STOCK MARKET FORECASTING USING DEEP LEARNING: COMPREHENSIVE ANALYSIS OF LSTM, GRU AND HYBRID MODEL

Haris Roy Pulinilkunnathil

261608

Submitted for the degree of MSc. Data Science

DR Konstantinos Koumatos

University of Sussex

Date : 08 September 2023

Declaration

I am submitting this document as part of the MSc Data Science degree requirements at the University of Sussex. I, Haris Roy Pulinilkunnathil, affirm that the work in this dissertation is mainly my own. Where I have used or referred to others' work, I have appropriately acknowledged it. The primary research and writing are personally conducted and crafted by me.

Haris Roy Pulinilkunnathil, 08 September 2023

Acknowledgements

I am deeply grateful to Dr. Konstantinos Koumatos for his steadfast support and expert guidance throughout this dissertation. Dr. Koumatos, your insightful feedback, invaluable resources, and commitment to my development have been instrumental in shaping this work. Your mentorship has been a beacon, enlightening my academic journey.

Special mention must be made of my mother, whose unyielding encouragement, wisdom, and faith have been my guiding light. Her unwavering support and belief in my abilities have been the cornerstone of my perseverance in this master's degree. Alongside her, the rest of my family has provided a consistent foundation of encouragement and trust, allowing me to confidently move forward.

I must also acknowledge my friends' enduring support and understanding throughout this academic journey. Their resilience and steadfastness have provided a firm foundation during the rigorous phases of this degree.

Finally, I extend my gratitude to the cohort of Data Science postgraduate students with whom I have had the privilege to study. The collective wisdom, rigorous debates, and shared knowledge have significantly enriched this academic experience.

Contents

LIST OF FIGURES

LIST OF TABLES

Abstract

This dissertation evaluates deep learning's potential in forecasting the Nifty 50 stock index, a key indicator of India's equity markets. Three models—LSTM, GRU, and a Hybrid of both—were analysed. While the GRU showcased superior precision in traditional metrics, visual analysis revealed the LSTM's predictions as most aligned with actual stock values, underscoring the multifaceted nature of stock forecasting. The study spotlights the promise of deep learning in stock market predictions, emphasizing the need for comprehensive evaluation approaches and integrating diverse data streams in future endeavours.

Chapter 1: Introduction

The global financial landscape, a vast and interconnected network, is deeply influenced by the intricacies and dynamics of stock markets. These stock markets, beyond their immediate function, are pivotal economic platforms. Their primary role involves serving as bustling arenas where financial instruments—stocks, bonds, and derivatives—are traded. However, their significance extends much further. They act as reflective mirrors, capturing and displaying the fiscal health of corporations, from burgeoning startups to established conglomerates. Moreover, they offer insights into the economic pulse of nations, from emerging economies to developed powerhouses.

The art and science of forecasting stock market trends have long been considered a golden chalice in finance. The potential rewards of accurate prediction are immense, given that it can dramatically influence investment decisions, financial strategies, and, ultimately, monetary outcomes. On the flip side, misjudgements can lead to significant financial downturns. Historically, the quest to fathom the volatile and inherently unpredictable nature of stock prices has led to the development of various tools and techniques. Mathematical models grounded in rigorous formulas and statistical models rooted in data-driven analyses have been the traditional stalwarts in this endeavour.

However, the landscape of stock market forecasting is witnessing a paradigm shift. The recent technological strides, particularly in deep learning, signal the dawn of a new epoch. Deep learning, characterised by its sophisticated neural architectures capable of handling and processing substantial volumes of data, offers a nuanced approach to pattern recognition. This approach allows it to identify intricate correlations and dependencies that may remain obscure to conventional analytical models. As such, deep learning is increasingly being recognised as an indispensable instrument in contemporary computational research and applications. Its promise is not just incremental improvement but a transformative change that can potentially significantly eclipse traditional forecasting methodologies' capabilities and accuracies significantly.

1.1 Research Problem Definition

Within the vast expanse of the global financial landscape, specific indices like the **Nifty 50** stand out as significant barometers of economic health. The Nifty 50, representing 50 of the largest and most actively traded stocks on the National Stock Exchange of India, serves as a crucial benchmark reflecting the pulse of the Indian equity market. Accurately predicting the movements of such a pivotal index is a nuanced endeavour that often eludes even the most sophisticated traditional forecasting methods. This dissertation seeks to ascertain the effectiveness of modern deep learning models—LSTM, GRU, and a Hybrid Model combining LSTM and GRU—in addressing this intricate challenge. By honing in on the Nifty 50, this research aims to elucidate the comparative strengths and potential pitfalls of each model and provide insights that might be applicable to other major stock indices worldwide.

1.2 Research Objectives

1. Foundational Understanding and Model Implementation:

- To comprehensively understand and elucidate the theoretical foundation of LSTM, GRU, and the Hybrid Model, which amalgamates features of both LSTM and GRU.
- To implement these models to forecast stock market trends, specifically focusing on the Nifty 50 index.

2. Exploratory Data Analysis and Data Preparation:

- To conduct a thorough exploratory data analysis (EDA) on the Nifty 50 dataset, encompassing visualisations, statistical summaries, and time-series decomposition.
- To preprocess the data, ensuring it is aptly scaled and structured for time-series forecasting with deep learning models.

3. Empirical Analysis, Evaluation and Comparative Study

- To train, validate, and test the LSTM, GRU, and Hybrid Models on the dataset, emphasising their predictive accuracy and computational efficiency.
- To optimise these models through hyperparameter tuning, refining their parameters to enhance their forecasting capabilities.
- To methodically assess the models' performance using evaluative metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

- To contrast the models' forecasted values with actual stock values, striving to identify the model that most accurately mirrors real-world trends.

1.3 Overview of Deep Learning Models Used for Time Series Forecasting

1. **Long Short-Term Memory (LSTM)**: A sophisticated variant of traditional RNNs, LSTMs are adept at processing long-term sequential data, making them a favoured choice for time series forecasting.
2. **Gated Recurrent Unit (GRU):** A derivative of LSTMs, GRUs streamline the architecture by combining the forget and input gates. This often translates to quicker model training and, in some cases, enhanced performance.
3. **Hybrid Model:** A blend of LSTM and GRU characteristics, the Hybrid Model is designed to harness the distinct features of both models.

1.4 Significance of the Study

Deep learning is changing how we tackle problems in many fields, making complex tasks more manageable. One area it is making a difference in is predicting stock market movements. Traditional methods often need to catch up, but deep learning lets us see patterns more clearly. This study looks closely at three deep learning models (LSTM, GRU, and the Hybrid Model) to see how well they predict the movements of the Nifty 50, a key measure of India's stock market. What we learn here could help improve how we forecast stocks in the future, combining the best of finance and modern technology.

Chapter 2: Background

2.1 Historical Overview of Stock Market Forecasting Techniques

The history of stock market forecasting may be traced back to the earliest stock exchanges when traders used simple chart patterns to predict future price trends. Known as chartists or technical analysts, these pioneers operated under the assumption that a stock's price contained all relevant information. To foresee future price changes, they carefully examined price charts. Tools like line charts, bar charts, and later candlestick charts became essential for these traders. Fundamental analysts have departed from technical analysts' exclusive emphasis on price and volume data in favour of a more comprehensive investigation into the inherent worth of stock. Their methodology included thoroughly evaluating elements like a company's financial stability, the management's skill, market state, and general economic indicators. Their objective, which was the cornerstone of many long-term investment strategies, was to determine whether a company was undervalued or overvalued.

Quantitative models first appeared due to the technological developments of the 1960s and 1970s, which were characterised by the widespread use of computers. These advanced models, built on mathematical equations and algorithms, were designed to forecast stock values. Regression and time series analyses are well-established approaches that finally made way for the revolutionary Black-Scholes model for option pricing. The computing power of computers allowed traders and analysts to sort through and analyse vast amounts of data quickly.

Parallel to this, the 20th century saw the emergence of revolutionary theories that would forever change how modern finance would be structured. The ground-breaking Modern Portfolio Theory (MPT) by Harry Markowitz introduced the idea of optimising portfolio returns about a predetermined amount of risk. The Efficient Market Hypothesis (EMH), which asserts that stock prices always reflect all available information, was developed concurrently. This claim claimed that continuously exceeding the market was pointless. Despite the EMH's fair share of problems, it sparked passionate discussions and prompted in-depth research into the nuances of market anomalies and forecasting practices.

As the 20th century ended and the 21st century ushered in, the world saw a parabolic rise in processing power and an overabundance of data. This convergence sped up the development of sophisticated algorithmic trading tactics. Thanks to these algorithms, computers could start

trading at breakneck speeds while being anchored to predetermined parameters. These cutting-edge algorithms were skilled at searching large datasets, identifying trends, and quickly executing trades.

Traditional time series forecasting methods have long dominated the field of stock market forecasting. These econometric and statistically based approaches have been used to analyse and forecast future values based on previously recorded data points. This method has relied heavily on methods like seasonal-trend decomposition using LOESS (STL) of time series, exponential smoothing, and auto-regressive integrated moving average (ARIMA). They put a strong foundation for financial predictions in place by concentrating on identifying linear correlations, trends, and seasonality in the data (Rouf *et al.*, 2021).

However, as the complexity of the financial markets increased, and a flood of data became accessible, a paradigm shift was brought about by the development of deep learning. To analyse different types of data, deep learning, a subset of machine learning, uses neural networks with numerous layers (thus the name "deep"). Its strength is its capacity to recognise and understand non-linear relationships, making it especially appropriate for frequently displaying non-linear dynamics for financial datasets. Deep learning has applications in various areas of finance, including algorithmic trading, credit scoring, fraud detection, and portfolio management. Its introduction marked a transformative phase, bridging the gap between vast data processing needs and the intricacies of financial markets. With the ability to analyse sequential data, deep learning models like LSTM and GRU have further solidified their position as the industry leaders in financial time series forecasting, providing a more sophisticated and nuanced way compared to more conventional techniques.

2.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are an advanced form of Recurrent Neural Networks (RNNs). LSTMs were developed by Sepp Hochreiter and Jürgen Schmidhuber in 1997 to address limitations with conventional RNNs, precisely the vanishing gradient problem that restricted their capacity to learn from lengthy sequences. The network may choose which information to keep, discard, or pass on thanks to the unique design of LSTMs, which is characterised by complex cell structures with many gates. This architecture successfully captures long-term dependencies in data. Since their introduction, LSTMs have emerged as a

critical component of deep learning for problems involving sequential data, finding use in fields like speech recognition, time series forecasting, and natural language processing.

A vanilla LSTM (basic LSTM) unit comprises a memory cell and three primary gates: input, output, and forget. Gers and colleagues later added the forget gate to allow the network to delete its memory, even though it was not part of the original LSTM architecture (Van Houdt, Mosquera and Nápoles, 2020). The connection between this memory cell and the gates regulates the information flow within the cell, which allows it to store information over a range of time intervals. This fundamental structure is the most prevalent form.

The basic building blocks of an LSTM are interconnected memory sub-networks. These blocks are intended to maintain their state over time while controlling the data flow through particular gates. A typical LSTM block incorporates these gates with input signals, output signals, activation functions, and even peephole connections, as shown in Fig. 1(Van Houdt, Mosquera and Nápoles, 2020). Additionally, the output of the block is looped back, connecting repeatedly to both the input and gates of the block.

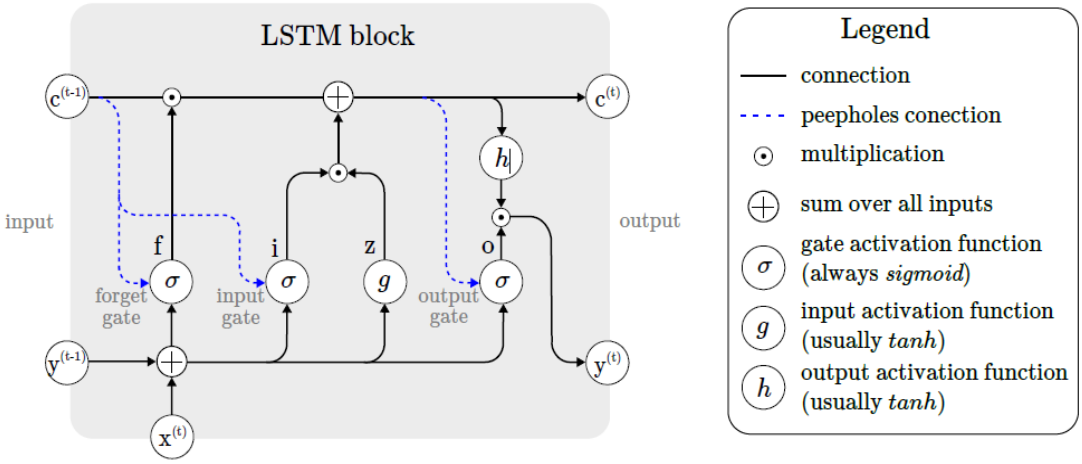

Figure 1: Architecture of a standard vanilla LSTM block

Consider a network with M inputs, N processing blocks, to understand better how the LSTM model operates. Within this recurrent neural network, the forward pass is as follows:

Block Input: The input to a block is updated by merging the present input $x^{(t)}$ with the output from the prior LSTM unit $y^{(t-1)}$. Mathematically, it is given by:

$$z^{(t)} = g(W_z x^{(t)} + R_z y^{(t-1)} + b_z) \quad (1)$$

In the above equation (1) W_z and R_z are the weight matrices associated with $x^{(t)}$ and $y^{(t-1)}$ and b_z is the bias vector. The bias vector plays a crucial role in neural networks. It is akin to an intercept in a linear equation, providing the model with additional freedom. Adding this bias vector allows the neuron's output to be shifted and adjusted, enabling the model to fit the data more effectively.

