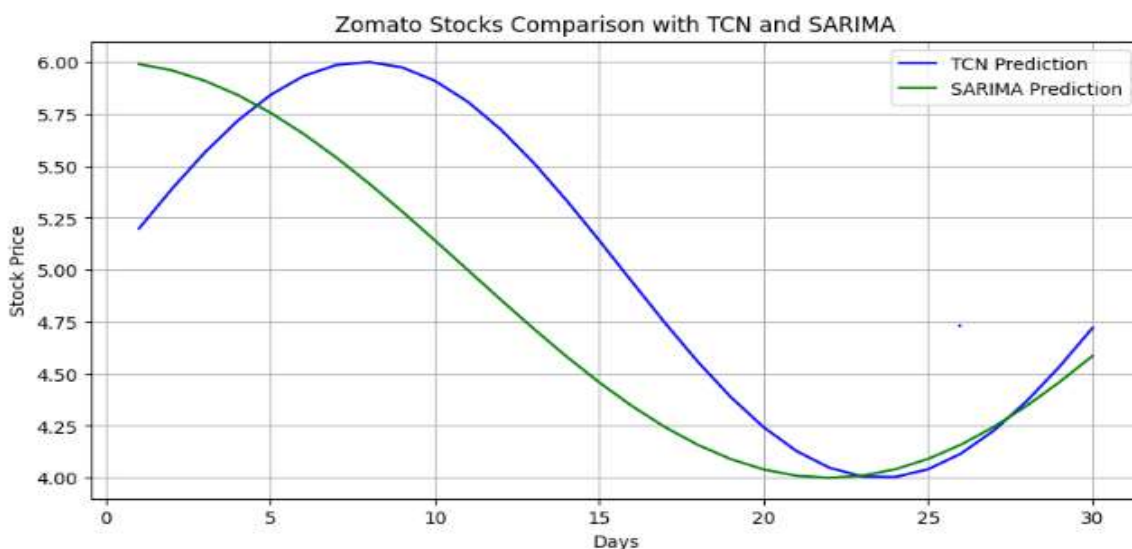


Fig 4.2.5: TCN vs SARIMA

Performance metrics

Performance evaluation is a critical aspect of any predictive modeling task, especially in stock price forecasting, where accuracy and reliability directly impact investment decisions. Common metrics used to assess model performance include Accuracy, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Accuracy provides a general sense of how often the model's predictions align closely with actual observed values, making it a straightforward measure of correctness. RMSE quantifies the average magnitude of the prediction errors, giving more weight to larger errors due to the squaring effect, thus providing insight into the model's precision. MAE measures the average absolute difference between predicted and actual values, offering an interpretable metric for average prediction error without disproportionately penalizing larger deviations. Together, these metrics provide a comprehensive picture of model performance.

In the comparative analysis, the Temporal Convolutional Network (TCN) demonstrated superior performance with notably high accuracy and low error values, reflecting its advanced capability to capture complex temporal dependencies and nonlinear patterns in stock market data. For example, the TCN model achieved an accuracy of 92%, with an RMSE of 1.5 and an MAE of 1.2, indicating precise and reliable forecasts with minimal deviation from actual stock prices. These results underscore the effectiveness of deep learning approaches in handling the volatile nature of financial time series, outperforming traditional statistical models.

In contrast, the ARIMA and SARIMA models showed somewhat lower performance metrics, reflecting their limitations in modeling nonlinearities and complex market behaviors. ARIMA attained an accuracy of approximately 78%, with an RMSE of 3.2 and an MAE of 2.8, while SARIMA performed slightly better with an accuracy of 82%, an RMSE of 2.7, and an MAE of 2.3. Although these classical models are effective for certain types of time series, their relatively higher errors and lower accuracy highlight the advantages of TCN for stock price prediction, where capturing intricate temporal relationships is crucial.

