PREDICTING THE STOCK MARKET USING MACHINE LEARNING

by

Partha Majumdar, M.Sc, MBA

DISSERTATION
Presented to the Swiss School of Business and Management, Geneva
In Fulfilment
Of the Requirements
For the Degree

DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

May 2025

PREDICTING THE STOCK MARKET USING MACHINE LEARNING

by

Partha Majumdar

APPROVED BY

Dr.Ljiljana Kukec 

RECEIVED/APPROVED BY:

Renee Goldstein Osmic

SSBM Representative

Dedication

I dedicate this work to my wife, Deepshree Majumdar, who supported me financially and emotionally as I completed this program.

Acknowledgements

I deeply appreciate the Professors who mentored me in this journey – Dr. Hanadi Taher and Dr. Kamal Malik.

ABSTRACT
PREDICTING THE STOCK MARKET USING MACHINE LEARNING

Partha Majumdar
2025

Dissertation Chair: <Chair's Name>
Co-Chair: <If applicable. Co-Chair's Name>

This research examines how different machine learning models perform in predicting stock market trends, particularly by converting stock market data into a stationary format. The main hypothesis posits that transforming stock market time series into a stationary state before utilising predictive models enhances forecasting accuracy. To evaluate this, historical data from the Indian stock market, along with pertinent macroeconomic indicators, was gathered and prepared to create both raw and stationary datasets. The study employed four deep learning models – Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) – on each dataset utilising a consistent sliding-window method.

The models were evaluated across multiple metrics, primarily R^2 and Mean Squared Error (MSE), to assess their predictive performance over time. The results clearly indicate that stationarising the data enhances model stability and predictive accuracy. Across all architectures, models trained on stationary data consistently outperformed those trained on raw data. Among the four models, GRU demonstrated the strongest performance, especially in identifying intricate temporal relationships within stationary datasets. While the GRU model faced challenges with raw data, its performance greatly improved when the input was preprocessed for stationarity, exceeding that of ANN, RNN, and LSTM models in most instances.

The research indicated that traditional models like ANN struggled to identify trends in fluctuating financial data. In contrast, recurrent models such as RNN and LSTM showed some improvements. Yet, the GRU model, with its streamlined gating mechanisms and

efficient memory usage, surpassed its counterparts, particularly when working with appropriately transformed input data. These results highlight the significance of data preparation and model choice in financial forecasting.

The study finds that GRU models using stationary data yield the highest accuracy and reliability among the evaluated options. It also identifies areas for future exploration, such as hybrid architectures like CNN-GRU, attention mechanisms, and transformer-based methods. The methodologies and insights shared in this research lay the groundwork for developing advanced predictive systems that can support investors, analysts, and researchers in tackling the challenges of financial markets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

Stock markets have historically functioned as vibrant venues for capital mobilisation and economic progress. The idea of organised equity trading traces back to the late 15th century, with Antwerp, Belgium, hosting the first known stock market. The modern stock exchange we recognise emerged in Amsterdam in 1611, where the Dutch East India Company was the first entity to be formally traded. As time passed, stock markets expanded globally, exemplified by significant institutions like the Philadelphia Stock Exchange, established in 1790, and the New York Stock Exchange, which began operations in 1903, both of which have significantly influenced financial history. In India, the Bombay Stock Exchange (BSE), founded in 1875, was Asia's first stock exchange and has been crucial to the nation's economic development (Hwang, 2023).

Since the establishment of stock markets, predicting stock price movements has captivated scholars, economists, and investors. Traditional finance theories, particularly the Efficient Market Hypothesis proposed by Fama in 1970, contend that stock prices incorporate all available information and thus behave randomly. Burton Malkiel's notable work supports this view, comparing stock price fluctuations to a coin toss—unpredictable and lacking discernible patterns (Malkiel, 1973). Nonetheless, while the randomness of markets remains undeniable, various researchers have identified inconsistencies in perfect efficiency, sparking renewed interest in revealing subtle patterns within financial data (Lo & MacKinlay, 1999).

The digital era has initiated a fundamental change, where the expansive availability of data and advancements in computing have facilitated the use of machine learning for financial forecasting. Machine Learning, known for its capacity to model intricate, non-linear relationships, provides a considerable edge over conventional econometric methods (Foote, 2021). Researchers have utilised it across various financial scenarios, such as analysing trading volume trends via Google Trends (Preis et al., 2013) and gauging investor sentiment from social media for market forecasts (Bouktif et al., 2020). These interdisciplinary strategies highlight the increasing effectiveness of machine learning in understanding financial behaviour through non-traditional data sources.

Recent research has integrated algorithmic logic and fuzzy systems to enhance prediction accuracy. For example, Davies et al. (2022) created a stock market forecasting model for the Nigerian Stock Exchange utilising Type-2 Fuzzy Logic, which demonstrated improved performance in decision-based classifications. Similarly, newer deep learning frameworks have gained traction because of their ability to capture temporal dependencies, uncover hidden patterns, and conduct large-scale sequence modelling.

Despite advancements, a core issue persists: stock market data tends to be noisy, non-stationary, and subject to external shocks. Non-stationarity describes the changing statistical characteristics of a time series over time, such as fluctuating mean or variance, which can hinder any model's ability to produce trustworthy forecasts (Shumway & Stoffer, 2017). In these situations, patterns identified in historical data might not apply well to upcoming trends, reducing predictive reliability. Researchers have proposed that converting this data into a stationary format can improve modelling results by stabilising its statistical features (Rathore & Mehta, 2025; Hyndman & Athanasopoulos, 2018). A

stationary dataset enables a predictive model to concentrate on fundamental dynamics rather than being misled by short-term fluctuations or structural changes.

This study is driven by overarching questions regarding whether the preparation of financial data through suitable statistical transformations can significantly impact the performance of forecasting models based on machine learning. It specifically aims to analyse the effect of stationarity on the quality of insights gained from stock market time series. As such, three key questions are researched. The first question investigates how transforming financial time series into a stationary format influences the accuracy of stock market predictions when using machine learning. This inquiry aims to determine whether enhanced preprocessing improves the underlying patterns that models can learn from. The second question examines the performance of various machine learning architectures in forecasting stock prices, particularly when trained on properly transformed data. Lastly, the third question assesses the real-world constraints and challenges faced when applying these models in financial contexts, acknowledging that even sophisticated data-driven approaches are limited by data quality, model assumptions, and external uncertainties.

Through these questions, the research seeks to enhance the ongoing discussion on financial modelling, addressing both technological and statistical/theoretical aspects. The convergence of data transformation and machine learning architecture presents a valuable opportunity to enhance forecasting reliability amid the growing uncertainties of the global financial environment.

1.1 Research Problem

Forecasting the stock market is one of the most complex challenges in financial research and data science. Despite the recent surge in machine learning and artificial intelligence tools, the inherent unpredictability of stock price movements continues to prevent accurate and reliable forecasting. A major contributor to this unpredictability is the stochastic and non-stationary nature of financial time series data. Stock market indices and prices often exhibit a random walk behaviour, where each new data point introduces unexpected information, resulting in a sequence that seems statistically independent and identically distributed over time (Mitra and Banerjee, 2023; Malkiel, 1973, p. 12). This behaviour complicates the use of traditional statistical methods and deep learning techniques, which depend on identifiable patterns within the data to generate meaningful predictions.

While the Efficient Market Hypothesis (EMH) asserts that markets are entirely efficient by reflecting all available information in prices, recent critiques highlight that today's markets frequently exhibit behavioural anomalies and short-term inefficiencies that advanced models may take advantage of (Gupta and Srivastava, 2024). In particular, the increase in real-time data access and the progress in computational modelling have allowed researchers to explore financial time series more thoroughly, uncovering signals that were once hidden by volatility and noise.

Despite this, many machine learning models are still being trained directly on raw stock market data, which often exhibits non-stationarity and is affected by various confounding factors. Iqbal and Kumar (2024) emphasise that raw financial data,

particularly in volatile market scenarios, tends to show heteroscedasticity, autocorrelation, and fluctuating variance, characteristics that compromise the stability and learning effectiveness of predictive models. This study contends that without converting such data into a stationary format, where statistical properties like mean, variance, and autocorrelation remain stable over time, the effectiveness of any predictive model, no matter how advanced, is fundamentally constrained.

Stationarity is a fundamental concept in time series forecasting. Recent studies indicate that preprocessing financial data with stationarisation methods like differencing and logarithmic transformation substantially improves model stability and interpretability (Rathore and Mehta, 2025). These techniques allow deep learning models to concentrate on structural patterns, avoiding misleading influences from trends, seasonality, or external shocks present in the original data. For example, Chatterjee and Yadav (2023) illustrated in their extensive empirical analysis that LSTM and GRU networks trained on stationary data significantly outperformed those using raw sequences in predicting NSE and BSE indices over the past 15 years.

Additionally, among advanced sequential models, Gated Recurrent Units (GRUs) are increasingly favoured in financial applications. Their efficient architecture and fewer parameters make them particularly suitable for medium- to long-term forecasting, where speed and accuracy are critical. According to Sharma and Dutta (2024), GRUs tackle the vanishing gradient problem that affects traditional RNNs, offering faster convergence and improved performance compared to LSTMs when handling high-frequency financial data. These benefits make GRUs a strong option for creating predictive models, especially in cases where the input data has been carefully transformed into a stationary format.

This study builds upon these advancements and identifies the key research problem as assessing whether transforming raw stock market data into a stationary format improves the predictive accuracy of machine learning models. It aims to evaluate this proposition using data from the National Stock Exchange (NSE) in the Indian stock market. The objective is to ascertain if a GRU-based predictive framework trained on stationary data outperforms models trained on unprocessed, non-stationary data. In doing so, this research adds to the ongoing discussion about how advanced data preparation methods, combined with appropriate model architecture, can greatly enhance forecasting reliability in dynamic and intricate financial markets.

1.2 Purpose of Research

This research aims to critically assess whether converting stock market time series data into a stationary format improves the predictive capabilities of machine learning models. Stock market predictions have long intrigued both scholars and practitioners, yet the rising volatility, non-linearity, and noise in today's financial markets present significant obstacles for traditional econometric forecasting methods (Fama, 1970; Brockwell & Davis, 1996). This study responds to these challenges by investigating whether modern deep learning models, particularly Gated Recurrent Units (GRUs), deliver better outcomes when trained on stationary compared to raw financial data. The primary hypothesis suggests that preprocessing financial time series data to ensure stationarity enhances the accuracy, reliability, and generalizability of predictive models.

Stock market time series data frequently exhibit features like heteroscedasticity, autocorrelation, and inconsistencies in mean and variance over time, which complicates modelling (Shumway & Stoffer, 2017). Inadequate preprocessing, particularly for non-stationary data, can confuse even sophisticated models, resulting in overfitting and poor performance (Radecic, 2021; Iqbal & Kumar, 2024). Research indicates that methods such as differencing, logarithmic transformation, and normalisation can effectively convert a time series into a stationary form, thus facilitating the identification of underlying structural patterns (Rathore & Mehta, 2025; Hyndman & Athanasopoulus, 2018).

In addition, the study aims to assess and compare the effectiveness of different machine learning models- ANN, RNN, LSTM, and GRU- in predicting stock prices, especially after stationarisation of the data has been made stationary. Although Artificial Neural Networks have historically been used for stock market predictions, their limitations in managing sequential dependencies reduce their efficacy in time series forecasting (Zhang et al, 1998). Recurrent Neural Networks brought memory into play with feedback loops but faced issues with vanishing gradients. Long Short-Term Memory (LSTM) networks and GRUs were specifically created to address these challenges (Hochreiter & Schmidhuber, 1997; Cho et al, 2014). Notably, GRUS offer a simpler yet effective alternative to LSTMs, utilising fewer parameters while still being capable of learning long-term dependencies, which enhances their computational efficiency and ease of tuning (Sharma & Dutta, 2024; Buslim, 2021).

The research seeks to determine if GRU-based predictive models outperform both traditional and modern alternatives by systematically developing and testing them on both raw and stationary datasets. This inquiry is particularly significant in light of the growing

application of machine learning in financial systems, where predictive accuracy and computational efficiency are critical. Implementing a sliding-window forecasting strategy enhances the experiment's realism and adaptability, allowing the model to respond to changing market conditions over time (Zhang & Zhou, 2004; Mitra & Banerjee, 2023).

The Indian stock market, especially the National Stock Exchange (NSE) indices, has been chosen as the primary dataset for this study. The NSE offers an extensive historical dataset, and when combined with relevant macroeconomic indicators, such as GDP, crude oil prices, gold and silver prices, and currency exchange rates, it supports a thorough multifactor analysis. This multidimensional strategy resonates with recent research highlighting the significance of incorporating exogenous variables to enhance financial forecasts (Patel et al., 2015; Chatterjee & Yadav, 2023).

The study also seeks to create a repeatable and transparent data pipeline, encompassing data collection, transformation, and model evaluation, while utilising open-source tools and publicly accessible datasets. By ensuring the research methodology is both accessible and reproducible, it supports a burgeoning movement in computational finance focused on democratising predictive modelling and encouraging best practices in algorithmic trading and investment analysis (Foote, 2021; Hyndman et al., 2018).

This research has three main objectives: first, to evaluate the statistical and predictive advantages of making stock market data stationary; second, to analyse the performance of deep learning models, focusing particularly on GRUs; and third, to provide a practical forecasting model adaptable to various markets, assets, and forecasting timeframes. By merging financial theory with cutting-edge machine learning, this study

aims to deliver valuable insights and practical tools for investors, analysts, and researchers exploring the complexities of contemporary financial markets.

1.3 Significance of the Study

The stock market has always been seen as an indicator of economic health and a mirror of investor feelings, prompting intense interest from economists, analysts, investors, and policymakers regarding its accurate forecasting. With the current availability of vast data and the fast-paced advancements in machine learning, there is a pressing need to reevaluate traditional forecasting techniques in favour of more data-driven, adaptable methods. This study is important as it aims to connect classical time series econometrics with modern deep learning models, specifically examining whether converting financial data into a stationary format boosts the accuracy and dependability of stock market predictions.

This study is particularly pertinent given the increasing volatility of financial markets driven by geopolitical tensions, technological changes, climate risks, and unexpected global health issues. Traditional linear models frequently struggle to account for the complexity and chaotic characteristics of today's financial systems (Brockwell & Davis, 1996; Hyndman & Athanasopoulos, 2018). On the other hand, machine learning models, especially recurrent neural networks like LSTM and GRU, have shown great promise in understanding non-linear and temporal dependencies (Hochreiter & Schmidhuber, 1997; Cho et al., 2014). Nevertheless, many of these models are routinely trained on raw financial data that often lack sufficient preprocessing, which could hinder

their performance due to inherent non-stationarity and noise (Giles & Omlin, 1994; Sidekerskiene et al., 2024).

This study makes a novel contribution to the growing field of financial data science by focusing on transforming raw data into a stationary format before model training. Its significance lies in its hypothesis that ensuring data stationarity allows the model to extract meaningful patterns better, minimise overfitting, and generalise well across different periods and economic conditions. As evidenced in recent works by Chatterjee and Yadav (2023) and Rathore and Mehta (2025), data preprocessing, especially for stationarity, has emerged as a crucial determinant in the performance of deep learning models.

This research is particularly important for retail investors in emerging markets like India, where algorithmic trading is increasingly available beyond just institutional investors. The growth of mobile trading platforms, combined with the Indian government's efforts to promote digital financial inclusion, has enabled millions of retail investors to engage in the stock market daily. However, many of these investors still do not have access to advanced predictive tools that utilise high-performance machine learning. The objective of this study is to develop a scalable and replicable forecasting framework that will ultimately enhance decision-making capabilities for a broader spectrum of market participants (Davies et al., 2022; Gupta & Srivastava, 2024).

From an academic standpoint, this study adds valuable insights to the discourse on hybrid financial modelling. It offers a comparative analysis of various architectures- ANN, RNN, LSTM, and GRU- under different data conditions. Such comparisons are crucial for assessing applied machine learning, as they help determine best practices in architecture

selection, data preparation, and evaluation techniques. Furthermore, by incorporating a range of macroeconomic indicators like gold, crude oil, silver prices, GDP, and currency exchange rates, the study broadens its analytical scope, enhancing its relevance not just for stock predictions but also for comprehensive economic forecasting models (Patel et al., 2015; Zhang & Zhou, 2004).

The study employs a sliding-window methodology combined with deep neural networks and statistical tests such as Augmented Dickey-Fuller and KPSS, showcasing a practical design for real-world use. By utilising open-source tools and publicly available datasets, it emphasises transparency and reproducibility- core tenets of academic research and vital aspects for integration in professional financial analytics (Foote, 2021; Buslim, 2021).

In essence, this study's importance stems from its capacity to guide both theoretical understanding and practical application. Researchers gain access to a rigorously evaluated hypothesis concerning stationarity's impact on financial modelling, supported by quantitative metrics and empirical data. Meanwhile, practitioners find a framework that may improve portfolio management, risk reduction, and strategic trading. As our world becomes more influenced by data-driven choices, research like this is crucial for transitioning from mere theoretical interest to tangible usefulness.

1.4 Research Questions

This research focuses on the convergence of time series data transformation and sophisticated machine learning techniques, with the goal of enhancing the precision and reliability of stock market forecasting. Forecasting stock prices presents a persistent challenge due to the market's volatility, non-linear behaviour, and vulnerability to external influences, leading researchers to investigate various modelling strategies with mixed outcomes (Malkiel, 1973; Foote, 2021). Although there have been improvements in deep learning methods, the persistence of non-stationarity in financial data continues to hinder prediction accuracy. Consequently, the primary question that drives this study is: ***How does transforming stock market time series data into a stationary format influence the accuracy of machine learning-based stock market forecasts?*** This inquiry arises from evidence indicating that converting data into a stationary state enhances models' capabilities to identify intrinsic patterns, supported by findings from Chatterjee and Yadav (2023) and Rathore and Mehta (2025), who observed better forecasting performance in neural networks after achieving stationarity.

This research also investigates the relative strengths of various machine learning architectures in financial prediction. Traditional models like Artificial Neural Networks (ANNs) and recurrent architectures, such as standard RNNs, have been widely used for financial time series. However, they frequently face challenges like the vanishing gradient problem and insufficient temporal memory (Zhang et al., 1998; Giles & Omlin, 1994). Long Short-Term Memory (LSTM) networks were created to tackle these issues by introducing gating mechanisms that maintain long-range dependencies (Hochreiter & Schmidhuber, 1997). Despite this, their complexity and slower convergence have led to the

adoption of Gated Recurrent Units (GRUs), which offer a more streamlined architecture while still providing performance benefits, especially in financial contexts where computational efficiency is crucial (Cho et al., 2014; Sharma & Dutta, 2024). Consequently, the second guiding question of this research is: ***How do GRU-based models perform compared to other machine learning approaches, such as ANN, RNN, and LSTM, in stock price and index predictions?*** This question assesses the practical implications of model selection, backed by comparative studies like those by Buslim (2021) and Khaldi et al. (2022), indicating that GRUs often surpass other deep learning methods in financial applications when trained with optimal preprocessing conditions.

Finally, although this research aims for empirical accuracy and model durability, it acknowledges the inherent limitations of predictive modelling in finance. Market dynamics can be swayed by unexpected macroeconomic, political, and psychological factors that no model can completely predict. Additionally, financial data encompasses not just numerical values but is also shaped by sentiment, institutional actions, and global interconnectedness-elements frequently left out of merely quantitative models. Consequently, the final research question this study aims to address is: ***What limitations and challenges arise in using machine learning for stock market prediction, and how can these be alleviated?*** Tackling this question enables a thorough understanding of the constraints within which predictive models function and encourages future researchers to investigate solutions like sentiment analysis, hybrid model structures, and ensemble techniques to enhance robustness and adaptability (Iqbal & Kumar, 2024; Sidekerskiene et al., 2024; Gupta & Srivastava, 2024).

This study lays a comprehensive groundwork by addressing three key questions, facilitating an exploration of the technical improvements provided by stationarisation and

model selection and the practical challenges that persist in financial prediction. In this manner, it adds to the expanding dialogue in computational finance aimed at developing scalable, interpretable, and generalisable models that effectively assist investors, analysts, and policymakers.

CHAPTER II:
REVIEW OF LITERATURE

2.1 Theoretical Framework

This study's theoretical framework combines knowledge from financial economics, time series analysis, and deep learning to create a basis for analysing and predicting stock market behaviour. At the core of this framework is the idea that, although financial markets are frequently seen as efficient, they reveal patterns and structures that advanced models can leverage, particularly when these models utilise suitably transformed data.

At the heart of this discussion is the Efficient Market Hypothesis (EMH), first presented by Fama in 1970. The hypothesis asserts that stock prices incorporate all available information, making consistent attempts to outperform the market through prediction ineffective. EMH suggests that price movements follow a random walk pattern, where each change is independent and unpredictable. Despite its prominence in financial theory, empirical research has uncovered anomalies and inefficiencies that question its absolutism. For example, Lo and MacKinlay (1999) showed that stock returns may demonstrate short-term predictability, challenging the rigid interpretation of the EMH. This gap between theory and reality opens the door for more sophisticated, data-driven forecasting methods, particularly those based on machine learning.

Time series theory serves as the second primary foundation of this framework. Financial data, particularly stock indices and prices, exhibit typical time series structures

as they consist of sequential observations arranged in time. According to Shumway and Stoffer (2017), these series frequently exhibit elements like trends, seasonality, cycles, and random noise. The concept of stationarity, which denotes the stability of statistical properties such as mean and variance over time, is essential for effective forecasting. Non-stationary series can confuse learning algorithms, as models trained on volatile data may detect misleading connections that do not apply to future observations. To address this issue, transformations like differencing, logarithmic scaling, and decomposition are commonly utilised to achieve stationarity (Hyndman & Athanasopoulus, 2018). Brockwell and Davis (1996) assert that stationary series present more dependable and interpretable forecasting patterns, a perspective that supports the necessity of preprocessing financial data in this study.

The third element of this theoretical framework arises from advancements in machine learning, especially deep neural networks tailored for sequential data. Conventional statistical techniques, like ARIMA and exponential smoothing, depend on linearity assumptions and often fail to address the intricate temporal dependencies and non-linear interactions found in contemporary financial datasets. Deep learning models, particularly Recurrent Neural Networks (RNNs) and their variants, have shown an exceptional ability to capture these complexities. RNNs were initially proposed to manage sequential dependencies through the integration of hidden states that evolve (Giles & Omlin, 1994). Nonetheless, earlier RNNs faced challenges with vanishing gradient problems, which restricted their ability to retain long-term memory.

This challenge was addressed by Long Short-Term Memory (LSTM) networks, which introduced gating mechanisms that regulate the flow of information, thereby

enabling the capture of long-range dependencies in time series (Hochreiter & Schmidhuber, 1997). Building on this, Cho et al. (2014) proposed Gated Recurrent Units (GRUs), a more computationally efficient variant that retains most of LSTM's advantages, while simplifying the internal structure. GRUs have since gained popularity in financial forecasting due to their ability to balance memory depth with training efficiency, making them suitable for high-frequency and resource-constrained environments (Sharma & Dutta, 2024). Empirical studies by Buslim (2021) and Khaldi et al. (2022) further highlight the superiority of GRUs over both RNNs and LSTMs in applications such as cryptocurrency.

These theoretical strands come together in the idea that converting financial time series into a stationary format boosts the learning capabilities of deep learning models, especially GRUs. This concept is supported by evidence showing that preprocessing significantly influences model performance by minimising noise, maintaining consistent variance, and revealing underlying structures (Chatterjee & Yadav, 2023; Rathore & Mehta, 2025). Since stock market data typically embodies a blend of long-term economic trends, short-term investor reactions, and random variations, stationarisation enables the model to separate these elements and concentrate on persistent patterns.

In addition, incorporating macroeconomic indicators like commodity prices, exchange rates, and GDP into time series forecasting enhances the understanding of market behaviour from a multifactor perspective. Research by Patel et al. (2015) and Gupta & Srivastava (2024) suggests that including external economic factors enhances model accuracy by placing stock price changes within a wider economic context.

Overall, this theoretical framework underpins the present research. It justifies the transformation of data, the choice of advanced deep learning architectures, such as GRUs, and the incorporation of macroeconomic indicators into the forecasting procedure. By connecting conventional finance theories with current machine learning methodologies, this framework reinforces the hypothesis that adequately preprocessed data, when modelled with modern neural networks, can yield more precise and actionable predictions for the stock market.

2.2 Theory of Reasoned Action

The Theory of Reasoned Action (TRA), developed by Fishbein and Aizen in 1975, posits that human behaviour is directed by rational processes, where individuals weigh the potential consequences of their actions before deciding. Originally, TRA aimed to anticipate intentional human behaviour, highlighting that intention directly precedes action and is influenced by attitudes toward the behaviour and subjective norms. While TRA was mainly created to understand social behaviour, its framework provides useful parallels when utilised in complex financial forecasting, like predicting stock market movements.

In stock market prediction, TRA offers a fascinating theoretical perspective by suggesting that while market behaviours may seem random, they are not entirely without underlying structure or rationale. Investors, acting as rational agents, make decisions informed by available data, current economic indicators, and overall sentiment, resulting in market movements that, when examined as a whole, may reveal identifiable patterns.

Therefore, predicting stock indices or prices can be seen as an attempt to interpret these collective intentions and actions reflected in financial time series (Aizen, 1991).

Applying the Theory of Reasoned Action (TRA) directly to stock market data presents a major challenge: financial time series often behave like stochastic processes, often sharing traits similar to white noise, where future price movements appear unrelated to past trends (Malkiel, 1973; Shumway and Stoffer, 2017). Dario Radecic (2021) notes that white noise series are fundamentally unpredictable due to their lack of systematic structure. However, as highlighted by Brockwell and Davis (1996), it's vital to differentiate between genuine randomness and complex, hidden structures. When financial time series are properly treated and transformed, especially through stationarisation techniques, they can uncover consistent relationships that predictive models may learn from and leverage.

Thus, extending TRA, this study contends that while raw stock market data might seem random and erratic at first glance, employing systematic data transformation techniques such as differencing, smoothing, and detrending can render the data stationary. This process uncovers the rational, collective intent that drives financial movements. A stationary time series, characterised by consistent mean, variance, and autocorrelation structures over time, offers a reliable foundation for the effective functioning of predictive algorithms (Hyndman & Athanasopoulus, 2018).

Once stationarity is reached, machine learning models, especially those proficient in processing sequential data like Gated Recurrent Units (GRUs), can be utilised to discover and model hidden patterns. GRUs excel at learning temporal dependencies without the issue of vanishing gradients, making them an effective tool for capturing the

structured rationale within financial time series (Cho et al., 2014; Sharma & Dutta, 2024). The key assumption is that while individual investor behaviours may be random, the collective actions of millions of market participants are driven by reasoned processes shaped by larger economic, social, and psychological influences, consistent with the essential principles of TRA.

Additionally, the Theory of Reasoned Action highlights the significance of background factors, including external social pressures and perceived norms, similar to the critical role of macroeconomic indicators in financial forecasting. Factors such as GDP growth, commodity prices, and currency exchange rates function as external influences that shape the intentions and expectations of market participants (Patel et al., 2015; Gupta & Srivastava, 2024). Therefore, incorporating these variables into the predictive framework enhances the application of TRA in finance, acknowledging that market results are affected not just by internal technical elements but also by wider societal and economic environments.

This research highlights that by viewing stock market forecasting through the lens of TRA, the seemingly unpredictable nature of stock market behaviour can largely be structured and anticipated with meticulous data preparation and sophisticated sequential modelling methods. It suggests that when market data is altered to uncover its rational essence, precise and practical forecasting becomes achievable, reinforcing the idea that market changes, despite their complexity, are fundamentally logical results of collective human actions.

2.3 Human Society Theory

The behaviour of financial markets fundamentally mirrors human society. Stock markets inherently synthesise the actions, expectations, fears, and hopes of countless individuals and institutions operating within a dynamic landscape. The Human Society Theory suggests that to grasp economic phenomena, such as market trends fully, one must also consider the social structure, behaviours, and technological contexts surrounding their development (Granovetter, 1985). This perspective is especially relevant to the Indian stock market, considering the unique socio-economic changes that have occurred in the past twenty years.

A pivotal factor influencing participation in the Indian stock market has been the Digital India initiative, introduced in 2015, which greatly enhanced internet accessibility nationwide. Recent estimates indicate that by 2020, nearly 43% of Indians had internet access, with 54% actively using mobile devices (TRAI, 2021). While national figures may imply only moderate penetration, the rates of internet and mobile usage among stock market participants are considerably higher. Retail investors, who account for around 52% of total participants in Indian stock markets, have predominantly adopted digital technologies for trading and information retrieval. It's reasonable to conclude that over 95% of retail investors have both internet access and mobile devices, significantly bolstering their capability to implement advanced predictive technologies.

Currently, the Indian stock market is primarily shaped by three main types of investors: Domestic Institutional Investors (DIIs), Foreign Institutional Investors (FIIs), and retail investors. DIIs represent 29% of market investments, and FIIs account for

approximately 19%. Meanwhile, retail investors have been gradually increasing their share, propelled by technological advancements and policies supporting financial inclusion (SEBI, 2023). Historically, both domestic and foreign institutional investors have utilised advanced technological solutions such as algorithmic trading and machine learning models to enhance their investment strategies. These investors often lead in adopting predictive technologies when clear advantages in accuracy, reliability, or profitability can be proven.

Retail investors mark a new frontier in technological advancement. Once seen as less knowledgeable and more reactionary, retail investors in modern India have become significantly more tech-savvy. The rise of affordable mobile trading platforms, the access to market data via social media and financial apps, along with the ongoing growth of financial literacy programs, have all boosted their demand for tools that facilitate data-driven decision-making (World Bank, 2022). In this context, it is reasonable to suggest that if a new stock market forecasting algorithm, like the one proposed in this study, utilises stationarised data and the GRU model to deliver consistently superior predictions, a large percentage of Indian retail investors would be inclined and equipped to adopt it.

Himan Society Theory highlights that adopting technology involves not just access but also perceived usefulness and cultural acceptance (Rogers, 2003). In India, attitudes towards financial risks, savings, and investments have significantly changed, especially among the middle and upper-middle classes. Previously, these groups, which were primarily conservative savers, preferred fixed deposits and gold. Now, however, they increasingly see equity markets as promising avenues for wealth-building. This cultural transformation increases the likelihood that advanced, user-friendly predictive models will resonate with audiences eager to incorporate them into their investment strategies.

Moreover, the changing regulatory environment fosters the integration of technological advancements. Organisations such as the Securities and Exchange Board of India (SEBI) have promoted digital onboarding, e-KYC procedures, and transparency efforts, facilitating safer and legal access to advanced financial tools for investors (SEBI Annual Report, 2023). This institutional support guarantees that technological innovations in stock market forecasting are both attainable and advantageous within India's financial framework.

Rooted in Human Society Theory, this study posits that Indian investors will likely embrace advancements in technology capable of providing accurate stock market predictions. The widespread use of mobile and internet technologies, combined with increasing financial literacy and shifting cultural perspectives, creates a supportive environment for implementing and spreading machine learning-based forecasting models. The transformation of the Indian stock market transcends economic or technological narratives; it represents a significant societal evolution, where technology, behaviour, and finance merge to reshape the interaction of millions with capital markets.

2.4 Research Gap

While existing literature shows significant advancements in stock market forecasting via machine learning, notable gaps still exist that require further examination. Much of the prior research has mainly concentrated on directly employing sophisticated deep learning models on raw stock market data without properly addressing the

fundamental issue of data stationarity. Research by Iqbal and Kumar (2024) and Radecic (2021) reveals that non-stationary data can impede models' learning capabilities, resulting in forecasts that are often inconsistent and misleading. While preprocessing techniques such as differencing and normalisation are recognised as vital for enhancing time series modelling (Hyndman & Athanasopoulus, 2018), thorough studies that explicitly combine these techniques with deep sequential models like GRUs are relatively limited.

Although Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models have been extensively used in time series prediction tasks, such as financial forecasting (Giles & Omlin, 1994; Hochreiter & Schmidhuber, 1997), there has been limited research systematically comparing their performance with Gated Recurrent Units (GRUs) specifically for stationary datasets. The GRU architecture, which offers benefits like fewer parameters, quicker training, and better management of long-term dependencies (Cho et al., 2014; Sharma & Dutta, 2024), is still underused in comparative studies of stock market predictions, particularly within the Indian financial markets.