Input Gate: The input gate is refreshed by amalgamating the current input $x^{(t)}$, the prior output $y^{(t-1)}$, and the previous cell state $c^{(t-1)}$. This is represented as:

$$i^{(t)} = \sigma(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i) \tag{2}$$

Here (2), $\odot$ denotes point-wise multiplication of vectors, and W_i, R_i and p_i are the weights associated with $x^{(t)}, y^{(t-1)}$ and $c^{(t-1)}$ respectively. The term b_i stands for the bias vector associated with this gate.

Forget Gate: The LSTM evaluates which data from the previous cell states $c^{(t-1)}$, should be discarded. This decision is based on the current input, previous outputs, previous cell states, and other factors: This decision is based on the current input, previous outputs, previous cell states, and other factors:

$$f^{(t)} = \sigma(W_f x^{(t)} + R_f y^{(t-1)} + p_f \odot c^{(t-1)} + b_f) \tag{3}$$

In (3) W_f, R_f and p_f represent the weights corresponding to $x^{(t)}, y^{(t-1)}$ and $c^{(t-1)}$respectively, while b_f is the associated bias vector.

Cell: The block input $z^{(t)}$, input gate values $i^{(t)}$, and forget gate values $f^{(t)}$ are combined with the previous cell state to update the cell state.

$$c^{(t)} = z^{(t)} \odot i^{(t)} + c^{(t-1)} \odot f^{(t)} \tag{4}$$

Output Gate: The output gate can be calculated by this step, which combines the input $x^{(t)}$, the output of that LSTM unit $y^{(t-1)}$ and the cell value $c^{(t-1)}$ in the last iteration.

$$o^{(t)} = \sigma(W_o x^{(t)} + R_o y^{(t-1)} + p_o \odot c^{(t)} + b_o) \tag{5}$$

W_o, R_o and p_o are the weights corresponding to $x^{(t)}, y^{(t-1)}$ and $c^{(t-1)}$ respectively, with b_o being the related bias vector.

Block Output: Finally, the block's output is calculated by combining the current cell value $c^{(t)}$ with the current output gate's value as follows:

$$y^{(t)} = g(c^{(t)}) \odot o^{(t)} \tag{6}$$

In the above-mentioned procedures σ, g and h denote point-wise non-linear activation functions. As a gate activation function, the logistic sigmoid $\sigma(x) = \frac{1}{1+e^{1-x}}$ is used, but the block input and output activation function are frequently implemented using the hyperbolic tangent $g(x) = h(x) = \tanh(x)$.

2.3 Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) was introduced by Chung (2014) as a modification to the LSTM architecture. The aim was to simplify the design of the LSTM while preserving its capacity to capture and remember long sequences(Fig 2). While GRUs and LSTMs share similarities, the GRU adopts a more streamlined approach. Instead of separate forget and input gates like the LSTM, the GRU consolidates them into a singular update gate. Additionally, the GRU fuses the distinct hidden and cell states found in LSTMs into one unified state. As a result, GRUs possess only half the gates of LSTMs, leading to their reputation as a more compact and efficient LSTM alternative.

The accompanying figure 2 shows the layouts of three different neural networks (A) a basic Recurrent Neural Network (RNN) cell, (B) a Long-short Term Memory(LSTM) cell, and (C) presents the design of the Gated Recurrent Unit (GRU) cell.

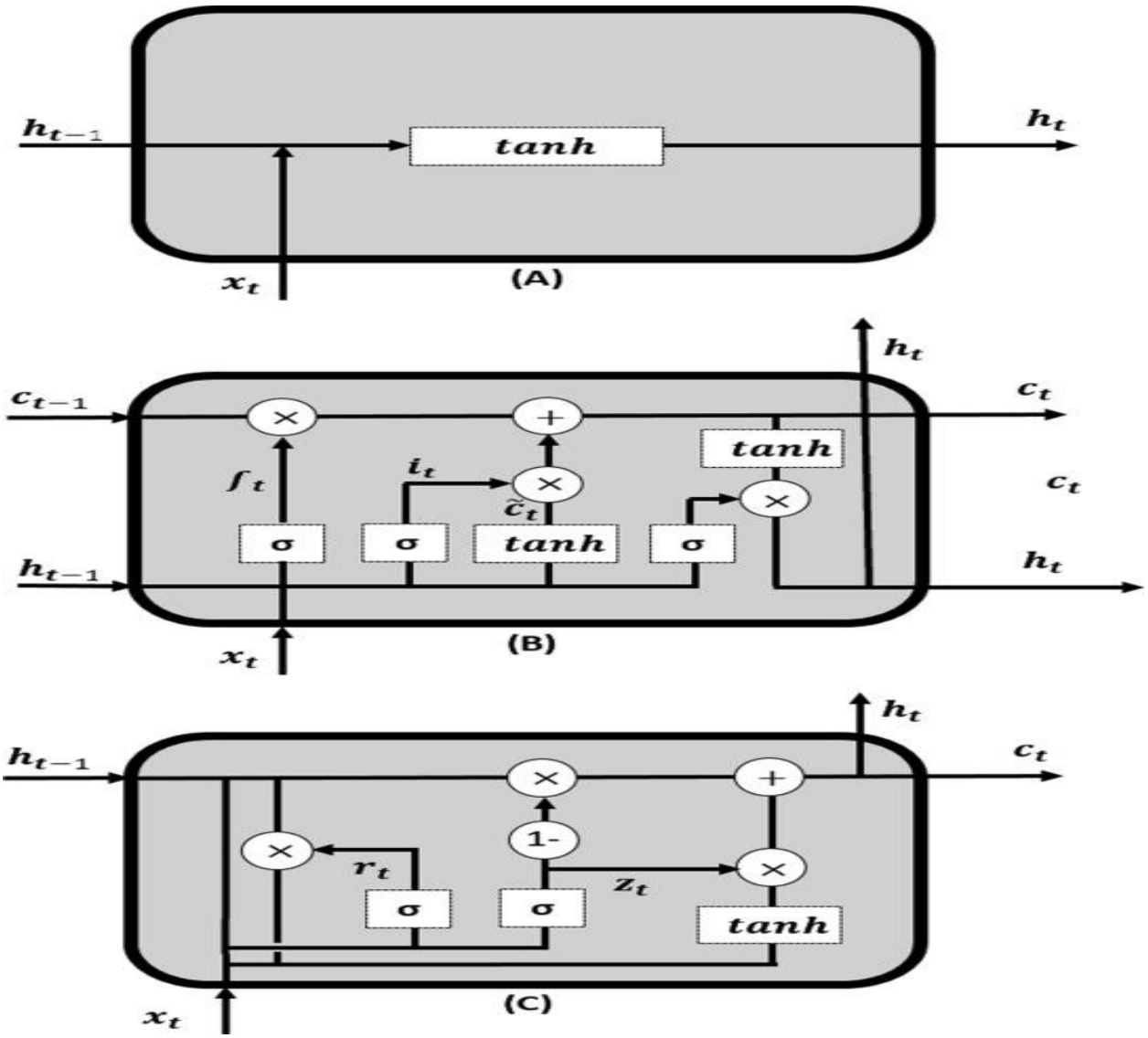

Figure 2: Schematic representation of (A) RNN, (B) LSTM and (C) GRU

Within the GRU architecture, the two primary gates are the update and reset gates, with the hidden state of the GRU described by a specific equation. In all the below equations * denotes element-wise application

$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t \tag{7}$$

The update gate, which determines how much of the GRU unit gets updated, is represented by the following equation:

$$z_t = \sigma(W_z * [h_{t-1}, x_t]) \tag{8}$$

The reset gate is determined by the equation below:

$$r_t = \sigma(W_r *[h_{t-1}, x_t]) \tag{9}$$

The reset gate's hyperbolic tangent function is also known as the "new remember" gate and the following function defines it.

$$h_t = tanh(W *[r_t * h_{t-1}, x_t]) \tag{10}$$

In conclusion, the Gated Recurrent Unit (GRU) emerges as a refined and efficient alternative to the LSTM. Its more compact design, achieved by fusing gates and unifying states, does not diminish its capacity to model long-term dependencies, solidifying its place as a powerful entity in deep learning.

2.4 Hybrid Model (LSTM+GRU)

The recognised advantages and inherent drawbacks of LSTMs and GRUs were the foundation for developing the hybrid LSTM-GRU model. While LSTMs' intricate gating mechanisms excel at capturing long-term dependencies, they have a higher computational cost. In contrast, GRUs simplify the gating in their search for efficiency but can sacrifice depth. The hybrid concept was designed as a compromise combining the best aspects of both worlds.

The hybrid model fundamentally combines LSTM and GRU cells. This complex pattern represents a careful integration rather than a simple juxtaposition:

LSTM Layers: As the deep memory chambers, these layers are adept at holding onto long-term dependencies, a characteristic of their unique cell state alongside the hidden state. Their trio of gates — input, forget, and output — meticulously regulate the flow of information, ensuring relevant data is stored, and irrelevant data is purged.

GRU Layers: These layers act as the agile processors. By merging the forget and input gates into a single update gate and having a joint representation for the cell and hidden state, GRUs expedite information propagation.

Figure 3 (Ullah *et al.*, 2021) presents the architectural design of the Hybrid model. The initial hidden layer employs a GRU in this configuration, followed by an LSTM in the subsequent layer. This arrangement effectively amalgamates the capabilities of both models into a singular, cohesive structure.

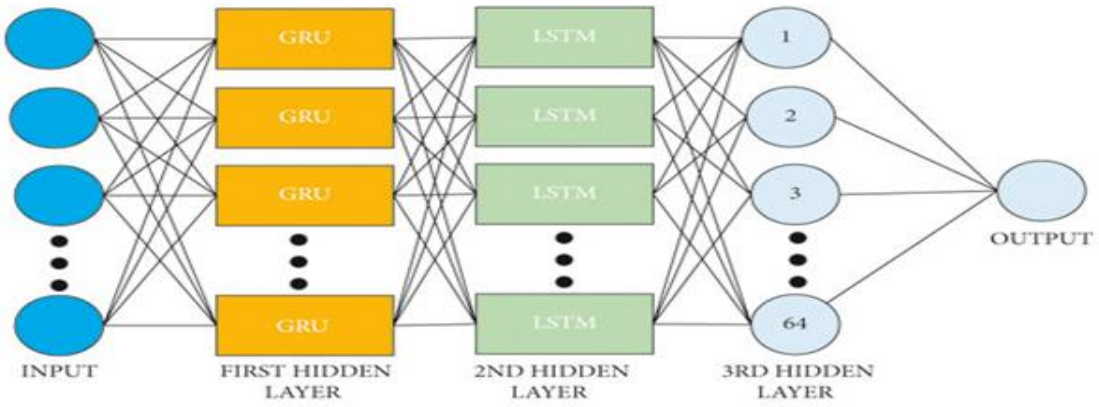

Figure 3: Architecture of Hybrid Model

The hybrid LSTM-GRU model, which effectively integrates the advantages of both architectures, represents a substantial advancement in sequence modelling. This hybrid model provides a richer learning experience by efficiently collecting current nuances and crucial information over the long term. While the LSTM layers offer significant insights, the GRU layers assure computational pragmatism, making the model suitable for real-world applications. It finds a harmonious compromise between depth and efficiency. This hybrid approach's built-in flexibility makes it stand out. The design can be adjusted depending on the difficulty of the task at hand, enabling researchers to change the proportion of LSTM and GRU layers to ensure the model is most suited to the problem's requirements.

This ground-breaking design offers evidence that ground-breaking discoveries frequently result from the fusion of well-established concepts. Although the first results across several domains show promise, it is crucial to approach the hybrid model with rigorous diligence. We can only truly appreciate and utilise its full potential through meticulous fine-tuning, validation, and testing across various problems.

Chapter 3: Related Works

Predicting stock market price movements has always been challenging due to the intricate dynamics. Pramod and Pm (2021) delved into this complexity, emphasising the challenges, and highlighting the potential of machine learning to address them. In particular, they focused on the Long Short-Term Memory (LSTM) network, a type of recurrent neural network that offers the capability of weight adjustment for specific data points through stochastic gradient descent. Traditional prediction methodologies like the ARIMA and multiple linear regression models have historically been foundational tools for stock price predictions (Pramod and Pm, 2021). However, the rise of computational capabilities and machine learning has paved the way for more advanced techniques that promise enhanced accuracy.