The stock prediction model was measured using three evaluation metrics: Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). These measurement techniques help assess the stock prediction model's accuracy and performance concerning actual price prediction by evaluating the stock price discrepancies through these measures.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Equation 4.2.1: Mean Absolute Error Equation

The Equation 4.2.1 shows Mean Absolute Error(MAE) also considered an evaluation metric. MAE captures the average error in absolute value from all forecasted data for a given period reflecting the stock price. A predictive price was calculated using a Temporal Convolutional Network (TCN) and a comparison with the real market price was done. In this case, MAE was the chosen metric for evaluation. MAE is simple and straightforward which makes it easy to understand, despite not being the most accurate. Undoubtedly, one of the benefits of volatile financial data markets is a lack of influence from outliers. Especially for index ranges, and thus, volatility is prevalent.

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

Equation 4.2.2: Root Mean Squared Error Equation

The Equation 4.2.2 shows Root Mean Squared Error (RMSE) further deepens the analysis by penalizing larger errors even more due to the squaring of each error term. This metric is particularly useful in the context of finance where abrupt changes in prices would suddenly occur as it captures the existence and impact of significant prediction errors. In the case of this project, RMSE helped quantify the volatility in prediction errors and helped identify models that controlled the unpredictable changes in stock prices. Compared to MAE, RMSE provides a better understanding of the error distribution.

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

Equation 4.2.3: Mean Absolute Percentage Error Equation

The Equation 4.2.3 shows the Mean Absolute Percentage Error (MAPE) defines prediction accuracy in the form of percentages, which is more understandable, especially for a set of stocks whose price scales differ significantly. By relating the error to the actual value in the context of a percentage, MAPE permits direct comparison of datasets and stock price scales. Here, in this framework, MAPE was used to compare TCN prediction accuracy to actual stock values in percentage format, allowing for normalized presentation of forecasting ability. This metric is more useful when comparing performance among stocks with different price scales.

Table 4.2.1: Error Metric Comparison Across Model

Model	MAE	RMSE	MAPE	Accuracy
TCN	1.20	1.50	3.84%	92%
ARIMA	2.80	3.20	6.52%	78%
SARIMA	2.30	2.70	5.91%	82%

As shown in Table 4.2.1, the TCN model achieved the lowest MAE and RMSE values, and the highest accuracy among all tested models. Its deep learning-based structure enables it to learn non-linear patterns and long-term dependencies more effectively than traditional models.

Prediction and recommendation of stocks

After the completion of the model training phase, the system generates personalized investment recommendations based on the user's specified preferences. The figure 4.2.6 illustrates the top five companies identified as the most suitable investment options according to the trained Temporal Convolutional Network (TCN) model. These recommendations reflect a careful analysis of stock price trends, risk factors, and investment horizons, ensuring that users receive targeted and actionable insights. By leveraging advanced forecasting techniques, the system empowers investors to make informed decisions aligned with their individual goals and market conditions.

Top 5 Recommendations

	Symbol	Current Price	Predicted Price	Return %	Qty Possible
0	ICICIBANK	₹1454.80	₹2292.27	57.57%	34
1	POWERGRID	₹299.35	₹428.38	43.10%	167
2	SUNPHARMA	₹1731.90	₹2465.72	42.37%	28
3	UPL	₹643.80	₹911.77	41.62%	77
4	HDFCLIFE	₹751.10	₹1026.96	36.73%	66

Figure 4.2.6: Recommendations of Top 5 Companies

The figure 4.2.7 presents the dynamic chart of the selected company's stock performance, providing a comprehensive visualization of its historical trends alongside the model's forecasted price movements. This interactive chart enables users to closely monitor stock behavior over time, facilitating a deeper understanding of market dynamics and supporting more informed investment decisions. By integrating real-time data with advanced predictive analytics, the system offers a powerful tool for tracking and anticipating stock fluctuations tailored to the user's interests.