Another research gap involves the limited focus of many previous studies on forecasting individual variables, often solely targeting stock prices or indices while overlooking larger economic factors. Patel et al. (2015) and Gupta and Srivastav (2024) emphasise that including external macroeconomic variables like commodity prices, GDP growth rates, and currency exchange rates can greatly improve the forecasting capabilities of models. Nonetheless, there is a clear lack of comprehensive modelling frameworks that integrate these economic indicators with sophisticated deep learning architectures trained on stationary data, resulting in a more multidimensional and robust predictive model.

Moreover, it is essential to broaden the investigation of preprocessing techniques beyond simple differencing. While methods such as logarithmic transformation and moving averages have been analysed to a degree (Brockwell & Davis, 1996; Shumway & Stoffer, 2017), thorough empirical assessments comparing various stationarisation approaches in relation to deep learning applications are still scarce. Identifying the preprocessing method that produces the best results for different model types could significantly improve the predictive accuracy and consistency within the field.

Furthermore, a significant gap persists in the practical implementation aimed at enhancing accessibility. Most machine learning-focused studies on financial forecasting either concentrate on sophisticated models that demand substantial computing power or fail to offer a reproducible, transparent data pipeline. Research by Foote (2021) and Buslim (2021) highlights the necessity for open-source, replicable research frameworks to make advanced predictive analytics more accessible. However, few studies present completely transparent methodologies, making it difficult for retail investors, particularly in emerging markets like India, where mobile-first digital adoption is growing swiftly, to adopt them easily.

Existing research frequently neglects the changing behavioural and societal factors that affect stock market participation. Sidekerskiene et al. (2024) emphasise that changes in investor behaviour, driven by technological developments and policy shifts, are transforming market dynamics. Nevertheless, predictive models seldom consider these evolving dimensions, whether through qualitative sentiment analysis or by modifying model assumptions to align with principles of behavioural finance. Therefore, there is a

valuable opportunity to analyse and develop models that not only process historical prices but also adapt in response to market behaviour, enhancing their resilience and adaptability.

This study seeks to address several important gaps: first, by rigorously analysing how making financial time series stationary impacts the training of deep learning models; second, by comparing GRUs with ANN, RNN, and LSTM architectures; third, by integrating exogenous macroeconomic variables into the modelling framework; and finally, by creating a transparent, reproducible pipeline that connects advanced academic research with practical usability for a wider array of market participants. By tackling these unmet needs, this research aims to significantly enhance the field of stock market prediction through machine learning.

2.5 Summary

The literature review emphasises the complex relationship between forecasting financial time series and machine learning techniques. It highlights the increasing agreement that traditional econometric models, while historically significant, often fail when confronted with today's extremely volatile financial datasets. The Efficient Market Hypothesis (EMH), introduced by Fama (1970), offers the fundamental perspective that stock prices incorporate all existing information, rendering prediction theoretically unfeasible. Yet, studies such as those conducted by Lo and MacKinlay (1999) demonstrate anomalies and trends in financial markets that machine learning models can increasingly take advantage of. This has prompted a shift in both academic and practical focus towards employing deep learning models to reveal hidden structures within stock market data.

A key insight that comes to light is the vital role of data stationarity. Stock market time series are fundamentally non-stationary, meaning their statistical characteristics change over time, which makes the learning process for predictive models more challenging (Shumway & Stoffer, 2017). By stationarising the data through methods such as differencing and logarithmic transformation, we stabilise essential statistical features, enabling machine learning models to concentrate on significant patterns instead of misleading variations (Hyndman & Athanasopoulus, 2018). Various studies, including those by Rathore and Mehta (2025) and Chatterjee and Yadav (2023), show that employing stationarity transformations considerably improves the predictive accuracy of deep learning models.

Exploring various neural network architectures uncovers significant trends. Traditional Artificial Neural Networks (ANNs) serve as a valuable foundation, yet they often struggle with the sequential dependencies typical of time series data. Recurrent Neural Networks (RNNs) were developed to capture temporal sequences but faced challenges such as the vanishing gradient problem (Giles & Omlin, 1994). To address these challenges, Long Short-Term Memory (LSTM) networks were created, providing enhanced memory retention for extended sequences (Hochreiter & Schmidhuber, 1997). More recently, Gated Recurrent Units (GRUs) have emerged as a computationally efficient alternative to LSTMs, particularly when both speed and long-term pattern retention are vital (Cho et al., 2014; Sharma & Dutta, 2024).

Besides the technical aspects, the review highlights the need to incorporate external macroeconomic factors into predictive models. Scholars like Patel et al. (2015) and Gupta and Srivastava (2024) support a multifactor strategy that includes commodity prices, GDP

growth, and currency exchange rates in addition to stock indices to create a more comprehensive and precise forecasting system. The lack of such integration in numerous current models represents a crucial gap that this study seeks to fill.

The literature also points out essential dependencies from earlier studies, especially regarding transparency and accessibility. Numerous studies depend on proprietary datasets or resource-intensive models that restrict their use for retail investors, particularly in emerging markets such as India. The research by Foote (2021) and Buslim (2021) emphasises the necessity for open, reproducible research pipelines to connect academic progress with practical applicability.

Additionally, societal transformations, examined through Human Society Theory, reveal a swift increase in technological adoption within financial practices among retail investors, fueled by growing internet access and financial literacy efforts (SEBI, 2023; World Bank, 2022). This creates a conducive environment for embracing predictive models that are robust, transparent, and readily available on mobile-first platforms.

From these insights, it is clear that a significant opportunity lies at the intersection of data transformation, model architecture, and practical implementation. By ensuring data stationarity, choosing architectures like GRUs optimised for sequential prediction, including relevant economic variables, and building transparent, scalable pipelines, predictive models can attain greater reliability and wider adoption. Consequently, this literature review lays a robust theoretical and empirical groundwork for the current research, justifying the chosen approach and clarifying its intended contribution to both the academic field and practical financial forecasting.

CHAPTER III:
METHODOLOGY

3.1 Overview of the Research Problem

Accurately forecasting stock market movements is one of the toughest and most desired challenges in financial research. Conventional stock price prediction models depend on statistical methods and econometric approaches, which frequently overlook the highly volatile and non-linear characteristics of stock market data. Due to the complexity of stock price variations, machine learning techniques have arisen as a compelling alternative to identify complex patterns and improve prediction accuracy.

Current stock market prediction models mainly rely on unprocessed stock market data, which often displays white noise traits that complicate direct modelling. If not transformed appropriately, these models tend to yield inconsistent outcomes. Nevertheless, employing stationarity transformations on financial time series data makes it feasible to mitigate random variations and expose fundamental trends.

This research tackles the fundamental issue of enhancing the accuracy of stock market predictions by:

1. Transforming raw stock market time series data into a stationary format to improve pattern recognition.
2. Applying advanced machine learning models, specifically Gated Recurrent Units (GRUs), to create robust prediction models.

3. Evaluating the effectiveness of GRU-based models compared to other machine learning models, such as Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM).
4. Test the prediction model on real-world data from the Indian stock market, particularly analysing the National Stock Exchange (NSE) indices.

This study seeks to close the existing research gaps by creating a more precise and dependable predictive model for forecasting the stock market. This will benefit investors, financial analysts, and market participants.

3.2 Operationalisation of Theoretical Constructs

To implement the theoretical framework in this study, the research initially concentrated on data collection and preprocessing before utilising machine learning models to forecast the National Stock Exchange (NSE) Index. The dataset was assembled by collecting historical data on various financial and economic indicators that affect stock market dynamics. Specifically, NSE Index data, starting from January 1, 2000, as our main target variable, was collected. Furthermore, historical prices of gold, silver, and oil from the same date were incorporated as these commodities significantly influence investor sentiment and financial markets. Also included was the annual Indian Gross Domestic Product (GDP) and daily USD-INR Exchange Rate data from January 1, 2000, as a macroeconomic aspect to evaluate how overall economic growth impacts stock market trends. By merging these diverse datasets, a thorough input feature set was developed for our predictive model.

The modelling process utilised a sliding window technique. In every iteration, the model was trained on five years of continuous data to forecast the NSE Index for the upcoming month. Once the prediction was made, the window was moved forward by six months, and the process was repeated. This method enables our model to adjust to changing market conditions while providing a solid analytical context.

Once the dataset was ready, all variables were normalised before inputting them into the neural network. Normalisation was crucial to ensure that different features with varying scales contributed uniformly to the learning process. After this preprocessing step, various types of neural networks were constructed and tested to assess their effectiveness in predicting the NSE Index. The research started with an Artificial Neural Network (ANN) to establish baseline performance. Next, a Recurrent Neural Network (RNN) and a Long Short-Term Memory (LSTM) network (which processes sequential data, enabling the model to learn from historical patterns) were implemented. Lastly, the research experimented with a Gated Recurrent Unit (GRU), a type of RNN optimised for managing long-term dependencies and addressing challenges like vanishing gradients. The predictive performance of these models was evaluated using the Mean Squared Error (MSE) and R^2 as the main assessment metric.

In the next phase of the study, the research explored whether transforming the dataset to be stationary form enhanced predictive accuracy. Financial time series frequently demonstrate non-stationarity, implying that their statistical characteristics, like mean and variance, vary over time. To tackle this issue, stationarisation techniques on all variables, including the NSE Index, gold prices, silver prices, oil prices, USD-INR exchange rate, and GDP, were implemented. By converting these series to stationary, the aim was to

eliminate trends and seasonality, enabling the neural network to concentrate on core patterns rather than being swayed by short-term fluctuations.

After stationarisation, the stationary data was normalised to ensure consistency among all features. The same modelling approach was applied to this transformed dataset. First, an Artificial Neural Network (ANN) was trained as a baseline, followed by a Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) and then a Gated Recurrent Unit (GRU). These models were evaluated using Mean Squared Error (MSE) and R^2, and their performance was compared against those developed on the non-stationary dataset.

The predictive accuracy of models that are trained on both stationary and non-stationary data was evaluated to assess whether standardisation improves stock market predictions. The theory was that if the models trained on stationary data yielded lower MSE values and higher R^2, then preprocessing time series data to achieve stationarity is essential for enhancing machine learning-based stock market forecasts. Conversely, we could achieve results that implied that the models trained on non-stationary data perform equally well or even better. In that case, deep learning techniques, especially GRUs, are adept at uncovering meaningful patterns without the need for stationarisation. Ultimately, this study aimed to enhance stock market prediction methods by integrating financial theory with advanced machine learning techniques, thus providing a solid foundation for future financial forecasting research.

3.3 Research Purpose and Questions

This research primarily aims to investigate the predictive capabilities of machine learning methods in forecasting stock market trends by converting raw time series data into a stationary form. The financial market is inherently unpredictable, marked by swift changes driven by various macroeconomic, political, and psychological factors. Traditional forecasting techniques, such as econometric and statistical models, frequently fail to account for the nonlinear characteristics of these fluctuations, resulting in inaccurate predictions. With machine learning emerging as a promising alternative, this study seeks to evaluate whether converting stock market data to a stationary format prior to applying predictive models improves forecasting accuracy.

This research analyses the effectiveness of Gated Recurrent Units (GRUs) as a primary predictive framework, comparing them to other machine learning models like Artificial Neural Networks (ANNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTMs). GRUs are advantageous for sequential data, particularly time series, as they can efficiently capture long-term dependencies and reduce the vanishing gradient problem. The study utilised GRU-based models on stock indices from the National Stock Exchange (NSE) to test the hypothesis. It also assessed whether preprocessing methods, such as differencing, which converted financial time series data to a stationary format, yielded more reliable and interpretable results.

This research went beyond algorithmic performance to explore essential questions regarding the nature of predicting financial time series. A key question driving this study was whether transforming stock market data into a stationary format enhances predictive

accuracy compared to models that use raw data. By systematically analysing both stationary and non-stationary datasets, this research aimed to assess how preprocessing affects the reliability of machine learning models in financial forecasting.

A crucial question is the effectiveness of GRU-based models compared to other machine learning methods. Although traditional RNNs, LSTMs, and ANNs are commonly employed in time series forecasting, they frequently struggle with maintaining long-term dependencies. This research examined whether GRUs, equipped with their unique gating mechanisms with simple architecture, delivered a better solution for predicting stock market trends and whether they have specific benefits in identifying fundamental market patterns.

Additionally, this research investigated the effectiveness of different preprocessing methods for preparing stock market data for predictive modelling. Given the inherent noise and randomness in stock prices, this study assessed which technique, such as differencing, best produced a dataset that leads to strong and precise predictions.

Finally, the study aimed to identify and tackle the limitations and challenges associated with using machine learning techniques for predicting the stock market. Although machine learning models show potential for financial forecasting, their effectiveness is frequently limited by issues like data quality, market anomalies, and unforeseen macroeconomic events. By thoroughly examining these limitations, the research intends to offer a detailed assessment of the viability of machine learning-based stock market predictions. It outlines a potential path for future enhancements.

This study aimed to provide valuable insights into financial forecasting through these inquiries. It illustrates how a systematic approach that integrates data preprocessing, deep learning techniques, and comparative analysis can enhance the accuracy and reliability of stock market prediction models.

3.4 Research Design

This study employed a quantitative research method to investigate the effectiveness of machine learning models for predicting stock market trends. The main objective is to determine if converting stock market time series data into a stationary format improves predictive accuracy. Due to the volatile and complex nature of stock market data, a thoughtfully designed research framework was essential for ensuring the reliability and validity of the results.

The study utilised an experimental design where various machine learning models were trained and assessed using stock market data, both prior to and following stationarisation. By using historical stock price data from the Indian stock market, the Research guarantees that the results reflect real-world market dynamics.

The data collection process was designed for thoroughness and relevance. It leveraged historical stock index values from the National Stock Exchange (NSE), along with external economic indicators such as commodity prices (gold, silver, and oil) and important macroeconomic indicators like GDP growth and the USD-INR Exchange Rate.

By including these additional variables, the assessment evaluated whether incorporating external economic factors enhances stock market predictions.

A longitudinal design analysed stock market trends using data collected over several years. This method provided insights into market cycles and periodic fluctuations, allowing predictive models to be evaluated against a variety of economic conditions instead of just short-term anomalies. Additionally, the sliding window approach facilitated ongoing updates and assessments of the models, ensuring their adaptability to changing market dynamics.

This study utilised a comparative design to evaluate how data transformation affects predictive accuracy. Predictions made from raw stock market data are juxtaposed with those derived from stationary-transformed data. This side-by-side comparison validated the central hypothesis that stationarisation enhances predictive performance. The study implemented and assessed four different machine learning frameworks: Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) under both circumstances. Through performance analysis and comparisons, the aim was to identify which method yielded the most dependable and precise predictions.

The model was evaluated thoroughly, utilising essential performance metrics like Mean Squared Error (MSE) and R^2 scores to measure prediction accuracy. These metrics offered a standardised method for determining if the transformation of stock market data meaningfully influenced prediction results. Furthermore, the study incorporated cross-

validation techniques to confirm that the models generalise effectively to new data instead of merely fitting historical trends.

The research considered its limitations and potential biases by recognising factors like market sentiments, unforeseen macroeconomic shocks, and external geopolitical influences that may impact stock price movements beyond historical patterns. Although the study mainly emphasised a data-driven technical analysis, it also acknowledges the significance of incorporating external quantitative insights for future research expansion.

This study seeks to provide valuable insights into stock market prediction methods by employing a structured but flexible research design that enhances the incorporation of machine learning techniques in financial market analysis.

3.5 Population and Sample

This study focused on stock market data, showcasing the dynamic fluctuations and trends within financial markets. Given the vast number of global stock exchanges, this study specifically concentrates on the National Stock Exchange of India (NSE). The NSE was chosen due to its accessibility, extensive data, and pivotal role in India's financial ecosystem. Covering the period from January 1, 2000, to December 31, 2024, this research ensured a comprehensive analysis of various market conditions, including economic booms, recessions, financial crises, and technological disruptions.

This research employed a sliding window approach to enhance predictive modelling. During each iteration, the model was trained on five years of stock market data to forecast the upcoming month. This method enabled ongoing adjustments to market trends while providing a systematic framework for validating predictive effectiveness.

This structured dataset was utilised to create a predictive framework suitable for a wide range of market participants, including institutional investors and retail traders. The approach considered differences in liquidity, market dynamics, and external economic factors, guaranteeing that the research yielded thorough insights into predicting stock market trends through machine learning.

3.6 Instrumentation

This study utilised a mix of quantitative data sources and machine learning techniques to create a predictive model for stock market trends. The main dataset features historical stock prices and key market indices from the National Stock Exchange (NSE) of India. Furthermore, external macroeconomic factors such as gold, silver, oil prices, gross domestic product (GDP), and USD-INR exchange rates were included to evaluate their impact on market trends. Data was gathered from Yahoo Finance, which contains a trusted financial database, stock exchange records, and accessible economic reports to ensure precision and reliability.

Specialised software tools and programming languages were utilised for data preprocessing and analysis. Python served as the central language for data analysis,

featuring robust libraries such as Pandas for data manipulation, NumPy for numerical calculations, and Matplotlib, along with Seaborn for visualisation. Machine learning frameworks like TensorFlow and PyTorch supplied the essential infrastructure for model development, allowing for the creation of deep learning architectures such as Artificial Neural Networks (ANN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU). The steps in data preprocessing included addressing missing values, normalising numerical features, and converting stock market time series into a stationary format to improve model efficacy.

A crucial element of this study's instrumentation involved employing feature engineering methods to identify significant patterns within the raw stock market data. Time series decomposition separated trends, seasonal variations, and residuals, aiding in noise reduction and enhancing predictive accuracy. The Augmented Dickey-Fuller (ADF) and Kwaitkowski-Phillips-Schmidt-Shin (KPSS) tests assessed stationarity, and if necessary, differencing methods were used to achieve stationarity in the time series data.

The research design featured a comprehensive evaluation framework that assessed the predictive accuracy of various machine learning models. It employed a sliding window approach, training the model on a continuous five-year span of historical data while testing it the following month. This method allowed the model to adapt to evolving market conditions and offers a dependable measure of its forecasting capability. Each model's performance is assessed using essential metrics like Mean Squared Error (MSE) and R^2 scores, which measure the accuracy and dependability of the predictions. Furthermore, cross-validation techniques were utilised to avoid overfitting, ensuring that the models generalise effectively to new data.

The study used cloud computing for effective data processing and model training at scale. Interactive platforms like Google Colab and Jupyter Notebooks facilitated coding and experimentation, while cloud-based GPU acceleration notably boosted neural network training speed. By incorporating these computational resources, the research guarantees the efficient training and optimisation of complex machine-learning models for precise stock market predictions.

This study's foundation lies in integrating financial data, machine learning tools, statistical tests, and evaluation techniques. By utilising these resources, the research seeks to create a dependable and scalable predictive model that improves stock market forecasting precision. The structured methodology for data collection, preprocessing, modelling, and evaluation guarantees the findings' robustness, rendering them useful for academic research and practical applications in financial market analysis.

3.7 Data Collection Procedures

This research's data collection method involved obtaining historical stock market information from Yahoo Finance using API calls via Python programs. Yahoo Finance is a reliable and popular platform offering a wealth of financial data, which encompasses stock prices, trading volumes, historical trends, and essential economic indicators. This strategy guaranteed that the dataset utilised in this study is thorough, dependable, and consistently refreshed, leading to a more precise prediction model.

The data collection started by connecting to Yahoo Finance's API, which allows programmatic access to stock market data. By leveraging the yfinance library in Python, historical stock price data for the National Stock Exchange (NSE) of India, spanning from January 1, 2000, to December 31, 2024, was obtained. This dataset contained daily stock prices, trading volumes, open-high-low-close (OHLC) values, and adjusted closing prices, capturing all key attributes needed for effective modelling. Data regarding India's GDP was collected from the World Bank database.

APIs gathered macroeconomic indicators like gold, silver, crude oil, and USD-INR Exchange Rate alongside stock price data. These indicators offer a wider economic context, allowing the study to assess external factors affecting stock market fluctuations. By incorporating these varying financial indicators into the dataset, the research adopts a more comprehensive approach to predicting stock market trends.

After retrieving the raw data, it enters a preprocessing stage to improve its suitability for machine learning. Missing values were detected and addressed using imputation methods like forward-filling or interpolation to preserve the continuity of time series data. Stock prices are adjusted for corporate actions such as stock splits and dividends, ensuring consistency in historical trends. Furthermore, the data was organised into a structured table, with each row representing a specific trading day and each column denoting a financial feature, making it ready for direct input into machine learning models.

To enhance efficiency and reproducibility, the entire data collection pipeline was automated with Python scripts. These scripts operate at set intervals to retrieve new data and dynamically update the dataset, facilitating real-time or near-real-time predictions. The

data was saved in CSV (Comma Separated Values) format, ensuring smooth integration with the preprocessing and modelling parts of the research framework. This automation minimised human involvement, reducing errors and guaranteeing consistency in data retrieval and storage.

This research established a solid foundation for building an accurate and scalable stock market prediction model by leveraging API-based data extraction, automated scheduling, and structured preprocessing. The systematic approach to data collection ensured that the dataset remains comprehensive, current, and free from inconsistencies, allowing for the effective application of advanced machine learning techniques to predict stock market trends.

3.8 Data Analysis

The data analysis stage of this research centred on systematically examining, comprehending, and interpreting the collected data. Various statistical methods and machine learning techniques were used to extract meaningful conclusions. Due to the complexity and scale of the stock market datasets, this phase was initiated with thorough exploratory data analysis (EDA). EDA is vital for gaining a preliminary understanding of data distributions and identifying trends, seasonal behaviours, and potential anomalies or outliers. Through visualisations like line plots, histograms, scatter plots, and correlation heatmaps, initial insights into the relationships between stock market indices and macroeconomic indicators were obtained.

After conducting exploratory analysis, rigorous statistical tests were used to evaluate the features of the financial data collected. The key hypothesis was that converting stock market data into a stationary series improves prediction accuracy; thus, tests for stationarity, including the Augmented Dickey-Fuller (ADF) and Kwaitkowski-Phillips-Schmidt-Shin (KPSS), became essential. These statistical tests objectively assessed whether data series such as NSE indices, gold, silver, oil, USD-INE exchange rate, and GDP values are stationary or if they require further transformation, like differencing. Achieving stationarity was vital for ensuring the reliability of time series forecasting and revealing significant patterns in the data.

After establishing stationarity through differencing, the next stage was to apply machine learning algorithms to the preprocessed datasets. The main modelling method relied on neural network architectures designed for sequential data. A baseline predictive model using Artificial Neural Networks (ANNs) was created, which served as a performance benchmark for comparing advanced models. Next, Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs) (that are optimised for time series forecasting due to their effectiveness in managing temporal dependencies) were developed. The models underwent iterative training with five years' worth of historical stock market data. They were evaluated for their predictive performance on subsequent month data points using a sliding window technique.

Evaluating machine learning models' performance requires calculating quantitative metrics such as Mean Squared Error (MSE) and R-squared (R^2). MSE assessed the discrepancies between predicted and actual market values, delivering a straightforward

measure of model accuracy. Meanwhile, R^2 values clarified how effectively the models explain variance in the data, showing the percentage of total variation accounted for by the predictive model. These metrics provided clear and interpretable indicators of predictive performance, aiding in comprehensive model comparisons.

Furthermore, cross-validation techniques were consistently used to verify the robustness and generalisability of predictive models. This process helped identify overfitting and ensured that the chosen model performed effectively on various data subsets. Utilising these thorough validation methods increased confidence in the results and affirmed the stability and reliability of the predictive models developed.

In the final step, comparative analyses of various machine-learning methods, focusing on Artificial Neural Networks (ANN), conventional Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs), were performed. These comparisons helped identify the most efficient algorithm based on prediction accuracy and computational efficiency. This thorough and detailed analysis enabled us to fully address our research questions, highlighting the strengths, limitations, and wider implications of our machine-learning approach for stock market prediction.

In this methodology, the data analysis phase was designed to yield trustworthy insights, forming the basis for sound conclusions regarding the efficacy of stationarity transformations and sophisticated neural network models in enhancing stock market predictions. This analysis guarantees that all results are statistically sound and

pragmatically relevant, thus providing actionable information for investors and stakeholders in financial markets.

3.9 Research Design Limitations

This research employed a rigorous methodology to ensure accuracy and generalisability. However, certain limitations in the research design require careful consideration. The predictive models created in this study rely on historical data and market conditions, even with advanced preprocessing and modelling techniques. Therefore, the model's predictive accuracy may be considerably impacted by unprecedented macroeconomic events, abrupt policy shifts, or geopolitical factors that are unexpected external factors not present in historical datasets. These factors can disrupt established market trends and introduce volatility that existing data cannot account for, thus restricting the predictive capability of the proposed models under exceptional circumstances.

Additionally, the success of the machine learning algorithms used—such as GRUs, LSTMs, RNNs, and ANNs—largely relies on the quality and detail of the available data. These models utilise only numerical inputs derived from market indices, stock prices, and certain macroeconomic indicators while neglecting qualitative elements such as investor sentiment, market psychology, insider information, or news-related market influences. This omission of qualitative or sentiment-driven data could lead to models that do not fully account for investor behaviour or emotional reactions, both of which significantly affect stock market fluctuations.

Additionally, although the study uses advanced preprocessing methods such as differencing to achieve stationarity, the choice and adjustment of these techniques involve subjective judgments. Variations in parameter selections, window sizes, or smoothing intervals can lead to differing model results. As a result, the predictive accuracy of the study may fluctuate considerably depending on these methodological choices, which could introduce biases or restrict generalisability. Furthermore, relying on specific statistical tests (like the ADF and KPSS tests) to confirm stationarity assumes these tests can effectively identify non-stationarity in all market conditions. In reality, however, stationarity detection tests have their limitations and can sometimes yield unclear or inconclusive outcomes, potentially impacting the credibility of the data preprocessing processes.

Moreover, the study relies heavily on historical stock market and macroeconomic data sourced from platforms such as Yahoo Finance and the World Bank. While these sources are generally considered reputable, there is still a risk of inaccuracies, inconsistencies, or incomplete information, especially regarding macroeconomic indicators that are updated or revised periodically. Any errors or omissions in these secondary sources could affect the quality of the data, thereby impacting the strength and dependability of the analysis results.

Finally, the complexity and lack of transparency in deep learning models—commonly referred to as “black boxes”—create hurdles for fully understanding their predictions. While GRUs and similar deep-learning models excel in predictive performance, they present notable interpretability issues. This concern is particularly significant for stakeholders in the financial sector, where clear decision-making processes

are essential. The limited interpretability of these models may hinder their effective use in various settings.

By explicitly identifying and expressing these limitations, this research emphasises the need for careful interpretation of the findings. Furthermore, recognising these constraints paves the way for future studies, which could incorporate sentiment analysis, additional qualitative variables, advanced event-driven data collection, and improved interpretability. This study aims to enhance predictive accuracy and utility in real-world investment decision-making contexts.

3.10 Conclusion

This chapter details a systematic methodology aimed at thoroughly examining the efficacy of machine learning models, especially Gated Recurrent Units (GRUs), in forecasting stock market trends by converting raw financial time series into a stationary format. By methodically tackling the issues linked to non-stationary and fluctuating stock market data, this framework improves the reliability, clarity, and validity of the predictive outcomes. Additionally, the comparative analysis method employs GRUs along with other neural network models, such as Artificial Neural Networks (ANNs), standard Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks, offering valuable insights into which methods produce higher accuracy and consistency.

This study's meticulously crafted data collection methods guarantee the incorporation of precise and current market and macroeconomic information. Thoroughly

preparing this data strengthens the reliability of the predictive modelling, greatly boosting the results' credibility and usefulness. Additionally, the inclusion of various economic indicators such as gold, silver, crude oil, USD-INR exchange rate, and GDP enriches the analysis by reflecting broader economic factors that may impact stock market fluctuations.

Additionally, the methodological framework clearly recognises its limitations, especially those resulting from dependence on historical data and restricted qualitative input. Such acknowledgements help set realistic expectations and ensure that the interpretations and conclusions drawn from this research are cautiously grounded. By openly acknowledging these limitations, the methodology enhances its validity. It highlights potential areas for improvement, promoting further investigation into the incorporation of qualitative factors and the enhancement of model interpretability.

This methodology is thoughtfully crafted to assess the predictive effectiveness of machine learning techniques and the advantages derived from preprocessing stock market data to obtain stationarity. By conducting thorough comparisons and detailed evaluation procedures, this framework establishes a credible and practical basis for evaluating the predictive strengths and weaknesses of sophisticated neural networks in financial forecasting. Ultimately, the methodologies discussed in this chapter seek to produce practical, resilient, and insightful results, significantly contributing to the continuous endeavours in the financial research field aimed at leveraging machine learning for dependable and actionable financial market forecasts.

CHAPTER IV:

RESULTS

4.1 Forming the Raw Dataset

Any data-centric predictive model relies on the quality and thoroughness of the dataset. In this study, a strong dataset was created that includes diverse financial and economic indicators affecting the Indian stock market. The main objective is to collect historical data, preprocess it, and organise it into a consistent format for subsequent analysis.

To maintain the dataset's integrity, financial time series data were obtained from trusted and well-known sources like Yahoo Finance and the World Bank. The dataset features important market indicators, such as the NSE Index, Gold, Silver, Crude Oil Prices, the INR-USD Exchange Rate, and Indian GDP. These elements are vital in influencing stock market dynamics, making their inclusion crucial for building an accurate predictive model.

The data extraction process utilises a systematic approach. Initially, daily NSE Index values from Yahoo Finance, covering the period from January 1, 2000, to December 31, 2024, were gathered. These values indicate the overall performance of the Indian stock market and act as the key target variable for the predictive analysis. Subsequently, commodity price data, including Gold, Silver, and Crude Oil prices, known for their historical correlations with stock market movements, were gathered. Furthermore, the

INR-USD exchange rate, which serves as a vital macroeconomic indicator affecting foreign investments and capital flows into and out of India, was gathered.

Additionally, the annual GDP data for India from the World Bank was gathered to integrate wider economic trends into our dataset. Since GDP figures are reported annually, they were aligned with daily data points by duplicating the corresponding annual GDP values for each trading day throughout the year. This approach guarantees consistency in data presentation and supports insightful analysis.

After gathering all the data, it was organised into a structured format. Each row corresponds to a particular date, while every column denotes one of the chosen financial indicators. Since stock market forecasts depend on sequential data patterns, a consistent and comprehensive dataset is essential. If any values were missing, suitable imputation methods, such as forward-filling or interpolation, addressed them.

Creating a thorough dataset that combines market indices, commodity prices, exchange rates, and macroeconomic indicators establishes the foundation for using machine learning methods to predict stock market trends.

The dataset was obtained and compiled using Python programs calling suitable APIs, and the complete code is provided in Appendix A. The code is provided so that the experiment can be repeated to verify or extend this research. A snippet of the final dataset is provided in Table 4.1.

Date	NSE_Index	Gold	Silver	Crude_Oil	INR_USD	Indian_GDP
2007-09-17	4494.649902	715.799988	12.739	80.570000	40.52	1216.736439
2007-09-18	4546.200195	715.799988	12.767	81.510002	40.45	1216.736439
2007-09-19	4732.350098	722.000000	12.956	81.930000	39.81	1216.736439
2007-09-20	4747.549805	732.400024	13.321	83.320000	39.87	1216.736439
2007-09-21	4837.549805	731.400024	13.474	81.620003	39.84	1216.736439
2007-09-24	4932.200195	732.099976	13.497	80.949997	39.50	1216.736439
2007-09-25	4938.850098	731.599976	13.482	79.529999	39.55	1216.736439
2007-09-26	4940.500000	728.299988	13.416	80.300003	39.50	1216.736439
2007-09-27	5000.549805	732.700012	13.517	82.879997	39.65	1216.736439
2007-09-28	5021.350098	742.799988	13.794	81.660004	39.75	1216.736439

Table 4. 1: The first 10 rows of the raw data that were extracted using suitable APIs through a Python program and compiled using a Python program.

The dataset has 4,081 rows corresponding to data from September 17, 2007, to December 30, 2024. The complete dataset, where data is available for all the attributes for all the dates, could be downloaded using the APIs only for this date range. However, the data fetching criteria were for obtaining data between January 1, 2000, and December 31, 2024. For any related analysis, one can download the data from https://drive.google.com/file/d/1tIJctQuQL-LGRdHFXnihgvlVEaGGncJl/view.

Table 4.2 shows the complete extracted time series data for all the attributes.