Researchers have introduced models integrating various technical indicators, including investor sentiment markers and financial data (Gao, Wang and Zhou, 2021). In these methodologies, the dimension reduction is often executed using depth learning LASSO and PCA approaches. LSTM models have been lauded for their efficacy in forecasting stock market trends globally. The GRU architecture, a streamlined version of the LSTM, has also shown promise in stock forecasting endeavours. Researchers amalgamated various technical indicators from the Shanghai Composite Index data in a comprehensive study. They employed dimension reduction techniques and deep learning models like LSTM and GRU for forecasting. Their analysis, which included models like LASSO-LSTM, LASSO-GRU, PCA-LSTM, and PCA-GRU, provided a comprehensive perspective on stock price prediction (Gao, Wang and Zhou, 2021).

A ground-breaking approach was introduced in another study that combined the strengths of bidirectional long short-term memory (Bi-LSTM) and gated recurrent unit (GRU) networks (Karim, Foysal and Das, 2022). Their hybrid deep learning model aimed to enhance the accuracy of stock price forecasts. Using the NIFTY-50 stock market data for evaluation, they demonstrated that their hybrid model could offer accurate long-term forecasts. This hybrid Bi-LSTM-GRU model consistently outperformed its counterparts, reinforcing the idea of leveraging multiple neural network architectures for enhanced prediction accuracy (Karim, Foysal and Das, 2022).

Deep learning models have recently gained popularity in various fields, notably the Indian National Stock Exchange. Recurrent Neural Network (RNN), long short-term memory network (LSTM), and convolutional neural network (CNN) are a few examples of architectures that

have been used in studies to predict the trends of the NIFTY 50 stock market. Comparative evaluation highlights the value of feature selection and hyperparameter optimisation in raising prediction accuracy. Especially noteworthy is that LSTMs consistently beat RNNs and CNNs in terms of prediction accuracy as evaluated by Mean Squared Error (MSE) (Fathali, Kodia and Said, 2022).

Time series forecasting (TSF) is pivotal in predicting future data sequences from historical data. One of the most significant parameters affecting TSF's accuracy is the size of the look-back window (Shi, 2022). In a seminal study, Shi (2022). utilised deep learning algorithms to investigate the effects of various window sizes and forecasting durations. Their choice of dataset was the Beijing Air Quality Dataset, and their model comparison involved the Transformer model, which is renowned for its proficiency in language processing, against models like RNN, LSTM, and GRU. Their findings highlighted the superior performance of Transformer models for most predictions, further suggesting optimal look-back windows for different forecasting durations (Shi, 2022).

Chapter 4: Methodology

4.1 Overview

In this research, we follow a structured approach to predict NIFTY stock prices using advanced neural network models. Starting with data acquisition, we process the raw stock prices to make them suitable for modelling, ensuring they accurately represent the underlying trends and patterns. Subsequently, we employ various neural network architectures - LSTM, GRU, and a hybrid model of the two - to learn from the data. These models are fine-tuned to achieve optimal performance using various techniques. Lastly, we evaluate the models' forecasting abilities using several metrics, ensuring a comprehensive assessment of their capabilities.

A visual representation of the methodology is provided in the subsequent flowchart, guiding us through each step of the process.

Figure 4: Study Overview

4.2 Dataset Description

The study uses a Yahoo Finance dataset that shows the Nifty 50 index's performance from January 1, 2008, to December 30, 2022. The Nifty 50 index is a crucial tool for determining market trends in India because it summarises the performance of 50 well-known Indian companies.

There are seven columns in the dataset:

- Date: Indicates the specific trade day.
- Open: The index's value at the start of every trading day.
- High: The index's peak value for that day.
- Low: The day's lowest recorded value for the index.
- Close: The index's closing price at the close of trade.
- Adj Close: Adjusts the closing price to consider factors like dividends and stock splits.
- Volume: Denotes the total number of shares traded that day, providing insights into the level of trading activity.

With the help of this dataset, which acts as the study's basis, it is possible to analyse the trends and behaviours of the Nifty 50 index over 15 years in detail.

4.3 Data Preprocessing

During the initial data preprocessing phase, the dataset's structure was thoroughly examined using the df.info() function, revealing the data types of each column and any missing values present. A forward-fill method was utilised to address these missing values, populating gaps with the last valid non-null value. Furthermore, the 'Date' column was converted to a datetime data type to facilitate time-based operations in our subsequent analysis. Three backup copies were created to ensure the preservation of the original dataset, acting as safeguards against potential data modifications.

4.4 Exploratory Data Analysis (EDA)

A crucial component of our investigation into the NIFTY stock data is the exploratory data analysis (EDA). Using EDA, we set out on a quest to find obscure patterns, identify anomalies, and comprehend the underlying structure of our dataset, laying the groundwork for further modelling and forecasting activities.

4.4.1 NIFTY Closing Prices

The stock market is like a complex puzzle of many pieces influenced by the economy, politics, and societal trends. When we look at this big picture, one thing often catches our attention because of its importance: the final prices at which stocks are sold each day (closing prices).

- Closing prices: A stock's closing price indicates how investors feel about it at the end of a trading day. The trajectory of NIFTY's closing prices provides information on market valuation, investor sentiment, and the state of the stock. The upward trend in Figure 5 indicates periods of bullish sentiment, while any noticeable dips might symbolise market corrections or bearish phases.

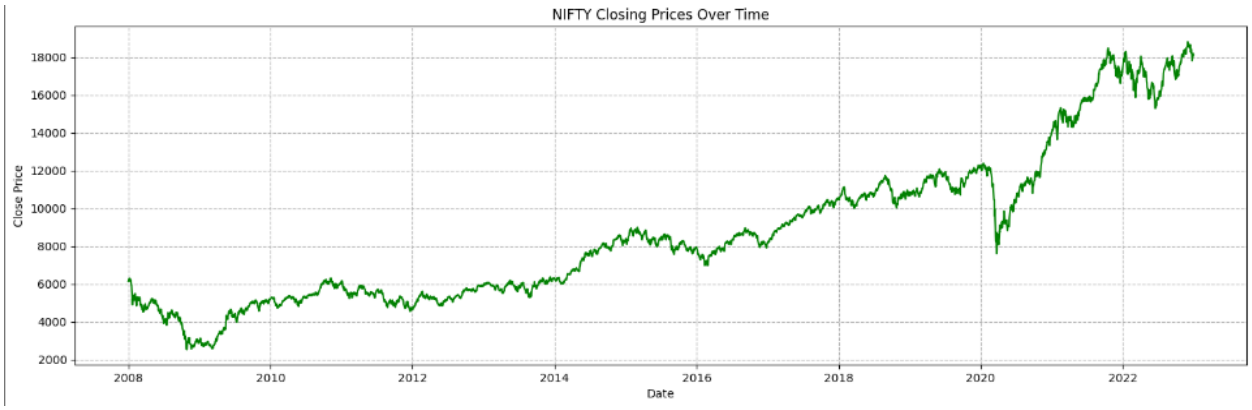

Figure 5: Historical Trend of Nifty 50 Closing Prices

4.3.2 Correlation Heatmap for Feature Relationships

The correlation heatmap provides a comprehensive visualisation that displays the linear relationships between various attributes of the NIFTY stock. Correlation values range from -1, indicating a perfect inverse relationship, to 1, signifying a perfect direct relationship. Values near 0 indicate minimal linear correlation.

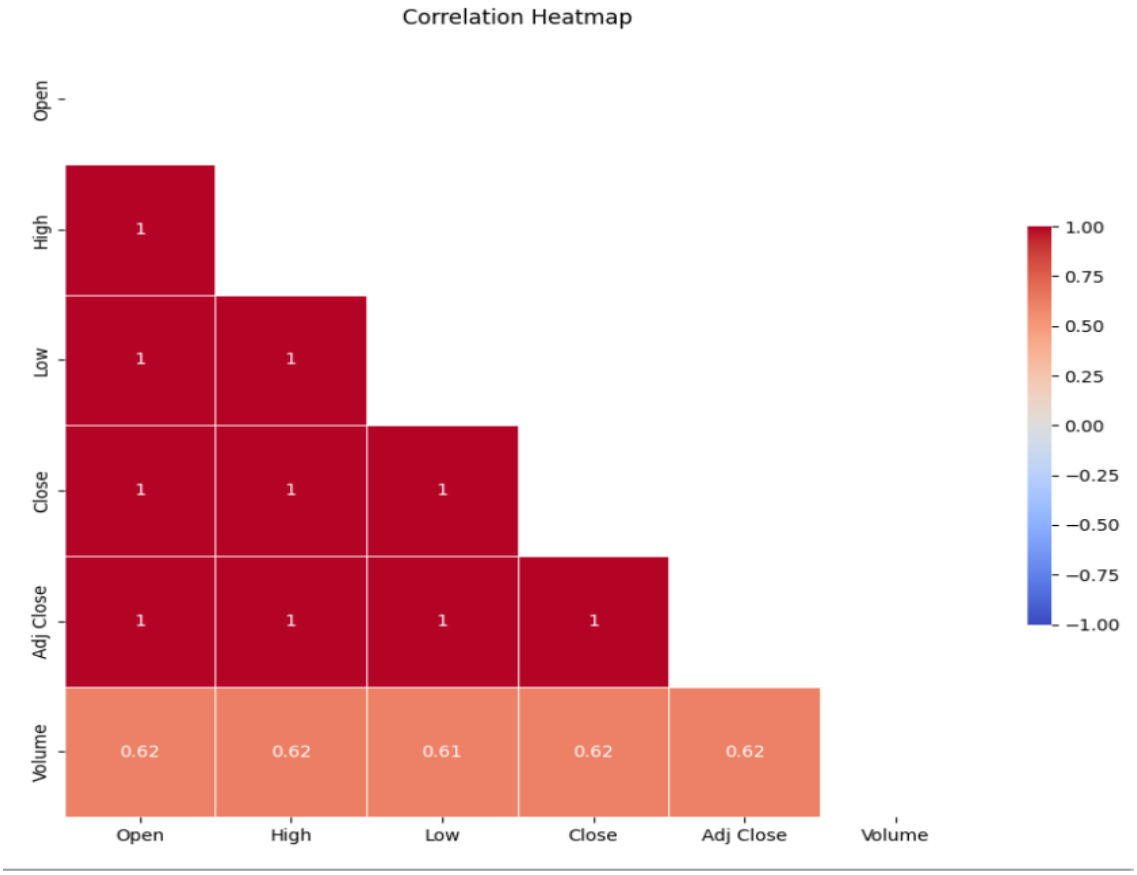

Figure 6: Correlation heatmap showcasing the relationships between various attributes of the NIFTY stock.

In stock trading, we can occasionally identify patterns that aid in comprehending how the market functions. One clear pattern is how the highest daily stock prices relate to the number of stocks traded. More of that stock is bought and traded on days when the price is high.

A correlation value of 0.61 denotes a moderately positive correlation. This shows a corresponding rise in trading activity on days when the stock price reaches its high. This correlation could stem from factors such as increased buying interest or potential selling pressures during high-price days.

Similarly, another relationship is between the stock's opening prices and the volume of trades. With a correlation coefficient of 0.61, there is a clear indication that the initial trading price

often sets the tone for the day's trading volume. Specifically, a significant surge in the opening price could catalyse heightened trading interest throughout the day, reflecting the market's response to the stock's opening stance.

4.3.3 Return distribution

The daily and monthly return distributions were plotted to show how volatile and risky the NIFTY stock is. Such visualisations help risk assessment in stock trading by highlighting usual variations and potential return outliers.

1. Distribution of daily returns: The histogram in Figure 7 below provides a snapshot of NIFTY's stock volatility. Predominantly, the returns are clustered around 0%, indicating that most days witness minimal stock movement. Most fluctuations lie within a 5% range, offering a sense of predictability. However, rare spikes or drops emphasise the market's potential unpredictability.

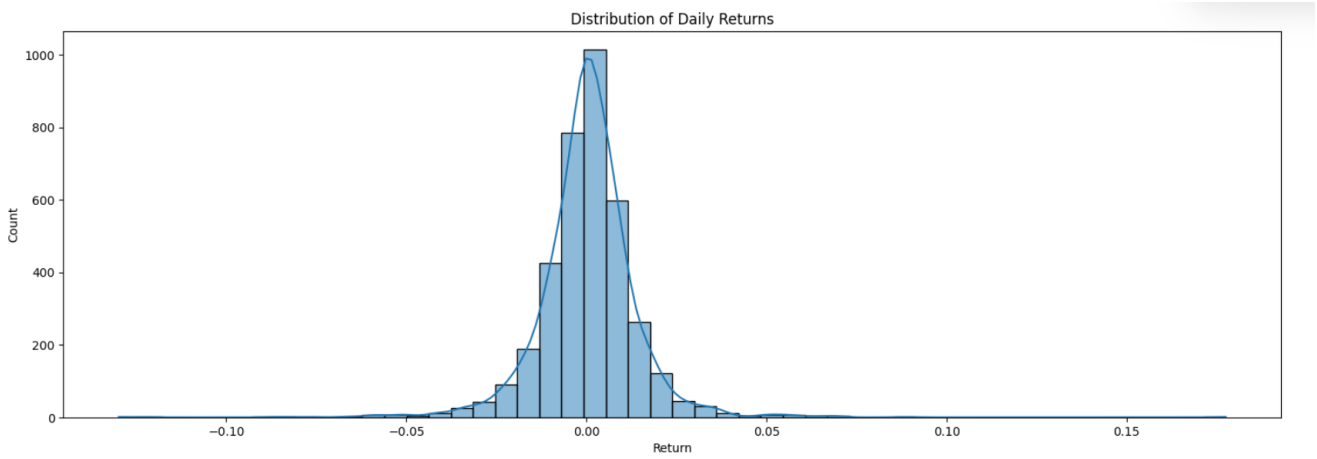

Figure 7: Distribution of NIFTY's Daily returns over 15 years

- Most returns are centred around 0%, suggesting stability.
- Returns predominantly fluctuate within a [-0.05,0.05] range.
- Occasional significant returns highlight the need for investor vigilance.