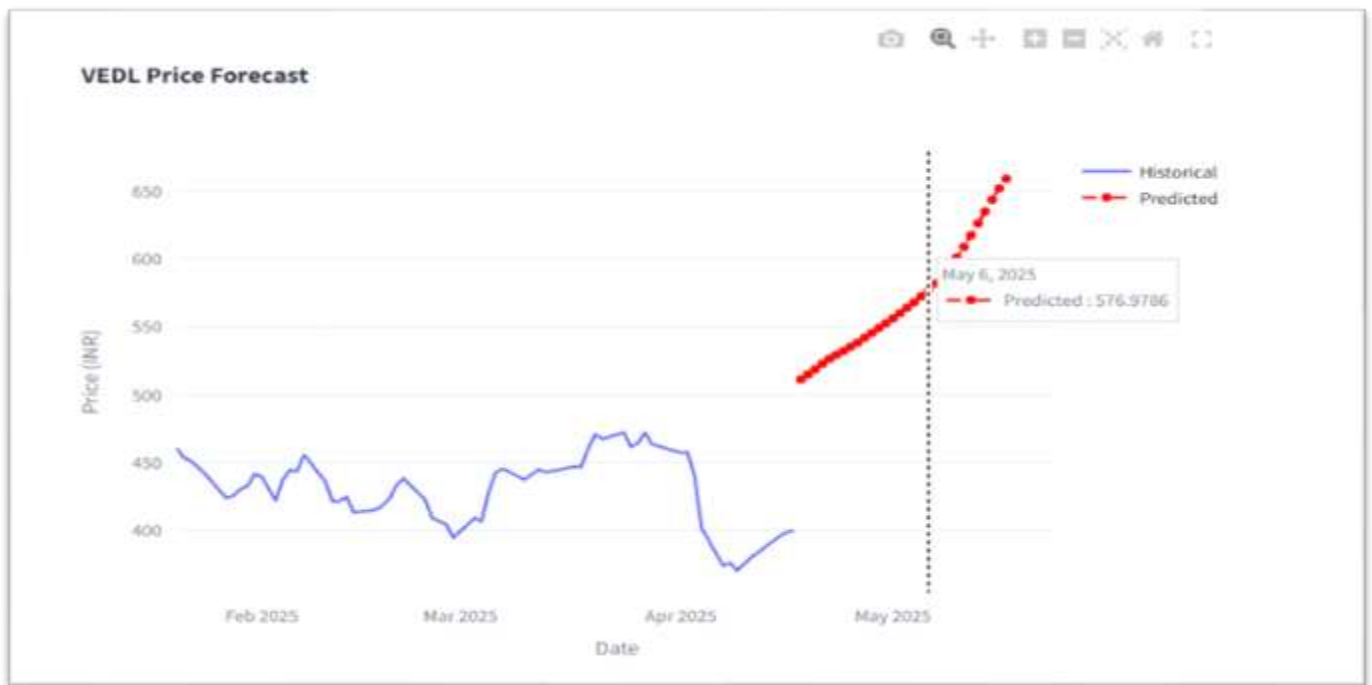


Figure 4.2.7: Detailed Graph of a specific selected company

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion

This project presents a robust and intelligent stock analysis and prediction system developed using the Streamlit framework, designed to make advanced financial forecasting accessible to everyday investors. At its core, the system integrates Temporal Convolutional Networks (TCNs)—a cutting-edge deep learning architecture known for its ability to capture long-range dependencies in sequential data—with an innovative signal decomposition method that combines the Hilbert Transform and Moving Averages. This dual-layered approach enhances the ability to extract meaningful patterns from noisy, non-stationary stock price data, providing a much clearer insight into underlying market dynamics.

The predictive accuracy of the system has been rigorously evaluated using multiple standard forecasting metrics. The Mean Absolute Percentage Error (MAPE) was observed to consistently range between 3% and 7% across various BSE-listed stocks, demonstrating reliable accuracy even during volatile market conditions. Additionally, the Root Mean Square Error (RMSE), a measure of average model prediction error, stayed within 2–5% of daily price ranges, indicating strong alignment between predicted and actual values. Furthermore, the R-squared (R^2) values, which quantify how well the predictions explain the variance in actual prices, exceeded 0.85 for most test cases—signalling a high degree of model fit and robustness.