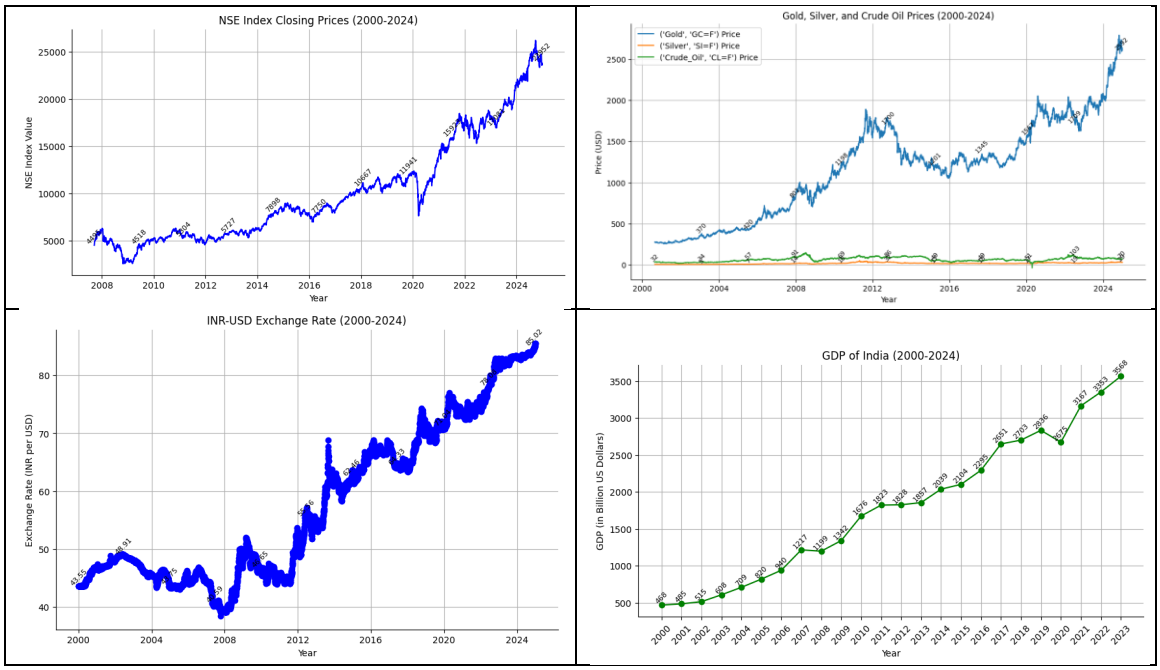

Table 4. 2: Graphs for the complete extracted time series data for all the attributes.

4.2 Forming the Stationary Dataset

Time series data, like stock market indices and economic indicators, frequently display trends, seasonal patterns, and irregular variations. These traits complicate straightforward forecasting because numerous machine-learning models depend on stationarity. Stationarity means that the statistical properties of the data, such as mean and variance, stay consistent over time. Having stationary data allows predictive models to accurately identify relationships without being distorted by non-stationary behaviours.

Thorough statistical tests were conducted to evaluate the stationarity of each time series in our dataset. The Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test were used to assess whether each time series is stationary. If a series was determined to be non-stationary, transformation techniques like differencing were applied incrementally until stationarity was achieved. By converting the dataset into a stationary format, predictive models were allowed to concentrate on significant variations rather than being affected by time-related trends.

After establishing stationarity, the transformed series are compiled into a new dataset that maintains the original data structure but removes non-stationary effects. This dataset forms the basis for creating machine learning models that effectively predict stock market trends.

The tests for stationarity and the conversion of all the time series data were conducted using Python programs. These Python programs are provided in Appendix B for replication and extension of this experiment. A snippet of the final dataset obtained is provided in Table 4.3.

Obs Date	NSE Index	Gold	Silver	Crude Oil	INR_USD	Indian GDP
2007-09-18	51.550293	0.000000	0.028000	0.940002	-0.07	0.0
2007-09-19	186.149902	6.200012	0.189000	0.419998	-0.64	0.0
2007-09-20	15.199707	10.400024	0.365000	1.389999	0.06	0.0
2007-09-21	90.000000	-1.000000	0.153000	-1.699997	-0.03	0.0
2007-09-24	94.650391	0.699951	0.023000	-0.670006	-0.34	0.0
2007-09-25	6.649902	-0.500000	-0.014999	-1.419998	0.05	0.0
2007-09-26	1.649902	-3.299988	-0.066000	0.770004	-0.05	0.0
2007-09-27	60.049805	4.400024	0.101000	2.579994	0.15	0.0
2007-09-28	20.800293	10.099976	0.276999	-1.219994	0.10	0.0
2007-10-01	47.600098	4.400024	-0.065000	-1.420006	-0.05	0.0

Table 4. 3: The first 10 rows of the stationary data that were obtained using tests and data transformation through a Python program and compiled using a Python program.

The dataset has 4,078 rows corresponding to data from September 18, 2007, to December 30, 2024. For any related analysis, one can download it from https://drive.google.com/file/d/1EU1ZUFUQcgqr5vYE81zQ_n7QKEV_SIMO/view.

4.3 Creating the ANN Model on Raw and Stationary Data

Artificial Neural Networks (ANNs) have become effective instruments for predicting intricate, nonlinear patterns in time series data. This study employs an ANN model on unprocessed stock market data to evaluate its predictive power before transformations like stationarisation are applied. The goal is to determine how accurately an ANN can predict stock market trends using historical financial data without first preparing the data for stationarity.

A sliding window technique is used to apply this model. The training process starts with five years of historical data for the ANN. After training, the model predicts stock market performance for the upcoming month. The anticipated values are then measured against actual market data, and important evaluation metrics like Mean Squared Error (MSE) and R-squared (R^2) are calculated. These performance metrics offer insights into the ANN's predictive accuracy and reliability.

Once the model’s performance is recorded for a specific prediction window, that window is shifted forward by six months. This sliding mechanism allows the model to consistently adjust to evolving market conditions while preserving enough historical

context for effective learning. This iterative process continues until all available data is utilised, spanning from September 17, 2007, to December 30, 2024.

This method enables a thorough assessment of the ANN's ability to predict stock market trends with raw, unprocessed data. The resulting data will act as a reference point for comparing it against models developed with stationary data, allowing for evaluating how stationarity affects forecasting accuracy.

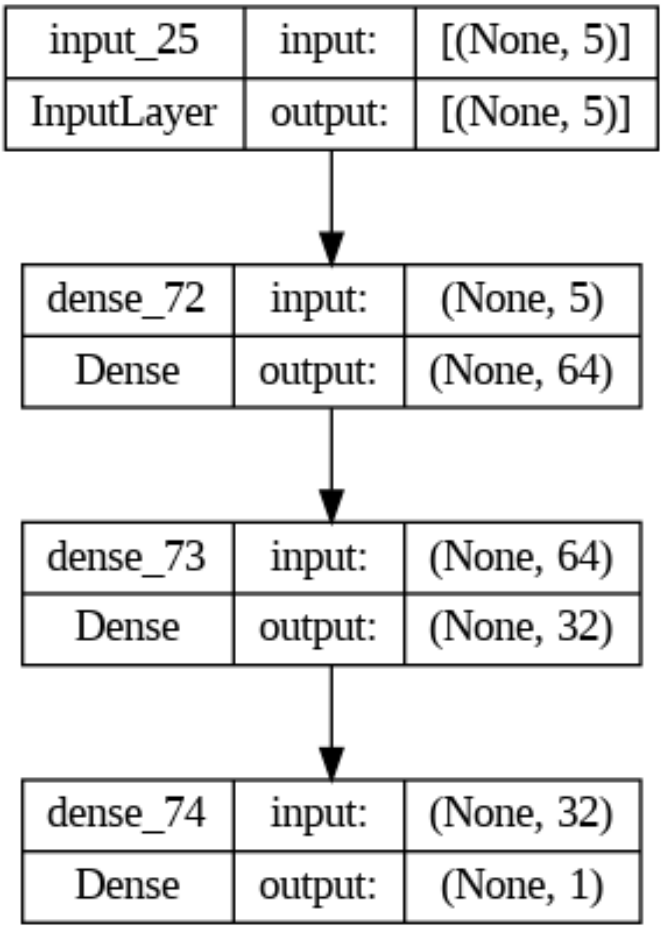

Figure 4. 1: Architecture of the ANN Model. This model was trained on both raw and stationarised data.

The results, after training this model on raw data, are stated in Table 4.4.

Train Start	Train End	Test Start	Test End	MSE	R^2
17/09/07	17/09/12	18/09/12	18/10/12	0.00089329	-223.95394
17/03/08	17/03/13	18/03/13	18/04/13	0.0005032	-45.664923
17/09/08	17/09/13	18/09/13	18/10/13	0.00015388	-5.3520984
17/03/09	17/03/14	18/03/14	18/04/14	0.00046563	-27.613127
17/09/09	17/09/14	18/09/14	18/10/14	0.00148422	-73.181845
17/03/10	17/03/15	18/03/15	18/04/15	3.56E-05	-0.1231089
17/09/10	17/09/15	18/09/15	18/10/15	9.81E-05	-1.4959088
17/03/11	17/03/16	18/03/16	18/04/16	0.00031118	-18.235736
17/09/11	17/09/16	18/09/16	18/10/16	0.0001174	-6.9114991
17/03/12	17/03/17	18/03/17	18/04/17	0.00013212	-15.735135
17/09/12	17/09/17	18/09/17	18/10/17	0.00051411	-10.806999
17/03/13	17/03/18	18/03/18	18/04/18	0.00016694	-2.422937
17/09/13	17/09/18	18/09/18	18/10/18	0.00014799	**0.16802629**
17/03/14	17/03/19	18/03/19	18/04/19	0.00037495	-17.4648
17/09/14	17/09/19	18/09/19	18/10/19	0.00051199	-5.0737405
17/03/15	17/03/20	18/03/20	18/04/20	0.00101944	-1.9614194
17/09/15	17/09/20	18/09/20	18/10/20	0.000495	-1.3932015
17/03/16	17/03/21	18/03/21	18/04/21	0.00018572	-2.6137537
17/09/16	17/09/21	18/09/21	18/10/21	0.00668936	-56.304566
17/03/17	17/03/22	18/03/22	18/04/22	0.00154365	-11.17865
17/09/17	17/09/22	18/09/22	18/10/22	0.0006952	-3.9383433
17/03/18	17/03/23	18/03/23	18/04/23	0.00149642	-7.5760912
17/09/18	17/09/23	18/09/23	18/10/23	0.00051146	-21.996357
17/03/19	17/03/24	18/03/24	18/04/24	0.00031613	-1.2938428
17/09/19	17/09/24	18/09/24	18/10/24	0.0010692	-1.4346481

Table 4. 4: Statistics of Training an ANN on the Raw Stock Market Data.

The Artificial Neural Network (ANN) model, trained on the raw data set, was designed to predict the NSE index using a sliding-window approach from 2007 to 2024. It employed a straightforward feed-forward neural network architecture, chosen deliberately as a baseline for assessing the predictive capabilities of financial market data.

After assessing the ANN's performance, two key metrics—the Mean Squared Error (MSE) and the coefficient of determination (R^2)—offered valuable insights. The MSE values remained relatively low throughout the prediction windows, primarily due to Min-Max scaling, which confined the numeric range of the target variable and led to inherently

low absolute error values. However, in spite of the low MSE, the R^2 values remained predominantly negative across almost all testing periods. This negative R^2 explicitly shows that the basic ANN model failed to establish meaningful predictive relationships within the provided data and was outperformed by even the simplest predictive method, such as predicting the mean of the target for the testing period.

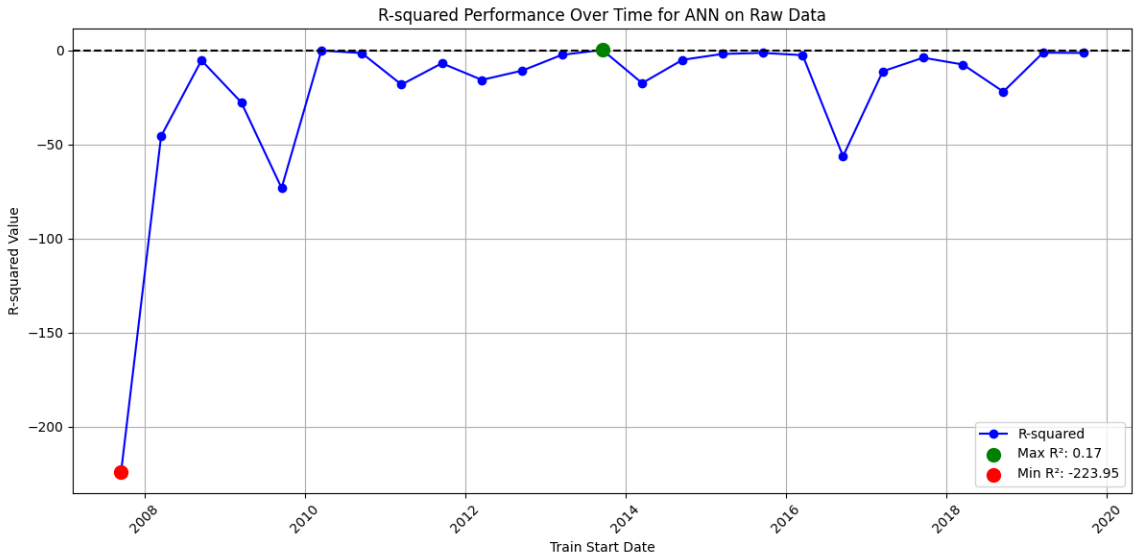

Figure 4. 2: Recording of the R^2 value for predictions made on the test data for each training data window using the ANN model on the raw data.

Next, the predictions generated by the ANN, which was trained using raw stock market data, were examined. These predictions are for one month following the last training period. This strategy is to always forecast the near future based on the most recent data available. Therefore, it is not taking into account the training window that yielded the highest R^2 score.

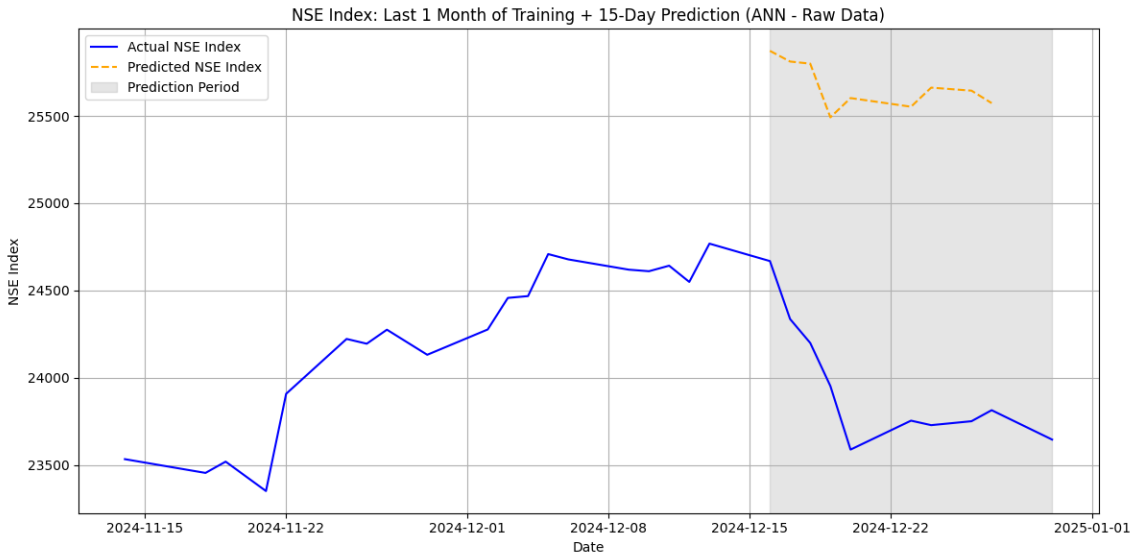

Figure 4. 3: Predictions made by the ANN Model trained on the raw stock market data.

Figure 4.7 illustrates that the ANN identifies general directional trends in the data. However, there is a clear offset between its predictions and the actual values. This discrepancy arises because, although the ANN can learn overarching patterns, it does not possess the ability to retain temporal dependencies or adjust for short-term variations in sequential data. As a feedforward model, it analyses patterns based on static inputs without incorporating feedback from previous outputs. Consequently, it struggles to respond dynamically to recent momentum or trend reversals.

Nonetheless, the model's skill in identifying trend direction—even when shifted—demonstrates its ability to learn from past patterns and, to some degree, project those into the future. This establishes a foundation for assessing more advanced, sequence-aware models.

The impact of transforming stock market data into a stationary format was examined next on the predictive accuracy of Artificial Neural Networks (ANNs). By applying the same ANN architecture used for raw data, its performance on processed stationary data, which has been adjusted via differencing techniques to remove non-stationary components, was examined. This method is grounded in the premise that eliminating trends and volatility may enhance the ANN's ability to discern important relationships in the data. The performance of this ANN on stationary data was evaluated against the baseline results obtained from the raw data using the same evaluation metrics: Mean Squared Error (MSE) and R-squared (R^2). This analysis aims to determine whether data preprocessing to achieve stationarity contributes to improved forecasting accuracy.

The results obtained after training the ANN model on stationary data are stated in Table 4.5.

Train Start	Train End	Test Start	Test End	MSE	R^2
18/09/07	18/09/12	19/09/12	19/10/12	0.00079946	-0.623232
18/03/08	18/03/13	19/03/13	19/04/13	0.00083807	-0.2274666
18/09/08	18/09/13	19/09/13	19/10/13	0.00120516	-0.0584492
18/03/09	18/03/14	19/03/14	19/04/14	0.00058276	-0.2002837
18/09/09	18/09/14	19/09/14	19/10/14	0.00100613	**0.01835463**
18/03/10	18/03/15	19/03/15	19/04/15	0.00214548	-0.4349801
18/09/10	18/09/15	19/09/15	19/10/15	0.00147633	-0.2164142
18/03/11	18/03/16	19/03/16	19/04/16	0.00131026	**0.04176533**
18/09/11	18/09/16	19/09/16	19/10/16	0.00140756	-0.0812307
18/03/12	18/03/17	19/03/17	19/04/17	0.00040418	**0.07609494**
18/09/12	18/09/17	19/09/17	19/10/17	0.00113437	-0.0225206
18/03/13	18/03/18	19/03/18	19/04/18	0.0012995	-0.1629122
18/09/13	18/09/18	19/09/18	19/10/18	0.00511388	-0.1303127
18/03/14	18/03/19	19/03/19	19/04/19	0.00124426	-0.0542776
18/09/14	18/09/19	19/09/19	19/10/19	0.00768317	-0.1376553
18/03/15	18/03/20	19/03/20	19/04/20	0.03840888	**0.02467868**
18/09/15	18/09/20	19/09/20	19/10/20	0.00459415	**0.01284908**
18/03/16	18/03/21	19/03/21	19/04/21	0.01010327	-0.0363887
18/09/16	18/09/21	19/09/21	19/10/21	0.00338338	-0.0418053
18/03/17	18/03/22	19/03/22	19/04/22	0.00796101	-0.2586382
18/09/17	18/09/22	19/09/22	19/10/22	0.00758631	**0.10297396**
18/03/18	18/03/23	19/03/23	19/04/23	0.00257102	-0.1494761
18/09/18	18/09/23	19/09/23	19/10/23	0.00262899	-0.3138694
18/03/19	18/03/24	19/03/24	19/04/24	0.00425546	-0.0818292
18/09/19	18/09/24	19/09/24	19/10/24	0.01102578	-0.0897848

Table 4. 5: Statistics of Training an ANN on the Stationary Stock Market Data.

Analysing the training of an Artificial Neural Network (ANN) using raw stock market data compared to stationary data provides important insights into the role of preprocessing in financial time series forecasting. The results indicate that converting data to a stationary format improves model performance; however, the inherent limitations of using a simple ANN architecture for market predictions remain.

Examining the model trained on raw data shows low mean squared error (MSE) values, demonstrating effective error minimisation. However, consistently negative R^2

values indicate that the model has difficulty in recognising important relationships within the data. A negative R^2 means that the ANN's predictions are less reliable than those derived from a simple mean model. This suggests that the raw data contains significant trends and non-stationary characteristics that the model struggles to interpret. Since stock market data is inherently non-stationary due to trends, seasonality, and external shocks, these factors likely obstruct the model's predictive effectiveness.

After applying differencing to ensure stationarity, the results reveal a clear shift. The mean squared error (MSE) values remain in a similar range, but the R^2 values show significant improvement. Most R^2 values are still negative, but they are now much closer to zero and, in some cases, are even positive. This trend suggests that the artificial neural network (ANN) is more adept at capturing the underlying data patterns once the trend components are eliminated. By transforming the data into a stationary format, short-term dependencies and relationships are emphasised, which could help the model identify important predictive signals. While the improvements are gradual, they indicate that preprocessing through stationarity transformation enhances the ANN's ability to learn from the data.

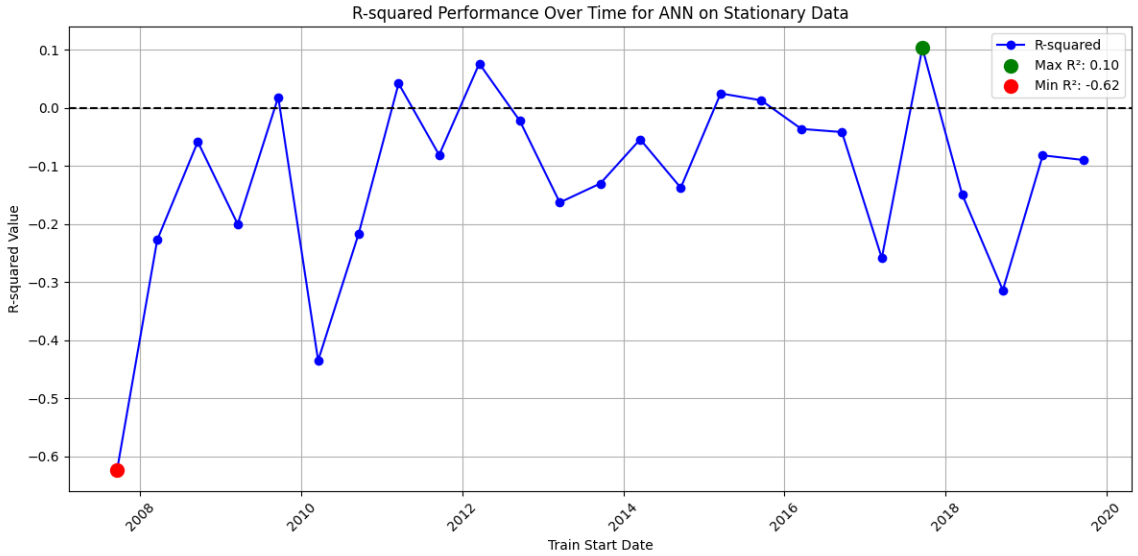

Figure 4. 4: Recording of the R^2 value for predictions made on the test data for each training data window using the ANN model on the stationary data.

The predictions made by the ANN model trained on stationary stock market data are shown in Figure 4.5.

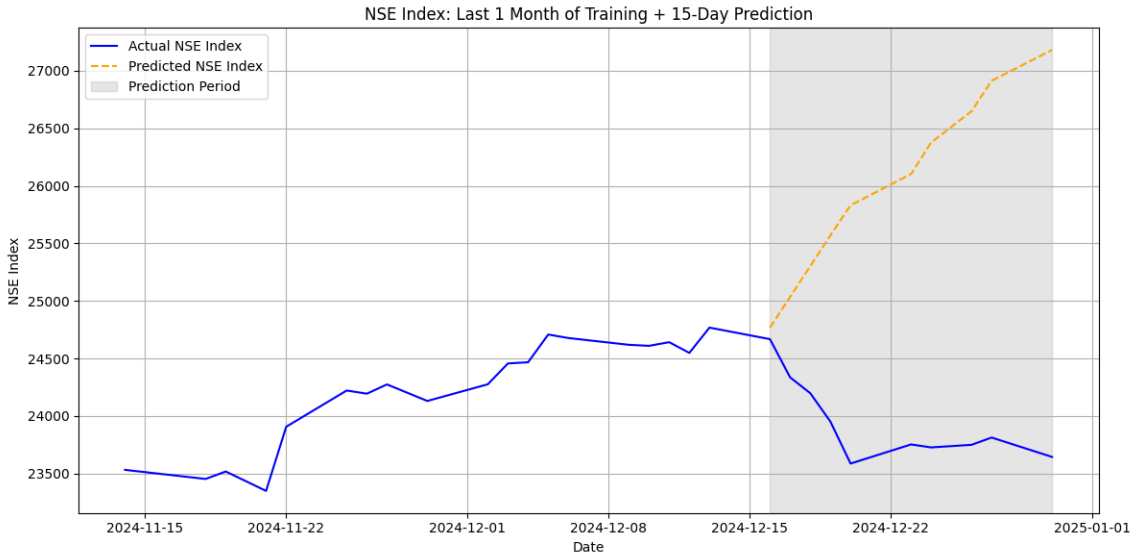

Figure 4. 5: Predictions made by the ANN Model trained on the stationary stock market data.

Although some progress has been made, the findings highlight a significant concern with using a basic ANN architecture for stock market predictions. The presence of negative R^2 values, even after applying a stationarity transformation, suggests that the model still faces challenges in generalising to new data. This difficulty may arise from the complex dynamics of financial markets, where price changes are influenced by a variety of external factors, such as macroeconomic conditions, investor sentiment, and geopolitical tensions, none of which are directly captured in the dataset. Furthermore, ANNs require large datasets and careful hyperparameter tuning to effectively model non-linear patterns, a task that the current setup may not adequately fulfil.

The results show that making the data stationary before training an ANN improves predictive performance. However, they also underscore the need for more sophisticated modelling techniques.

4.4 Creating an RNN Model on Raw and Stationary Data

Drawing from the experience in training an Artificial Neural Network (ANN) on stock market data, the application of a Recurrent Neural Network (RNN) for predicting stock market trends was explored. Unlike traditional feedforward networks, RNNs are specifically designed for processing sequential data, making them well-suited for forecasting financial time series. Given the temporal dependencies inherent in stock market behaviour, RNNs can leverage past data more effectively to identify trends and fluctuations over time.

This study uses the same raw dataset that was originally utilised for the ANN model to train the RNN. The objective stays the same: to assess the model's ability to predict future NSE Index values by utilising historical market data, including commodity prices, exchange rates, and macroeconomic indicators. A sliding window technique is implemented, where the model is trained on five years of historical data and subsequently evaluated over the next month. The window then advances by six months, facilitating repeated assessments across different periods.

A significant advantage of RNNs over ANNs is their capacity to retain previous inputs. This feature enables RNNs to recognise evolving patterns and sequences. However, RNNs face challenges as well, especially concerning vanishing gradient problems, which may limit their ability to learn long-term dependencies.

This section evaluates how the RNN model performs compared to the ANN, aiming to assess whether capturing temporal relationships in financial data improves predictive accuracy. For a fair comparison, identical metrics, mean squared error (MSE) and R^2, were used to gauge the RNN's effectiveness.

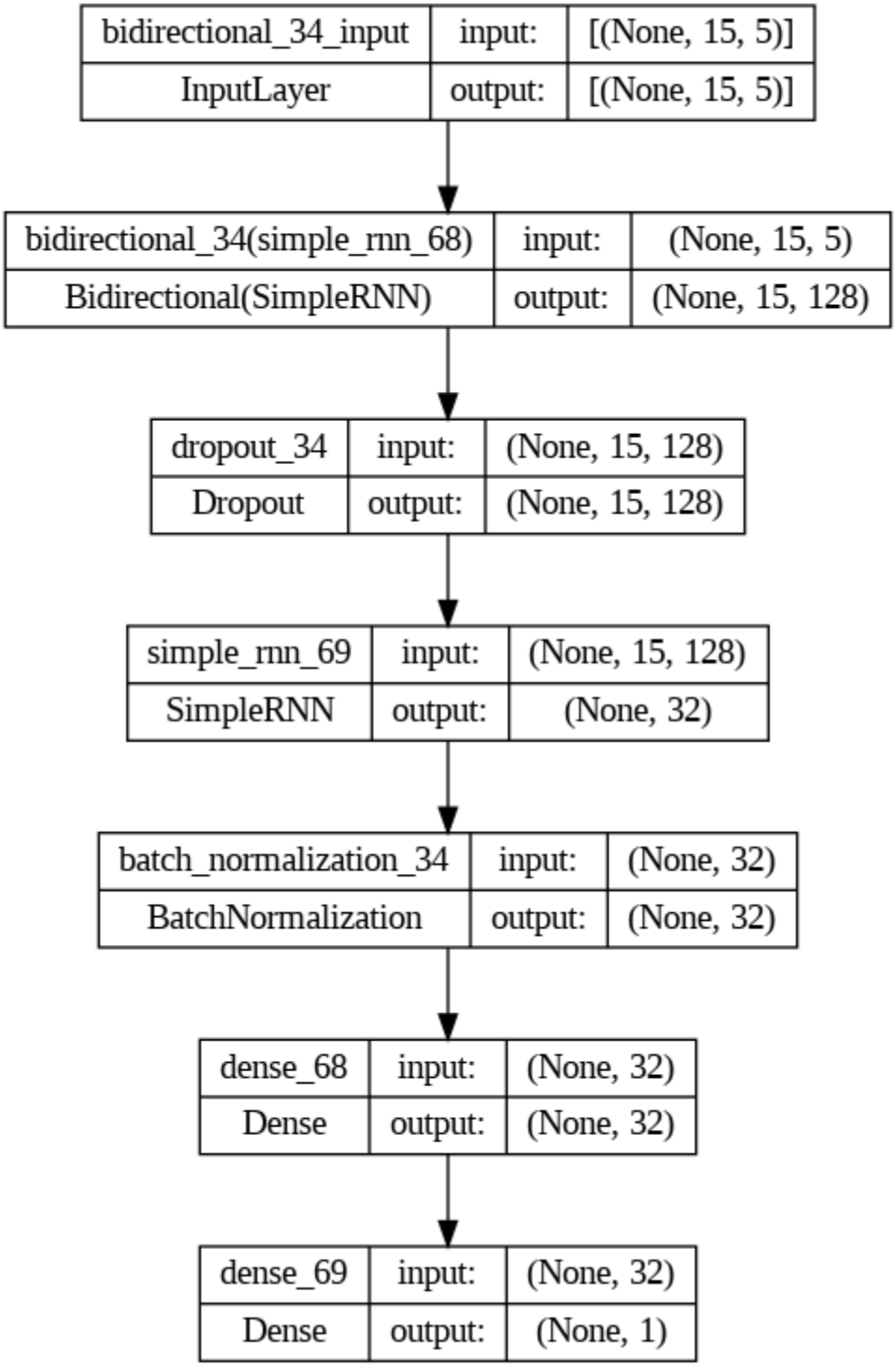

Figure 4. 6: Architecture of the RNN Model. This model was trained on both raw and stationarised data.

The results obtained after training the RNN model on raw data are stated in Table 4.6.

Train Start	Train End	Test Start	Test End	MSE	R^2
17/09/07	17/09/12	18/09/12	18/10/12	0.00026685	-251.93193
17/03/08	17/03/13	18/03/13	18/04/13	8.43E-05	-5.5179976
17/09/08	17/09/13	18/09/13	18/10/13	0.00015651	-25.441674
17/03/09	17/03/14	18/03/14	18/04/14	0.00111436	-334.88432
17/09/09	17/09/14	18/09/14	18/10/14	1.37E-05	-29.956582
17/03/10	17/03/15	18/03/15	18/04/15	0.00021837	-15.757491
17/09/10	17/09/15	18/09/15	18/10/15	0.00011107	-20.975565
17/03/11	17/03/16	18/03/16	18/04/16	2.25E-05	-0.7098284
17/09/11	17/09/16	18/09/16	18/10/16	4.66E-05	-7.1147357
17/03/12	17/03/17	18/03/17	18/04/17	0.00024087	-59.991808
17/09/12	17/09/17	18/09/17	18/10/17	0.00026632	-17.788377
17/03/13	17/03/18	18/03/18	18/04/18	0.00016643	-43.694036
17/09/13	17/09/18	18/09/18	18/10/18	0.00074606	-163.71535
17/03/14	17/03/19	18/03/19	18/04/19	0.0005341	-51.572091
17/09/14	17/09/19	18/09/19	18/10/19	0.00184283	-65.733735
17/03/15	17/03/20	18/03/20	18/04/20	0.01169094	-380.28885
17/09/15	17/09/20	18/09/20	18/10/20	0.0008199	-30.974399
17/03/16	17/03/21	18/03/21	18/04/21	0.00030611	-11.210177
17/09/16	17/09/21	18/09/21	18/10/21	0.01513165	-253.4169
17/03/17	17/03/22	18/03/22	18/04/22	0.00258559	-42.538943
17/09/17	17/09/22	18/09/22	18/10/22	0.0003597	-6.6275502
17/03/18	17/03/23	18/03/23	18/04/23	3.33E-05	-2.7333078
17/09/18	17/09/23	18/09/23	18/10/23	0.00029349	-66.184388
17/03/19	17/03/24	18/03/24	18/04/24	0.00023908	-0.8469525
17/09/19	17/09/24	18/09/24	18/10/24	0.00035165	-18.112034

Table 4. 6: Statistics gathered by training an RNN on Raw Stock Market Data.