2. Distribution of Monthly Returns: The distribution of monthly returns offers insights into longer-term trends and movements of the NIFTY stock. Unlike daily returns, which primarily revolve around short-term fluctuations, monthly returns encapsulate broader market dynamics. The following Figure 8 illustrates the distribution of Monthly returns.

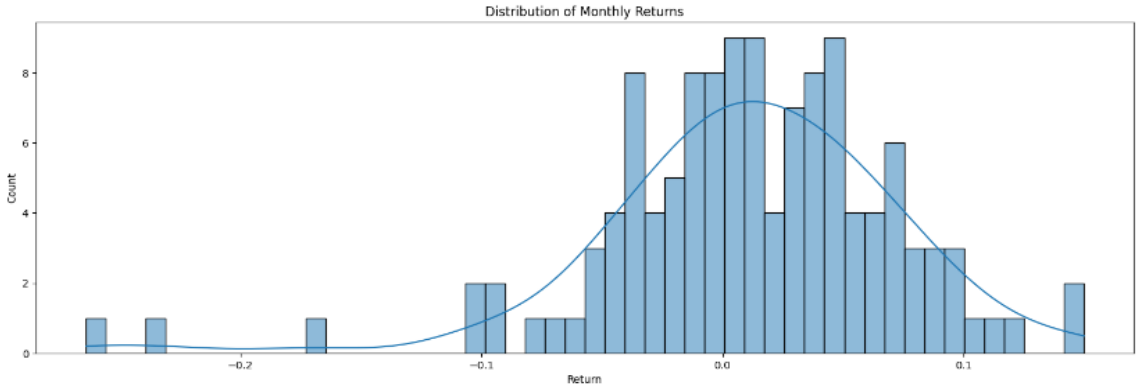

Figure 8: Distribution of NIFTY's Daily returns over 15 years

From the above Figure 8, we can identify the following points.

- Most monthly returns are centred around 0%
- Most returns fall within a 10% fluctuation range, specifically between [−0.1,0.1] [−0.1,0.1].
- There is a distinct occurrence of higher positive monthly returns, with some reaching up to 15%.
- The presence of these higher monthly returns suggests the potential for substantial gains over more extended periods, even if short-term daily fluctuations appear minimal.

4.4.4 Temporal structures- Autocorrelation insights

Understanding the temporal connections present in a dataset is crucial in time series analysis. Autocorrelation functions (ACF) and partial autocorrelation functions (PACF) are powerful tools in this endeavour, shedding light on potential lagged relationships and guiding the configuration of time series forecasting models.

1. ACF is a statistical tool that measures the correlation between a series and its lagged version. It helps to identify how much a data point in a series is influenced by its past data points.
2. PACF measures the correlation between a series and its lagged version after removing the correlation effect due to the terms at shorter lags. It helps identify the direct relationship between an observation and its lag without the influence of intermediate observations.

The following Figure 9 indicates the ACF and PACF graph for my dataset.

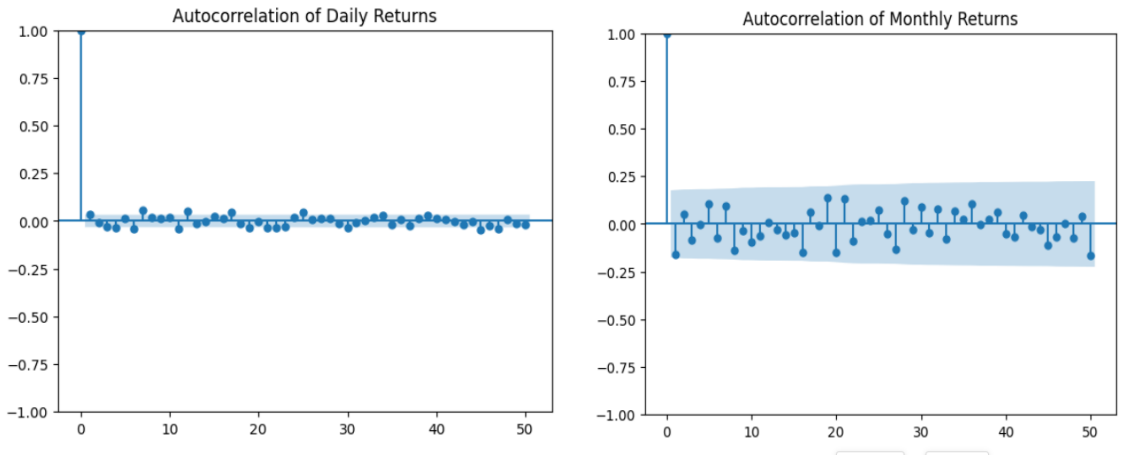

Figure 9: ACF and PACF graph of NIFTY Stock returns

- Immediate Lags (Lag 0): Both the ACF and PACF values at lag 0 are unity. This stems from the inherent nature of datasets, where a series is always perfectly correlated with itself.
- Short-Term Lags: Examining the initial few lags (e.g., 1, 2, 3), we discern that the values for ACF and PACF are small. This translates to the conclusion that returns share a minimal correlation from one day to its immediate successor. Elucidate, the return on day *t* correlates at a modest 0.0337 with the return on its preceding day *t-1* (lag 1). Such weak correlations indicate that the preceding day's performance does not majorly dictate the returns of a given day.
- Extended Lags: Venturing into more considerable lags, there is a palpable fluctuation in the ACF and PACF values. These oscillations, however, do not scale to exceptionally high values, implying an absence of pronounced autocorrelation patterns in returns over elongated durations.
- Overall Dynamics: The ACF and PACF measurements move around zero on a larger scale, with irregular peaks or valleys. This pattern typifies a white noise series, suggesting that the returns are random and are not influenced by their historical values.

4.4 Model Implementation

Predictive model implementation is the focus of this section, which also serves as the research's core. Since stock market data is inherently complicated, we have picked three cutting-edge models to identify the patterns and connections in the dataset. Modify and preprocess the data and ensure optimal performance; each model is accompanied by a unique encoding technique.

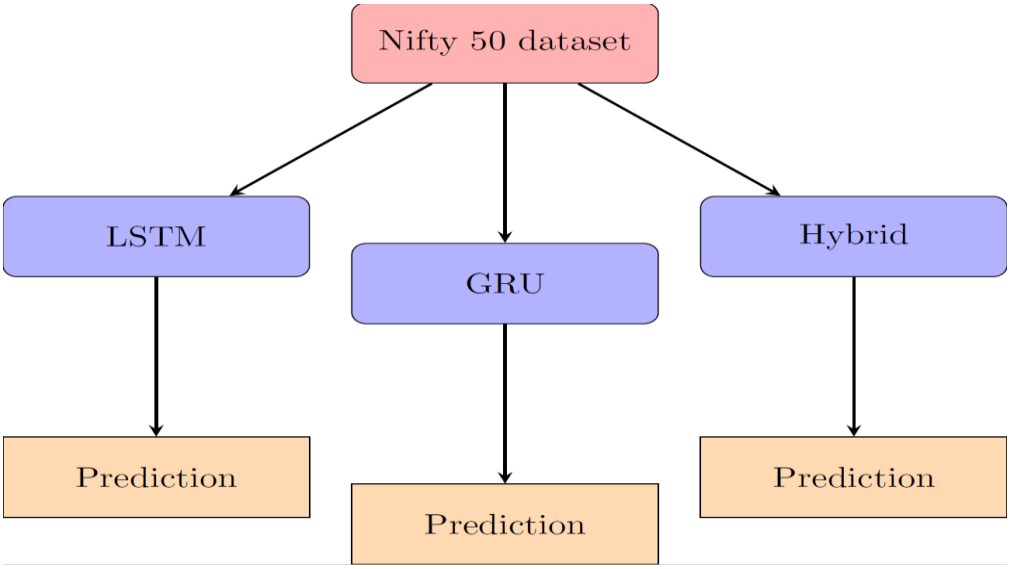

Figure 10: Outline of the model implementation

4.4.1 LSTM Implementation

Data preprocessing is the first step in the process, ensuring that the data is compatible with the LSTM model. The closing prices retrieved from the Nifty 50 dataset were transformed using the MinMaxScaler due to the model's sensitivity to scale. By restricting the values to those between 0 and 1, this scaling created the ideal learning environment for the LSTM.

The data was partitioned once it had been scaled; 20% was set aside for testing, with a sizeable portion, 80%, designated for training. However, the LSTM is better at identifying patterns than sequences. The training data were made into sequences that lasted 60 days to use this potential. The data is reshaped into 3D arrays to be compatible with the input specifications of the LSTM. The meticulous preparation did not stop there. The training set underwent another split to validate the model during training, setting aside 20% for validation.

The architecture of the LSTM model is shown in Fig 11. It consists of multiple layers, each layer serving a distinct purpose.

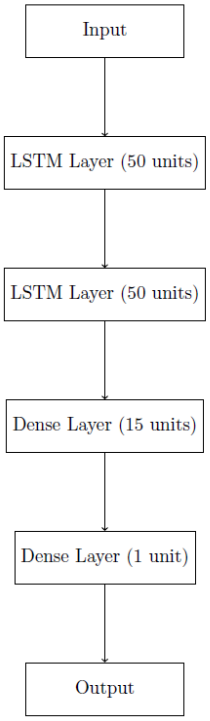

Figure 11: LSTM Architecture

The foundation was laid with the LSTM layer, housing 50 units. This layer was designed to return sequence, ensuring seamless integration with the subsequent layers.

The second LSTM layer mirrored the initial layer in terms of units but diverged by not returning sequences. Following the LSTM layers were two Dense Layers. The first layer housed 15 units, and the concluding layer, which represented the predicted closing price, contained a singular unit.

The model, once architected, was compiled, utilising the Adam optimiser and the mean squared error as the yardstick for loss.

Training an LSTM model was approached with precision. We employed the Early Stopping technique, which monitors the validation loss and stops training if no improvement is observed over consecutive epochs. This method effectively prevents overfitting. The model's training was set to run for three epochs, but early stopping could intervene if the model's performance did not progress.

After the training phase, the model was evaluated. Predictions generated were in a scaled format, so they were transformed back to their original scale to represent the actual stock prices. After the prediction phase, the model underwent a thorough test to determine its accuracy. The following statistical metrics were used:

Mean Squared Error (MSE): This metric measures the average squared difference between the estimated and actual values. A lower MSE indicates a better fit to the data.

Root Mean Squared Error (RMSE): Representing the square root of MSE, RMSE provides a clearer perspective of the error in the same units as the target variable.

Mean Absolute Error (MAE): This metric computes the average of absolute differences between the predicted and actual values, offering insights into the magnitude of errors regardless of their direction.

The values derived from these metrics provided a comprehensive understanding of the model's prediction capability in both magnitude and direction.

The LSTM model, trained on historical data, was transitioned to a forward-looking role. Its primary task was forecasting the closing prices for an upcoming 60-day interval. The methodology adopted was a rolling-window strategy. The model consistently received a data segment comprising the latest 60 days. As new predictions emerged, this data window was refreshed by integrating the recent prediction, ensuring the model's predictive decisions were based on the most current data available.

An in-depth hyperparameter optimisation was undertaken to elevate the LSTM model's predictive accuracy. Utilising Kerastuner's advanced functionalities, the objective was to pinpoint the optimal values for several pivotal hyperparameters:

1. LSTM Units: This refers to the number of memory cells in the LSTM layer, directly correlating with the model's ability to discern intricate data patterns.
2. Dropout Rate: As a countermeasure against overfitting, this parameter specifies the fraction of randomly excluded neurons during the training phase, promoting model generalisation.
3. Learning Rate: This parameter governs the magnitude of adjustments made to the model's internal weights during each training iteration. Knowing how swiftly and accurately the model converges to an optimal solution is crucial.

The optimisation strategy was anchored around the RandomSearch technique, which systematically explores diverse hyperparameter value combinations. After multiple experimental runs, the most favourable hyperparameter set was identified. This optimised configuration became the foundation for constructing a superior LSTM model architecture primed for enhanced forecasting precision.

4.4.2 GRU Implementation

The 'Close' column of the Nifty 50 dataset was normalised using MinMax scaling, like the LSTM. However, to capitalise on the GRU's strength in analysing sequences, we organised the data into 60-day chunks. This setup helps the model use past data effectively for future predictions. The data was reshaped into a three-dimensional format to fit the GRU's input

needs. We split the data in the same format used for LSTM and set aside some training data for validation, allowing us to monitor the model's progress during training.

Fig. 12 shows the GRU model's architectural layout. The model is divided into layers, each created with a specific purpose and functionality.

The initial GRU layer, which has 50 units, is the base layer. To provide a seamless transfer of data to the following layers, this layer was designed to be able to return sequences.