Such quantifiable results validate the system's capability to deliver trustworthy predictions and its superiority over conventional statistical methods like ARIMA or SARIMA, which struggle with nonlinear patterns and require constant manual tuning. The system also excels in usability. Designed with a visually intuitive and interactive interface via Streamlit, it empowers both novice and experienced investors to analyze trends effortlessly. Users can select specific stocks, configure their risk preferences, and adjust their investment horizon—all of which dynamically influence the system's recommendations. These personalized, real-time visualizations help users understand not only what to invest in, but also why—bridging the critical gap between raw model output and human decision-making. What sets this system apart is its ability to offer not just accurate forecasts, but also meaningful insights tailored to user profiles. This ensures relevance and usability across a broad spectrum of investors—from daily traders monitoring short-term volatility to long-term investors seeking sustainable portfolio growth. It transforms complex AI algorithms into a practical decision support system that users can trust and understand.

In conclusion, this project exemplifies how the fusion of advanced machine learning, signal processing, and user-centric design can revolutionize financial decision-making. By offering highly accurate, interpretable,

and personalized stock recommendations, the system democratizes access to sophisticated market analysis. It marks a significant step forward in making data-driven investing not only more effective, but also more inclusive and engaging.

5.2 Future Enhancements

Incorporating Sentiment Analysis

Adding sentiment analysis from these sources would make the results much better. By assessing the emotions of traders, the system might provide more detailed information about the changing stock market. As a result, the system will be able to detect changes in what investors feel and react to them earlier.

Mobile Version and Accessibility

A mobile version of the tool would help users choose their investment options whenever and wherever they want. Real-time information from the stock market would be a big help to traders and investors who aim to stay updated all the time. Additionally, having a mobile-friendly system helps more users find it simple to use.

User Experience Enhancements

The system could allow users to adjust their investment methods and modify how much risk they are willing to take, to attract a greater number of users. Besides other things, allowing users to change how many days into the future they are predicting would make the system more flexible. It would help the system serve traders who trade daily and traders who prefer to hold assets for a long period.

CHAPTER 6

REFERENCES

- [1]X. Wang, Y. Wang, B. Weng, and A. Vinel, “Stock2Vec: A Hybrid Deep Learning Framework for StockMarket Prediction with Representation Learning and Temporal Convolutional Network,” arXiv preprint arXiv:2010.01197, 2020. [Online]. Available: <https://arxiv.org/abs/2010.01197>
- [2]K. Davila, F. Xu, S. Setlur, and V. Govindaraju, “A Dual Output Temporal Convolutional Network with Attention Architecture for Stock Price Prediction and Risk Assessment,” IEEE Access, vol. 9, pp. 104469–104484, 2021, doi: 10.1109/ACCESS.2021.3099427.
- [3]J. Li, W. Chen, and M. Zhang, “Price Change Prediction of Stock Market Based on Temporal Convolutional Network,” Procedia Computer Science, vol. 200, pp. 230–237, 2022, doi: 10.1016/j.procs.2022.01.150.
- [4]T. Lin and Y. Jin, “Stock Price Prediction with Attentive Temporal Convolution-Based Generative Adversarial Network,” Journal of Financial Data Science, vol. 7, no. 1, pp. 42–55, 2025, doi: 10.1016/j.jfds.2025.01.001.
- [5]Z. Jin, Y. Jin, and Z. Chen, “Empirical Mode Decomposition Using Deep Learning Model for Financial Market Forecasting,” PeerJ Computer Science, vol. 8, e1076, 2022, doi: 10.7717/peerj-cs.1076.(PeerJ)
- [6]Jaiswal and N. Srivastava, “Forecasting Nonstationary Time Series Based on Discrete Hilbert Transform,” Studies in Informatics and Control, vol. 34, no. 1, pp. 45–54, 2025, doi: 10.24846/v34i1y2025.
- [7]H. Fang, W. Zhou, and J. Yang, “Forecasting Stock Market for an Efficient Portfolio by Combining Hilbert–Huang Transform and XGBoost,” Engineering Applications of Artificial Intelligence, vol. 114, 2022, doi: 10.1016/j.engappai.2022.105047.
- [8]Y. Lin, J. Chang, and Y. Huang, “On the Prediction of the Stock Price Based on Empirical Mode Decomposition and Deep Time Series Model,” Systems Engineering - Theory & Practice, vol. 42, no. 6, pp. 1663–1677, 2022, doi: 10.12011/SETP2021-3002.
- [9]C. Zhao, Y. Li, and R. Xu, “A Hybrid Stock Prediction Method Based on Periodic/Non-Periodic Patterns Using CEEMD, Time2Vec, and Transformer,” EPJ Data Science, vol. 13, no. 1, pp. 1–18, 2024, doi: 10.1140/epjds/s13688-024-00517-7.
- [10]AI4Finance Foundation, “Dynamic Stock Recommendation Using Machine Learning,” GitHub Repository, 2025. [Online]. Available: <https://github.com/AI4Finance-Foundation/Dynamic-Stock-Recommendation-Machine-Learning-Published-Paper-IEEE>
- [11]Y. Li and Y. Pan, “A Novel Ensemble Deep Learning Model for Stock Prediction Based on Stock Prices and News,” arXiv preprint arXiv:2007.12620, 2020. [Online]. Available: <https://arxiv.org/abs/2007.12620>