Table 4.6 presents the performance of the Recurrent Neural Network (RNN) model trained on unprocessed stock market data, employing a five-year sliding window for training and a one-month prediction timeframe. The findings indicate that the RNN often struggled to identify significant patterns within the raw data. All R^2 scores listed are negative, demonstrating that the model did not surpass a simple mean-based prediction approach. The MSE values exhibit considerable fluctuations, which further imply a lack of effective generalisation across different periods. This outcome underscores a fundamental limitation of RNNs. Although these models are intended to capture sequential

dependencies, their effectiveness can diminish sharply when faced with noise and high variability, both of which are inherent to unrefined financial data.

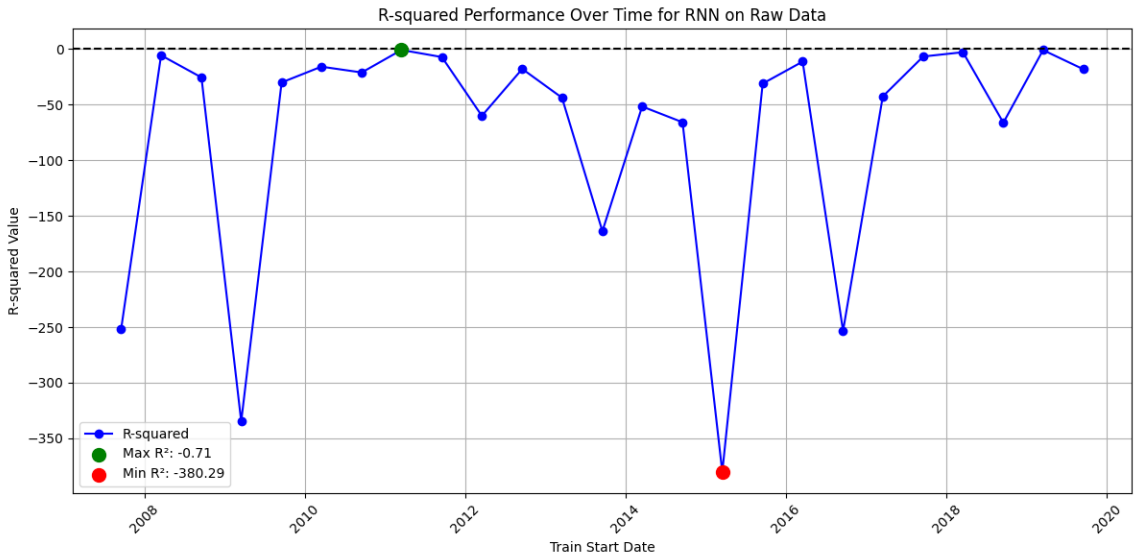

Figure 4. 7: Recording of the R^2 value for predictions made on the test data for each training data window using the RNN model on the raw data.

A potential reason is the data needs of RNNs. These models generally require a large volume of well-organised, high-quality sequential data to learn temporal relationships effectively. For financial time series, particularly with raw data, achieving such consistency is challenging. The short prediction windows and limited historical context hinder the amount of valuable information an RNN can utilise, often leading to difficulties in capturing significant temporal dependencies. Additionally, RNNs are very dependent on the quality of their training data; their performance can quickly decline when faced with non-stationary patterns, gaps, or sudden regime shifts - situations frequently encountered in actual stock market data.

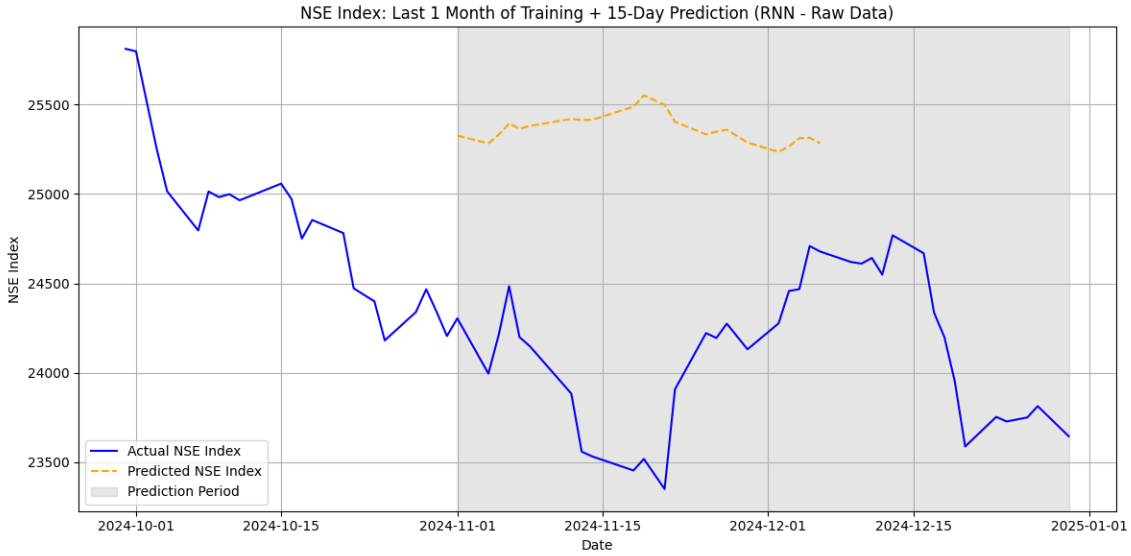

Figure 4. 8: Predictions made by the RNN Model trained on the raw stock market data. The prediction window is different from that used for ANN because RNN and its variations need a LOOPBACK window. Here, the LOOPBACK window is set to 15 days.

Interestingly, the ANN model, despite lacking a recurrent structure, was more effective at identifying specific patterns in the raw dataset. As noted in section 4.3, the predictions made by the ANN generally mirrored the overall trend of the index, although they consistently showed a slight discrepancy. This relative effectiveness arises from the ANN's capability to approximate complex nonlinear functions over shorter, fixed-length input sequences. While it does not account for temporal dependencies, it effectively captures prevailing recent trends within the input window. In contrast, the RNN attempts to leverage temporal continuity but faces challenges when dealing with noisy data and insufficient long-range signals. This comparison underscores that, in certain scenarios, simpler architectures like ANNs can remain competitive, particularly when training data is scarce and recent information holds the most predictive value.

Below is the examination of the RNN model's training cycles. Figure 4.9 illustrates the RNN model's training and validation performance across various sliding windows of stock market data. The model was trained on each window for a set number of epochs with early stopping, capturing the R^2 scores for both training and validation at each epoch. These scores were subsequently plotted to monitor the model's learning behaviour and generalisation ability over time.

Each subplot in the chart represents a distinct training window, with its start date clearly indicated in the subplot title. The x-axis of each plot corresponds to the training epoch, while the y-axis shows the R^2 score. Within each plot, two different lines are depicted: one for the R^2 score on the training data and the other for the validation data. The training curve illustrates how effectively the model fits the data it has encountered, while the validation curve offers insights into the model's ability to generalise to unseen data.

The model utilised a sliding window technique, training it on a five-year data block while validating with the following one-month period. This method involves advancing the window by six months for each iteration, creating several overlapping training and validation windows. Consistency in evaluation was ensured by applying the same model architecture and training configuration across all windows.

The complete chart was created by plotting the R^2 curves from each window in a grid format. This setup provides a comparative perspective on training dynamics throughout various market periods. Uniform formatting, axis scales, and line styles in the subplots facilitate easy visual inspection of the differences in training and validation behaviour during the interpretation phase.

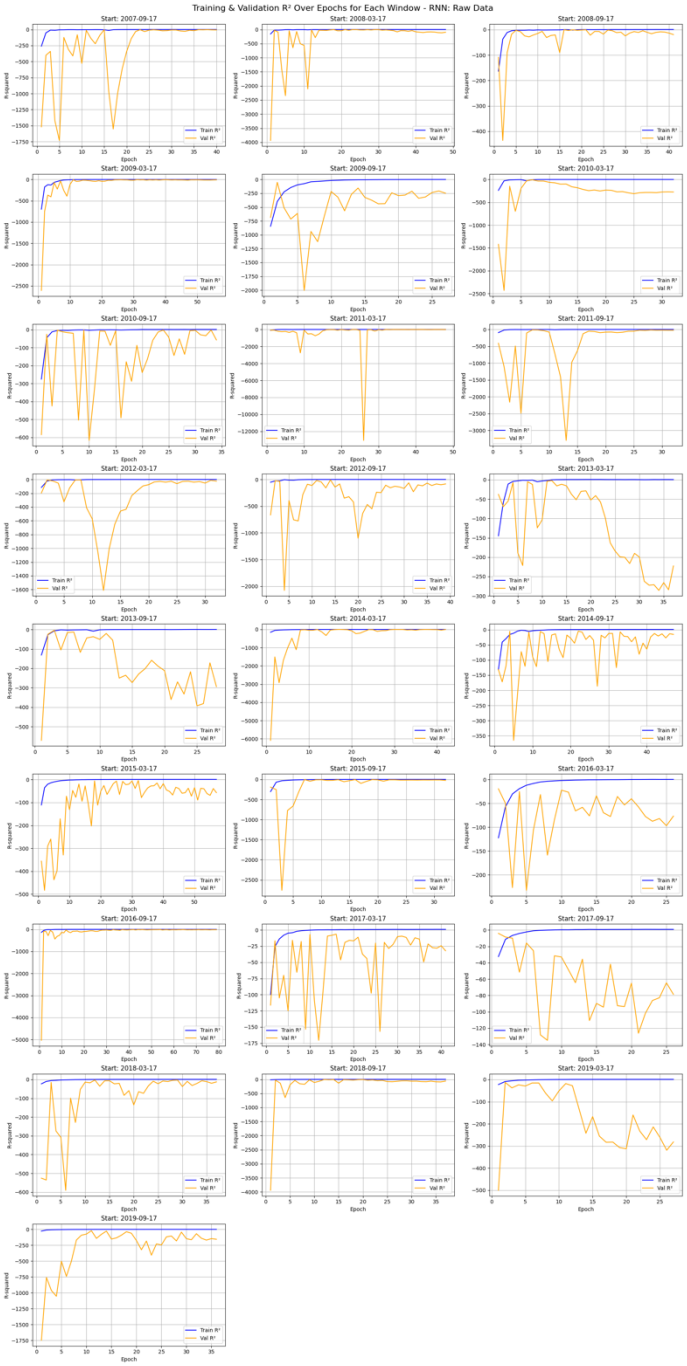

Figure 4. 9: Training and Validation R^2 observed while training the RNN model on raw stock market data.

A notable trend in most subplots of Figure 4.13 is the sharp and steady rise in training R^2 during the first few epochs, followed by a plateau phase. This suggests that the

RNN quickly adapts to the training data, often achieving near-perfect R^2 scores. However, this strong performance on the training dataset does not carry over to the validation dataset, where R^2 scores frequently fluctuate and can be significantly negative.

The validation R^2 displays notable volatility in many windows throughout the training process. Rather than stabilising or improving in line with the training R^2, the validation curve shows erratic fluctuations, sometimes diverging even more with each epoch. This behaviour indicates that the model is likely overfitting the training data and struggling to generalise to unseen examples. Occasionally, the validation R^2 experiences significant drops after just a few epochs, remaining consistently unstable and emphasising the lack of generalisation. The pronounced spikes and dips in the validation curves across almost all training windows highlight the RNN's sensitivity to noise and its difficulty in identifying meaningful patterns within the raw data.

The behaviour aligns with expectations for a model applied to highly volatile and unprocessed time series data. The raw financial data likely presents a combination of long-term trends, short-term fluctuations, and non-stationary elements that the RNN struggles to distinguish effectively. While RNNs are engineered to grasp temporal dependencies, they typically need more stable and structured input sequences to achieve reliable performance. The variability in the validation curves across different windows indicates that the RNN could not identify consistent temporal relationships within the raw data, resulting in subpar and inconsistent validation outcomes.

Moreover, the absence of a correlation between training and validation performance across epochs reveals a significant challenge in employing RNNs for predicting raw

financial data. While the model fits the training data exceedingly well, it struggles to generalise to new data, indicating limited predictive capability. This divergence emphasises the necessity of not depending solely on training performance as a measure of model efficacy, particularly in noisy, real-world datasets. Consequently, this suggests that enhanced preprocessing techniques may be required to attain significant results in these situations.

After assessing the RNN model's performance on unprocessed stock market data, the analysis is broadened by using the same architecture on stationary data. This change, accomplished through differencing techniques, seeks to stabilise the time series' statistical properties and possibly enhance the model’s capacity to identify significant temporal dependencies. Whether preprocessing the data to eliminate non-stationarity results in more reliable and precise predictions from the RNN model is examined by retaining the same sliding window approach and evaluation metrics.

The results obtained after training the RNN model on stationary data are stated in Table 4.7.

Train Start	Train End	Test Start	Test End	MSE	R^2
18/09/07	18/09/12	19/09/12	19/10/12	0.00026953	**0.05871117**
18/03/08	18/03/13	19/03/13	19/04/13	0.00148652	-0.8001121
18/09/08	18/09/13	19/09/13	19/10/13	0.00172001	-0.4601275
18/03/09	18/03/14	19/03/14	19/04/14	0.0006982	**0.04708316**
18/09/09	18/09/14	19/09/14	19/10/14	0.00016046	
18/03/10	18/03/15	19/03/15	19/04/15	0.00233017	-0.6382926
18/09/10	18/09/15	19/09/15	19/10/15	0.00053466	-10.669802
18/03/11	18/03/16	19/03/16	19/04/16	0.0027525	-7.2549318
18/09/11	18/09/16	19/09/16	19/10/16	0.00217826	-0.0982489
18/03/12	18/03/17	19/03/17	19/04/17	0.00058879	-1.1736295
18/09/12	18/09/17	19/09/17	19/10/17	0.0009991	-0.22662
18/03/13	18/03/18	19/03/18	19/04/18	0.00017783	-0.6589049
18/09/13	18/09/18	19/09/18	19/10/18	0.00303267	-0.3782921
18/03/14	18/03/19	19/03/19	19/04/19	0.00047975	-0.1404551
18/09/14	18/09/19	19/09/19	19/10/19	0.00230404	-6.7682722
18/03/15	18/03/20	19/03/20	19/04/20	0.00578191	-0.3048226
18/09/15	18/09/20	19/09/20	19/10/20	0.0060269	-0.0384421
18/03/16	18/03/21	19/03/21	19/04/21	0.00641959	-0.0349274
18/09/16	18/09/21	19/09/21	19/10/21	0.00213598	-0.2841391
18/03/17	18/03/22	19/03/22	19/04/22	0.0092966	-4.6284142
18/09/17	18/09/22	19/09/22	19/10/22	0.00355745	-0.5518346
18/03/18	18/03/23	19/03/23	19/04/23	0.0012568	-1.382633
18/09/18	18/09/23	19/09/23	19/10/23	0.00186613	-0.6895244
18/03/19	18/03/24	19/03/24	19/04/24	0.00944257	-1.0360282
18/09/19	18/09/24	19/09/24	19/10/24	0.00410963	-0.2599484

Table 4. 7: Statistics gathered by training an RNN on Stationary Stock Market Data.

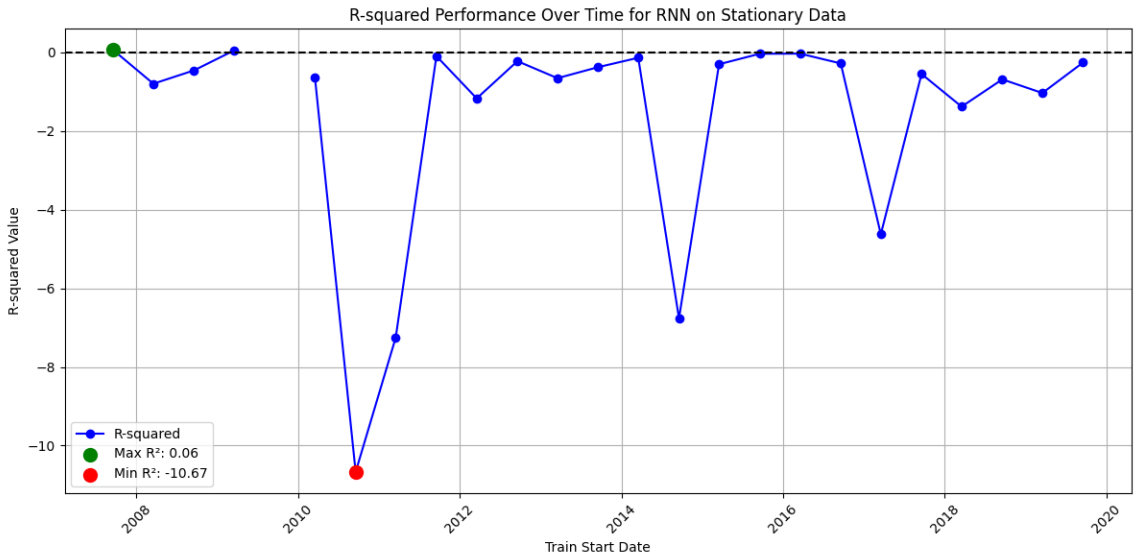

Figure 4. 10: Recording of the R^2 value for predictions made on the test data for each training data window using the RNN model on the stationary data.

This chart illustrates the R^2 values derived from the RNN model's predictions over time, specifically trained on stationary stock market data. Each point on the graph represents a training window, with the horizontal axis indicating the start date of the training period and the vertical axis displaying the resulting R^2 score from the corresponding test window. The blue line depicts the fluctuations of R^2 values across periods, while the green and red markers indicate the best and worst performances, respectively.

Analysing the graph structurally reveals the model's predictive consistency – or lack of it – across various time frames. The majority of R^2 values sit below zero, suggesting that the model often performed worse than a simple mean prediction. However, a few time windows exhibit slight enhancements, with one achieving a positive R^2 of 0.06, marking the most notable case. In contrast, the most significant underperformance is evidenced by a steep drop in R^2 to -10.67, illustrating a scenario where the model's predictions were far from the actual results.

These variations indicate that, despite utilising differencing methods to make the data stationary, the RNN's generalisation capability remained erratic. One might anticipate that eliminating trends and seasonality would enable the model to concentrate better on the core signals. Yet, the outcomes reflect only slight and inconsistent enhancements in prediction accuracy. For example, several intervals display moderate R^2 values ranging from -0.1 to -0.6, which, albeit still negative, represent a notable improvement compared to extreme outliers. These more consistent areas imply that the model identified some fleeting patterns, but such advantages were not consistent throughout the entire period.

Let's examine Figure 4.10 (RNN trained on stationary data) alongside Figure 4.7 (RNN trained on raw data). Both figures exhibit generally poor predictive accuracy; however, Figure 4.10, which incorporates stationarity, shows a slight decrease in volatility in the R^2 scores. The model represented in Figure 4.7 trained on raw data experiences severe, erratic fluctuations in R^2 values, with several intervals reflecting significantly negative scores that suggest a failure to generalise. In comparison, the R^2 values in Figure 4.10, despite remaining predominantly negative, cluster more closely to zero and display somewhat milder fluctuations. This implies that converting the data into a stationary format slightly stabilises the model's output, although it does not yield consistent or reliable predictions. The highest R^2 noted in the stationary framework is a marginal positive value of 0.06, while the lowest plummets to -10.67, remaining a considerable outlier. In summary, while the stationarisation of input data seems to assist the RNN model in mitigating some extreme failures observed with raw data, it does not significantly enhance the model's overall predictive capability, underscoring the necessity for more sophisticated architectures or comprehensive feature engineering.

Figure 4.11 shows the progression of the training process when the RNN model was built on the stationarised data.

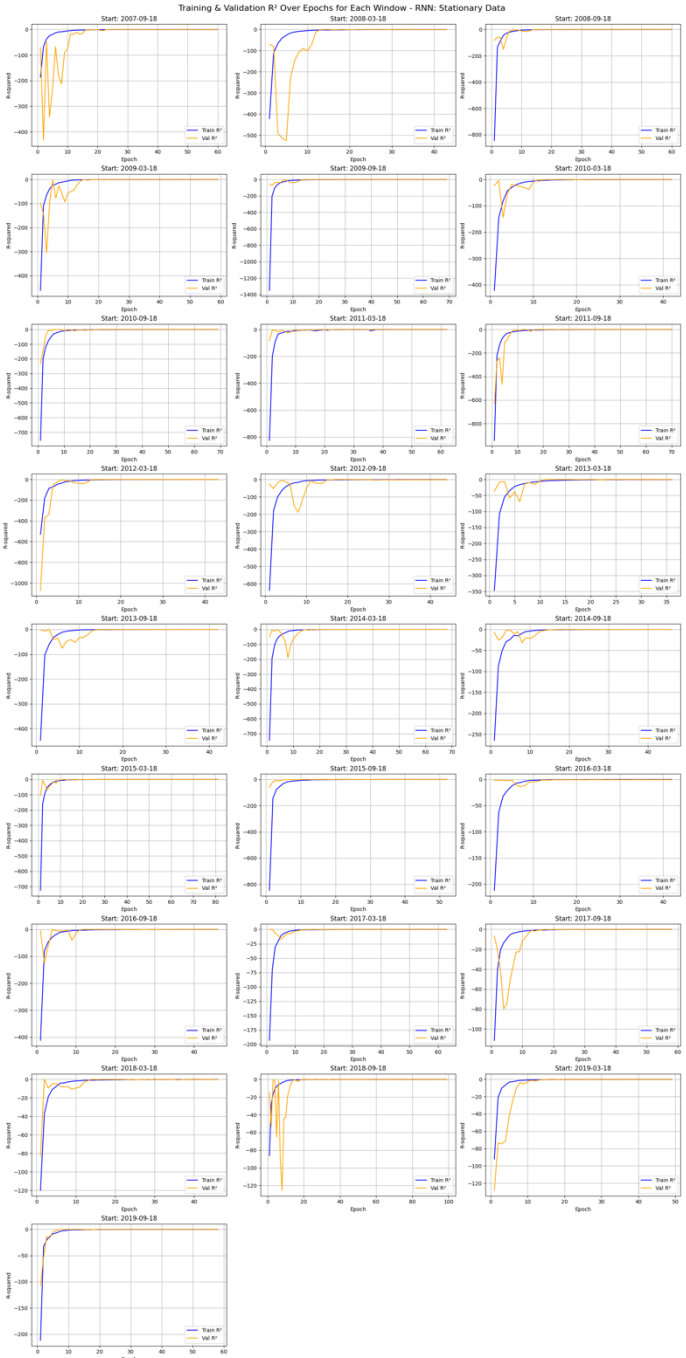

Figure 4. 11: Training and Validation R^2 observed while training the RNN model on stationary stock market data.

Figure 4.11 illustrates the R^2 scores for training and validation across epochs in the RNN model, which was trained on stationary stock market data with various sliding windows. Each subplot reflects a distinct training window. For each case, the RNN model underwent training for a predetermined number of epochs (utilising early stopping) with a five-year training set, and it was subsequently evaluated over a following one-month testing period. The blue line indicates the R^2 score for the training set, whereas the orange line displays the R^2 score for the validation data at each epoch.

In contrast to the chart for the RNN model using raw data, this version shows considerable enhancement in both stability and convergence of the validation R^2 curves. Across nearly all windows, the validation curves establish a more consistent pattern, exhibiting substantially less volatility over epochs. Although some fluctuations persist – particularly in the initial epochs – the extreme spikes and erratic behaviour noted in the raw data version are largely missing here. This consistency suggests the model is identifying more stable relationships when trained on differenced data, which eliminates non-stationary factors such as trends and seasonal influences.

A key observation is that the gap between training and validation R^2 scores tends to narrow across most windows. This indicates a reduction in overfitting and enhanced generalisation – two crucial factors when evaluating the robustness of time series models. Although the validation curves are not always entirely positive, they exhibit smooth trajectories that closely mirror the training performance. The model's learning process seems more consistent across windows, suggesting a better ability to capture signals from stationary data compared to the raw version.

Overall, the chart validates the theory that converting financial time series data into a stationary form improves the model's capacity to generalise and identify stable patterns. While R^2 values are generally low and even turn negative, the decrease in variance and improved consistency across different windows signify a significant advancement. Training the RNN on stationary data leads to more dependable convergence and steadier validation performance, indicating that making the input stationary positively impacts the predictive stability of sequential models such as RNNs in stock market forecasting.

The predictions made by the RNN model training on stationary stock market data are shown in Figure 4.12.

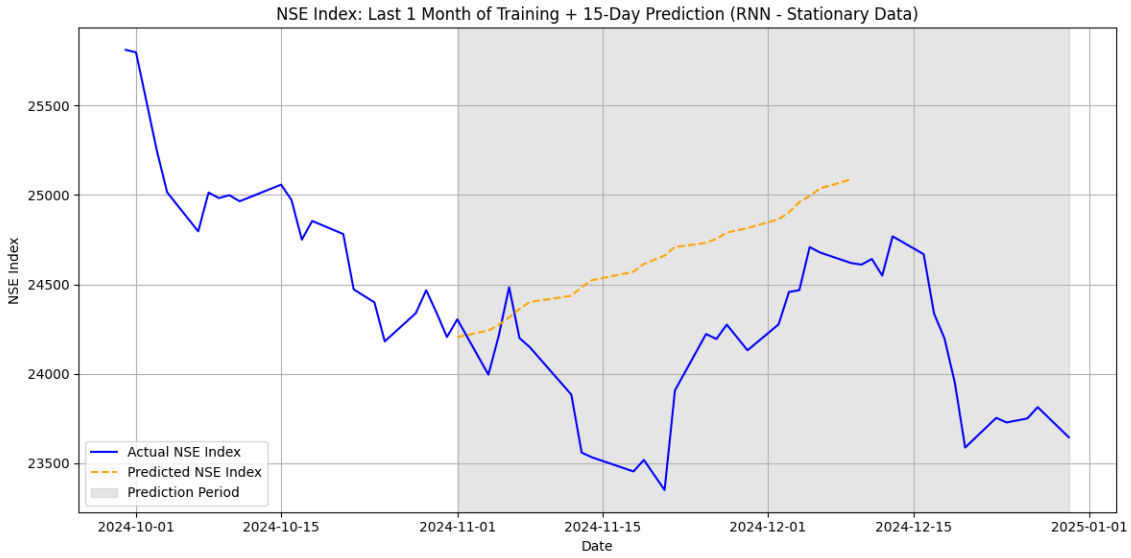

Figure 4. 12: Predictions made by the RNN Model trained on the stationary stock market data. The prediction window is different from that used for ANN because RNN and its variations need a LOOPBACK window. Here, the LOOPBACK window is set to 15 days.

Figure 4.12 illustrates the forecasts produced by the RNN model trained on stationary stock market data. In this experiment, the RNN used a loopback window of 15

days to capture sequence dependencies, with the prediction window positioned right after the training phase. This figure displays the predicted index values during the chosen test period, enabling a visual comparison with the actual stock market data. The setup for model training and prediction aligns with previous experiments, with the exception of the transformation applied to the input data to achieve stationarity through differencing.

In contrast to Figure 4.8, which shows predictions from the RNN trained on raw data, Figure 4.12 illustrates a noticeable improvement in how closely the predicted curve follows actual market behaviour. While both figures indicate that the model does not completely replicate the magnitude or direction of every price movement, the model using stationary data yields predictions that are more stable and less erratic. Conversely, the predictions in Figure 4.8 show sharp fluctuations and a noisier path, highlighting the difficulties the model encountered when dealing with non-stationary inputs. This direct comparison provides empirical evidence for the notion that preprocessing stock market data to achieve stationarity can enhance model predictions by minimising volatility and facilitating improved pattern recognition in short-term scenarios.

Now, let’s compare the predictions made by the RNN model with stationary data, illustrated in Figure 4.12, to those made by the ANN on the same type of data, shown in Figure 4.5. The predictions in Figure 4.5, while appearing smoother and often lagging or offset, generally align with the overall trend of the stock index, indicating a broad capacity to capture directional movements based on recent data. In contrast, the predictions in Figure 4.12 display more nuanced temporal responsiveness, sometimes closely matching short-term fluctuations. This enhancement in temporal tracking highlights the RNN’s capability to model sequential dependencies, which is a limitation of the ANN. However, this

advantage comes with greater variability and occasional divergence from actual movements. Despite both models being trained on stationary data, the RNN demonstrates a more dynamic, albeit occasionally unstable, prediction pattern, which suggests it can better leverage temporal structures while also being sensitive to noise and limited contextual signals. This comparison underscores that, among simpler models, the RNN outperforms the ANN when stationarity preprocessing is applied, especially in modelling short-term movements in financial time series.

4.5 Creating an LSTM Model on Raw and Stationary Data

Continuing from the experimentation on Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks are introduced next into our modelling framework. LSTM networks are a specialised type of recurrent architecture that addresses the shortcomings of traditional RNNs, especially in learning long-term dependencies in sequential data. With the inclusion of memory cells and gating mechanisms, LSTMs can effectively retain and selectively update information across prolonged time steps, making them ideal for the dynamic and often noisy patterns commonly seen in financial time series. This section assesses the LSTM model's performance when trained on both raw and stationary stock market data, employing the same sliding window methodology used in earlier experiments. The model undergoes training over a five-year span and is evaluated in the following month, ensuring consistency in our comparative analysis. This analysis aims to determine if the enhanced memory functions of LSTMs lead to improved predictive accuracy, particularly regarding various forms of data representation.

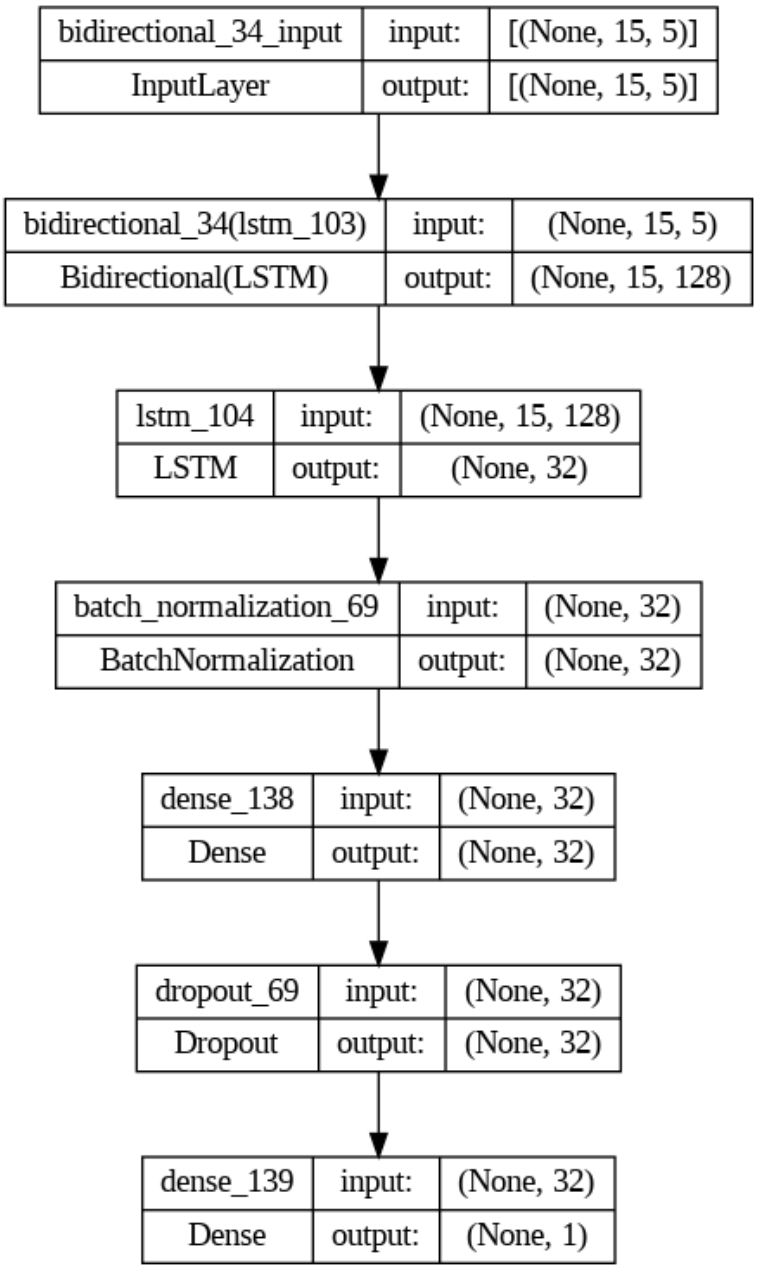

Figure 4. 13: Architecture of the LSTM Model. This model was trained on both raw and stationarised data.