While maintaining the 50-unit design of its predecessor, the second GRU layer takes a different approach to processing. This layer's primary function is to process and refine the received sequences because, unlike the previous layer, it does not return sequences.

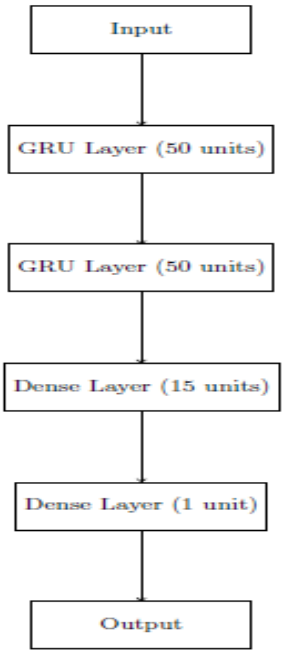

Figure 12: GRU Model Architecture

After the architectural design of our GRU model was complete, it went through a compilation step. We employed mean squared error served as our loss function. To enhance the optimization process, we used Adam optimiser, an optimisation algorithm renowned for computing adaptive learning rates for each parameter and we incorporated an early stopping mechanism to support the training dynamics and guarantee the model's effectiveness. This prudent feature halts the training process if there is a stagnation in the model's performance improvement. After the training phase, the model's predictive accuracy was analysed.

The initial predictions, articulated in a normalised form, transformed to revert to their original scale, ensuring they resonated with the actual stock prices. Looking beyond just historical analysis, the GRU model was also responsible for making future predictions. Adopting a rolling-window strategy, the model continually refreshed its data foundation, ensuring its forecasts were permanently anchored in the most recent data available. Using a rolling window method, the model often updated its database, ensuring that its forecasts were always based on the most recent information.

We entered the world of hyperparameter optimisation in our pursuit of perfection. Leveraging the advanced capabilities of Keras-tuner, a powerful library for hyperparameter tuning in neural networks, we adopted a systematic approach to fine-tune the model's parameters, to their optimal values. The number of units in the GRU layers and the optimiser's learning rate were the main focuses of this optimisation. The RandomSearch algorithm, which began a thorough study of various configurations, assisted this complex process. This project's zenith was the discovery of the ideal model configuration, which established a new standard for prediction accuracy.

4.4.3 Hybrid Model Implementation

The Hybrid model emerges as a ground-breaking approach to stock price prediction. Seamlessly amalgamating the prowess of LSTM and GRU units, it introduces the novel Time2Vec layer. The primary objective is to capitalise on the synergies of these components to provide unparalleled precision in decoding temporal data sequences. Borrowing from our established workflow for the LSTM, the 'Close' column from the Nifty 50 dataset underwent a MinMax scaling normalisation. Such a procedure ensures that data remains within a manageable range, making it ideal for neural networks. The scaled data was then strategically partitioned: 80% was dedicated to training, and the remaining 20% was reserved for testing. A subset of the training data, about 20%, was further earmarked for validation, allowing for real-time assessment during the model's training journey.

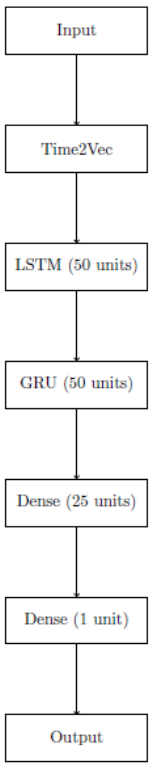

Figure 13: Hybrid Model Architecture

The architectural blueprint of the Hybrid model is both innovative and intricate. The Time2Vec Layer stands as its inaugural component.

Unlike conventional layers, Time2Vec obtains linear and cyclical time-based patterns, enhancing the model's time-sensitive comprehension. After the Time2Vec, the architecture integrates a 50-unit LSTM layer. This layer is adept at identifying and memorising long-term dependencies, thus enriching the model's pattern recognition capabilities over extended data sequences.

The next component is a 50-unit GRU layer, which meticulously refines the data sequences, ensuring that the most pertinent insights are conveyed to subsequent layers. Rounding off the architecture are two dense layers, with the mandate of synthesising the insights derived and projecting the anticipated stock price.

After designing the model, we started setting it up for training. We chose the Adam optimiser because it is known for adapting well during training. Combined with the mean squared error, this setup ensures the model learns quickly and predicts accurately. We also added a feature called 'early stopping'. If the model stops improving during training, this feature stops it altogether. This prevents the model from memorising the training data (overfitting) and saves time.

When the model finished training, we adjusted the predictions. We had scaled down the data for training, so we scaled it back up to compare with the actual stock prices. We used measures like MSE, RMSE, and MAE to see our model's accuracy. We also set the model to predict stock prices for the next 60 days, always using the latest data for its predictions.

We set out on a careful journey to fine-tune the parameters of the Hybrid model, ensuring its maximal performance by leveraging the power of the Keras tuner. The main focuses of this optimisation were the number of units in the LSTM, GRU, dense layers, and the number of units. The best model configuration was identified at the end of this meticulous process, which set a new standard for stock forecast accuracy.

4.5 Evaluating model predictions against actual stock market data.

For our research, we used a dataset that had stock market prices of 60 days which are equivalent to the days which models forecasted. We compared this real data to predictions from LSTM, GRU, and a Hybrid model.

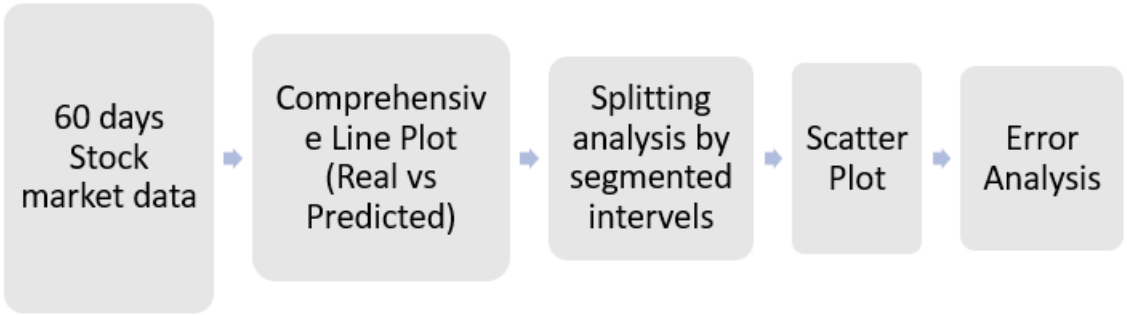

Figure 14:Overview of evaluation

We made line graphs to see how close the model predictions came to the real data over the 60 days. We also looked at specific times, like the first 20 days, to see how the models did in the short term.

Using scatter plots, we compared accurate stock prices to predicted values. We drew a 'perfect prediction' line on these plots. They were off if the model's predictions were far from this line.

Lastly, we checked how far off the model predictions were from the real data. We used a moving average of five days to understand these errors better. Our method combined visuals with numbers to see how well the models did. The results will help us see which model best predicts stock prices.

Chapter 5: Results and Discussion

The empirical findings from the LSTM, GRU, and hybrid models for stock market forecasting are discussed in this section. Each model's performance is discussed in detail, including the results of hyperparameter tuning. Their performance indicators are thoroughly compared. We next go into interpreting the findings, emphasising how the model projections affect stock market forecasts, addressing the earlier research objectives and assumptions, and examining how deep learning could enhance predictive accuracy. We also examine each model's advantages and disadvantages to draw conclusions that were revealed by examining the data.

5.1 LSTM Model

Key performance measures used to assess the LSTM model's performance include Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Squared Error (MSE). By displaying the correlation between expected and actual stock prices, we also shed light on the model's capability to predict trends and market volatility accurately.

The LSTM model's performance was quantitatively assessed using the following metrics:

- Mean Absolute Error (MAE): The calculated MAE of 535.55 indicates the average absolute difference between the predicted and actual stock prices. This metric highlights the model's accuracy in forecasting stock prices, implying that the model accurately predicts stock price trends and movements because the forecasts are close to the actual prices.
- Root Mean Square Error (RMSE): With an RMSE of 611.23, the LSTM model can predict stock price variations with a relatively small error magnitude. A lower RMSE indicates that the model captures price fluctuations more closely.
- Mean Squared Error (MSE): The calculated MSE of 373601.83 measures the average squared difference between the predicted and actual stock prices. This metric complements the RMSE in quantifying the model's predictive accuracy.

Collectively, these indicators demonstrate the LSTM model's capacity to produce forecasts that, on average, closely match stock values. The small values of MAE and RMSE indicate that the model captures the general trends and fluctuations within the stock market dataset. However, the specific interpretations of these metrics should be considered within the context of the stock market's volatility and inherent unpredictability.

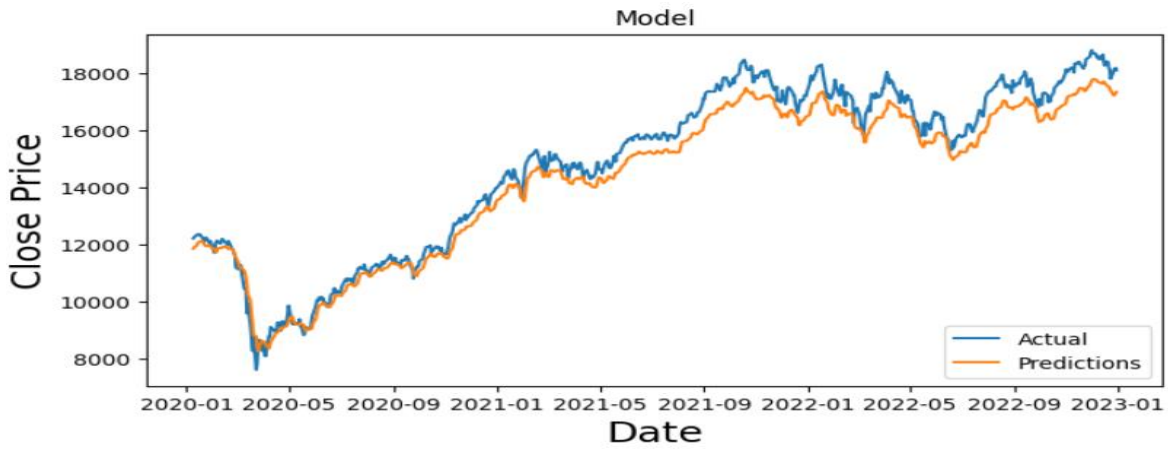

Figure 15:Actual vs. Prediction graph of the LSTM Model

The following Figure 15 is the visual representation of the Actual vs. predicted Graph, and it showcases the alignment between predicted and actual stock prices.

Figure 15 shows the actual close price of a stock (blue line) and the predicted close price by the LSTM model (red line) from January 2020 to January 2023. The x-axis shows the date, and the y-axis shows the close price. Overall, the predicted prices are generally close to the actual prices. However, there are a few notable deviations. For example, in January 2021, the predicted price was much lower than the actual price. This deviation could be because there was a significant event in the stock market that month that the model did not consider. Collectively, these indicate the LSTM model's capacity to produce forecasts that, on average, closely match stock values.

The following Learning Curves Graph (Fig 16) illustrates the convergence of training and validation errors over multiple epochs. These curves offer insights into the model's training process and its generalisation ability to new data. A close alignment between the training and validation errors indicates that the model is neither underfitting nor overfitting. These curves are crucial for hyperparameter optimisation and training strategy refinement.

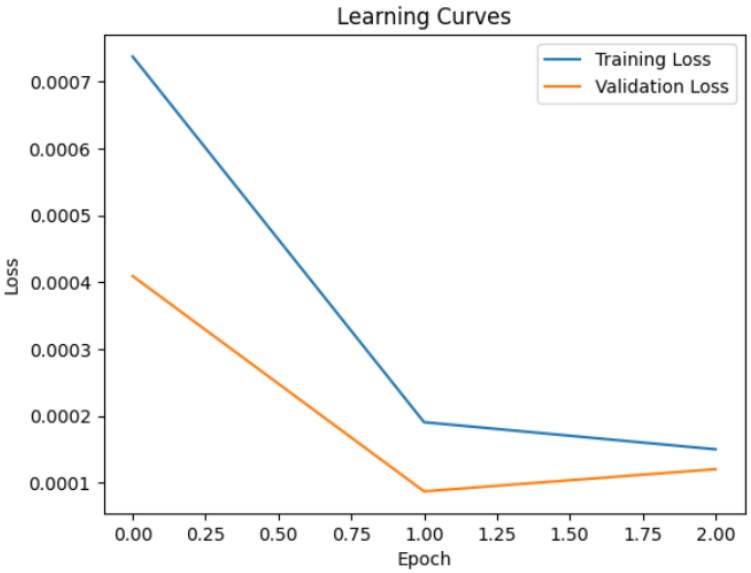

Figure 16: Learning curves of LSTM

In the Learning curves graph x-axis shows the epoch, and the y-axis shows the loss. The training loss decreases as the epoch increases, and the validation loss decreases similarly. This indicates that the model is not overfitting the training data. The ideal situation is for the training and validation losses to converge to a low value.

In this case, the training and validation losses converge to a low value after epoch 2. This means the model cannot overfit the training data and can generalize to new data.