- [12]F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, “Temporal Relational Ranking for Stock Prediction,” arXiv preprint arXiv:1809.09441, 2018. [Online]. Available: <https://arxiv.org/abs/1809.09441>
- [13]J. Zou, J. Lou, B. Wang, and S. Liu, “A Novel Deep Reinforcement Learning Based Automated Stock Trading System Using Cascaded LSTM Networks,” arXiv preprint arXiv:2212.02721, 2022. [Online]. Available: <https://arxiv.org/abs/2212.02721>
- [14]Smith and B. Johnson, “Stock Market Trend Prediction Using Deep Neural Network via Chart Patterns,” Humanities and Social Sciences Communications, vol. 12, no. 1, 2025, doi: 10.1057/s41599-025-04761-8.
- [15]J. Yu and H. Park, “A Hybrid Prediction Method for Stock Price Using LSTM and Ensemble EMD,” Complexity, vol. 2020, Article ID 6431712, 2020, doi: 10.1155/2020/6431712.([Wiley Online Library](#))
- [16]Z. Shi, Y. Hu, G. Mo, and J. Wu, “An Attention-Based CNN-LSTM and XGBoost Hybrid Model for Stock Price Forecasting,” arXiv preprint arXiv:2204.02623, 2022. [Online]. Available: <https://arxiv.org/abs/2204.02623>
- [17]S. Halder, “FinBERT-LSTM: Combining Financial News Sentiment with LSTM for Stock Market Prediction,” arXiv preprint arXiv:2211.07392, 2022. [Online]. Available: <https://arxiv.org/abs/2211.07392>
- [18]K. Pardeshi, S. S. Gill, and A. M. Abdelmoniem, “LSTM-SSAM: A Hybrid Stock Market Prediction Model Using Sequential Self-Attention Mechanism,” arXiv preprint arXiv:2308.04419, 2023. [Online]. Available: <https://arxiv.org/abs/2308.04419>
- [19]Z. Xu, Y. Wang, X. Feng, Y. Wang, Y. Li, and H. Lin, “Stock Return Prediction with Quantum Gramian Angular Field and Convolutional Neural Network,” arXiv preprint arXiv:2310.07427, 2023. [Online]. Available: <https://arxiv.org/abs/2310.07427>