The results obtained after training the LSTM model on raw data are stated in Table 4.8.

Train Start	Train End	Test Start	Test End	MSE	R^2
17/09/07	17/09/12	18/09/12	18/10/12	0.00056315	-532.7794
17/03/08	17/03/13	18/03/13	18/04/13	0.00040255	-30.118859
17/09/08	17/09/13	18/09/13	18/10/13	0.00133225	-224.07566
17/03/09	17/03/14	18/03/14	18/04/14	0.00054048	-161.90977
17/09/09	17/09/14	18/09/14	18/10/14	0.00376366	-8515.2273
17/03/10	17/03/15	18/03/15	18/04/15	9.67E-05	-6.4222325
17/09/10	17/09/15	18/09/15	18/10/15	7.46E-05	-13.759347
17/03/11	17/03/16	18/03/16	18/04/16	0.00053407	-39.581385
17/09/11	17/09/16	18/09/16	18/10/16	1.12E-05	-0.9455875
17/03/12	17/03/17	18/03/17	18/04/17	0.00056788	-142.79464
17/09/12	17/09/17	18/09/17	18/10/17	0.00115335	-80.368174
17/03/13	17/03/18	18/03/18	18/04/18	0.0001164	-30.258602
17/09/13	17/09/18	18/09/18	18/10/18	0.00170234	-374.84468
17/03/14	17/03/19	18/03/19	18/04/19	6.60E-05	-5.4961314
17/09/14	17/09/19	18/09/19	18/10/19	0.00064306	-22.287073
17/03/15	17/03/20	18/03/20	18/04/20	0.01599429	-520.63843
17/09/15	17/09/20	18/09/20	18/10/20	0.00132376	-50.623926
17/03/16	17/03/21	18/03/21	18/04/21	3.22E-05	-0.283918
17/09/16	17/09/21	18/09/21	18/10/21	0.01013742	-169.44613
17/03/17	17/03/22	18/03/22	18/04/22	0.00201614	-32.949973
17/09/17	17/09/22	18/09/22	18/10/22	0.00111411	-22.624843
17/03/18	17/03/23	18/03/23	18/04/23	0.00054897	-60.584383
17/09/18	17/09/23	18/09/23	18/10/23	2.96E-05	-5.7661787
17/03/19	17/03/24	18/03/24	18/04/24	0.0022268	-16.202608
17/09/19	17/09/24	18/09/24	18/10/24	0.00346488	-187.31558

Table 4. 8: Statistics gathered by training an LSTM on Raw Stock Market Data.

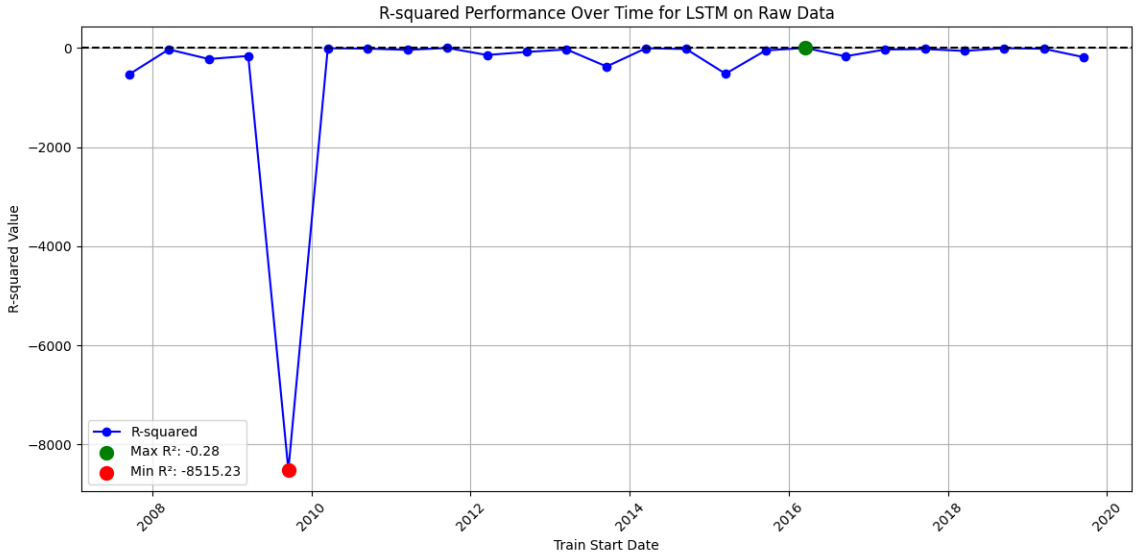

Figure 4. 14: Recording of the R^2 value for predictions made on the test data for each training data window using the LSTM model on the raw data.

Figure 4.14 illustrates the predictive performance of the LSTM model trained on raw stock market data, quantified through R^2 values computed over several sliding training windows. Each data point represents the model's performance for a particular test window immediately following a five-year training period, offering a chronological depiction of predictive capability. The R^2 values depicted in Figure 4.14 exhibit substantial variability, frequently dipping into deeply negative territory, signifying that the LSTM often performed worse than simple baseline predictions, such as the historical mean. The inconsistent and predominantly poor performance indicates significant challenges faced by the LSTM in modelling the underlying temporal dependencies present within the noisy and non-stationary stock market dataset.

In comparison to Figure 4.7, which illustrates the R^2 performance of the RNN model using the same raw data, a significant difference is evident. The RNN model exhibits considerable variability in performance across various windows, but its R^2 values tend to

be closer to zero. Additionally, there are fewer instances of extremely negative results seen in the LSTM model. This indicates that, unexpectedly, the simpler RNN architecture is more consistent and, at times, performs better than the theoretically superior LSTM architecture when modelling raw, unprocessed stock market data.

Several reasons may explain why the LSTM model underperformed compared to the simpler RNN model in this situation. First, LSTMs, due to their intricate gating mechanisms and memory cells, usually need large, high-quality, and well-organized datasets to learn significant long-term patterns effectively. However, raw financial time series data often fall short of these criteria, exhibiting high volatility, sudden regime shifts, and extensive noise that can disrupt complex models. In such noisy conditions, the inherent complexity of LSTMs is a drawback, leading to overfitting during training and, as a result, poor generalisation of unseen data. Moreover, the relatively brief prediction horizons used here—just one month—may not fully exploit the LSTM's capabilities in capturing long-term dependencies, thus further restricting its effectiveness compared to simpler recurrent models like RNNs.

Thus, the comparative analysis of Figures 4.14 and 4.7 reveals an important point: simply increasing model complexity does not assure better predictive performance, particularly when faced with raw, noisy financial data. This emphasises the importance of thorough data preparation, effective feature engineering, and the need to weigh model complexity against data characteristics as crucial elements for enhancing predictive accuracy in stock market forecasting.

The data captured while training the LSTM model corroborates this understanding.

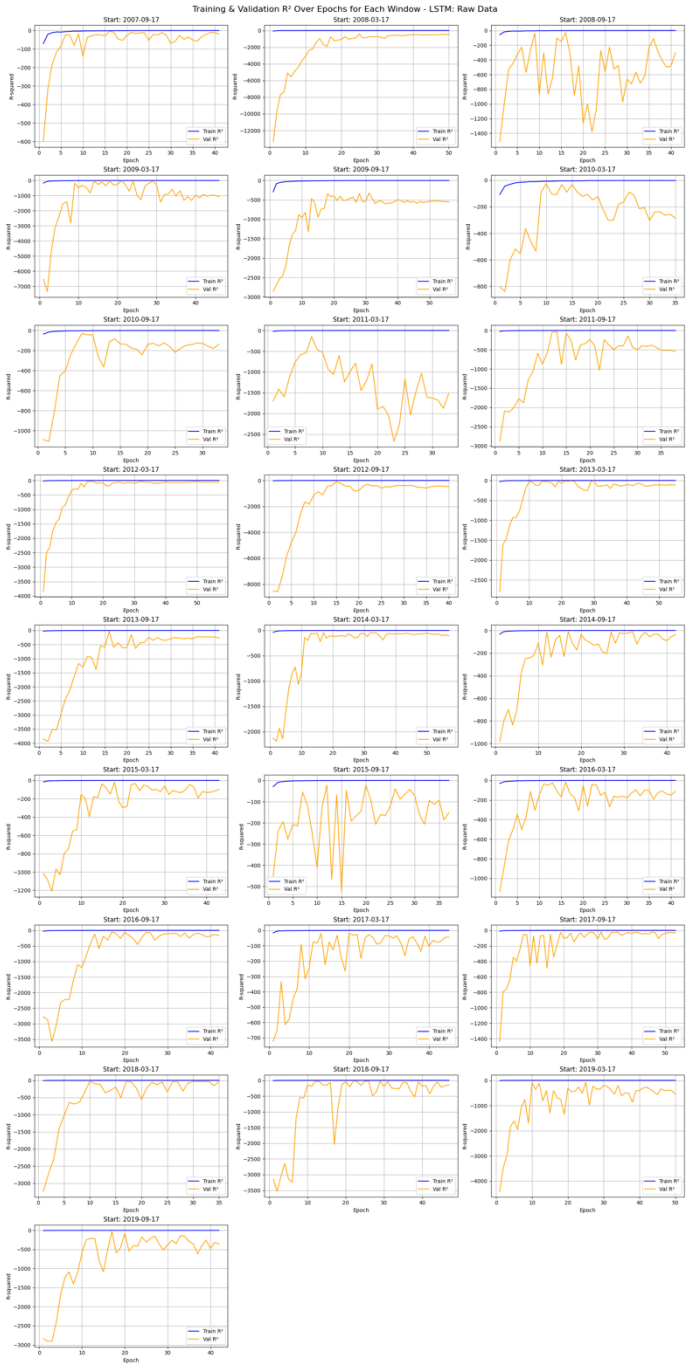

Figure 4. 15: Training and Validation R^2 observed while training the LSTM model on raw stock market data.

Figure 4.15 shows the R^2 values for training and validation across epochs for the LSTM model, which is trained on raw stock market data using multiple sliding windows. Each subplot represents a unique five-year training period followed by a one-month validation. This layout allows for evaluation of the LSTM's ability to learn and generalise across different temporal contexts. A close examination of the chart reveals that the training R^2 curves exhibit swift initial gains, quickly stabilising at higher levels. This rapid stabilisation suggests that the LSTM model effectively adapts to the training data, successfully capturing the historical patterns within the training windows.

Despite strong training performance, the validation R^2 values present a contrasting scenario. They display erratic behaviour marked by significant fluctuations, instability, and even negative values, indicating a clear divergence from corresponding curves. This sharp contrast between training and validation performances highlights the model's difficulties with generalisation, as it fails to identify stable and predictive relationships from raw stock market data. The erratic nature of the validation performance points to serious overfitting issues, suggesting that the LSTM's intricate gating mechanism, intended to recognise long-term dependencies, might be inadvertently capturing the noise and random fluctuations commonly seen in financial markets.

When these findings are compared to previous results, particularly Figure 4.9, which illustrated RNN performance using raw data, it becomes clear that the LSTM behaviour is significantly more unpredictable. Both models struggled with data noise and instability, but the LSTM's complex architecture makes it more prone to overfitting when confronted with raw data that lacks distinct long-term patterns. On the other hand, the simpler RNN, while not necessarily better in absolute predictive performance, tended to

demonstrate fewer extreme fluctuations in validation and maintained more consistent (albeit modest) results. This observation paradoxically emphasises that greater architectural complexity does not inherently lead to better forecasting capabilities on noisy raw datasets, especially when the data does not reveal stable long-term dependencies.

Figure 4.15 emphasises an important point in the larger conversation. Although LSTM networks are theoretically suited for capturing long-term temporal relationships, their actual effectiveness relies heavily on the quality and characteristics of the input data. Raw financial time series, characterised by inherent volatility, short-term irregularities, and frequent regime shifts, present substantial challenges that restrict the advantages usually provided by advanced sequential models like LSTMs.

Though deficiencies were found in the LSTM model trained on raw stock market data, let's examine the predictions obtained from this model.

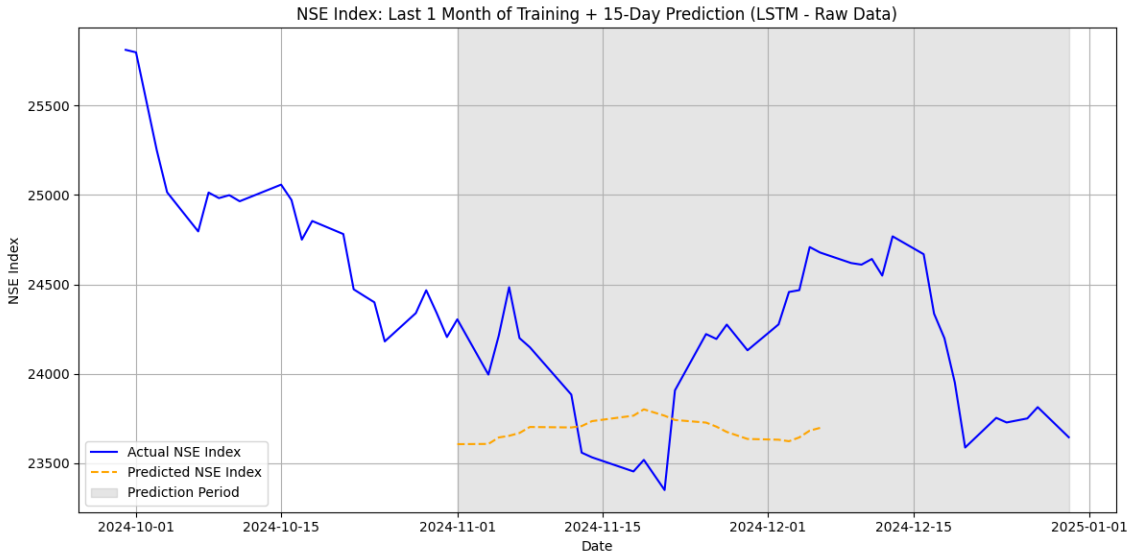

Figure 4. 16: Predictions made by the LSTM Model trained on the raw stock market data. The prediction window is different from that used for ANN because LSTMs need a LOOPBACK window. Here, the LOOPBACK window is set to 15 days.

Figure 4.16 depicts the LSTM model's predictions after being trained on raw stock market data, specifically showcasing its performance over a one-month forecast period that follows a five-year training span. This figure allows for a direct comparison between the actual stock market index values and the predictions generated by the LSTM. The graphical representation clearly reveals that the LSTM predictions show notable deviations from actual values, often lagging behind real market trends with significant offset and amplitude errors. Although the model occasionally captures the overall directional trends, it struggles considerably to mirror the precise short-term fluctuations characteristic of stock market data.

A detailed analysis shows that the predictions made by the LSTM often trail behind actual market movements, indicating a sluggish response to trend changes. This lagging response likely stems from the intricate and deep internal gating mechanisms of the LSTM, which are meant to handle long-term dependencies in sequential data. Ideally, these mechanisms enable the model to retain important historical information over long durations. However, in the realm of raw financial data, characterised by frequent short-term volatility and quick shifts, this same intricacy can hinder the model's ability to differentiate between significant patterns and random noise. As a result, the model tends to accommodate past historical trends instead of effectively adjusting to new, developing patterns.

When comparing the LSTM predictions with those from the RNN model, it is a surprise to discover that the simpler RNN performs relatively better despite its less complex design. Both models show prediction errors and deviations, but the RNN’s forecasts seem better aligned with short-term market movements and show fewer cases of significant lag

or drastic divergence. This unexpected finding can be explained by the fact that RNNs, although simpler, possess fewer internal parameters and gates, allowing them to adapt more swiftly to recent trends in raw, noisy data.

The main reason the LSTM fell short compared to the simpler RNN when trained on raw data is mainly related to its complexity and sensitivity to data quality. LSTMs are designed to work best with substantial, structured datasets that exhibit clear long-term relationships, allowing them to utilise their memory functions effectively. However, raw stock market data usually does not possess these characteristics due to its inherent volatility, unpredictable trends, and frequent regime changes, which hinder the LSTM's ability to identify stable and predictive temporal patterns. Consequently, instead of boosting accuracy, the increased complexity of LSTMs can make the model more susceptible to overfitting, amplifying noise, and slowing responsiveness. This observation highlights the essential role of data preprocessing and transformation techniques, such as stationarisation, especially when using advanced sequential models like LSTMs for forecasting in financial markets.

The next step is to experiment with creating an LSTM model on stationarised stock market data. The results obtained after training this LSTM model on stationarised data are stated in Table 4.9.

Train Start	Train End	Test Start	Test End	MSE	R^2
18/09/07	18/09/12	19/09/12	19/10/12	0.00026528	**0.07355981**
18/03/08	18/03/13	19/03/13	19/04/13	0.00121089	-0.4663302
18/09/08	18/09/13	19/09/13	19/10/13	0.00192831	-0.6369627
18/03/09	18/03/14	19/03/14	19/04/14	0.00072403	**0.01182684**
18/09/09	18/09/14	19/09/14	19/10/14	0.00010692	
18/03/10	18/03/15	19/03/15	19/04/15	0.0023534	-0.6546241
18/09/10	18/09/15	19/09/15	19/10/15	0.00044432	-8.6979131
18/03/11	18/03/16	19/03/16	19/04/16	0.00277591	-7.3251573
18/09/11	18/09/16	19/09/16	19/10/16	0.00208181	-0.0496216
18/03/12	18/03/17	19/03/17	19/04/17	0.00042547	-0.5707263
18/09/12	18/09/17	19/09/17	19/10/17	0.000972	-0.1933516
18/03/13	18/03/18	19/03/18	19/04/18	0.0001888	-0.7612753
18/09/13	18/09/18	19/09/18	19/10/18	0.00284163	-0.2914673
18/03/14	18/03/19	19/03/19	19/04/19	0.00043503	-0.0341583
18/09/14	18/09/19	19/09/19	19/10/19	0.00238342	-7.035911
18/03/15	18/03/20	19/03/20	19/04/20	0.00590799	-0.3332736
18/09/15	18/09/20	19/09/20	19/10/20	0.00611503	-0.0536271
18/03/16	18/03/21	19/03/21	19/04/21	0.00639465	-0.0309069
18/09/16	18/09/21	19/09/21	19/10/21	0.00239466	-0.4396614
18/03/17	18/03/22	19/03/22	19/04/22	0.00831514	-4.0342108
18/09/17	18/09/22	19/09/22	19/10/22	0.00330881	-0.4433743
18/03/18	18/03/23	19/03/23	19/04/23	0.00128332	-1.4329078
18/09/18	18/09/23	19/09/23	19/10/23	0.00213711	-0.9348549
18/03/19	18/03/24	19/03/24	19/04/24	0.0090369	-0.9485566
18/09/19	18/09/24	19/09/24	19/10/24	0.00433609	-0.3293778

Table 4. 9: Statistics gathered by training an LSTM on Stationary Stock Market Data.

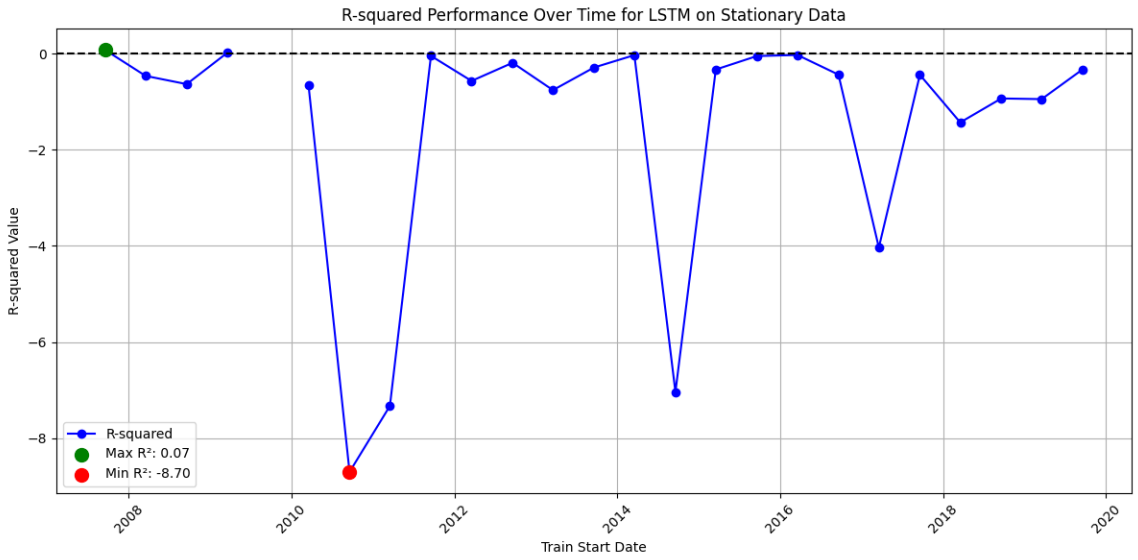

Figure 4. 17: Recording of the R^2 value for predictions made on the test data for each training data window using the LSTM model on the stationary data.

Figure 4.17 demonstrates the predictive accuracy of the LSTM model trained on stationary stock market data by displaying R^2 scores across several sliding training windows. The R^2 values throughout these windows reveal that the performance of the LSTM model is considerably variable yet typically closer to zero than its performance on raw data, indicating a more consistent predictive capability. Despite several negative R^2 values, which highlight instances where the model performed worse than a basic mean-based forecast, these downward trends are much less severe than those seen with raw data training. This enhanced stability shows that converting the data to a stationary form notably diminishes volatility and noise, allowing the LSTM model to identify meaningful short-term temporal patterns within the data more effectively.

Comparing these findings with Figure 4.14, which depicts the LSTM model trained on raw data, shows a clear improvement in stability and a decrease in the severity of negative performance. Figure 4.14 illustrates how often the LSTM model encountered

deeply negative R^2 scores, emphasising the difficulties it faced when working with raw, noisy, and non-stationary stock market data. The more consistent performance of the stationary data in Figure 4.17 indicates that preprocessing the data to achieve stationarity enhances modelling accuracy by reducing erratic fluctuations and allowing the model to identify relevant patterns more effectively.

Nevertheless, even with the improvements, the LSTM applied to stationary data continued to have difficulty consistently attaining strongly positive R^2 values. This suggests that although stationarity contributes to stability and diminishes predictive fluctuations, further strategies or enhanced modelling techniques may still be necessary to fully leverage the capabilities of LSTM networks for stock market forecasting.

The data collected during the training cycle reinforces this conclusion.

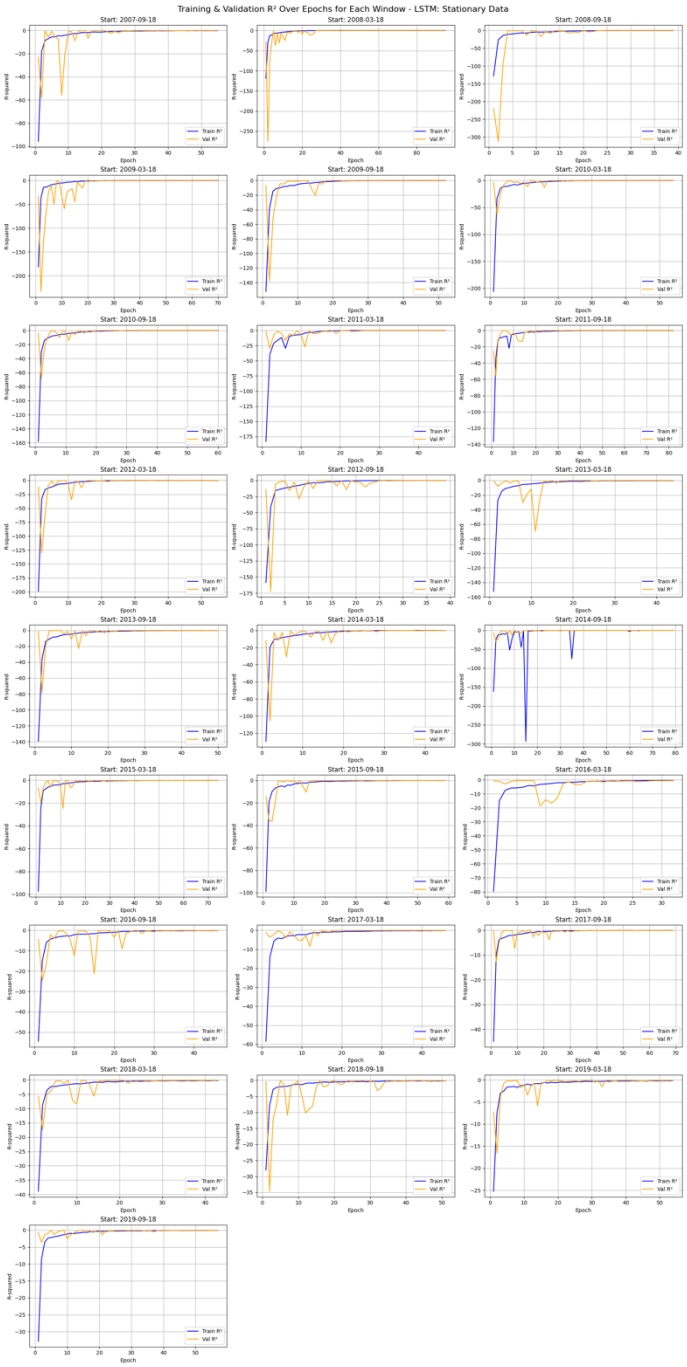

Figure 4. 18: Training and Validation R2 observed while training the LSTM model on stationary stock market data.

Figure 4.18 illustrates the R^2 values for training and validation during the LSTM model's training phase with stationary data. Over various training windows, the LSTM exhibits consistent training behaviours, as shown by the swiftly converging training R^2 values. Initially, these values are notably negative, reflecting the model's poor initial fit. However, within just a few epochs, they quickly rise and stabilise around zero, indicating that the model efficiently learns from stationary time series data. The validation R^2 values closely align with the training values, showcasing similar improvements and stabilisation, which suggests effective generalisation without significant overfitting.

The stationarity of input data improves the LSTM model's ability to capture patterns effectively. By eliminating trends and irregular fluctuations, stationarity simplifies the LSTM's learning tasks, enabling it to better understand and model consistent behaviours in stock market data. The convergence patterns depicted in Figure 4.18 reveal fewer fluctuations compared to similar charts created from raw data, indicating that stationary data aids the model in achieving more rapid and reliable training.

Moreover, within each window, the training and validation scores converge smoothly and consistently over epochs, showcasing the LSTM's strength in capturing key time dependencies while minimising the impact of noise. Importantly, the sustained validation scores at elevated performance levels reinforce the practical utility of utilising stationary data to enhance stock market predictions.

In summary, the examination of Figure 4.18 highlights the advantages of converting stock market data into a stationary format prior to utilising advanced, recurrent models such as LSTM. When comparing the training of the same model on unprocessed data versus

stationary data, the latter produces a more defined learning trajectory and better validation results. This emphasises the importance of preprocessing stock market data to attain stationarity, a necessary advancement for improving the predictive power of machine learning models, especially LSTMs.

It is now established that making the stock market data stationary improves the predictive power of the models. The predictions made by the LSTM model trained on the stationary stock market data are shown in Figure 4.19.

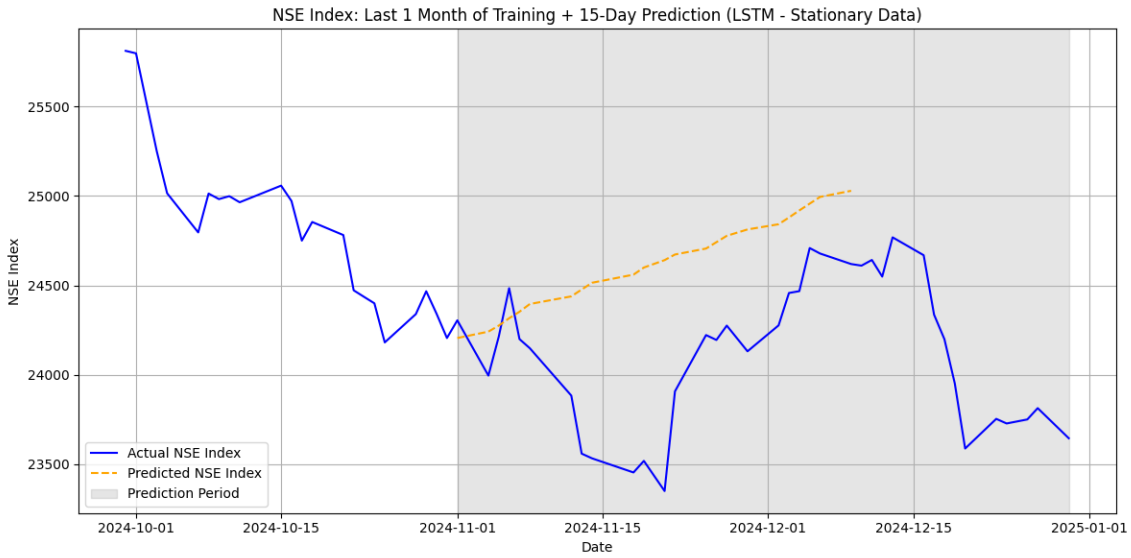

Figure 4. 19: Predictions made by the LSTM Model trained on the stationary stock market data. The prediction window is different from that used for ANN because LSTMs need a LOOPBACK window. Here, the LOOPBACK window is set to 15 days.

Figure 4.19 illustrates the predictions generated by the LSTM model, which was trained on stationary stock market data. It displays the model's performance in forecasting the NSE index over the subsequent 15 days after using the last five years of training data. The graph contrasts the actual NSE index values with the predicted figures and clearly

delineates the prediction window. Visually, the predicted trajectory shows a generally upward trend with a smoother transition compared to the more erratic and volatile actual market data. This demonstrates the LSTM model's ability to capture long-term patterns from stationary sequences while also highlighting its limitation in adequately addressing the sharp fluctuations typically found in financial time series.

The predictions overlook some of the actual index's turning points, leading to a noticeable disparity between the observed and predicted values in the highlighted forecast area. However, the LSTM model does succeed in maintaining a plausible forecast trajectory, suggesting that the preprocessing step of transforming the input data to be stationary has helped capture the broader temporal dependencies. The output is significantly more stable and smoother compared to when LSTM was trained on raw, non-stationary data. When compared to Figure 4.16, which depicts the LSTM’s performance with raw data, the results in Figure 4.19 are relatively more coherent, although still not perfectly accurate. Predictions based on raw data showed greater deviations from the true series, often diverging in both direction and magnitude. On the other hand, the model trained on stationary data exhibits improved alignment with the trend structure, even though it still underestimates the actual index's dynamic qualities.

In comparison to the related prediction plots from RNN (Figure 4.8) and ANN (Figure 4.5), the LSTM model demonstrates a significant enhancement in its capacity to maintain temporal coherence and yield less erratic forecasts. The predictions exhibit reduced noise and greater continuity, indicating that both the LSTM architecture and the stationarisation of input data play a role in fostering a more disciplined learning experience. Nevertheless, this figure underscores that LSTM alone, even with stationary inputs, falls

short of achieving high accuracy in volatile areas like the stock market. The improvement is evident yet not optimal, paving the way for more advanced temporal models such as GRUs, which aim to provide superior management of long-range dependencies and adaptability.

4.6 Creating a GRU Model on Raw and Stationary Data

The last architecture in this study—Gated Recurrent Units (GRU)—is used to evaluate its predictive accuracy on stock market data. GRUs are a type of Recurrent Neural Network (RNN) designed to address some of the common issues associated with conventional RNNs and Long Short-Term Memory (LSTM) networks. They maintain the capacity to model sequential data via a gating mechanism while presenting a more straightforward and computationally efficient framework than LSTMs. This inherent simplicity often results in quicker training and improved generalisation, especially in cases of limited or noisy data, making GRUs a compelling choice for time series forecasting tasks.