Fig 17 shows the predicted stock price graph for 60 days, which displays the predicted stock prices over 60 days, with the y-axis representing the predicted stock price and the x-axis indicating the number of days into the future. This graph allows us to assess the LSTM model's ability to forecast stock prices in the short term.

Figure 17: Prediction using LSTM for 60 days.

From the graph, we can understand that the expected stock price decreases as the number of days increases and expected stock price is more volatile in the short term than in the long term. This could be because the model is taking account the fact that the stock price is likely to fluctuate over time.

The LSTM model's hyperparameter tweaking method produced ideal settings that improve its stock market forecasting performance. The following hyperparameters were found to be the ideal configuration after extensive testing:

```
# Get the best hyperparameters from the tuner
best_hyperparameters = tuner.get_best_hyperparameters(num_trials=1)[0]

# Print the best hyperparameters
print("Best Hyperparameters:")
print("Number of LSTM units:", best_hyperparameters.get('units'))
print("Dropout rate:", best_hyperparameters.get('dropout'))
print("Learning rate:", best_hyperparameters.get('learning_rate'))
```

```
Best Hyperparameters:
Number of LSTM units: 128
Dropout rate: 0.05
Learning rate: 0.0001
```

Figure 18: Best hyperparameters for LSTM

This results which is shown above are obtained by using RandomSearchCV for LSTM model. The selection of 128 LSTM units highlights the model's architectural complexity, enabling it to uncover intricate temporal patterns within the data. This heightened complexity empowers the model to capture fine-grained details and complexities, effectively boosting its predictive capabilities.

A regularisation process is introduced during training by including a dropout rate of 0.05, effectively allaying worries about overfitting. The model is encouraged to generalise more by randomly eliminating connections between LSTM units, which improves performance on untried data. Like gradient descent optimisation, a lower learning rate 0.0001 ensures accurate weight updates and slow convergence to optimal solutions. This methodical strategy reduces overshooting and makes more accurate predictions.

5.2 GRU Model

The GRU model's performance was critically assessed using key metrics, which are used for LSTM, as well as providing a clear perspective on its forecasting capabilities:

- MAE: The recorded MAE for GRU model is 392.42. This suggests that the model’s prediction deviate by approximately 392.42 units from the actual stock prices.
- RMSE: The GRU model has an RMSE of 441.55. This value is relatively small, points to the model’s efficiency in predicting stock price fluctuations, suggesting that it resonates well with actual market shifts.
- MSE: The MSE for the GRU model is 194969.95. This enhances the RMSE's comprehension of the model's capacity to generate predictions nearly coinciding with actual stock prices.

The above metrics show that the GRU model exhibits strong performance, making it an efficient tool for this forecasting task.

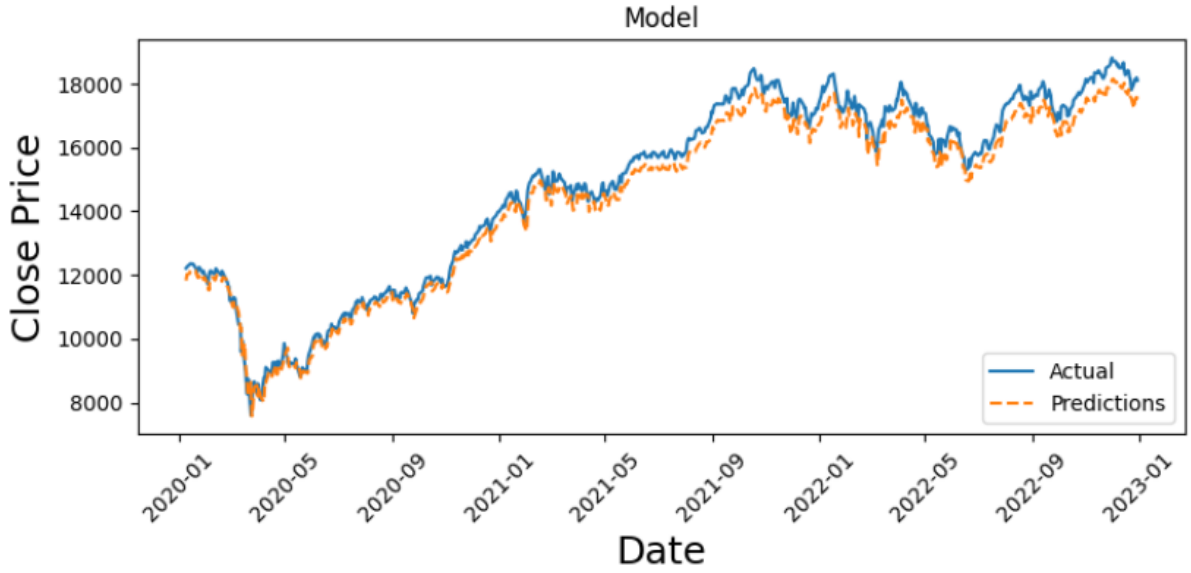

Figure 19: Actual vs predictions graph of the GRU Model

In Fig. 19, the GRU model's predicted closing price is shown as a red line, while the actual closing price of a stock is shown as a blue line, spanning from January 2020 to January 2023. The vertical axis indicates the closing price, while the horizontal axis represents time.

Even if most predictions and prices closely match, some notable deviations exist. Notably, the model's prediction for January 2021 was far lower than the actual value. This discrepancy may have resulted from a critical stock market event that the model missed. Like September 2022, the model's forecast significantly outperformed the actual number.

The Learning curve for the GRU, as depicted in Figure 20, offers a visual insight into the model’s training over successive epochs.

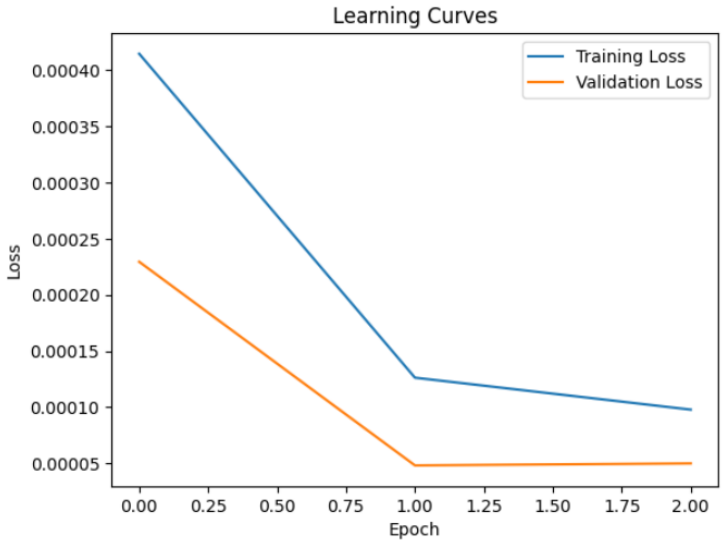

Figure 20: Learning Curves for GRU

On the graph, the horizontal line (x-axis) shows the number of times the model has been trained (epochs), and the vertical line (y-axis) shows the errors or mistakes the model made during training and testing.

From the graph, we can see a few things:

1. Both training and testing errors go down together as the model is trained more, which is a good sign. It means the model is learning well without memorizing the training data.
2. After training the model a couple of times (around the second time or epoch), the errors do not reduce much further. This means the model has learned as much as it can from the data given.
3. Since both training and testing errors are low and close together, the model is doing a good job. It is not just memorizing the training data but is also able to make good predictions on new data it has not yet seen.

In simple terms, Figure 20 tells us that the GRU model works well for the given task. After training it a few times, it learns to make good predictions without making many mistakes.

The stock price that the GRU model predicts will be for the next 60 days is shown in Figure 21 below. Future days are shown on the x-axis at the bottom of the graph, while anticipated stock prices are shown on the y-axis.

From the following graph (figure 21) we can understand that the model predicts that the stock price will increase at first. The model predicts that the stock price will decrease somewhat as we project further into the future. This suggests that the model is aware of how frequently stock prices fluctuate.

The forecasted prices see significant ups and downs in the early days. This could result from the model considering recent news or events that suddenly alter stock prices. However, the prices do not fluctuate less than we look farther out.

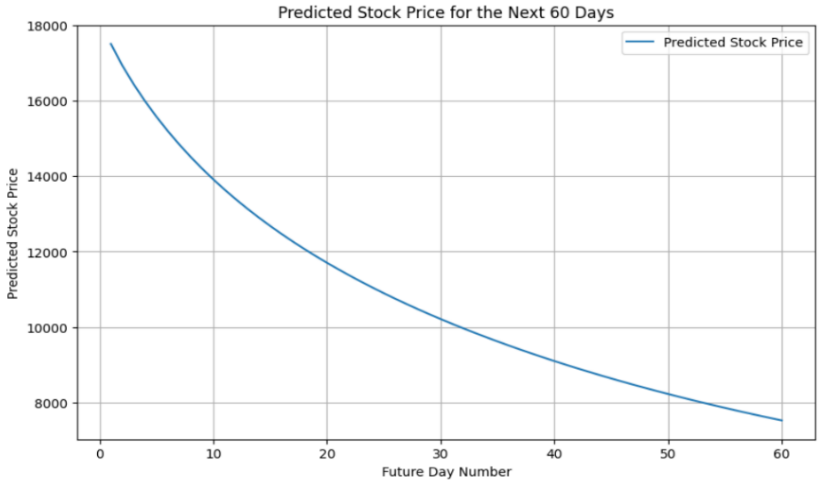

Figure 21: Prediction using GRU for 60 days.

In the following figure 22 we can find the best hyperparameters for the GRU model.

```
print("Best Hyperparameters:")
for hp, value in best_hyperparameters.values.items():
    print(f"{hp}: {value}")
```

```
Best Hyperparameters:
units_1: 300
units_2: 90
learning_rate: 0.0001
```

Figure 22: The best hyperparameters for the GRU Model

Ensuring that a GRU model performs at its peak entails fine-tuning its hyperparameters. Our approach to pinpointing these optimal settings harnessed the power of RandomSearchCV an efficient method for exploring a wide array of hyperparameter combinations. The speed at which the model learns is called the learning rate. If it moves slowly, the model may notice some crucial information. It takes a long time to learn if it moves too slowly. The learning rate of our GRU model is set to 0.0001, which we determined to be the ideal value. It aids in the model's efficient learning without dragging or hurrying.

5.3 Hybrid Model

The Hybrid model, designed to amalgamate the strengths of the LSTM and GRU models, has been evaluated on the performance as same as the LSTM model and GRU model.

MAE: The calculated MAE for Hybrid model is approximately 435.81. An MAE of 435.81 means that, on average, the predictions made by the model are off by about 435.81 units from real stock price.

RMSE: The Hybrid Model exhibits an RMSE of approximately 523.47. This metric gives us a sense of the model's prediction error magnitude. Our RMSE indicates that the model is adept at accurately simulating the stock market's volatility, which is shown by a lower RMSE score.

MSE: The average squared difference between the model's forecasted stock prices and the actual stock prices is 274018.97, showing the model's projections' overall accuracy and dependability.

Discussing these metrics in tandem, it becomes evident that the Hybrid Model exhibits a commendable performance in stock price forecasting. The model's aptitude to capture the essence of stock market patterns and movements is highlighted by the relatively low values of MAE and RMSE. The model has struck a balance in leveraging the strengths of its constituent models, resulting in predictions that closely align with actual stock market behaviours.

The following figure 23 represents the actual vs predicted price of Hybrid model.

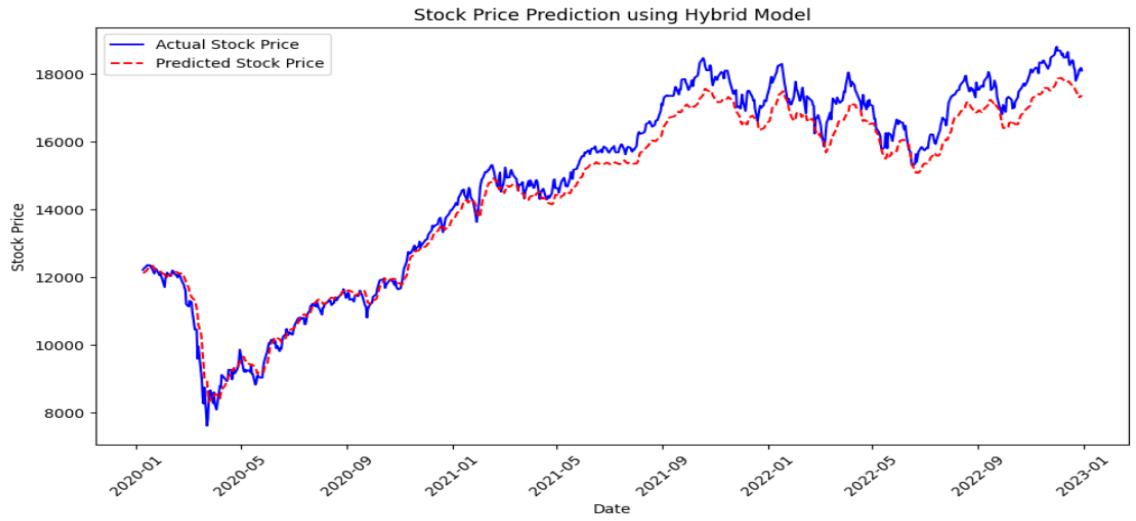

Figure 23: Actual vs Predicted stock price.