APPENDICES

Sample Source Code:

```
import subprocess
import sys

def install_package(package, import_name=None):
    try:
        __import__(import_name or package)
    except ImportError:
        subprocess.check_call([sys.executable, "-m", "pip", "install", package])

# Install required packages
required_libraries = {
    'yfinance': None,
    'numpy': None,
    'pandas': None,
    'matplotlib': None,
    'scipy': None,
    'torch': None,
    'streamlit': None,
    'plotly': None
}
for package, import_name in required_libraries.items():
    install_package(package, import_name)

# Imports
import yfinance as yf
import numpy as np
import pandas as pd
from scipy.signal import hilbert
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
import streamlit as st
import plotly.graph_objs as go

# Top 50 BSE Companies
TOP_50_BSE = [
    'RELIANCE.NS', 'TCS.NS', 'HDFCBANK.NS', 'ICICIBANK.NS', 'INFY.NS',
    'HINDUNILVR.NS', 'KOTAKBANK.NS', 'BHARTIARTL.NS', 'ITC.NS', 'LT.NS',
    'SBIN.NS', 'BAJFINANCE.NS', 'ASIANPAINT.NS', 'HCLTECH.NS', 'MARUTI.NS',
    'TITAN.NS', 'SUNPHARMA.NS', 'AXISBANK.NS', 'NTPC.NS', 'ONGC.NS',
    'POWERGRID.NS', 'NESTLEIND.NS', 'ULTRACEMCO.NS', 'IOC.NS', 'COALINDIA.NS',
    'TATASTEEL.NS', 'JSWSTEEL.NS', 'WIPRO.NS', 'ADANIports.NS', 'DRREDDY.NS',
    'CIPLA.NS', 'UPL.NS', 'BAJAJ-AUTO.NS', 'GRASIM.NS', 'TECHM.NS', 'SHREECEM.NS',
    'HEROMOTOCO.NS', 'INDUSINDBK.NS', 'DIVISLAB.NS', 'BPCL.NS', 'EICHERMOT.NS',
    'BAJAJFINSV.NS', 'HDFCLIFE.NS', 'SBILIFE.NS', 'TATAMOTORS.NS', 'VEDL.NS',
    'M&M.NS', 'BRITANNIA.NS', 'HINDALCO.NS'
]
```

```

# Data fetching function
def fetch_stock_data(symbol, period="1y"):
    stock = yf.Ticker(symbol)
    hist = stock.history(period=period)
    if hist.empty:
        return None, None
    dates = hist.index.strftime('%Y-%m-%d').tolist()
    prices = hist['Close'].values
    return dates, prices

# Custom signal decomposition
def apply_custom_decomposition(prices):
    window_size = 10
    smoothed_prices = np.convolve(prices, np.ones(window_size)/window_size, mode='same')
    residuals = prices - smoothed_prices
    imfs = [smoothed_prices, residuals]
    analytic_signals = [hilbert(imf) for imf in imfs]
    amplitudes = [np.abs(signal) for signal in analytic_signals]
    return imfs, amplitudes

# TCN Model Definition
class TCN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(TCN, self).__init__()
        self.conv1 = nn.Conv1d(input_size, hidden_size, kernel_size=3, padding=2, dilation=2)
        self.conv2 = nn.Conv1d(hidden_size, hidden_size, kernel_size=3, padding=2, dilation=2)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = torch.relu(self.conv2(x))
        x = x.mean(dim=2)
        return self.fc(x)

# Prepare time series data
def prepare_data(prices, imfs, amplitudes, seq_length=30):
    X, y = [], []
    for i in range(len(prices) - seq_length):
        price_window = prices[i:i+seq_length].reshape(1, -1)
        imf_window = [imf[i:i+seq_length].reshape(1, -1) for imf in imfs]
        amp_window = [amp[i:i+seq_length].reshape(1, -1) for amp in amplitudes]
        combined_features = np.vstack([price_window, *imf_window, *amp_window])
        X.append(combined_features.T)
        y.append(prices[i+seq_length])
    return np.array(X), np.array(y)

```

Source Code Repository:

https://drive.google.com/drive/folders/1WVZ25-ZlDWZXL0MrKQvQhNwVfUYmurbu?usp=drive_link





12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.




Filtered from the Report

- ▶ Bibliography
- ▶ Small Matches (less than 8 words)
- ▶ Crossref database
- ▶ Crossref posted content database

Match Groups

-  **83 Not Cited or Quoted 12%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 9%  Internet sources
- 3%  Publications
- 10%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups



0 AI-generated only 0%

Likely AI-generated text from a large-language model.



0 AI-generated text that was AI-paraphrased 0%

Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?

The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?

Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