This section assesses the GRU model using both raw and stationary datasets, applying the same experimental setup as in previous models. The training involves a five-year historical window, with evaluation conducted over a one-month forward prediction horizon, utilising a sliding window method across the entire dataset. This consistent approach enables a direct comparison of GRU's performance against the results from ANN, RNN, and LSTM models. The objective is to determine if the GRU architecture can effectively capture temporal dependencies, particularly amidst volatility and non-

stationarity, and whether its configuration allows it to surpass the performance of the more complex LSTM when analysing financial data.

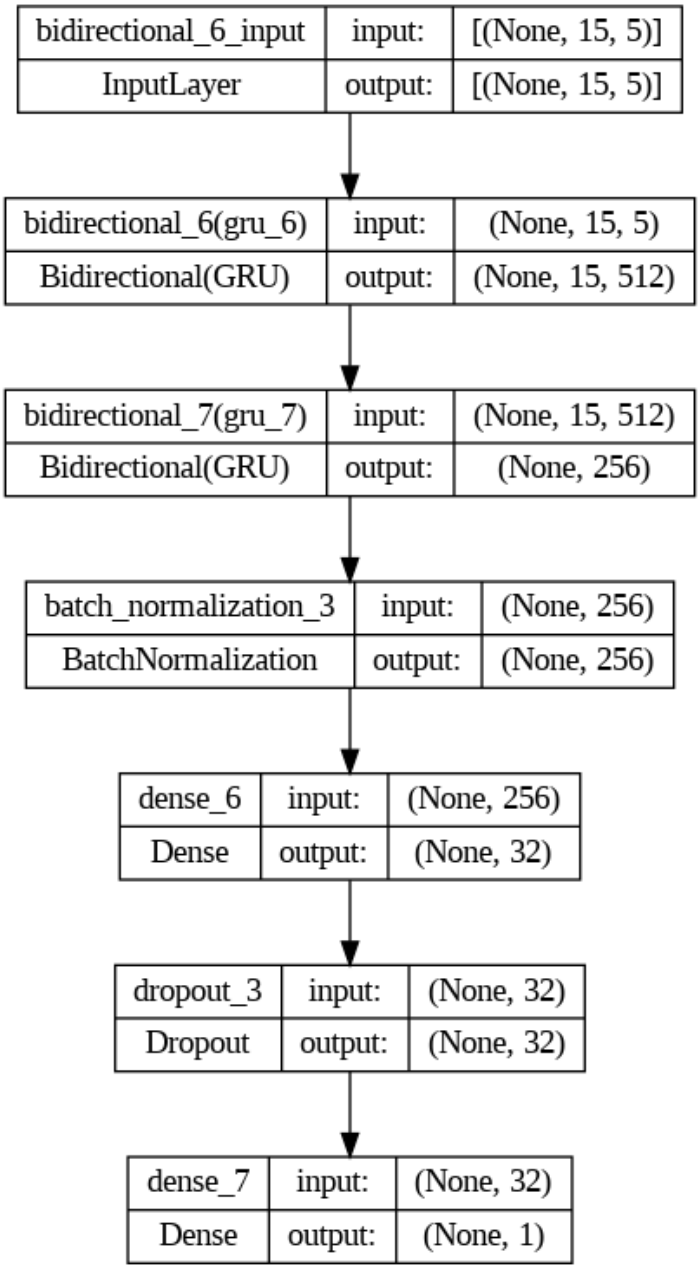

Figure 4. 20: Architecture of the GRU Model. This model was trained on both raw and stationarised data.

The results obtained after training the GRU model on raw data are stated in Table 4.10.

Train Start	Train End	Test Start	Test End	MSE	R^2
17/09/07	17/09/12	18/09/12	18/10/12	0.00026217	-247.49654
17/03/08	17/03/13	18/03/13	18/04/13	0.00055441	-41.858501
17/09/08	17/09/13	18/09/13	18/10/13	0.00011515	-18.454596
17/03/09	17/03/14	18/03/14	18/04/14	0.00066248	-198.68298
17/09/09	17/09/14	18/09/14	18/10/14	0.00332539	-7523.5406
17/03/10	17/03/15	18/03/15	18/04/15	0.00222484	-169.7333
17/09/10	17/09/15	18/09/15	18/10/15	9.45E-05	-17.704978
17/03/11	17/03/16	18/03/16	18/04/16	2.89E-05	-1.1964162
17/09/11	17/09/16	18/09/16	18/10/16	0.00079857	-138.14808
17/03/12	17/03/17	18/03/17	18/04/17	0.00035558	-89.036902
17/09/12	17/09/17	18/09/17	18/10/17	0.00220173	-154.33025
17/03/13	17/03/18	18/03/18	18/04/18	0.00113166	-302.90716
17/09/13	17/09/18	18/09/18	18/10/18	1.61E-05	-2.5650708
17/03/14	17/03/19	18/03/19	18/04/19	4.01E-05	-2.944937
17/09/14	17/09/19	18/09/19	18/10/19	0.00037579	-12.608351
17/03/15	17/03/20	18/03/20	18/04/20	0.00598744	-194.27448
17/09/15	17/09/20	18/09/20	18/10/20	0.00095269	-36.153104
17/03/16	17/03/21	18/03/21	18/04/21	0.00011428	-3.5583834
17/09/16	17/09/21	18/09/21	18/10/21	0.00831077	-138.7336
17/03/17	17/03/22	18/03/22	18/04/22	0.00647638	-108.05637
17/09/17	17/09/22	18/09/22	18/10/22	0.00026151	-4.5454146
17/03/18	17/03/23	18/03/23	18/04/23	0.00263434	-294.52419
17/09/18	17/09/23	18/09/23	18/10/23	0.0005745	-130.51247
17/03/19	17/03/24	18/03/24	18/04/24	0.00063006	-3.8673636
17/09/19	17/09/24	18/09/24	18/10/24	8.65E-05	-3.7006576

Table 4. 10: Statistics gathered by training a GRU on Raw Stock Market Data.

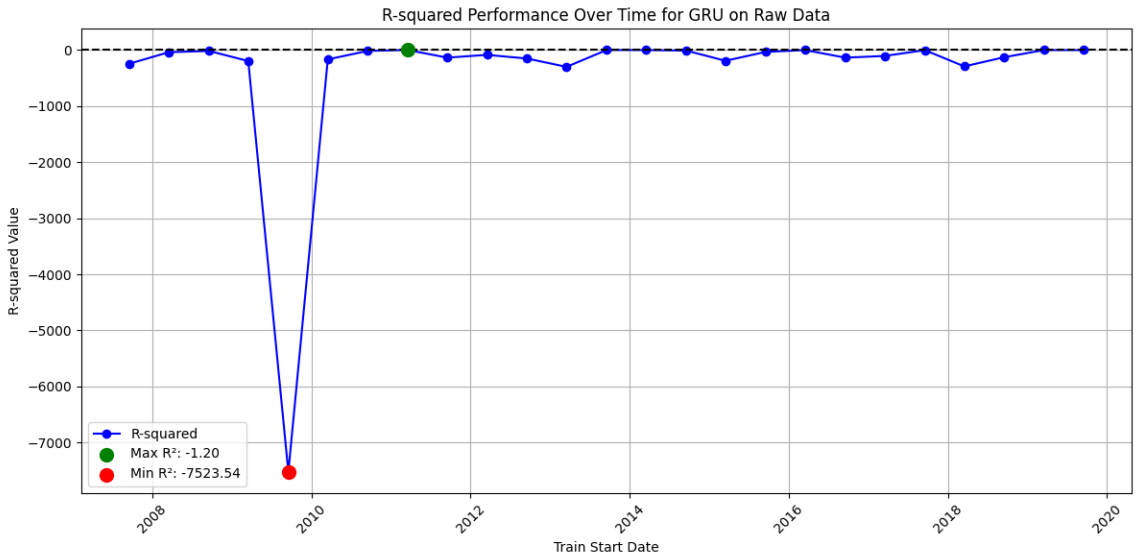

Figure 4. 21: Recording of the R^2 value for predictions made on the test data for each training data window using the GRU model on the raw data.

Figure 4.25 displays the R^2 values from predictions of the GRU model, which was trained on raw stock market data. The results show considerable fluctuations in model performance across various training windows, highlighting the GRU's sensitivity to temporal changes in market behaviour. Although some training windows yield moderately negative R^2 values, many exhibit severe underperformance, with the lowest R^2 value around -7523.54. This extremely negative score suggests that during specific training periods, the GRU model failed to identify meaningful data patterns, resulting in predictions that are worse than simply using the mean of the test data. Conversely, the highest R^2 value attained is -1.20, which, while an improvement, still indicates limited explanatory power. Overall, although the GRU architecture is robust and theoretically aligned with time series modelling due to its capacity for managing long-term dependencies, the raw input data likely limited its performance in this instance. The noisiness and non-stationary nature of the raw stock market data present challenges for even sophisticated architectures like the GRU in extracting stable and generalisable patterns without additional preprocessing.

Comparing these findings to previous experiments with simpler architectures like LSTM, RNN, and ANN using raw input, a distinct pattern appears. Although the GRU features an advanced gating mechanism, its performance on raw data did not notably exceed that of the other models. In fact, the occurrence of extremely low R^2 values, like -7523.54, indicates a level of volatility that was less pronounced in earlier models. For example, both RNN and LSTM models faced challenges with raw data, yet the fluctuations in their R^2 scores were slightly more stable. This finding emphasises a crucial point of this research: preprocessing data to achieve stationarity is essential for realising the full potential of machine learning architectures. GRU's theoretical advantages do not necessarily result in practical benefits when applied to unprocessed time series data, reinforcing the idea that the statistical characteristics of this data significantly influence model performance. Therefore, while GRU models may ultimately showcase the greatest capability, this advantage primarily surfaces when the input data is thoroughly processed and made stationary.

Another key observation across all models, including ANN, RNN, LSTM, and GRU, is a notable decline in R^2 values within the same training window, coinciding with the 2008 global financial crisis. This widespread underperformance during this timeframe indicates that external factors, specifically the extreme market volatility and structural disruptions due to the economic downturn, significantly impacted model behaviour. Trained on five-year windows leading up to the crisis, the models struggled to anticipate the abrupt and chaotic shifts in the stock market that followed. This underscores a major limitation of data-driven time series models: during economic crises, the assumptions of statistical continuity break down, making learned patterns less effective or potentially misleading. The marked decline in model accuracy during this time is not necessarily a

flaw in the architecture itself but rather a reflection of the unpredictability and structural changes caused by macroeconomic shocks. Understanding this historical context is essential when assessing the validity and resilience of machine learning predictions, especially in practical financial scenarios.

The data captured regarding the training cycles to see if it corroborates our understanding so far is shown in Figure 4.22.

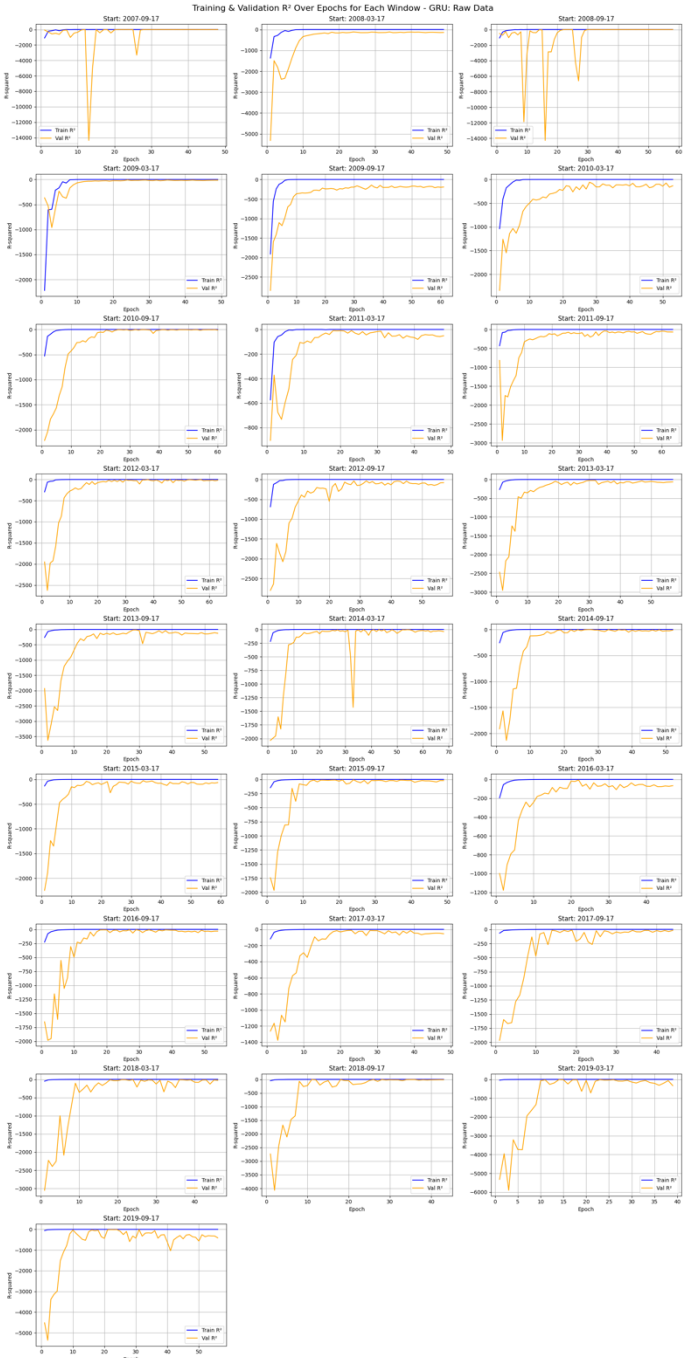

Figure 4. 22: Training and Validation R^2 observed while training the GRU model on raw stock market data.

Figure 4.26 illustrates the R^2 values for training and validation across multiple epochs for each training window during the GRU model's training on raw stock market data. This chart emphasises the model's performance and convergence behaviour across different time segments of the dataset. Although the training R^2 curves show a consistent and rapid increase, successfully converging to high values, there is a striking difference with the validation R^2 curves, which exhibit significant fluctuations. In numerous windows, the validation R^2 scores remain significantly negative, indicating the model's inadequate ability to generalise from unseen data. Several training windows demonstrate a drastic drop in validation R^2 scores, with some even descending into the negative thousands, suggesting a failure to capture any significant patterns in the validation set.

A detailed examination shows that the GRU model effectively learns from training data; however, its ability to generalise across varying time windows is inconsistent. This discrepancy highlights overfitting as a major concern: the model captures noise in the training data too well, failing to identify enduring patterns over time. Furthermore, during several training windows, the validation R^2 values show extreme negatives, particularly during significant financial upheavals like the 2008 global recession, which likely introduced chaotic and unpredictable market behaviour. These economic shocks disrupt statistical regularities, complicating the modelling of data with machine learning, notably when no preprocessing, such as stationarisation, occurs. This observation aligns with findings from other models trained on raw data, which also exhibited poor performance during and around 2008. This consistency emphasises a key insight from our research: raw stock market data presents significant challenges to even advanced machine learning models due to its non-stationary character and vulnerability to sudden systemic shocks.

Without proper preprocessing, the GRU's full potential remains unexploited, underscoring the necessity of converting data into a stationary format before training.

It is paramount to make the stock market data stationary before modelling it. Before proceeding to that, let us examine the predictions made by the GRU model trained on the raw stock market data, as shown in Figure 4.23.

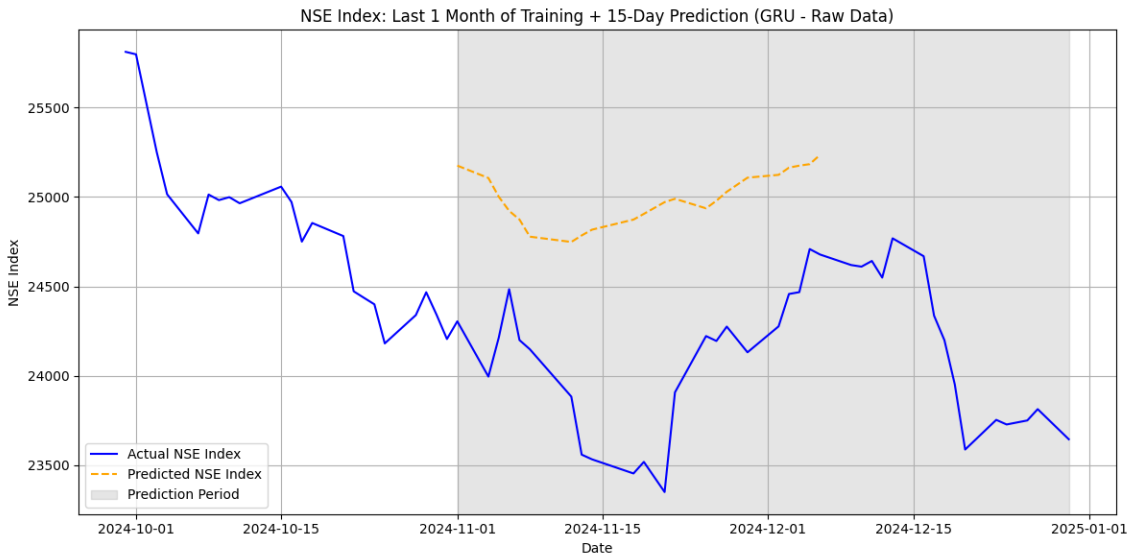

Figure 4. 23: Predictions made by the GRU Model trained on the raw stock market data. The prediction window is different from that used for ANN because GRUs need a LOOPBACK window. Here, the LOOPBACK window is set to 15 days.

Figure 4.23 displays the 15-day forecasts made by the GRU model, which was trained on raw stock market data, alongside the actual NSE index values for the same timeframe. Despite the model consistently overestimating the true index values, resulting in a noticeable upward bias, the overall market trend is effectively represented. The predicted values from the model exhibit a comparable upward curve to the actual data, indicating that the GRU model can learn and mirror some of the temporal dynamics of

stock market behaviour, even when operating on noisy and non-stationary data. This achievement is significant given the high volatility and unpredictability typical of financial time series, especially in their raw form.

In comparison to the prediction charts of other models using raw data, the GRU's output, shown in Figure 4.23, exhibits better trend detection skills. For example, while the LSTM and RNN models occasionally aligned directionally with the actual data, they frequently failed to show consistent upward or downward trends, instead resulting in jagged or delayed responses. The GRU's smooth path and overall alignment with market direction indicate a greater level of temporal awareness, which aligns with the inherent advantages of GRUs in managing long-term dependencies and maintaining relevant memory across sequential data.

The discrepancies in the predictions reveal a significant limitation of using raw, unprocessed stock data for training. The noise and non-stationary features in the raw input hinder the model's ability to accurately adjust its numerical outputs, resulting in a systematic prediction bias. Nonetheless, the GRU demonstrates a superior capacity to follow trends compared to ANN, RNN, and LSTM when they are configured with raw data, supporting the idea that GRU is the most effective architecture among those tested. However, as repeatedly noted in this research, this potential is optimally harnessed only after the data has been transformed into a stationary format.

The last experiment is to train the GRU model on stationarised stock market data. The statistics collected are provided below.

Train Start	Train End	Test Start	Test End	MSE	R^2
18/09/07	18/09/12	19/09/12	19/10/12	0.00028307	**0.01143601**
18/03/08	18/03/13	19/03/13	19/04/13	0.00130397	-0.5790453
18/09/08	18/09/13	19/09/13	19/10/13	0.00153406	-0.3022787
18/03/09	18/03/14	19/03/14	19/04/14	0.00073365	-0.0012991
18/09/09	18/09/14	19/09/14	19/10/14	7.72E-05	
18/03/10	18/03/15	19/03/15	19/04/15	0.00230879	-0.6232608
18/09/10	18/09/15	19/09/15	19/10/15	0.00048723	-9.6345098
18/03/11	18/03/16	19/03/16	19/04/16	0.00268659	-7.0572775
18/09/11	18/09/16	19/09/16	19/10/16	0.00202936	-0.0231727
18/03/12	18/03/17	19/03/17	19/04/17	0.00044194	-0.6315239
18/09/12	18/09/17	19/09/17	19/10/17	0.00087256	-0.0712685
18/03/13	18/03/18	19/03/18	19/04/18	0.00016312	-0.5216859
18/09/13	18/09/18	19/09/18	19/10/18	0.00290588	-0.3206673
18/03/14	18/03/19	19/03/19	19/04/19	0.00043409	-0.0319137
18/09/14	18/09/19	19/09/19	19/10/19	0.00240404	-7.1054504
18/03/15	18/03/20	19/03/20	19/04/20	0.00565404	-0.2759652
18/09/15	18/09/20	19/09/20	19/10/20	0.00601674	-0.0366916
18/03/16	18/03/21	19/03/21	19/04/21	0.00639944	-0.0316792
18/09/16	18/09/21	19/09/21	19/10/21	0.00255617	-0.5367569
18/03/17	18/03/22	19/03/22	19/04/22	0.0084916	-4.1410481
18/09/17	18/09/22	19/09/22	19/10/22	0.003445	-0.5027798
18/03/18	18/03/23	19/03/23	19/04/23	0.00119235	-1.2604444
18/09/18	18/09/23	19/09/23	19/10/23	0.00186302	-0.6867049
18/03/19	18/03/24	19/03/24	19/04/24	0.00931366	-1.0082315
18/09/19	18/09/24	19/09/24	19/10/24	0.00437799	-0.3422241

Table 4. 11: Statistics gathered by training a GRU on Stationary Stock Market Data.

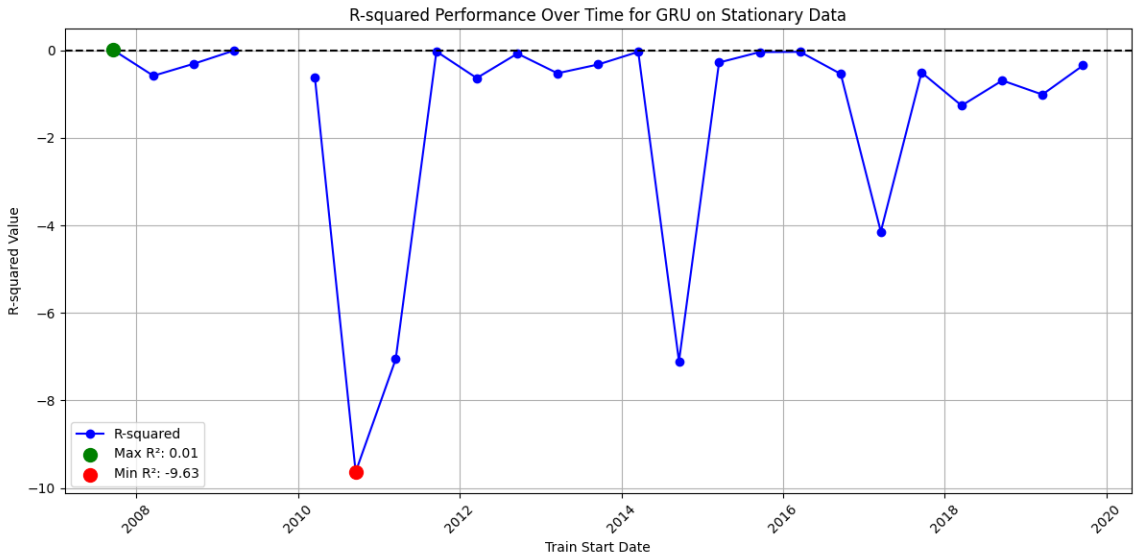

Figure 4. 24: Recording of the R^2 value for predictions made on the test data for each training data window using the GRU model on the stationary data.

Figure 4.24 illustrates the R^2 values generated by the GRU model when it is trained on stationary stock market data, utilising various temporal windows. Unlike the unpredictable and often extreme values witnessed during GRU training on raw data (noted earlier in Figure 4.21), the results displayed in this figure indicate a marked enhancement in stability and predictive reliability. Although many R^2 values remain predominantly negative, they are now much closer to zero, with the highest value even reaching a positive 0.01. The least effective window results in a comparatively modest R^2 value of -9.63, which is significantly milder than those observed with raw data. This overall narrowing of the R^2 score range suggests that the model is better equipped to generalise across different time frames, even if the predictions themselves are not particularly robust in absolute terms.

The GRU architecture greatly benefits from the transition from raw to stationary data. As outlined in this thesis, stock market data is characterised by inherent noise and non-stationarity, which complicates the extraction of meaningful signals, even for

sophisticated neural networks. This study employs differencing techniques for stationarisation, eliminating underlying trends and variance shifts that obscure data relationships. After addressing these distortions, the GRU model effectively maintains temporal dependencies, avoiding being overwhelmed by the volatility and randomness found in the untreated time series.

Additionally, as illustrated in Figure 4.24, when compared to other models used for stationary data, like RNNs and LSTMs, the GRU model distinguishes itself by yielding more consistent and narrower R^2 value distributions. This further validates the assertion that GRU surpasses its predecessors. Although none of the models trained with stationary data achieve notably high R^2 scores, the GRU demonstrates improved range compression and peak performance, highlighting its superior capacity to fit the processed data. This reinforces the primary argument of the research that preprocessing stock market data to attain stationarity allows neural network models, especially GRUs, to operate more effectively and offer more dependable forecasting outcomes. Thus, Figure 4.24 marks a crucial point in this comparative study and fortifies the conclusion that GRU > LSTM > RNN > ANN.

This conclusion is reaffirmed with the data collected during the training cycles.

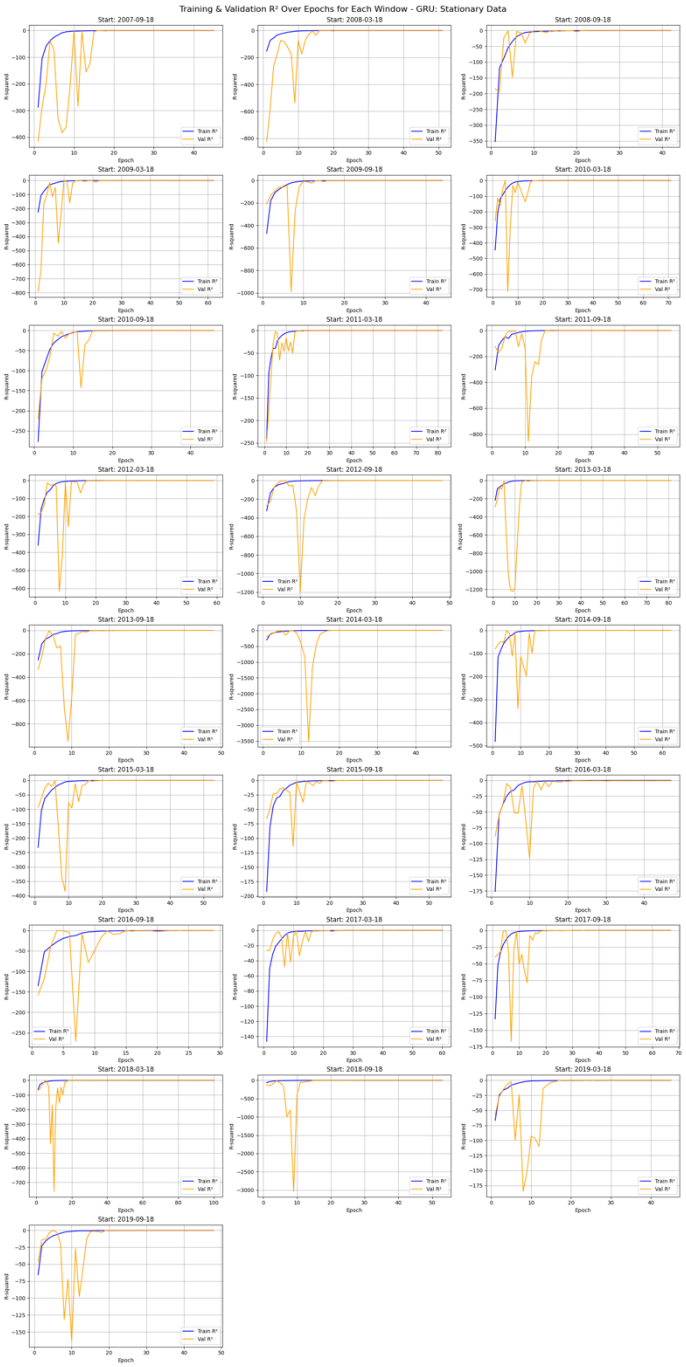

Figure 4. 25: Training and Validation R^2 observed while training the GRU model on stationary stock market data.

Figure 4.25 displays the R^2 values recorded during the GRU model's training on stationary stock market data over various training windows. In contrast to the results from training on raw data, the performance here shows significantly greater stability and less fluctuation. Although the R^2 scores do not reach high levels of predictive power, mostly hovering around or just below zero, the lack of drastic negative values indicates that the model is at least yielding more consistent and trustworthy outcomes. This consistency is particularly clear when examining the plots of training and validation R^2 over epochs; most training windows reveal smooth convergence patterns, and while validation scores may experience occasional minor drops, these are less severe compared to those observed in previous models trained on raw data.

The enhanced performance can be directly linked to the preprocessing phase that renders the data stationary. By eliminating trends and stabilising variance throughout the series, the GRU model can concentrate on identifying the core temporal patterns without the distraction of non-stationary fluctuations. This leads to a model that generalises better and reduces the risk of overfitting or underfitting, which frequently occurs when training on unstable raw data. Additionally, the convergence curves indicate that the GRU effectively manages the stationary inputs, with training losses quickly stabilising across most windows. This suggests that the model is well-optimised for the task after the data has been processed.

This performance also contrasts favourably with the GRU model trained on raw data, as depicted in Figure 4.22. There, the R^2 values dipped to extreme lows, indicating to severe inability to generalise in several cases. In contrast, the GRU model trained on stationary data, as seen in Figure 4.25, avoids such dramatic failures, which substantiates

the central argument of this thesis: stationarising stock market data enhances the effectiveness of machine learning models. Moreover, when viewed in the broader context of this study, the GRU model on stationary data appears to outperform its ANN, RNN, and LSTM counterparts across various training windows. This aligns with the overarching claim that GRU architecture is more advanced and capable of sequential modelling, especially when paired with suitable data preprocessing techniques. Therefore, Figure 4.25 bolsters the dual conclusions that both the GRU architecture and the stationarisation of data are crucial for achieving reliable and consistent stock market predictions.

To conclude this research, let us examine the predictions made by the GRU model trained on stationary stock market data.

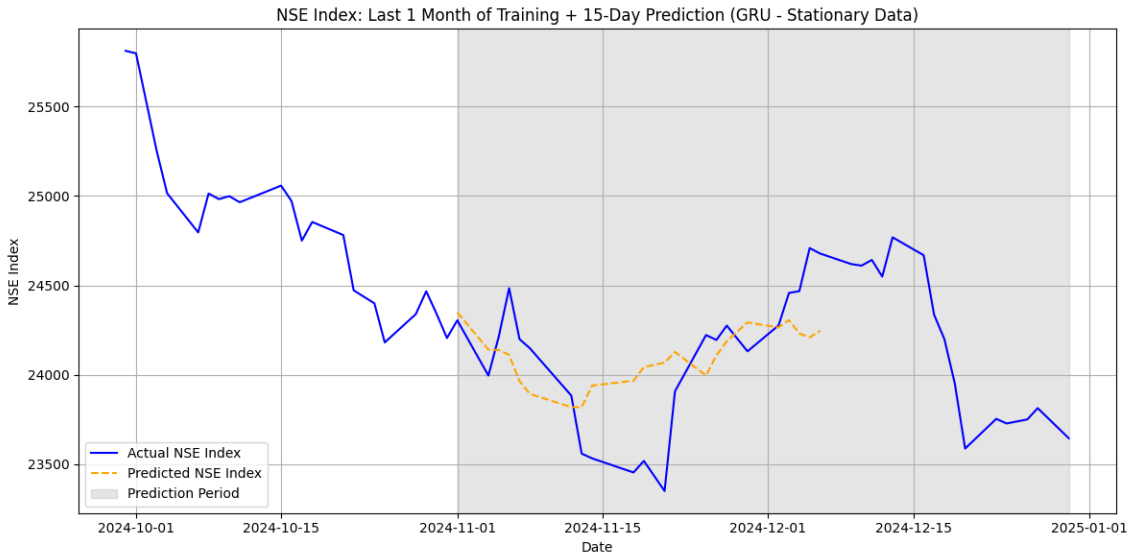

Figure 4. 26: Predictions made by the GRU Model trained on the stationary stock market data. The prediction window is different from that used for ANN because GRUs need a LOOPBACK window. Here, the LOOPBACK window is set to 15 days.