The graph (figure 23) shows the comparison between the red line, which represents the Hybrid model's performance from January 2020 to January 2023, and the blue line, which represents the actual stock prices, from January 2020 to January 2023.

The learning curve of the hybrid model (figure 24) shows that the training loss decreases as the epoch increases. This is a good sign, as the model is learning to fit the training data better. The training loss is the loss of the model on the training dataset. The validation loss is the loss of the model on the validation dataset. The validation loss decreases as the epoch increases and at a similar rate to the training loss. This suggests that the model is balanced with the training data.

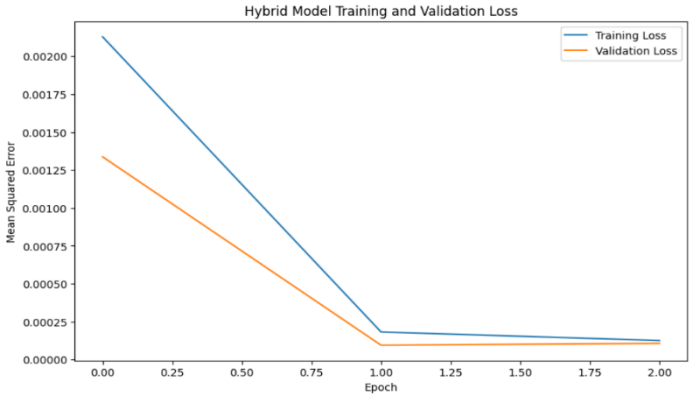

Figure 24: Learning curve of the Hybrid Model

The ideal situation is for the training and validation losses to converge to a low value. In this case, the training and validation losses converge to a low value after epoch 2. This means that the model is not overfitting the training data and can generalize to new data.

From the figure 25 the graph presents a 60-day stock price forecast using a hybrid model. On the initial day, the predicted stock price is higher than the actual, hinting at an anticipated rise in stock value. However, as days progress, the model forecasts a decline, suggesting expected stock market fluctuations.

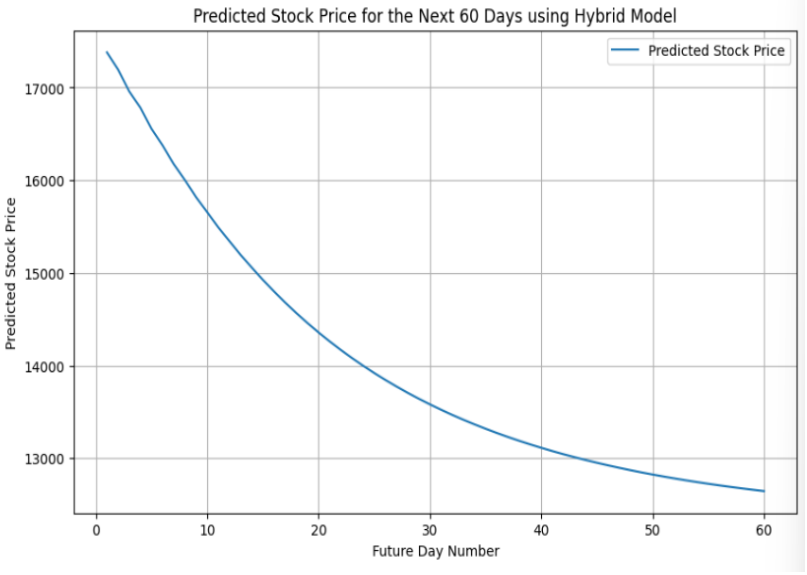

Figure 25:Prediction using Hybrid for 60 days.

Short-term predictions appear more volatile due to immediate market influencers like news events or fresh economic data. In contrast, longer term projections seem more stable. Although the graph of the hybrid model helps determine the direction of stock prices, its forecasts should be evaluated considering the model's constraints and the inherent unpredictability of the stock market.

After plotting the graph, we shifted towards the hyperparameter tuning of the Hybrid model and this are the results we got from the RandomSearchCV search for hyperparameter tuning.

```
best_model = tuner.get_best_models(num_models=1)[0]
best_hyperparameters = tuner.get_best_hyperparameters(num_trials=1)[0]

print(best_hyperparameters.values)
```

```
{'lstm_units_1': 50, 'gru_units': 30, 'dense_units': 30}
```

Figure 26: Best hyperparameters for Hybrid Model

After we settled on specific settings to improve its predictions after using RandomSearchCV from Keras Tuner to fine-tune our hybrid model. We chose 50 units for the LSTM layer to help the model understand long patterns in stock data. Then, we added a GRU layer with 30 units to focus on shorter, recent trends. Finally, a dense layer with 30 units wraps everything up to give the final forecast. These settings let our model look at both long-term trends and short-term changes to predict stock prices more accurately.

5.4 Comparative analysis of Models

5.4.1 Comparison of Models Architecture

The architecture of a model is, after all, the backbone upon which its capabilities and limitations rest. By comparing the architectures, we get a sense of the complexity and specificity of each model, shedding light on their inherent strengths and potential vulnerabilities.

The LSTM, GRU, and Hybrid models all fall under the umbrella of recurrent neural networks and have distinct structural and functional elements. The LSTM is particularly adept at capturing long-term dependencies in data sequences, while the GRU offers efficiency by simplifying some of the LSTM's mechanisms. On the other hand, the Hybrid model is an ambitious amalgamation, attempting to harness the strengths of LSTM and GRU, supplemented by the innovative Time2Vec layer.

Readers can quickly grasp each model's hierarchical layers, unit configurations, and unique features by presenting a comparative table of their architectures which is given below (Table 1). This architectural insight sets the stage for a more informed interpretation of their subsequent performance metrics. Essentially, before we evaluate "how well" each model performed, it is crucial to understand "how" each model works. This foundational knowledge ensures the subsequent analysis is contextualized, allowing for a more holistic understanding of the results.

Aspect	LSTM	GRU	Hybrid
Feature scaling	MinMaxScaler	MinMaxScaler	MinMaxScaler
Model Architecture	• 2 LSTM layers • 2 Dense layers	• 2 GRU layers • 2 Dense layers	• Time2Vec layer • LSTM layer • GRU layer • 2 Dense layers
Optimisation	• Adam optimizer • Mean Squared Error (loss function) • Early Stopping	• Adam optimizer • Mean Squared Error (loss function) • Early Stopping	• Adam optimizer • Mean Squared Error (loss function) • Early Stopping
Hyperparameter Tuning	• RandomSearch technique	• RandomSearch technique	• RandomSearch technique
Prediction focus	• Long term dependencies	• Sequences	• Combines long-term and short-term predictions

Table 1: Comparison of model's architecture

When looking at the design of each model, the LSTM stands out because it is good at understanding long-term trends in data. This makes it great for predicting stock prices, as past data can be significant. While other models have their strengths, the LSTM seems best for our stock data based on its design.

5.4.2 Comparison of Models performance metrices

To provide a clear perspective on the performance of each model, we have tabulated the key metrics—Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Squared Error (MSE)—for the LSTM, GRU, and Hybrid Models.

This table offers a direct comparison, aiding in quickly identifying each model's strengths and areas of improvement. Please refer to the table below for a detailed breakdown.

Model	MAE	RMSE	MSE
LSTM	535.55	611.23	373601.83
GRU	392.42	441.55	194969.95
Hybrid Model	435.81	523.47	274018.97

Table 2: Comparison of performance metrics of models

Upon examining the performance metrics of the LSTM, GRU, and Hybrid models, it is evident that each offers distinct advantages in stock price forecasting. The LSTM model has an MAE of 535.55, suggesting a moderate deviation from actual prices. In contrast, the GRU model is the most accurate, with an MAE of 392.42, followed closely by the Hybrid model at 435.81. The RMSE and MSE values further reinforce the GRU's superior performance. While the LSTM captures general trends, the GRU offers a more precise forecast. The Hybrid model, combining elements of both, strikes a balance between them. In essence, for the most accurate predictions, the GRU model seems the optimal choice, according to this metrices. But there are several other factors also influence the real-world performance of this models.

The performance superiority of the GRU model, even when compared to the Hybrid model, can be attributed to several inherent characteristics, chief among them being its architecture and the dynamics of its training process.

Firstly, the GRU's streamlined architecture offers a compelling advantage. Unlike the LSTM, which uses a trio of gates—input, forget, and output—the GRU simplifies this structure by employing only two gates, namely the reset and update gates. This architectural simplification reduces the number of parameters, potentially making the model less prone to overfitting and allowing for faster training. In contexts where data may not have long-term dependencies or where computational efficiency is paramount, this lightweight design can be particularly effective, leading to better generalization of unseen data.

Secondly, the training dynamics are pivotal in shaping a model's performance. Parameters like learning rate, batch size, and other hyperparameters can significantly influence how well a model adapts to the data. For this dataset, it is plausible that the specific dynamics chosen were incredibly harmonious with the GRU's architecture. This synergy can result in a more robust and optimised model, enabling the GRU to outperform even the Hybrid model, which attempts to merge the strengths of both LSTM and GRU but might also inherit their complexities.

The GRU's architectural simplicity and favourable training dynamics could have positioned it as the most adept model for this specific forecasting task, highlighting the importance of model choice and training strategy in neural network applications.

5.4.3 Comparison of Models Forecasting with real world corresponding data

After assessing the models' architecture and performance indicators, we further evaluated their applicability by contrasting their forecasts with actual stock prices over 60 days. This real-world comparison gives us a priceless viewpoint and enables us to evaluate the models' performance in a real-world market environment.

The following graph compares the real stock values with the predictions of three different models:

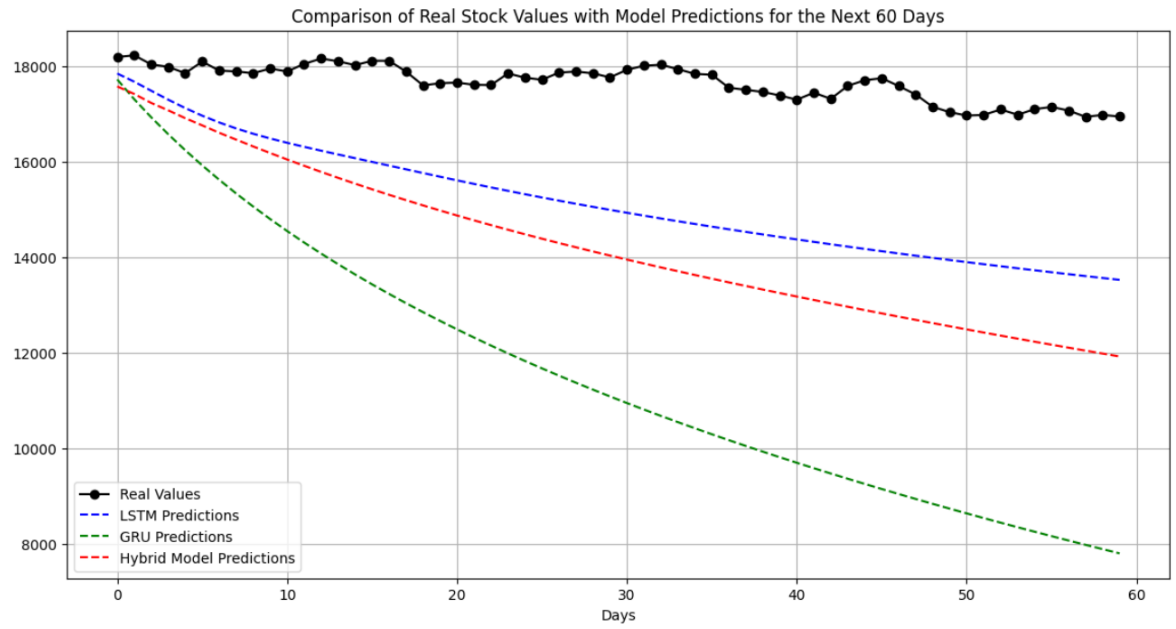

Figure 27: Comparison of real stock values with model predictions

The graph above (Figure 27) shows that LSTM's predictions with blue line closely match actual stock prices, demonstrating its superior prediction ability in this situation. The red line representing the Hybrid model predictions also closely trails the real stock values, although it doesn't achieve the precision of the GRU model. Meanwhile, the GRU model, represented by green line, tends to deviate more noticeably from the actual stock values, indicating room for improvement in its forecasting capabilities.

Another important finding is the growing discrepancy between model forecasts and actual stock values over time. This points to a need for more stock price forecasting that is inherent.

When we look further into the future, projections are inevitably more prone to inaccuracy because of the numerous unknown factors affecting stock prices.

We have visualised the prediction errors for each model to better comprehend the differences between the stock value predictions made by our algorithms and the actual stock values.

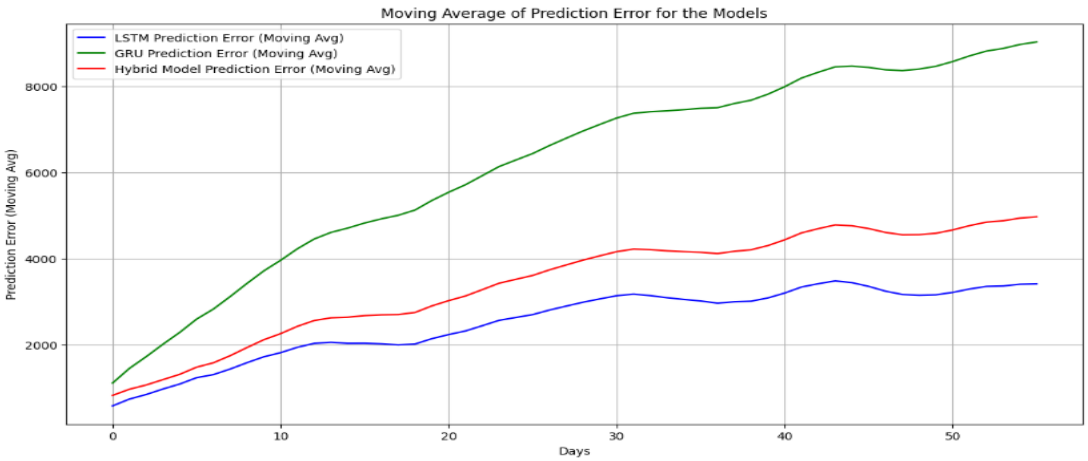

Figure 28: Moving average prediction error of models.

Figure 28 shows the moving average of prediction error for the LSTM, GRU and Hybrid Models. The moving average is calculated by averaging the prediction error over a time window. The LSTM model has the lowest moving average prediction error, followed by the Hybrid model, and the GRU model has the highest moving average. This means LSTM is the most accurate for predicting stock prices.

The graph also demonstrates that as the number of days rises, the moving average forecast error does too. This is because forecasting the stock price for the future is more challenging than doing it for the present.

Chapter 6: Conclusion and Future work

Deep learning, a subset of machine learning, has witnessed a meteoric rise in its application across diverse domains, from image recognition to natural language processing. Within finance, particularly stock market forecasting, the intricacies of deep learning models offer a potential avenue for more accurate predictions. This study embarked on an intricate journey to discern the capabilities of three such advanced models: the LSTM, GRU, and the Hybrid model, against the backdrop of the Nifty 50 stock index—a significant reflection of India's bustling equity market.

With its specialised architecture, the Long Short-Term Memory (LSTM) model is designed to capture and remember patterns over long sequences. It also makes it exceptionally equipped to understand and predict broader market trends that evolve over extended periods. The LSTM exhibited a Mean Absolute Error (MAE) of 535.55 and a Root Mean Square Error (RMSE) of 611.23 in the study. These metrics suggest that the model's forecasts, on average, deviate from the actual stock prices by these amounts, with the RMSE giving more weight to more significant errors. While these figures are commendable, they indicate areas where the model might be further refined for enhanced accuracy.

Contrastingly, the Gated Recurrent Unit (GRU) model, a more recent evolution in RNNs, simplifies the LSTM's architecture. By reducing the number of gates, the GRU aims for computational efficiency without compromising the model's ability to capture patterns. This was evident in its performance metrics, with an MAE of 392.42 and an RMSE of 441.55. These figures underscore the GRU's precision and hint at its potential to be a front-runner in stock market forecasting.

The Hybrid model represents an ambitious endeavour to merge the best of both worlds. By integrating the long-term memory prowess of the LSTM with the efficiency of the GRU, this model aspired to offer comprehensive forecasting capabilities. Its performance metrics, with an MAE of 435.81 and an RMSE of 523.47, position it between the LSTM and GRU in terms of accuracy.

However, an intriguing revelation emerged when the study ventured beyond metrics. When forecasts over a 60-day horizon were visually scrutinised, the LSTM's predictions displayed a closer congruence with real-world stock prices. This divergence between metric-based evaluations and visual analysis accentuates the multifaceted nature of stock market forecasting and stresses the need for a holistic evaluation mechanism.

One persistent challenge, evident across all models, was the growing disparity between predicted and actual stock prices as the forecasting horizon extended. This accentuates the inherent unpredictability of stock markets, which are influenced by many factors, including global events, economic policies, and industry-specific news.

The realm of stock market forecasting remains a labyrinth of complexities, even armed with the sophisticated tools of deep learning. While models like the LSTM, GRU, and Hybrid provide illuminating insights and enhanced prediction capabilities, they are not silver bullets. The myriad factors influencing stock markets ensure that while predictions can be informed and refined, they can never be infallible. This study is a testament to the strides made in predictive modelling and the journey ahead. As we look to the future, integrating even more diverse datasets and capturing a more comprehensive array of influencing factors might be the next frontier in enhancing forecasting accuracy.

Looking to the future, several avenues beckon exploration to enhance the forecasting prowess of these models:

1. Incorporate Additional Data Streams:
    1. News Sentiment Analysis: Real-time sentiment analysis of news articles, tweets, or financial reports could provide nuanced insights, enabling the models to anticipate stock price movement better.
    2. Macroeconomic Indicators: Integrating broader economic indicators, like GDP growth or unemployment rates, might offer a more comprehensive forecasting framework.
    3. Technical Indicators**:** Metrics from technical analysis can further refine prediction capabilities.
2. Advanced Model Exploration:
    1. Attention Mechanisms & Transformer Architectures: Deploying models like BERT or GPT-3 or integrating attention mechanisms might uncover previously unseen patterns in stock data.
3. Model Interpretability: Tools like LIME (Local Interpretable Model -agnostic Explanations) or SHAP (Shapely Additive Explanations) could demystify the decision-making processes of these models, enhancing trust in their predictions.
4. Expanding Horizons:

1. Ensemble Methods & Real-time Forecasting Platforms: Combining predictions or developing platforms for real-time forecasting might offer more robust predictions.
2. Diversifying Financial Instruments & Geographical Expansion: Testing these models on other financial instruments or global stock indices could assess their universality.

5. Risk and Behavioural Insights: Integrating risk management strategies or delving into behavioural finance could offer a more rounded forecasting approach, encompassing both quantitative and psychological facets of stock trading.

In conclusion, while the foray into deep learning for stock market forecasting through this study has unveiled promising strategies, it also highlights the journey yet to be traversed. The stock market remains a complex tapestry of intertwining threads, and while deep learning offers a magnifying glass to discern patterns, the picture is vast and evolving. The road ahead beckons with challenges and opportunities waiting to be explored.

Bibliography

Adamjee, U. (2023) "Exploratory data analysis on stock market data - MLearning.ai - medium," *Medium*, 5 March. Available at: https://medium.com/mlearning-ai/exploratory-data-analysis-on-stock-market-data-5d99fbdf3b04#:~:text=Exploratory%20Data%20Analysis%20(EDA)%20is,S%26P%20500%20dataset%20using%20Python.

Bhattacharya, A. (2022) "Deep Hybrid Learning — a fusion of conventional ML with state of the art DL," *Medium*, 6 August. Available at: https://towardsdatascience.com/deep-hybrid-learning-a-fusion-of-conventional-ml-with-state-of-the-art-dl-cb43887fe14.

Chung, J. (2014) *Empirical evaluation of gated recurrent neural networks on sequence modeling*. Available at: https://arxiv.org/abs/1412.3555.

Dobilas, S. (2022) "GRU Recurrent Neural Networks — A Smart Way to Predict Sequences in Python," *Medium*, 26 February. Available at: https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6.

Fathali, Z., Kodia, Z. and Said, L.B. (2022) "Stock market prediction of NIFTY 50 Index Applying machine learning techniques," *Applied Artificial Intelligence*, 36(1). Available at: https://doi.org/10.1080/08839514.2022.2111134.

Gao, Y., Wang, R. and Zhou, E.-M. (2021) "Stock prediction based on optimized LSTM and GRU models," *Scientific Programming*, 2021, pp. 1–8. Available at: https://doi.org/10.1155/2021/4055281.

Godfried, I. (2023) "Advances in Deep Learning for Time Series Forecasting and Classification: Winter 2023 Edition," *Medium*, 28 February. Available at: https://towardsdatascience.com/advances-in-deep-learning-for-time-series-forecasting-and-classification-winter-2023-edition-6617c203c1d1.

Guven, M. and Uysal, F. (2023) “Time Series Forecasting performance of the novel deep learning algorithms on stack Overflow website data,” *Applied Sciences*, 13(8), p. 4781. Available at: https://doi.org/10.3390/app13084781.

Karim, Md.E., Foysal, Md.F.A. and Das, S. (2022) “Stock price prediction using Bi-LSTM and GRU-Based Hybrid Deep Learning approach,” in *Lecture notes in networks and systems*, pp. 701–711. Available at: https://doi.org/10.1007/978-981-19-3148-2_60.

Kazemi, S.M. (2019) *Time2VEc: Learning a vector representation of time*. Available at: https://arxiv.org/abs/1907.05321.

Kogilavani, S.V. *et al.* (2022) “Investigation of applying machine learning and hyperparameter tuned deep learning approaches for arrhythmia detection in ECG images,” *Computational and Mathematical Methods in Medicine*, 2022, pp. 1–12. Available at: https://doi.org/10.1155/2022/8571970.

Kumar, A. (2022) *Correlation Concepts, Matrix & Heatmap using Seaborn - Data Analytics*. Available at: https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/.

Lianne & Justin @ Just into Data (2021) “3 Steps to Forecast Time Series: LSTM with TensorFlow Keras | Towards Data Science,” *Medium*, 13 December. Available at: https://towardsdatascience.com/3-steps-to-forecast-time-series-lstm-with-tensorflow-keras-ba88c6f05237.

Long Short-Term Memory Networks (LSTM)- simply explained! | Data Basecamp (2022). Available at: https://databasecamp.de/en/ml/lstms.

LSTM multi-step hyperparameter optimization with Keras Tuner | peterspython.com (no date). Available at: https://www.peterspython.com/en/blog/lstm-multi-step-hyperparameter-optimization-with-keras-tuner.

Moghar, A. and Hamiche, M. (2020) "Stock market prediction using LSTM Recurrent Neural network," *Procedia Computer Science*, 170, pp. 1168–1173. Available at: https://doi.org/10.1016/j.procs.2020.03.049.

Muhammad, A.U., Li, X. and Feng, J. (2019) "Using LSTM GRU and hybrid models for streamflow forecasting," in *Springer eBooks*, pp. 510–524. Available at: https://doi.org/10.1007/978-3-030-32388-2_44.

Parmezan, A.R.S., De Souza, V. and Batista, G.E. a. P.A. (2019) "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model," *Information Sciences*, 484, pp. 302–337. Available at: https://doi.org/10.1016/j.ins.2019.01.076.

Patra, G.R. and Mohanty, M.N. (2022) "An LSTM-GRU based hybrid framework for secured stock price prediction," *Journal of Statistics and Management Systems*, 25(6), pp. 1491–1499. Available at: https://doi.org/10.1080/09720510.2022.2092263.

Pramod and Pm, M.S. (2021) "Stock price prediction using LSTM," *ResearchGate* [Preprint]. Available at: https://www.researchgate.net/publication/348390803_Stock_Price_Prediction_Using_LSTM.

Rouf, N. *et al.* (2021) "Stock market Prediction Using Machine Learning Techniques: A decade survey on methodologies, recent developments, and future directions," *Electronics*, 10(21), p. 2717. Available at: https://doi.org/10.3390/electronics10212717.

Seo, Y., Kim, S.-W. and Singh, V.P. (2015) "Estimating spatial precipitation using regression kriging and Artificial Neural Network Residual kriging (RKNNRK) hybrid approach," *Water Resources Management*, 29(7), pp. 2189–2204. Available at: https://doi.org/10.1007/s11269-015-0935-9.

Shi, J. (2022) *Time Series Forecasting (TSF) using various deep learning models*. Available at: https://arxiv.org/abs/2204.11115.

Spencer, A. (2022) "5 minute EDA: Correlation heatmap - 5 minute EDA - medium," *Medium*, 12 May. Available at: https://medium.com/5-minute-eda/5-minute-eda-correlation-heatmap-b57bbb7bae14.

Stock Market Analysis and Prediction for Nifty50 using LSTM Deep Learning Approach (2022). Available at: https://ieeexplore.ieee.org/document/9754148.

Suryanarayanan, R. (2022) "Time series forecast using deep learning - geek culture - medium," *Medium*, 6 January. Available at: https://medium.com/geekculture/time-series-forecast-using-deep-learning-adef5753ec85.

Ullah, I. *et al.* (2021) "Software defined network enabled Fog-to-Things hybrid deep learning driven cyber threat detection system," *Security and Communication Networks*, 2021, pp. 1–15. Available at: https://doi.org/10.1155/2021/6136670.

Van Houdt, G., Mosquera, C. and Nápoles, G. (2020) "A review on the long short-term memory model," *Artificial Intelligence Review*, 53(8), pp. 5929–5955. Available at: https://doi.org/10.1007/s10462-020-09838-1.

Wang, C. *et al.* (2020) "The real-time big data processing method based on LSTM or GRU for the smart job shop production process," *Journal of Algorithms & Computational Technology*, 14, p. 174830262096239. Available at: https://doi.org/10.1177/1748302620962390.

Yan, H. and Ouyang, H. (2017) "Financial Time Series Prediction based on deep learning," *Wireless Personal Communications*, 102(2), pp. 683–700. Available at: https://doi.org/10.1007/s11277-017-5086-2.