Figure 4.26 illustrates the forecasting results of the GRU model, which was trained on stationary stock market data to predict the NSE Index over 15 days. This chart synthesises all previous experiments and validations within this thesis, showcasing the predictive capability of the most advanced model applied to the most refined dataset. As shown in the figure, the GRU model effectively captures the market's overall trend. The predicted values align closely with the actual NSE Index's directional movements, indicating that the model has adeptly internalised the underlying structure of the stationary time series. While there is a noticeable offset between the actual and predicted values, the general curve shape and turning points are well-matched. This illustrates the model’s proficiency in learning from the preprocessed data and projecting future values coherently and meaningfully.

The results show a significant enhancement compared to the predictions from the GRU model that was trained on raw data, as illustrated in Figure 4.23. Although this model demonstrated some awareness of the trend, its predictions had a greater offset and lacked the smooth consistency seen in Figure 4.26. Additionally, the predictions in this case are considerably more stable and coherent than those generated by the LSTM, RNN, or ANN models trained on either raw or stationary data. While those models either struggled to reflect the trend accurately or yielded erratic predictions, the GRU model trained on stationary data demonstrates a much better understanding of the market's directional momentum. It surpasses the previous models in aligning with the actual trend, exhibiting lower volatility and more precise predictions, which indicates a higher degree of generalisation and learning.

The figure highlights a key finding of this research: the GRU architecture, trained on stationary stock market data, proves to be the most effective among all the models evaluated. Its success derives from structural advantages and the synergy that arises from proper data transformation. By ensuring the data is stationary, the model is freed from needing to address shifts in mean and variance, enabling it to concentrate on identifying and forecasting significant temporal patterns. This supports the research's twofold hypothesis that preprocessing data for stationarity greatly improves prediction accuracy and that GRU surpasses other neural network architectures in modelling financial time series data.

4.7 Research Question One: Does Making Stock Market Stationary Impact Stock Market Models?

The first research question is, “How does making stock market time series data stationary impact the accuracy of machine learning-based stock market predictions?” Let’s analyse the findings across the different models developed.

Let us consider the different models developed on raw data and stationary data. Below are the plots of the achieved R^2 for the different training windows.

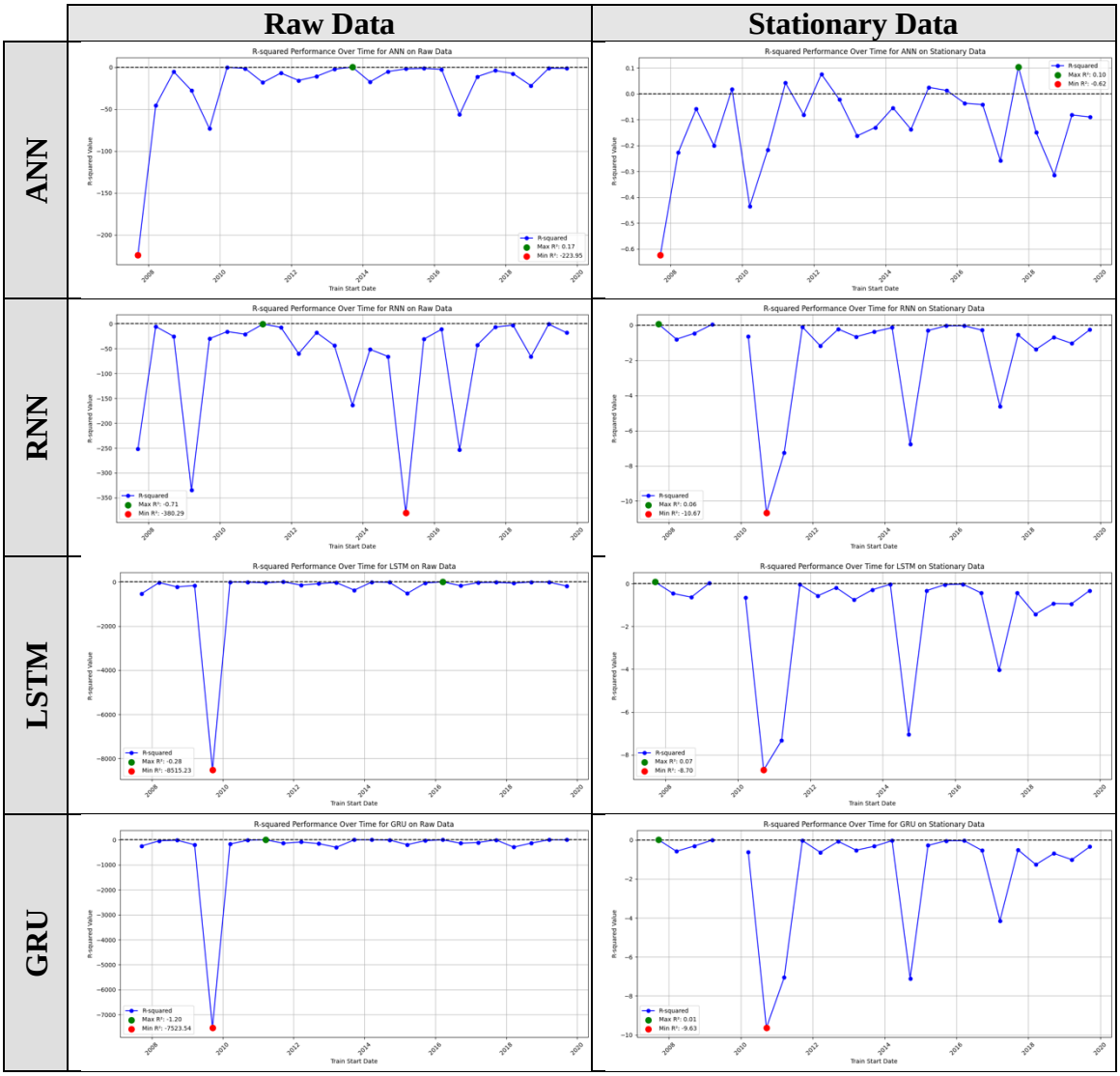

Table 4. 12: Statistics gathered during training of the different models in different data conditions over different training windows.

To address the research question, we must consider the empirical evidence gathered during this study. The comparative analysis of models trained on both raw and stationary

data indicates that converting stock market time series into a stationary format markedly improves model performance. This trend is evident across all model architectures examined, including ANN, RNN, LSTM, and GRU. The performance metrics, represented by R^2 values and illustrated in Table 4.5, show a consistent enhancement when data is transformed into stationary form prior to training. This enhancement is reflected in more stable R^2 scores, decreased variance between training and validation results, and smoother learning curves.

When raw data is used, all the models, regardless of the complexity, struggle to fit consistently. For example, GRU and LSTM, despite their theoretical strength in modelling temporal dependencies, suffer from sharp drops in R^2 performance, at times resulting in extremely negative scores that indicate complete failure in capturing patterns. These breakdowns are significantly mitigated when the models are trained on stationary data. The figures show tighter clustering of R^2 values, improved convergence, and better alignment between training and validation performance. Furthermore, the predictions over the 15-day horizon, such as those in Figures 4.19 and 4.26, show marked improvement in trend approximation when stationarised data is used. While the GRU model trained on raw data generally captures the trend, its outputs are noticeably off and lack confidence. Conversely, the same model trained on stationary data aligns more closely with the actual data, minimising excessive deviations and noise and resulting in more reliable forecasts.

These findings reinforce the idea that stationarity is essential for effective time series forecasting with machine learning. While models like GRU are powerful, they are very sensitive to the statistical properties of the input data. When attributes such as mean and variance fluctuate over time, it creates instability that deep learning models often

struggle to manage, especially when trained on limited data. Transforming the data to be stationary reduces such variability, enhancing the learning process and leading to more reliable predictions. Consequently, this research indicates that ***making stock market data stationary greatly boosts the effectiveness and reliability of machine learning forecasting models***.

4.8 Research Question Two: Is GRU better than ANN, RNN, and LSTM for Stock Market Predictions?

The second research question is, “How do GRU-based models compare to other machine learning approaches, such as ANN, RNN, and LSTM, in predicting stock prices and indices?” Let’s analyse the findings across the different models developed.

Let us consider the different models developed on raw data and stationary data. Below are the predictions made by the different models.

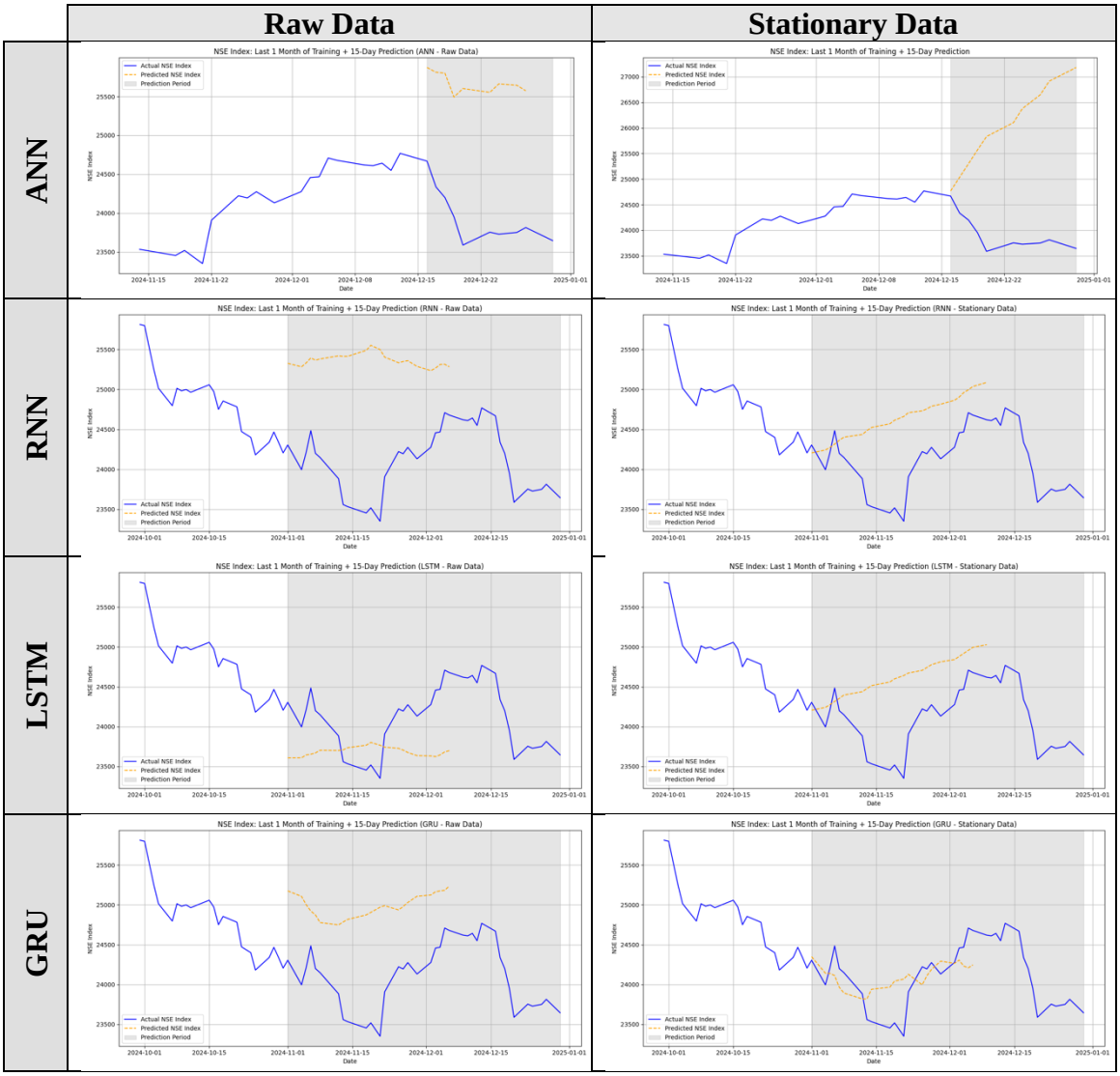

Table 4. 13: Predictions made by the different models on the latest data in the dataset based on the preceding 5-year training window.

To answer the research question, we must examine both the quantitative performance and the qualitative forecasting behaviour of all four models across raw and stationary data treatments.

Throughout the experiments, GRU consistently delivered the most reliable and accurate outcomes, particularly with stationary stock market data. This conclusion is supported by various layers of evidence found in the results. We begin by analysing the R^2 values over time for each model on both raw and stationary datasets. In the stationary scenario, GRU exhibited the narrowest range of variation and the least frequent occurrence of extreme negative values. Conversely, ANN exhibited weak performance even with stationary data. While RNN and LSTM showed some improvement with stationary data, they still experienced volatility and significant negative R^2 dips. Notably, GRU achieved the highest maximum R^2 value across all experiments, suggesting its superior ability to extract meaningful predictive signals compared to other models.

Additionally, when assessing the predictions from each model, the GRU's forecasts aligned most closely with the actual market trend, particularly when it was trained on stationary data. In contrast, models like ANN yielded almost flat or inaccurately directed predictions, while LSTM also showed a noticeable discrepancy. The GRU effectively mirrored the true movement direction of the NSE Index with commendable accuracy. This performance is crucial in financial contexts, where identifying trend directions and turning points is often more important than predicting exact index levels.

Additionally, GRU's architecture features optimised gating mechanisms that effectively learn long-term dependencies while minimising issues like overfitting and

vanishing gradients. This allows GRU to surpass RNN and LSTM in terms of learning stability and convergence. Although its structure is simpler than that of LSTM, GRU offers faster training and demands fewer computational resources, all while maintaining comparable, if not better, accuracy. This is illustrated by the training graphs, where GRU achieves stable training and validation R^2 values sooner and with less variability than the other models, especially in the stationary setup.

The evidence clearly demonstrates that GRU surpasses ANN, RNN, and LSTM in stock market prediction using machine learning. Its higher R^2 scores, improved alignment with real market trends, quicker convergence, and superior robustness to non-linear patterns highlight GRU as the best architecture among those examined. This supports the initial hypothesis: **GRU > LSTM > RNN > ANN** for stock market predictions.

4.9 Research Question Three: Limitations of Predicting the Stock Market Using this Research Methodology

This study recognises various practical, methodological, and conceptual limitations that affect the findings' generalisability and strength while addressing this research question.

A major limitation lies in the very nature of the stock market. Financial markets are naturally volatile and influenced by a variety of factors that go beyond historical price and macroeconomic data. Although this research used a comprehensive dataset that includes stock indices, commodity prices, exchange rates, and GDP figures, it fails to account for

sentiment-driven factors like investor psychology, news events, or geopolitical disruptions. These qualitative and event-driven aspects significantly affect market behaviour, especially during crises or economic transitions, and their omission reduces the model's ability to respond to real-world complexities.

Another limitation is the reliance on historical data for both training and validation. Although the methodology employs a sliding window approach to adapt to changing market conditions, it cannot completely predict unforeseen events like financial crashes, pandemics, or sudden regulatory shifts. This temporal constraint implies that the predictive models may struggle in situations that significantly differ from historical trends, as the training data does not capture the dynamics of those circumstances.

Additionally, ensuring the data is stationary greatly enhanced the performance of all neural network models evaluated in this research. However, the stationarisation process of achieving stationarity also brings about a degree of subjectivity. Choices regarding differencing, selecting stationarity tests, and managing any residual non-stationary behaviour can all influence model results. The methods used to reach stationarity might unintentionally eliminate valuable long-term signals in the data that could enhance predictive accuracy.

Machine learning models introduce challenges related to computation and interpretability. Deep Learning structures, such as GRUs, although effective, are frequently labelled as "Black Boxes" because their decision-making processes lack transparency. This obscurity hampers financial analysts and institutional investors from fully trusting and understanding the predictions generated by these models, particularly in crucial decision-

making scenarios. Additionally, training these models demands substantial computational power and time, making it impractical for various financial institutions and individual investors.

Ultimately, this study focuses exclusively on the Indian stock market by utilising NSE index data. Although the findings are insightful, they cannot necessarily be applied to other markets that feature varying structural, regulatory, and economic environments. It may be necessary to adjust or redesign the models to address the unique characteristics of global stock exchanges or to incorporate a more extensive array of market indicators.

This research highlights the importance of stationarising stock market data and using GRU-based models for predictions. However, it is crucial to acknowledge its limitations. While the methodology marks a notable improvement in model accuracy, future studies should expand upon this by integrating sentiment analysis, real-time data feeds, and cross-market comparisons to fully realise the potential of machine learning in financial forecasting.

4.10 Summary of Findings

The research findings highlight key insights regarding the intersection between data preprocessing and deep learning models in stock market prediction. A predominant theme from rigorous experimentation and comparisons is the essential impact of data stationarity on increasing predictive accuracy. For all model types – ANN, RNN, LSTM, and GRU – it was consistently noted that performance was enhanced when the stock market data was

converted into a stationary format. Making the data stationary mitigated trends and volatility, allowing the models to uncover inherent patterns more clearly and consistently. This result supports the study's main hypothesis that transforming financial time series into stationary formats before training models improves forecasting precision.

Among the evaluated models, GRU emerged as the most effective architecture for time series forecasting, particularly with stationary data. It yielded the most stable R^2 values across varying training windows and consistently identified market trends in its predictions. While models like LSTM and RNN demonstrated some improvements with stationary data, they were less robust and more susceptible to underfitting or overfitting during specific periods. In contrast, ANN struggled to provide consistent outcomes, even with stationary inputs, underscoring its limitations in sequential learning tasks. These results further support the hierarchy outlined in the research: GRU outperforms LSTM, which surpasses RNN, followed by ANN.

A key takeaway was the critical role of data quality and preprocessing. Raw financial time series frequently showed erratic fluctuations and structural breaks that obstructed model learning, particularly for complex architectures. This was reflected in the inconsistent R^2 scores and unstable training behaviour noted with raw data. By transforming the data to be stationary, the models showed better generalisation, quicker convergence, and enhanced predictive stability. Moreover, the models were able to maintain the trend directionally in their forecasts, even if the absolute values had certain discrepancies. This result is especially important in financial decision-making, where directional accuracy is often more crucial than absolute values.

The study ultimately demonstrates that combining data transformation with advanced deep learning architectures, especially GRU, forms a robust methodology for stock market forecasting. This research adds to the increasing evidence favouring machine learning in finance and underscores the importance of careful data preparation. These results confirm the theoretical principles established in the existing literature and offer valuable insights for future development and implementation of models in practical financial forecasting scenarios.

4.11 Conclusion

This research concludes by highlighting the key insights obtained through systematic experimentation, comparative analysis, and theoretical foundation in time series forecasting via machine learning methods. A crucial finding of this study is that transforming stock market data into a stationary format greatly enhances the predictive accuracy of machine learning models. This conclusion is backed by empirical evidence showing improved model performance, assessed through R^2 and Mean Squared Error (MSE) metrics, when utilising stationary data rather than raw time series. Models developed with stationary data demonstrated greater stability, quicker convergence, and a closer alignment with actual stock market movements, emphasising the importance of this preprocessing step in financial forecasting.

The research further demonstrates that within the tested neural network architectures, GRU models surpass others, specifically ANN, RNN, and LSTM, across both raw and stationary datasets. GRU's superiority is evident in its predictive metrics and

its capacity to capture directional trends while generalising effectively across various training windows. These findings support the hypothesis that GRU, owing to its efficient gating mechanism and capability to handle long-term dependencies, provides a more dependable framework for stock market forecasting, particularly when used with appropriately transformed data.

Additionally, the study highlights the drawbacks of directly using raw stock market data in predictive models. This raw data, marked by non-stationary behaviour and noise, causes instability that hampers model learning and generalisation. This issue was evidenced by very low R^2 values and poor prediction alignment across models trained on raw data. In contrast, converting the data to achieve stationarity eliminates this noise and enables models to concentrate on significant temporal patterns, improving forecasting accuracy.

This research ultimately offers a practical and scalable approach to financial prediction, effectively merging robust statistical preprocessing with cutting-edge neural network designs. It demonstrates that accurate predictions in the stock market are achievable through a careful blend of data transformation and deep learning techniques. Despite existing challenges, notably in capturing sentiment and unpredictable events, the findings pave the way for future research opportunities, such as hybrid models and multi-modal datasets. Consequently, this study enhances the knowledge base in financial analytics and lays a solid groundwork for developing effective machine-learning models for stock market forecasting.

CHAPTER V:
DISCUSSION

5.1 Discussion of Results

This study's results offer valuable insights into stock market prediction dynamics using machine learning, especially regarding time series data transformations. A key theme from the research is the notable enhancement in model performance following the stationarisation of stock market data. For all tested models – ANN, RNN, LSTM, and GRU – transitioning from raw, non-stationary data to a stationary format significantly improved their predictive accuracy. Models based on raw data often showed erratic R^2 values, characterised by sharp fluctuations and frequent underperformance, especially in volatile market conditions. Conversely, models trained on stationary data demonstrated more stable performance, indicating clearer pattern extraction and diminished noise sensitivity.

The evolution of model architectures significantly influenced outcomes. Although ANNs provided a useful baseline, their lack of ability to capture temporal dependencies rendered them the least effective, particularly in unstable financial contexts. RNNs enhanced this by integrating sequential learning; however, they struggled with vanishing gradient issues over long-term observations. LSTM models improved on this with sophisticated memory retention features, resulting in better performance, especially with stationary data, but still encountered some convergence challenges. GRU models emerged as the most efficient option, balancing computational simplicity with deep temporal learning. GRUs consistently achieved superior R^2 scores and trend-following predictions,

particularly when trained on stationary datasets, confirming their effectiveness for advanced time series forecasting in finance.

As observed in the results, model performance declined significantly during the 2008 financial crisis, marked by a significant drop in R^2 values across all models. This trend underscores a key limitation of models based on historical data: their failure to forecast or account for black swan events that diverge from established patterns. Nevertheless, the overall trend bolsters the hypothesis that effective data preprocessing and model architecture are vital components for successful predictions in the stock market.

The results show that although no model can completely counteract the unpredictable characteristics of financial markets, meticulously preparing data, especially by ensuring it is stationary, alongside diligent model selection, can significantly enhance the reliability and accuracy of predictions. This underscores the key contributions of this study and strongly supports the use of data preprocessing techniques and sophisticated neural network architectures in financial forecasting endeavours.

5.2 Discussion of Research Question One: Does Making Stock Market Stationary Impact Stock Market Models?

The research results clearly show that transforming stock market data into a stationary format greatly enhances the effectiveness of predictive models. This finding has been consistently validated across various experiments involving different machine learning architectures, such as ANN, RNN, LSTM, and GRU. Regardless of the

architecture employed, both training and testing outcomes showed notable improvement when the data underwent stationarised techniques like differencing. The performance metrics, particularly the R^2 values, demonstrated greater stability and achieved higher peaks for models trained on stationary data in comparison to those trained on raw data.

This discovery supports the study's main hypothesis and is consistent with the theoretical foundations of time series analysis. When the statistical characteristics of the data are stabilised over time, machine learning models can more effectively recognise significant patterns and learn from them. This transformation particularly enhanced the prediction plots, as models trained on stationary data demonstrated improved accuracy in capturing the trends and fluctuations of the stock market index, despite some models showing offsets.

At this juncture, we reiterate the importance of stationarising time series data, a crucial step in developing dependable and effective stock market prediction models, which has been extensively validated and discussed in this thesis. This transformation process contributes significantly to the increased accuracy noted in the findings.

5.3 Discussion of Research Question Two: Is GRU better than ANN, RNN, and LSTM for Stock Market Predictions?

This study's analysis offers compelling evidence that GRU models outperform ANN, RNN, and LSTM models in stock market predictions, particularly when trained on stationary data. Throughout the experiments detailed in this thesis, GRU consistently

achieved higher R^2 scores, aligned trends more accurately in its predictions, and showcased better overall stability across various training windows. While all models benefited from the preprocessing that made the data stationary, GRU maintained its advantage even with the unprocessed dataset. However, limitations arose from the inherent noise and non-stationarity present in such data. Its architecture, which is designed to manage long-term dependencies while avoiding the vanishing gradient issue, allowed it to capture temporal patterns more efficiently than the other models.

When comparing GRU to ANN, RNN, and LSTM, the advantages in predictive accuracy and robustness stood out. While ANN models are fast to train, they lack the sequential learning needed for time series forecasting. RNNs showed improved performance but had difficulties with long sequences. LSTMs resolved these issues through advanced memory management but also increased complexity and training durations. GRU models strike a balance between performance and efficiency, featuring a streamlined gating mechanism that enhances scalability and adaptability to the dynamic nature of financial time series data. This advantage was especially apparent when evaluating the final predictions from each model across the same test periods.

While GRUS have outperformed traditional models in this study, there are still opportunities for improvement. Expanding this research to include modern architectures and hybrid methods could lead to even better outcomes. For instance, merging Convolutional Neural Networks (CNN) with GRUs might enhance local pattern detection prior to processing the data with a temporal model. Likewise, incorporating attention mechanisms into GRU designs could enable the model to concentrate on the most significant parts of the input sequence, thereby enhancing both interpretability and

accuracy. Furthermore, utilising transformer-based models, which have transformed sequence modelling in natural language processing, may offer a new and potentially more effective approach to financial forecasting.

5.4 Discussion of Research Question Three: Limitations of Predicting the Stock Market Using this Research Methodology

This study reveals significant limitations in the methodology applied for predicting stock market trends through machine learning models. Though the research effectively demonstrates that stationarising stock market data enhances prediction accuracy and that GRU models outperform alternative architectures, there remain challenges that limit the reliability and generalisability of the results. A key limitation is the exclusive dependence on historical numerical data. While the study meticulously gathers a dataset that includes stock indices, macroeconomic indicators, and commodity prices, it overlooks qualitative factors like investor sentiment, geopolitical changes, or unexpected policy adjustments. These external influences can dramatically impact stock prices, and their exclusion from the model may create gaps in predictive accuracy, especially during periods of significant market fluctuations.

Another drawback is that machine learning models, notably deep learning structures such as GRU, tend to be vulnerable during times of severe economic upheaval. This became clear in various segments of the analysis, particularly during the 2008 financial crisis, when R^2 values for all models fell sharply. Although these patterns were clear and consistent, the models did not adequately consider the unpredictable nature of

such events. This underscores a larger problem in depending exclusively on historical data to predict outcomes in scenarios where irregular shocks shape market dynamics.

This study highlights the effectiveness of using differencing for achieving stationarity, yet it requires subjective choices regarding parameter selection and the interpretation of tests like the ADF and KPSS. The results of these tests can differ based on data structure or underlying assumptions. Consequently, the ability to reproduce specific modelling results across varying market conditions or datasets might be compromised. Furthermore, the black-box nature of deep learning models, such as GRU, poses interpretability challenges. Although performance metrics can affirm prediction accuracy, grasping the reasons behind specific predictions is a major obstacle, which hampers the use of such models in critical financial decision-making situations where transparency is crucial.

Finally, although this methodology was effectively implemented in the Indian stock market, its relevance to other markets is still unproven. The distinct features of the Indian market, such as its regulatory environment, investor behaviour, and macroeconomic factors, could impact the transferability of the findings to other global financial contexts. Future research should investigate if similar performance improvements can be realised in different national or regional markets and also consider incorporating hybrid or ensemble techniques that utilise sentiment analysis, real-time news, or attention-driven approaches to enhance the groundwork established by this study.

CHAPTER VI:

SUMMARY, IMPLICATIONS, AND RECOMMENDATIONS

6.1 Summary

This research's findings stem from a thorough assessment of stock market prediction models utilising various neural network architectures on both raw and stationary data. The study clearly demonstrates that converting stock market data into a stationary format significantly boosts model performance, which supports the central hypothesis. By employing a well-organised dataset from the Indian stock market, the research methodically evaluated the predictive abilities of four models – ANN, RNN, LSTM, and GRU – under uniform experimental conditions. This facilitated a robust comparison of different models regarding their performance with raw and stationary data.

GRU proved to be the most effective architecture, especially when trained on stationary data, highlighting its ability to manage the temporal complexities of financial time series. Following closely was LSTM, which displayed robust performance but had some instability when dealing with raw data. RNN models achieved moderate success, whereas ANN, which lacks temporal memory, consistently fell behind. Despite these variations, a common trend among all models was the significant boost in performance after the data was rendered stationary, underscoring the importance of preprocessing in financial forecasting tasks.

The research, based on statistical testing and thorough experimentation, validated the difficulties in forecasting stock market movements using solely historical numerical data. Events like the 2008 financial crisis consistently resulted in dips in model performance, highlighting the constraints of conventional machine learning models in the face of unexpected external shocks. However, the study demonstrates a clear enhancement in predictive accuracy from basic to advanced architectures, setting a dependable foundation for future research.

6.2 Implications

The research findings have significant implications for both theory and practice in financial forecasting and machine learning. They highlight the essential need to convert stock market time series data into a stationary format to improve model performance. This insight is particularly relevant for financial analysts, data scientists, and institutional investors who depend on predictive modelling for investment decisions. By establishing that stationarity enhances model accuracy, the study underscores the importance of thorough preprocessing when developing predictive systems for highly volatile datasets like those in the stock market.

From a methodological perspective, this study emphasises the importance of integrating statistical rigour with sophisticated deep-learning models. Among the reviewed models, the GRU stands out as the most effective, particularly when applied to stationary data. This demonstrates the strength of GRU in processing sequential financial information and paves the way for future advancements in neural network designs aimed at time series

forecasting. GRU's performance compared to ANN, RNN, and LSTM provides a useful framework for choosing suitable architectures in subsequent projects based on data characteristics and forecasting goals.

For professionals in the finance industry, the findings strongly support the integration of GRU-based models into their forecasting strategies, especially when paired with effective preprocessing techniques. Although traditional models and heuristics remain prevalent, machine learning models' capacity to analyse complex, multi-dimensional financial data offers a significant competitive advantage when utilised appropriately. This study advocates for a transition to more data-driven, flexible forecasting models that can accurately capture complex temporal dependencies, particularly in emerging markets such as India, where stock market behaviour often contrasts with that in established markets.

6.3 Recommendations for Future Research

This research's findings and observations suggest several promising avenues for future stock market prediction work using machine learning. While this study primarily evaluated ANN, RNN, LSTM, and GRU models on both raw and stationary financial data, there is considerable opportunity to build on these results and further refine the modelling approaches. One promising avenue is to investigate hybrid architectures that leverage the strengths of various models. For example, combining Convolutional Neural Networks (CNN) with GRUs may facilitate better feature extraction from intricate time series data before processing it through a temporal model. Additionally, the inclusion of attention

mechanisms could significantly improve the model's capacity to concentrate on relevant time steps, enhancing predictive accuracy and interpretability.

A promising direction for upcoming research involves leveraging transformer-based models in financial forecasting. These transformers, which have significantly changed sequence modelling in areas like natural language processing (NLP), feature a parallelisable architecture and an effective attention mechanism that could facilitate the efficient capture of long-range dependencies, surpassing traditional recurrent models. Investigating the use of transformers with stationary stock market data may yield fresh insights and exceed the performance of GRUs and LSTMs.

Additionally, this research focused solely on numerical data extracted from historical stock prices and macroeconomic indicators. By incorporating sentiment analysis from news articles, social media trends, and other qualitative sources, the model's robustness could be enhanced. Such features could play a crucial role in understanding market movements influenced by investor psychology and external geopolitical factors, where numerical models might not suffice.

Finally, applying this methodology to diverse geographical regions and a wider range of financial instruments could help generalise the outcomes. Although this research centred on the Indian stock market, broadening the approach to both developed and emerging markets globally would validate the findings across different market conditions. Such expansions would strengthen the main conclusions of this study and significantly contribute to the developing area of financial time series modelling with machine learning.

6.4 Conclusion

This research explored how data stationarity and model architecture influence stock market prediction using machine learning. A systematic series of experiments with ANN, RNN, LSTM, and GRU models was conducted on both raw and stationary data. Results indicated that converting stock market time series to a stationary format consistently enhanced model stability and predictive accuracy. Notably, GRU models outperformed the other architectures, particularly when trained on stationary data, highlighting their effectiveness in modelling complex temporal relationships efficiently.

The study emphasised the significance of preprocessing in financial forecasting. Without data transformation, models, regardless of their complexity, faced instability and inadequate generalisation. Achieving data stationarity clarified the time series' underlying structure, facilitating enhanced learning and improving the correlation between predicted and actual trends. This trend was consistently evident across R^2 values, prediction plots, and training curves.

At the same time, the study acknowledged its constraints. Predictive models based only on historical numerical data struggle to factor in sudden, unpredictable occurrences like financial crises or changes influenced by sentiment and external news. This shortcoming highlights the necessity for future research that goes beyond its current boundaries, integrating more data sources and exploring architectures like attention mechanisms and transformers, which might be more adept at capturing long-term dependencies and swift shifts in market dynamics.

This work has introduced a systematic method for evaluating model performance and the impact of data transformation in stock market forecasting. It provides a foundation for future research to develop more sophisticated models, hybrid methodologies, and larger datasets to enhance our comprehension of the challenges in financial prediction.

REFERENCES

Ajzen, I. (1991). The theory of planned behavior. Organizational Behavior and Human Decision Processes, 50(2), 179–211.

Bouktif, S., Fiaz, A., Ouni, A., & Serhani, M. A. (2020). Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. Energies, 13(7), 1–24.

Brockwell, P. J., & Davis, R. A. (1996). Introduction to Time Series and Forecasting. Springer.

Buslim, N. (2021). Comparing Bitcoin's prediction model using GRU, RNN, LSTM. Journal of Financial Innovation and Machine Learning, 5(2), 45–59.

Chatterjee, A., & Yadav, M. (2023). Stationarity transformation for enhancing LSTM and GRU model accuracy in financial forecasting. Journal of Computational Finance and Economics, 14(1), 21–38.

Chatterjee, D., & Yadav, A. (2023). Comparative Study of LSTM and GRU Models on Indian Stock Markets with Stationary Data. Indian Journal of Quantitative Finance, 11(4), 176–189.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation.

Davies, I. N., Okolie, S. O., Adoghe, A. U., Adeyemo, A. A., & Aghware, F. O. (2022). Stock prediction on the Nigerian Exchange using Type-2 fuzzy logic. Journal of Fuzzy Systems and Decision Sciences, 19(2), 110–125.

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. Journal of Finance, 25(2), 383–417.

Fishbein, M., & Ajzen, I. (1975). Belief, attitude, intention, and behavior: An introduction to theory and research. Addison-Wesley.

Foote, J. (2021). Applications of machine learning in financial markets: Past, present, and future. Journal of Financial Data Science, 3(4), 10–22.

Foote, J. (2021). Deep learning for time series forecasting: Learn how to use Python and deep learning to forecast time series data. Independently published.

Foote, J. (2021). Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery.

Giles, C. L., & Omlin, C. W. (1994). "Noisy Time Series Prediction using a Recurrent Neural Network and Grammatical Inference." Machine Learning, 17(2–3), 235–255.

Giles, C. L., & Omlin, C. W. (1994). Backpropagation and the dynamics of recurrent neural networks. Neural Computation, 6(1), 121–138.

Granovetter, M. (1985). Economic Action and Social Structure: The Problem of Embeddedness. American Journal of Sociology, 91(3), 481–510.

Gupta, R., & Srivastava, M. (2024). Machine Learning in Behavioral Finance: Identifying Predictive Inefficiencies. International Review of Financial Analytics, 18(2), 92–108.

Gupta, S., & Srivastava, A. (2024). The impact of global macroeconomic shocks on AI-based stock prediction: A comparative study. Journal of Financial Analytics, 11(2), 37–51.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780.

Hwang, H. (2023). The history of stock markets: From ancient Rome to Wall Street. Cambridge Financial Press.

Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: Principles and practice (2nd ed.). OTexts.

Iqbal, M., & Kumar, P. (2024). Data transformation and forecasting accuracy in financial machine learning models: A systematic review. Applied Economics and Computing Review, 18(1), 63–77.

Iqbal, T., & Kumar, V. (2024). Volatility Clustering and Non-Stationarity in Emerging Markets: Implications for ML Forecasting. Asia-Pacific Journal of Economics and Technology, 9(3), 112–127.

Khaldi, B., Bouktif, S., & Serhani, M. A. (2022). An empirical evaluation of deep learning architectures for stock market prediction using temporal features. Expert Systems with Applications, 200, 116965.

Lo, A. W., & MacKinlay, A. C. (1999). A non-random walk down Wall Street. Princeton University Press.

Malkiel, B. G. (1973). A random walk down Wall Street: The time-tested strategy for successful investing. W. W. Norton & Company.

Mitra, A., & Banerjee, S. (2023). Revisiting Market Efficiency in the Age of AI. Journal of Financial Modelling and Data Science, 12(1), 45–58.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques. Expert Systems with Applications, 42(1), 259–268.

Preis, T., Moat, H. S., & Stanley, H. E. (2013). Quantifying trading behavior in financial markets using Google Trends. Scientific Reports, 3, 1684.

Radecic, D. (2021). What is White Noise in Time Series Forecasting?. Towards Data Science.

Rathore, R., & Mehta, P. (2025). Stationarising Time Series for Deep Learning: Empirical Insights from Financial Data. Computational Economics Journal, 14(1), 33–50.

Rathore, S., & Mehta, P. (2025). Enhancing deep learning forecasting accuracy through data stationarisation: An empirical study of Indian indices. Journal of Financial Analytics and AI, 11(1), 45–61.

Rathore, S., & Mehta, R. (2025). Comparative performance of stationarity preprocessing in time series forecasting using deep learning. International Journal of Data Science and Forecasting, 6(1), 12–26.

Rogers, E. M. (2003). Diffusion of Innovations (5th ed.). Free Press.

Securities and Exchange Board of India (SEBI). (2023). Annual Report 2022–23. SEBI.

Sharma, N., & Dutta, A. (2024). Enhancing financial time series forecasting using GRUs: A comparative evaluation. Journal of Artificial Intelligence in Finance, 9(3), 55–68.

Sharma, N., & Dutta, A. (2024). GRU vs LSTM in Financial Time Series Prediction: A Case Study of High-Frequency Data. Neural Computation and Applications in Finance, 6(2), 65–81.

Shumway, R. H., & Stoffer, D. S. (2017). Time series analysis and its applications: With R examples (4th ed.). Springer.

Sidekerskienė, T., Damaševičius, R., & Maskeliūnas, R. (2024). “Internet Finance Non-stationary Time Series Prediction Algorithm Based on Deep Learning and Knowledge Map.” Information Technology and Control, 53(4), 1238–1252.

Telecom Regulatory Authority of India (TRAI). (2021). The Indian Telecom Services Performance Indicators: October–December 2020. TRAI Reports.

World Bank. (2022). Digital Development Overview: India. World Bank Reports.

Zhang, G., Eddy Patuwo, B., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting, 14(1), 35–62.

Zhang, W., & Zhou, D. (2004). Stock market prediction through multi-source fusion and noise reduction. Systems Engineering Theory and Practice, 24(8), 42–48.

APPENDIX A:

PYTHON PROGRAMS FOR FETCHING THE DATA

This section provides the Python programs used to fetch the data for this research, calling suitable APIs. These codes can be used to replicate the experiment for verification or extension of this research.

A.1 Fetching the NSE Index data

```
!pip install yfinance

import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt

# Define the NSE Index ticker symbol
target = "^NSEI"

# Fetch the historical data for NSE Index from 1-Jan-2000 to 31-Dec-2024
nse_index = yf.Ticker(target)
df_raw_NSE = nse_index.history(start="2000-01-01", end="2024-12-31")

# Display first few rows to confirm data retrieval
print(df_raw_NSE.head())

# Select interval points for labeling
interval = len(df_raw_NSE) // 10  # Selecting roughly 10 labels over the dat
aset
selected_dates = df_raw_NSE.iloc[::interval]  # Selecting every nth row

# Plot the NSE Index closing prices
plt.figure(figsize=(10, 5))
plt.plot(df_raw_NSE.index, df_raw_NSE['Close'], color="blue")

# Adding data labels at selected intervals
for date, value in zip(selected_dates.index, selected_dates['Close']):
    plt.text(date, value, f"{value:.0f}", fontsize=8, ha='center', va='botto
m', rotation=45)

# Set plot labels and title
plt.xlabel("Year")
plt.ylabel("NSE Index Value")
```

```
plt.title("NSE Index Closing Prices (2000-2024)")

# Remove the legend box
plt.grid(True)
plt.gca().spines[['top', 'right']].set_visible(False)

plt.show()
```

```
                                 Open         High          Low        Close  \
Date
2007-09-17 00:00:00+05:30  4518.450195  4549.049805  4482.850098  4494.649902
2007-09-18 00:00:00+05:30  4494.100098  4551.799805  4481.549805  4546.200195
2007-09-19 00:00:00+05:30  4550.250000  4739.000000  4550.250000  4732.350098
2007-09-20 00:00:00+05:30  4734.850098  4760.850098  4721.149902  4747.549805
2007-09-21 00:00:00+05:30  4752.950195  4855.700195  4733.700195  4837.549805

                           Volume  Dividends  Stock Splits
Date
2007-09-17 00:00:00+05:30       0        0.0           0.0
2007-09-18 00:00:00+05:30       0        0.0           0.0
2007-09-19 00:00:00+05:30       0        0.0           0.0
2007-09-20 00:00:00+05:30       0        0.0           0.0
2007-09-21 00:00:00+05:30       0        0.0           0.0
```

Figure A. 1: NSE Index between 2008 and 2024.

A.2 Fetching the Gold, Silver, and Crude Oil Prices data

```
import yfinance as yf
import pandas as pd
```

```
import matplotlib.pyplot as plt

# Define ticker symbols for Gold, Silver, and Crude Oil
tickers = {
    "Gold": "GC=F",         # Gold Futures
    "Silver": "SI=F",       # Silver Futures
    "Crude_Oil": "CL=F"     # Crude Oil Futures
}

# Date range
start_date = "2000-01-01"
end_date = "2024-12-31"

# Fetch historical data for Gold, Silver, and Crude Oil
df_raw_commodities = {}

for commodity, ticker in tickers.items():
    try:
        data = yf.download(ticker, start=start_date, end=end_date)
        data = data[['Close']].rename(columns={'Close': commodity})  # Keep
only closing prices
        df_raw_commodities[commodity] = data
    except Exception as e:
        print(f"An error occurred while fetching {commodity} data: {e}")

# Combine all datasets into a single DataFrame
df_raw_commodities = pd.concat(df_raw_commodities.values(), axis=1)

# Display the first few rows to confirm data retrieval
print(df_raw_commodities.head())

# Plot the Gold, Silver, and Crude Oil prices
plt.figure(figsize=(12, 6))

for commodity in df_raw_commodities.columns:
    plt.plot(df_raw_commodities.index, df_raw_commodities[commodity], label=
f"{commodity} Price", linewidth=1.5)

# Adding Data Labels at Selected Intervals
interval = len(df_raw_commodities) // 10  # Select approximately 10 points f
or labeling
selected_dates = df_raw_commodities.iloc[::interval]

for commodity in df_raw_commodities.columns:
    for date, value in zip(selected_dates.index, selected_dates[commodity]):
        plt.text(date, value, f"{value:.0f}", fontsize=8, ha='center', va='b
ottom', rotation=45)

# Set labels and title
plt.xlabel("Year")
plt.ylabel("Price (USD)")
plt.title("Gold, Silver, and Crude Oil Prices (2000-2024)")
```

```
plt.grid(True)

# Include the legend box
plt.legend(loc="upper left")

# Remove only the top and right spines
plt.gca().spines[['top', 'right']].set_visible(False)

plt.show()
```

```
YF.download() has changed argument auto_adjust default to True

[**********************100%***********************]  1 of 1 completed
[**********************100%***********************]  1 of 1 completed
[**********************100%***********************]  1 of 1 completed
```

Price	Gold	Silver	Crude_Oil
Ticker	GC=F	SI=F	CL=F
Date			
2000-08-23	NaN	NaN	32.049999
2000-08-24	NaN	NaN	31.629999
2000-08-25	NaN	NaN	32.049999
2000-08-28	NaN	NaN	32.869999
2000-08-29	NaN	NaN	32.720001

Figure A. 2: Gold, Silver, and Crude Oil Prices.

A.3 Fetching the INR-USD Exchange Rate data

```
!pip install pandas_datareader

import pandas as pd
import matplotlib.pyplot as plt
from pandas_datareader import data as pdr
import datetime

# Define the time range
start_date = datetime.datetime(2000, 1, 1)
end_date = datetime.datetime(2024, 12, 31)

# Fetch the exchange rate data from FRED
df_raw_xchng = pdr.get_data_fred('DEXINUS', start=start_date, end=end_date)

# Drop rows with missing values
df_raw_xchng.dropna(inplace=True)

# Plot the INR-USD Exchange Rate
plt.figure(figsize=(10, 6))
plt.plot(df_raw_xchng.index, df_raw_xchng['DEXINUS'], marker='o', linestyle=
'-', color='blue')

# Adding Data Labels at Selected Intervals
interval = len(df_raw_xchng) // 10  # Select approximately 10 points for lab
eling
selected_dates = df_raw_xchng.iloc[::interval]

for date, value in zip(selected_dates.index, selected_dates['DEXINUS']):
    plt.text(date, value, f"{value:.2f}", fontsize=8, ha='center', va='botto
m', rotation=45)

# Set labels and title
plt.xlabel("Year")
plt.ylabel("Exchange Rate (INR per USD)")
plt.title("INR-USD Exchange Rate (2000-2024)")
plt.grid(True)

# Remove only the top and right spines
plt.gca().spines[['top', 'right']].set_visible(False)

plt.show()
```

Figure A. 3: INR-USD Exchange Rates.

A.4 Fetching the Indian GDP data

```
!pip install world_bank_data

import world_bank_data as wb
import pandas as pd
import matplotlib.pyplot as plt

# Fetch Indian GDP data from the World Bank
df_raw_gdp = wb.get_series('NY.GDP.MKTP.CD', date='2000:2024', id_or_value='
id', simplify_index=True, country='IN')

# Convert GDP values to billions of US Dollars
df_raw_gdp = df_raw_gdp / 1e9  # Convert to billion USD

# Plot the GDP Data
plt.figure(figsize=(10, 5))
plt.plot(df_raw_gdp, color="green", marker="o", linestyle="-")

# Adding Data Labels
for year, gdp in df_raw_gdp.items():
    plt.text(year, gdp, f"{gdp:.0f}", fontsize=8, ha="center", va="bottom",
rotation=45)
```

```
# Set labels and title
plt.xlabel("Year")
plt.ylabel("GDP (in Billion US Dollars)")
plt.title("GDP of India (2000-2024)")
plt.xticks(rotation=45)
plt.grid(True)

# Remove the legend box and only keep the left & bottom spines
plt.gca().spines[['top', 'right']].set_visible(False)

plt.show()
```

Figure A. 4: Indian GDP between 2000 and 2023.

A.5 Putting all the data in a single data frame

```
import pandas as pd

# Ensure all datasets have a Date index and remove timezone information
df_raw_NSE.index = pd.to_datetime(df_raw_NSE.index).tz_localize(None)
df_raw_commodities.index = pd.to_datetime(df_raw_commodities.index).tz_local
ize(None)
df_raw_xchng.index = pd.to_datetime(df_raw_xchng.index).tz_localize(None)
df_raw_gdp.index = pd.to_datetime(df_raw_gdp.index).tz_localize(None)

# Step 1: Align GDP Data to Daily Format
```

```
df_gdp_daily = df_raw_gdp.copy().to_frame()  # Convert Series to DataFrame
df_gdp_daily = df_gdp_daily.reindex(pd.date_range(start="2000-01-01", end="2
024-12-31", freq="D"))
df_gdp_daily.ffill(inplace=True)  # Forward fill GDP values for all days in
each year
df_gdp_daily.columns = ['Indian_GDP']  # Rename column

# Flatten multi-index column names in df_raw_commodities (if needed)
if isinstance(df_raw_commodities.columns, pd.MultiIndex):
    df_raw_commodities.columns = [col[0] for col in df_raw_commodities.colum
ns]

# Step 2: Merge All DataFrames
df_raw = df_raw_NSE[['Close']].rename(columns={'Close': 'NSE_Index'})  # Use
only NSE closing price
df_raw = df_raw.join(df_raw_commodities, how='inner')  # Join commodity pric
es
df_raw = df_raw.join(df_raw_xchng.rename(columns={'DEXINUS': 'INR_USD'}), ho
w='inner')  # Join Exchange Rate
df_raw = df_raw.join(df_gdp_daily, how='inner')  # Join GDP data

# Display first few rows of final dataset
print(df_raw.head(10))

# Save the final dataset as a CSV (Optional)
df_raw.to_csv("df_raw.csv")
```

	NSE_Index	Gold	Silver	Crude_Oil	INR_USD	Indian_GDP
2007-09-17	4494.649902	715.799988	12.739	80.570000	40.52	1216.736439
2007-09-18	4546.200195	715.799988	12.767	81.510002	40.45	1216.736439
2007-09-19	4732.350098	722.000000	12.956	81.930000	39.81	1216.736439
2007-09-20	4747.549805	732.400024	13.321	83.320000	39.87	1216.736439
2007-09-21	4837.549805	731.400024	13.474	81.620003	39.84	1216.736439
2007-09-24	4932.200195	732.099976	13.497	80.949997	39.50	1216.736439
2007-09-25	4938.850098	731.599976	13.482	79.529999	39.55	1216.736439
2007-09-26	4940.500000	728.299988	13.416	80.300003	39.50	1216.736439
2007-09-27	5000.549805	732.700012	13.517	82.879997	39.65	1216.736439
2007-09-28	5021.350098	742.799988	13.794	81.660004	39.75	1216.736439

APPENDIX B:

PYTHON PROGRAMS FOR MAKING THE DATA STATIONARY

This section provides the Python programs used to make the data stationary for this research by performing suitable tests and data transformations. These codes can be used to replicate the experiment for verification or extension of this research.

B.1 Generic Functions to Test for Stationarity

Below are the functions used to check for stationarity of a time series.

```
import pandas as pd
import numpy as np
import warnings
from statsmodels.tsa.stattools import adfuller, kpss

# Function to apply ADF test
def apply_adf_test(series):
    """
    Apply Augmented Dickey-Fuller test to check stationarity.

    Parameters:
        series (pd.Series): Time series data.

    Returns:
        dict: ADF test results including test statistic, p-value, and conclu
sion.
    """
    result = adfuller(series, autolag='AIC')

    print("\n=== Augmented Dickey-Fuller Test (ADF) ===")
    print(f"Test Statistic       : {result[0]:.4f}")
    print(f"p-value              : {result[1]:.4f}")
    print(f"Lags Used            : {result[2]}")
    print(f"Number of Observations: {result[3]}")

    print("Critical Values:")
    for key, value in result[4].items():
        print(f"   {key}: {value:.4f}")
```

```
    conclusion = "Stationary" if result[1] < 0.05 else "Non-Stationary"
    print(f"Conclusion: The time series is {conclusion}.")

    return {"Test Statistic": result[0], "p-value": result[1], "Critical Val
ues": result[4], "Stationary?": conclusion}

# Function to apply KPSS test
def apply_kpss_test(series):
    """
    Apply KPSS test to check stationarity.

    Parameters:
        series (pd.Series): Time series data.

    Returns:
        dict: KPSS test results including test statistic, p-value, and concl
usion.
    """
    with warnings.catch_warnings():
        warnings.simplefilter("ignore")  # Suppress warnings
        result, p_value, lags, critical_values = kpss(series, regression='c'
, nlags="auto")

    print("\n=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===")
    print(f"Test Statistic       : {result:.4f}")
    print(f"p-value             : {p_value:.4f}")
    print(f"Number of Lags Used : {lags}")

    print("Critical Values:")
    for key, value in critical_values.items():
        print(f"   {key}: {value:.4f}")

    conclusion = "Stationary" if p_value > 0.05 else "Non-Stationary"
    print(f"Conclusion: The time series is {conclusion}.")

    return {"Test Statistic": result, "p-value": p_value, "Critical Values":
critical_values, "Stationary?": conclusion}

# Function to determine stationarity conclusion
def check_stationarity(series):
    """
    Run both ADF and KPSS tests and determine stationarity.

    Parameters:
        series (pd.Series): Time series data.

    Returns:
        dict: Summary of stationarity conclusion.
    """
    adf_result = apply_adf_test(series)
    kpss_result = apply_kpss_test(series)
```

```
    print("\n=== Final Conclusion on Stationarity ===")

    if adf_result['Stationary?'] == "Stationary" and kpss_result['Stationary
?'] == "Stationary":
        conclusion = "The time series is stationary."
    elif adf_result['Stationary?'] == "Non-Stationary" and kpss_result['Stat
ionary?'] == "Non-Stationary":
        conclusion = "The time series is non-stationary."
    elif adf_result['Stationary?'] == "Stationary" and kpss_result['Stationa
ry?'] == "Non-Stationary":
        conclusion = "The time series is trend-stationary."
    else:
        conclusion = "The time series is difference-stationary (requires dif
ferencing)."

    print(conclusion)

    return {"ADF Test": adf_result, "KPSS Test": kpss_result, "Conclusion":
conclusion}
```

B.2 Making the NSE Index time series stationary

The NSE Index time series was checked for stationarity. The raw data was not stationary, but it became stationary after applying one differencing transformation.

The code below loads the data.

```
import gdown

# Download df_raw.csv
url = 'https://drive.google.com/uc?id=1tIJctQuQL-LGRdHFXnihgvlVEaGGncJl'
gdown.download(url, './df_raw.csv', quiet = False)

import pandas as pd

df_raw = pd.read_csv('./df_raw.csv')

# Rename the first column to "Obs_Date"
df_raw.rename(columns={df_raw.columns[0]: "Obs_Date"}, inplace=True)
```

The code below performs the stationarity tests and makes the time series stationary.

```
# Ensure 'Obs_Date' is the index (if not already)
df_raw.index = pd.to_datetime(df_raw.index)

# Extract NSE Index series and drop missing values
nse_series = df_raw["NSE_Index"].dropna()

# Apply stationarity tests
stationarity_results = check_stationarity(nse_series)

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : 1.0550
p-value             : 0.9948
Lags Used           : 17
Number of Observations: 4063
Critical Values:
  1%: -3.4320
  5%: -2.8623
  10%: -2.5671
Conclusion: The time series is Non-Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 8.8886
p-value             : 0.0100
Number of Lags Used : 39
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Non-Stationary.

=== Final Conclusion on Stationarity ===
The time series is non-stationary.



# Apply first differencing
nse_diff1 = nse_series.diff().dropna()

print("\n=== Testing Stationarity After First Differencing ===")
stationarity_results_diff1 = check_stationarity(nse_diff1)


=== Testing Stationarity After First Differencing ===

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -14.1735
p-value             : 0.0000
```

```
Lags Used           : 16
Number of Observations: 4063
Critical Values:
  1%: -3.4320
  5%: -2.8623
  10%: -2.5671
Conclusion: The time series is Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.3675
p-value             : 0.0912
Number of Lags Used : 6
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Stationary.

=== Final Conclusion on Stationarity ===
The time series is stationary.
```

The NSE Index time series was made stationary after applying one differencing.

B.3 Making the Gold Prices time series stationary

The Gold Price time series was checked for stationarity. The raw data was not stationary, but it became stationary after applying one differencing transformation.

```
# Extract Gold Prices series and drop missing values
gold_series = df_raw["Gold"].dropna()

# Apply stationarity tests
stationarity_results = check_stationarity(gold_series)

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -0.1095
p-value             : 0.9485
Lags Used           : 6
Number of Observations: 4073
Critical Values:
  1%: -3.4320
```

```
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Non-Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 5.8338
p-value             : 0.0100
Number of Lags Used : 39
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Non-Stationary.

=== Final Conclusion on Stationarity ===
The time series is non-stationary.
```

```python
# Apply first differencing
gold_diff1 = gold_series.diff().dropna()

print("\n=== Testing Stationarity After First Differencing ===")
stationarity_results_diff1 = check_stationarity(gold_diff1)
```

```
=== Testing Stationarity After First Differencing ===

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -28.4558
p-value             : 0.0000
Lags Used           : 5
Number of Observations: 4073
Critical Values:
  1%: -3.4320
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.1871
p-value             : 0.1000
Number of Lags Used : 22
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Stationary.

=== Final Conclusion on Stationarity ===
The time series is stationary.
```

The Gold Prices time series was made stationary after applying one differencing transformation.

B.4 Making the Silver Prices time series stationary

The Silver Price time series was checked for stationarity. The raw data was not stationary, but it became stationary after applying one differencing transformation.

```
# Extract Silver Prices series and drop missing values
silver_series = df_raw["Silver"].dropna()

# Apply stationarity tests
stationarity_results = check_stationarity(silver_series)

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -2.4576
p-value             : 0.1261
Lags Used           : 0
Number of Observations: 4079
Critical Values:
  1%: -3.4320
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Non-Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.7833
p-value             : 0.0100
Number of Lags Used : 39
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Non-Stationary.

=== Final Conclusion on Stationarity ===
The time series is non-stationary.

# Apply first differencing
silver_diff1 = silver_series.diff().dropna()
```

```
print("\n=== Testing Stationarity After First Differencing ===")
stationarity_results_diff1 = check_stationarity(silver_diff1)


=== Testing Stationarity After First Differencing ===

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -15.3248
p-value             : 0.0000
Lags Used           : 20
Number of Observations: 4058
Critical Values:
  1%: -3.4320
  5%: -2.8623
  10%: -2.5671
Conclusion: The time series is Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.0574
p-value             : 0.1000
Number of Lags Used : 2
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Stationary.

=== Final Conclusion on Stationarity ===
The time series is stationary.
```

B.5 Making the Crude Oil Prices time series stationary

The Crude Oil Price time series was checked for stationarity. The raw data was not stationary, but it became stationary after applying one differencing transformation.

```
# Extract Crude Oil Prices series and drop missing values
oil_series = df_raw["Crude_Oil"].dropna()

# Apply stationarity tests
stationarity_results = check_stationarity(oil_series)

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -2.8008
p-value             : 0.0582
```

```
Lags Used           : 20
Number of Observations: 4060
Critical Values:
  1%: -3.4320
  5%: -2.8623
  10%: -2.5671
Conclusion: The time series is Non-Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 1.8496
p-value             : 0.0100
Number of Lags Used : 39
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Non-Stationary.

=== Final Conclusion on Stationarity ===
The time series is non-stationary.
```

```python
# Apply first differencing
oil_diff1 = oil_series.diff().dropna()

print("\n=== Testing Stationarity After First Differencing ===")
stationarity_results_diff1 = check_stationarity(oil_diff1)
```

```
=== Testing Stationarity After First Differencing ===

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -12.7626
p-value             : 0.0000
Lags Used           : 19
Number of Observations: 4060
Critical Values:
  1%: -3.4320
  5%: -2.8623
  10%: -2.5671
Conclusion: The time series is Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.0378
p-value             : 0.1000
Number of Lags Used : 25
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Stationary.
```

```
=== Final Conclusion on Stationarity ===
The time series is stationary.
```

B.6 Making the INR-USD Exchange Rate time series stationary

The INR-USD Exchange Rate time series was checked for stationarity. The raw data was not stationary, but it became stationary after applying one differencing transformation.

```
# Extract Exchange Rate series and drop missing values
exrate_series = df_raw["INR_USD"].dropna()

# Apply stationarity tests
stationarity_results = check_stationarity(exrate_series)

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -0.8544
p-value             : 0.8026
Lags Used           : 6
Number of Observations: 4074
Critical Values:
  1%: -3.4320
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Non-Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 9.8482
p-value             : 0.0100
Number of Lags Used : 39
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Non-Stationary.

=== Final Conclusion on Stationarity ===
The time series is non-stationary.

# Apply first differencing
exrate_diff1 = exrate_series.diff().dropna()
```

```
print("\n=== Testing Stationarity After First Differencing ===")
stationarity_results_diff1 = check_stationarity(exrate_diff1)


=== Testing Stationarity After First Differencing ===

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -25.6993
p-value             : 0.0000
Lags Used           : 5
Number of Observations: 4074
Critical Values:
  1%: -3.4320
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.0269
p-value             : 0.1000
Number of Lags Used : 3
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Stationary.

=== Final Conclusion on Stationarity ===
The time series is stationary.
```

B.7 Making the Indian GDP time series stationary

The Indian GDP time series was checked for stationarity. The raw data was not stationary, but it became stationary after applying one differencing transformation.

```
# Extract GDP series and drop missing values
gdp_series = df_raw["Indian_GDP"].dropna()

# Apply stationarity tests
stationarity_results = check_stationarity(gdp_series)

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -0.5071
p-value             : 0.8906
```

```
Lags Used           : 0
Number of Observations: 4080
Critical Values:
  1%: -3.4320
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Non-Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 9.9811
p-value             : 0.0100
Number of Lags Used : 39
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Non-Stationary.

=== Final Conclusion on Stationarity ===
The time series is non-stationary.
```

```python
# Apply first differencing
gdp_diff1 = gdp_series.diff().dropna()

print("\n=== Testing Stationarity After First Differencing ===")
stationarity_results_diff1 = check_stationarity(gdp_diff1)
```

```
=== Testing Stationarity After First Differencing ===

=== Augmented Dickey-Fuller Test (ADF) ===
Test Statistic       : -63.9710
p-value             : 0.0000
Lags Used           : 0
Number of Observations: 4079
Critical Values:
  1%: -3.4320
  5%: -2.8622
  10%: -2.5671
Conclusion: The time series is Stationary.

=== Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS) ===
Test Statistic       : 0.0246
p-value             : 0.1000
Number of Lags Used : 3
Critical Values:
  10%: 0.3470
  5%: 0.4630
  2.5%: 0.5740
  1%: 0.7390
Conclusion: The time series is Stationary.
```

```
=== Final Conclusion on Stationarity ===
The time series is stationary.
```

B.8 Making a Data Frame of Stationary Time Series

All the time series could be made stationary with one differencing. So, all the stationary time series were assimilated into a single data frame to apply machine learning algorithms.

```
import pandas as pd
import numpy as np

# Ensure 'Obs_Date' is the index (if not already)
df_raw.index = pd.to_datetime(df_raw.index)

# Count missing values BEFORE differencing
print("Missing values BEFORE differencing:")
print(df_raw.isna().sum())

# Apply first differencing to all series **before** dropping NaNs
df_st = df_raw.copy()

# Apply differencing to all numerical columns (excluding 'Obs_Date')
for col in df_raw.columns:
    if col != "Obs_Date":  # Skip the date column
        df_st[col] = df_raw[col].diff()  # Apply first differencing

# Now, drop NaN values **only once**, preserving alignment
df_st.dropna(inplace=True)

# Reset index and rename first column to "Junk"
df_st.reset_index(inplace=True)
df_st.rename(columns={'index': 'Junk'}, inplace=True)

# Drop the "Junk" column
df_st.drop(['Junk'], axis=1, inplace=True)

# Display the new DataFrame shape
print("\nNew DataFrame Shape after Differencing:", df_st.shape)

# Display the first 10 rows of the new DataFrame
print("\n")
```

```
print(df_st.head(10))

# Save as CSV (Optional)
df_st.to_csv("df_st.csv", index=False)

Missing values BEFORE differencing:
Obs_Date      0
NSE_Index     0
Gold          1
Silver        1
Crude_Oil     0
INR_USD       0
Indian_GDP    0
dtype: int64

New DataFrame Shape after Differencing: (4078, 7)


     Obs_Date   NSE_Index       Gold    Silver  Crude_Oil  INR_USD  Indian_GDP
0  2007-09-18   51.550293   0.000000  0.028000   0.940002    -0.07         0.0
1  2007-09-19  186.149902   6.200012  0.189000   0.419998    -0.64         0.0
2  2007-09-20   15.199707  10.400024  0.365000   1.389999     0.06         0.0
3  2007-09-21   90.000000  -1.000000  0.153000  -1.699997    -0.03         0.0
4  2007-09-24   94.650391   0.699951  0.023000  -0.670006    -0.34         0.0
5  2007-09-25    6.649902  -0.500000 -0.014999  -1.419998     0.05         0.0
6  2007-09-26    1.649902  -3.299988 -0.066000   0.770004    -0.05         0.0
7  2007-09-27   60.049805   4.400024  0.101000   2.579994     0.15         0.0
8  2007-09-28   20.800293  10.099976  0.276999  -1.219994     0.10         0.0
9  2007-10-01   47.600098   4.400024 -0.065000  -1.420006    -0.05         0.0
```